



slington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS6P05NI Project

Final Report

40% Individual Coursework

Submission: Final Submission

Academic Semester: Spring Semester 2025

Credit: 30 Credit Year Long Module

Student Name: Rhythm Paudel

London Met ID: 22067983

College ID: NP01CP4A220282

Assignment Due Date: Wednesday, April 30, 2025

Assignment Submission Date: Wednesday, April 30, 2025

Submitted To: Aadesh Tandukar / Oshriya Manandhar

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

22067983 Rhythm Paudel.docx

 Islington College, Nepal

Document Details

Submission ID

trn:oid:::3618:93494550

Submission Date

Apr 29, 2025, 4:57 PM GMT+5:45

Download Date

Apr 29, 2025, 4:59 PM GMT+5:45

File Name

22067983 Rhythm Paudel.docx

File Size

58.1 KB

68 Pages

8,633 Words

47,986 Characters

3% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

-  **27 Not Cited or Quoted 3%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 1%  Internet sources
- 0%  Publications
- 2%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Abstract

This report documents various stages of the TourNepal application creation process.¹The TourNepal application was developed to solve problems usually faced by tourists in Nepal. The application will facilitate users with an authentic review system and live updates via notifications, eliminating the problem of fake review farms. This documentation involves a detailed description of all the steps involved in development, from the planning phase to execution. The report mainly consists of four segments: introduction, background, development, and testing. The application was developed under Agile methodology and Scrum framework. All the stages in the report are also presented accordingly. Additionally, several issues like legal, social, or ethical that could be faced after the deployment of the application have also been discussed in the documentation. The application provides safe and informed information; however, the potential of the application is enormous, for which the future works also have been discussed in the report.

Acknowledgement

This project would not have been completed without a proper guidance and support of many respected teachers and individuals.

First and foremost, I would like to thank Islington College for providing me with an excellent academic opportunity, where I could showcase my knowledge in solving a problem faced by many in the current scenario. Islington College has been a great help as many resources were provided as needed.

I would like to extend my gratitude to my internal supervisor Mr. Aadesh Tandukar, whose timely help, reviews and criticism helped me improve the quality of the application. He helped me to steer the direction of the application in the right way. Equally, I am thankful to my external supervisor Ms. Oshriya Manandhar. She helped me to develop my application by sharing her real-world experience on mobile application development.

I also acknowledge the contribution made by my peers, and my seniors who helped me tackle several issues. Without the support of mentioned people, and institution, the project would not have been completed as expected.

Table of Contents

Chapter 1. Introduction.....	1
1.1. Current Scenario	2
1.2. Problem Statement.....	4
1.3. Project as a solution	6
1.4. Aims and objectives	8
1.4.1. Aims.....	8
1.4.2. Objectives	8
1.5. Structure of the report	9
1.5.1. Background	9
1.5.2. Development.....	9
1.5.3. Testing and analysis	9
1.5.4. Conclusion.....	9
Chapter 2. Background	10
2.1. Targeted Users.....	10
2.2. End User.....	10
2.2.1. Client Information.....	10
2.3. Understanding the solution	11
2.3.1. Technologies.....	11
2.4. Similar Projects	13
2.4.1. Trip Advisor	13
2.4.2. Culture Trip.....	14
2.4.3. Visit Nepal Official Guide.....	15
2.5. Comparison of similar projects	16
2.6. Comparison analysis	18
Chapter 3. Development	19
3.1. Considered Methodologies	19
3.2. Selected Methodology	20

3.3.	Survey results.....	22
3.3.1.	Post-Survey result	22
3.4.	Requirement analysis	23
3.5.	Design.....	24
3.5.1.	Logo.....	24
3.5.2.	System architecture design	25
3.5.3.	UI Design.....	27
3.5.4.	Logical data model	30
3.5.5.	State diagram	31
3.5.6.	Sequence diagram	32
3.5.7.	Activity diagram	37
3.6.	Implementation.....	42
3.6.1.	Sprint 1 (01/11/2024-14/11/2024)	42
3.6.2.	Sprint 2 (15/11/2024-28/11/2024)	42
3.6.3.	Sprint 3 (29/11/2024-12/12/2024)	42
3.6.4.	Sprint 4 (13/12/2024-26/12/2024)	46
3.6.5.	Sprint 5 (27/12/2024-09/01/2025)	49
3.6.6.	Sprint 6 (10/01/2025-23/01/2025)	50
3.6.7.	Sprint 7 (24/01/2025-06/02/2025)	53
3.6.8.	Sprint 8 (07/02/2025-20/02/2025)	56
3.6.9.	Sprint 9 (21/02/2025-06//03/2025).....	58
3.6.10.	Sprint 10 (07/03/2025-20/03/2025)	59
3.6.11.	Sprint 11 (21/03/2024-03/04/2025)	61
Chapter 4.	Testing and Analysis	62
4.1.	Test plan.....	62
4.1.1.	Unit testing test plan	62
4.1.2.	API/ Integration test plans	64
4.1.3.	System test plans	68
4.2.	Unit testing.....	71
4.2.1.	Server start testing (UT-01)	71
4.2.2.	MongoDB connection test (UT-02)	73
4.2.3.	Login unit testing (UT-03)	75

4.2.4.	User operations testing (UT-04).....	76
4.2.5.	Registration unit testing (UT-05)	78
4.2.6.	Emergency contact operations unit testing (UT-06)	79
4.2.7.	Location retrieval testing (UT-07)	80
4.2.8.	Request for notification access testing (UT-08)	82
4.2.9.	Token storage in mobile device testing (UT-09)	84
4.2.10.	Gemini place description generation testing (UT-10)	87
4.2.11.	Gemini invalid place description generation testing (UT-11).....	88
4.2.12.	Password hashing testing (UT-12).....	89
4.3.	API/Integration testing	90
4.3.1.	Authentication testing.....	90
4.3.2.	Emergency contact fetch testing.....	103
4.3.3.	Map/Location testing	104
4.3.4.	Notification testing	122
4.3.5.	User details fetch/manipulation testing	131
4.3.6.	Admin tasks testing.....	156
4.4.	System testing.....	183
4.4.1.	System authentication testing	183
4.4.2.	Location tasks testing	206
4.4.3.	Profile and reviews testing	227
4.4.4.	Admin tasks testing.....	233
4.5.	Critical analysis.....	243
Chapter 5.	Conclusion	244
5.1.	Legal, Social and Ethical Issues	244
5.1.1.	Legal issues.....	244
5.1.2.	Social Issues	244
5.1.3.	Ethical Issues	245
5.2.	ADVANTAGES.....	246
5.3.	Limitations	247
5.4.	Future work	248
Chapter 6.	References	249

Chapter 7. Appendix	254
7.1. Appendix A: Phases of methodology	254
7.1.1. Initiation phase	254
7.1.2. Sprint planning phase	254
7.1.3. Execution phase	254
7.1.4. Review Phase	254
7.1.5. Sprint Restrospective Phase	255
7.1.6. Final Testing and Deployment Phase	255
7.2. Appendix B: Selected methodology.....	256
7.3. Appendix C: Considered methodology	257
7.3.1. Waterfall Methodology	257
7.3.2. Kanban Methodology	258
7.3.3. Spiral Methodology	259
7.4. Appendix D: Requirement analysis.....	260
7.5. Appendix E: Logical data model	262
7.6. Appendix F: Client description and requirements	263
7.7. Appendix G: Post survey result.....	265
7.7.1. Questions and responses.....	266
7.8. Appendix G: Legal, Social and Ethical Issues	272
7.8.1. Legal issues.....	272
7.8.2. Social issues	272
7.8.3. Ethical issues	273
7.9. Appendix H: Designs	274
7.9.1. Gantt Chart	274
7.9.2. Work Breakdown Structure (WBS)	277
7.9.3. Use Case.....	278
7.9.4. Data-flow diagram	279
7.9.5. System flowchart	280
7.9.6. Sequence diagram	281
7.9.7. Activity diagram	290
7.9.8. Wireframes.....	293

7.9.9.	User Intefrace (UI)	297
7.10. Appendix	I: Code logic	301
7.10.1.	Server setup logic	301
7.10.2.	Persistent login logic	302
7.10.3.	Token storage	303
7.10.4.	Review addition logic	304
7.10.5.	Notification sending logic	307
7.10.6.	Nearby place code logic	309
7.10.7.	Description generation	311
7.11. Appendix	I: Software Requirement Specification (SRS)	313
7.11.1.	Introduction	313
7.11.2.	Intended Audiences and Reading Suggestions	313
7.11.3.	Project Scope	314
7.11.4.	System features	315
7.11.5.	User class and characteristics	316
7.11.6.	Operating Environment	316
7.11.7.	Design and Implementation Constraint	317
7.11.8.	Assumption and dependencies	317
7.11.9.	Functional Requirements	319
7.11.10.	External Interfaces Requirements	323
7.11.11.	Non-functional Requirements	324
7.11.12.	Safety Requirements	325
7.12. Appendix	J: Screenshot of the system	326
7.13. Appendix	K: Client Approval Letter	333
7.14. Appendix	L: Development links	334
7.13.1.	Trello board Link	334
7.13.2.	GitHub repository link	334

Table of Figures

Figure 1: Growth of tourists in Nepal.	2
Figure 2: Trip Advisor.....	13
Figure 3: Culture Trip	14
Figure 4: Visit Nepal Official Guide	15
Figure 5: Agile Methodology Diagram.....	19
Figure 6: Scrum Methodology Framework.....	20
Figure 7: TourNepal Logo	24
Figure 8: System Architecture Design	26
Figure 9: Search screen (UI)	27
Figure 10: Notification screen (UI)	28
Figure 11: Emergency Contacts (UI).....	29
Figure 12: Logical Data Model	30
Figure 13: State diagram	31
Figure 14: Explore popular destinations (Sequence Diagram)	32
Figure 15: Emergency contacts (Sequence Diagram).....	33
Figure 16: Account deletion request (Sequence Diagram)	34
Figure 17: Admin document approval (Sequence Diagram).....	35
Figure 18: Reviews (Sequence diagram)	36
Figure 19: Explore popular destinations (Activity Diagram).....	37
Figure 20: Emergency contacts (Activity Diagram).....	38
Figure 21: Account deletion request (Activity Diagram).....	39
Figure 22: Destination description (Activity Diagram).....	40
Figure 23: Posting a review (Activity Diagram).....	41
Figure 24: Backend project Structure Setup	43
Figure 25: Frontend project Structure Setup	44
Figure 26: Google map Api testing	45
Figure 27: MongoDB cluster creation	46
Figure 28: Basic backend configuration setup	47
Figure 29: Login controller code	47
Figure 30: Register controller code.....	48

Figure 31: Google map Api development	49
Figure 32: Firebase setup	49
Figure 33: Log in screen	50
Figure 34: Sign up screen.....	51
Figure 35: Destination listing static	52
Figure 36: Google map Api integration (Backend).....	53
Figure 37: Google map API integration (Backend 02)	53
Figure 38: Google map Api integration frontend	54
Figure 39: Destination search (Google maps Api).....	55
Figure 40: Verified user posting a review.....	56
Figure 41: Unverified user posting a review	57
Figure 42: User edit/verification via admin panel.....	58
Figure 43: Review filtering admin panel.....	59
Figure 44: Notification implementation	60
Figure 45: Notification mobile with test message	60
Figure 46: Emergency contact list mobile UI	61
Figure 47: Server not started	71
Figure 48: Import error	72
Figure 49: Corrected import.....	72
Figure 50: MongoDB connection test.	74
Figure 51: MongoDB connection test 2	74
Figure 52: Login unit testing	75
Figure 53: Test cases of user actions	76
Figure 54: Unit test results for user actions	77
Figure 55: Register unit testing	78
Figure 56: Emergency contacts unit testing.....	79
Figure 57: Requesting user location	80
Figure 58: Installing react-native-get-location package	81
Figure 59: Getting user current location	81
Figure 60: User current location	81
Figure 61: Android permission for post notification.....	82

Figure 62: Requesting notification access after login	82
Figure 63: Logging in (notification test).....	83
Figure 64: Notification request prompt.....	83
Figure 65: Installing react-native-keychain	84
Figure 66: Storing/retrieving tokens code snippet	85
Figure 67: Storing token after login.....	85
Figure 68: Application initialization check for tokens	85
Figure 69: Logging in (token storage test)	86
Figure 70: Displaying token in console	86
Figure 71: Gemini description test	87
Figure 72: Gemini invalid place description test	88
Figure 73: Bcrypt installation	89
Figure 74: Password comparison unit test.....	89
Figure 75: Valid credentials login test	91
Figure 76: Incomplete body registration	93
Figure 77: Registration false value handling.....	95
Figure 78: User creation API testing	97
Figure 79: User creation in mongoDB	97
Figure 80: Logging in for refresh token.....	99
Figure 81: Requesting new access token	99
Figure 82: Refresh token span	100
Figure 83: Encrypted refresh token retrieval.....	101
Figure 84: Requesting access token with expired refresh token	101
Figure 85: Token request without refresh token.....	102
Figure 86: Getting contacts list	103
Figure 87: Google map API data response.....	105
Figure 88: Requesting destinations without location	106
Figure 89: Response without destination type	107
Figure 90: Corrected code snippet for destination type check.....	108
Figure 91: Place fetching without destination type	109
Figure 92: Location list API fetch	110

Figure 93: Empty list for name based search	111
Figure 94: Wrong data type of radius.....	112
Figure 95: Corrected code for place search by name.....	113
Figure 96: Search by place name result	114
Figure 97: Retrieval of base64 image	116
Figure 98: Base64 image conversion	116
Figure 99: Gemini API description generation	117
Figure 100: Gemini API description generation without name	118
Figure 101: Reviews of Patan Darbar square.....	120
Figure 102: Review fetching	120
Figure 103: Review fetching for place without reviews	121
Figure 104: Requesting reviews without location	122
Figure 105: Retrieving access token for admin	123
Figure 106: Pasting access token in bearer token for notification	124
Figure 107: Sending notification to a mobile phone(API)	124
Figure 108: API testing notification on phone	125
Figure 109: Notification history updated in mongoDB	125
Figure 110: Sending notification to multiple mobile phones(API).....	126
Figure 111: Sending notifications without tokens	127
Figure 112: User status before notification token update.....	128
Figure 113: Logging in for token update	129
Figure 114: Notification token update after login.....	129
Figure 115: Invalid notification token handling	130
Figure 116: Retrieving user details from email.....	131
Figure 117: Setting expiration token to 10 seconds(user).....	133
Figure 118: Retrieving user detail with expired access token	133
Figure 119: User deletion status before deletion request	134
Figure 120: Deletion request bearer token	135
Figure 121: Deletion request change via API	135
Figure 122: User deletion status after deletion request	136
Figure 123: Updating user deletion without bearer token(1).....	137

Figure 124: Updating user deletion without bearer token(2).....	137
Figure 125: Fetching user notifications	138
Figure 126: Logging out from application	140
Figure 127: Confirming logout	141
Figure 128: Error when restarting the application	141
Figure 129: Corrected logic for conditional rendering	142
Figure 130: Searching labim mall to post a review	144
Figure 131: Updating user status from pending to verified in database	144
Figure 132: Trying to post a review after user status was verified	145
Figure 133: Corrected user details fetching during application initialization	146
Figure 134: Trying to add a review after verifying user status	148
Figure 135: Server error during login	149
Figure 136: Adding a review(API)	151
Figure 137: Editing an existing comment(Unsuccessful).....	151
Figure 138: Comment before updating(API).....	152
Figure 139: Updating comment(API)	152
Figure 140: Comment update failed(API)	153
Figure 141: Incorrect code snippet for updating comment	153
Figure 142: Corrected code for updating comment	154
Figure 143: Editing an existing comment(Unsuccessful).....	154
Figure 144: Updating comment after correction	155
Figure 145: Updated comment success in database.....	155
Figure 146: Admin login testing	156
Figure 147: Refresh token response set on cookie	157
Figure 148: Requesting a new access token(Admin)	158
Figure 149: User log in (response with non-encrypted refresh token).....	159
Figure 150: Admin log in for setting up cookie	159
Figure 151: Replacing admin refresh token with users refresh token.....	160
Figure 152: Requesting admin access token from users refresh token.....	160
Figure 153: Editing non-existing emergency contact.....	161
Figure 154: Editing emergency contacts without values.....	162

Figure 155: TourNepalService number before update	163
Figure 156: Updating TourNepalService contact number	164
Figure 157: TourNepalService updated number	164
Figure 158: Contact list before deletion	165
Figure 159: Deleting existing emergency contact(API).....	166
Figure 160: Database after deletion of emergency contact	166
Figure 161: Before addition on new contact in database.....	167
Figure 162: Adding a contact(API)	168
Figure 163: Updated contact list after contact addition in database	168
Figure 164: Getting all user without access token	169
Figure 165: Retrieving all users in database	170
Figure 166: Creating new user via admin API	172
Figure 167: Addition of new user via admin API in database.....	172
Figure 168: Editing user via admin API.....	174
Figure 169: Updated user in database after updating via admin API	174
Figure 170: User deleting via admin API	175
Figure 171: Admin reviews retrieval(API)	176
Figure 172: Existing review before deletion(API).....	177
Figure 173: Deleting a review via admin API	178
Figure 174: After review deletion via admin API	178
Figure 175: Admin persistent login(unsuccesful)	179
Figure 176: Admin panel login	179
Figure 177: Logged in admin panel	180
Figure 178: Login page despite cookies being saved.....	180
Figure 179: Admin persistent login	181
Figure 180: Using cookie parser in the backend server	181
Figure 181: Admin panel homepage	182
Figure 182: Registration navigation	184
Figure 183: Invalid email format error	185
Figure 184: Registering without and with name field	187
Figure 185: Registering without and with email and birth date field	188

Figure 186: Continuing with different password.....	190
Figure 187: Registering fields(1).....	192
Figure 188: Registering user(2)	193
Figure 189: Registering user(3)	194
Figure 190: Registering user(4)	195
Figure 191: User registration in database.....	195
Figure 192: Registering with an existing email	196
Figure 193: Registering with existing email testing(1)	197
Figure 194: Registering with existing email testing(2)	197
Figure 195: Registration failed with same email	198
Figure 196: Bypassing document during registration	199
Figure 197: Logging in with valid credentials.....	200
Figure 198: Incorrect password error message	201
Figure 199: Unregistered user trying to login.....	202
Figure 200: Login without any fields	203
Figure 201: Logging in (test for persistent login)	204
Figure 202: Persistent login	205
Figure 203: Default destinations search	207
Figure 204: Navigating to search section	209
Figure 205: Restaurants filtering with radius	210
Figure 206: Destinations fetching by radius.....	212
Figure 207: Refreshing radius slider to update restaurants.....	214
Figure 208: Search any place feature.....	216
Figure 209: Search filter for fetched location	217
Figure 210: Navigating via main search screen.....	219
Figure 211: Navigating to location via description page.....	220
Figure 212: Description of restaurants.....	222
Figure 213: Description of less popular restaurants.	224
Figure 214: Description for destinations	226
Figure 215: Reuploading documents.....	228
Figure 216: Verified user leaving a review	230

Figure 217: Unverified user trying to post a review	232
Figure 218: Admin logging in	233
Figure 219: Admin dashboard.....	234
Figure 220: Navigating to add user section	235
Figure 221: Creating a new user	235
Figure 222: Addition of new user	236
Figure 223: Profile view of added user on Mobile app.....	236
Figure 224: User status before verification	237
Figure 225: Verifying user(1)	238
Figure 226: Verifying user(2)	238
Figure 227: Verified user status	239
Figure 228: Notification casting	240
Figure 229: Casting notification to all users.....	240
Figure 230: Casting notification to specific user	241
Figure 231: Notification list on different accounts	242
Figure 232: Waterfall methodology	257
Figure 233: Kanban methodology.....	258
Figure 234: Spiral methodology.....	259
Figure 235: Overall application experience (survey response).....	266
Figure 236: Most relevant features (survey response)	266
Figure 237: Search feature (survey response)	267
Figure 238: Application accuracy (survey response)	267
Figure 239: User-friendliness (survey response).....	268
Figure 240: Similar application comparison (survey response).....	268
Figure 241: Application usefulness (survey response)	269
Figure 242: Potentiality of application (survey response).....	269
Figure 243: Registration process (survey response)	270
Figure 244: Search functionality effectiveness (survey response).....	270
Figure 245: Review system effectiveness (survey response).....	271
Figure 246: Additional comments (survey response).....	271
Figure 247: Gantt chart (1)	274

Figure 248: Gantt chart (2)	275
Figure 249: Gantt chart (3)	275
Figure 250: Gantt chart (4)	276
Figure 251: Work breakdown structure (WBS)	277
Figure 252: Use case diagram.....	278
Figure 253: Dataflow diagram.....	279
Figure 254: System flowchart	280
Figure 255: Login process sequence diagram.....	281
Figure 256: Registration process sequence diagram	282
Figure 257: User registration (code snippet)	283
Figure 258: User registration by admin (code snippet)	284
Figure 259: Authentication flow sequence diagram	286
Figure 260: Authentication flow code snippet	287
Figure 261: Location fetching code snippet	288
Figure 262: Login activity diagram	290
Figure 263: Registration activity diagram	291
Figure 264: Authentication flow activity diagram.....	292
Figure 265: Login and registration wireframe	293
Figure 266: Mobile screen wireframes.....	294
Figure 267: Emergency contacts wireframe (admin)	295
Figure 268: Notifications screen wireframe (admin)	295
Figure 269: Review management screen wireframe (admin)	296
Figure 270: User management screen wireframe (admin)	296
Figure 271: Registration, login and home screen	297
Figure 272: Profile screen.....	298
Figure 273: Users page (admin panel UI).....	298
Figure 274: User detail (admin panel UI).....	299
Figure 275: Emergency contacts (admin panel UI)	299
Figure 276: Review management (admin panel UI)	300
Figure 277: Notification screen (admin panel UI)	300
Figure 278: Server setup (code snippet)	301

Figure 279: Persistent user login	302
Figure 280: Token Storage code snippet	303
Figure 281: Review box disable code snippet	304
Figure 282: Adding a comment (frontend)	304
Figure 283: User verification before posting a review code snippet	305
Figure 284: Adding review code snippet.....	306
Figure 285: Notification handling admin panel code snippet	307
Figure 286: Notification handling code snippet.....	308
Figure 287: Nearby place fetching backend code snippet.....	309
Figure 288: Nearby place search mobile app code snippet.....	310
Figure 289: Selective search code snippet.....	310
Figure 290: Description generation mobile app code snippet.....	311
Figure 291: Description generation Gemini code snippet.....	312
Figure 292: Login screen (mobile)	326
Figure 293: Signup screen (mobile).....	327
Figure 294: Main screens (mobile)	328
Figure 295: Home screen (admin)	329
Figure 296: Users Management page (admin)	329
Figure 297: Edit user page (admin)	330
Figure 298: Add user page (admin)	330
Figure 299: Emergency contacts page (admin).....	331
Figure 300: Review management page (admin).....	331
Figure 301: Notification handling page (admin)	332

Table of Tables

Table 1: Similar Project comparison.	17
Table 2: Unit test plan	63
Table 3: API/ Integration test plan	67
Table 4: System test plans	70
Table 5: Server starting test (Unsuccessful)	71
Table 6: Server start test (successful).....	73
Table 7: MongoDB connection test.	73
Table 8: Login unit testing	75
Table 9: User operations testing	76
Table 10: Registration unit testing.....	78
Table 11: Emergency contact operation unit testing	79
Table 12: User location retrieval testing.....	80
Table 13: Request for notification access testing.....	82
Table 14: Token storage in mobile device testing	84
Table 15: Gemini place description generation testing	87
Table 16: Gemini invalid place description generation testing	88
Table 17: Password hashing testing	89
Table 18: Empty body login test.....	90
Table 19: Successful login testing.....	91
Table 20: Incomplete body registration	92
Table 21: Registration false value handling	94
Table 22: Registration completion testing	96
Table 23: Requesting access token with refresh token.....	98
Table 24: Requesting access token with expired refresh token.....	100
Table 25: Token request without refresh token	102
Table 26: Retrieving emergency contacts	103
Table 27: Google map API test	104
Table 28: Requesting destinations without location	106
Table 29: Missing destination type for places(unsuccesful)	107
Table 30: Missing destination type for place request(successful).....	108

Table 31: Location fetch testing	109
Table 32: Search by name testing(unsuccesful)	110
Table 33: Search by name testing (successful)	113
Table 34: Base64 Conversion testing	115
Table 35: Description generation for place	117
Table 36: Description generation without name	118
Table 37: Existing review fetching.....	119
Table 38: Review fetching for place without reviews	121
Table 39: Requesting reviews without location	122
Table 40: Testing for push notification (API).....	123
Table 41: Multiple device notification casting	126
Table 42: Sending notification without device tokens	127
Table 43: Updating notification token	128
Table 44: Improper notification token handling	130
Table 45: User detail fetching(API)	131
Table 46: User details fetching with expired token.....	132
Table 47: User deletion request update	134
Table 48: Deletion request without providing access token	136
Table 49: Retrieving user notification history	138
Table 50: User logging out of the application(unsuccesful).....	139
Table 51: User logging out of the application	142
Table 52: Adding a review after user status is verified(unsuccesful).....	143
Table 53: Adding a review after user status is verified	147
Table 54: Starting the application when server is not active	149
Table 55: Adding a comment(API)	150
Table 56: Admin login testing(API)	156
Table 57: Admin requesting new access token	157
Table 58: Requesting access token from mobile application user refresh token	158
Table 59: Editing non-existent emergency contacts	161
Table 60: Editing emergency contacts without values	162
Table 61: Editing emergency contacts	163

Table 62: Deleting emergency contact number	165
Table 63: Adding a new contact(API)	167
Table 64: Fetching all user without access token	169
Table 65: Fetching all user with valid access token	170
Table 66: Adding new user with valid access token	171
Table 67: Admin editing user(API)	173
Table 68: Deleting a user via admin API	175
Table 69: Admin retrieving review of all places	176
Table 70: Admin deleting a review(API)	177
Table 71: Invalid email format during registration	183
Table 72: Empty fields during registration test	186
Table 73: Registering without non-matching password	189
Table 74: Registration of user testing	191
Table 75: Bypassing document registration testing	199
Table 76: Logging in with valid credentials test.....	200
Table 77: Logging in with incorrect password	201
Table 78: Unregistered user logging in	202
Table 79: Logging in without credentials	203
Table 80: Persistent login in mobile application	204
Table 81: Viewing restaurants in default settings.....	206
Table 82: Refreshing radius slider to update restaurants	208
Table 83: Destinations fetching by radius	211
Table 84: Refreshing radius slider to update restaurants	213
Table 85: Search any place	215
Table 86: Search filter for fetched location.....	217
Table 87:Navigating to google maps via application	218
Table 88: Description of restaurants	221
Table 89: Description for less popular restaurants.....	223
Table 90: Description for destinations	225
Table 91: Reuploading documents	227
Table 92: Verified user leaving a review	229

Table 93: Unverified user leaving a review	231
Table 94: Admin login test.....	233
Table 95: Admin adding user	234
Table 96: Verifying user status	237
Table 97: User class (admin)	316
Table 98: User class (user)	316
Table 99: Functional requirements.....	322
Table 100: Non-functional Requirements.....	325
Table 101: Safety Requirements.....	325
Table 102: Client approval letter	333

Chapter 1. Introduction

Nepal is a developing country and is highly dependent on tourism for its economic growth (Gautam, n.d.). However, the facilities provided to tourists are sometimes not as expected. The nature and scenarios alone cannot satisfy the tourists travelling in places. The expectations of tourists are sometimes not matched with actual destinations on reaching there. This leaves a bad impression on some places, and affects the overall review of the country.

The problem is not new; however, a solution to this problem has not been effectively provided. The application is aimed to start by solving a problem at a time, which will eventually help the tourist to know their destinations better. This will give a head start to tourists before travelling.

TourNepal, a mobile application, is focused on enhancing the experience of the tourist during their stay in Nepal. The application is aimed to deliver a proper experience during the visit of the places, and restaurants listed on the application. This application has the potential for a large scale of functionalities which will help tourists as a great assistant. The application was made by taking the problem context which needs an improvement in Nepal.

1.1. Current Scenario

Nepal is widely popular for its beautiful culture and various breathtaking natural scenarios among tourists. The average number of tourists in Nepal has increased from 230,000 in 2020 to 600,000 in 2022. Nepal had approximately 1.2 million tourists in the last year 2024 which shows the increase of 13 percent than than of year 2023 (Nepal Republic Media Limited, 2025). According to the World Travel and Tourism Council, the tourism sector of Nepal single-handedly contributes to 6.2% of the total GDP of Nepal (CollegesNp, 2023).

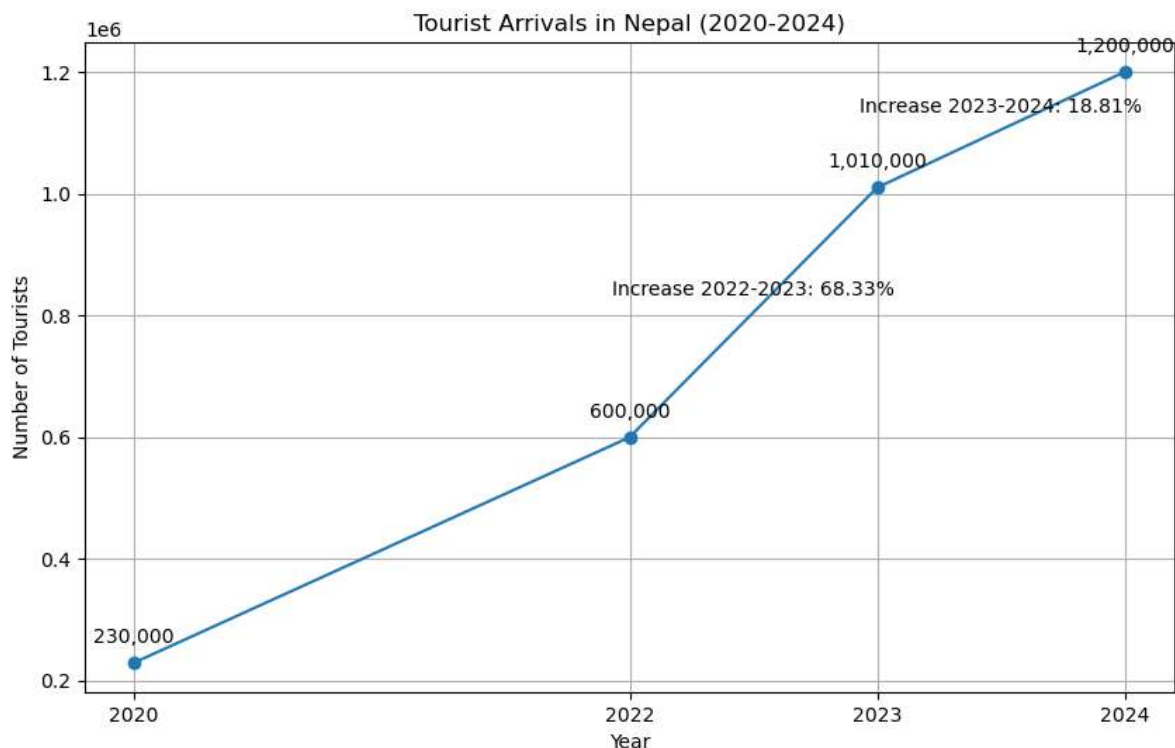


Figure 1: Growth of tourists in Nepal.

(The graph does not reflect accurate data for 2021, 2023, and 2024.)

The more incoming tourists result in more turnover of money in various sectors, among which restaurants and tourist attractions come at the top of the list. Some tourists might also be in some kind of emergency without any help, which will leave them stranded in a foreign

country. This shows the improper management of tourism sector in Nepal in current scenario.

1.2. Problem Statement

Tourism acts as a primary source of exchange in culture, as it is acknowledged by everyone travelling in the destined country. The popularity of a country also determines its trade relations with others, which directly is associated with how good the tourists think the country is with visitors. Along with the increasing number of tourists, the scams and pricing (also known as tourist pricing) will start to skyrocket which will leave the tourists in the dilemma of Nepal being a price-worthy country.

Tourists in Nepal always have a hard time adapting to the environment. This problem is taken as an opportunity by the locals where they try to scam a tourist by a tremendous amount'. In the fiscal year 2018/19 tourist police of Nepal recorded a total of 1252 cases filed by tourists related to fraud. (Dhungana, 2019). Many tourists also find navigating the emergency services very confusing as the accessibility of the emergency contacts is not always clear. (Karki, 2023).

Tourists also fall into the trap of non-authorized people, pretending to be part of the official guide community, who end up asking for money double or thrice of original asks after the completion of a short guide in famous destinations. Sometimes in a place where no entrance fee or very minimal entrance fee is charged, scammers or so-called official guide members tend to charge more amount due to a lack of awareness among the tourists. (Mosaic Adventure, 2024).

Mostly before travelling to a place, tourists rely on the reviews of Google Maps. However, reviews are easily bought, and the review section can be filled up with thousands of reviews in a matter of seconds with the help of various bots. All these mentioned problems, could be countered with a single solution of a proper and trusted reviews.

Due to the lack of a trusted platform for reviews, not just the tourists but also the business owners have to face several problems. Some restaurants or attractions are focused on a certain type of group only, which leaves the other type of group not suitable for it. Some places are meant to attract tourists only, where local people might not have the best experience, which results in a poor review of the place. The same review is being watched by everyone, including tourists, which reduces the traffic of the place. Tourists also find it

hard to navigate to emergency contacts, as Nepal holds a variety of numbers for various purposes in case of an emergency. Emergency contacts' addresses are meant to be used in an emergency, and the hassle of finding the correct number during an emergency leaves the tourists panicking more.

1.3. Project as a solution

This project has been designed to overcome such problems faced by tourists and focuses on contributing to the tourism sector for its development. After the completion of the project, tourists in Nepal will minimize the chance of getting scammed on famous destinations.

This application has primarily focused on making a solution for the tourist in many different ways. The application takes the current problem faced by most tourist and has features a tourist would love to have. The application not only focuses on current user but makes it useful for new users too.

The following are the key features of the application:

- A radius based search feature will be provided to user, where users can have their choice more personalized and accesible.
- Users can leave a review on the places they visited which will make other users in the application to view the accurate review based on tourist experience.
- Reviews are monitored by the admin, making sure the reviews are authentic and no foul language are used.
- Description of any popular places are generated by AI ensuring the information is latest.
- Admin can create user for mobile application user, if user faces some trouble creating it via mobile application.
- User can request for profile deletion, which needs to be approved by the admin.
- Admin can make changes to the user information, if anything needs to be corrected.
- Users can access the list of emergency contact numbers list from the application, making it convenient for them to contact any national bodies without the need of hassling through the internet to find numbers.

- Users will also be notified through the app if any natural disaster occurs or events are taking place which will be sent via admin panel.
- The application will also ask the user for their visa or passport to ensure no fake reviews will be posted or permitted in the application. (The visa needs an approval from the admin).
- Emergency contacts are updated if any changes are made through the admin panel, along with sending notifications to a user for messages related to a certain user, like document verified, or a bulk message to alert or notify all the users.

1.4. Aims and objectives

1.4.1. Aims

This project aims to deliver a application for tourists traveling in Nepal that will enhance their travel experience with authenticity and navigations along with minimizing risks in certain aspects.

1.4.2. Objectives

- To develop a android application for tourist, that enhances the experience of the user during their travel.
- To ensure the elimination for fake reviews reviews, by adding a verification during posting of review by user.
- To help the tourist make an informed decision before travelling to places, with proper authenticated review.
- To facilitate tourist with emergency bodies number without having to hussle in case of emergency.
- To enable admin for monitoring users, during their stay in Nepal, and keep track of tourists in case any incidents caused by the tourists.
- To notify the users with real-time updates for any occuring events, festivals, or any disasters to keep them updated and ensure to help users make plans accordingly.
- To provide the user with proper feedback from actual tourist experience in the place making the feedback reliable and relatable.
- To provide user with real-time place suggestions for restaurants or destinations according to the preference of the user.
- To implement a documentation verification system for ensuring the trust and reliability of the application.

1.5. Structure of the report

1.5.1. Background

This portion of the document will discuss about the types of the end users and expected target audience to use this application. Furthermore, the information on client will also be discussed thoroughly. The discussion on the topic of this app 'TourNepal' not being the only the type of application in tourism sector will take place along with the comparison of similar applications.

1.5.2. Development

In the development section, a detailed walkthrough on the development process will be explained. It will outline the entire structure of the development process, along with the used methodologies and considered methodologies. This section will cover step-by-step breakdown of development phases according to the methodology used.

1.5.3. Testing and analysis

This section of the documentation will cover the reliability of the system, by performing various types of test necessary. Several tests and their results will be documented and in case of any failures, the correction measures will be discussed. Along with testing the analysis of the system will be performed as well.

1.5.4. Conclusion

Overall development and progress of the project will be summarized in the conclusion. This section will also reflect the learning, challenges faced, and objectives achieved throughout the development of the project. Additionally, the issues that may be faced socially, ethically, or legally will also be discussed in this portion. The advantages and limitations will also be briefed followed by the possible issues that could be faced. Furthermore, the future work of application will be discussed, where the potential of the application will be explained and discovered.

Chapter 2. Background

2.1. Targeted Users

The TourNepal application is developed to assist the tourists coming to Nepal for travelling purposes. The application is targeted for any type of tourists, especially for the first-time traveler because the application offers a variety of services, like emergency assistance, real-time place suggestion, and genuine reviews in places. The application is strictly for tourists, so any local users are not able to create an account, even if they create one, it will be deleted. The age requirement for the users is that they have been at least 16 years old. Among the tourists, only the tourists with a valid passport and visa are permitted to use the application.

2.2. End User

2.2.1. Client Information

Client Name: Triple R Tours and Travel

Client Description: Tripe R, a travels and tour company, recently coming into market for exposure and working on travel sector of Nepal has been chosen to be the client of this application.

[\(Client requirements and details can be found in Appendix F\)](#)

2.3. Understanding the solution

The application TourNepal is designed to be a solution for the major problem of the tourism sector in Nepal. The application is meant to be used by tourists visiting Nepal, as the application offers a variety of services that help them with their travel in Nepal. TourNepal will provide a seamless experience for tourists that has a very user-friendly experience, keeping all user actions within a few clicks away.

The development of the application is primarily divided into two different segments, frontend and backend.

- 1) **Frontend Development:** The user-interface for the mobile application is developed with React-Native, a java-script based framework popular in development of cross-platform application. Application developed in react-native ensures the single codebase can be used for every platform supported.
For admin panel, React.js is used which is used to handle all the user related activities of the application.
- 2) **Backend Development:** The server-side development is built using Node.js which is run-time environment of javascript. Express.js framework is used for handling Api related task (business logic).

2.3.1. Technologies

The application is integrated with third party services to ensure the smooth functionality of the applications.

MongoDB is used for database of the application. It will hold all the necessary details of the objects, or documents required for the application to work. MongoDB is noSQL database which offers flexibility in data while storing.

Similarly, google maps API is used for fetching details of the places. For location based searching , it helps in providing the data in according to preference of the user like restaurants or destinations.

For storing the documents, a third-party cloud is used known as cloudinary. The secured url is saved in the mongoDB of user documents. Using cloudinary reduces the working load on the mongoDB for saving user documents, making it an efficient solution.

2.4. Similar Projects

Many projects have been developed which were built to serve similar purpose. This section will cover similar projects that were developed taking the same aim of this project along with why this application fulfils the missing requirements and features that could make the tourist travel more convenient. A comparison will be made between the similar projects and TourNepal application as well.

2.4.1. Trip Advisor

Trip Advisor, a popular platform for tour guidance was launched to solve problems regarding trip-planning. Trip Advisor also claims to be the world largest trip accommodating platform. This application provides destinations based on location. The application is also used to plan trip and choose restaurants for dining purpose. Trip Advisor provides a feature of booking a table at restaurants if compatible too. Additionally, this application has more than a billion reviews on various places aggregated (Tripadvisor LLC, 2023). This application has more than 900 million registered users, expected to reach a billion users by the year 2025 (Condor Limited, 2025).

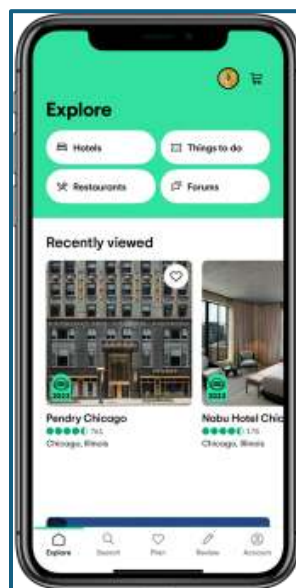


Figure 2: Trip Advisor

(Tripadvisor LLC, 2025).

2.4.2. Culture Trip

Culture Trip was founded in 2011 for solving similar purpose of travel problem. This application lets you plan the trip, suggest places for shopping, wellness and other famous destinations. Culture trip has members worldwide which provide the suggestion of the places according to the local experts and members around the world. Culture trip is also famous for simple design and limited features. Furthermore, this application lets the user to plan their trip and save it.

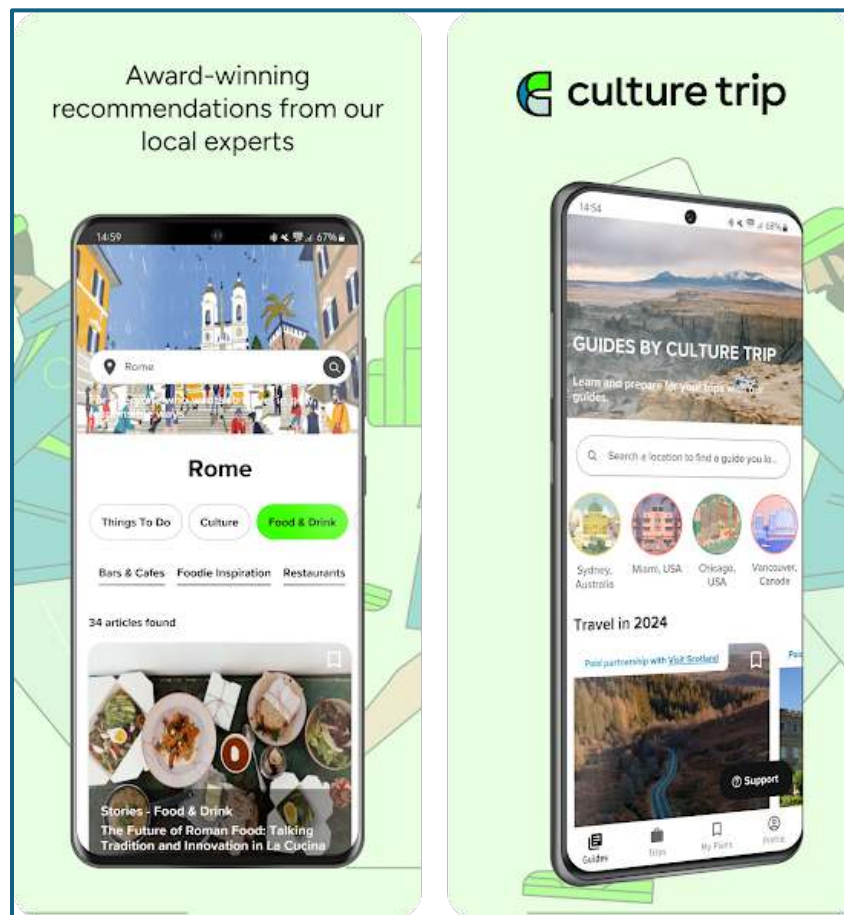


Figure 3: Culture Trip

(Google Play, 2025)

2.4.3. Visit Nepal Official Guide

This mobile application is developed by Ministry of Culture, Tourism and Civil Aviation, an official government body of Nepal. This application was developed to promote the places of Nepal. Through this application, an user can view major attractions or tourist destinations of Nepal. This application also provides an interface to book flights international or domestic by adding a hyperlink to the aviation website. For booking flights it is redirected to the website in the browser. The application provides various destinations all over the world, but these destinations are static and does not change according to the location of the user.

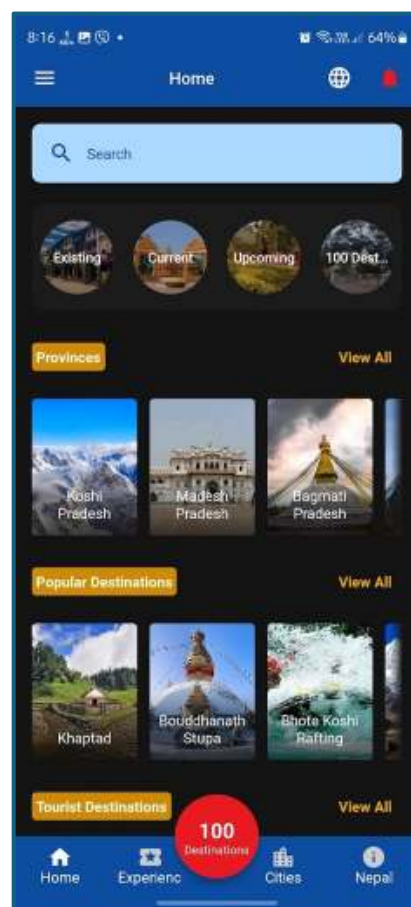


Figure 4: Visit Nepal Official Guide

(Google Play, 2024)

2.5. Comparison of similar projects

The above discussed project more or less serves the same purpose. However, some features were missing in those application that a user may frequently want to use.

For an example, TripAdvisor is a global travel guide, however it does not work in context of Nepal the same way it does for other countries. The reviews section also might not be accurate as business might post a fake review for the promotion. Likewise, the application, culture trip does not provide the user real-time location-based suggestion. Same with Visit Nepal Official Guide, the application only provides the user with static destinations without a review.

For inspiration of the application the emergency contacts was taken from Visit Nepal Official guide, and location based search was taken as inspiration from Culture Trip and Trip advisor. However, for location based search, in TourNepal, the search are more personalized where user can make a radius based search unlike in other place recommendation applications. However, verification and alert notifications are not found in any other similar projects making those a unique feature of the application.

Feature	Tripadvisor	CultureTrip	Visit Nepal Official Guide	Proposed Solution (Tour Nepal)
User Registration (Visa, passport upload)	✗	✗	✗	✓
Login and Authentication	✓	✓	✓	✓
Explore Popular Destinations	✓	✓	✓	✓
Nearby Places Integration	✓	✓	✗	✓
Verified User Ratings and Feedback	✗	✗	✗	✓
Emergency Contact Information	✗	✗	✓	✓
Admin Validation for documents	✗	✗	✗	✓
Profile Deletion Request	✓	✗	✗	✓
Notifications for alerts and events	✗	✗	✗	✓

Table 1: Similar Project comparison.

2.6. Comparison analysis

In contrast, though all the similar project aims to serve a similar project, most of the applications are lacking the primary features, like verified reviews, or emergency assistance. Similarly, culture trip and trip advisor are not specialized for Nepal, and Visit Nepal official guide has many missing features like location-based recommendation. TourNepal offers real-time notification services for events, or disaster and other combined features from similar projects making it best suited application for Nepal's context than other applications.

Chapter 3. Development

3.1. Considered Methodologies

I) Waterfall methodology

In this methodology, the stages should be well planned as there will not be any kind of iterations or working on current/previous after moving on to the next stage.

II) Kanban Methodology

Kanban methodology is used for maintenance tasks or on-going project where tasks are smoother. Additionally, Kanban does not provide a time restriction constraint which is highly required for this project.

III) Spiral Methodology

Spiral Methodology is a combination of waterfall and prototyping. Though spiral methodology requires constant feedback, a ready-to-present prototype needs to prepare in each iterations making this methodology more complex for this project.

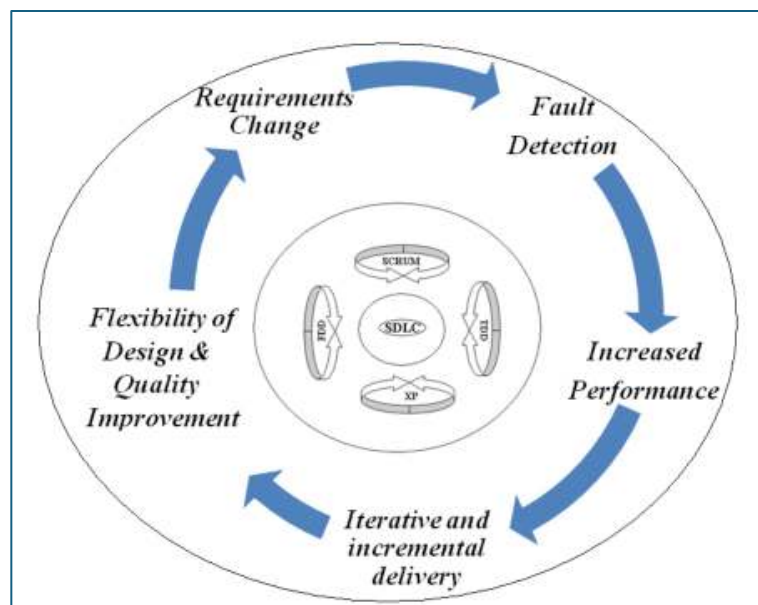


Figure 5: Agile Methodology Diagram

(Kumar & Bhatia, 2012)

[\(Details on each methodology can be found in Appendix C\)](#)

3.2. Selected Methodology

During research, several methodologies were considered to be implemented. Various methodologies like Waterfall, Spiral, or Kanban did not qualify with the nature of this project, which is why the mentioned methodologies were rejected. Among various methodologies in the industry, agile methodology was chosen for this project due to its flexibility. In agile, different frameworks are available to work with; among them, Scrum will be chosen as it seems to suit the best for this project. ([Phases of methodology can be found in Appendix B](#)).

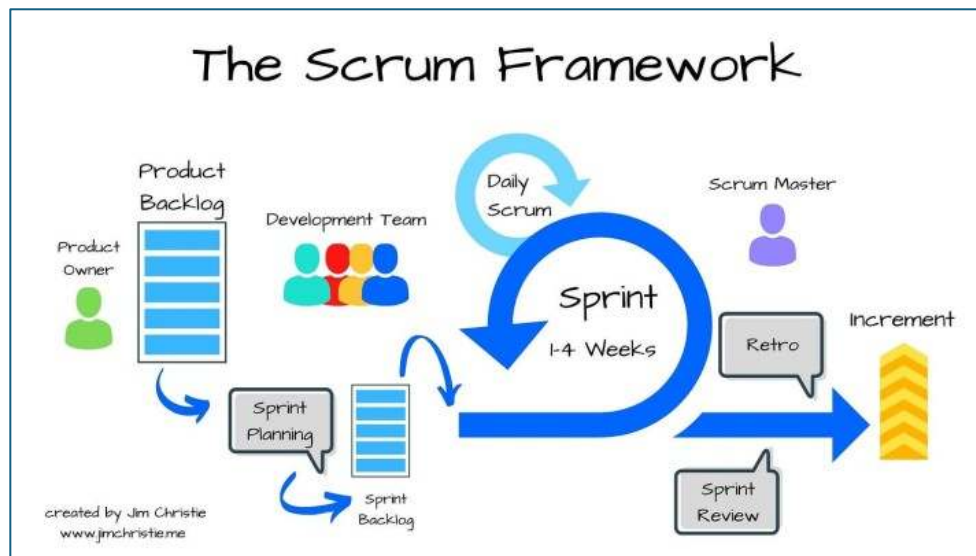


Figure 6: Scrum Methodology Framework

(Kouassi, 2024)

Scrum methodology is chosen as this will help to get a better understanding of features and a proper breakdown of the tasks. Multiple meetings in the development phase with supervisors will be held that will act as stakeholder meetings, which will then provide constant feedback and evaluation that needs to be addressed. ([Detailed discussion on scrum methodology can be found in Appendix A](#)).

The risk will be manageable as the project will be divided into multiple parts for development considering all the risks involved. Since Scrum also focuses on deliverables in each sprint, it will help to make sure the functions or needs are met for that sprint. Some other major reasons for selecting the scrum framework of agile methodologies for this project are listed below:

1. Iterative Development: Scrum provides a better chance of improving features allowing adjustment in projects.
2. Prioritization: Scrum ensures one focuses on the primary task of the sprint with the help of product backlog, making the project divided according to the prioritization of the steps.
3. Time management: Since the time of each sprint needs to be decided, scrum helps to ensure the tasks are completed within the timeframe, helping to maintain the discipline for finishing the project.

3.3. Survey results

The requirements for the application were thoroughly discussed with supervisors. Features were added and broken down with proper research on a weekly basis, with meetings with supervisors. Hence, no pre-survey was required. However, after the completion of the application, a post-survey was conducted, by taking the employees of the client for using the application.

3.3.1. Post-Survey result

Post-survey was conducted to eleven employees of the company Triple R Tours and Travel. All the respondents were asked about the application's usefulness, easiness, and potential of TourNepal. [\(Insights of Post-survey results can be found in Appendix G\)](#)

3.4. Requirement analysis

Requirement analysis is the process where the features, resources needed, and scopes are determined for having a clear perspective on the development of the application for providing the outcome as expected.

Below are the hardware and software requirements needed to build the project for smooth execution. [\(Details on requirement analysis can be found in Appendix D\)](#)

Hardware Requirements:

1. Desktop/Laptop
2. Android Phone/ Emulator
3. Processor: Intel core I5 9th Gen or more for Mac and Windows.
4. Proper Working Internet
5. 8GB of RAM or more.
6. 256GB of SSD or more.

These are recommended requirements to obtain best result, however higher the specification results in faster execution during the development.

Software Requirements:

1. Operating System: Windows OS or Mac OS
2. Text Editor/ IDE: Visual Studio Code
3. Postman: For API testing.
4. Git: Git is highly required for this project development for version controlling purposes to prevent any major errors that might occur when developing.
5. Frontend Development: JavaScript, React-Native.
6. React-Native is used for the development of this project, as it helps in cross-platform development, and is entirely based on JavaScript.
7. Backend Development: Express.js, Node.js
8. MongoDB: Since this project may require constant changes in features that might alter the data structure, the NoSQL-based database MongoDB is chosen.

3.5. Design

One of the most important aspects of the development of an application is designing. Designing maps out all the requirements of the system visually with the help of several diagrams. The flow of the process and application is well-planned in the designing phase, which makes it easier to develop an application with a proper breakdown of steps and features.

3.5.1. Logo



Figure 7: TourNepal Logo

3.5.2. System architecture design

System architecture defines a brief overview of the components on how they are inter-related with each other (Amazon Web Services, Inc., 2025). In TourNepal there are four main components that are interrelated to each-other.

External APIs are used for providing various cloud-based services like place fetching, image storing, or sending notifications as per the request of admin or mobile users. All of the logic is developed on the server side, where Node.js is used for handling business logic as well. Making the application based on a single language stack, i.e., JavaScript, was used for the development. Node.js is also known for its fast API response, which made this choice easier. With an active community and a rich ecosystem of Node.js integrated with various libraries like Express.js, Node.js was chosen for TourNepal.

For storing user documents, Cloudinary was used, which is a cloud-based media storage service. Additionally, using Cloudinary also decreases the workload on the server.

Likewise, the server communicates with the database (MongoDB) for altering or reading the data according to the need. MongoDB, being a NoSQL database, offers data flexibility unlike a relational database. The application database may need frequent changes in data structure, leaving the database for an easier modification when needed. SQL usually does not offer many features like array-based data storing, and if available, the complexities are not reduced. Using NoSQL makes sure that if there are any changes in business logic, it can be modified accordingly easily. MongoDB, providing an easier integration with Node.js, was used as a NoSQL database.

For any notification service, Firebase by Google was used. Firebase provides an easier configuration, with real-time notification service and cross-platform service.

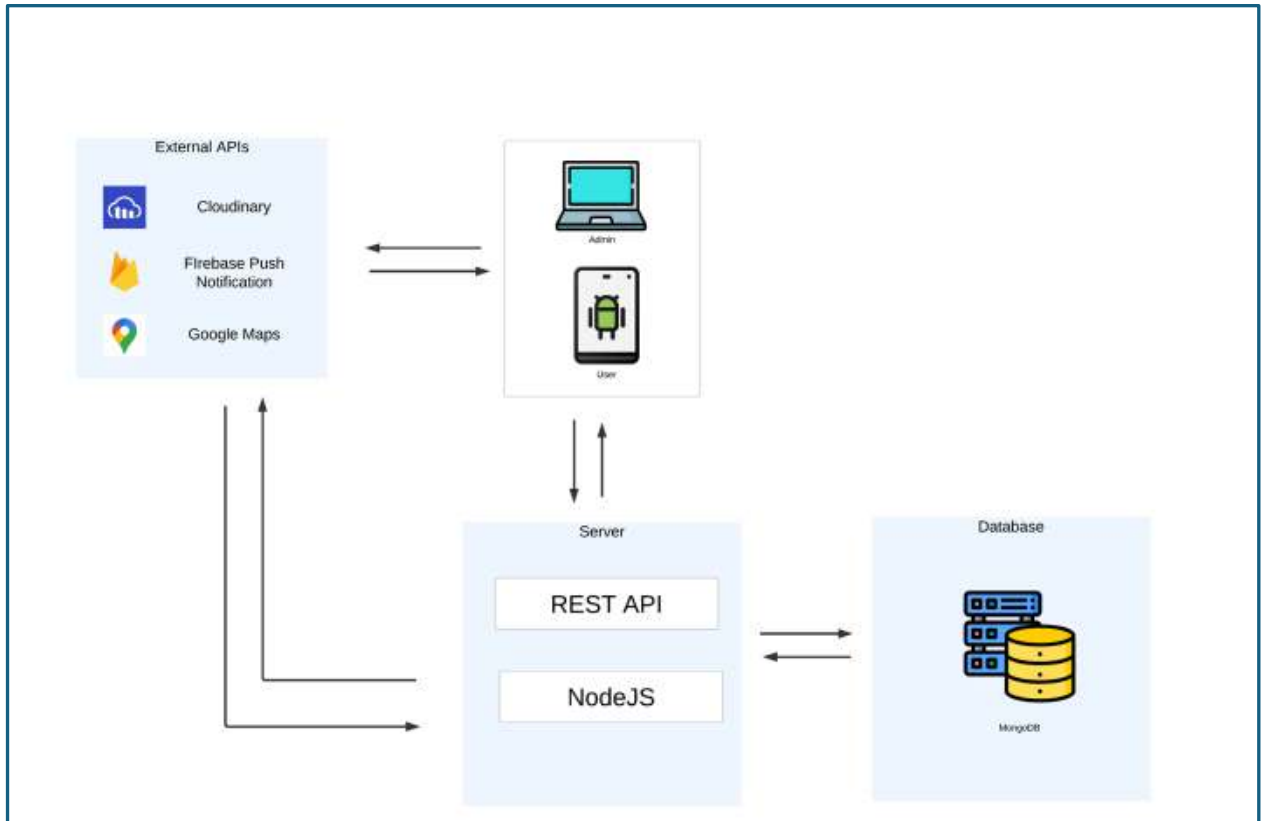


Figure 8: System Architecture Design

3.5.3. UI Design

I) Search screen

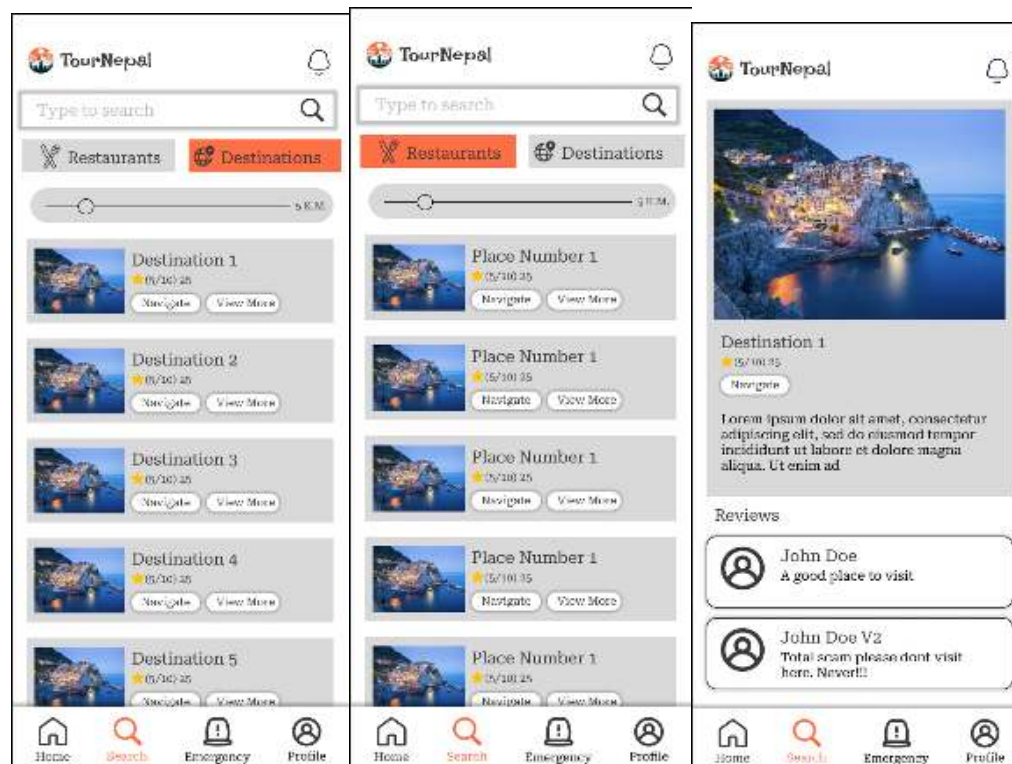


Figure 9: Search screen (UI)

II) Notifications screen

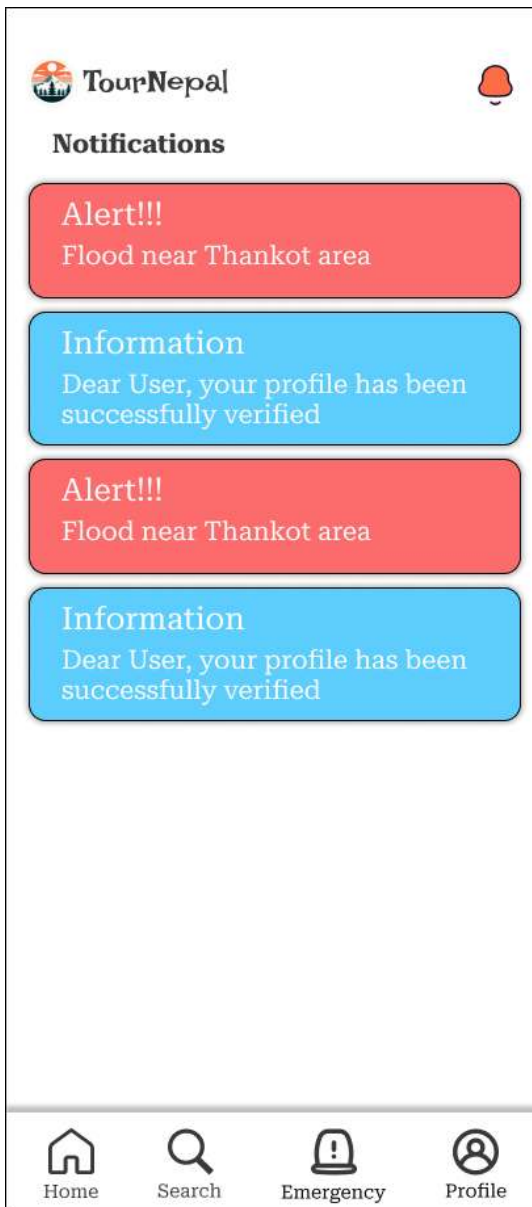


Figure 10: Notification screen (UI)

III) Emergency Contacts UI

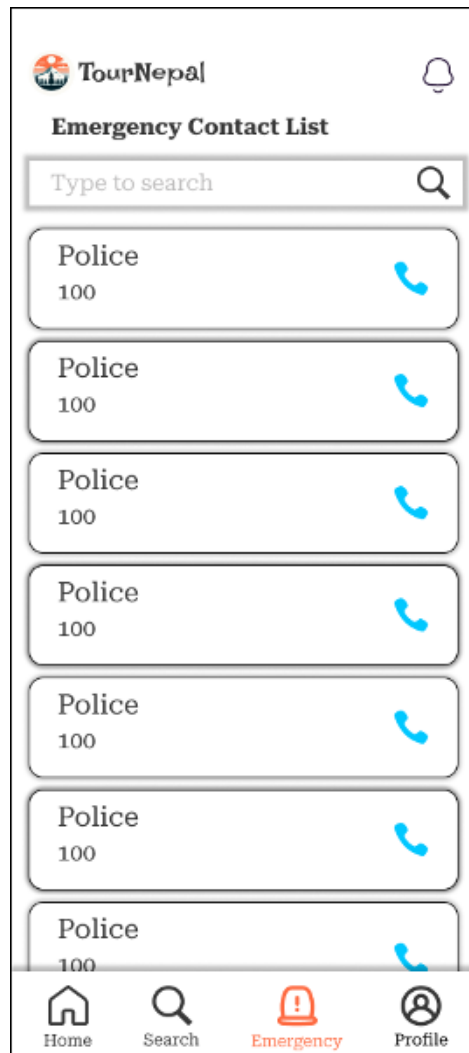


Figure 11: Emergency Contacts (UI)

3.5.4. Logical data model

Since MongoDB is NoSQL, an entity relationship diagram is not applicable due to its ever-changing schema. The following logical model represents the possible schema and overall structure of the database. ([Logical data model explanation can be found in Appendix E](#))

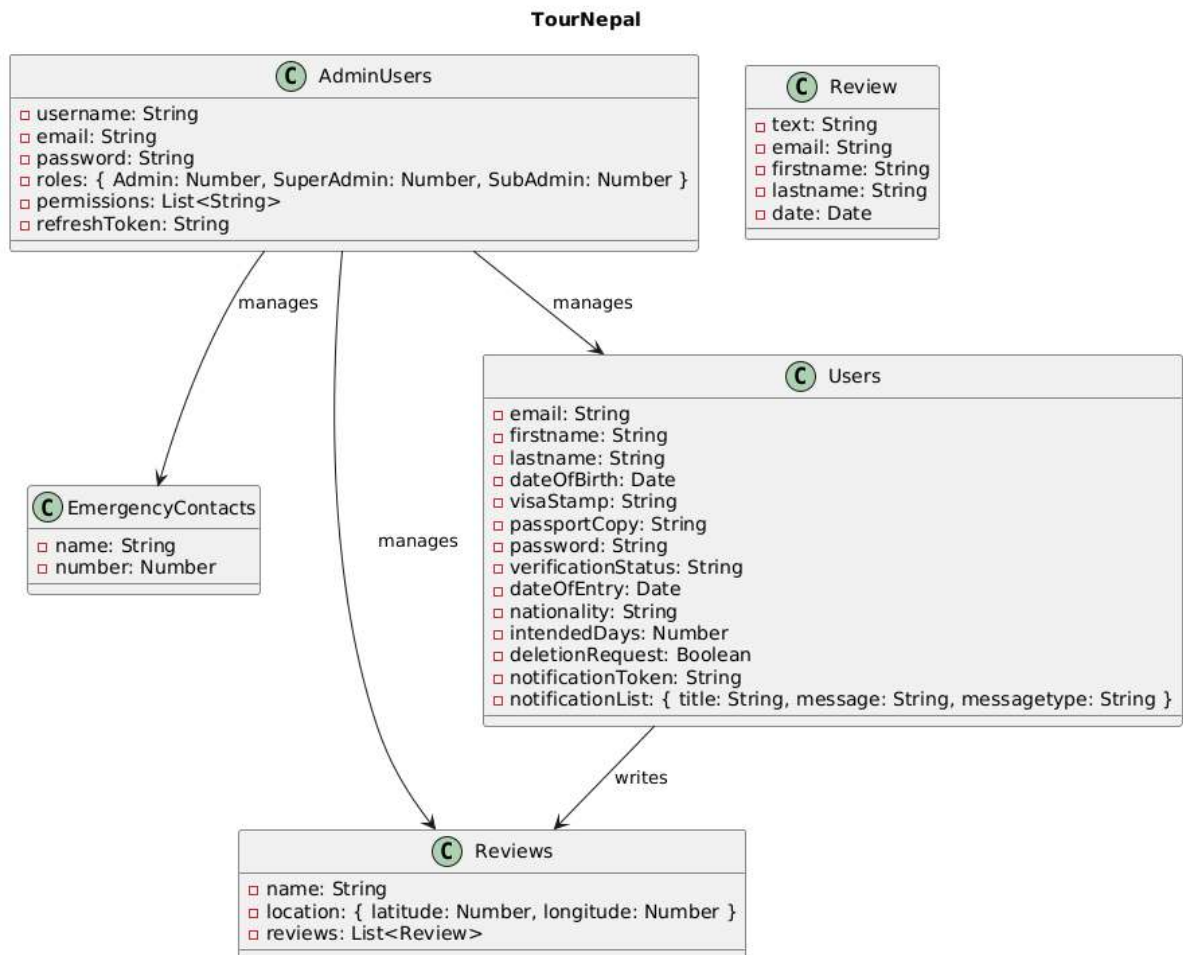


Figure 12: Logical Data Model

3.5.5. State diagram

State diagram is a diagram where the flow of response is explained from a single object of various activities and state of the application (Osis & Donins, 2017).

The following diagram shows the major states in the application and the transition between the states.

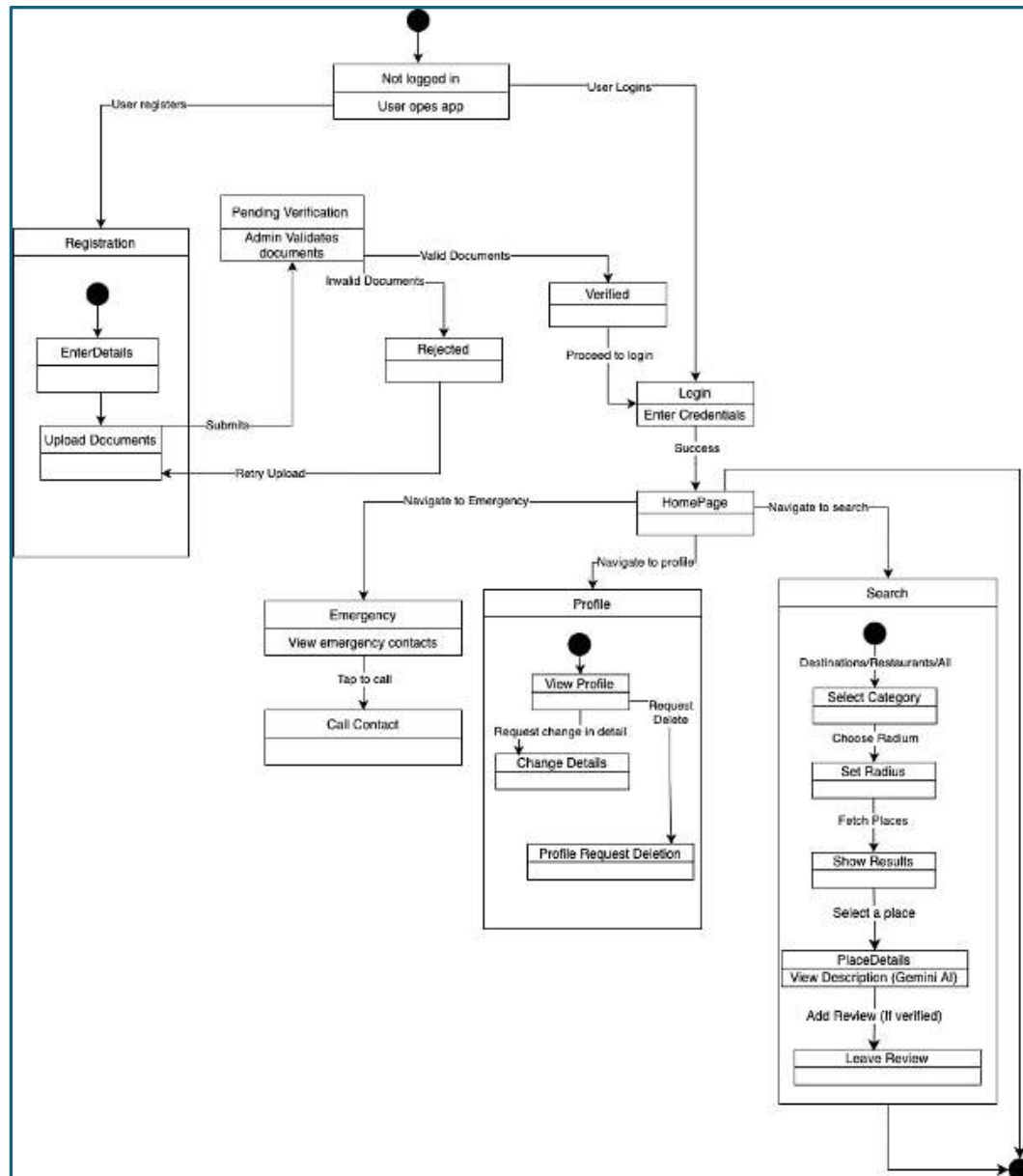


Figure 13: State diagram

3.5.6. Sequence diagram

Sequence diagram of the core functionalities of the application is provided below.

[\(Detailed description of sequence diagram can be found in Appendix H: 7.9.6\)](#)

I) Explore popular destinations

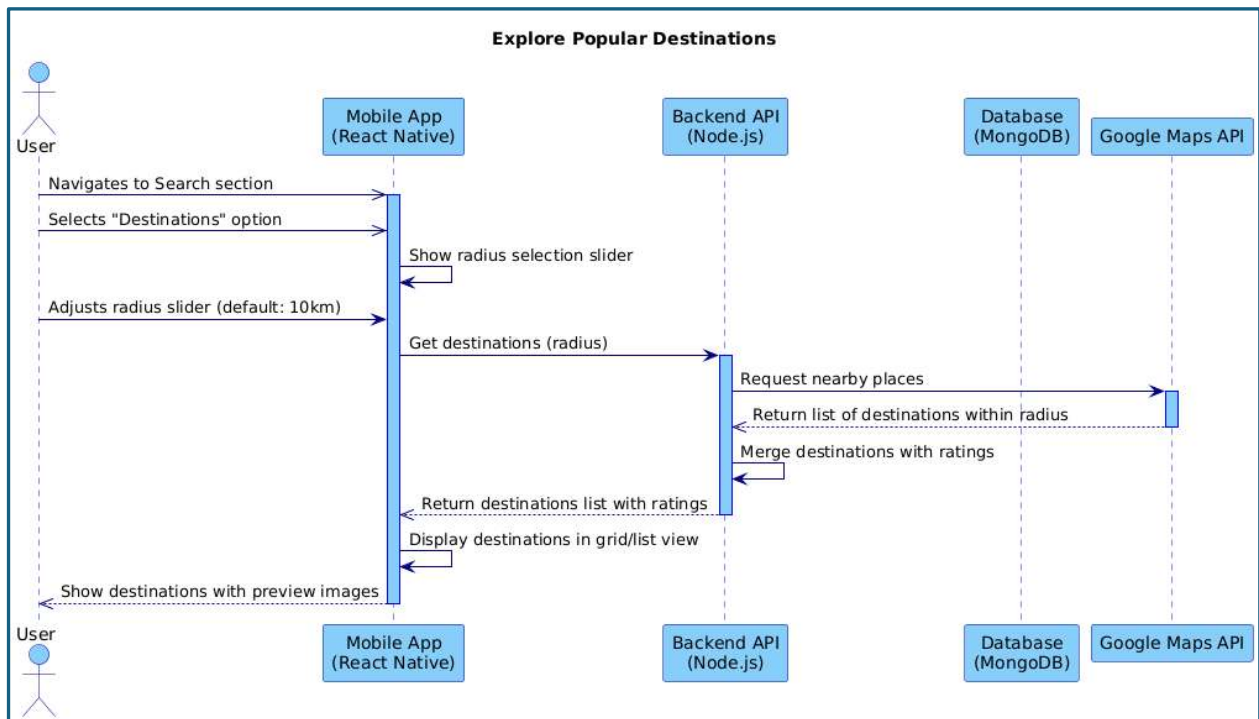


Figure 14: Explore popular destinations (Sequence Diagram)

II) Emergency contacts

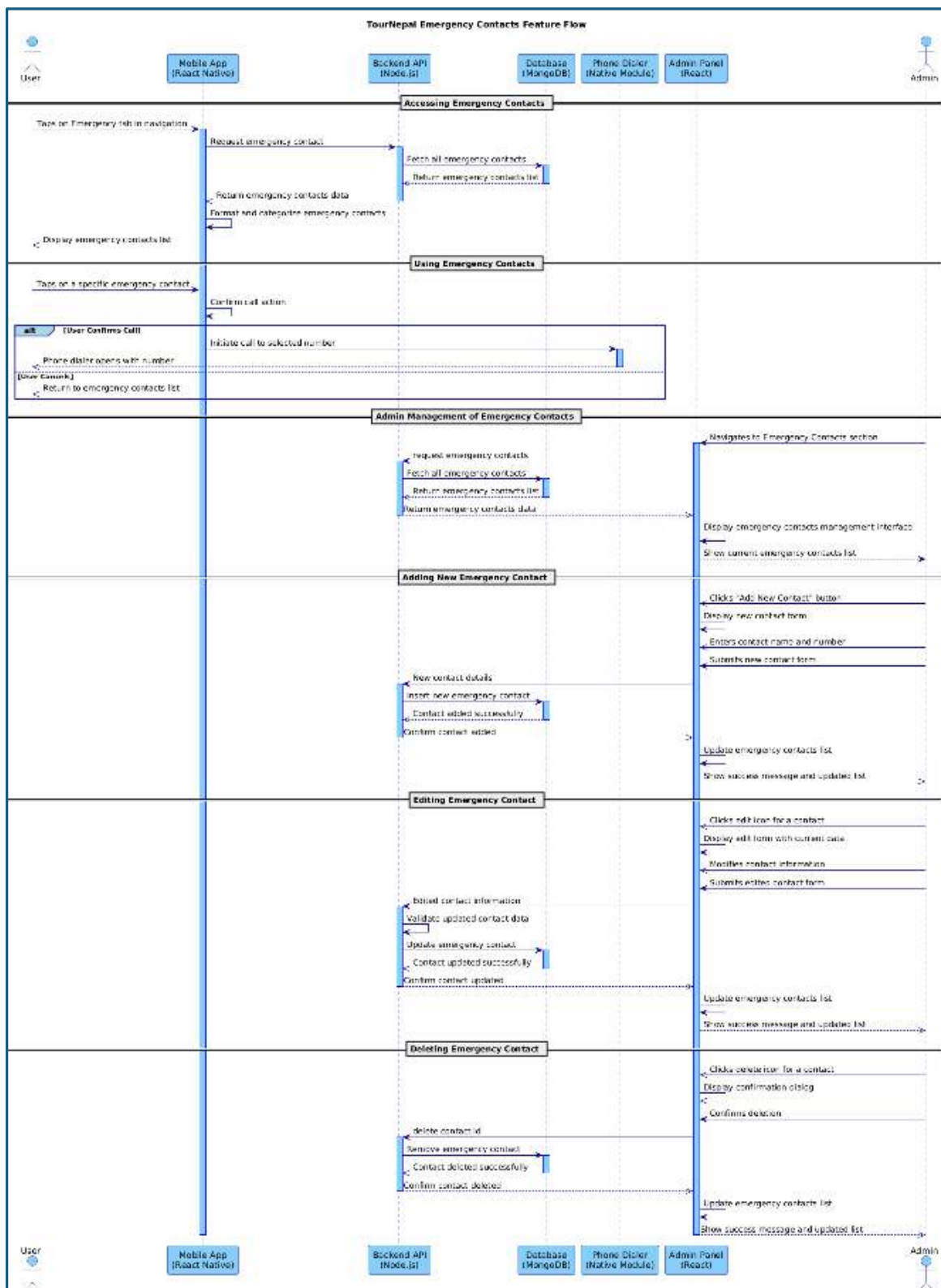


Figure 15: Emergency contacts (Sequence Diagram)

III) Account deletion request

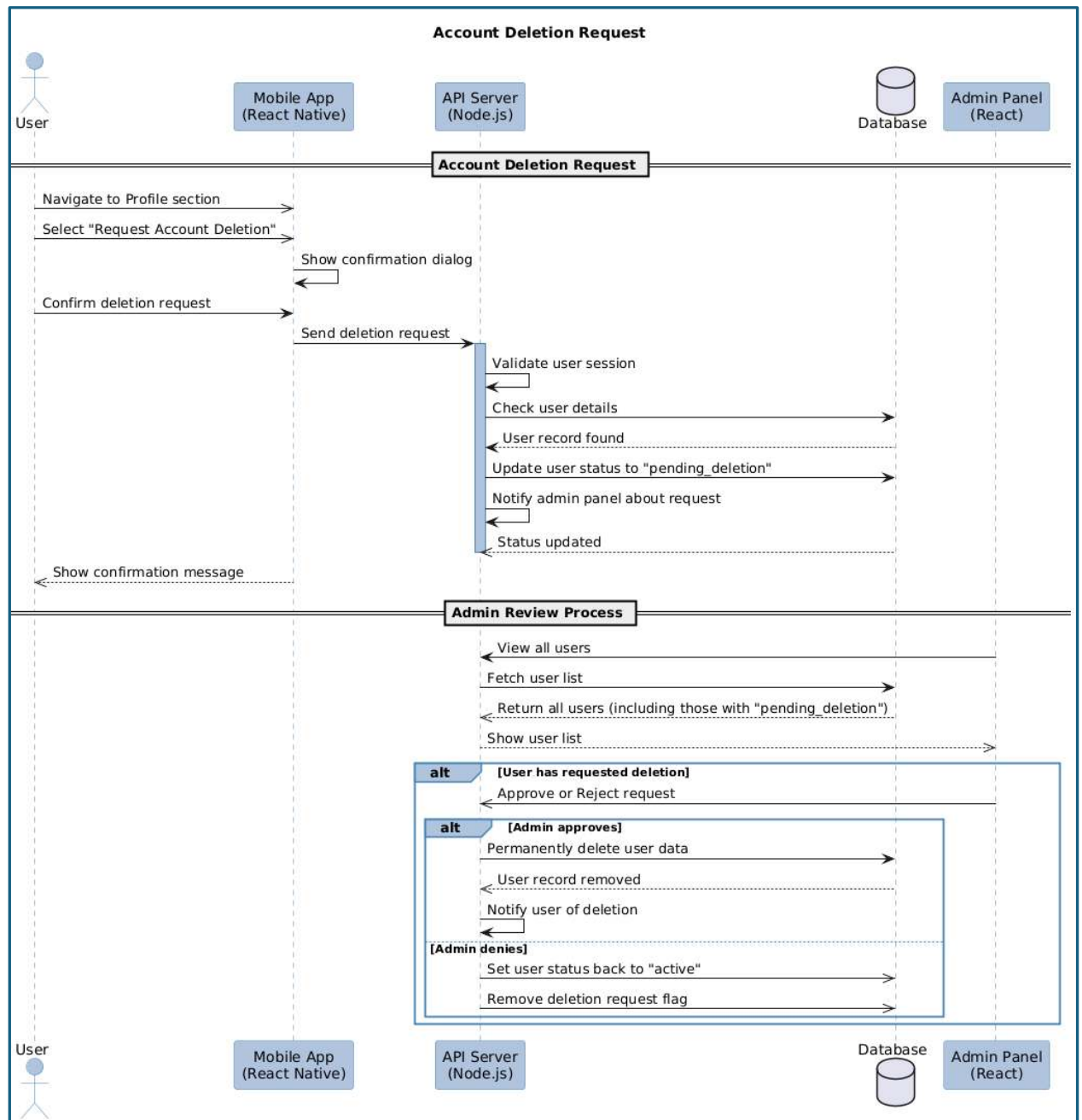


Figure 16: Account deletion request (Sequence Diagram)

IV) Admin document approval

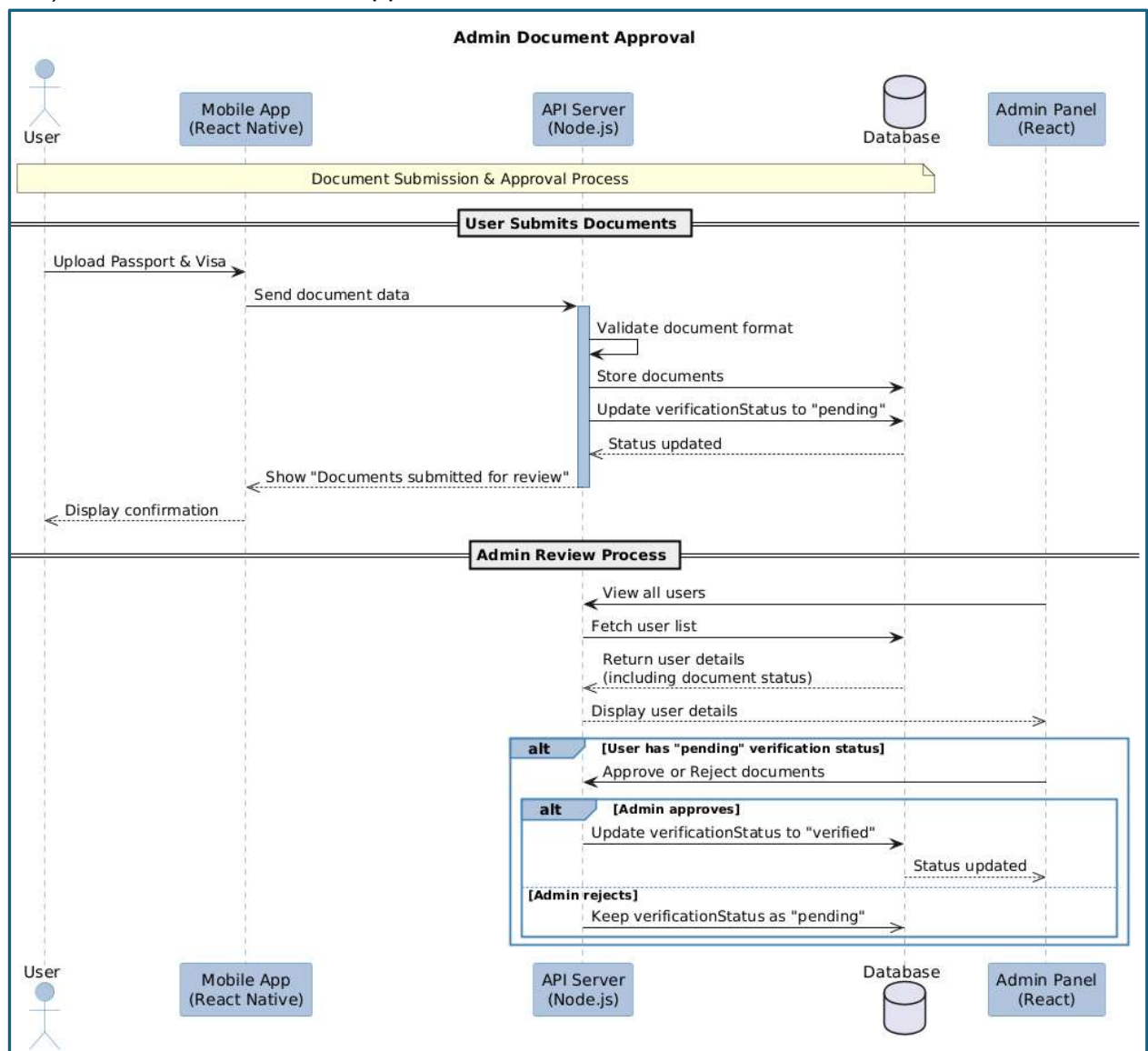


Figure 17: Admin document approval (Sequence Diagram)

V) Posting a review

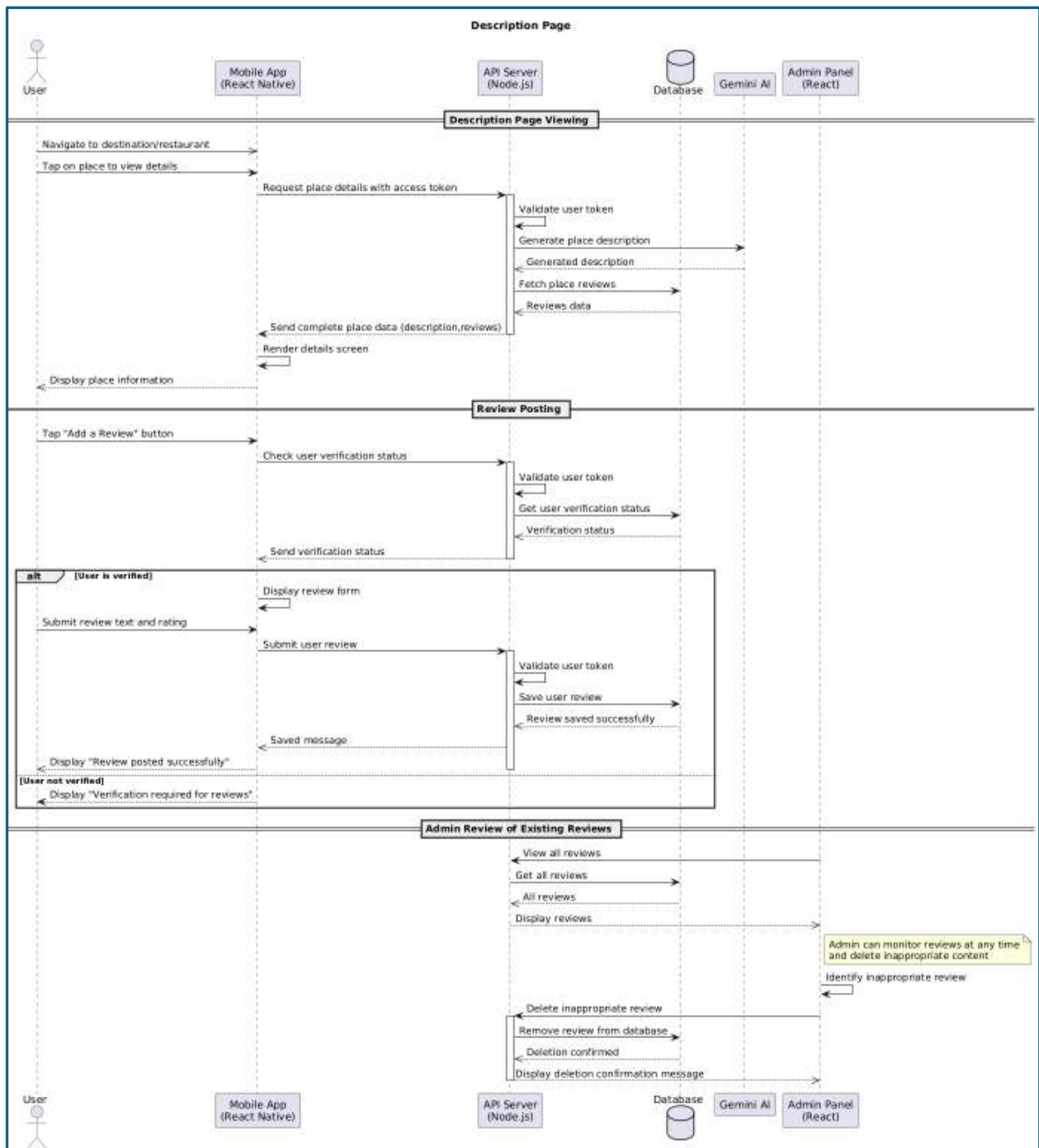


Figure 18: Reviews (Sequence diagram)

3.5.7. Activity diagram

Activity diagram of the core functionalities of the application is provided below.

[Other basic activity diagram can be found in the appendix H - 7.9.7\).](#)

I) Explore popular destinations

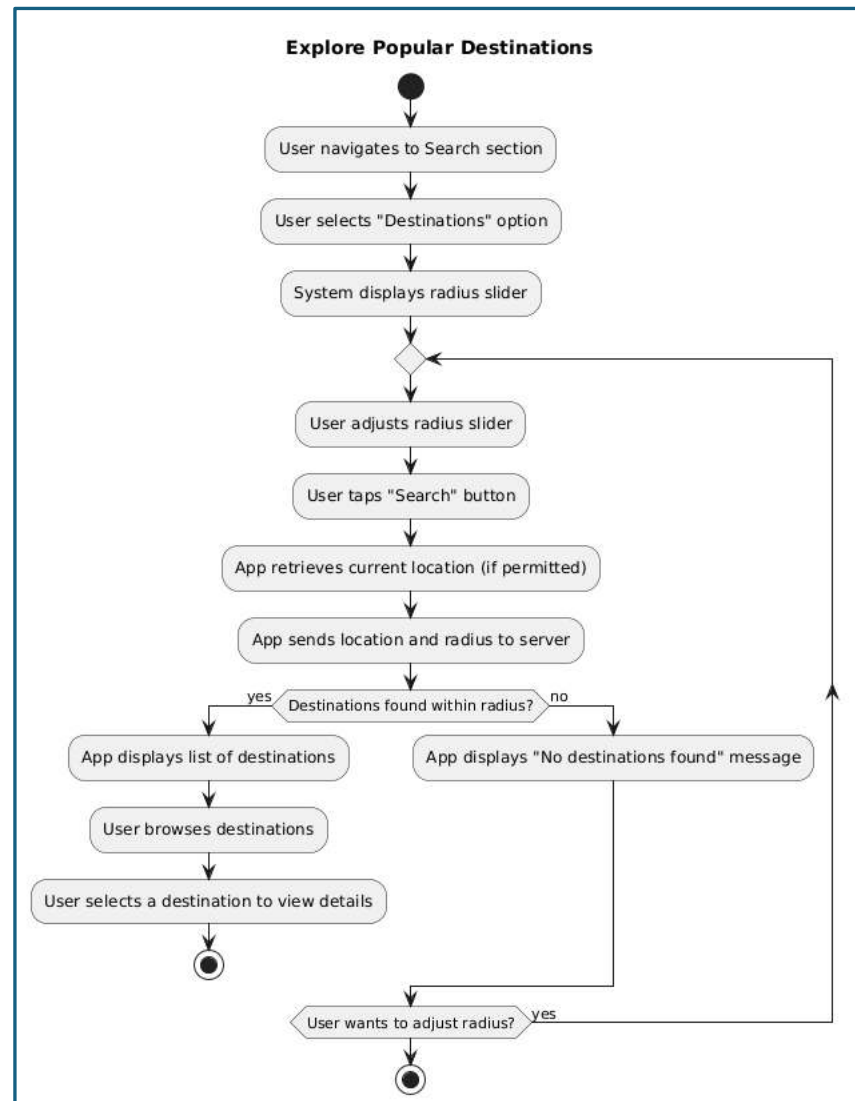


Figure 19: Explore popular destinations (Activity Diagram)

II) Emergency contacts

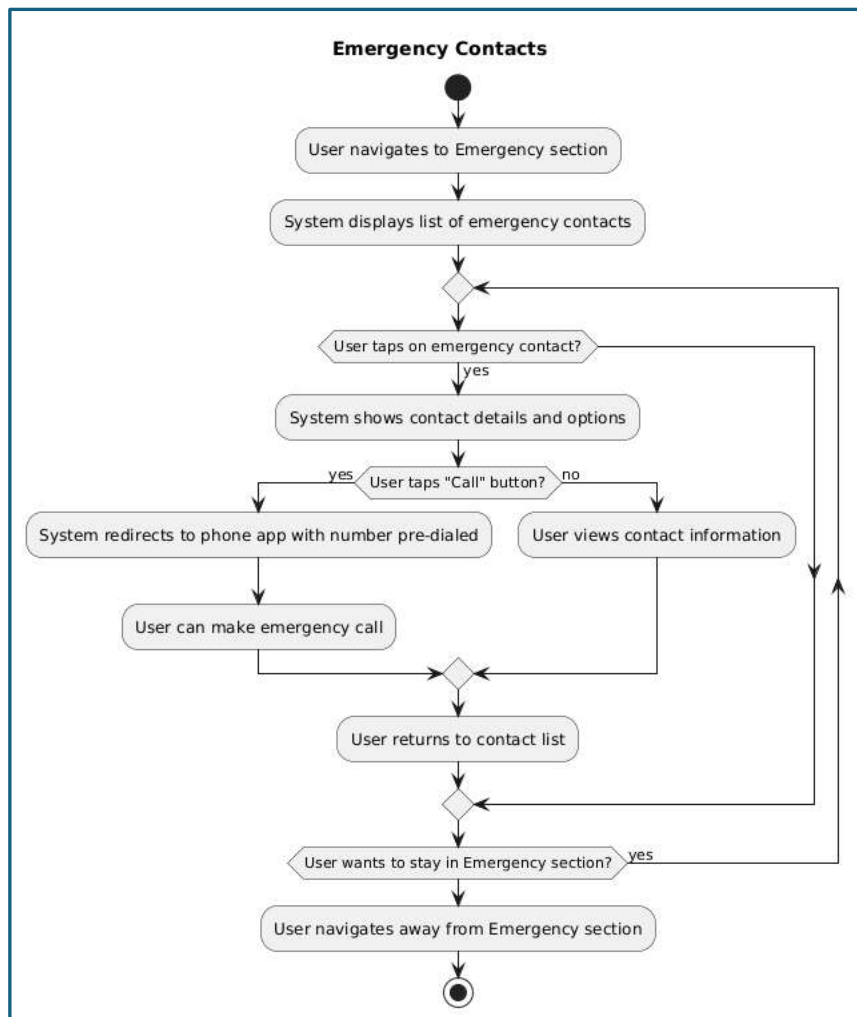


Figure 20: Emergency contacts (Activity Diagram)

III) Account deletion request

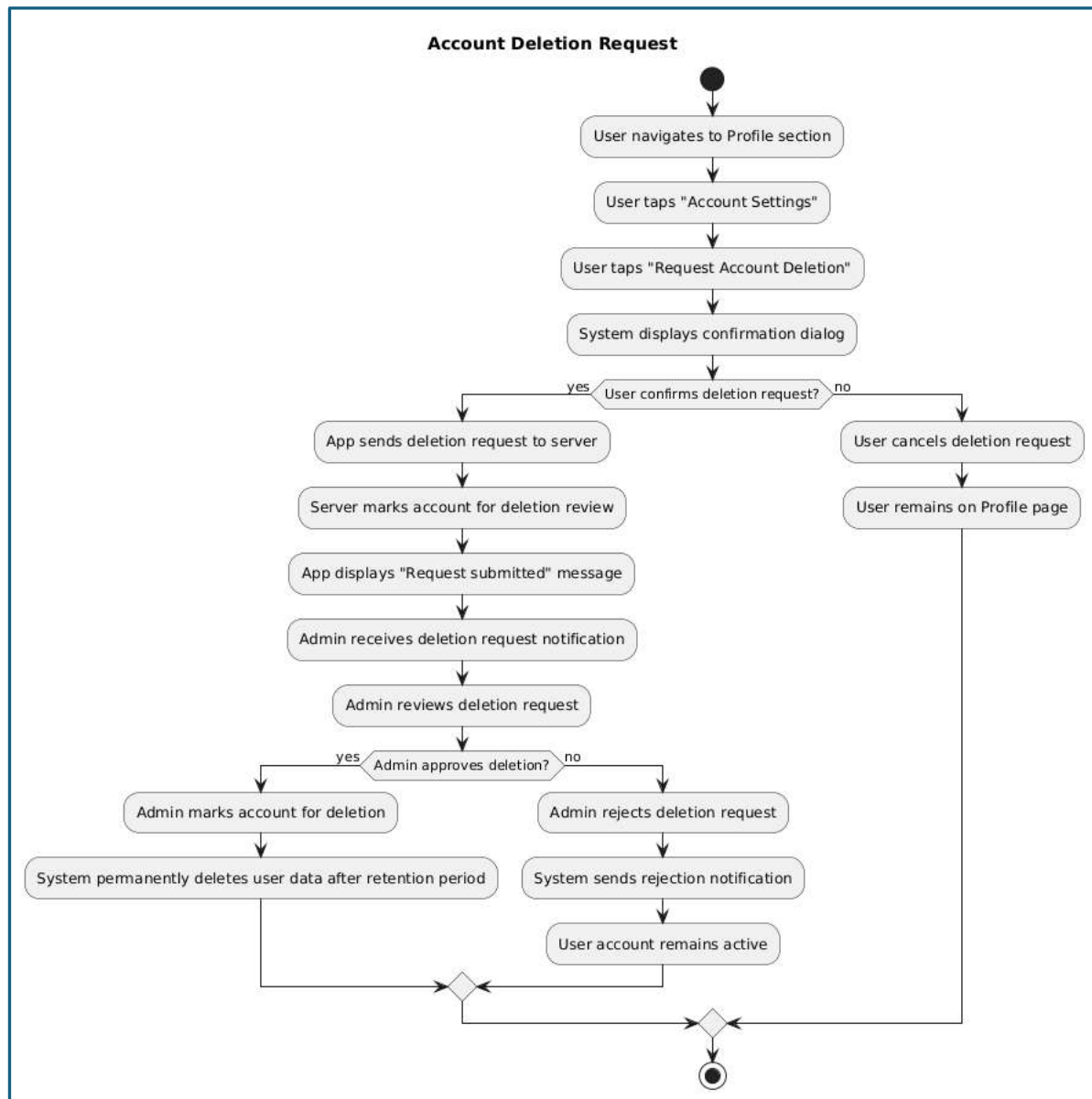


Figure 21: Account deletion request (Activity Diagram)

IV) Destination description

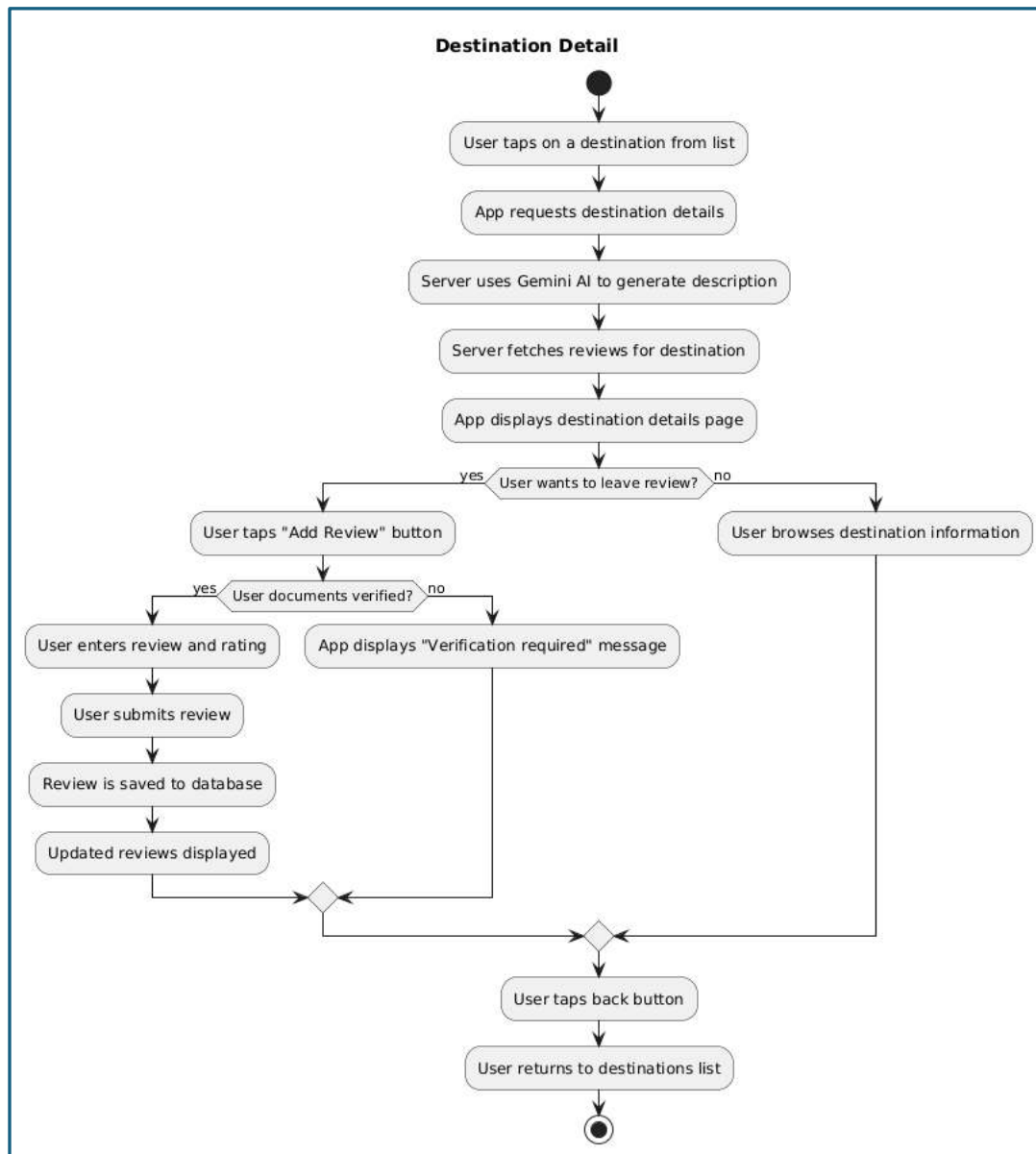


Figure 22: Destination description (Activity Diagram)

V) Posting a review

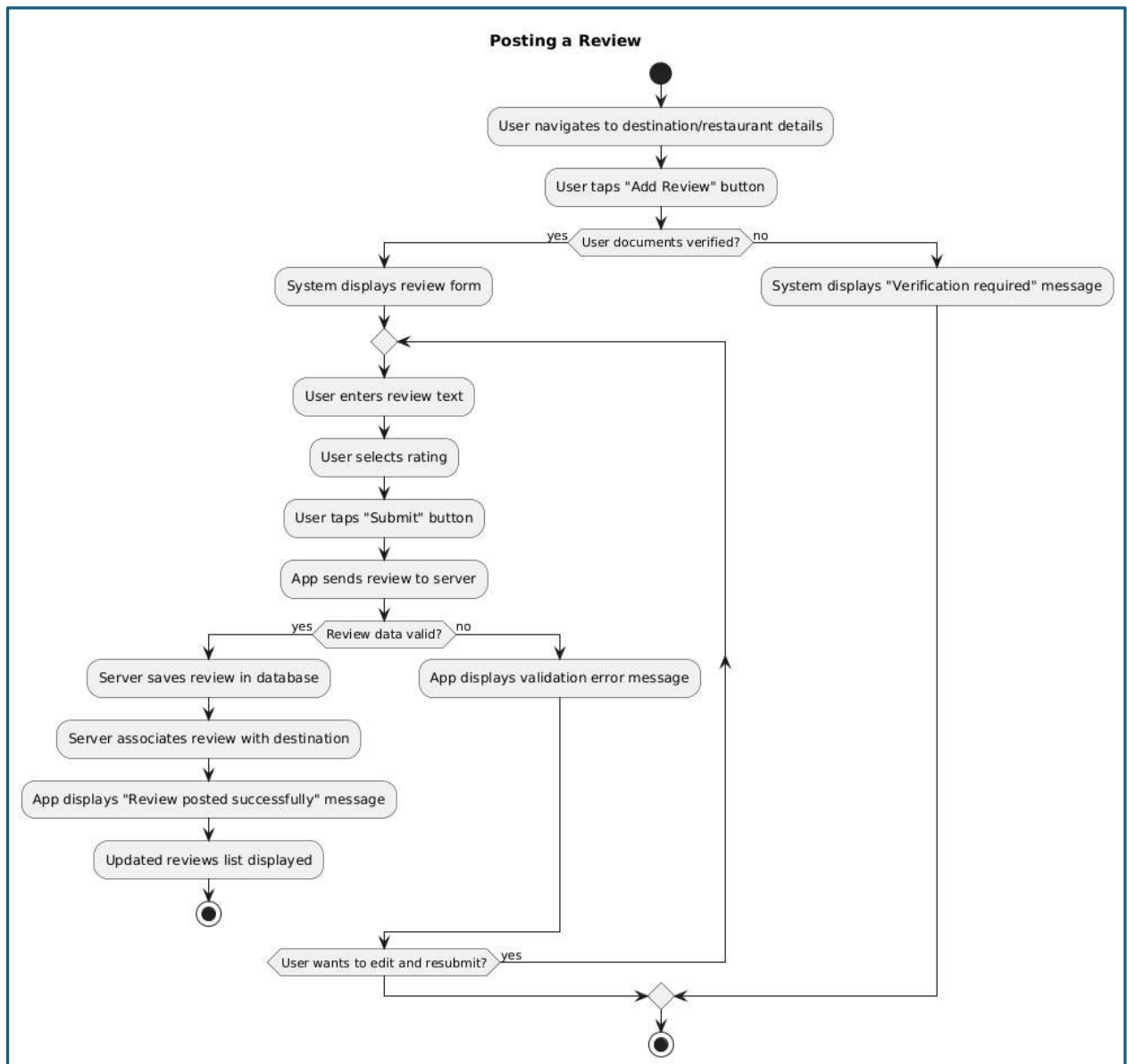


Figure 23: Posting a review (Activity Diagram)

3.6. Implementation

This section will cover the core development of the features in the application with code snippets and screenshots of the development. The project followed the Scrum methodology for development; hence, every step was separated into multiple sprints and was developed thoroughly.

3.6.1. Sprint 1 (01/11/2024-14/11/2024)

In the first sprint, the project was planned thoroughly. Many brainstorming tasks were performed with the client to understand the project. An interview was held with the client collecting requirements for the project. Since there are many technology stacks, proper research was performed on which to work. An SRS document was prepared that has requirements collected from an interview for understanding the application.

3.6.2. Sprint 2 (15/11/2024-28/11/2024)

With discussion with the client, the features required for the current scenario and context were also finalized. React native being cross-platform compatible with hot reload compatibility and Node.js offering a scalable for notifications like features was chosen. Additionally, the project being in a single language stack made it more efficient, and easy to understand the project.

3.6.3. Sprint 3 (29/11/2024-12/12/2024)

In the third sprint, the development of the project has been initiated. Since the application requires Google Maps API, proper research was performed on Google Maps API that included reading and understanding the official documentation of the Maps API. Furthermore, a simple setup of Google Maps API was performed to test the working of the API.

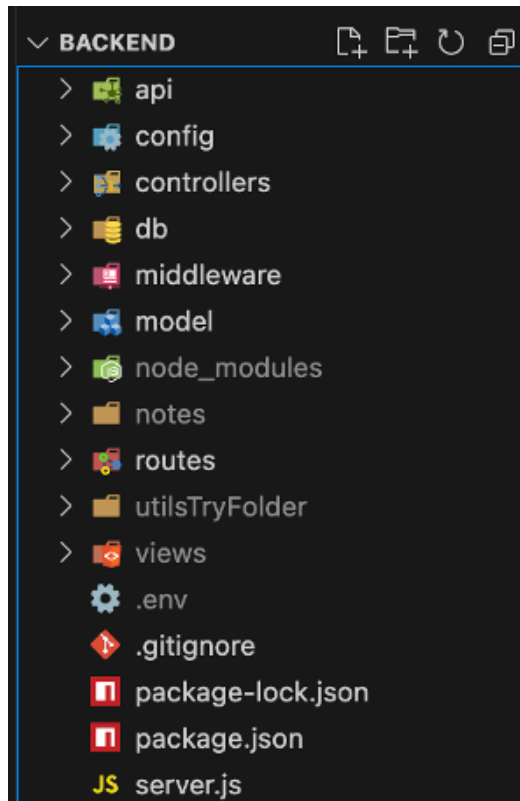


Figure 24: Backend project Structure Setup

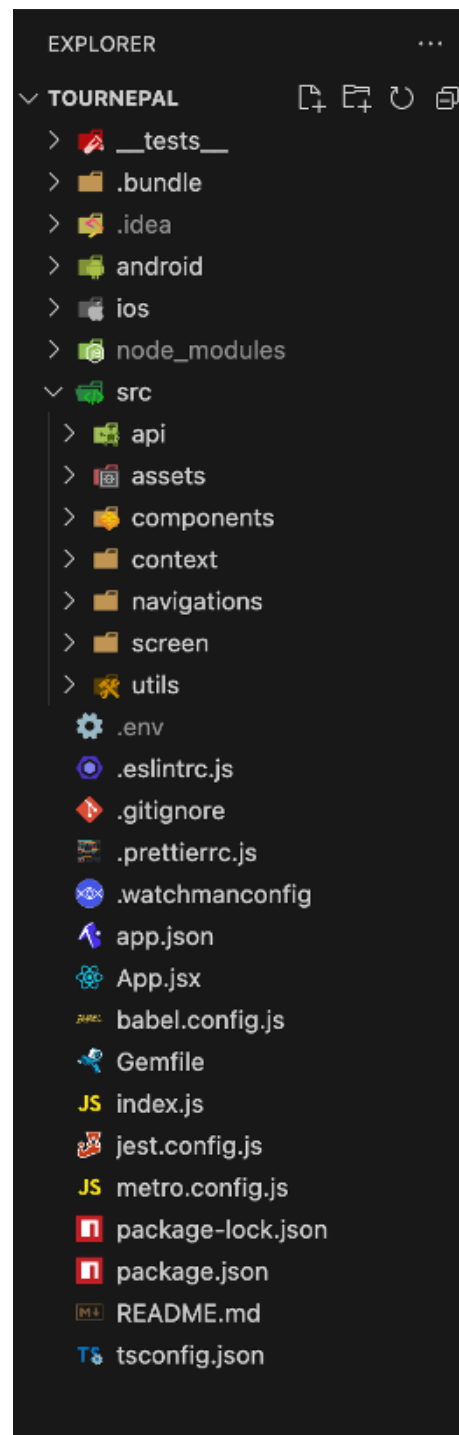


Figure 25: Frontend project Structure Setup

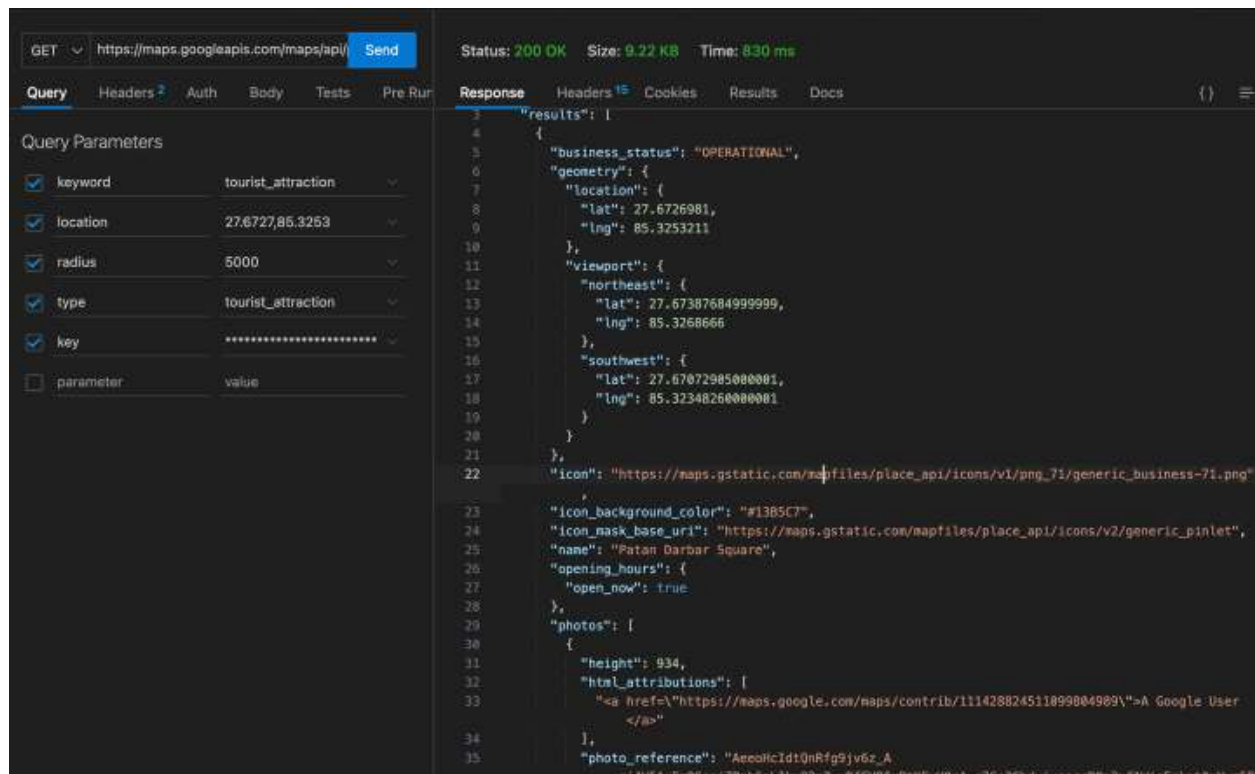


Figure 26: Google map Api testing

3.6.4. Sprint 4 (13/12/2024-26/12/2024)

Fourth step was a major leap for diving deep into the development since it will handle the entire operations and logic for the application. Before setting up the environments and framework, documentation from the official site was read to ensure the proper configuration of the dependencies. After setting up the dependencies, the task for registration and login was initiated. Since all the required data cannot be provided like visa photos, a dummy URL was used for testing purposes. Since the project will be using NoSQL, MongoDB is being used for storing data of users, contacts, and places; a new cluster was set up for storing data.

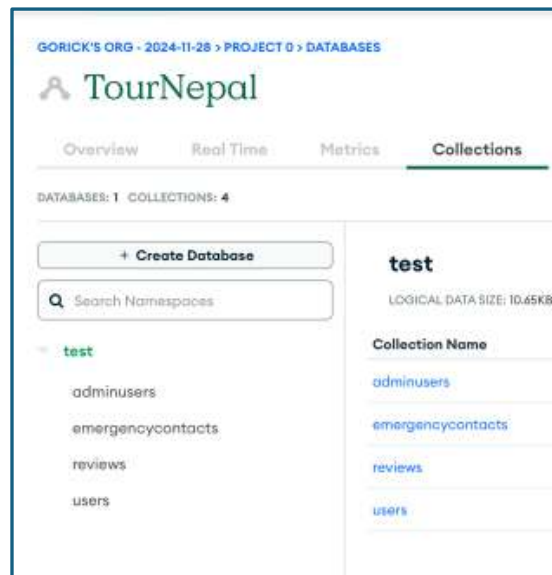


Figure 27: MongoDB cluster creation

```

JS server.js > [0] cors
1  require('dotenv').config();
2  const express = require('express');
3  const app = express();
4  const path = require('path');
5  const cors = require('cors');
6  const mongoose = require('mongoose');
7  const PORT = process.env.PORT || 3500;
8  const connectDB = require('./config/dbConn');
9  const { verifyRefreshToken } = require('./middleware/verifyUser');
10 const { verifyUserVerification } = require('./middleware/verifyDocumentStatus');
11 const bodyParser = require('body-parser');
12 const cookieParser = require('cookie-parser');
13
14
15
16 //connecting to database
17 connectDB();
18
19 const corsOptions = {
20   origin: 'http://localhost:5173', // Frontend URL
21   credentials: true,
22   optionsSuccessStatus: 200,
23 };
24 app.use(cors(corsOptions));
25
26 app.use(bodyParser.json());
27 app.use(express.urlencoded({ extended: false }));
28 app.use(express.json());
29
30
31 app.use(cookieParser());
32

```

Figure 28: Basic backend configuration setup

```

BACKEND/controllers/loginController.js
16 + const userDB = await User.findOne({ email }).exec();
17 + if (!userDB) return res.sendStatus(401);
18 +
19 + const match = await bcrypt.compare(password, userDB.password);
20 +
21 + if (match) {
22 +   //ASSIGNING Access Token to the user
23 +   const accessToken = jwt.sign({
24 +     "UserDetails": {
25 +       "email": userDB.email,
26 +       "firstname": userDB.firstname,
27 +       "lastname": userDB.lastname,
28 +       "dateOfBirth": userDB.dateOfBirth,
29 +       "visaStamp": userDB.visaStamp
30 +     },
31 +   },
32 +     process.env.ACCESS_SECRET_TOKEN,
33 +     {expiresIn: '1d'}
34 +   );
35 +
36 +
37 +   //Creating Refresh Token for persist login
38 +   const refreshToken = jwt.sign({
39 +     "email": userDB.email
40 +   },
41 +     process.env.REFRESH_SECRET_TOKEN,
42 +   );
43 +

```

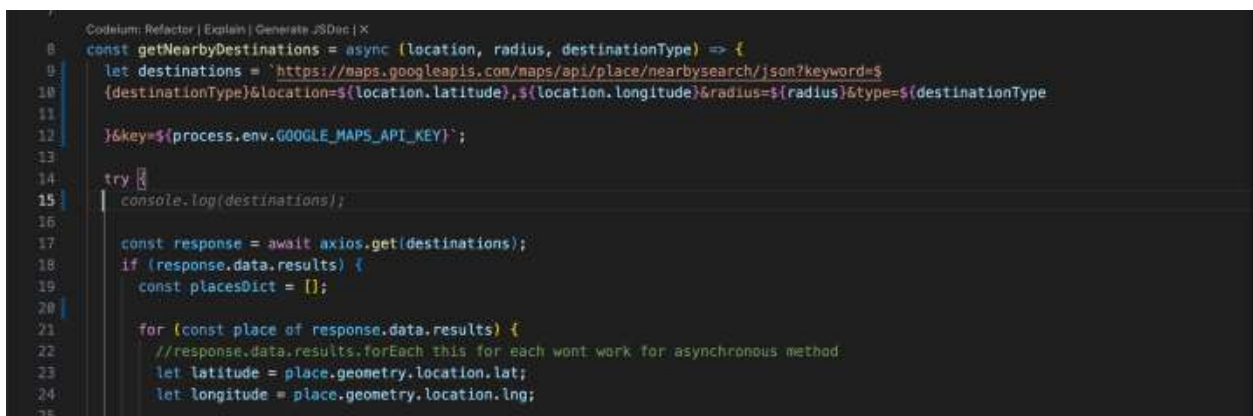
Figure 29: Login controller code

```
controllers > .js registerController.js > @ handleRegister > @ result
4   const handleRegister = async (req, res) => {
9
10  // check for duplicate usernames in the db
11  const duplicate = await User.findOne({ email: email }).exec();
12  if (duplicate) return res.sendStatus(409); //Conflict
13
14  try {
15    //encrypt the password
16    const hashedPwd = await bcrypt.hash(password, 10);
17
18    //create and store the new user
19    const result = await User.create({
20      "email": email,
21      "password": hashedPwd,
22      "firstname": firstname,
23      "lastname": lastname,
24      "dateOfBirth": dateOfBirth,
25      "visaStamp": visaStamp,
26      "passportCopy": passportCopy,
27      "verificationStatus": "pending" ,
28      "dateOfEntry": dateOfEntry,
29      "nationality": nationality,
30      "intendedDays": intendedDays,
31      "deletionRequest": false,
32      "notificationList": []
33    });
34
35    console.log(result);
36
37    res.status(201).json({ 'success': 'New user ${firstname} created!' });
38  } catch (err) {
39    res.status(500).json({ 'message': err.message });
40    console.log(err.message)
41  }
42 }
43
```

Figure 30: Register controller code

3.6.5. Sprint 5 (27/12/2024-09/01/2025)

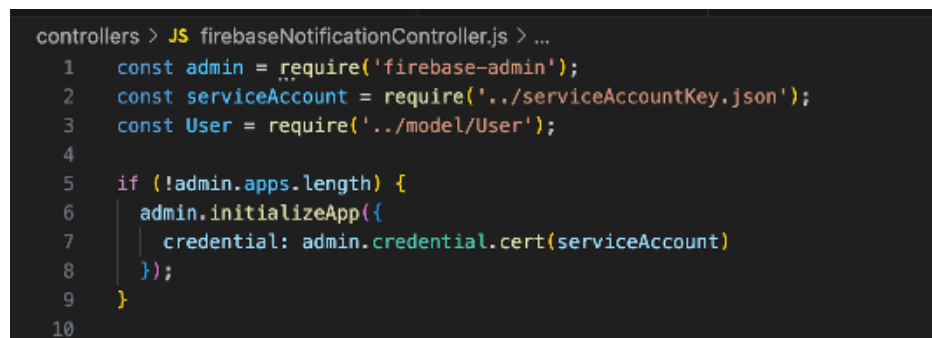
This sprint was responsible for developing Api for restaurants and destinations to fetch user places around them. Many Apis were tested for map integration but among them google Api was chosen as it fulfilled the requirement comparatively. After research on google maps Api, it was cleared that even Google Map Api came with certain limitations like not being able to provide description of places. For destinations around selected location, nearby search API by google was used. So to fetch information google gemini was used with very restricting prompt that made sure no incorrect information was fetched.



```
Codeium: Refactor | Explain | Generate | JSDoc | X
8 const getNearbyDestinations = async (location, radius, destinationType) => {
9   let destinations = `https://maps.googleapis.com/maps/api/place/nearbysearch/json?keyword=${
10     destinationType}&location=${location.latitude},${location.longitude}&radius=${radius}&type=${destinationType
11   }&key=${process.env.GOOGLE_MAPS_API_KEY}`;
12
13   try {
14     console.log(destinations);
15
16     const response = await axios.get(destinations);
17     if (response.data.results) {
18       const placesDict = [];
19
20       for (const place of response.data.results) {
21         //response.data.results.forEach this for each wont work for asynchronous method
22         let latitude = place.geometry.location.lat;
23         let longitude = place.geometry.location.lng;
24       }
25     }
26   }
27 }
```

Figure 31: Google map Api development

Along with google maps api, setup for firebase was also performed in Node.js.



```
controllers > JS firebaseNotificationController.js > ...
1  const admin = require('firebase-admin');
2  const serviceAccount = require('../serviceAccountKey.json');
3  const User = require('../model/User');
4
5  if (!admin.apps.length) {
6    admin.initializeApp({
7      credential: admin.credential.cert(serviceAccount)
8    });
9  }
10
```

Figure 32: Firebase setup

3.6.6. Sprint 6 (10/01/2025-23/01/2025)

In this sprint, screens for the mobile application were developed. Several main screens like, login, signup and listing of restaurants/destination was developed. In this sprint, the listing of places was static however, the login and registration was integrated with backend.

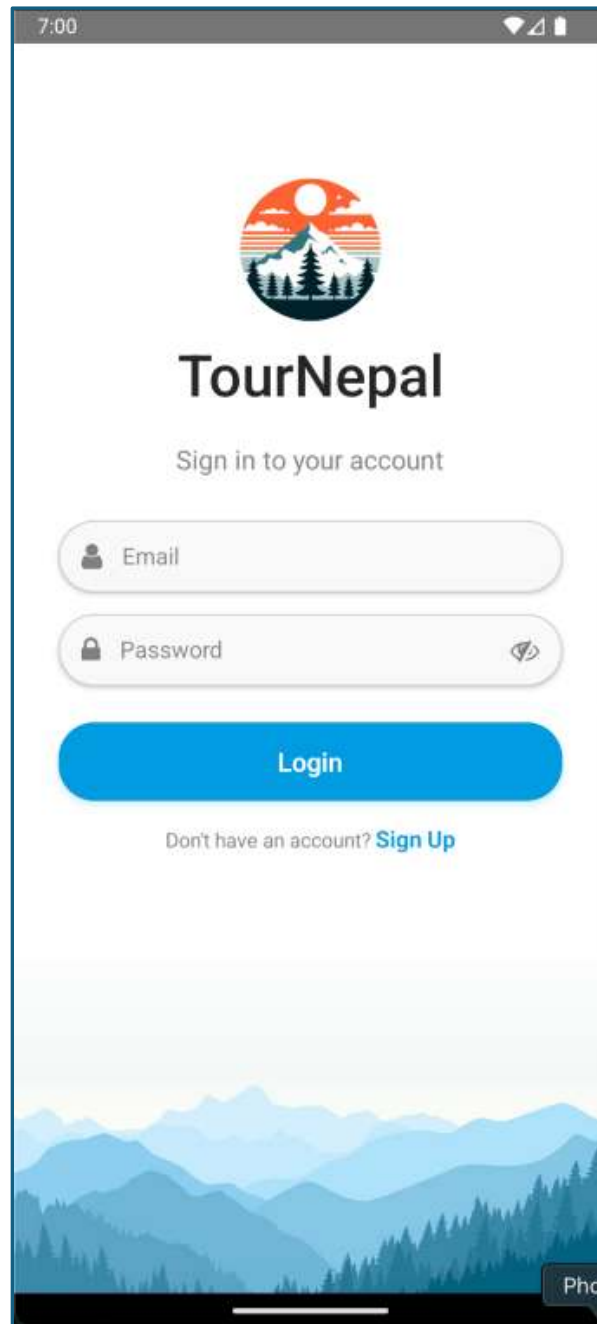
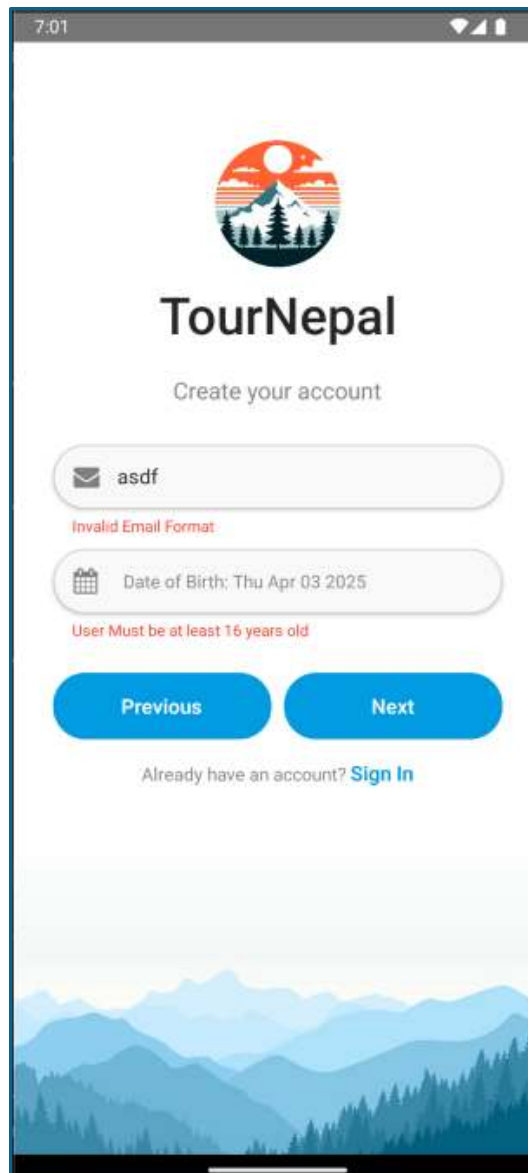


Figure 33: Log in screen



The image shows a mobile application interface for 'TourNepal'. At the top, there is a status bar with the time '7:01' and signal icons. Below this is a circular logo featuring a stylized mountain, trees, and a sun. The text 'TourNepal' is prominently displayed in a large, bold, black font. Underneath the logo, the text 'Create your account' is centered. The form consists of two input fields. The first field is for an email address, containing the text 'asdf', with a red error message 'Invalid Email Format' below it. The second field is for a date of birth, containing the text 'Date of Birth: Thu Apr 03 2025', with a red error message 'User Must be at least 16 years old' below it. Below the input fields are two blue buttons labeled 'Previous' and 'Next'. At the bottom of the form, there is a link that says 'Already have an account? Sign In'. The background of the screen features a scenic illustration of mountains and a forest.

7:01



TourNepal

Create your account

✉ asdf

Invalid Email Format

📅 Date of Birth: Thu Apr 03 2025

User Must be at least 16 years old

Previous Next

Already have an account? [Sign In](#)

Figure 34: Sign up screen

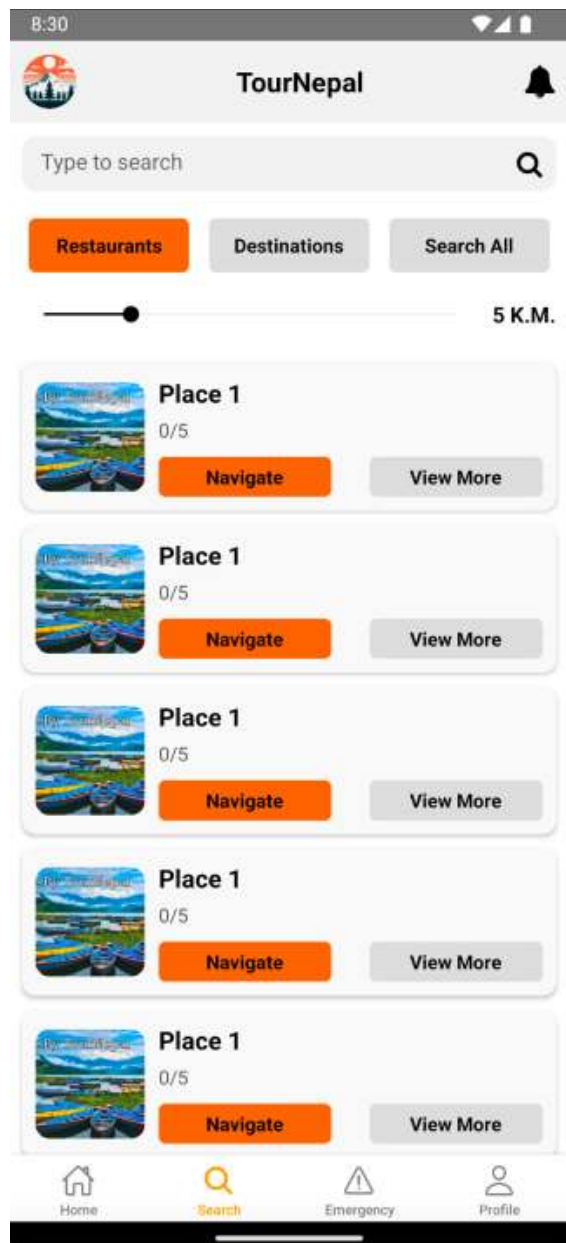


Figure 35: Destination listing static

3.6.7. Sprint 7 (24/01/2025-06/02/2025)

In this sprint, the previously developed Api will be integrated, with frontend destination/restaurants listing. Since google map nearby search Api does not provide image of the place, google map place photos was used for retrieving photo, where the URL was sent that was retrieved from the response object of the place.

```
const { returnDescription } = require('../api/geminiDescription');
const {getNearbyDestinations} = require('../api/googleMapsInt')
const Reviews = require('../model/Reviews');

const getNearbyPlaces = async (req,res)=>{
  const {latitude,longitude,radius,destinationType} = req.query
  const decodedRadius = radius
  console.log(radius);

  if(!latitude || !longitude || !radius) return res.status(400).json({ message: "Location and Latitude are required." });

  if(!destinationType) return res.status(400).json({message:"Destination type is required for location fetching."})

  //now requesting from google maps api

  const destinations = await getNearbyDestinations(latitude,longitude,decodedRadius,destinationType)

  // destinations.forEach((place)=>{
  //   // const dest
  // })
  res.status(200).json({"ref":destinations})
}
```

Figure 36: Google map Api integration (Backend)

```
const getNearbyDestinations = async (latitude,longitude, radius, destinationType) => {
  let lat = parseFloat(latitude)
  let lon = parseFloat(longitude)
  let rad = parseFloat(radius)
  let destinations = `https://maps.googleapis.com/maps/api/place/nearbysearch/json?keyword=${destinationType}&location=${lat},${lon}&radius=${rad}&ty`

  try {
    if (isNaN(rad)) {
      destinations = `https://maps.googleapis.com/maps/api/place/textsearch/json?query=${radius}+in+Nepal&region=np&key=${process.env.GOOGLE_MAPS_API}`
    }
    const response = await axios.get(destinations);
    if (response.data.results) {
      const placesList = []
    }
  }
}
```

Figure 37: Google map API integration (Backend 02)

```
35  useEffect(() => {  
36      Codeium: Refactor | Explain | Generate JSDoc | X  
37      const getPlaces = async () => {  
38  
39          setIsLoading(true);  
40          const places = await nearbyPlaces(radius * 1000, currentSelection, currentLocation);  
41  
42          setIsLoading(false);  
43          setAllPlaces(places);  
44          setFilteredPlaces(places);  
45      };  
46  
47      if (!locationPermission && currentSelection !== 'all') {  
48          getPlaces();  
49      } else if (!locationPermission) {  
50          Alert.alert(  
51              'Location Permission Required',  
52              'Please enable location permission to get nearby places',  
53              [  
54                  {  
55                      text: 'Open Settings',  
56                      onPress: () => Linking.sendIntent('android.settings.LOCATION_SOURCE_SETTINGS')  
57                  },  
58                  {  
59                      text: 'Cancel',  
60                      style: 'cancel'  
61                  }  
62              ],  
63              {cancelable: false}  
64          );  
65      }  
66  }, [radius, currentSelection, currentLocation]);
```

Figure 38: Google map Api integration frontend

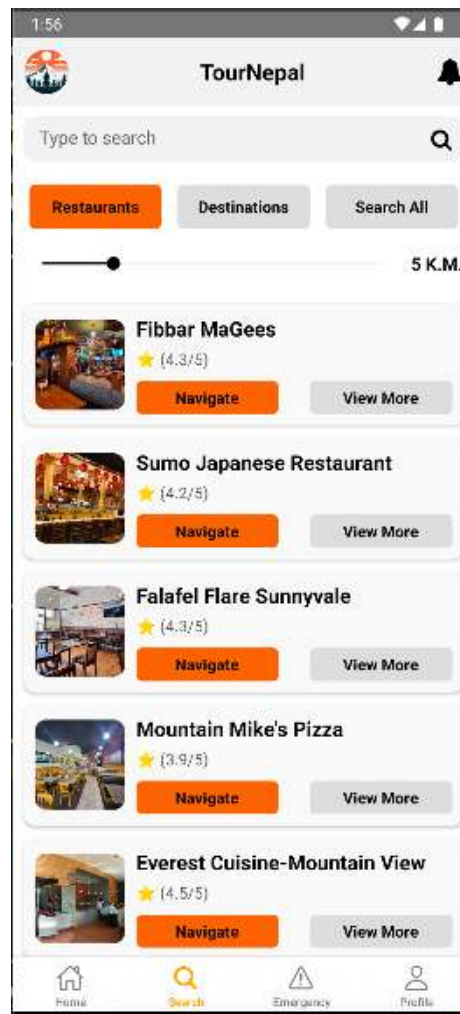


Figure 39: Destination search (Google maps Api)

3.6.8. Sprint 8 (07/02/2025-20/02/2025)

In this sprint, the feedback feature is implemented, which is one of the unique features of the application. Any user can view various destinations; however, only the user with a verified user status is allowed to leave a comment. This does not limit any user from viewing the reviews, whether verified or not.

When checking the user status, several security checks are performed. When a user tries to post a comment, the comment box is disabled if the user is not verified. After a user posts a comment, further token verification and user verification are performed in the backend, ensuring secure and genuine feedback

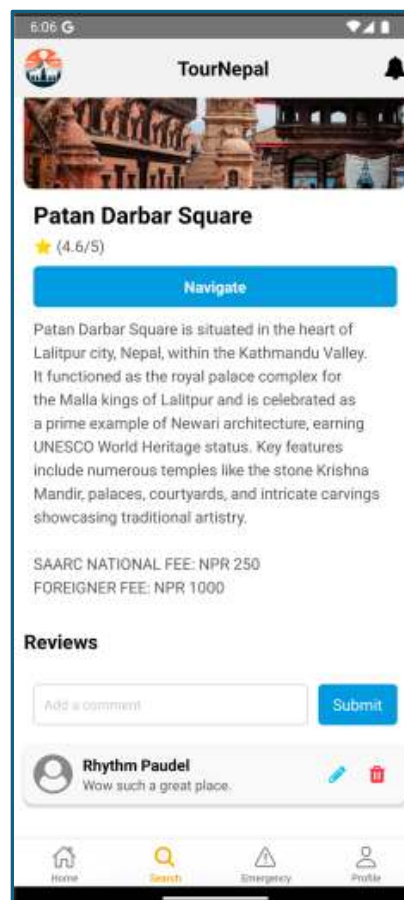


Figure 40: Verified user posting a review

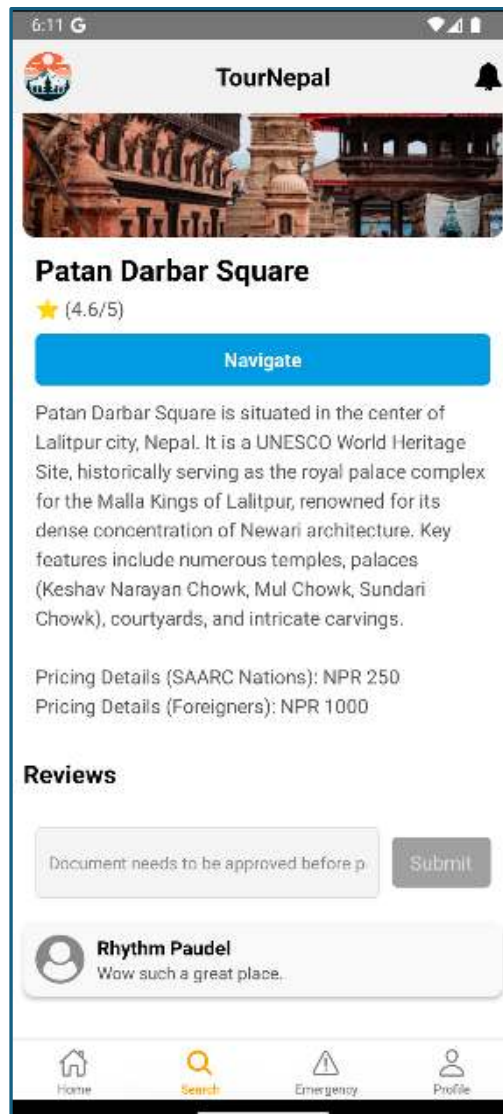


Figure 41: Unverified user posting a review

3.6.9. Sprint 9 (21/02/2025-06//03/2025)

In this sprint, all the major tasks related to the admin panel development were performed. It includes CRUD operations of emergency contacts and users. From the admin panel, the admin is also able to filter through all the reviews, and if they are inappropriate or unauthentic, the admin can delete the review for authenticity.

Admin is also able to verify user document and approve or deny delete the request of the user if any and suitable. Furthermore, from admin panel admin is also able to send a notification to a specific or all the users. However, notification sending feature was not implemented in this sprint as it involved complex documentation understanding and logic building process.

The screenshot displays the 'TourNepal Admin' interface. On the left is a dark sidebar with navigation links: Home, Users, Emergency Contacts, Reviews, Admins, Notifications, and Logout. The main content area is divided into two sections. The top section, 'Personal Information', contains input fields for First Name (Rhythm), Last Name (Paudel), Email (rhythm@gmail.com), Date of Birth (05/04/2025), Nationality (CA), Date of Entry (05/04/2025), and Indented Days of Stays (30). The bottom section, 'Document Verification', shows two large white boxes for 'Visa Copy' and 'Passport Copy', and a 'Document Status' dropdown menu currently set to 'Verified'. A user profile card at the bottom left of the sidebar shows the name 'rhythm' and email 'rhythmpaudel@gmail.com'.

Figure 42: User edit/verification via admin panel

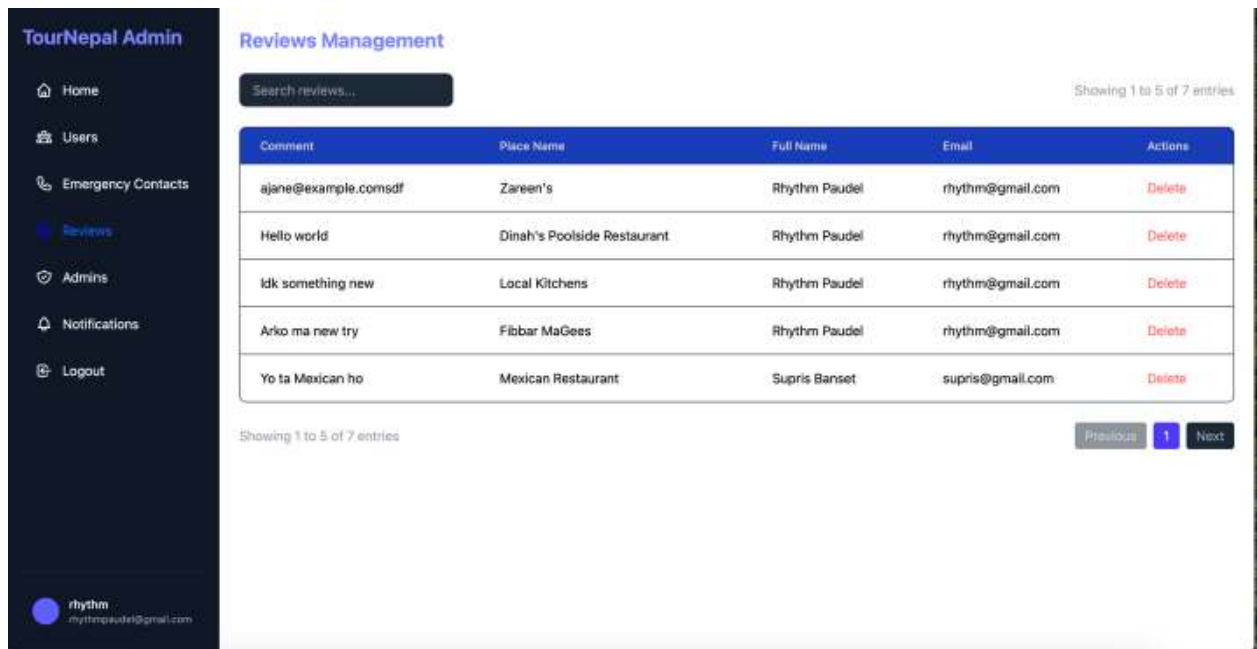


Figure 43: Review filtering admin panel

3.6.10. Sprint 10 (07/03/2025-20/03/2025)

In this sprint, the notification system was properly developed. For testing purposes, notifications were being sent from the Firebase console, which is the default way of sending notifications via Firebase. Using an endpoint, a working backend was prepared. After the development of the notification system in the backend, it was integrated with the frontend side.

The token for each user was assigned at the time of login. Every time a user logged in; their notification token was updated with the recent mobile device token. This made sure only one device was receiving notifications for an account which was last logged in.

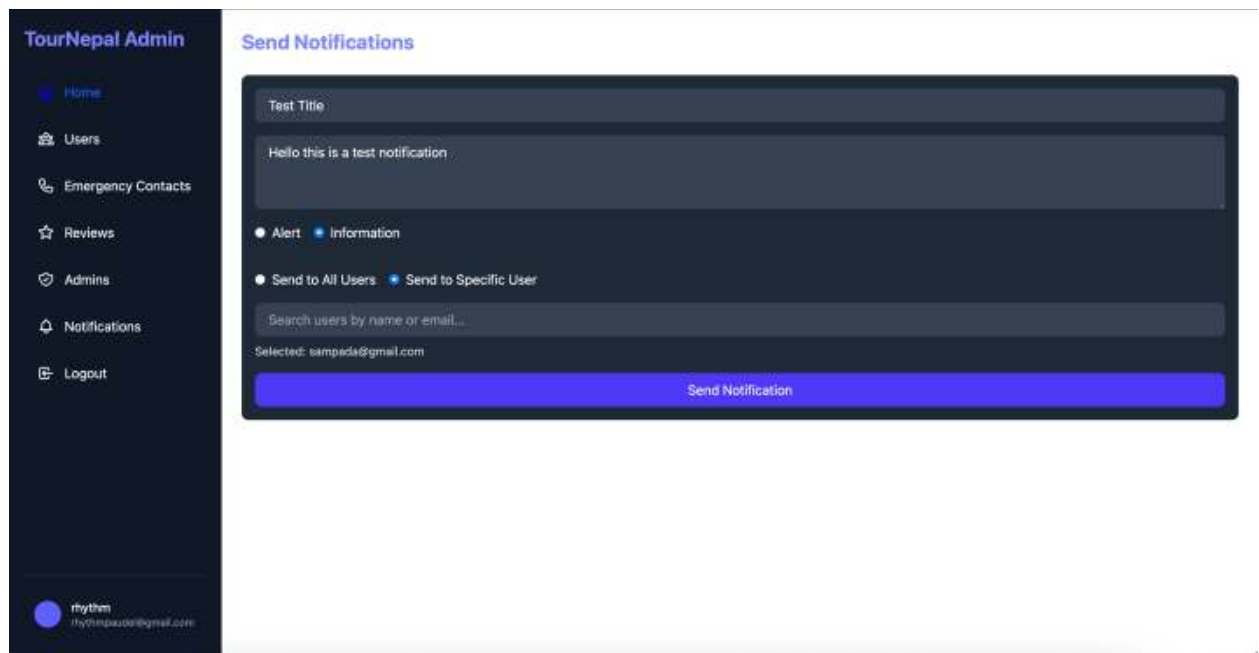


Figure 44: Notification implementation

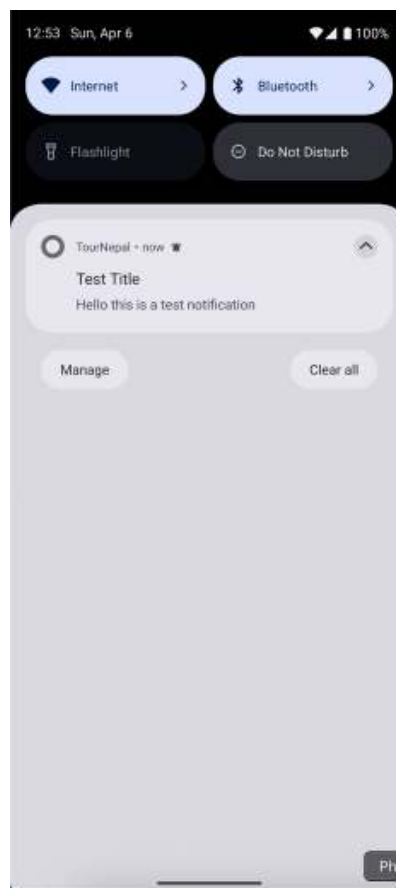


Figure 45: Notification mobile with test message

3.6.11. Sprint 11 (21/03/2024-03/04/2025)

This sprint is the final sprint for the development of the application/project. Finally, emergency contact features (previously holding static values) were implemented with updated and new emergency contacts directly from the database. The emergency numbers are editable from the admin panel and are updated in the mobile application as well. The user from the mobile application will be redirected when clicked on any phone number.

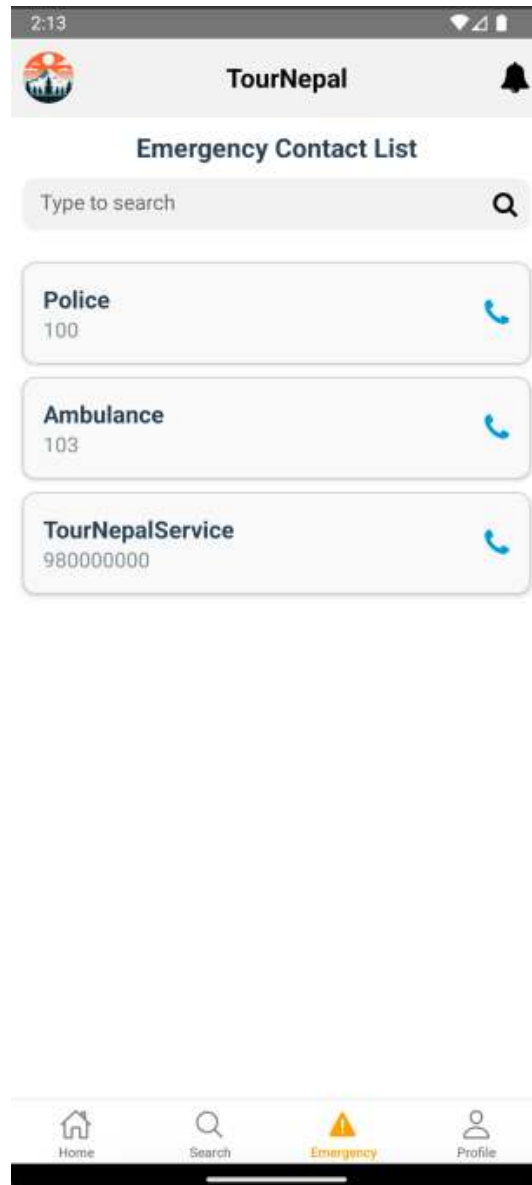


Figure 46: Emergency contact list mobile UI

Chapter 4. Testing and Analysis

This chapter holds an importance, as several types of testing will be performed, which will evaluate the reliability, accuracy and security of the application. The objective of this chapter is to verify the proper working of the application with proper error handling in case of system failure. Tests performed in this chapter highlights the strengths and limitations of the application.

4.1. Test plan

4.1.1. Unit testing test plan

Test cases	Objectives
UT-01	To test if server runs properly on specified port.
UT-02	To test if server is being connected to database.
UT-03	To test login functionality for different inputs and outcomes.
UT-04	To handle response in CRUD operations and notifications for users.
UT-05	To test register functionality for different inputs and outcomes.
UT-06	To handle response in CRUD operations for emergency contacts
UT-07	To test if location is being retrieved of user.
UT-08	To test if notification access is requested when already not granted.
UT-09	To test if tokens are stored in android storage
UT-10	To test if Gemini api is working and provide a description of place in proper format.
UT-11	To test if Gemini responds properly when invalid name is provided to it for description.
UT-12	To check if the password are being properly hashed and matched.

Table 2: Unit test plan

4.1.2. API/ Integration test plans

Test cases	Objectives
AIT-01	To test if the server respond with proper message when no body is provided for login.
AIT-02	To test if access token and encrypted token are sent after successful login.
AIT-03	To test if the server respond with proper message when improper body is provided during registration process.
AIT-04	To test if server handles false values during registration process.
AIT-05	To test if registration process is successful with all the fields in proper format.
AIT-06	To test if the user is able to get a new access token with existing refresh token.
AIT-07	To test if the server responds with proper status code if new access token is requested with expired token.
AIT-08	To test if the server responds with proper status code if access token is requested without refresh token.
AIT-09	To test if user/admin can retrieve emergency contacts.
AIT-10	To test if google maps API is working as expected.
AIT-11	To test if server responds with proper error code if no location is sent while requesting destinations.

AIT-12	To test if server responds properly if no destination type is provided (restaurants or tourist attraction).
AIT-13	To test if places are retrieved with valid location and radius.
AIT-14	To test if any searched place is retrieved if string is passed in radius field.
AIT-15	To test if base64 image from google place photos is working.
AIT-16	To test if description request is working for valid place name.
AIT-17	To test if server responds properly when no place name for description is provided.
AIT-18	To test if existing reviews for a location is fetched upon request.
AIT-19	To test if proper response is sent when no reviews are found.
AIT-20	To test if server response properly when no latitude and longitude are provided for reviews.
AIT-21	To test if notifications for a valid token is sent to mobile phone.
AIT-22	To test if notifications for multiple devices is sent mobile phones.
AIT-23	To test if server responds properly when body is passed without all the fields for notifications.
AIT-24	To test if user details is retrieved by user email or not.

AIT-25	To test if user detail is retrieved by user email, without/expired access token.
AIT-26	To test if user deletion request is updated.
AIT-27	To test if user deletion request is rejected when no access token is provided for verification.
AIT-28	To test if user notification token is updated.
AIT-29	To test if user notifications history is fetched.
AIT-30	To test if notification for improper/invalid tokens is handled properly.
AIT-31	To test if admin is able to login with valid credentials.
AIT-32	To test if admin is able to request new access token with existing refresh token.
AIT-33	To test if admin is able to request new access token with TourNepal app user's refresh token.
AIT-34	To test if admin is able to edit non-existing emergency contact number.
AIT-35	To test if admin is able to edit emergency contact without edit values.
AIT-36	To test if admin is able to edit contact with proper values.
AIT-37	To test if admin is able to delete contact with proper values
AIT-38	To test if admin is able to add a new contact with proper values.
AIT-39	To test if admin is able to get all users without valid access token.

AIT-40	To test if admin is able to retrieve all users with valid access token.
AIT-41	To test if admin is able to add a new user with valid access token
AIT-42	To test if admin is able to edit a user detail with valid access token.
AIT-43	To test if admin is able to delete a user.
AIT-44	To test if admin is able to get all reviews.
AIT-45	To test if admin is able to delete a review.
AIT-46	To test if user is able to logout successfully from the application.
AIT-47	To test if user is able to comment after the user is user status is verified from unverified.
AIT-48	To test if admin login is persistent after logging in.
AIT-49	To test if a proper message is shown when backend is not started.
AIT-50	To test if user is able to post a comment via API.
AIT-51	To test if user is able to edit a comment via API.

Table 3: API/ Integration test plan

4.1.3. System test plans

Test cases	Objectives
SYS-01	To test if user can provide invalid format for email during registration.
SYS-02	To test if user can continue without leaving any fields blank.
SYS-03	To test if user can register with non-matching password.
SYS-04	To test if user is able to register successfully with all credentials.
SYS-05	To test if user can register with an existing email.
SYS-06	To test if user can bypass document upload procedure.
SYS-07	To test if registered user is able to login with correct credentials.
SYS-08	To test if registered user can login with wrong password.
SYS-09	To test if unregistered user can login.
SYS-10	To test if user is able to login with empty fields.
SYS-11	To test if user stays logged after re-opening the application.
SYS-12	To test if user is able to view restaurants (default destination type) within 5 K.M (default radius).
SYS-13	To test if user is able to get newer restaurants when changing radius slider.

SYS-14	To test if user is able to view destinations when changing destination type.
SYS-15	To test if user is able to view new destinations after changing radius.
SYS-16	To test if user is able to search any places according without location and radius striction.
SYS-17	To test if user is able to search through filtered location from radius and destination type.
SYS-18	To test if user can navigate via google map of selected location.
SYS-19	To test if user can view description of selected restaurants.
SYS-20	To test if user can view description less popular for restaurants.
SYS-21	To test if user can view description for destinations.
SYS-22	To test if user can re-upload documents after verification rejected.
SYS-23	To test if verified user can leave a review in any place.
SYS-24	To test if unverified user can leave a review in any place.
SYS-25	To test if admin is able to login with valid credentials.
SYS-26	To test if admin can add a new user for mobile application.
SYS-27	To test if admin can verify/reject user document status.

SYS-28	To test if admin can send notification to specific/ all users.
---------------	--

Table 4: System test plans

4.2. Unit testing

4.2.1. Server start testing (UT-01)

Objective	To test if the server starts properly
Action	1) Opened terminal 2) Entered "npm run dev"
Expected result	Server should start properly in provided port number.
Actual result	Server did not start properly.
Conclusion	Test was not successful.

Table 5: Server starting test (Unsuccessful)

```

o rhythm@Rhythms-MacBook-Pro BACKEND % npm run dev

> backend@1.0.0 dev
> nodemon server

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
file:///Users/rhythm/Downloads/D/COLLEGE_FILES/FYP/FYP_PROJECT_FILES/FIRST_DRAFT/BACKEND/controllers/loginController.js:3
const jwt = require("jsonwebtoken");
            ^
ReferenceError: require is not defined
    at file:///Users/rhythm/Downloads/D/COLLEGE_FILES/FYP/FYP_PROJECT_FILES/FIRST_DRAFT/BACKEND/controllers/loginController.js:3:13
    at ModuleJobSync.runSync (node:internal/modules/esm/module_job:400:35)
    at ModuleLoader.importSyncForRequire (node:internal/modules/esm/loader:427:47)
    at loadESMFromCJS (node:internal/modules/cjs/loader:1565:24)
    at Module._compile (node:internal/modules/cjs/loader:1716:5)
    at Object..js (node:internal/modules/cjs/loader:1899:10)
    at Module.load (node:internal/modules/cjs/loader:1469:32)
    at Function._load (node:internal/modules/cjs/loader:1286:12)
    at TracingChannel.traceSync (node:diagnostics_channel:322:14)
    at wrapModuleLoad (node:internal/modules/cjs/loader:235:24)

Node.js v23.10.0
[nodemon] app crashed - waiting for file changes before starting...

```

Figure 47: Server not started

Reason: Since ES6 import was used throughout the server, commonjs import caused a proper for server to start.

```
import bcrypt from "bcrypt"
const jwt = require("jsonwebtoken");
const {encryptRefreshToken} = require('../api/refreshTokenEncryption')
const User = require('../model/User');

const handleLogin = async (req, res) => {
  const { email, password } = req.body;
  if (!email || !password)
    return res
      .status(400)
      .json({ message: "Username and password are required." });
}
```

Figure 48: Import error

Correction Measure: CommonJS import was changed to ES6 import.

```
const bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
const {encryptRefreshToken} = require('../api/refreshTokenEncryption')
const User = require('../model/User');

const handleLogin = async (req, res) => {
  const { email, password } = req.body;
  if (!email || !password)
    return res
      .status(400)
      .json({ message: "Username and password are required." });
}
```

Figure 49: Corrected import

Objective	To test if the server starts properly
Action	1) Opened terminal 2) Entered "npm run dev"
Expected result	Server should start properly in provided port number.
Actual result	Server did start properly in provided port number.
Conclusion	Test was successful.

Table 6: Server start test (successful)

4.2.2. MongoDB connection test (UT-02)

Objective	To check if database is being connected to the backend server
Action	1) Install mongoose package using npm 2) Specify database URI provided by MongoDB. 3) Run backend server
Expected result	Mongodb database should be connected on the provided database URI.
Actual result	Mongodb database was be connected on the provided database URI.
Conclusion	Test was successful.

Table 7: MongoDB connection test.

```
config > JS dbConn.js > [?] <unknown>
1  const mongoose = require('mongoose');
2
3  const connectDB = async () => {
4    try {
5      await mongoose.connect(process.env.DATABASE_URI, {
6        useUnifiedTopology: true,
7        useNewUrlParser: true
8      });
9    } catch (err) {
10     console.error(err);
11   }
12 }
13
14 module.exports = connectDB
```

Figure 50: MongoDB connection test.

```
JS server.js > ...
1  mongoose.connection.once('open', () => {
2    console.log('Connected to MongoDB');
3    app.listen(PORT, '0.0.0.0', () => console.log(`Server running on port ${PORT}`));
4  });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

○ rhythm@Rhythms-MacBook-Pro BACKEND % npm run dev

```
> backend@1.0.0 dev
> nodemon server

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
(node:6711) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has been removed in the next major version
(Use `node --trace-warnings ...` to show where the warning was created)
(node:6711) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has been removed in the next major version
Connected to MongoDB
Server running on port 3500
[]
```

Figure 51: MongoDB connection test 2

4.2.3. Login unit testing (UT-03)

Objective	To check if login logic is handled properly for every possible way
Action	Run login.test.js file with test cases defined.
Expected result	Should return with related code for different inputs.
Actual result	Returned with related code for different inputs.
Conclusion	Test was successful.

Table 8: Login unit testing

```

13 describe('Login Function', () => {
20   });
21
22 > test('should return error if email or password is missing', async () => { ...
26   });
27
28 > test('should return 401 if user not found', async () => { ...
40   });
41
42 > test('should return 403 if password does not match', async () => { ...
53   });
54
55 > test('should return access and refresh tokens on successful login', async () => { ...
83   });
84

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```

rhythm@Rhythms-MacBook-Pro BACKEND % npx jest testing_folder/login/login.test.js
PASS testing_folder/login/login.test.js
  Login Function
    ✓ should return error if email or password is missing (1 ms)
    ✓ should return 401 if user not found
    ✓ should return 403 if password does not match
    ✓ should return access and refresh tokens on successful login (1 ms)

Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:  0 total
Time:        0.396 s, estimated 1 s
Ran all test suites matching /testing_folder\/login\/login.test.js/i.
rhythm@Rhythms-MacBook-Pro BACKEND %

```

Figure 52: Login unit testing

4.2.4. User operations testing (UT-04)

Objective	To check if user related CRUD operations for possible inputs
Action	Run user.test.js file with test cases defined
Expected result	All the test should be passed with expected and toBe value matching.
Actual result	All the test was passed with expected and toBe value matching.
Conclusion	Test was successful.

Table 9: User operations testing

```

user.test.js
testing_folder > user-testing > user.test.js > ...
13 describe('User Controller', () => {
14
15   describe('addUser', () => {
16     > test('should create a new user successfully', async () => {
17       });
18
19     > test('should return error when user already exists', async () => {
20       });
21
22     > test('should return error when required fields are missing', async () => {
23       });
24
25     describe('editUser', () => {
26       > test('should update user successfully', async () => {
27         });
28
29       > test('should return error when user not found', async () => {
30         });
31     });
32
33     describe('deleteUser', () => {
34       > test('should delete user when requestType is true', async () => {
35         });
36
37       > test('should reject deletion when requestType is false', async () => {
38         });
39
40       > test('should return error when user not found', async () => {
41         });
42     });
43
44     describe('getAllUsers', () => {
45       > test('should return all users', async () => {
46         });
47
48       > test('should handle errors', async () => {
49         });
50     });
51
52     describe('getUserById', () => {
53       > test('should return user by ID', async () => {
54         });
55
56       > test('should return error when user not found', async () => {
57         });
58     });
59   });
60 });

```

Figure 53: Test cases of user actions

```
● rhythm@Rhythms-MacBook-Pro BACKEND % npx jest testing_folder/user-testing/user.test.js
PASS testing_folder/user-testing/user.test.js
  User Controller
    addUser
      ✓ should create a new user successfully (1 ms)
      ✓ should return error when user already exists (1 ms)
      ✓ should return error when required fields are missing
    editUser
      ✓ should update user successfully
      ✓ should return error when user not found (1 ms)
    deleteUser
      ✓ should delete user when requestType is true
      ✓ should reject deletion when requestType is false
      ✓ should return error when user not found
    getAllUsers
      ✓ should return all users (1 ms)
      ✓ should handle errors
    getUserById
      ✓ should return user by ID
      ✓ should return error when user not found
    getNotificationTokens
      ✓ should return notification tokens

Test Suites: 1 passed, 1 total
Tests:       13 passed, 13 total
Snapshots:   0 total
Time:        0.386 s, estimated 1 s
Ran all test suites matching /testing_folder/user-testing/user.test.js/i.
```

Figure 54: Unit test results for user actions

4.2.5. Registration unit testing (UT-05)

Objective	To check if register logic is handled properly for every possible way
Action	Run register.test.js file with test cases defined.
Expected result	Should return with related code for different inputs.
Actual result	Returned with related code for different inputs.
Conclusion	Test was successful.

Table 10: Registration unit testing

```

register.test.js
testing_folder > register > register.test.js > ...
8 describe('handleRegister', () => {
25
26 > test('should return 400 if any field is missing', async () => { ...
30 });
31
32 > test('should return 409 if email already exists', async () => { ...
39 });
40
41 > test('should return 201 and create new user if all data is valid', async () => { -
53 });
54
55 > test('should return 500 if an error occurs', async () => { ...
63 });
64 });
65

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
rhythm@Rhythms-MacBook-Pro BACKEND % npx jest testing_folder/register/register.test.js
PASS testing_folder/register/register.test.js
  handleRegister
    ✓ should return 400 if any field is missing
    ✓ should return 409 if email already exists
    ✓ should return 201 and create new user if all data is valid
    ✓ should return 500 if an error occurs

Test Suites: 1 passed, 1 total
Tests: 4 passed, 4 total
Snapshots: 0 total
Time: 0.219 s, estimated 1 s
Ran all test suites matching /testing_folder/register/register.test.js/i.
rhythm@Rhythms-MacBook-Pro BACKEND %
  
```

Figure 55: Register unit testing

4.2.6. Emergency contact operations unit testing (UT-06)

Objective	To check if emergency contact related CRUD operations for possible inputs
Action	Run emergency-contacts.test.js file with test cases defined
Expected result	All the test should be passed with expected and toBe value matching.
Actual result	All the test was passed with expected and toBe value matching.
Conclusion	Test was successful.

Table 11: Emergency contact operation unit testing

```

7   describe('Emergency Contacts Controller', () => {
10  >   test('should add a new contact', async () => {
23     });
24
25  >   test('should return error if contact already exists', async () => {
36     });
37
38
39  >   test('should update a contact successfully', async () => {
53     });
54
55  >   test('should return error if contact not found for update', async () => {
66     });
67
68
69  >   test('should delete a contact successfully', async () => {
79     });
80
81  >   test('should return error if contact not found for deletion', async () => {
91     });
92   });
93 }

```

```

rhythm@Rhythms-MacBook-Pro BACKEND % npx jest testing_folder/emergency-contacts/emergency-contacts.test.js
PASS testing_folder/emergency-contacts/emergency-contacts.test.js
  Emergency Contacts Controller
    ✓ should add a new contact (1 ms)
    ✓ should return error if contact already exists
    ✓ should update a contact successfully
    ✓ should return error if contact not found for update
    ✓ should delete a contact successfully (1 ms)
    ✓ should return error if contact not found for deletion
  Test Suites: 1 passed, 1 total
  Tests: 6 passed, 6 total
  Snapshots: 0 total
  Time: 0.197 s, estimated 1 s
  Ran all test suites matching /testing_folder/emergency-contacts/emergency-contacts.test.js/i.
rhythm@Rhythms-MacBook-Pro BACKEND %

```

Figure 56: Emergency contacts unit testing

4.2.7. Location retrieval testing (UT-07)

Objective	To check if the location of the user is being retrieved from the mobile phone
Action	1) Request using Android location permission for android phone. 2) Install "react-native-get-location" package using npm 3) Use react-native-get-location method "getLocation" to get current location to. 4) Run it when the application is opened. 5) Log it to the console. 6) Run the application. 7) Enable permission if not granted.
Expected result	User exact location should be displayed on the console.
Actual result	User exact location was displayed on the console.
Conclusion	Test was successful.

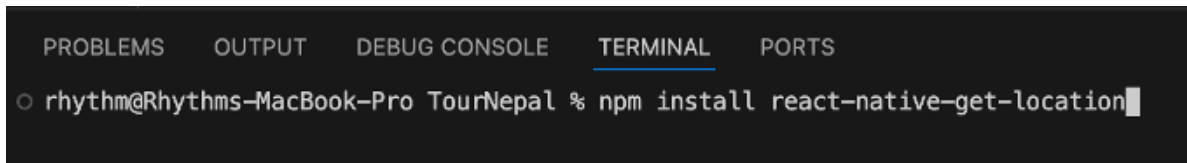
Table 12: User location retrieval testing

```

39 //requesting location permission
40 const getLocationAccess = async () => {
41   const granted = await PermissionsAndroid.request(
42     PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION,
43   ).catch(err => {
44     console.log(err);
45   });
46   return granted === PermissionsAndroid.RESULTS.GRANTED;
47 };
48

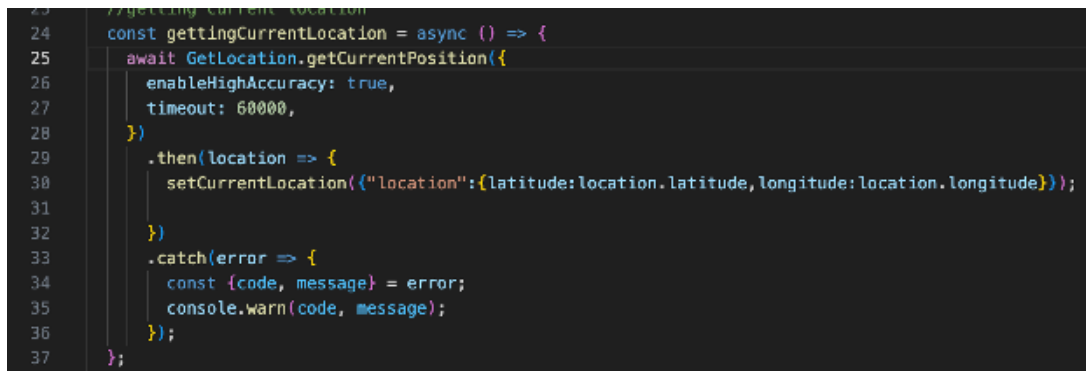
```

Figure 57: Requesting user location



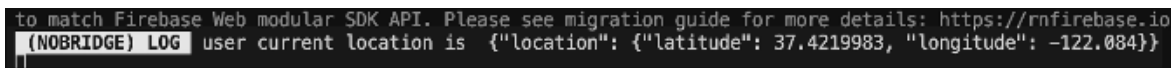
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
○ rhythm@Rhythms-MacBook-Pro TourNepal % npm install react-native-get-location
```

Figure 58: Installing react-native-get-location package



```
23 //getting current location
24 const gettingCurrentLocation = async () => {
25   await GetLocation.getCurrentPosition({
26     enableHighAccuracy: true,
27     timeout: 60000,
28   })
29   .then(location => {
30     setCurrentLocation({location:{latitude:location.latitude,longitude:location.longitude}});
31   })
32   .catch(error => {
33     const {code, message} = error;
34     console.warn(code, message);
35   });
36 };
```

Figure 59: Getting user current location



```
to match Firebase Web modular SDK API. Please see migration guide for more details: https://rnfirebase.io
(NOBRIDGE) LOG user current location is {"location": {"latitude": 37.4219983, "longitude": -122.084}}
```

Figure 60: User current location

4.2.8. Request for notification access testing (UT-08)

Objective	To check if the notification permission is requested upon login if not already granted
Action	1) Request Android push notification permission for android phone after login. 2) Log it to the console after user logs in. 3) Run the application and login.
Expected result	User should be prompted with notification access request.
Actual result	User was prompted with notification access request.
Conclusion	Test was successful.

Table 13: Request for notification access testing

```

16 export const androidNotificationToken = async()=>{
17     let token;
18     const requestUserPermission = async () => {
19         try {
20             const granted = await PermissionsAndroid.request(PermissionsAndroid.PERMISSIONS.POST_NOTIFICATIONS);
21             if (granted === PermissionsAndroid.RESULTS.GRANTED) {
22                 console.log("Notification permission granted");
23             } else {
24                 console.log("Notification permission denied");
25             }
26         }

```

Figure 61: Android permission for post notification

```

const userData = await loginUser(email,password)
if(userData && userData?.data.accessToken){
    //storing tokens securely
    await storeTokens(userData.data.accessToken,userData.data.encryptedToken)
    const details = await getUserDetail();
    setCurrUser(details.data); //setting the user session if the token is validated

    const notificationToken = await androidNotificationToken();
}

```

Figure 62: Requesting notification access after login

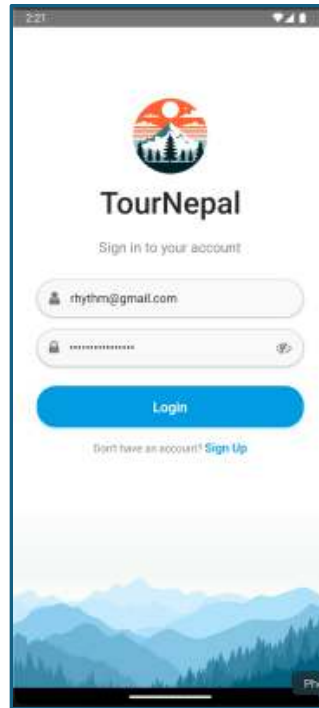


Figure 63: Logging in (notification test)

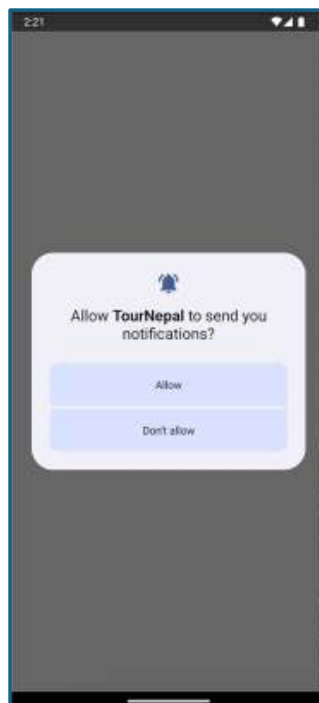
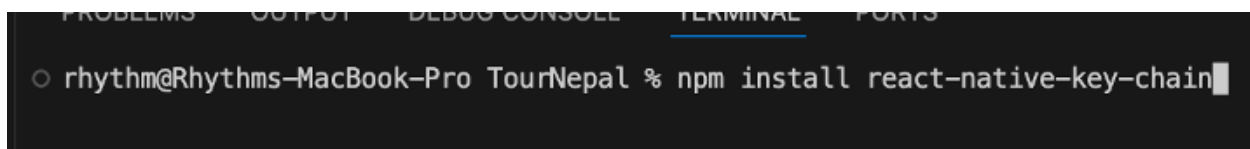


Figure 64: Notification request prompt

4.2.9. Token storage in mobile device testing (UT-09)

Objective	To check if the tokens are being saved in android device
Action	1) Install react-native-keychain package 2) Save tokens securely to the mobile device after login. 3) Retrieve and log the tokens after opening the application. 4) Login with valid credentials. 5) Close the application and reopen the application. 6) Check the console
Expected result	Tokens should be logged to the console.
Actual result	Tokens was logged to the console.
Conclusion	Test was successful.

Table 14: Token storage in mobile device testing



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
○ rhythm@Rhythms-MacBook-Pro TourNepal % npm install react-native-key-chain

```

Figure 65: Installing react-native-keychain

```

import * as Keychain from 'react-native-keychain'; //importing keychain

//for storing token in mobile device
export const storeTokens = async (accessToken, encryptedToken) => {
  try {
    //for storing access token and encrypted token in dictionary
    const authTokens = {accessToken, encryptedToken};
    await Keychain.setGenericPassword('auth', JSON.stringify(authTokens), {
      service: 'TourNepalTokens',
    });
  } catch (e) {
    console.error('Error storing tokens: ', e);
  }
};

//for retrieving tokens in mobile device
export const getToken = async () => {
  try {
    const authTokens = await Keychain.getGenericPassword({
      service: 'TourNepalTokens',
    });
    if (authTokens) {
      return JSON.parse(authTokens.password);
    }
    return "1st";
  } catch (e) {
    console.error('Error retrieving tokens: ', e);
    return "2nd";
  }
};

```

Figure 66: Storing/retrieving tokens code snippet

```

const userData = await loginUser(email,password)
if(userData && userData?.data.accessToken){
  //storing tokens securely
  await storeTokens(userData.data.accessToken,userData.data.encryptedToken)
  const details = await getUserDetail();
  setCurrUser(details.data); //setting the user session if the token is validated
}

```

Figure 67: Storing token after login

```

const checkToken = async () => {
  setIsLoading(true);
  try{
    const tokens = await getToken();
    if (tokens && tokens.encryptedToken && tokens.accessToken) {
      console.log(tokens)
    }
  }
}

```

Figure 68: Application initialization check for tokens

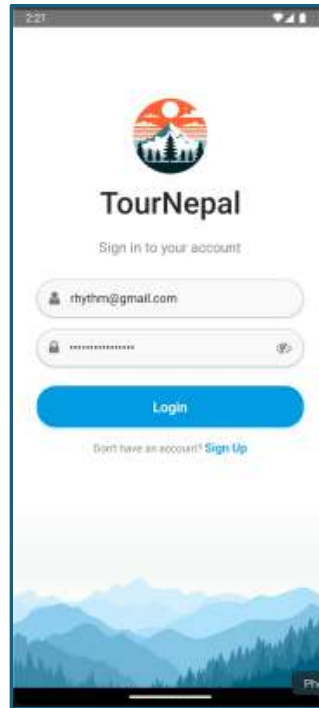


Figure 69: Logging in (token storage test)

[illegible]

Figure 70: Displaying token in console

4.2.10. Gemini place description generation testing (UT-10)

Objective	To test is gemini is able to generate a description of place
Action	Run geminiDescription.test.js file with a valid place name passed.
Expected result	Test should be passed with a description printed on the console.
Actual result	Test was passed with a description printed on the console.
Conclusion	Test was successful.

Table 15: Gemini place description generation testing

```

1  const { returnDescription } = require('./geminiDescription');
2
3  test('Gemini Place Description', async () => {
4
5      const result = await returnDescription('Patan Durbar');
6      expect(result).toBeDefined();
7
8
9      expect(result).toBeTruthly();
10
11
12      expect(typeof result).toBe('string');
13      expect(result.length).toBeGreaterThan(0);
14
15      expect(result).toContain('Durbar');
16  }, 15000);

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

rhythm@Rhythm-MacBook-Pro BACKEND % npx jest testing_folder/gemini-test/geminiDescription.test.js
console.log
Patan Durbar Square is situated in the heart of Lalitpur city, Nepal. This historic plaza served as the palace complex for the Malla kings of Lalitpur and is celebrated as a UNESCO World Heritage Site, showcasing exceptional Newari architecture. Key highlights include the intricately carved Krishna Mandir, the main courtyard Mul Chowk, and the Patan Museum housed within the former palace.
Pricing Details (SAARC Nations): NPR 250
Pricing Details (Foreigners): NPR 1000
    at log (testing_folder/gemini-test/geminiDescription.js:98:13)
PASS testing_folder/gemini-test/geminiDescription.test.js (10.299 s)
  ✓ Gemini Place Description (10150 ms)
Test Suites: 1 passed, 1 total
Tests:      1 passed, 1 total
Snapshots: 0 total
Time:       10.232 s
Ran all test suites matching /testing_folder/gemini-test/geminiDescription.test.js/i.
rhythm@Rhythm-MacBook-Pro BACKEND %

```

Figure 71: Gemini description test

4.2.11. Gemini invalid place description generation testing (UT-11)

Objective	To test is gemini is able to generate a proper output for non-exisiting place
Action	Run geminiDescription.test.js file with a invalid place name passed.
Expected result	Test should be passed with a proper output printed on the console.
Actual result	Test was passed with a proper output printed on the console.
Conclusion	Test was successful.

Table 16: Gemini invalid place description generation testing

```

testing_folder > gemini-test > geminiDescription.test.js > test('Gemini Place Description') callback
1  const { returnDescription } = require('./geminiDescription');
2
3  test('Gemini Place Description', async () => {
4
5      1
6      const result = await returnDescription('asdf dfwer');
7      expect(result).toBeUndefined();
8
9
10     expect(result).toBeTruthy();
11
12     expect(typeof result).toBe('string');
13     expect(result.length).toBeGreaterThan(0);
14
15
16
17
18 }, 15000);

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

● rhythm@Rhythms-MacBook-Pro BACKEND % npx jest testing_folder/gemini-test/geminiDescription.test.js
console.log
Detailed information is currently unavailable.
    at log (testing_folder/gemini-test/geminiDescription.js:98:15)
2
PASS testing_folder/gemini-test/geminiDescription.test.js (5.277 s)
  ✓ Gemini Place Description (5127 ms)

Test Suites: 1 passed, 1 total
Tests: 1 passed, 1 total
Snapshots: 0 total
Time: 5.389 s
Ran all test suites matching /testing_folder/gemini-test/geminiDescription.test.js/i.
○ rhythm@Rhythms-MacBook-Pro BACKEND %

```

Figure 72: Gemini invalid place description test

4.2.12. Password hashing testing (UT-12)

Objective	To check if the password are being properly hashed and matched
Action	1) Install bcrypt package 2) Run encryption.test.js file with a word passed in function to check.
Expected result	Password should be hashed and hashed password and initial word should match
Actual result	Password was be hashed and hashed password and initial word was matched.
Conclusion	Test was successful.

Table 17: Password hashing testing

```
rhythm@Rhythms-MacBook-Pro BACKEND % npm install bcrypt
```

Figure 73: Bcrypt installation

```
testing_folder > password-encryption-matching > encryption.test.js > test('Checking if password being hashed properly') callback > result
1  const {checkPass,comparePass} = require('./encryption')
2
3  test('Checking if password being hashed properly', async () => {
4    const result = await checkPass('random')
5    expect(result).toBeDefined();
6    console.log(result)
7  },15000);
8
9  test('Checking if password matching is working as expected for login/registration purpose',async ()=>{
10
11    const result = await checkPass('random')
12    const match = await comparePass('random',result)
13
14    expect(match).toBeDefined();
15    expect(match).toBe(true);
16
17  })
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
rhythm@Rhythms-MacBook-Pro BACKEND % npx jest testing_folder/password-encryption-matching/encryption.test.js
console.log
$2b$10$Q63WfEX2f8MS2AK4obJee8V.B9J2Pj7g0Mq7T5FWkVEUijCITz50
at Object.log (testing_folder/password-encryption-matching/encryption.test.js:6:11)
PASS testing_folder/password-encryption-matching/encryption.test.js
  ✓ Checking if password being hashed properly (53 ms)
  ✓ Checking if password matching is working as expected for login/registration purpose (90 ms)
Test Suites: 1 passed, 1 total
Tests: 2 passed, 2 total
Snapshots: 0 total
Time: 0.284 s, estimated 1 s
Run all test suites matching /testing_folder/password-encryption-matching/encryption.test.js/i.
rhythm@Rhythms-MacBook-Pro BACKEND %
```

Figure 74: Password comparison unit test

4.3. API/Integration testing

Thunderclient/Postman was used for API testing. Thunderclient extension was used in VS Code for easy access.

4.3.1. Authentication testing

For authentication backend server should be running.

1) Empty body login testing (AIT-01)

Objective	To test if the server respond with proper message when no body is provided for login.
Action	1) Change method to POST method. 2) On body section no json was be passed. 3) For URL http://localhost:3500/login was put. 4) Request was sent.
Expected result	Server should respond with proper error code and message.
Actual result	Server responded with proper error code and message.
Conclusion	Test was successful.

Table 18: Empty body login test

2) Successful login testing (AIT-02)

Objective	To test if access token and encrypted token are sent after successful login.
Action	1) Method was changed to POST method. On body valid credentials was passed. <pre>{ "email": "rhythm@gmail.com", "password": "rhythm@gmail.com" }</pre> 2) For URL http://localhost:3500/login was put. 3) Request was sent.
Expected result	Server should respond with access token and encrypted refresh token with 200 status code.
Actual result	Server responded with access token and encrypted refresh token with 200 status code.
Conclusion	Test was successful.

Table 19: Successful login testing

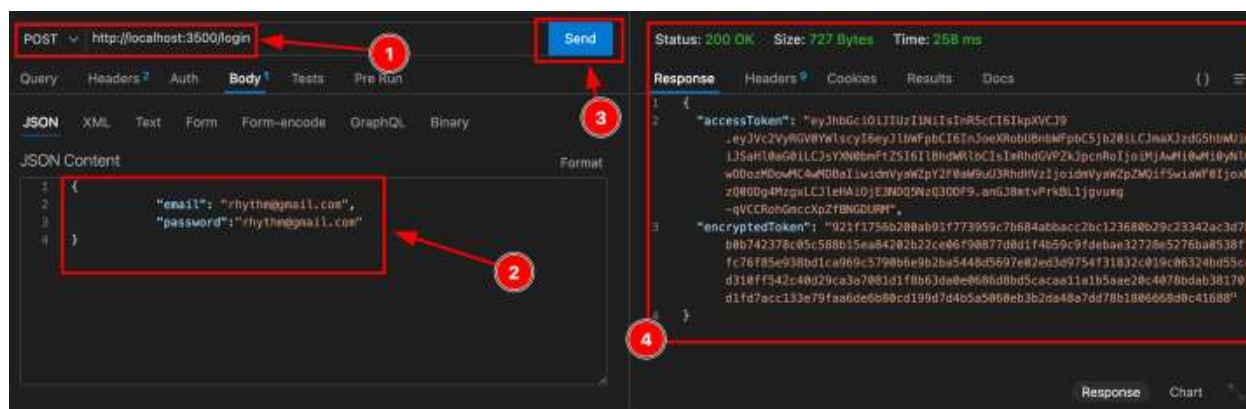


Figure 75: Valid credentials login test

3) AIT-03 Incomplete body during registration

Objective	To test if the server respond with proper message when improper body is provided during registration process.
Action	1) Method was changed to POST method. 2) On body email and password was skipped. { "firstname": "rhythm", "lastname": "paudel", "dateOfBirth": "2004-09-24", "visaStamp": "xys", "passportCopy": "passportCopy", "dateOfEntry": "2004-09-24", "nationality": "NP", "intendedDays": "3" } 3) For URL http://localhost:3500/register was put. 4) Request was sent.
Expected result	Server should respond with access token and encrypted refresh token with 200 status code.
Actual result	Server responded with access token and encrypted refresh token with 200 status code.
Conclusion	Test was successful.

Table 20: Incomplete body registration

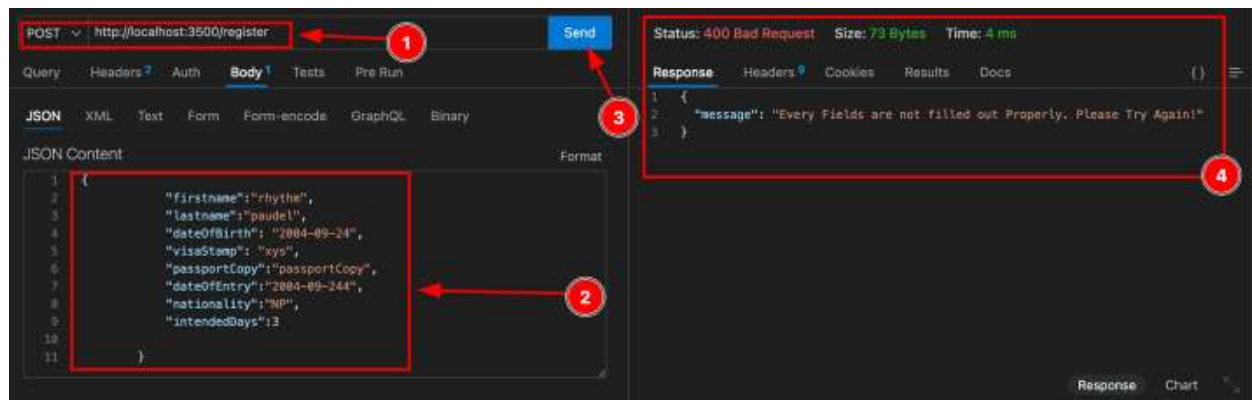


Figure 76: Incomplete body registration

4) AIT-04 False value handling during registration

Objective	To test if server handles false values during registration process.
Action	1) Method was changed to POST method. 2) On body email and password was given numeric value. { "email": 123, "password": 123, "firstname": "rhythm", "lastname": "paudel", "dateOfBirth": "2004-09-24", "visaStamp": "xys", "passportCopy": "passportCopy", "dateOfEntry": "2004-09-24", "nationality": "NP", "intendedDays": 3 } 3) For URL http://localhost:3500/register was put. 4) Request was sent.
Expected result	Server should respond with proper error code and message.
Actual result	Server responded with proper error code and message.
Conclusion	Test was successful.

Table 21: Registration false value handling

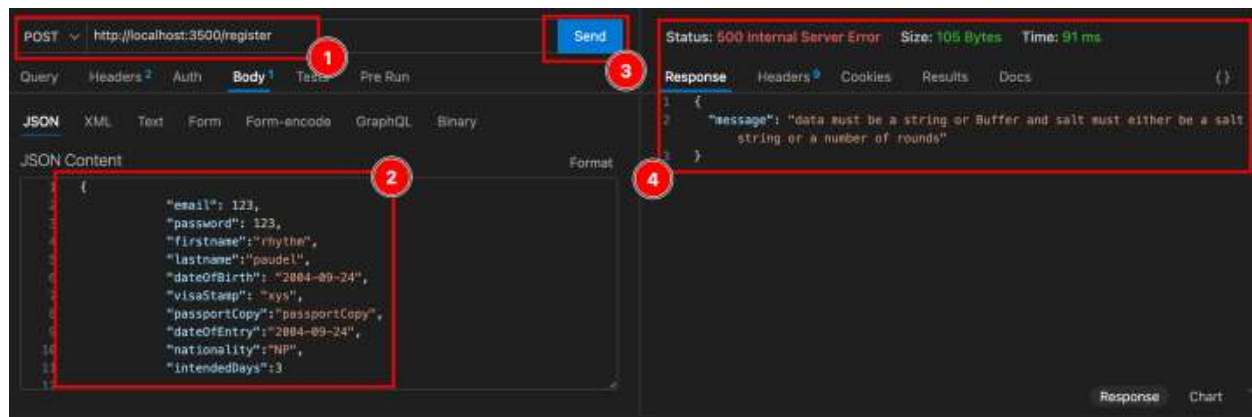


Figure 77: Registration false value handling

5) AIT-05 Registration completion testing

Objective	To test if registration process is successful with all the fields in proper format.
Action	1) Method was changed to POST method. 2) On body a proper json body with filled fields was passed. { "email": "islingtoncollege@gmail.com", "password": "islingtoncollege@gmail.com", "firstname":"rhythm", "lastname":"paudel", "dateOfBirth": "2004-09-24", "visaStamp": "xys", "passportCopy":"passportCopy", "dateOfEntry":"2004-09-24", "nationality":"NP", "intendedDays":3 } 3) For http://localhost:3500/register URL was put. 4) Request was sent.
Expected result	A new user should be registered successfully and proper response should be sent.
Actual result	A new user was registered and proper response was sent.
Conclusion	Test was successful.

Table 22: Registration completion testing

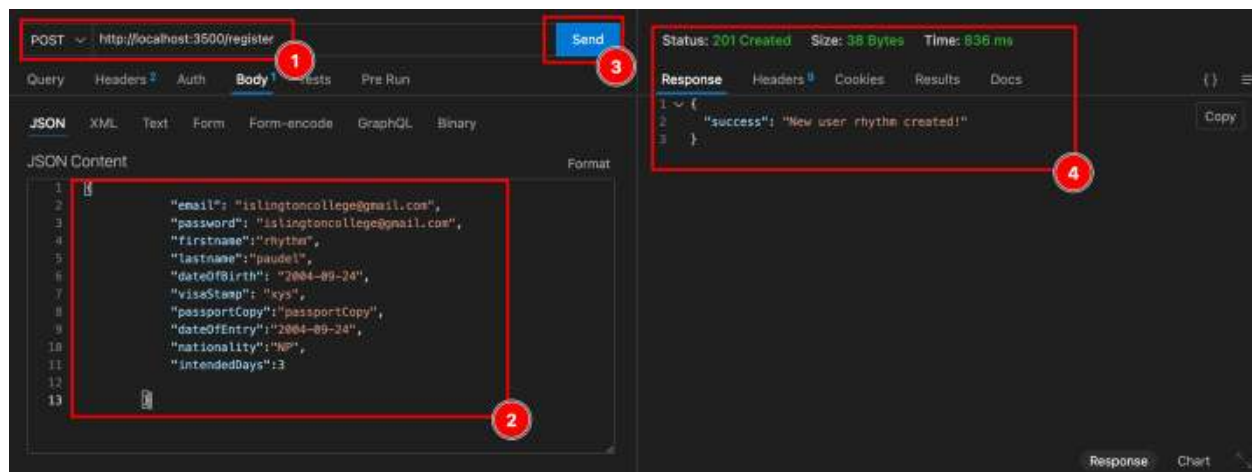


Figure 78: User creation API testing



Figure 79: User creation in mongoDB

6) AIT-06 Requesting access token with valid refresh token

Objective	To test if the user is able to get a new access token with existing refresh token.
Action	1) Method was changed to POST method. 2) Encrypted refresh token was generated by logging in with valid email and password (http://localhost:3500/login). 3) Encrypted refresh token was passed in body. 4) URL was changed to http://localhost:3500/refresh 5) Request was sent
Expected result	A new access token should be generated and sent as a response.
Actual result	A new access token was generated and sent as a response.
Conclusion	Test was successful.

Table 23: Requesting access token with refresh token

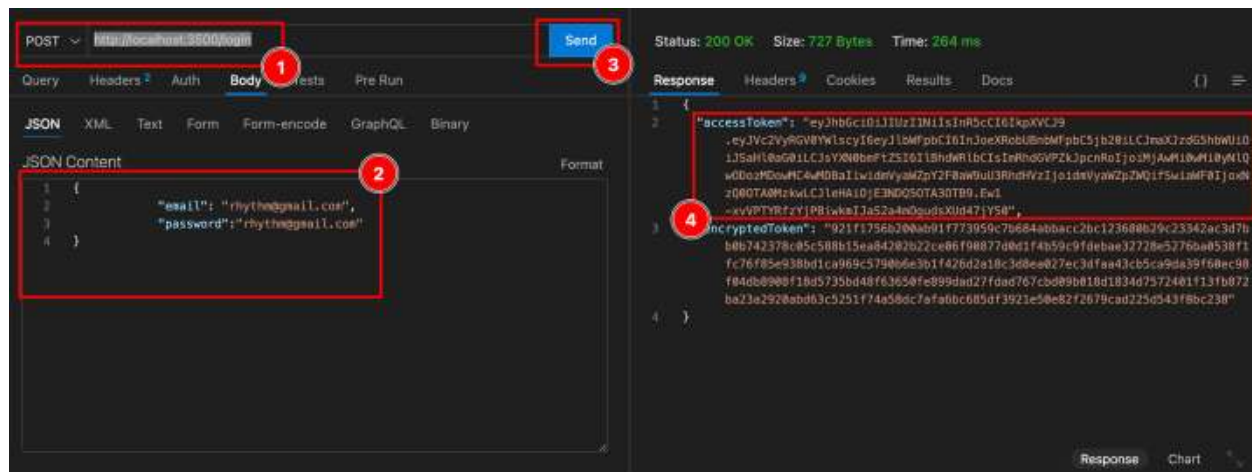


Figure 80: Logging in for refresh token

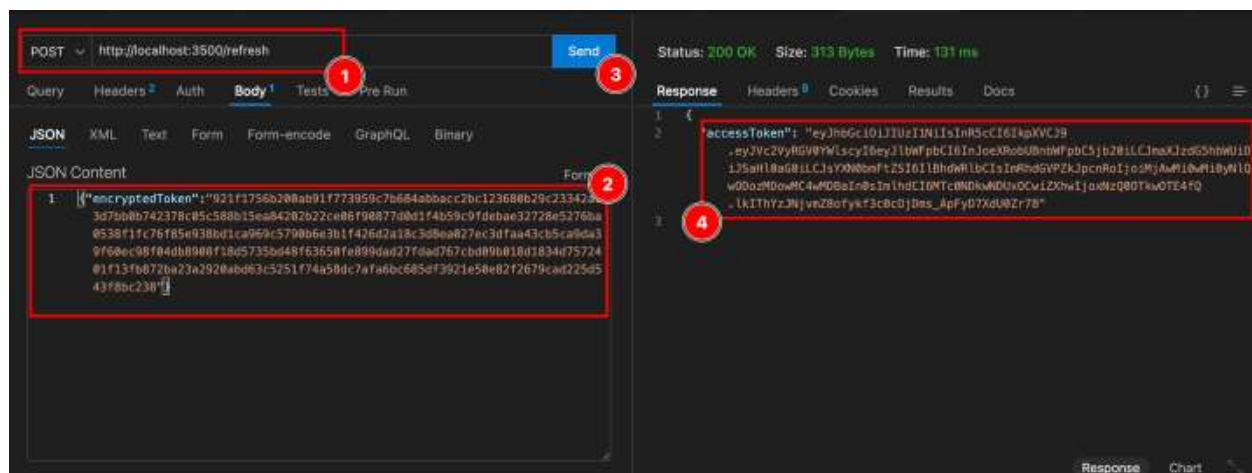


Figure 81: Requesting new access token

7) AIT-07 Requesting access token with expired refresh token

Objective	To test if the user can get a new access token with expired refresh token.
Action Add-ins	1) Method was changed to POST method. 2) Token span was kept 10 seconds 3) Encrypted token was generated by logging in with valid email and password (http://localhost:3500/login). 4) URL was changed to http://localhost:3500/refresh 5) Encrypted refresh token was passed in body after 10s. 6) Request was sent
Expected result	A proper message with error code should be sent as a response.
Actual result	A proper message with error code was sent as a response
Conclusion	Test was successful.

Table 24: Requesting access token with expired refresh token

```
const refreshToken = jwt.sign({
  |   "email": userDB.email
},
  process.env.REFRESH_SECRET_TOKEN,
  {expiresIn: '10s'}
);
```

Figure 82: Refresh token span

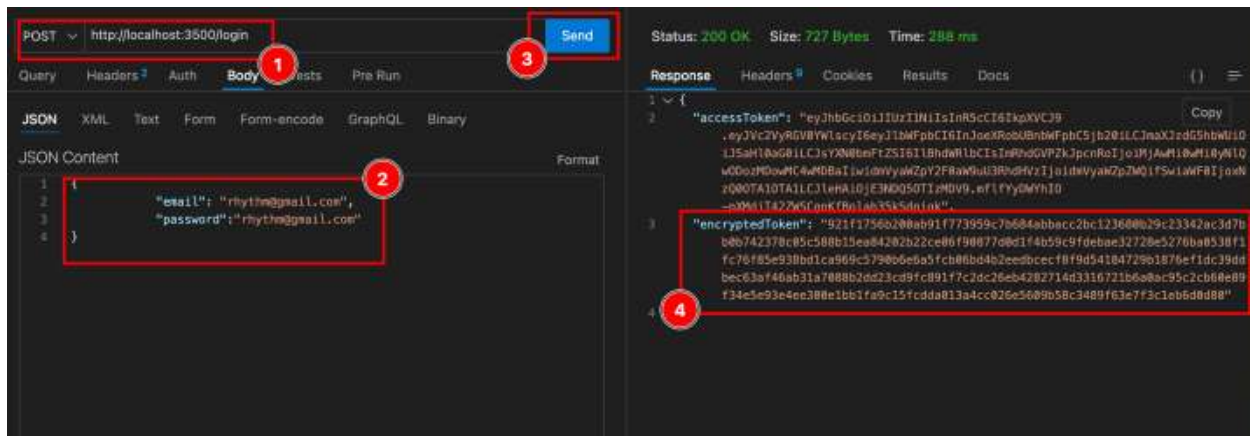


Figure 83: Encrypted refresh token retrieval

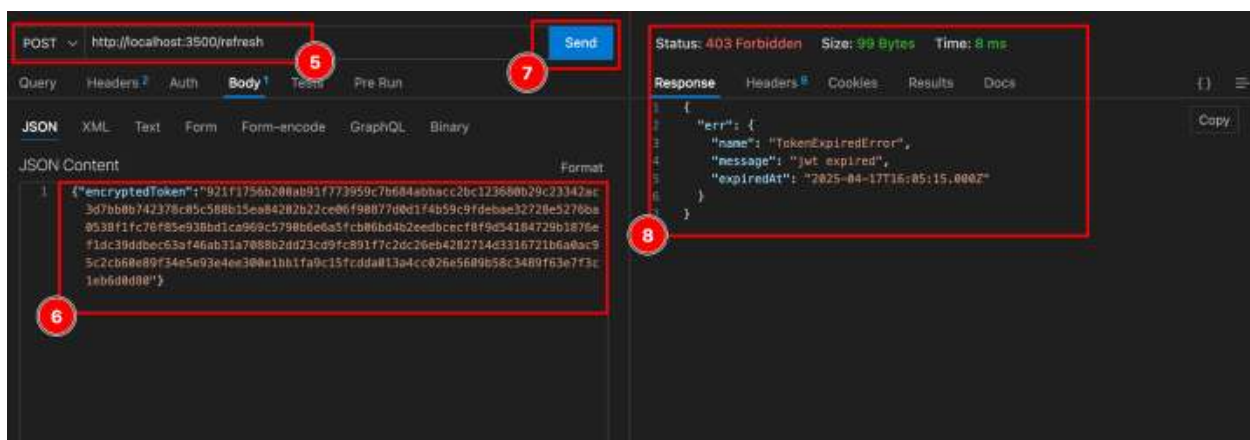


Figure 84: Requesting access token with expired refresh token

8) AIT-08 Token request without refresh token

Objective	To test if the server responds with proper status code if access token is requested without refresh token.
Action	1) Method was changed to POST method. 2) URL was set to http://localhost:3500/refresh 3) Request was sent without body
Expected result	A proper message with error code should be sent as a response.
Actual result	A proper message with error code was sent as a response
Conclusion	Test was successful.

Table 25: Token request without refresh token

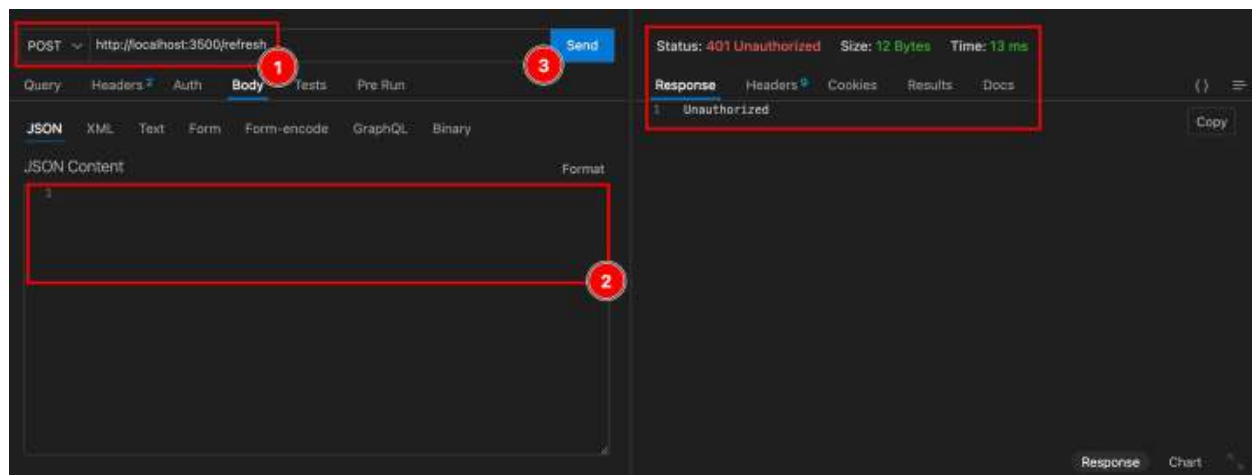


Figure 85: Token request without refresh token

4.3.2. Emergency contact fetch testing

AIT-09 Retrieving emergency contacts

Objective	To test if user/admin can retrieve emergency contacts.
Action	1) Method is set to get method 2) URL was set to http://localhost:3500/emergencycontacts 3) Request was sent.
Expected result	List of emergency contacts should be sent as a response.
Actual result	List of emergency contacts was sent as a response,
Conclusion	Test was successful.

Table 26: Retrieving emergency contacts

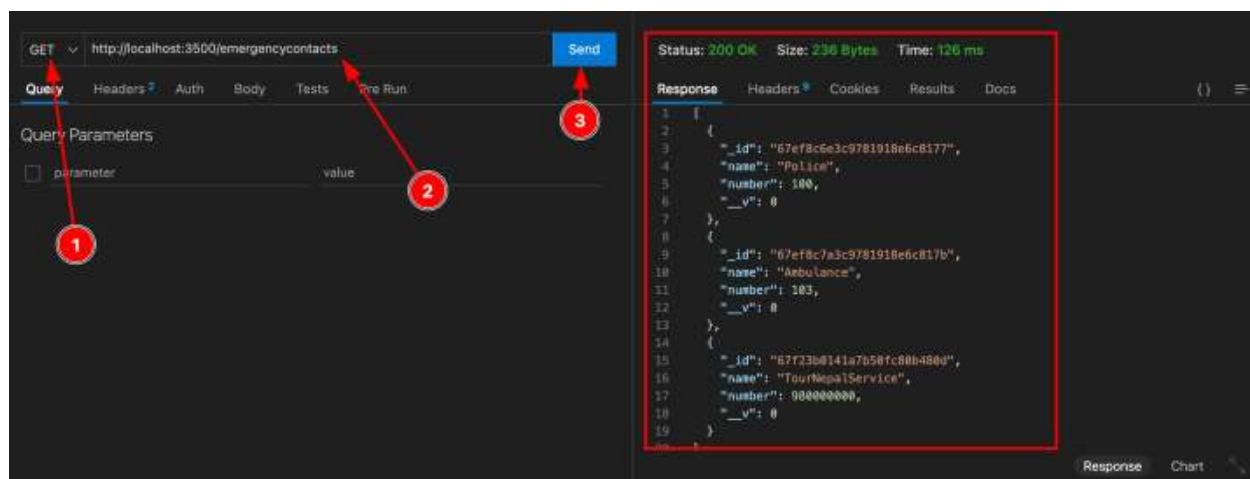


Figure 86: Getting contacts list

4.3.3. Map/Location testing

1) AIT-10 Google map API initial test

Objective	To test if google maps API is working as expected.
Action	1) Method is set to get method 2) URL was set to 'https://maps.googleapis.com/maps/api/place/nearbysearch/json' 3) Query Parameters were set Keyword=tourist_attraction Location= 27.6727,85.3253 Radius=5000 Type=tourist_attraction Key=GOOGLE_API_KEY 4) Request was sent
Expected result	List of nearby places should be retrieved from google maps.
Actual result	List of nearby places was retrieved from google maps.
Conclusion	Test was successful.

Table 27: Google map API test

The screenshot displays a REST client interface with a GET request to the Google Maps API. The request URL is `https://maps.googleapis.com/maps/api/place/nearbysearch/json?se...`. The query parameters are as follows:

parameter	value
keyword	tourist_attraction
location	27.6727,85.3253
radius	5000
type	tourist_attraction
key	AIzaSyA970HUZnqyE5O3C1kj14W6iUC

The response is a JSON object with the following structure:

```
{
  "icon": "https://maps.gstatic.com/mapfiles/place_api/icons/v1/png_71/generic_business-71.png",
  "icon_background_color": "#1385C7",
  "icon_mask_base_uri": "https://maps.gstatic.com/mapfiles/place_api/icons/v2/generic_nin1el",
  "name": "Garden of Dreams",
  "opening_hours": {
    "open_now": false
  },
  "photos": [
    {
      "height": 3824,
      "html_attributions": [
        "<a href='\"https://maps.google.com/maps/contrib/118680994823359126486/'>Cecilia Barantani</a>"
      ],
      "photo_reference": "AeeoncKPPbJ5M1vAwvTg3dbCXj8ZaJI0A9tjH0kswrJ9
      -ZPTUswr0gLBgidj3yi08oBapsmJ3K0byd9lkhBHj_1IVJSHLM
      -lw_Hpv21uPOAfDx85v_e_0s0kt9seuWxzcl4hxTVz2WUkxI278ch5
      -hAf02gMYUBANSduu11NB
      -nt70guhBoZUfOP4hAo8jjo0DeZasEPfJgnPkQCjWk04Gn2V8hOfz0DXBfuv1GJ
      -xEtqnVhD1LhMbPav---cpKB6SvLXGRMjMKVorNeeN4nzB2E33T
      -xps8Z77H5Y3gvumrPq1I",
      "width": 4832
    }
  ],
  "place_id": "ChIJi-BVIAIZ6zkRag6h8vq9K_U",
  "rating": 4.2,
  "reference": "ChIJi-BVIAIZ6zkRag6h8vq9K_U",
  "scope": "GOOGLE",
  "types": [

```

Figure 87: Google map API data response

2) AIT-11 Requesting destinations without location

Objective	To test if server responds with proper error code if no location is sent while requesting destinations.
Action	1) Method was set to set request. 2) URL was set to http://localhost:3500/location 3) Radius and destination type was set in query section, however location was not set (not checked). 4) Request was sent.
Expected result	A proper message should be received with proper error code.
Actual result	A proper message for no location was sent as a response.
Conclusion	Test was successful.

Table 28: Requesting destinations without location

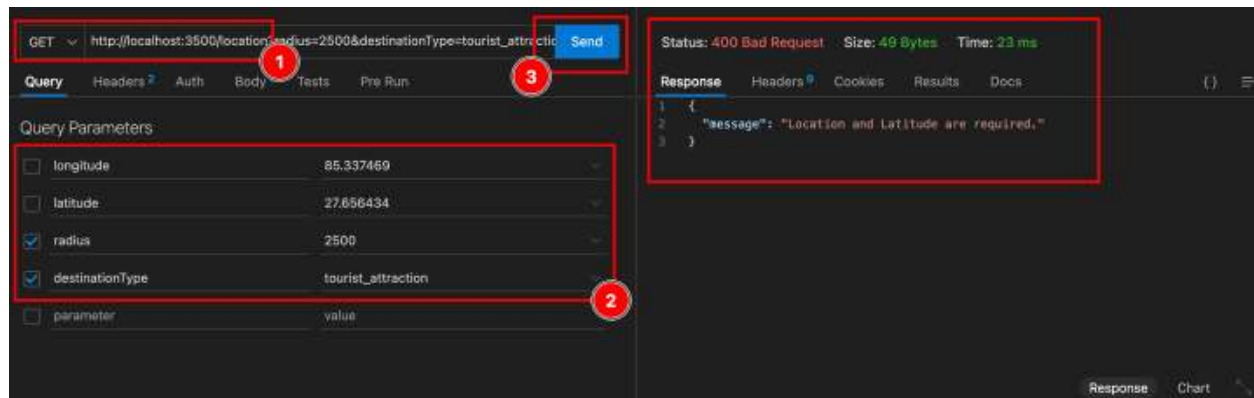


Figure 88: Requesting destinations without location

3) AIT-12 Missing destination type for place request

Objective	To test if server responds properly if no destination type is provided (restaurants or tourist attraction).
Action	1) Method was set to set request. 2) URL was set to http://localhost:3500/location 3) Query parameters was provided without destination_type "latitude": 27.65234, "longitude": 85.0909 "radius": 2500 4) Request was sent.
Expected result	The server should return response with proper error code and message.
Actual result	The server returned with 200 status code which was Okay status code.
Conclusion	Test was not succesful.

Table 29: Missing destination type for places(unsuccesful)

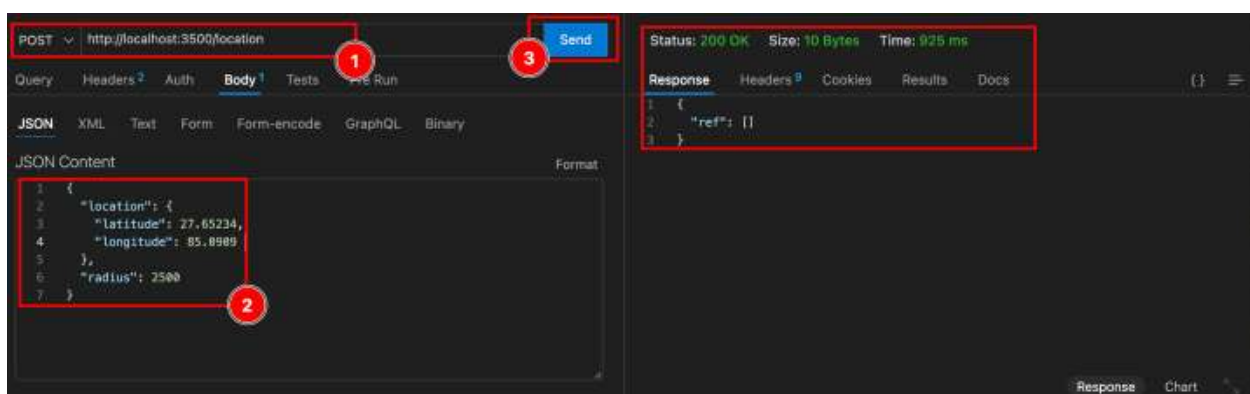


Figure 89: Response without destination type

Reason: Since, destination type was not checked if empty, the code was going in catch block as destination_type was undefined.

Correction measure: Destination type was checked for null values before place execution.

```
const {latitude,longitude,radius,destinationType} = req.query
const decodedRadius = radius
console.log(radius);

if(!latitude || !longitude || !radius) return res.status(400).json({ message: "Location and Latitude are required." });
if(!destinationType) return res.status(400).json({message:"Destination type is required for location fetching."})
```

Figure 90: Corrected code snippet for destination type check

Objective	To test if server responds properly if no destination type is provided (restaurants or tourist attraction).
Action	1) Method was set to set request. 2) URL was set to http://localhost:3500/location 3) Query parameters was provided without destination_type "latitude": 27.65234, "longitude": 85.0909 "radius": 2500 4) Request was sent.
Expected result	The server should return response with proper error code and message.
Actual result	The server returned response with proper error code and message.
Conclusion	Test was succesfull.

Table 30: Missing destination type for place request(successful)

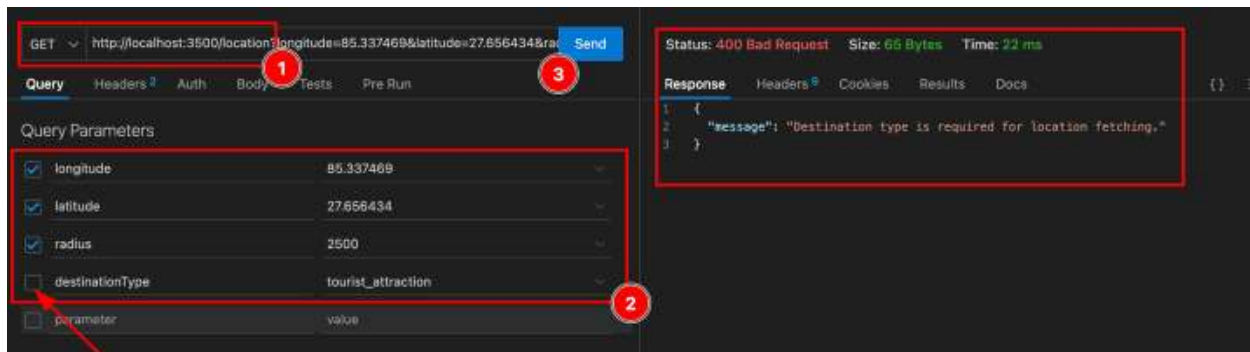


Figure 91: Place fetching without destination type

4) AIT-13 Location fetching with valid information

Objective	To test if places are retrieved with valid location and radius.
Action	1) Method was set to get request. 2) All the required parameters were passed. longitude=85.316379 latitude=27.677345 radius=2500 destinationType=tourist_attraction 3) Request was sent
Expected result	The server should return list of places along with base64 image.
Actual result	The server returned list of places along with base64 image.
Conclusion	Test was successful.

Table 31: Location fetch testing

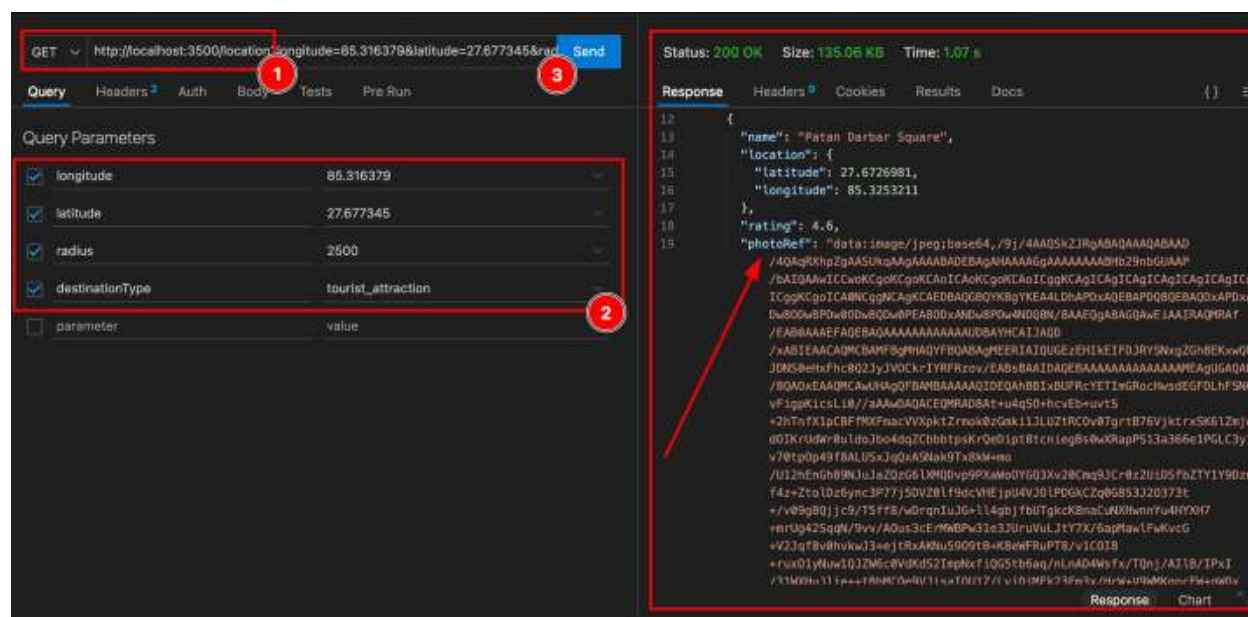


Figure 92: Location list API fetch

5) AIT-14: Search by name testing

Objective	To test if places are retrieved with valid location and radius.
Action	1) Method was set to get request. 2) All the required parameters were passed with radius string. longitude=85.316379 latitude=27.677345 radius=Patan+Durbar+Square destinationType=tourist_attraction on url http://localhost:3500/location 3) Request was sent
Expected result	The server should return searched place along with base64 image.
Actual result	The server returned empty list of places.
Conclusion	Test was not successful.

Table 32: Search by name testing(unsuccesful)

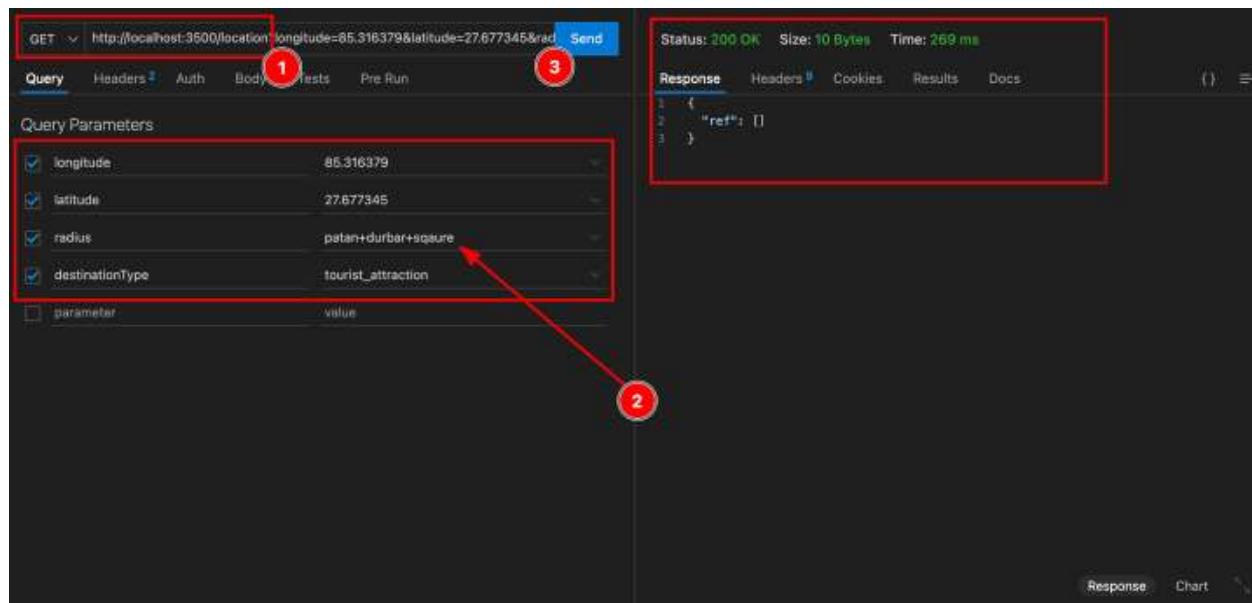
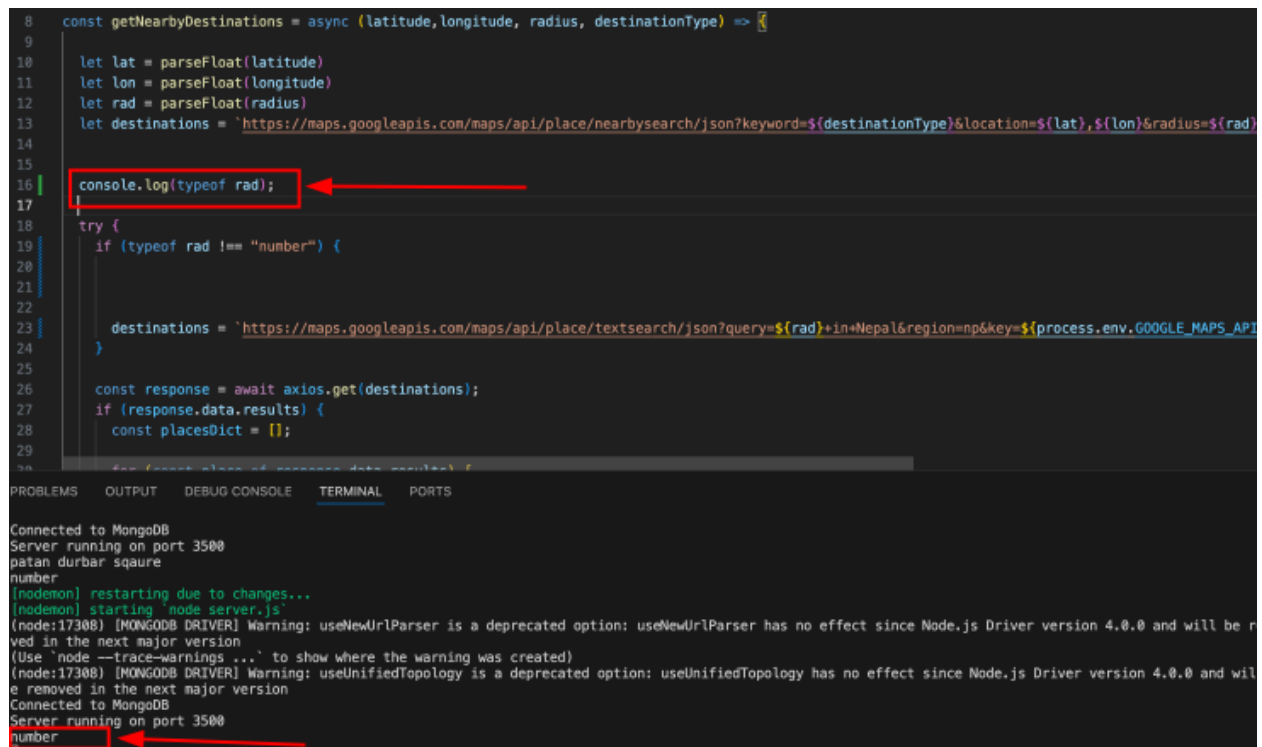


Figure 93: Empty list for name based search

Reason: Since every parameters passed in uri is taken as string, during place fetching it was converted to number. However, even if the passed string has characters in it the type is changed to number but value becomes NaN in javascript. So, API for text search of Google maps API will never be called.



```
8 const getNearbyDestinations = async (latitude, longitude, radius, destinationType) => {
9
10   let lat = parseFloat(latitude)
11   let lon = parseFloat(longitude)
12   let rad = parseFloat(radius)
13   let destinations = `https://maps.googleapis.com/maps/api/place/nearbysearch/json?keyword=${destinationType}&location=${lat},${lon}&radius=${rad}`
14
15
16   console.log(typeof rad);
17
18   try {
19     if (typeof rad !== "number") {
20
21
22       destinations = `https://maps.googleapis.com/maps/api/place/textsearch/json?query=${rad}+in+Nepal&region=np&key=${process.env.GOOGLE_MAPS_API_KEY}`
23     }
24
25     const response = await axios.get(destinations);
26     if (response.data.results) {
27       const placesDict = {};
28
29       for (const place of response.data.results) {
30
31       }
32     }
33   } catch (error) {
34     console.log(error);
35   }
36 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Connected to MongoDB
Server running on port 3500
patan durbar square
number
[nodemon] restarting due to changes...
[nodemon] starting 'node server.js'
(node:17308) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
(Use 'node --trace-warnings ...' to show where the warning was created)
(node:17308) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version
Connected to MongoDB
Server running on port 3500
number

Figure 94: Wrong data type of radius

Correction Measure: Instead of checking `typeof rad`, it should be checked if `isNaN(rad)`. The passed `rad` in API link for text search also should be replaced with `radius` as `NaN` place will not exist.

Objective	To test if places are retrieved with valid location and radius.
Action	1) Method was set to get request. 2) All the required parameters were passed with radius string. longitude=85.316379 latitude=27.677345 radius=Patan+Durbar+Square destinationType=tourist_attraction on url http://localhost:3500/location 3) Request was sent
Expected result	The server should return searched place along with base64 image.
Actual result	The server returned searched place along with base64 image.
Conclusion	Test was successful.

Table 33: Search by name testing (successful)

```

15
16
17 try {
18   if (!isNaN(rad)) {
19     destinations = `https://maps.googleapis.com/maps/api/place/textsearch/json?query=${radius}&ln=Nepal&region=no&key=${process.env.GOOGLE_MAPS_API}
20   }
21
22   const response = await axios.get(destinations);
23   if (response.data.results) {
24     const placesDict = [];
25
26

```

Figure 95: Corrected code for place search by name

GET Send

Query Headers Auth Body Tests Pre Run

Query Parameters

parameter	value
longitude	85.316379
latitude	27.677345
radius	patan+durbar+sqaure
destinationType	tourist_attraction

Status: 200 OK Size: 91.03 KB Time: 1.49 s

Response Headers Cookies Results Docs

```
{
  "ref": 1
}
```

Response Chart

Figure 96: Search by place name result

6) AIT-15 Base64 Conversion testing

Objective	To test if base64 image from google place photos is working.
Action	1) Method was set to get request. 2) All the required parameters were passed with radius string. longitude=85.316379 latitude=27.677345 radius=Patan+Durbar+Square destinationType=tourist_attraction 3) Request was sent on url http://localhost:3500/location 4) Retrieve base64 image was copied and pasted on https://base64.guru/converter/decode/image
Expected result	The pasted base64 image should output a valid image
Actual result	The pasted base64 image provided a valid image.
Conclusion	Test was successful.

Table 34: Base64 Conversion testing

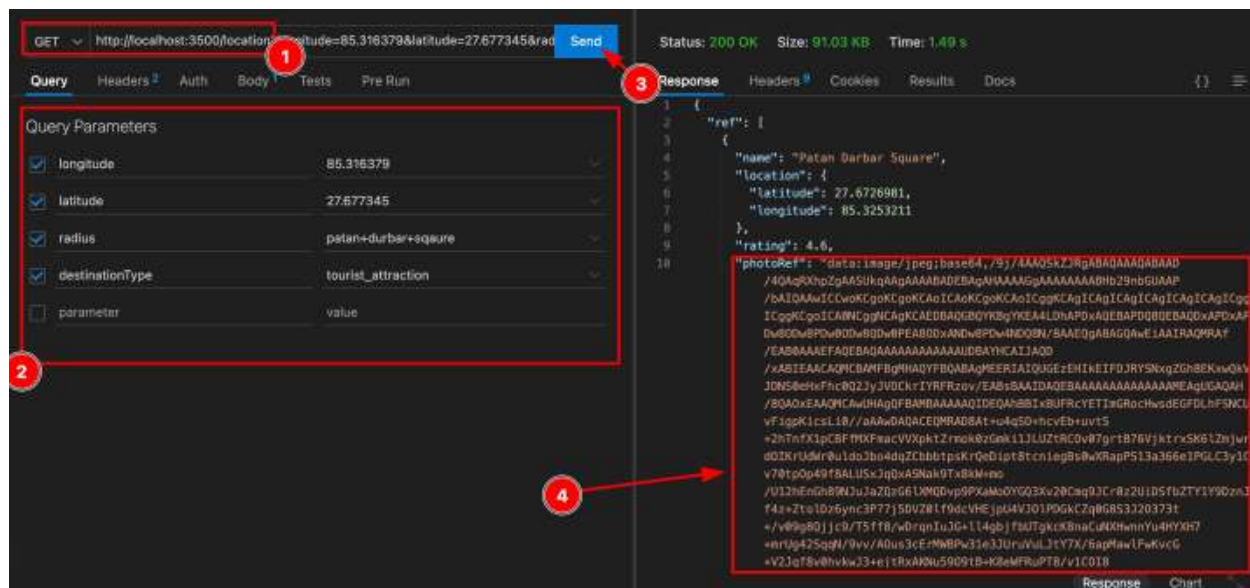


Figure 97: Retrieval of base64 image

Base64 to Image

Comments: 203 Rating: 4.2/5

Convert Base64 to image online using a free decoding tool which allows you to decode Base64 as image and preview it directly in the browser. In addition, you will receive some basic information about this image (resolution, MIME type, extension, size). And, of course, you will have a special link to download the image to your device. If you are looking for the reverse process, check [Image to Base64](#).

Base64*

Copy Clear Download

j9ij4AAQSkZJRgABAQAAQABAAQDyqGAgRXhpZgAASUkqAAAAQABAEBAgAAAAAAAAAAAAAAAAABHb29nbGUAAAPjYAIQAawICcwKCgoKCoKCAoIcAoKCGoKCAoIcCggKAgiCAGiCAGiCAGiCAGiCgglCggKCGoiCAONCcgCAEDBAQGBQBYKBKEA4LdPhADPwAQEBAPQSBQEBAQDxAPDw8ODw6PDwOODw8QDw0PEABODxANDw8PDw4NDQB8NIAEAEqgA8AGQAwEIAAIRAQMRAtIEABDAAEFAQEBQAAAAAAAAAAAAAAAAUDBAYHCAlJAQDIxABIEAACQMCBAMfBgMHAQYFBQA8agMEERIAIUQEzEHikeIFDJRYSXvgZGh8EKxwQKVJDN50eHxFhc0Q2JyJVOCkrIYRFZowIEAB8AAIDAQEBAAAAAAAAAAAAAAAAAMEAgUGAQHIBQAOEAAMCMCAwUHAgQFBAMBAAAAAQIDEQA8BBixBUFRcYETImGRocHwsdEGFDLhFSNCUVFiggKicsLIjjaAawDAQACEQMRAD8Bat+u4qS0+hcvEb+uvvtS+2htnfXtpCBFMXMFmacVVPkiZrmakOzmklJLUZIRCOv07grlB76vjktixSK6IZmjwdOKrUdWouldoJbo4dqZCbttpeKrQeDiptbtcniegBs0wXRapPS13a366eIPGLC3ylCv70tpOp49fBALUSxjqQxA5Nak9Tx8kW+majUI2tEnQh89NuJaZqzg6BXMQDvp9PXawWoYGQQ3xv20Cmg9JCrdzUIDShBTZYTY9DznJl4+zToldzbync3P77J5DV20If9dcVHEpU4VJOIPDGKczQOGSSJZ0373t+jv09g8Qjic9T6ff8/wDrnluwGI4lgbfjbUTgkcKBnaCuNXHwnny4UYXH7+mrUG42SqNj9vvyIQAus3CE/MWPw3Te3JUruVLjYT7j6apMawfwkwG+Y2JlgdvhkwwJ3+ejlRxAKNU59091B+K8eWFRuPTb/jYCOI8+ruxDTYNuw1QJZW6c0vdKSD5impNxfI

Decode Base64 to Image

Preview Image Toggle Background Color

File Info

Resolution: 400×240

MIME type: image/png

Figure 98: Base64 image conversion

7) AIT-16 Description generation for valid place

Objective	To test if description request is working for valid place name.
Action	1) Method was set to get request. 2) All the required parameters were passed. Name= "patan+durbar+square" 3) Request was sent on url http://localhost:3500/location/getDescription
Expected result	A proper description should be generated by gemini in response.
Actual result	A proper description was generated by gemini in response.
Conclusion	Test was successful.

Table 35: Description generation for place

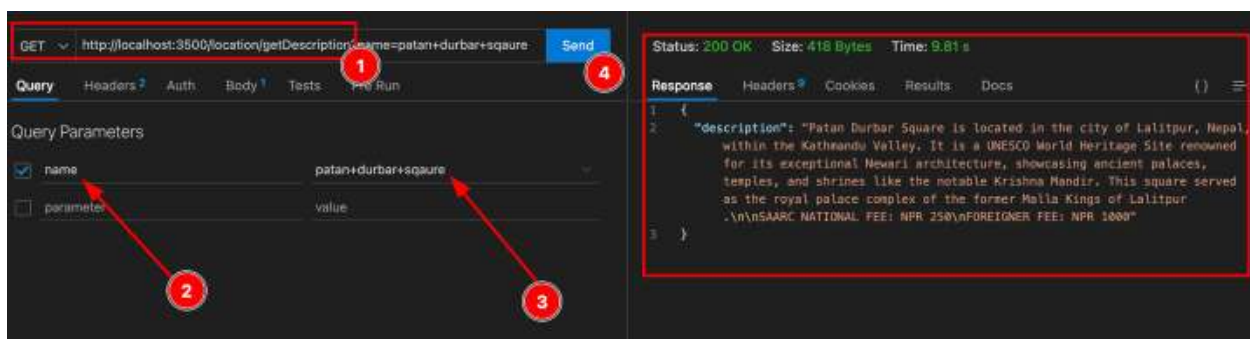


Figure 99: Gemini API description generation

8) AIT-17 Description generation without place name

Objective	To test if server responds properly when no place name for description is provided.
Action	1) Method was set to get request. 2) No parameters were passed. Name= 3) Request was sent on url http://localhost:3500/location/getDescription
Expected result	A proper response with error code should be returned by the server.
Actual result	A proper response with error code was returned by the server.
Conclusion	Test was successful.

Table 36: Description generation without name



Figure 100: Gemini API description generation without name

9) AIT-18 Existing review fetching

Objective	To test if existing reviews for a location is fetched upon request.
Action	1) Method was set to get request. 2) Required parameters were passed Lat=85.3253211 Lng=27.6726981 Latitude and longitude were passed for which reviews were present. 3) Request was sent http://localhost:3500/getReviews
Expected result	Server should send a response of list of reviews for that particular place.
Actual result	Server sent a response of list of reviews for that particular place.
Conclusion	Test was successful.

Table 37: Existing review fetching

```
  _id: ObjectId('67f10ecd8fecf6054c063461')
  ▼ location : Object
    latitude : 85.3253211
    longitude : 27.6726981
  ▼ reviews : Array (1)
    ▼ 0: Object
      text : "Wow such a great place."
      email : "rhythm@gmail.com"
      firstname : "Rhythm"
      lastname : "Paudel"
      _id : ObjectId('67f10ecd8fecf6054c063462')
      date : 2025-04-05T11:06:53.734+00:00
    name : "Patan Darbar Square"
    __v : 0
```

Figure 101: Reviews of Patan Darbar square

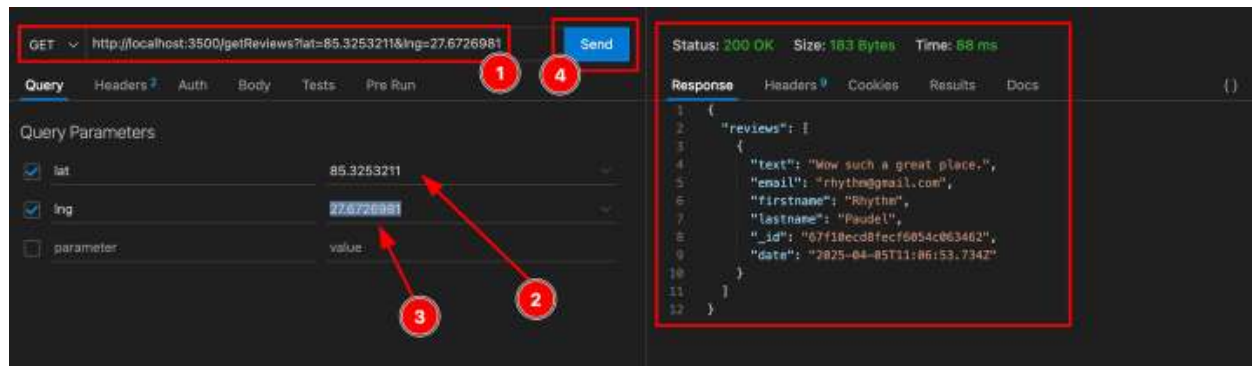


Figure 102: Review fetching

10) AIT-19 Review fetching for a place without reviews

Objective	To test if proper response is sent when no reviews are found.
Action	1) Method was set to get request. 2) Required parameters were passed Lat=85.3253210 Lng=27.6726980 Latitude and longitude were passed for which reviews were not present. 3) Request was sent http://localhost:3500/getReviews
Expected result	Server should send a proper response if no reviews are found.
Actual result	Server sent an empty list indicating no reviews found.
Conclusion	Test was successful.

Table 38: Review fetching for place without reviews



Figure 103: Review fetching for place without reviews

11) AIT-20 Requesting reviews without location

Objective	To test if server response properly when no latitude and longitude are provided for reviews.
Action	1) Method was set to get request. 2) No paramenters were passed 3) Request was sent on url http://localhost:3500/location/getReviews
Expected result	Server should send a proper response and proper status code.
Actual result	Server sent a proper response and proper status code.
Conclusion	Test was successful.

Table 39: Requesting reviews without location

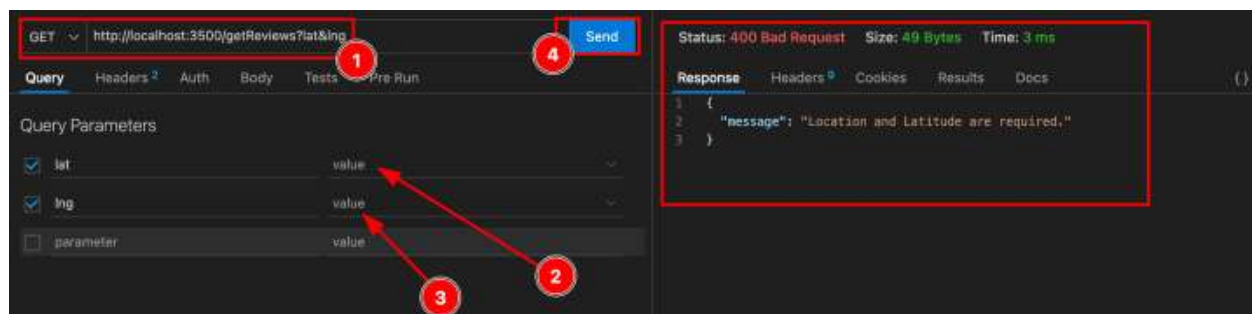


Figure 104: Requesting reviews without location

4.3.4. Notification testing

1) AIT-21 Testing for push notification via API

Objective	To test if notifications for a valid token is sent to mobile phone.
Action	1) Method was set to post request. 2) URL was set to localhost:3500/admin/login 3) Valid username and password was set Username: "rhythm" Password: "rhythm" 4) Access token was copied 5) Access token was pasted on Auth→Bearer section. 6) URL was set to localhost:3500/send-notification 7) Mobile token, email, title body and message type was set 8) Request was sent.
Expected result	Notification should be delivered to the mobile phone and should be saved in database as well.
Actual result	Notification was delivered to the mobile phone and was saved in database as well.
Conclusion	Test was successful.

Table 40: Testing for push notification (API)

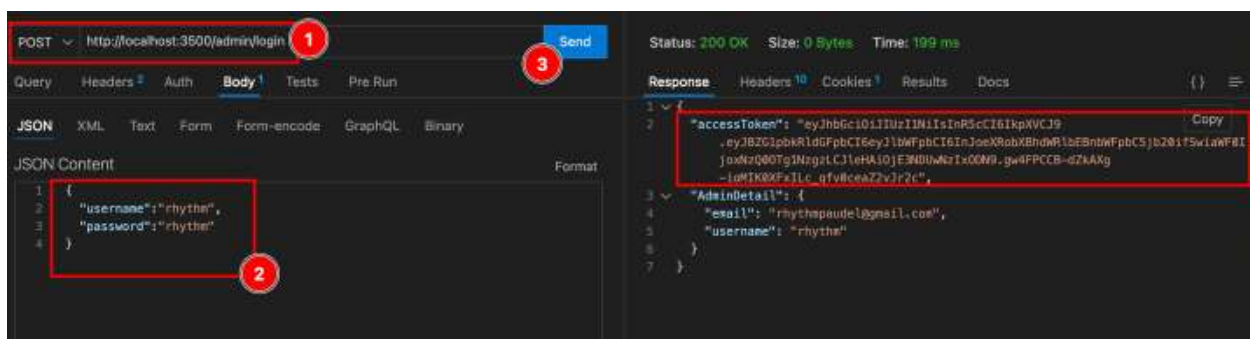


Figure 105: Retrieving access token for admin

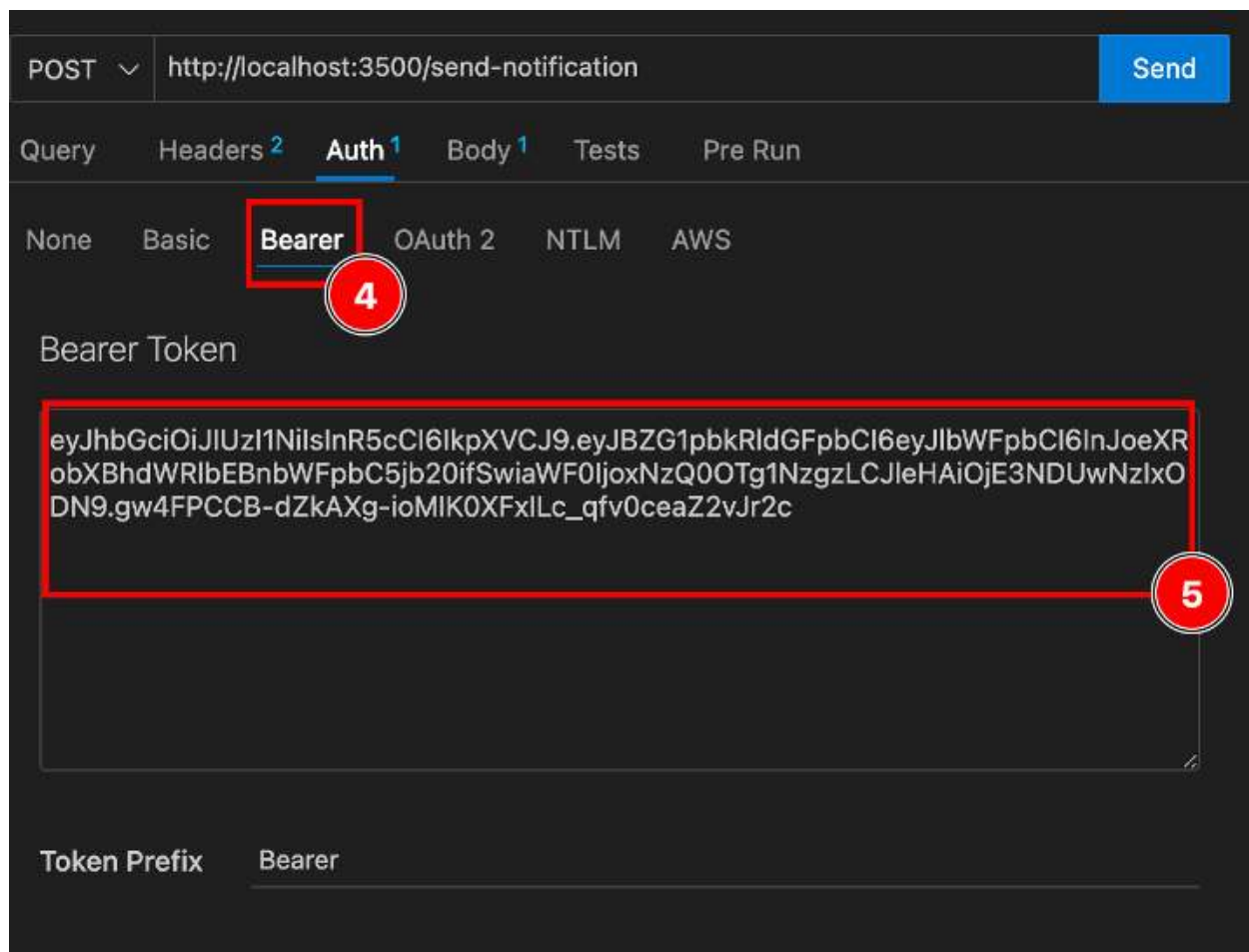


Figure 106: Pasting access token in bearer token for notification

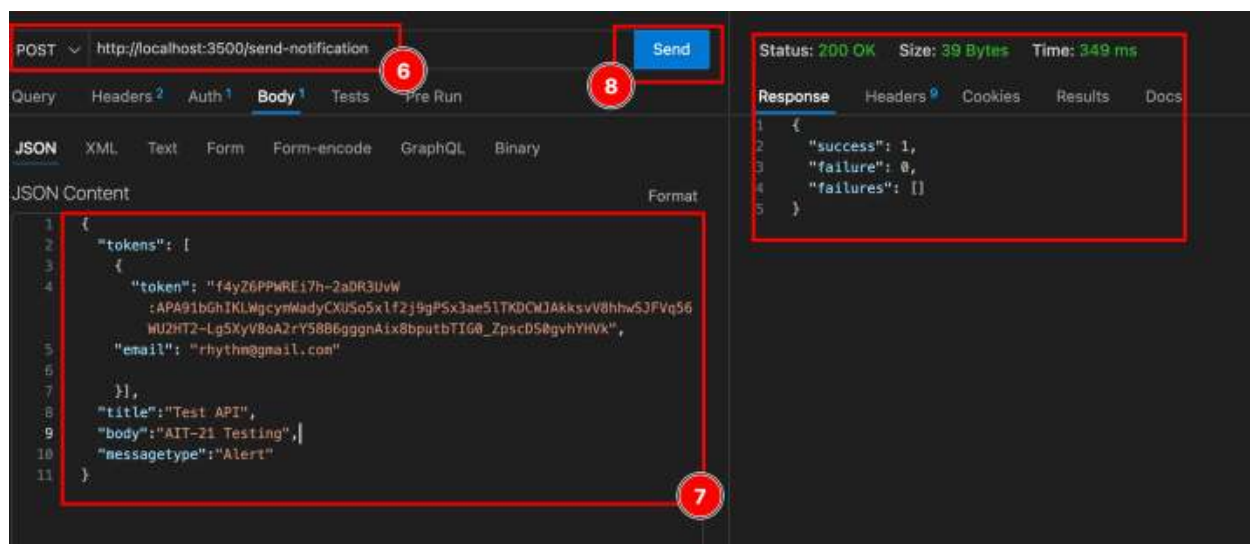


Figure 107: Sending notification to a mobile phone(API)

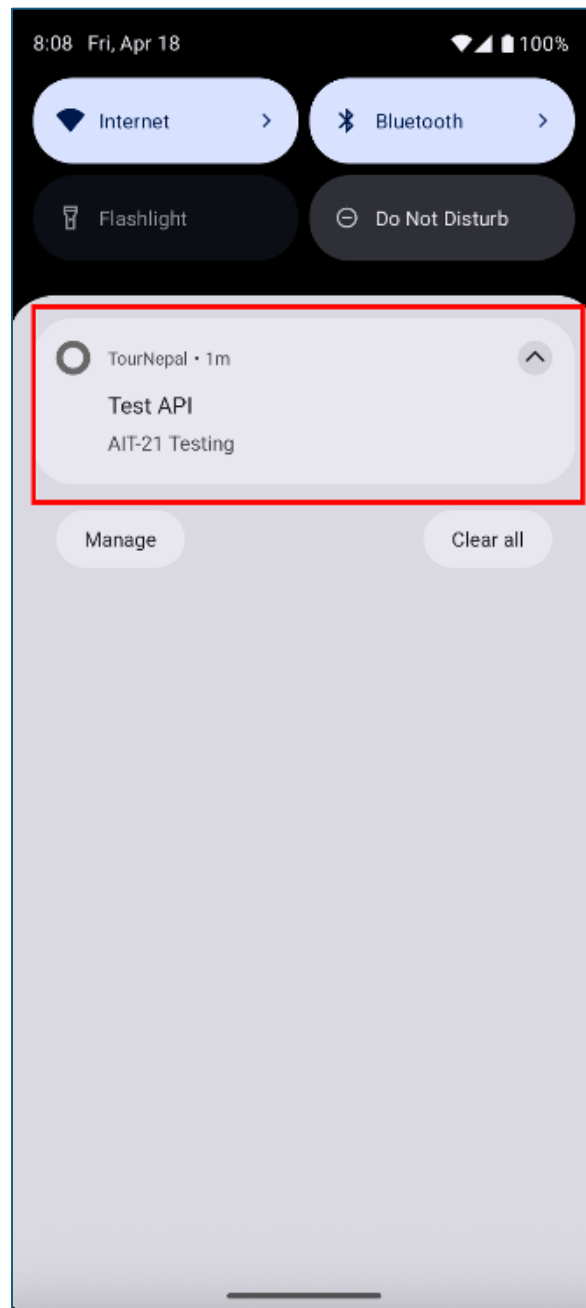


Figure 108: API testing notification on phone

```
17   ▼ 0: Object
18     title : "Test API,"
19     message : "AIT-21 Testing,"
20     messagetype : "Alert,"
```

Figure 109: Notification history updated in mongoDB

2) AIT-22 Multiple device notification casting

Objective	To test if notifications for multiple devices is sent mobile phones.
Action	1) Method was set to post request. 2) Access token was generated from admin login. 3) Access token was copied 4) Access token was pasted on Auth→Bearer section. 5) URL was set to localhost:3500/send-notification 6) Mobile token, email, title body and message type was set 7) Request was sent.
Expected result	Notification should be delivered to multiple mobile phones and should be saved in database as well.
Actual result	Notification was delivered to the multiple mobile phones and was saved in database as well.
Conclusion	Test was successful.

Table 41: Multiple device notification casting

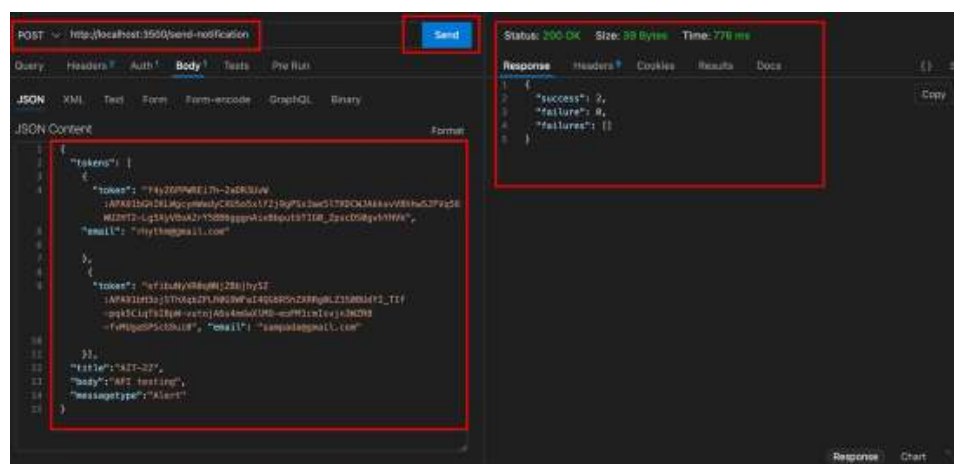


Figure 110: Sending notification to multiple mobile phones(API)

3) AIT-23 Sending notification without device tokens

Objective	To test if server responds properly when body is passed without all the fields for notifications.
Action	1) Method was set to post request. 2) Access token was generated from admin login. 3) Access token was copied 4) Access token was pasted on Auth→Bearer section. 5) URL was set to localhost:3500/send-notification 6) Email, title body and message type was set but tokens field was left empty. 7) Request was sent.
Expected result	Server should respond with proper status code and message.
Actual result	Server responded with proper status code and message.
Conclusion	Test was successful.

Table 42: Sending notification without device tokens

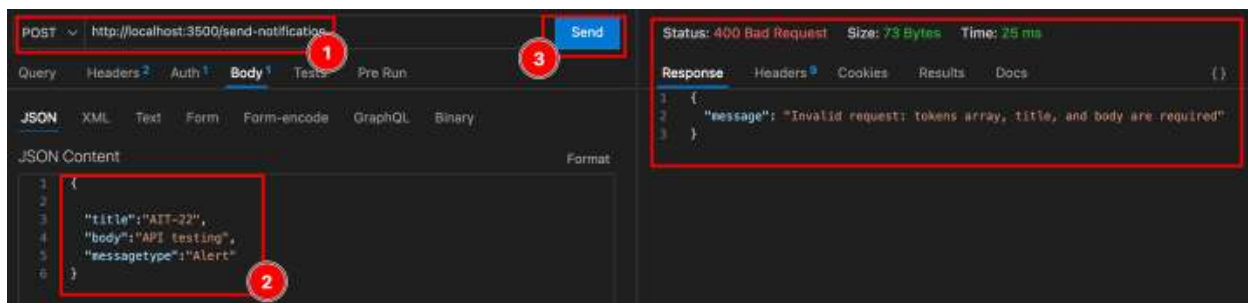


Figure 111: Sending notifications without tokens

4) AIT-28 Updating notification token

Objective	To test if user notification token is updated.
Action	1) Logged in with valid credentials from the application
Expected result	User notification token should be updated to match new device token.
Actual result	User notification token was updated to match new device token.
Conclusion	Test was successful.

Table 43: Updating notification token

```

_id: ObjectId('68911a95edd157385f461151')
email: "islingtoncollege@gmail.com"
firstname: "rhythm"
lastname: "paudel"
dateOfBirth: 2004-09-24T00:00:00.000+00:00
visaStamp: "xys"
passportCopy: "passportCopy"
password: "92b5185g6C2lU46Q7NMkkhKwldiZe0V2G6.W0FV0wQ91R5lrg2Jtv4o1W4Ym"
verificationStatus: "pending"
dateOfEntry: 2004-09-24T00:00:00.000+00:00
nationality: "NP"
intendedDays: 3
deletionRequest: false
__v: 0

```

Figure 112: User status before notification token update

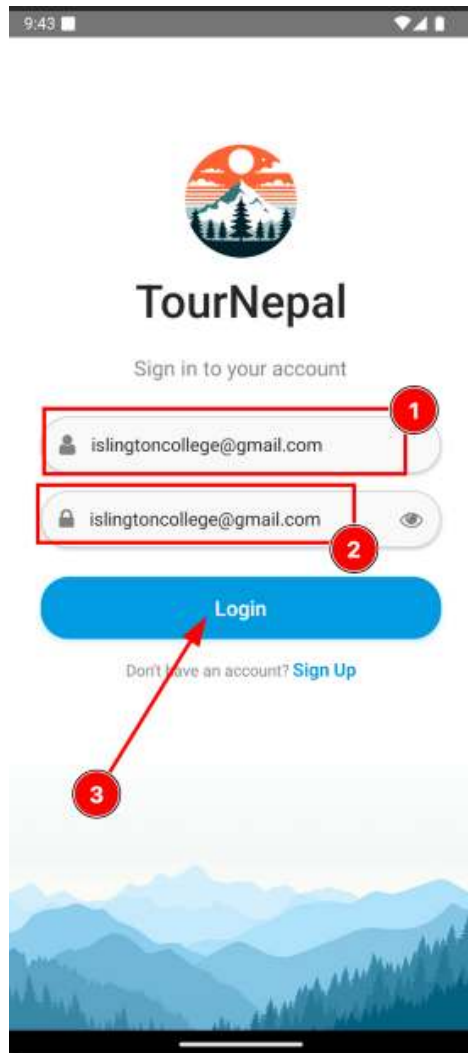


Figure 113: Logging in for token update

```
_id: ObjectId('66011a95edd157385f401151')
email: "islingtoncollege@gmail.com"
firstname: "rhythm"
lastname: "paul"
dateOfBirth: 2004-09-24T00:00:00.000+00:00
visaStamp: "xys"
passportCopy: "passportCopy"
password: "$2b$10$g6C7IH46Q?NWkKhKw1di3e8V26G.W0FVCwQ91R81rg2Jtv4n3W4Ym"
verificationStatus: "pending"
dateOfEntry: 2004-09-24T00:00:00.000+00:00
nationality: "NP"
intendedDays: 3
deletionRequest: false
__v: 0
notificationToken: "f4yz6PPWRE17h-zapR10vW:APA91bch1KLwgcywWedyCXU5o5xU12j9gP5x3ae5LYKDCW3..."
```

Figure 114: Notification token update after login

5) AIT-30 Improper notification token handling

Objective	To test if notifications for a invalid token handled properly
Action	1) Access token was copied from admin login. 2) Access token was pasted on Auth→Bearer section. 3) URL was set to localhost:3500/send-notification 4) Mobile token, email, title body and message type was set, however an invalid token was set. 5) Request was sent.
Expected result	Server should handle improper token with proper message in response.
Actual result	Server handled improper token with proper message in response.
Conclusion	Test was successful.

Table 44: Improper notification token handling

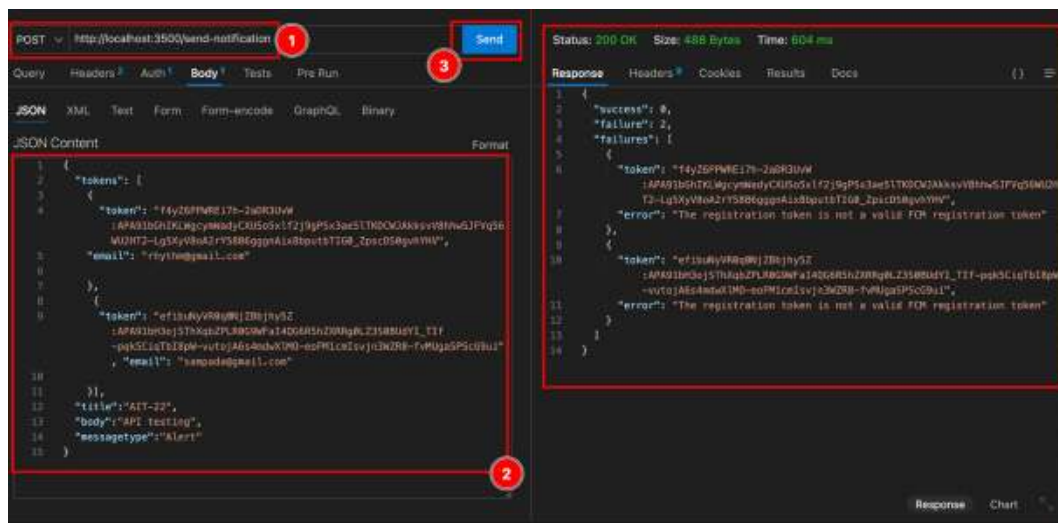


Figure 115: Invalid notification token handling

4.3.5. User details fetch/manipulation testing

1) AIT-24 User detail fetching via API

Objective	To test if user details is retrieved by user email or not.
Action	1) Method was set to get request. 2) Access token was generated from user login. 3) Access token was copied 4) Access token was pasted on Auth→Bearer section. 5) URL was set to localhost:3500/userdetail 6) Request was sent.
Expected result	Server should respond with user detail.
Actual result	Server responded with user detail.
Conclusion	Test was successful.

Table 45: User detail fetching(API)

Here email is decoded from access token.

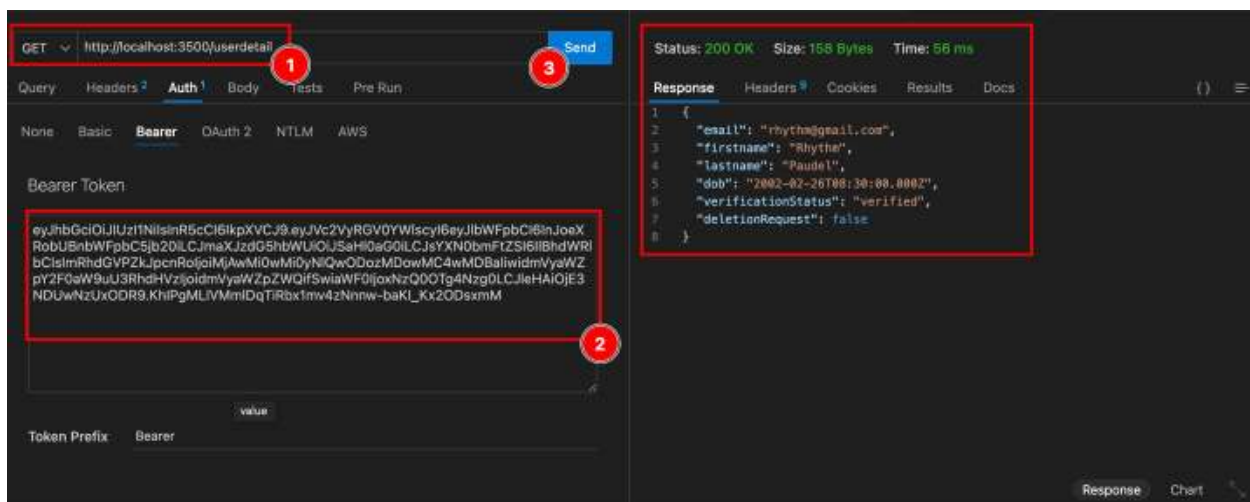


Figure 116: Retrieving user details from email

2) AIT-25 User details fetching with expired token

Objective	To test if user detail is retrieved by user email, without/expired access token.
Action	1) Access token expiration time was set to 10 seconds 2) Access token was generated from user login. 3) Access token was copied 4) Method was set to get request. 5) Access token was pasted after 10 seconds on Auth→Bearer section. 6) URL was set to localhost:3500/userdetail 7) Request was sent.
Expected result	Server should respond with proper status code and message.
Actual result	Server responded with proper status code and message.
Conclusion	Test was successful.

Table 46: User details fetching with expired token

```

21     if (match) {
22         //ASSISING Access Token to the user
23         const accessToken = jwt.sign({
24             "UserDetails": {
25                 "email": userDB.email,
26                 "firstname": userDB.firstname,
27                 "lastname": userDB.lastname,
28                 "dateOfBirth": userDB.dateOfBirth,
29                 "verificationStatus": userDB.verificationStatus,
30             },
31         },
32         process.env.ACCESS_SECRET_TOKEN,
33         {expiresIn: '10s'}
34     );
35
36

```

Figure 117: Setting expiration token to 10 seconds(user)

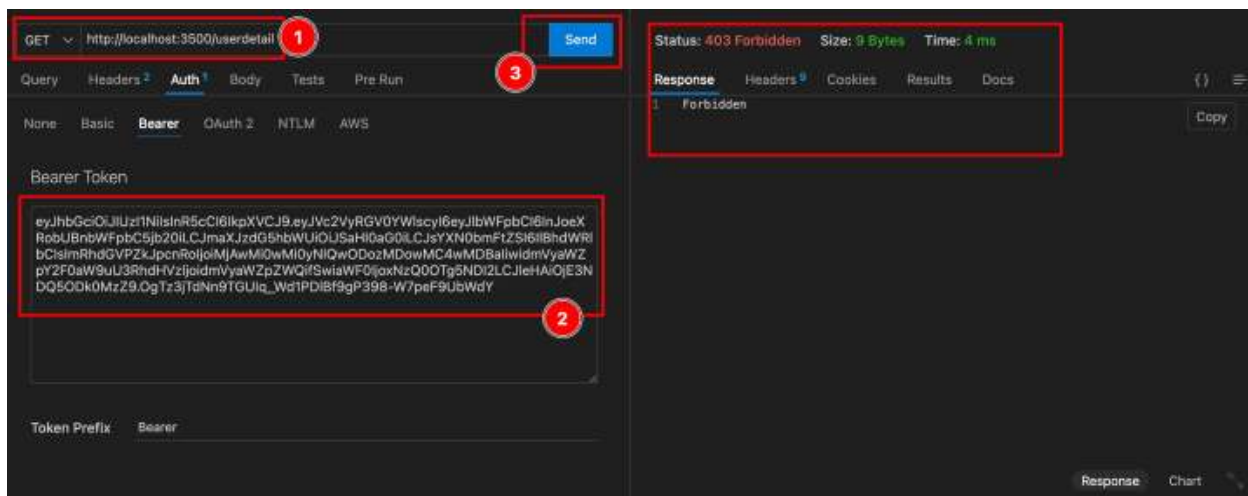


Figure 118: Retrieving user detail with expired access token

3) AIT-26 User deletion request update

Objective	To test if user deletion request is updated.
Action	1) Access token was generated from user login. 2) Access token was copied 3) Method was set to put request. 4) Access token was pasted on Auth→Bearer section. 5) URL was set to localhost:3500/userdetail/updateuser 6) Body with deletionRequest: true was provided 7) Request was sent.
Expected result	User deletion request should be updated to true.
Actual result	User deletion request was updated to true.
Conclusion	Test was successful.

Table 47: User deletion request update

1	<code>_id: ObjectId('67bed17c142da6ecab2baa6b')</code>
2	<code>email: "rhythm@gmail.com"</code>
3	<code>firstname: "Rhythm"</code>
4	<code>lastname: "Paudel"</code>
5	<code>dateOfBirth: 2002-02-26T08:30:00.000+00:00</code>
6	<code>visaStamp: "https://res.cloudinary.com/dvnnduvi3/image/upload/v1740558715/uhju9qnj"</code>
7	<code>passportCopy: "https://res.cloudinary.com/dvnnduvi3/image/upload/v1740558714/m3nzjvro"</code>
8	<code>password: "\$2b\$10\$4didC9Sm3Q8NF7vVCL9zXuu8w24zPsYU13gU9R06xmcd8z4UWjL62"</code>
9	<code>verificationStatus: "verified"</code>
10	<code>dateOfEntry: 2025-02-26T08:30:00.000+00:00</code>
11	<code>nationality: "CA"</code>
12	<code>intendedDays: 30</code>
13	<code>deletionRequest: false</code>
14	<code>__v: 0</code>
15	<code>notificationToken: "f4yZ6PPWREi7h-2aDR3UvW:APA91bGhIKLWgcymWadyCXUSo5xlf2j9gPSx3ae5lTKDCWJ"</code>

Figure 119: User deletion status before deletion request

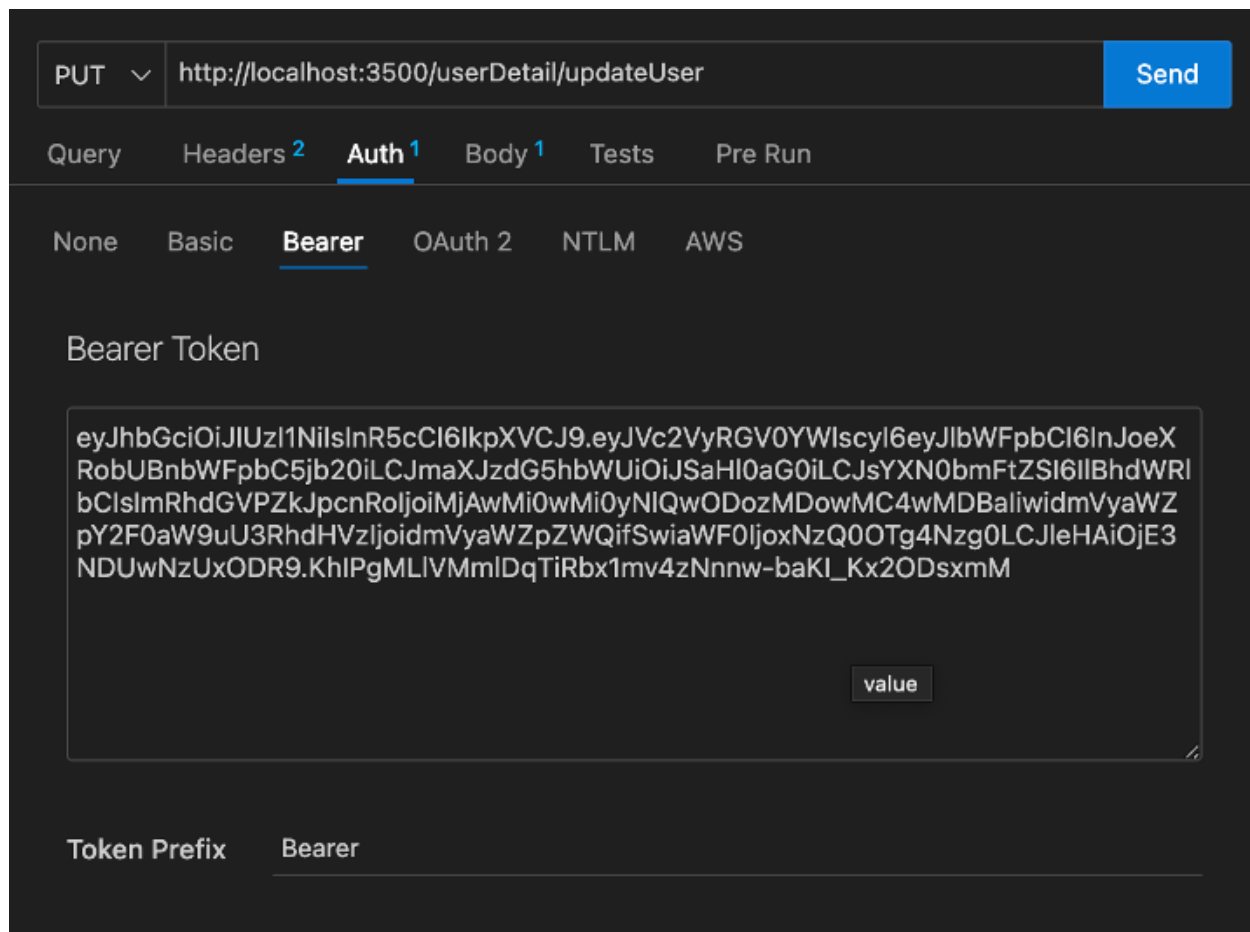


Figure 120: Deletion request bearer token

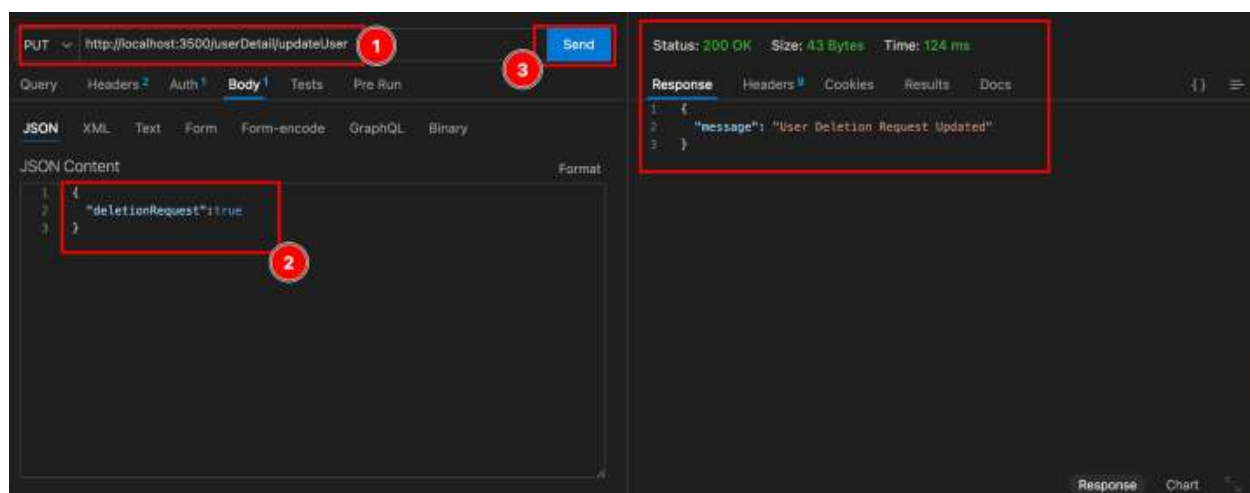


Figure 121: Deletion request change via API



Figure 122: User deletion status after deletion request

4) AIT-27 Deletion request without providing access token

Objective	To test if user deletion request is rejected when no access token is provided for verification.
Action	1) Method was set to put request. 2) URL was set to localhost:3500/userdetail/updateuser 3) Body with deletionRequest: true was provided 4) Request was sent.
Expected result	Deletion request should not be updated as no token was provided.
Actual result	Deletion request was not updated as no token was provided.
Conclusion	Test was successful.

Table 48: Deletion request without providing access token

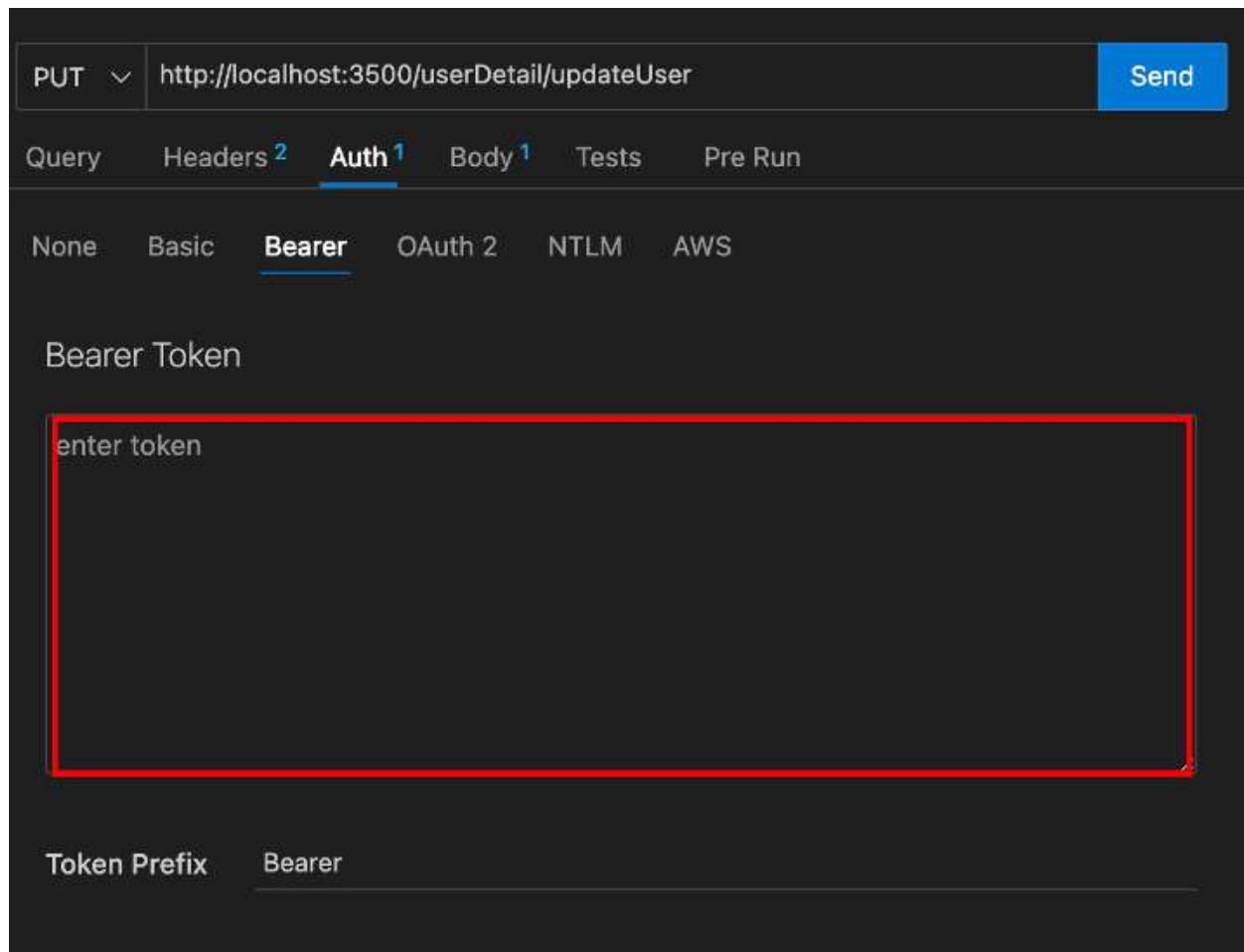


Figure 123: Updating user deletion without bearer token(1)

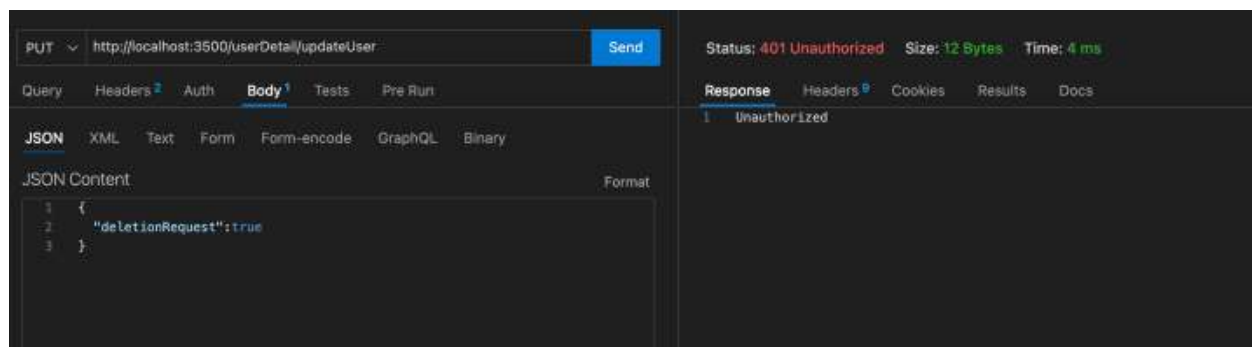


Figure 124: Updating user deletion without bearer token(2)

5) AIT-29 Retrieving user notification history

Objective	To test if user notifications history is fetched.
Action	2) Valid access token was copied and pasted on Auth→Bearer section. 3) Method was set to put request. 4) URL was set to localhost:3500/userdetail/getnotifications 5) Request was sent.
Expected result	List of notifications should be sent to the user as a response.
Actual result	List of notifications was sent to the user as a response.
Conclusion	Test was successful.

Table 49: Retrieving user notification history

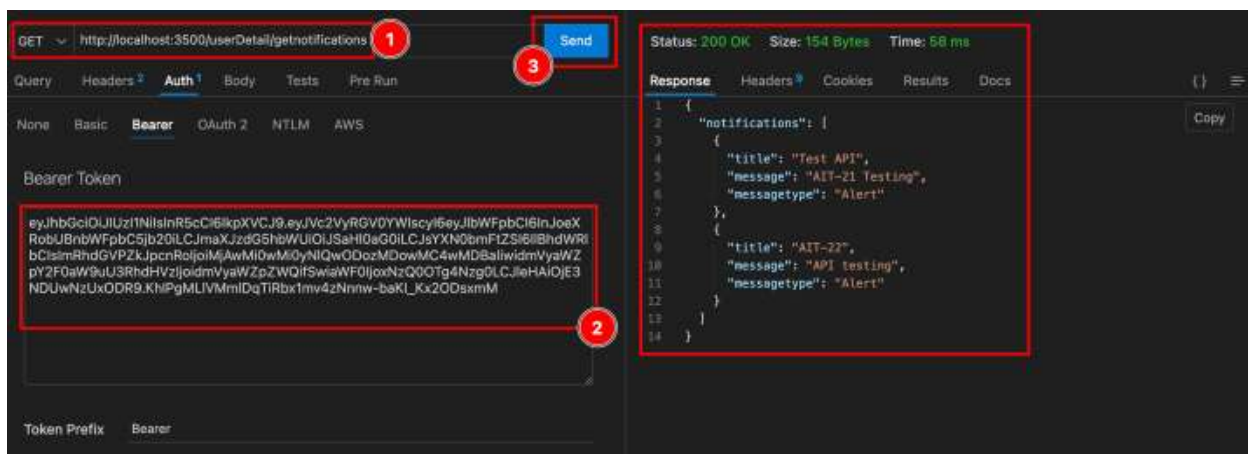


Figure 125: Fetching user notifications

6) AIT-46 User logging out of the application

Objective	To test if user is able to logout successfully from the application.
Action	1) Logged in to the application using valid credentials. 2) Navigated to profile section. 3) Clicked on Logout Button.
Expected result	Account should be logged out and login screen should be visible.
Actual result	Account was not logged out and exception was thrown when restarting the application.
Conclusion	Test was not successful.

Table 50: User logging out of the application(unsuccesful)

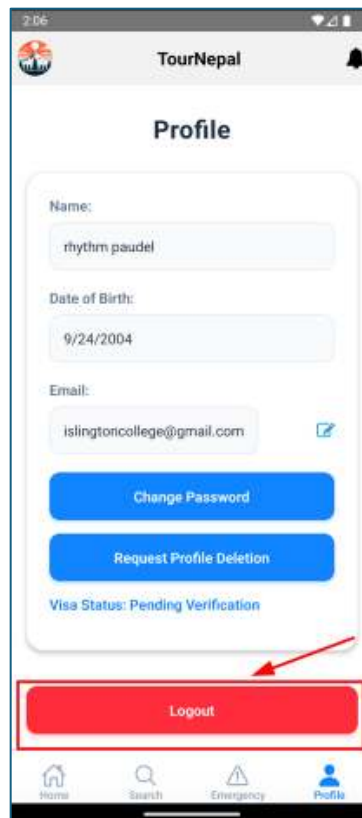


Figure 126: Logging out from application

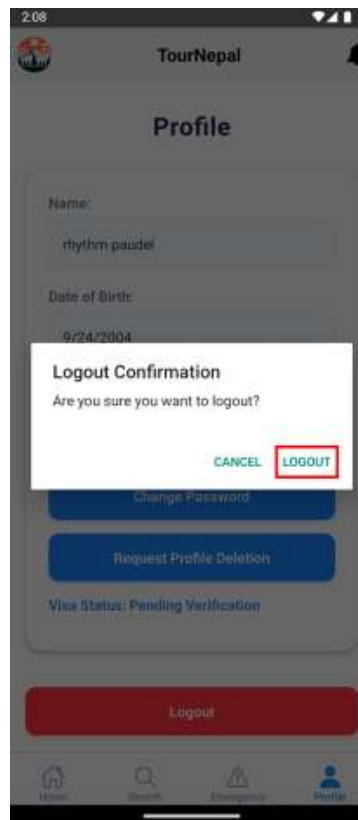


Figure 127: Confirming logout



Figure 128: Error when restarting the application

Reason: If the user was logged in then the user would only have main stack in the options which means everything but login and signup screen which created problem when user clicked logged out there were no screens to redirect to.

Correction Measure: Used conditional rendering as previously logging out was not working expectedly. Instead of redirecting just conditional rendering by setting authentication to false or true during login or logout

```

17 export default function RootNavigator() {
25
26+   return (
27+     <NavigationContainer>
28+       <RootStack.Navigator screenOptions={{
29+         headerShown: false,
30+       }}>
31+         {/AuthStack +/}
32+         {isAuthenticated ? (
33+           <<RootStack.Screen name="Auth" component={AuthStack} />
34+           <RootStack.Screen name="Mainstack" component={MainStack} />
35+         ) : (
36+           <RootStack.Screen name="Mainstack" component={MainStack} />
37+         )}
38+       </RootStack.Navigator>
39+     </NavigationContainer>
40+   );
41+ }
42
43+ const styles = StyleSheet.create({
44+ });
45+
18 export default function RootNavigator() {
25
26+   return (
27+     <NavigationContainer>
28+       <RootStack.Navigator
29+         screenOptions={{
30+           headerShown: false,
31+         }}>
32+         {/AuthStack +/}
33+         {isAuthenticated ? (
34+           <RootStack.Screen name="Auth" component={AuthStack} />
35+         ) : (
36+           <RootStack.Screen name="Mainstack" component={MainStack} />
37+         )}
38+       </RootStack.Navigator>
39+     </NavigationContainer>
40+   );
41+ }
42
43+ const styles = StyleSheet.create({});
44+
45+ //RootStack.Navigator screenOptions={{
46+ //  headerShown: false,
47+ // }};
  
```

Figure 129: Corrected logic for conditional rendering

Objective	To test if user is able to logout successfully from the application.
Action	1) Logged in to the application using valid credentials. 2) Navigated to profile section. 3) Clicked on Logout Button.
Expected result	Account should be logged out and login screen should be visible.
Actual result	Account was logged out and was redirected to login screen.
Conclusion	Test was successful.

Table 51: User logging out of the application

7) AIT-47 Adding a review after user status is verified

Objective	To test if user is able to logout successfully from the application.
Action	1) Logged in to the application using valid credentials. 2) Navigated to search section. 3) Clicked on search all and searched "Labim Mall". 4) Updated logged in user status from pending to verified. 5) Reopened the application 6) Navigated to search section. 7) Clicked on search all and searched "Labim Mall". 8) Tried to add a reviews.
Expected result	After the change in user verification status, user should be able to leave a review.
Actual result	After the change in user verification status user was not able to leave a review.
Conclusion	Test was not successful.

Table 52: Adding a review after user status is verified(unsuccesful)

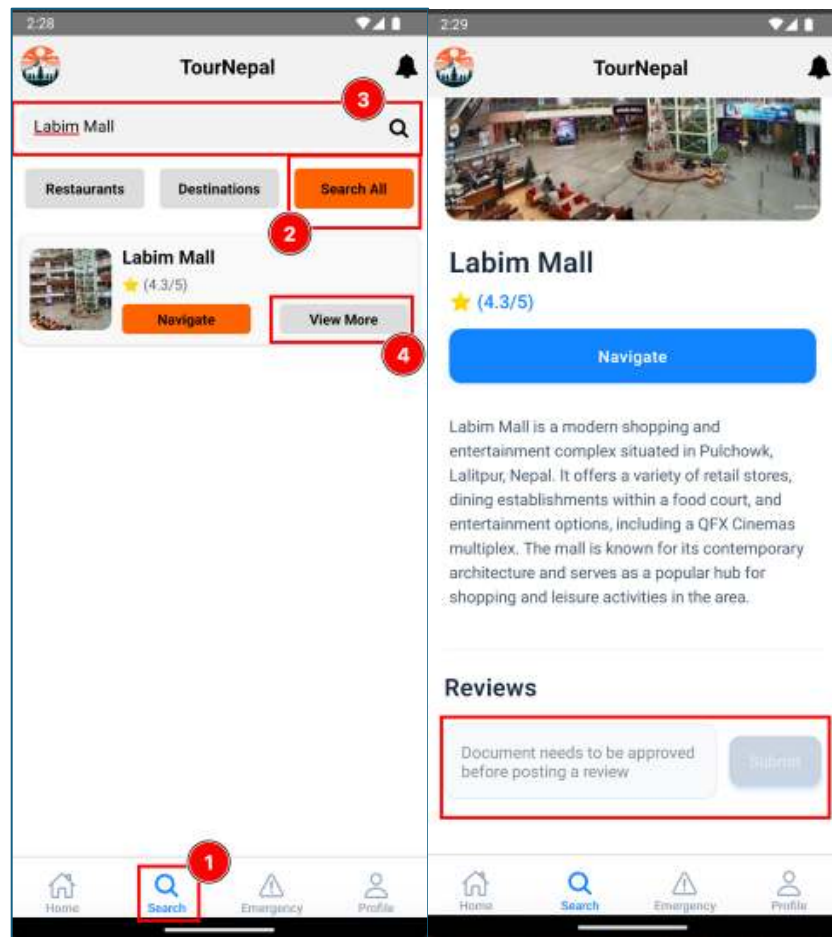


Figure 130: Searching labim mall to post a review



Figure 131: Updating user status from pending to verified in database

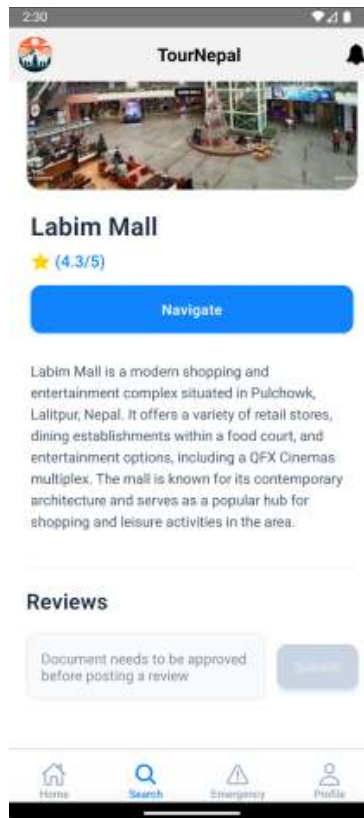
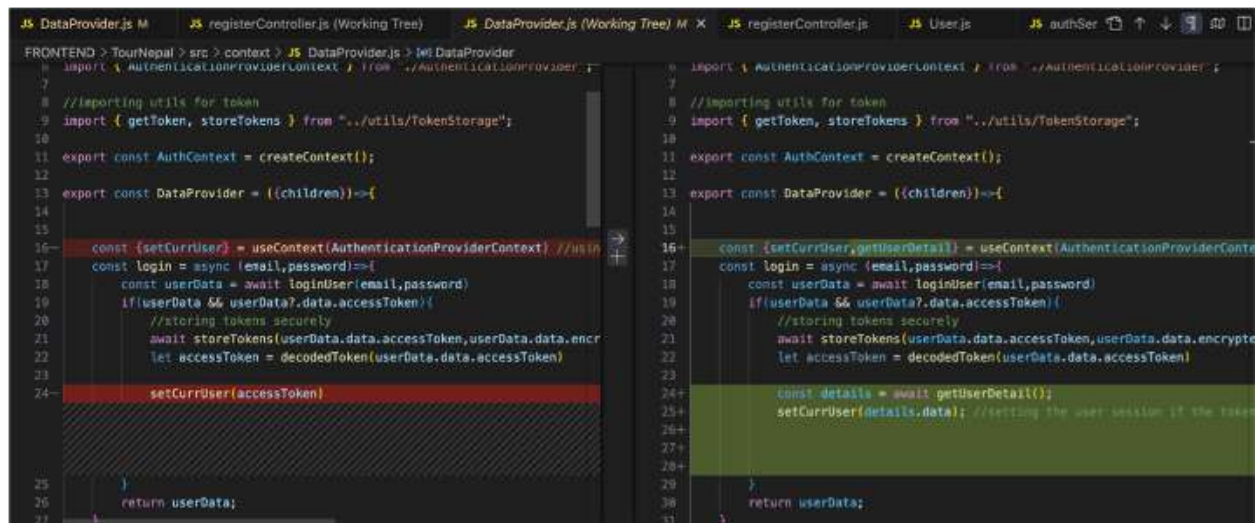


Figure 132: Trying to post a review after user status was verified

Reason: Details from accessToken was decoded that was saved in generic password. But this caused a problem even when the user status changed from pending to verified, unless the user re logged in, they were not able to post a comment.

Correction Measure: So to prevent this, everytime user logs in or app is opened user detail is requested from the server.



```
JS DataProvider.js M JS registerController.js (Working Tree) JS DataProvider.js (Working Tree) M X JS registerController.js JS User.js JS authSer.js
FRONTEND > TourNepal > src > context > JS DataProvider.js > let DataProvider
7 import { AuthenticationProviderContext } from '../AuthenticationProvider';
8 //importing utils for token
9 import { getToken, storeTokens } from '../utils/TokenStorage';
10
11 export const AuthContext = createContext();
12
13 export const DataProvider = ({children})=>{
14
15
16- const {setCurrUser} = useContext(AuthenticationProviderContext); //usin
17 const login = async (email,password)=>{
18   const userData = await loginUser(email,password)
19   if(userData && userData?.data.accessToken){
20     //storing tokens securely
21     await storeTokens(userData.data.accessToken,userData.data.encrypt
22     let accessToken = decodedToken(userData.data.accessToken)
23
24-     setCurrUser(accessToken)
25
26   }
27   return userData;
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
26
```

Objective	To test if user is able to logout successfully from the application.
Action	1) Logged in to the application using valid credentials. 2) Navigated to search section. 3) Clicked on search all and searched "Labim Mall". 4) Updated logged in user status from pending to verified. 5) Reopened the application 6) Navigated to search section. 7) Clicked on search all and searched "Labim Mall". 8) Tried to add a reviews.
Expected result	After the change in user verification status, user should be able to leave a review.
Actual result	After the change in user verification status user was able to leave a review.
Conclusion	Test was successful.

Table 53: Adding a review after user status is verified

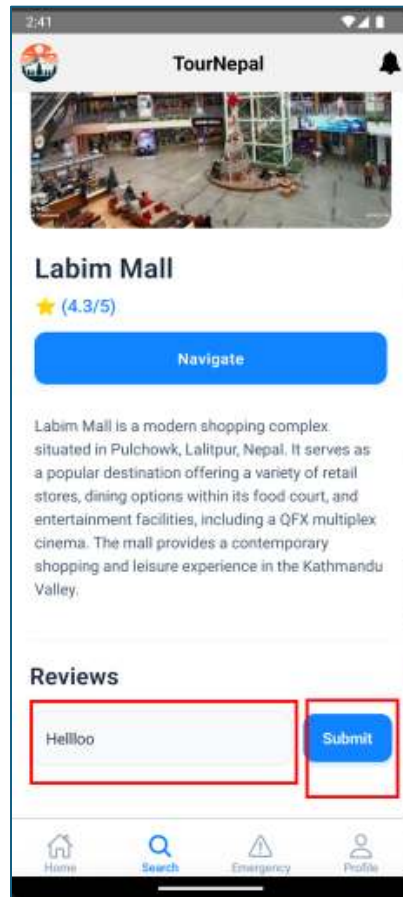


Figure 134: Trying to add a review after verifying user status

8) AIT-49 Starting the application when server is not active

Objective	To test if a proper message is shown when backend is not started.
Action	1) Logged in in the application.
Expected result	Proper message should be shown indicating server has not been started.
Actual result	Proper message was shown indicating some problem with the server.
Conclusion	Test was successful.

Table 54: Starting the application when server is not active

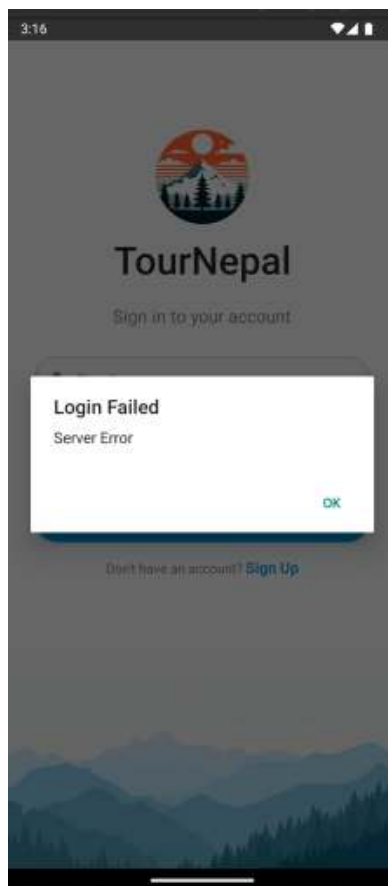


Figure 135: Server error during login

9) AIT-50 Adding a comment via API

Objective	To test if user is able to post a comment via API.
Action	1) User was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to http://localhost:3500/postComments 3) Required field was provided <pre>{ "location":{ "latitude":420, "longitude":420 }, "name": "Test Name", "commentBody": "AIT-50 Testing Review" }</pre> 4) Request was sent to post method.
Expected result	Review should be added in the database
Actual result	Review was added in the database.
Conclusion	Test was successful.

Table 55: Adding a comment(API)

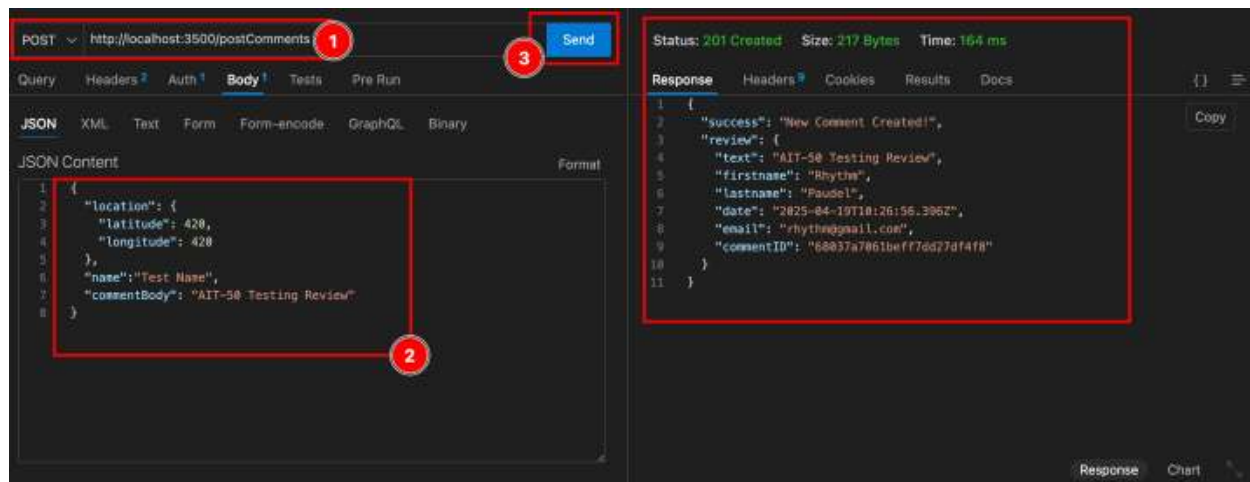


Figure 136: Adding a review(API)

10) AIT-51 Editing an existing comment

Objective	To test if user is able to edit a comment via API.
Action	1) User was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to http://localhost:3500/postComments/ update 3) Required field was provided 4) Request was sent to put method.
Expected result	Review should be updated in the database
Actual result	Review was not updated for the provided commentID but the first comment.
Conclusion	Test was not successful.

Figure 137: Editing an existing comment(Unsuccessful)



Figure 138: Comment before updating(API)

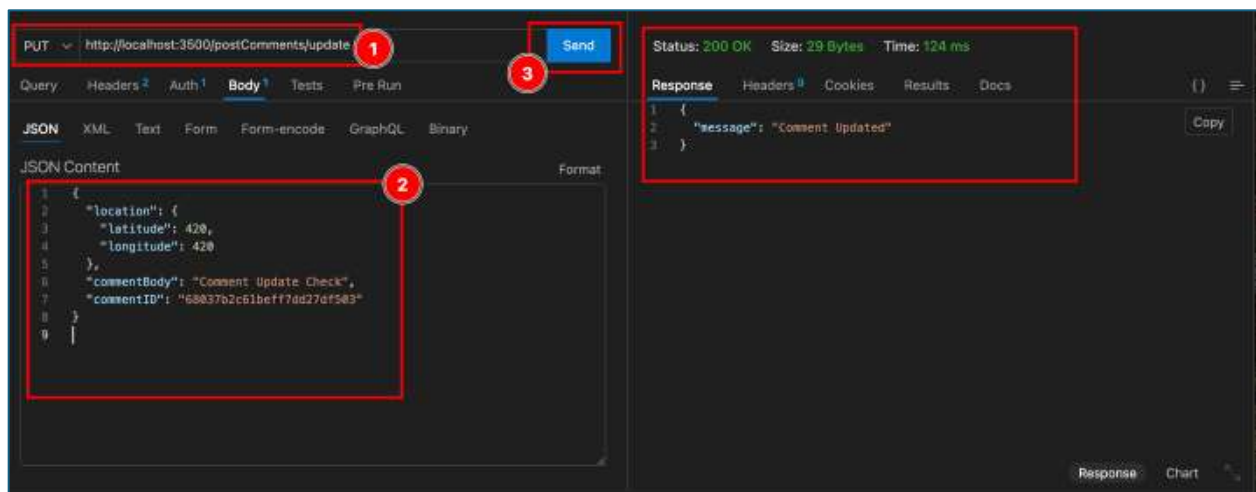


Figure 139: Updating comment(API)

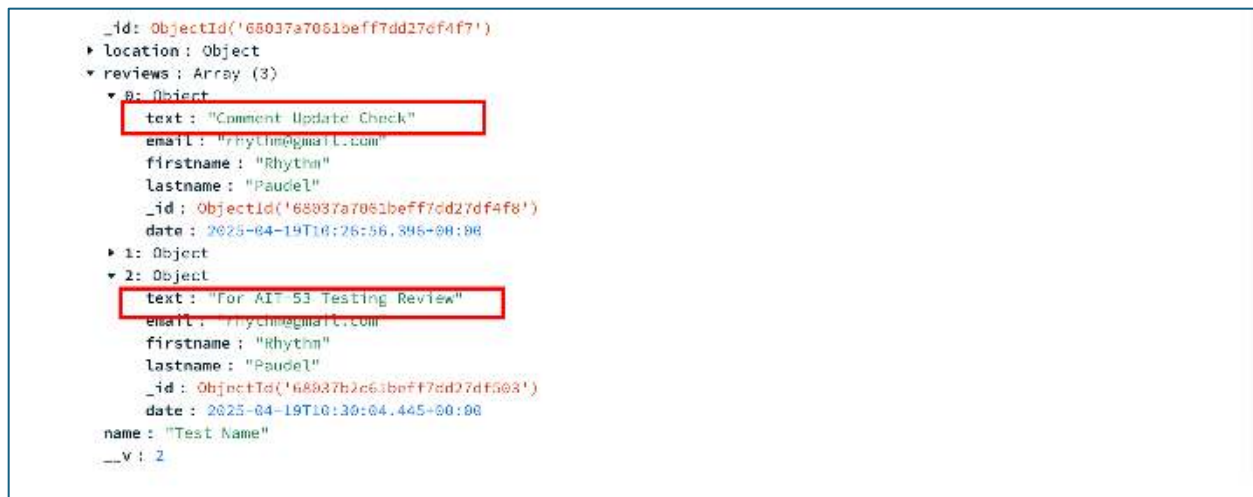


Figure 140: Comment update failed(API)

Reason: When updating comments following code in following screen shot was used. But even if the document is matched that exact comment may not be edited as just the document as a whole is matched. So additional condition was checked using \$elemMatch which finds exactly specific review is needed to be edited. The current code matches the document, however if no elemMatch is provided, first data of the array is updated.



Figure 141: Incorrect code snippet for updating comment

Correction Measure: \$elemMatch was used to fix the problem

```

try{
  const result = await Reviews.findOneAndUpdate(
    {
      "location.latitude":location.latitude,
      "location.longitude":location.longitude,
      "reviews._id":commentID,
      "reviews": {
        $elemMatch: {
          _id: commentID,
          email: email
        }
      }
    },
    {
      $set:{"reviews.$.text":commentBody}
    },
    {new:true}
  )
}

```

Figure 142: Corrected code for updating comment

Objective	To test if user is able to edit a comment via API.
Action	1) User was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to http://localhost:3500/postComments/update 3) Required field was provided 4) Request was sent to put method.
Expected result	Review should be updated in the database
Actual result	Review was updated in the database.
Conclusion	Test was successful.

Figure 143: Editing an existing comment(Unsuccessful)

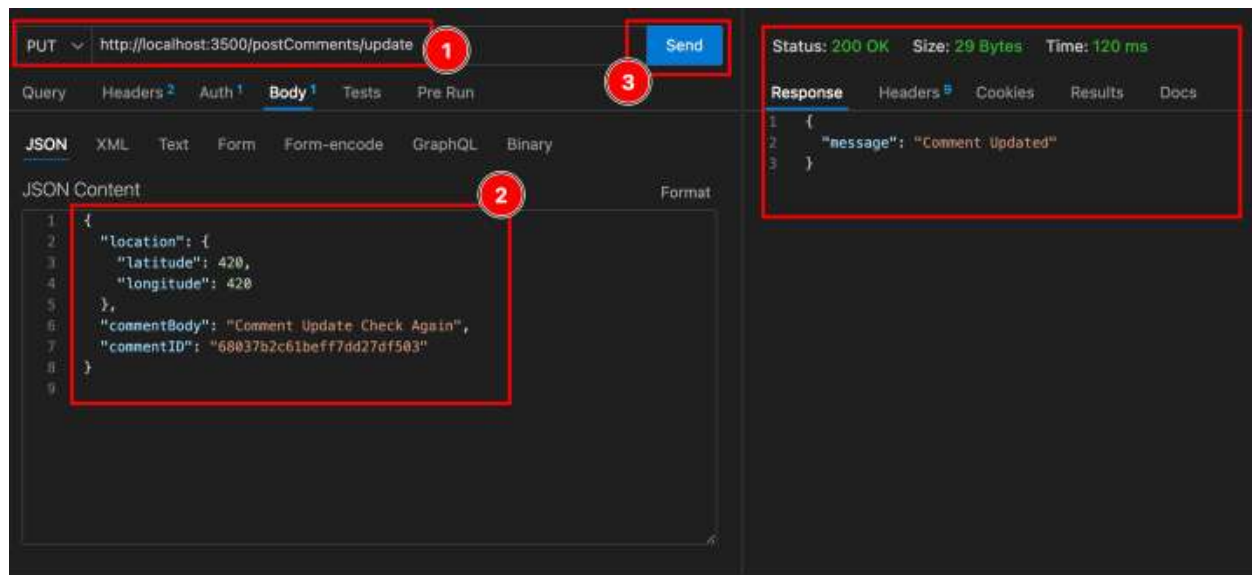


Figure 144: Updating comment after correction



Figure 145: Updated comment success in database

4.3.6. Admin tasks testing

1) AIT-31 Admin login testing via API

Objective	To test if notifications for a invalid token handled properly
Action	1) URL was set to localhost:3500/admin/login 2) Valid credentials was passed in body. Username:rhythm Password:rhythm 3) Request was sent.
Expected result	Server should sent Ok status with access token and should set cookie for refresh token
Actual result	Server snet Ok status with access token and cookie for refresh token was set.
Conclusion	Test was successful.

Table 56: Admin login testing(API)

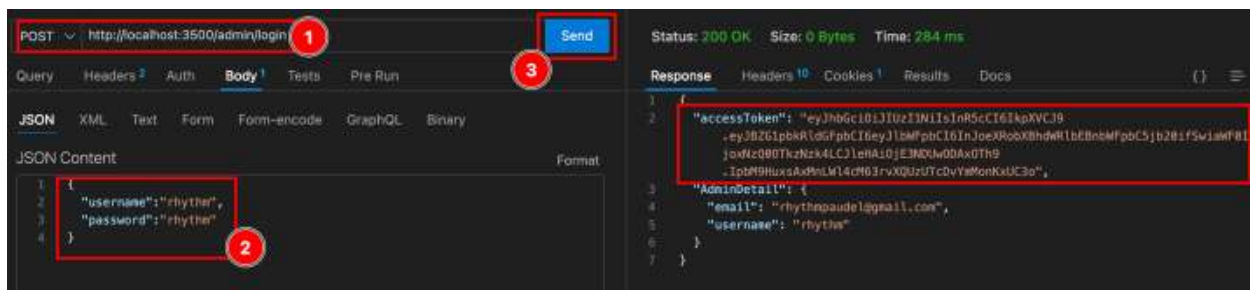


Figure 146: Admin login testing

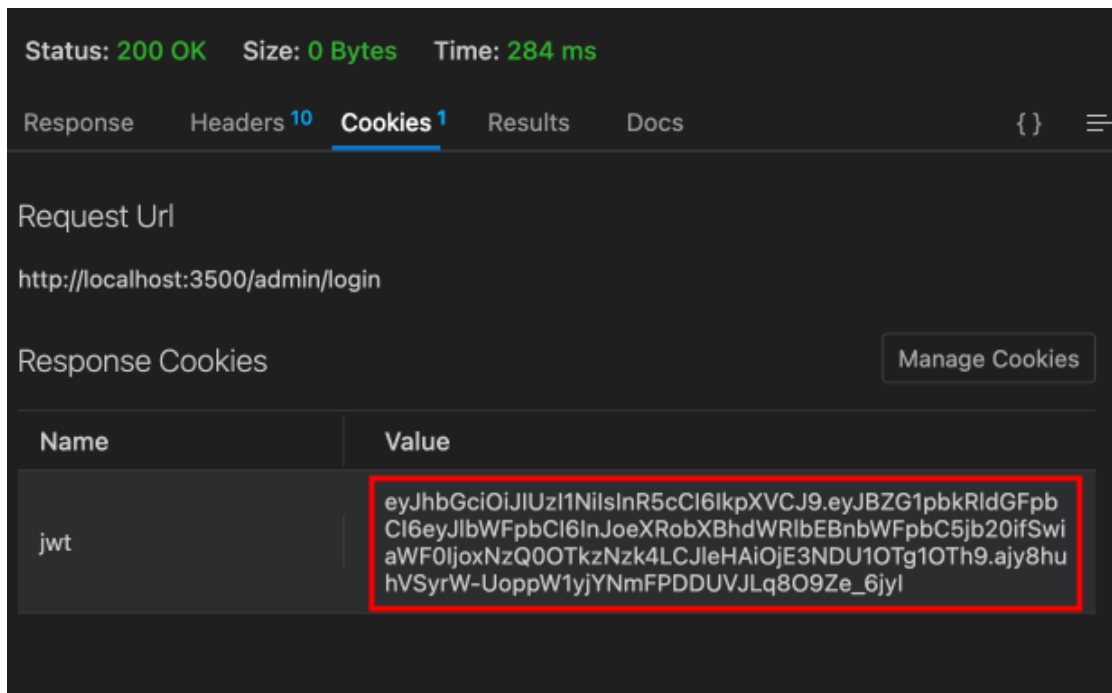


Figure 147: Refresh token response set on cookie

2) AIT-32 Admin requesting new access token

Objective	To test if admin is able to request new access token with exisiting refresh token.
Action	1) URL was set to localhost:3500/admin/refresh Cookies are automatically saved on thunderclient 2) Request was sent.
Expected result	Server should send back a new access token in response.
Actual result	Server sent back a new access token in response.
Conclusion	Test was successful.

Table 57: Admin requesting new access token

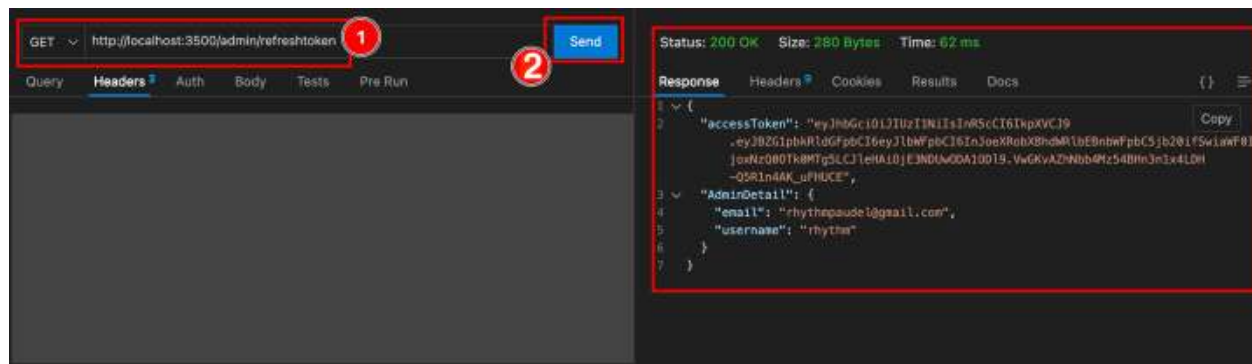


Figure 148: Requesting a new access token(Admin)

3) AIT-33 Requesting access token from mobile application user refresh token

Objective	To test if admin is able to request new access token with existing refresh token.
Action	1) URL was set to localhost:3500/login for user login 2) Valid email and password was set in body. 3) Non-encrypted token was copied 4) URL was set to localhost:3500/admin/login for setting up the cookie 5) URL was changed to localhost:3500/admin/refreshtoken 6) TourNepal app user's refresh token was pasted on cookie existing cookie 7) Request was sent
Expected result	Server should respond with proper message and error code.
Actual result	Server responded with proper message and error code.
Conclusion	Test was successful.

Table 58: Requesting access token from mobile application user refresh token

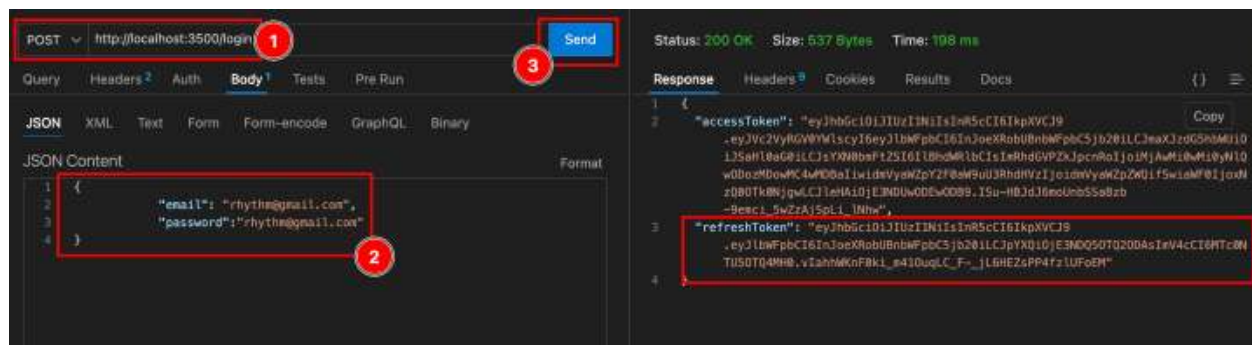


Figure 149: User log in (response with non-encrypted refresh token)

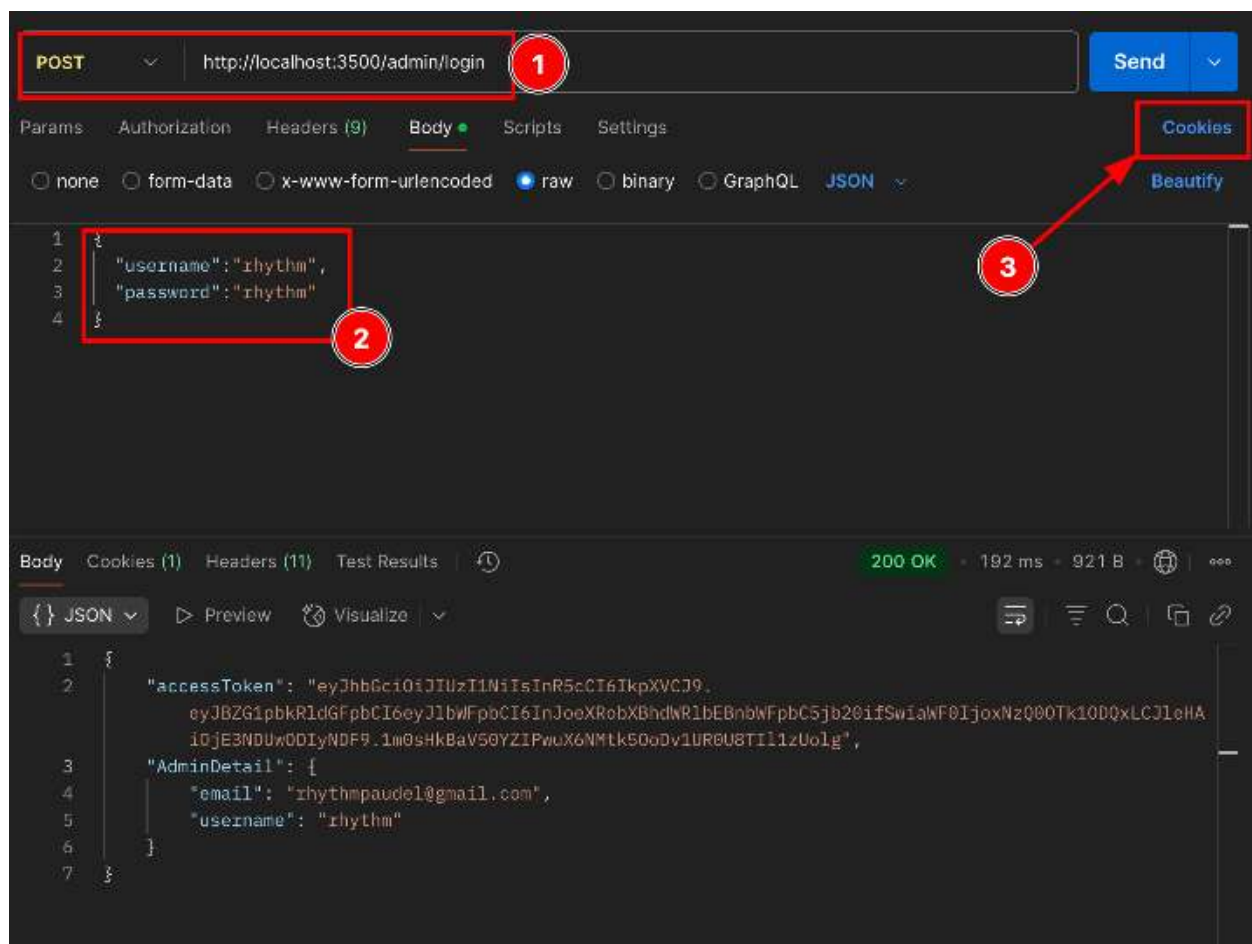


Figure 150: Admin log in for setting up cookie



Figure 151: Replacing admin refresh token with users refresh token

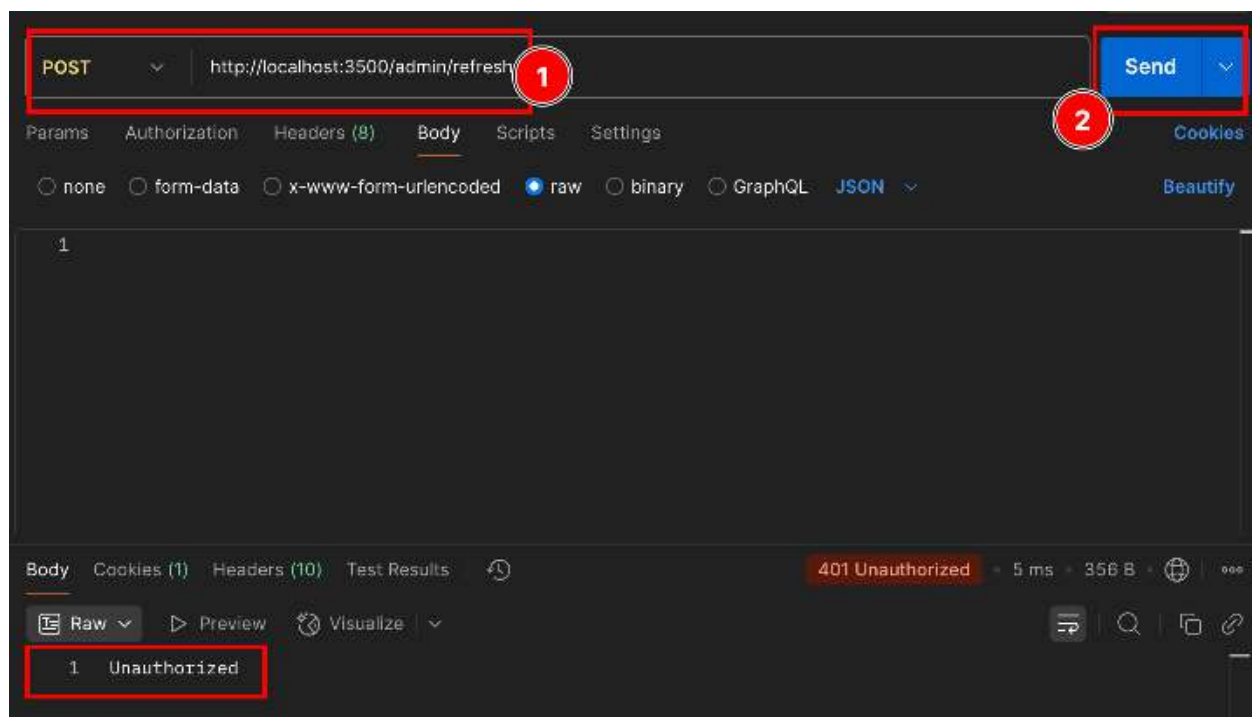


Figure 152: Requesting admin access token from users refresh token

4) AIT-34 Editing non-existent emergency contacts

Objective	To test if admin is able to edit non-existing emergency contact number.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to localhost:3500/admin/editContact 3) Non-existing id with edit values were passed. <pre>{ "id": "67503da1ded00442260ade14", "editValues": { "name": "police", "number": 100 } }</pre> 4) Request was sent
Expected result	Server should respond with proper message and error code.
Actual result	Server responded with proper message and error code.
Conclusion	Test was successful.

Table 59: Editing non-existent emergency contacts

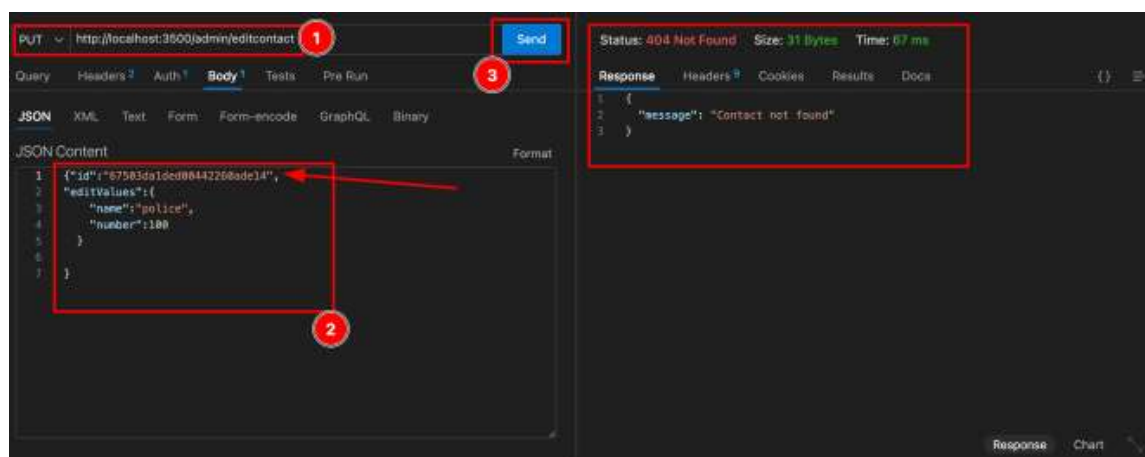


Figure 153: Editing non-existing emergency contact

5) AIT-35 Editing emergency contacts without edit values

Objective	To test if admin is able to edit existing emergency contact number without edit values.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to localhost:3500/admin/editContact 3) Id was passed. <pre>{ "id": "67503da1ded00442260ade14" }</pre> 4) Request was sent
Expected result	Server should respond with proper message and error code.
Actual result	Server responded with proper message and error code.
Conclusion	Test was successful.

Table 60: Editing emergency contacts without values

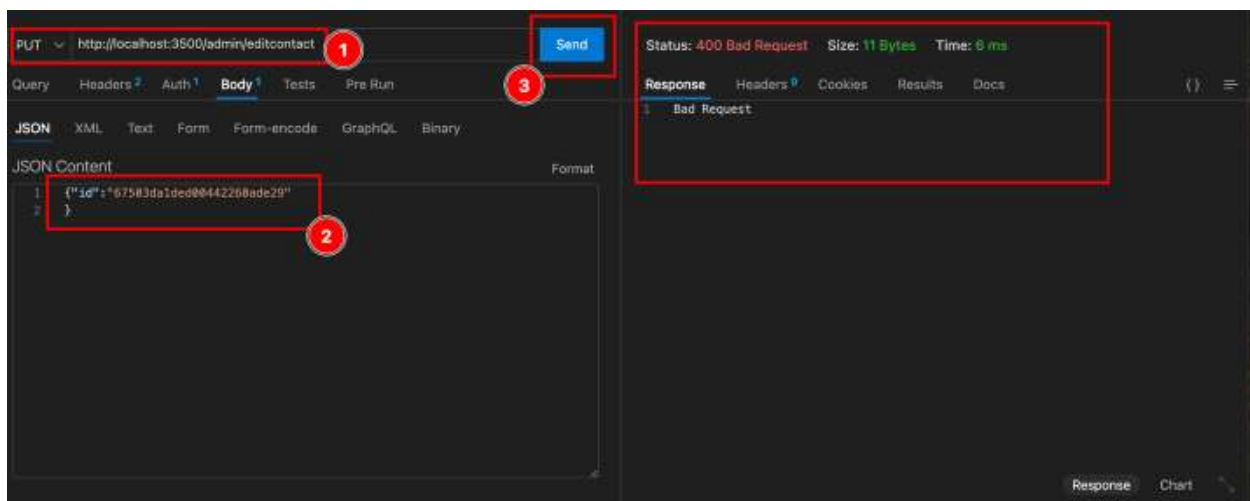


Figure 154: Editing emergency contacts without values

6) AIT-36 Editing emergency contacts

Objective	To test if admin is able to edit contact with proper values.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to localhost:3500/admin/editContact 3) Existing id with edit values were passed. <pre>{ "id": "67f23b0141a7b50fc80b480d" "editValues": { "number": 9811111111 } }</pre> 4) Request was sent
Expected result	Server should respond with proper message and error code.
Actual result	Server responded with proper message and error code.
Conclusion	Test was successful.

Table 61: Editing emergency contacts

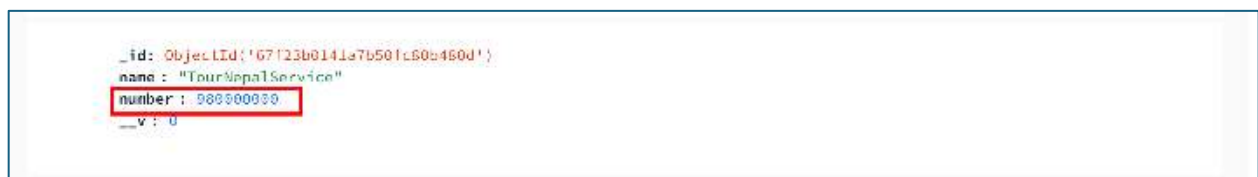


Figure 155: TourNepalService number before update

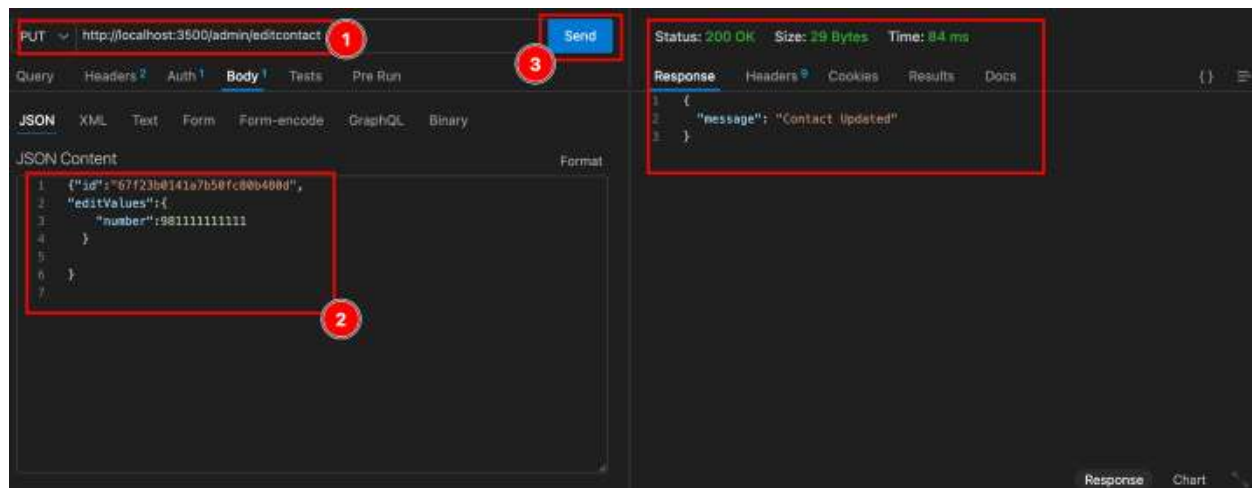


Figure 156: Updating TourNepalService contact number

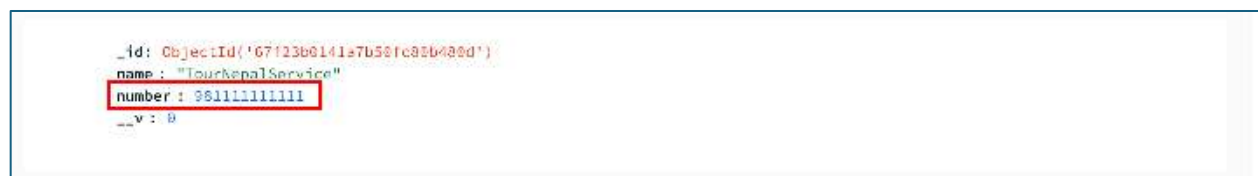


Figure 157: TourNepalService updated number

7) AIT-37 Deleting emergency contact number

Objective	To test if admin is able to delete contact with proper values
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to localhost:3500/admin/deletecontact/{id} 3) Existing id was passed in query. http://localhost:3500/admin/deletecontact/67f23b0141a7b50fc80b480d 4) Request was sent on delete method.
Expected result	Server should delete the contact.
Actual result	Server deleted the contact from the database.
Conclusion	Test was successful.

Table 62: Deleting emergency contact number

<pre> _id: ObjectId('67ef8c6e3c9781918e6c8177') name: "Police" number: 100 __v: 0 </pre>
<pre> _id: ObjectId('67ef8c7a3c9781918e6c817b') name: "Ambulance" number: 103 __v: 0 </pre>
<pre> _id: ObjectId('67f23b0141a7b50fc80b480d') name: "TourNepalService" number: 981111111111 __v: 0 </pre>

Figure 158: Contact list before deletion

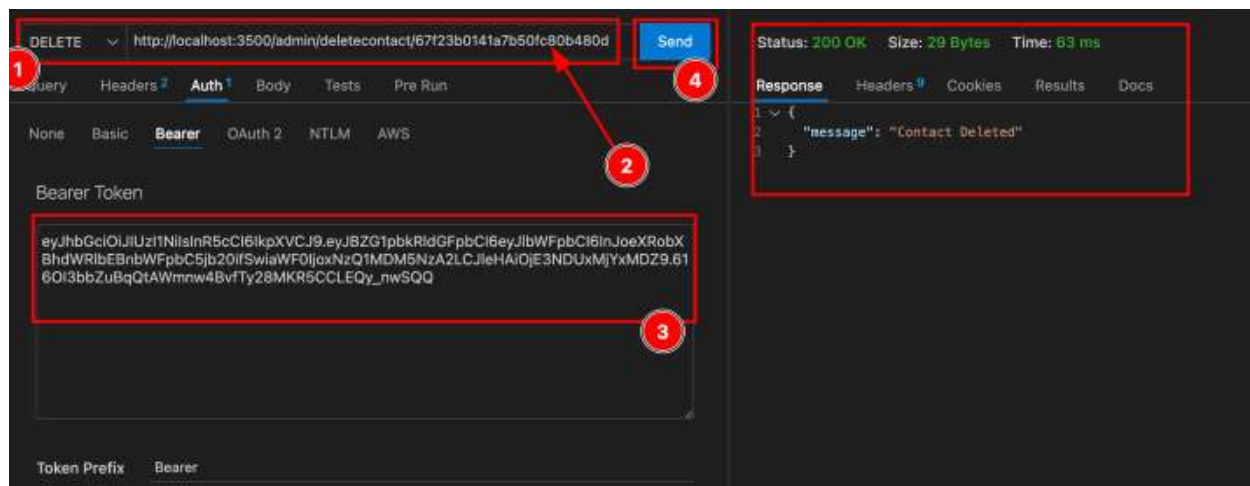


Figure 159: Deleting existing emergency contact(API)



Figure 160: Database after deletion of emergency contact

8) AIT-38 Adding a new contact via API

Objective	To test if admin is able to add a new contact with proper values.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to localhost:3500/admin/addContact 3) Body was passed with required fields <pre>{ "name": "AIT-38 Testing", "number": 11223344 }</pre> 4) Request was sent on post method.
Expected result	Server should add a new contact to the database
Actual result	Server added a new contact to the database.
Conclusion	Test was successful.

Table 63: Adding a new contact(API)

QUERY RESULTS: 1-2 OF 2 <pre> _id: ObjectId('67ef8c6e3c9781918e6c8177') name : "Police" number : 100 __v : 0 </pre> <pre> _id: ObjectId('67ef8c7a3c9781918e6c817b') name : "Ambulance" number : 103 __v : 0 </pre>

Figure 161: Before addition on new contact in database

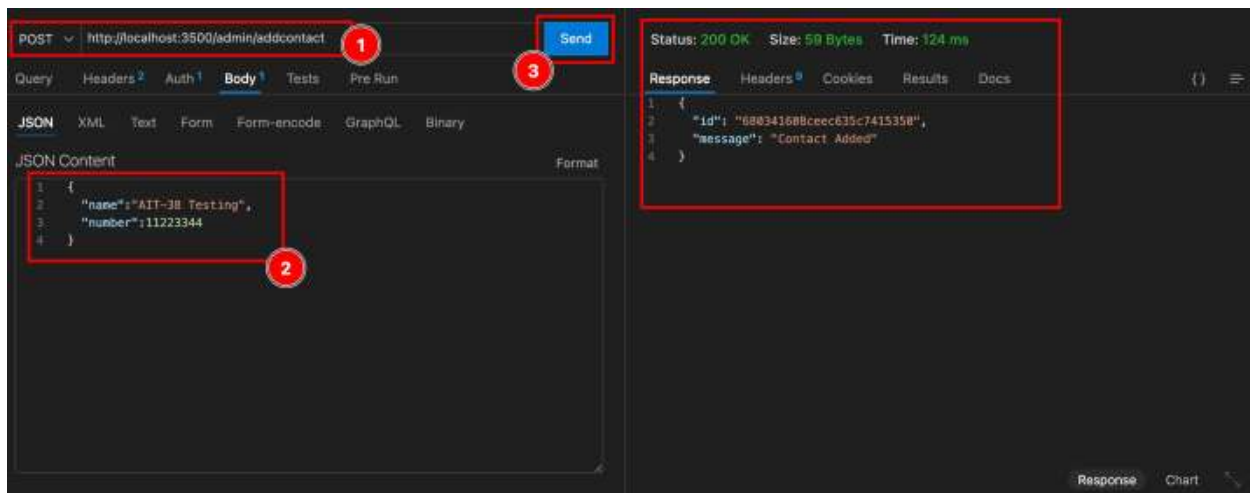


Figure 162: Adding a contact(API)



Figure 163: Updated contact list after contact addition in database

9) AIT-39 Fetching all user without access token

Objective	To test if admin is able to get all users without valid access token.
Action	1) URL was set to localhost:3500/admin/users 2) Request was sent on get method.
Expected result	Server should respond with proper error code and message
Actual result	Server responded with proper error code and message.
Conclusion	Test was successful.

Table 64: Fetching all user without access token

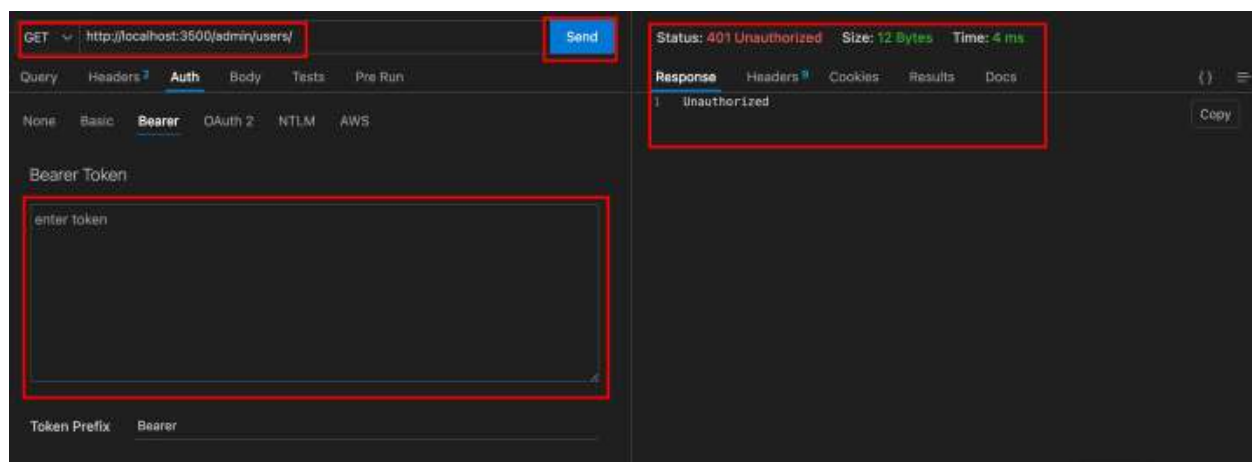


Figure 164: Getting all user without access token

10) AIT-40 Fetching all user with valid access token

Objective	To test if admin is able to retrieve all users with valid access token.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to localhost:3500/admin/users 3) Request was sent on get method.
Expected result	Server should return the list of all users in database.
Actual result	Server returned the list of all users in database.
Conclusion	Test was successful.

Table 65: Fetching all user with valid access token

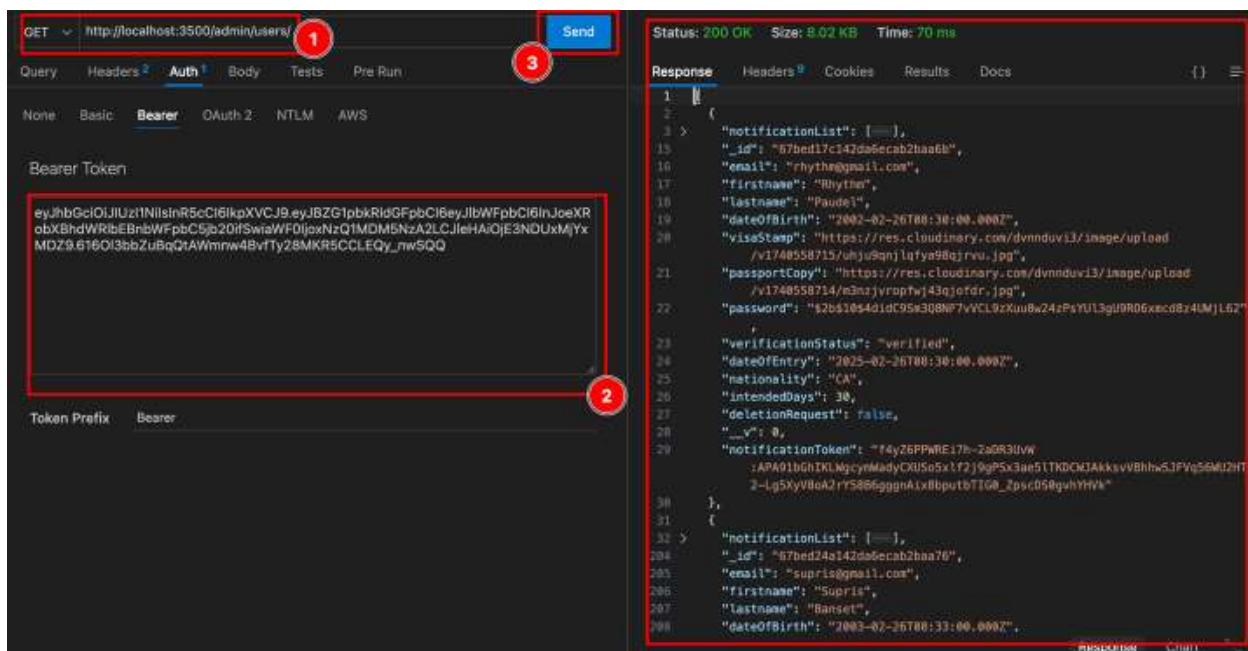


Figure 165: Retrieving all users in database

11) AIT-41 Adding a new user with valid access token

Objective	To test if admin is able to edit a user detail with valid access token.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to localhost:3500/admin/users/adduser. 3) Body was provided with all the required fields. <pre>{ "firstname": "AIT", "lastname": "41", "email": "ait-41@example.com", "dateOfBirth": "1990-05-15", "intendedDays": 30, "nationality": "US", "dateOfEntry": "2025-03-10", "password": "randomfornow", "visaStamp": "none", "passportCopy": "none" }</pre> 4) Request was sent on post method.
Expected result	Server should respond with proper error code and message
Actual result	Server responded with proper error code and message.
Conclusion	Test was successful.

Table 66: Adding new user with valid access token

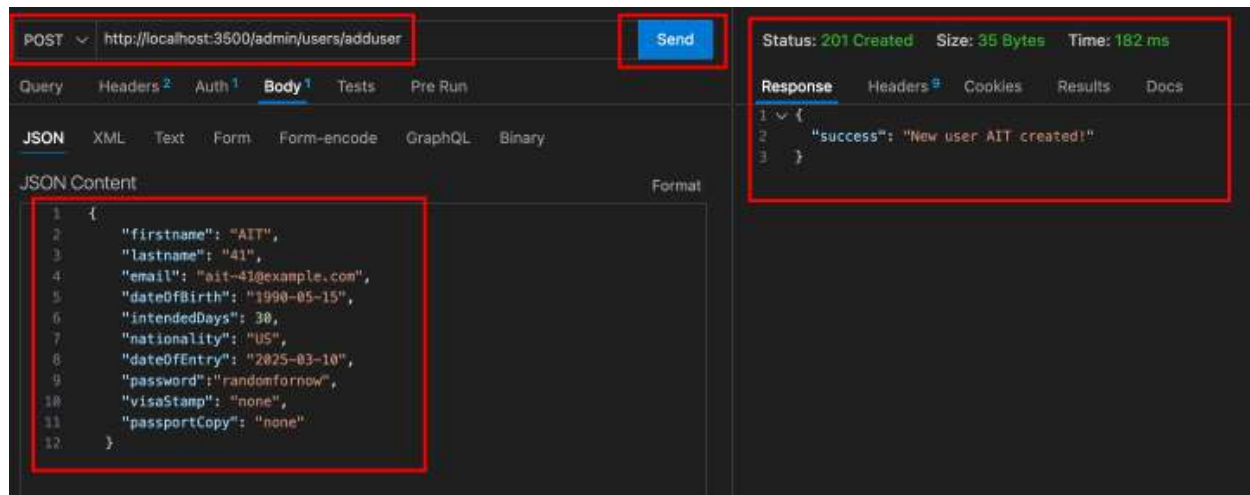


Figure 166: Creating new user via admin API



Figure 167: Addition of new user via admin API in database

12) AIT-42 Admin editing user via API

Objective	To test if admin is able to edit a user detail with valid access token.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to <code>localhost:3500/admin/users/{id}</code>. http://localhost:3500/admin/users/680347918ceec635c7415354 3) Body was provided with all the required fields. <pre>{ "user":{ "firstname": "BIT", "lastname": "42", "email": "edited-ait-41@example.com" } }</pre> 4) Request was sent on put method.
Expected result	Server should update the user in the database.
Actual result	Server updated the user in the database.
Conclusion	Test was successful.

Table 67: Admin editing user(API)

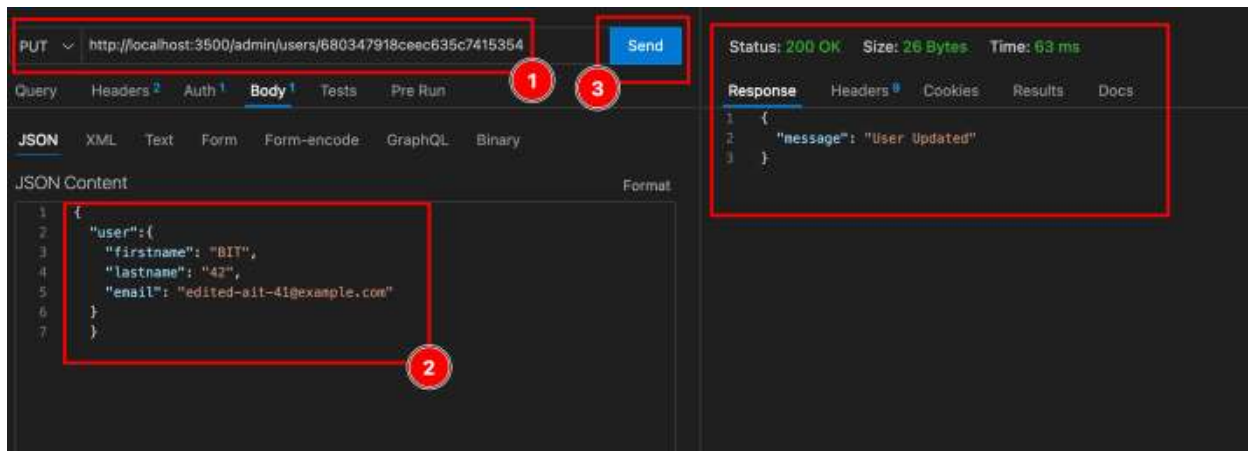


Figure 168: Editing user via admin API



Figure 169: Updated user in database after updating via admin API

13) AIT-43 Deleting a user via admin API

Objective	To test if admin is able to delete a user.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to localhost:3500/admin/users/{id}. http://localhost:3500/admin/users/680347918ceec635c7415354 3) Request was sent on delete method.
Expected result	Server should delete the user in the database.
Actual result	Server deleted the user in the database.
Conclusion	Test was successful.

Table 68: Deleting a user via admin API

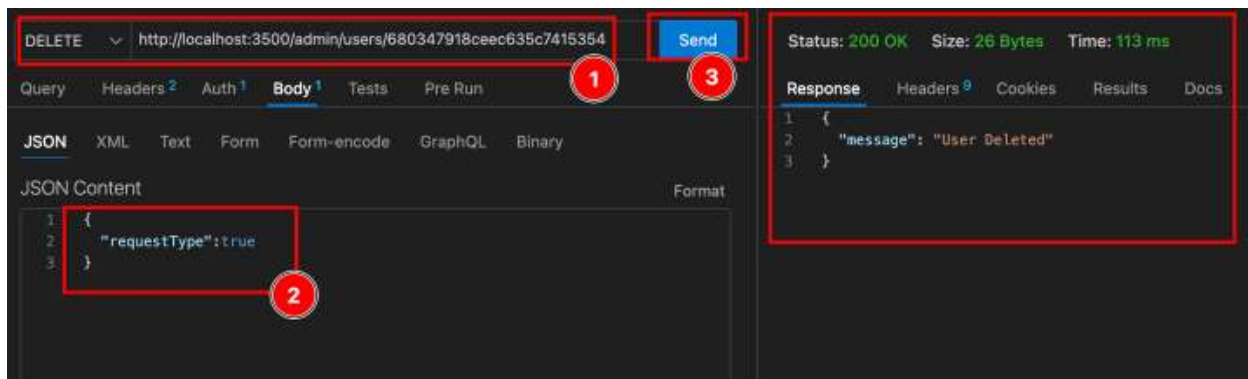


Figure 170: User deleting via admin API

14) AIT-44 Admin retrieving review of all places

Objective	To test if admin is able to get all reviews.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to <code>http://localhost:3500/admin/users/reviews/getreviews</code> 3) Request was sent on get method.
Expected result	Server should fetch all the reviews in the database
Actual result	Server fetched all the reviews in the database.
Conclusion	Test was successful.

Table 69: Admin retrieving review of all places

The screenshot displays a REST client interface with the following details:

- Query:** GET `http://localhost:3500/admin/reviews/getReviews`
- Auth:** Bearer Token
- Token Prefix:** Bearer
- Token:** `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjb2I6ImFkbWVzIiwiaWF0IjE1MDQ3ODU3LjE5eHoiOiE3NDUxMzQyNTd9.Hlq1oKZLJoR2gJkw9QqSgPeLbgSaU3n2Fn7-5a2ATXE`
- Status:** 200 OK, Size: 2.43 KB, Time: 66 ms
- Response (JSON):**

```

{
  "reviews": [
    {
      "location": {
        "latitude": -122.8795314,
        "longitude": 37.4162147
      },
      "_id": "67c9a35d3cacbeac38fb6223",
      "reviews": [
        {
          "text": "o jano@example.comsdf",
          "email": "rhythm@gmail.com",
          "firstname": "Rhythm",
          "lastname": "Paudel",
          "_id": "67c9a35d3cacbeac38fb6224",
          "date": "2025-03-06T13:38:05.522Z"
        }
      ],
      "name": "Zareen's",
      "__v": 0
    },
    {
      "location": {
        "latitude": -122.1199878,
        "longitude": 37.4878189
      },
      "_id": "67cc778b011334649d55209d",
      "reviews": [
        {
          "text": "Hello world",
          "email": "rhythm@gmail.com"
        }
      ]
    }
  ]
}

```

Figure 171: Admin reviews retrieval(API)

15) AIT-45 Admin deleting a review via API

Objective	To test if admin is able to delete a review.
Action	1) Admin was logged in and access token generated was pasted on Auth→Bearer section. 2) URL was set to http://localhost:3500/admin/reviews/deletereview/:locationid/:reviewid 3) Request was sent on get method.
Expected result	Server should fetch all the reviews in the database
Actual result	Server fetched all the reviews in the database.
Conclusion	Test was successful.

Table 70: Admin deleting a review(API)

QUERY RESULTS: 1-9 OF 9
<pre> _id: ObjectId('67c9a35d3cacbeac30fb6223') ▼ location: Object latitude: -122.0795314 longitude: 37.4162147 ▼ reviews: Array (1) ▼ 0: Object text: "ajane@example.comsdf" email: "rhythm@gmail.com" firstname: "Rhythm" lastname: "Paudel" _id: ObjectId('67c9a35d3cacbeac30fb6224') date: 2025-03-06T13:30:05.522+00:00 name: "Zareen's" __v: 0 </pre>

Figure 172: Existing review before deletion(API)

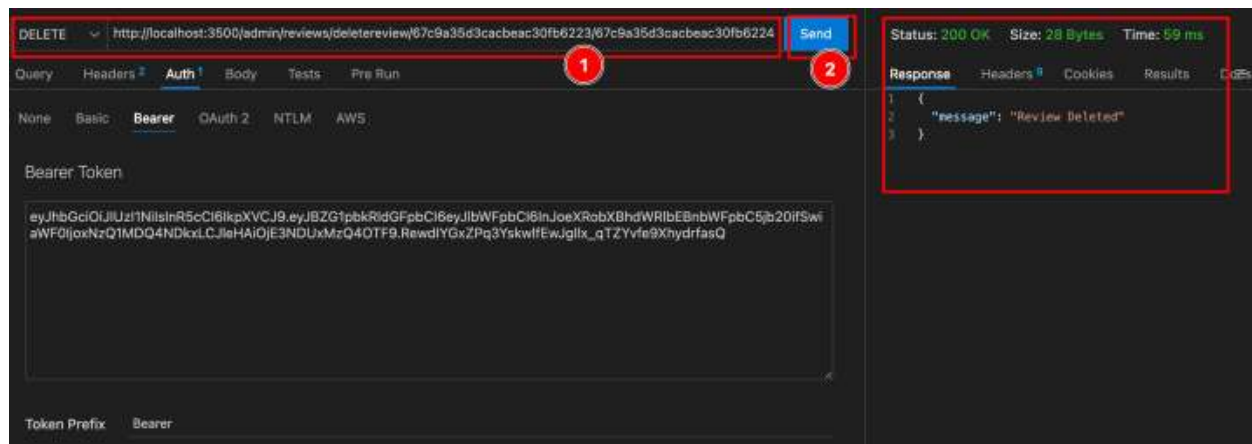


Figure 173: Deleting a review via admin API



Figure 174: After review deletion via admin API

16) AIT-48 Admin persistent login

Objective	To test if admin login is persistent after logging in.
Action	1) Logged in to the admin portal using valid credentials. 2) Refresh admin panel.
Expected result	Admin should be logged in even when the page is refreshed.
Actual result	Admin was logged in however, was not logged in when page was refreshed.
Conclusion	Test was not successful.

Figure 175: Admin persistent login(unsuccesful)

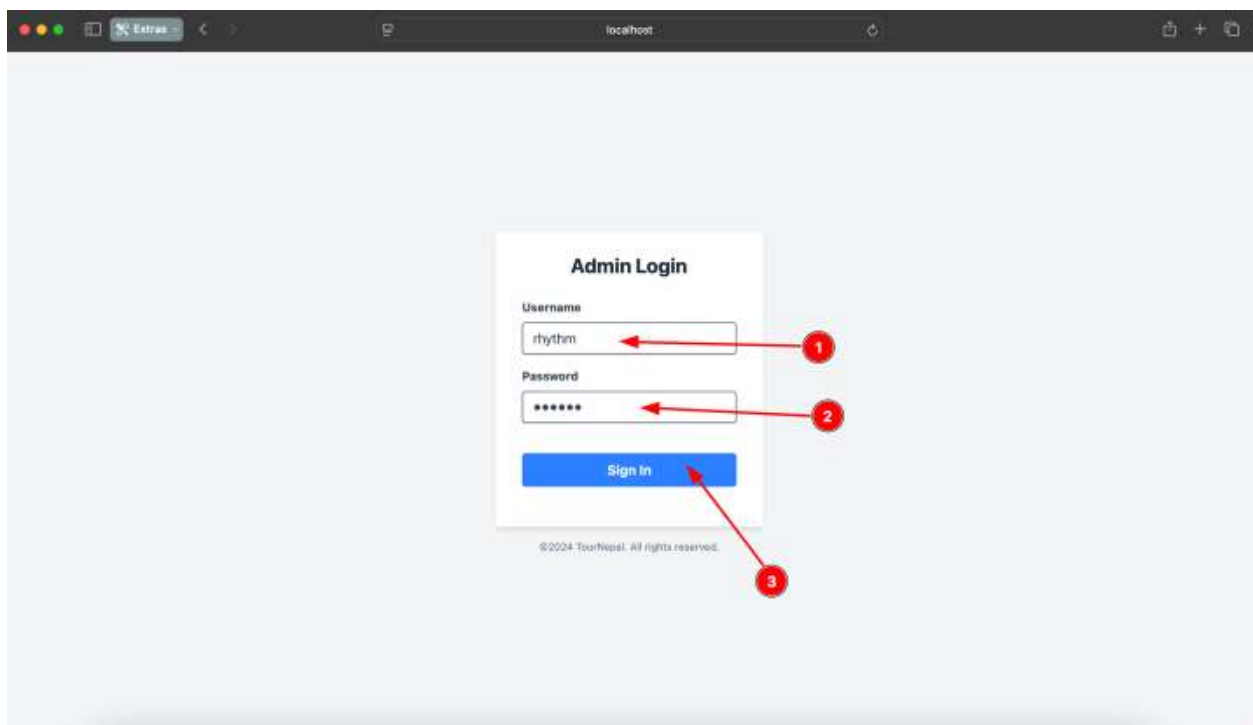


Figure 176: Admin panel login

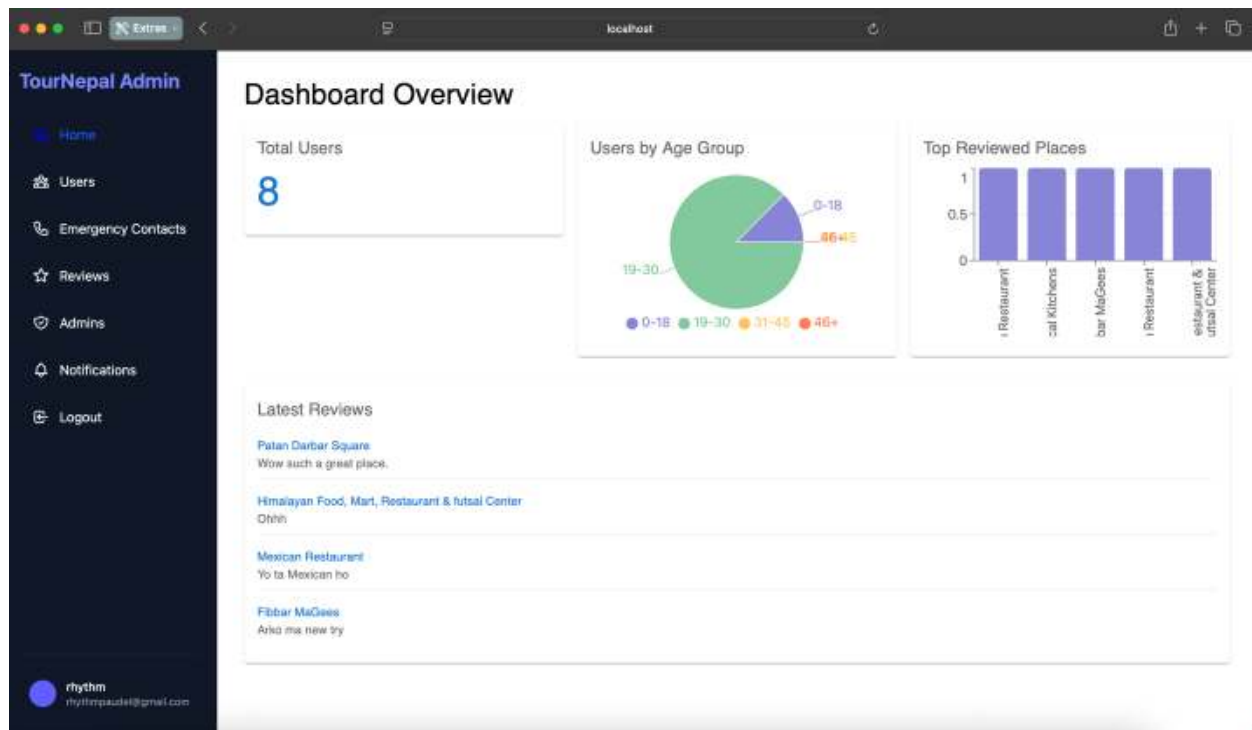


Figure 177: Logged in admin panel

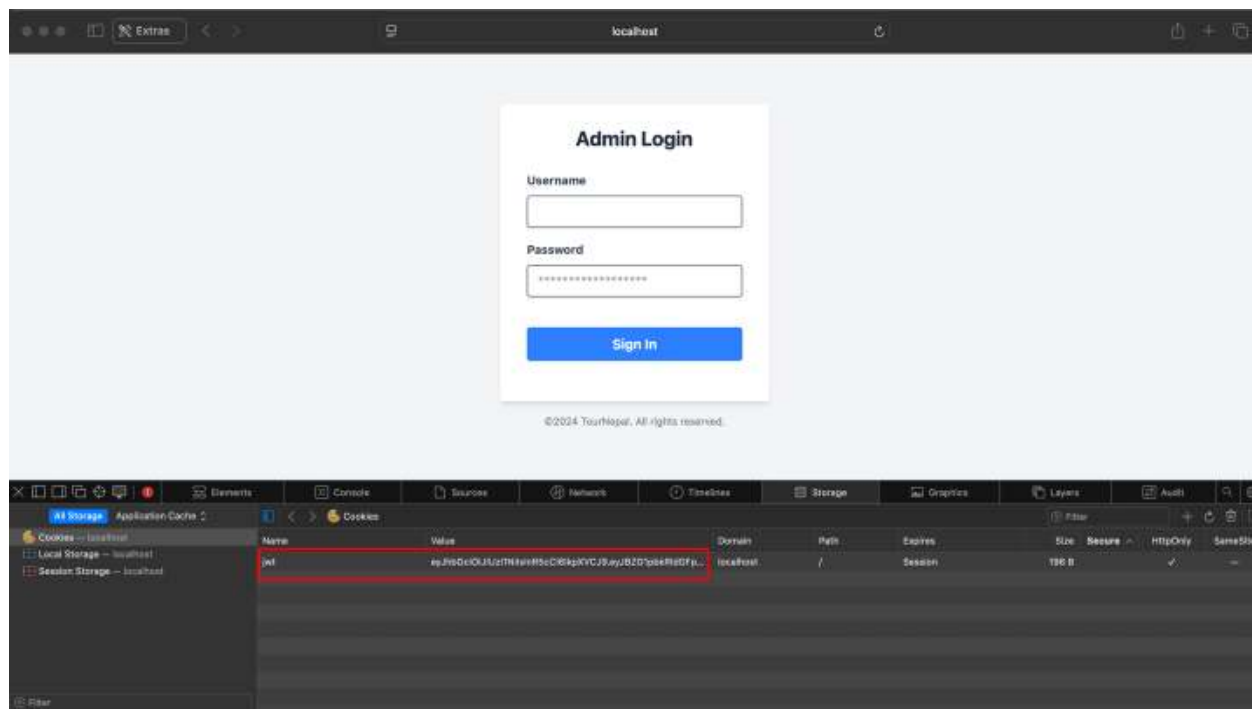


Figure 178: Login page despite cookies being saved

Reason: This was because the server was not able to parse the cookie from the browser.

Correction Measure: A package was installed to parse the cookie in the server.

Objective	To test if admin login is persistent after logging in.
Action	1) Logged in to the admin portal using valid credentials. 2) Refresh admin panel.
Expected result	Admin should be logged in even when the page is refreshed.
Actual result	Admin was logged in even when the page was refreshed.
Conclusion	Test was successful.

Figure 179: Admin persistent login

```

30
31 app.use(cookieParser());
32
33 app.use('/', require('./routes/root'));
34 app.use('/register', require('./routes/register'));
35 app.use('/login', require('./routes/login'));
36 app.use('/refresh', require('./routes/refresh'));
37 app.use('/logout', require('./routes/logout'));
38 app.use('/emergencycontacts', require('./routes/emergencyContacts'));
39 app.use('/location', require('./routes/api/location'));
40 app.use('/getReviews', require('./routes/api/getReviews'));
41 app.use('/send-notification', require('./routes/api/firebaseNotification'));
42 app.use('/admin', require('./routes/admin')); //handles all the admin related routes
43
44 //for verification of refresh token
45 app.use('/verifyJWT', require('./routes/verifyJWT'))
46

```

Figure 180: Using cookie parser in the backend server

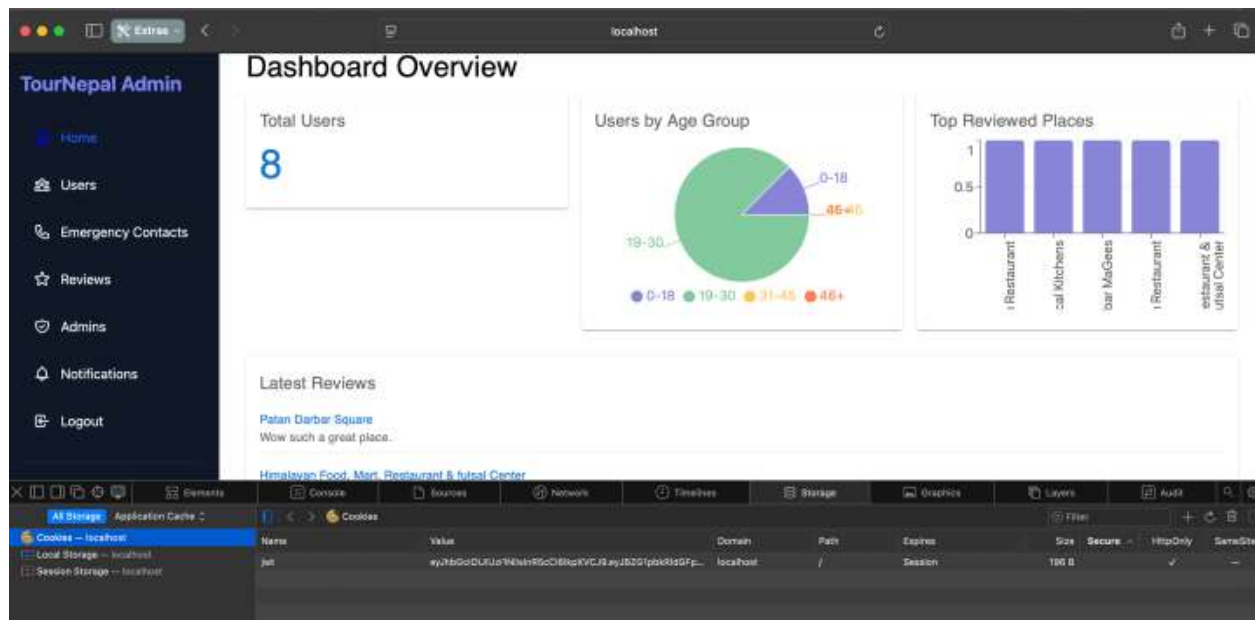


Figure 181: Admin panel homepage

4.4. System testing

4.4.1. System authentication testing

1) SYS-01 Invalid email format during registration

Objective	To test if user can provide invalid format for email during registration.
Action	1) Navigated to SignUp section. 2) Filled up first name and last name, pressed next. 3) Entered random invalid email address type. 4) Pressed next
Expected result	Registration process should not move further.
Actual result	Registration process did not move further
Conclusion	Test was successful.

Table 71: Invalid email format during registration

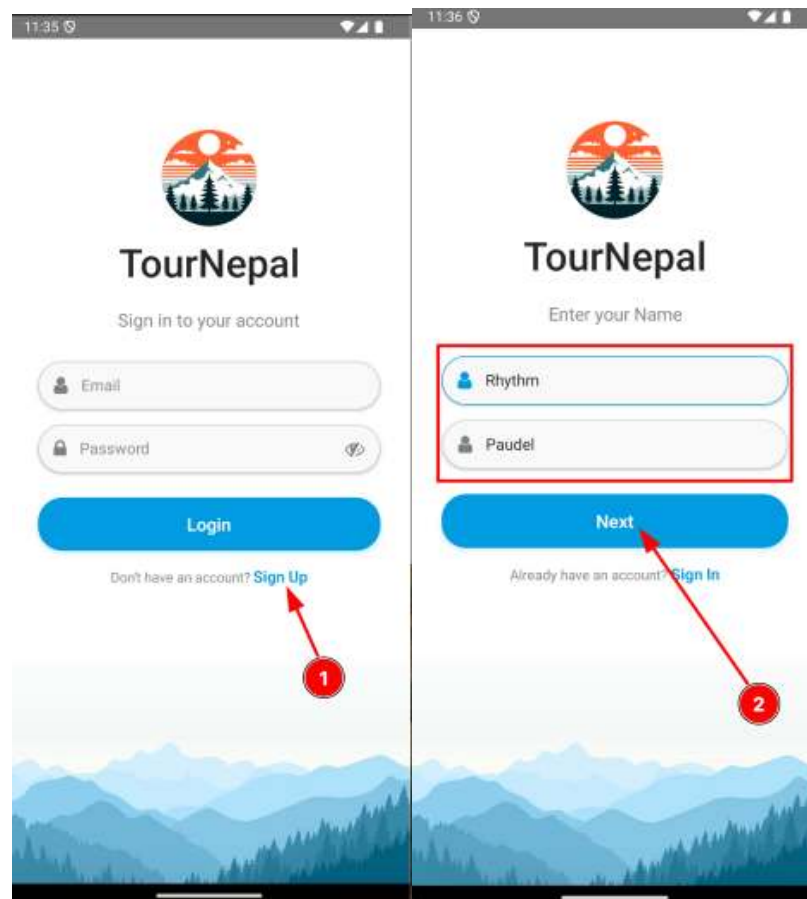


Figure 182: Registration navigation

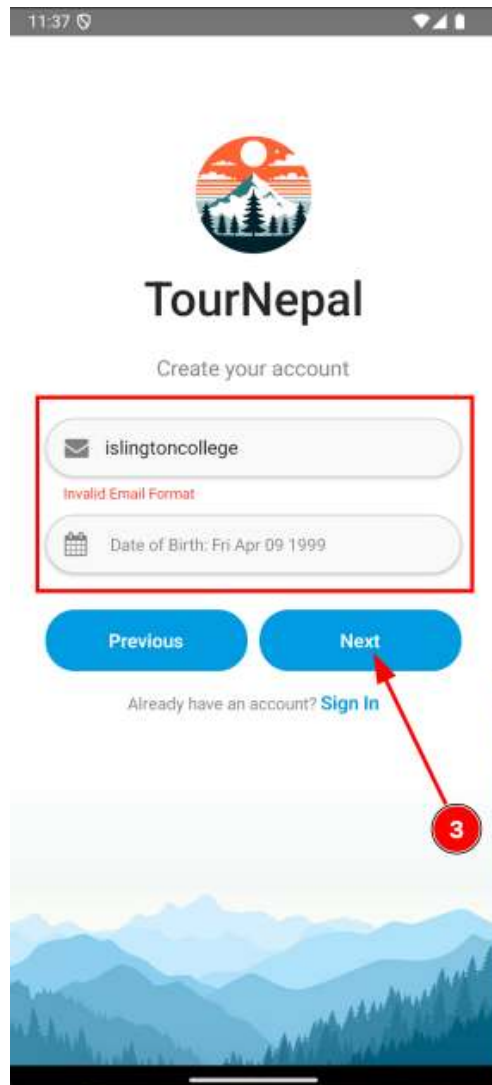


Figure 183: Invalid email format error

2) SYS-02 Empty fields during registration

Objective	To test if user can provide invalid format for email during registration.
Action	1) Navigated to Signup section. 2) Left first name and last name empty 3) Filled first name and last name, but left email empty. 4) Pressed next
Expected result	Registration process should not move further.
Actual result	Registration process did not move further
Conclusion	Test was successful.

Table 72: Empty fields during registration test



Figure 184: Registering without and with name field

The image displays two side-by-side screenshots of the TourNepal mobile application's registration screen. Both screens feature the TourNepal logo at the top, which consists of a circular icon with a mountain, trees, and a sun, followed by the text 'TourNepal' and the instruction 'Create your account'.

Left Screenshot (12:30): The registration form is shown with empty fields. The 'Email' field has a red error message 'Email is required' below it. The 'Date of Birth' field shows a placeholder date 'Sun Apr 20 2025' and a red error message 'User Must be at least 16 years old' below it. The form has two blue buttons labeled 'Previous' and 'Next'. At the bottom, it says 'Already have an account? [Sign In](#)'.

Right Screenshot (12:31): The registration form is shown with filled fields. The 'Email' field contains 'rhythmpaudel@gmail.com'. The 'Date of Birth' field shows 'Thu Apr 20 2000'. The form has two blue buttons labeled 'Previous' and 'Next'. At the bottom, it says 'Already have an account? [Sign In](#)'.

Figure 185: Registering without and with email and birth date field

3) SYS-03 Registering without non-matching password

Objective	To test if user can register without non-register password
Action	1) Navigated to Signup section. 2) Filled up fields until password screen popped up 3) Entered different password in required fields. 4) Pressed next
Expected result	Registration process should not move further.
Actual result	Registration process did not move further
Conclusion	Test was successful.

Table 73: Registering without non-matching password

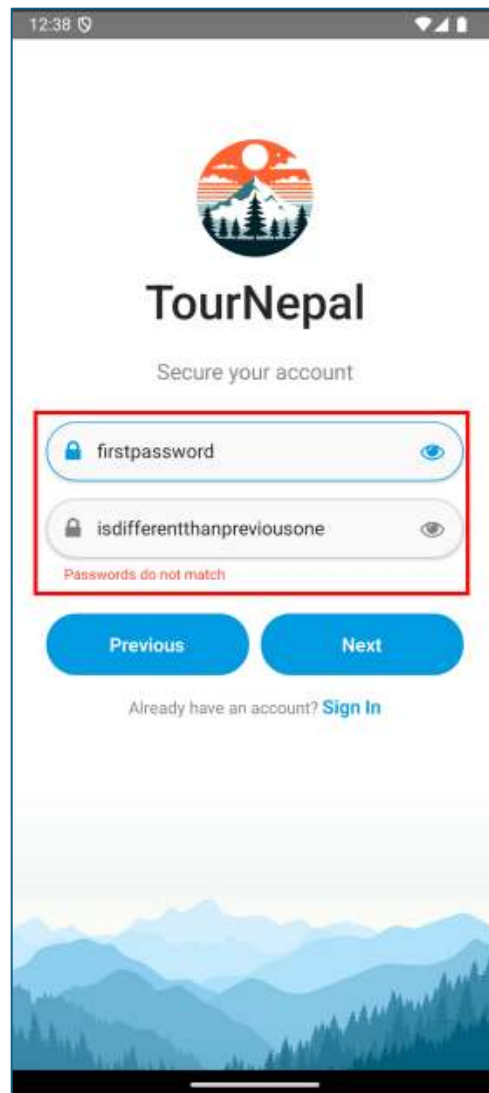


Figure 186: Continuing with different password

4) SYS-04 Registering a user

Objective	To test if user is able to register successfully with all credentials.
Action	1) Navigated to Signup section. 2) Filled up all the required credentials. 3) Camera was selected and photo was captured for visa and passport copy. 4) Country was selected from dropdown menu. 5) Clicked on Sign up
Expected result	Completion of registration process should create a user.
Actual result	Completion of registration process created a user.
Conclusion	Test was successful.

Table 74: Registration of user testing

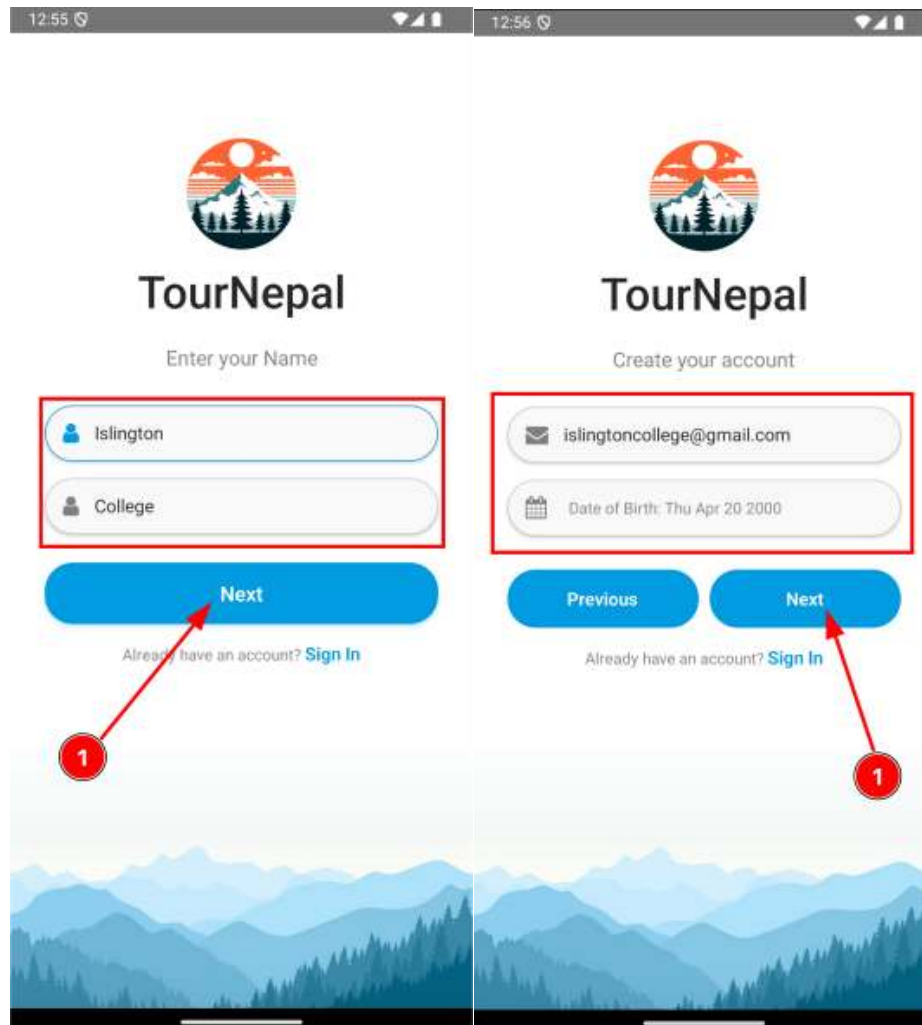


Figure 187: Registering fields(1)

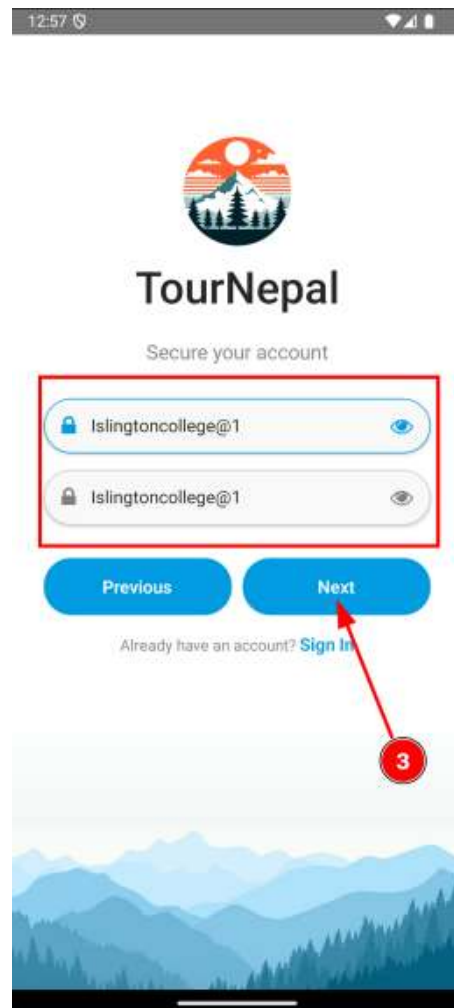


Figure 188: Registering user(2)

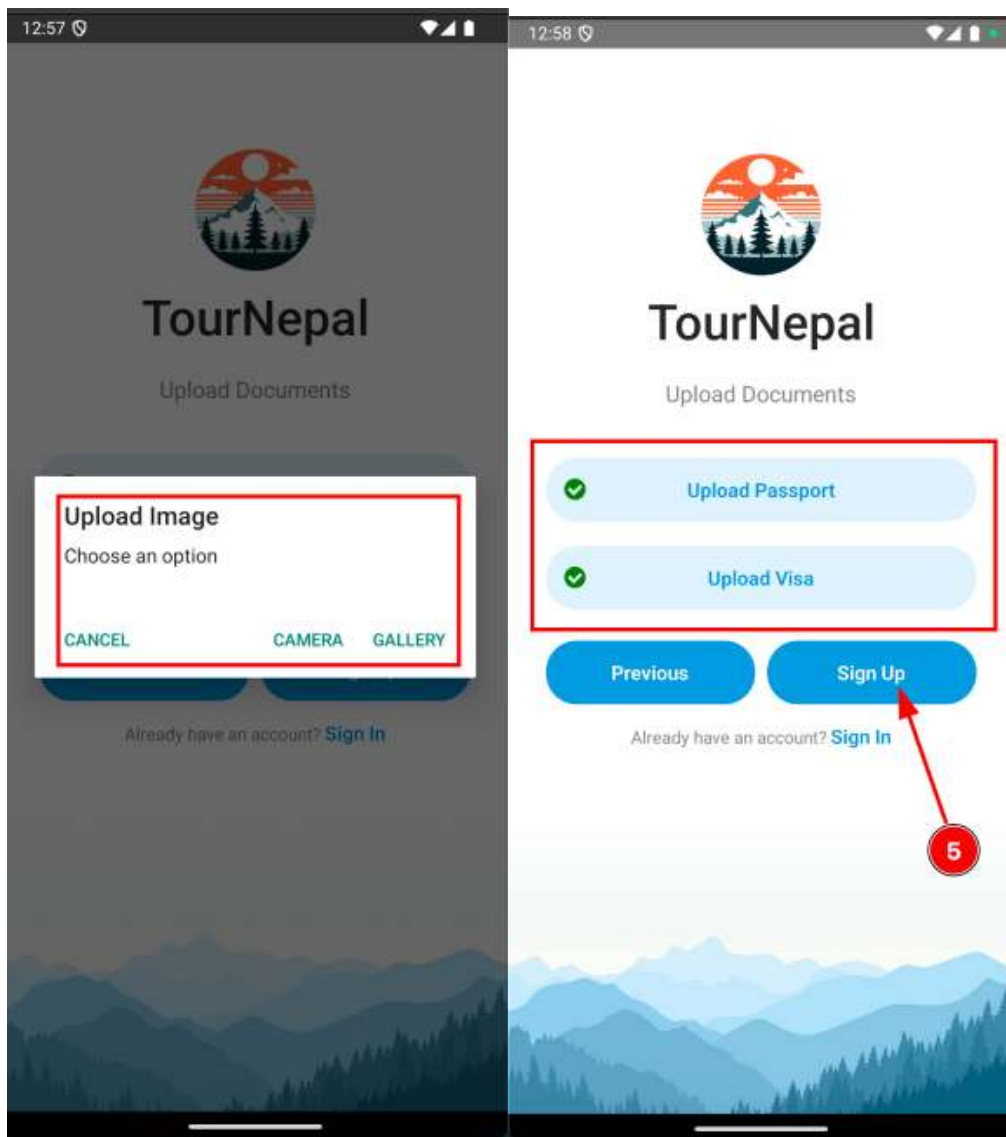


Figure 189: Registering user(3)

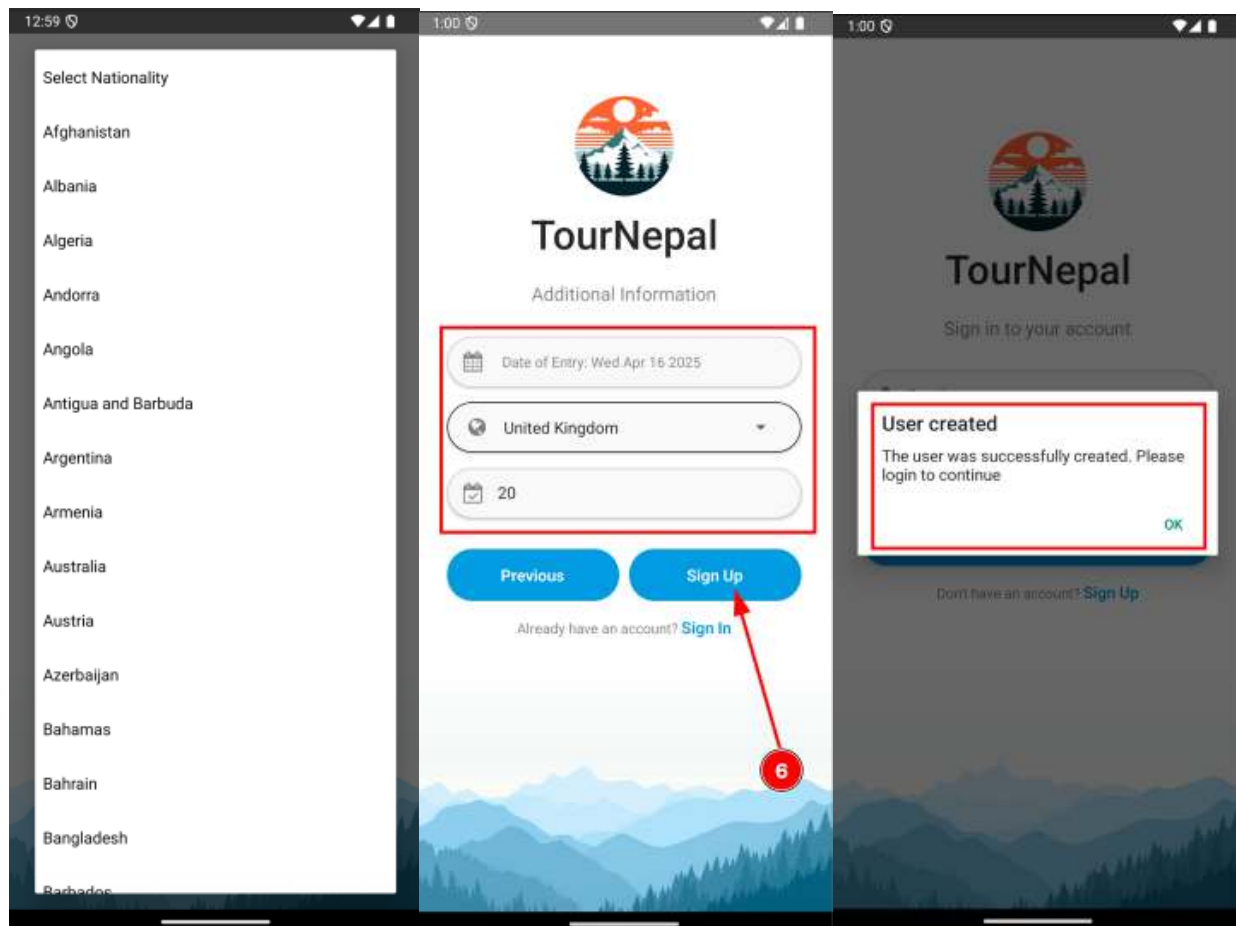


Figure 190: Registering user(4)

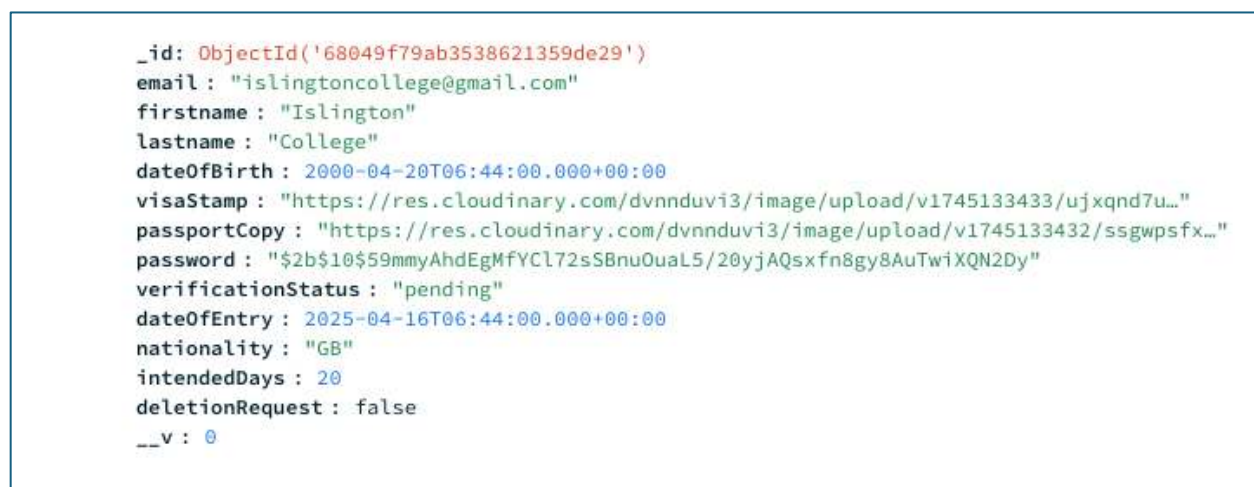


Figure 191: User registration in database

5) SYS-05 Registering with an existing email

Objective	To test if user can register with an existing email
Action	1) Navigated to Signup section. 2) Filled up all the details but used previously registered email. 3) Camera was selected and photo was captured for visa and passport copy. 4) Country was selected from dropdown menu. 5) Clicked on Sign up
Expected result	Registration should not be complete with an existing email.
Actual result	Registration was not completed with an existing email.
Conclusion	Test was successful.

Figure 192: Registering with an existing email

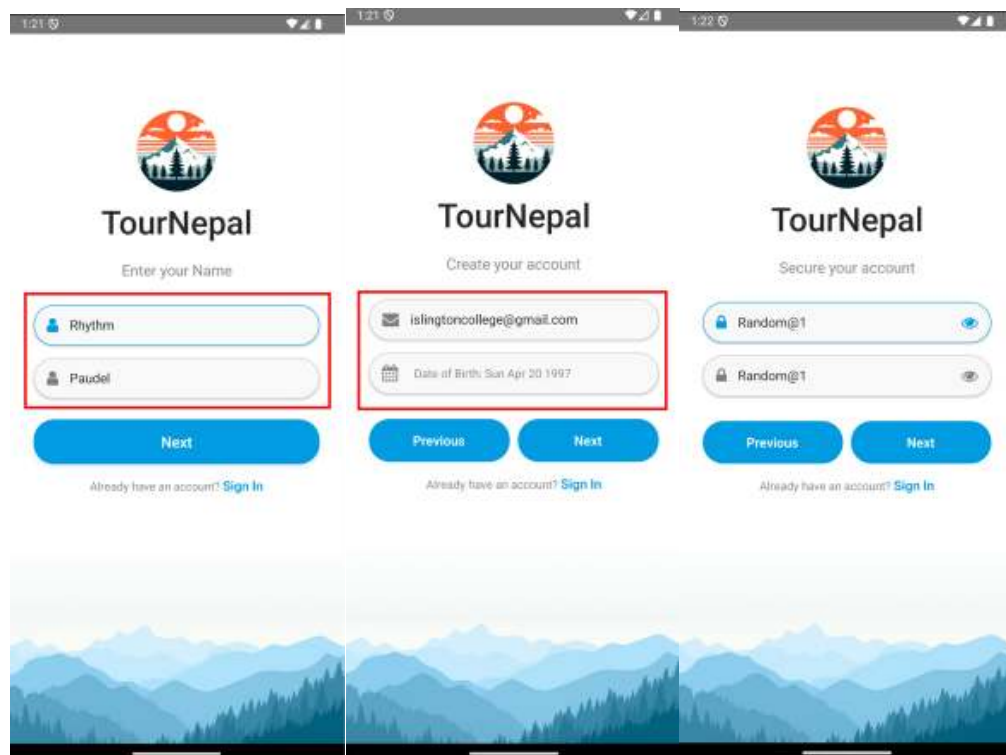


Figure 193: Registering with existing email testing(1)

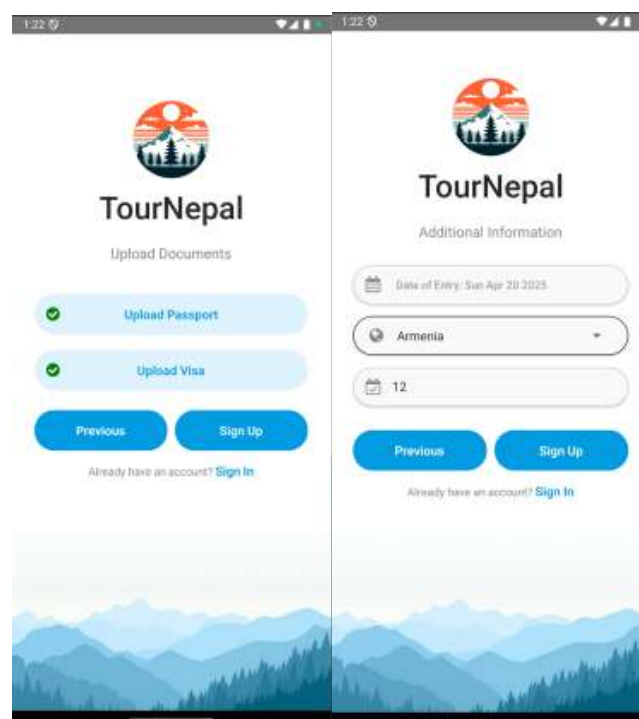


Figure 194: Registering with existing email testing(2)

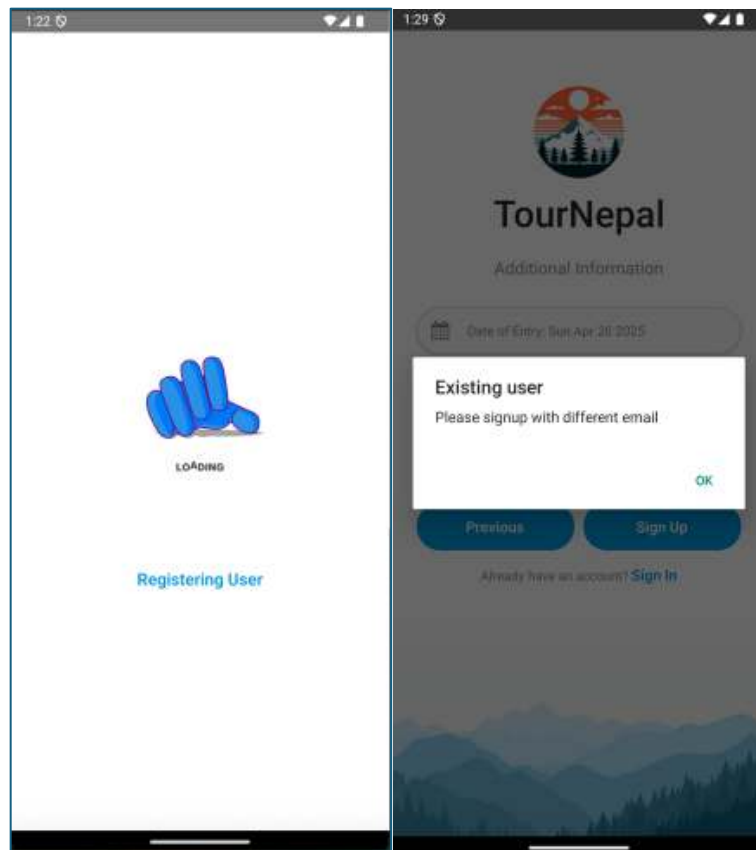


Figure 195: Registration failed with same email

6) SYS-06 Bypassing document registration

Objective	To test if user can bypass document upload procedure.
Action	1) Navigated to Signup section. 2) Filled up all the fields but pressed signup without uploading documents.
Expected result	Registration process should not move further.
Actual result	Registration process did not move further
Conclusion	Test was successful.

Table 75: Bypassing document registration testing

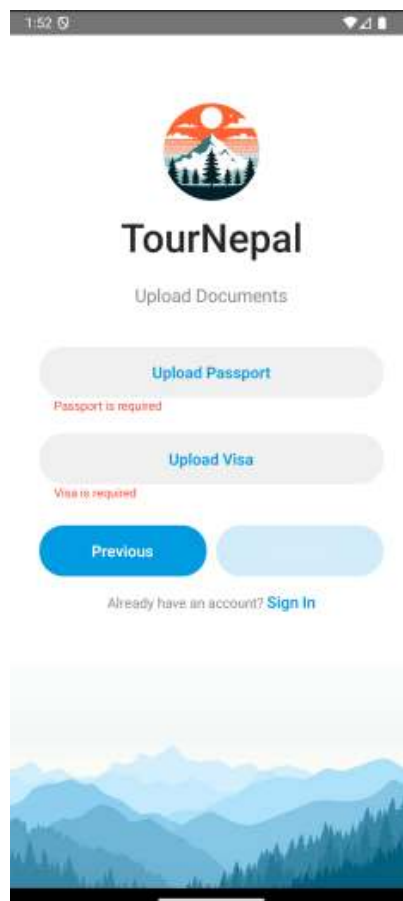


Figure 196: Bypassing document during registration

7) SYS-07 Logging in with valid credentials

Objective	To test if user can bypass document upload procedure.
Action	1) Entered valid registered user credentials
Expected result	User should be logged in and navigated to homescreen.
Actual result	User was logged in and navigated to homescreen.
Conclusion	Test was successful.

Table 76: Logging in with valid credentials test

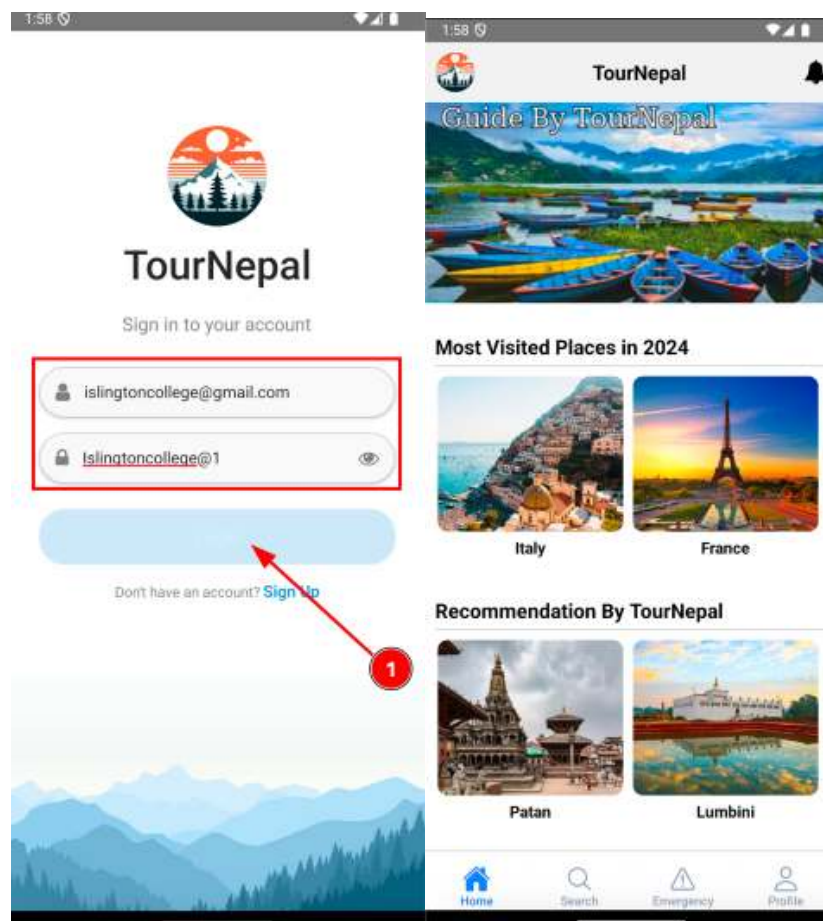


Figure 197: Logging in with valid credentials

8) SYS-08 Logging in with incorrect password

Objective	To test if registered user can login with wrong password
Action	1) Entered valid email address but entered incorrect password. 2) Pressed log in.
Expected result	User should not be logged in and should get a proper message.
Actual result	User should not be logged in and should get a proper message.
Conclusion	Test was successful.

Table 77: Logging in with incorrect password

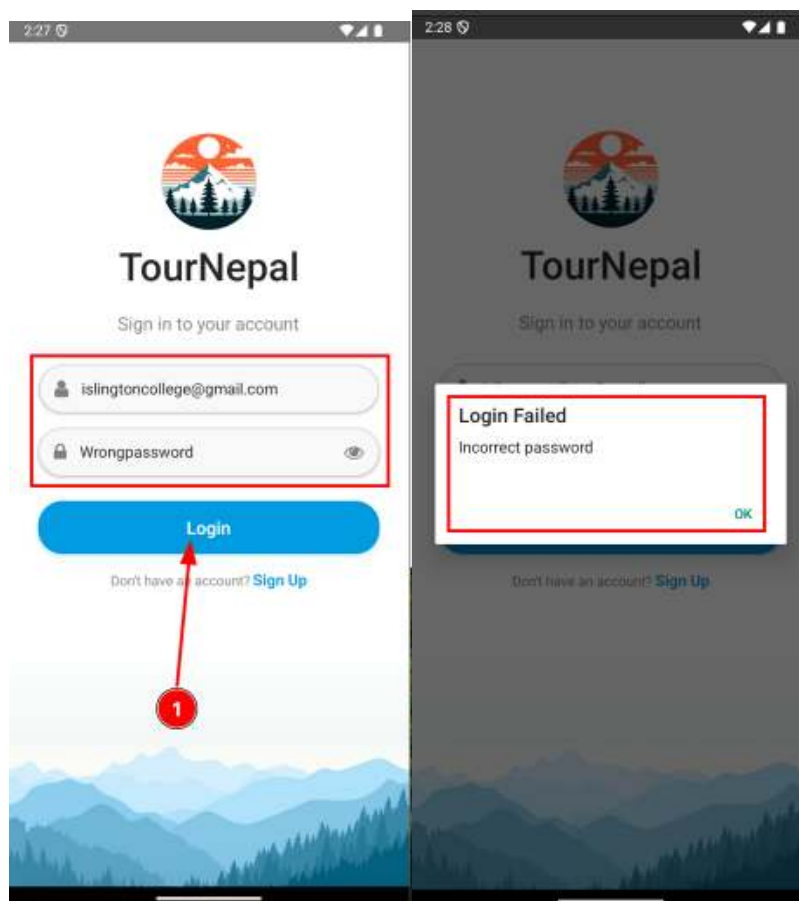


Figure 198: Incorrect password error message

9) SYS-09 Unregistered user logging in

Objective	To test if unregistered user can login
Action	1) Entered invalid email address and random password. 2) Pressed log in.
Expected result	User should be receive a proper message indicating user is not registered.
Actual result	User received a proper message indicating user was not registered.
Conclusion	Test was successful.

Table 78: Unregistered user logging in

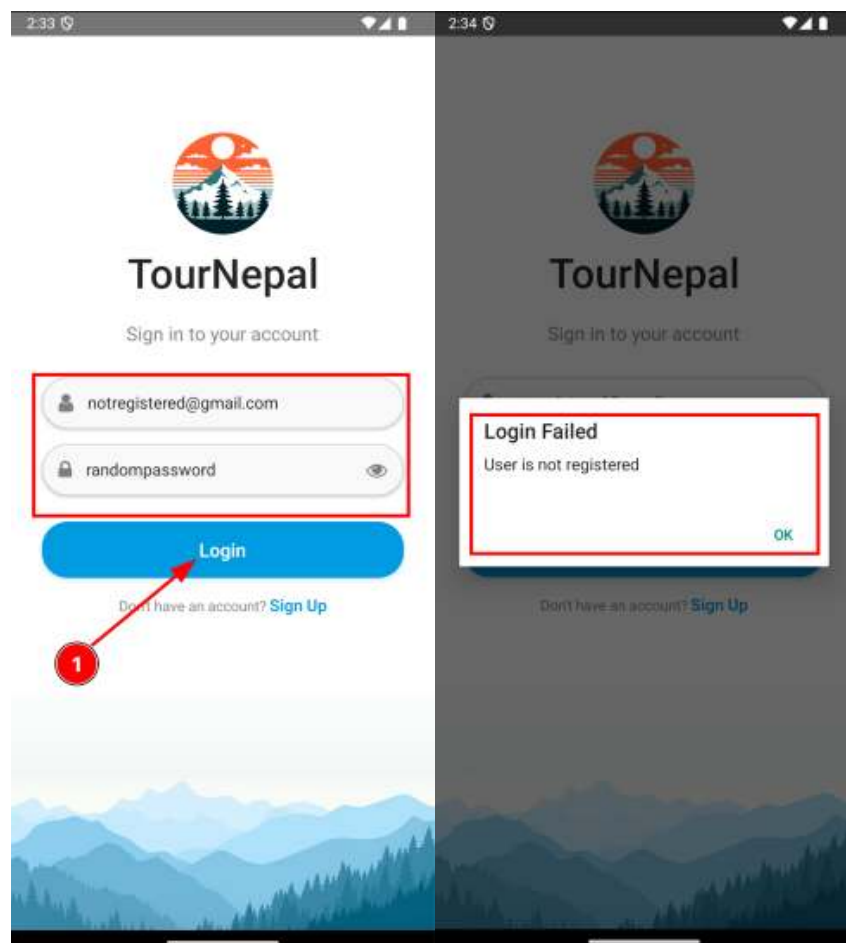


Figure 199: Unregistered user trying to login

10) SYS-10 Logging in without credentials

Objective	To test if user is able to login with empty fields
Action	1) Logged in without providing any fields for login 2) Pressed log in.
Expected result	User should be receive a proper message asking for fields to be filled.
Actual result	User received a proper message asking for fields to be filled.
Conclusion	Test was successful.

Table 79: Logging in without credentials

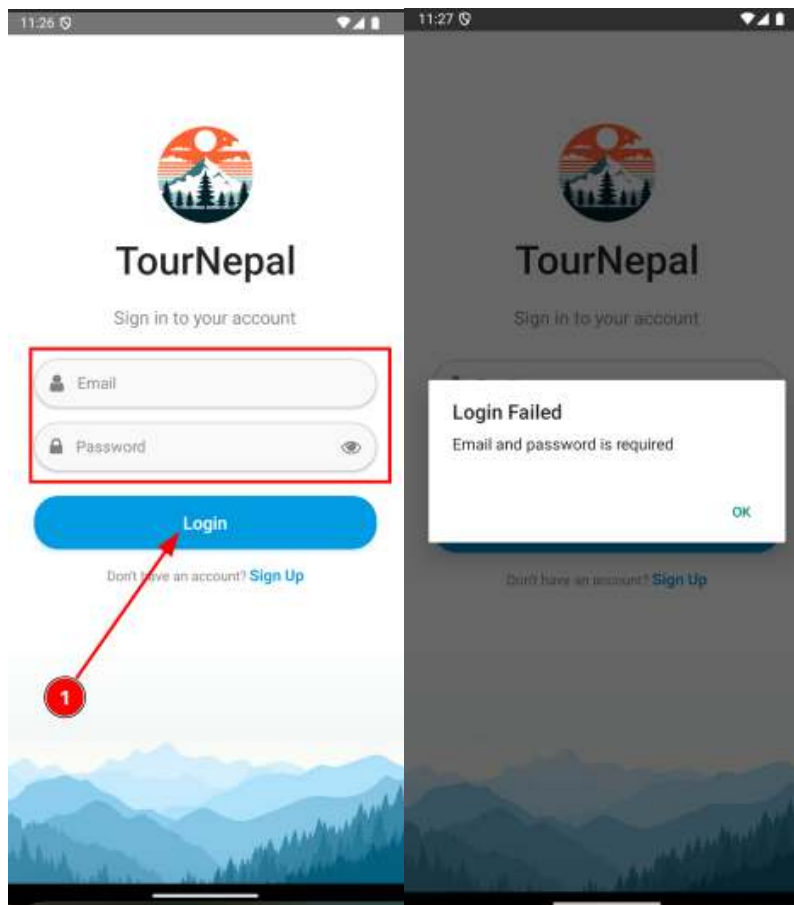


Figure 200: Login without any fields

11) SYS-11 Persistent login in mobile application

Objective	To test if user stays logged in after re-opening the application
Action	1) Logged in using valid credentials. 2) Closed the application. 3) Reopened the application.
Expected result	User should not have to re-login to the application after re-opening the application.
Actual result	User did not have to re-login to the application after re-opening the application.
Conclusion	Test was successful.

Table 80: Persistent login in mobile application

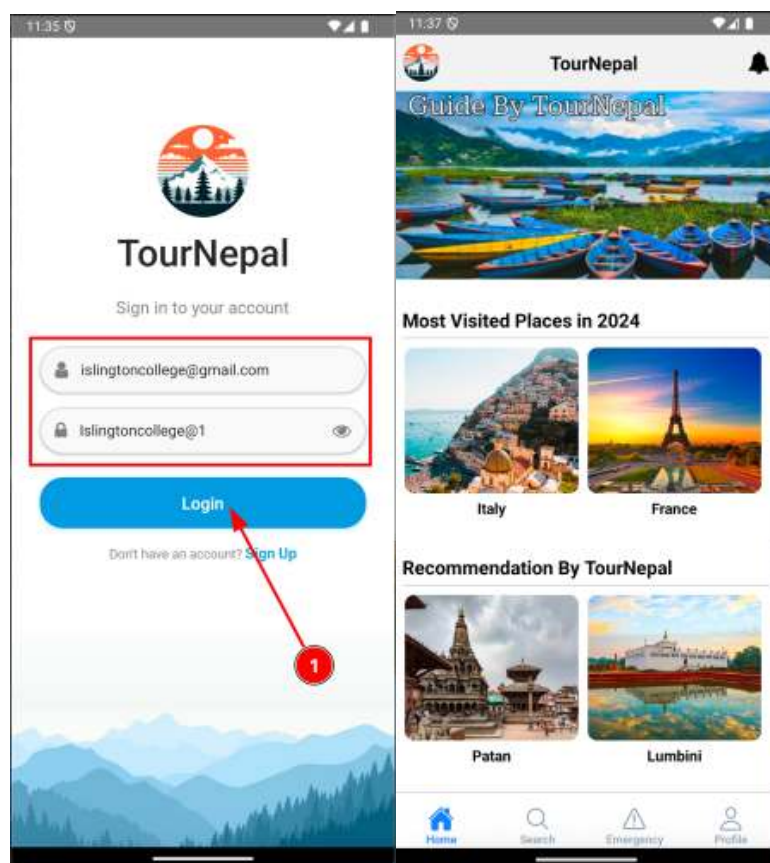


Figure 201: Logging in (test for persistent login)

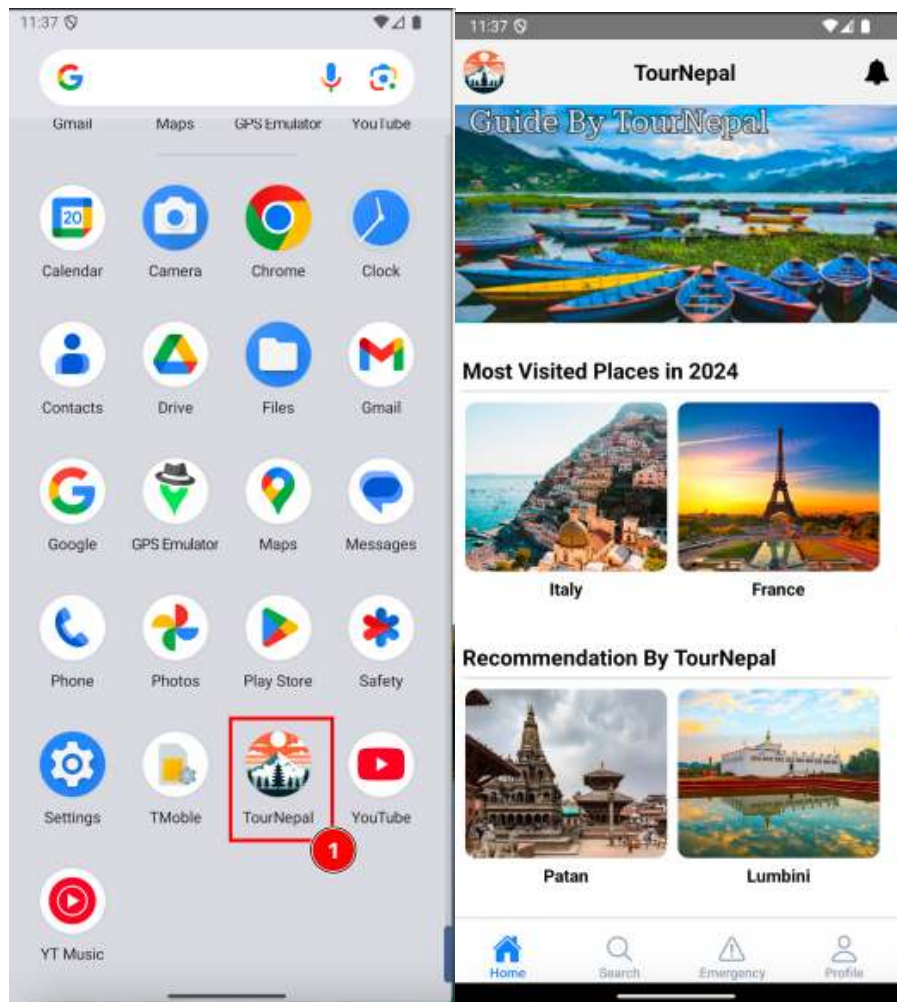


Figure 202: Persistent login

4.4.2. Location tasks testing

1) SYS-12 Viewing restaurants in default settings

Objective	To test if user is able to view restaurants (default destination type) within 5 K.M. (default radius type)
Action	1) Navigated to search section from the bottom navigation bar.
Expected result	User should be able to view restaurants near user's device location within 5K.M of radius.
Actual result	User was able to view restaurants near user's device location within 5K.M of radius.
Conclusion	Test was successful.

Table 81: Viewing restaurants in default settings

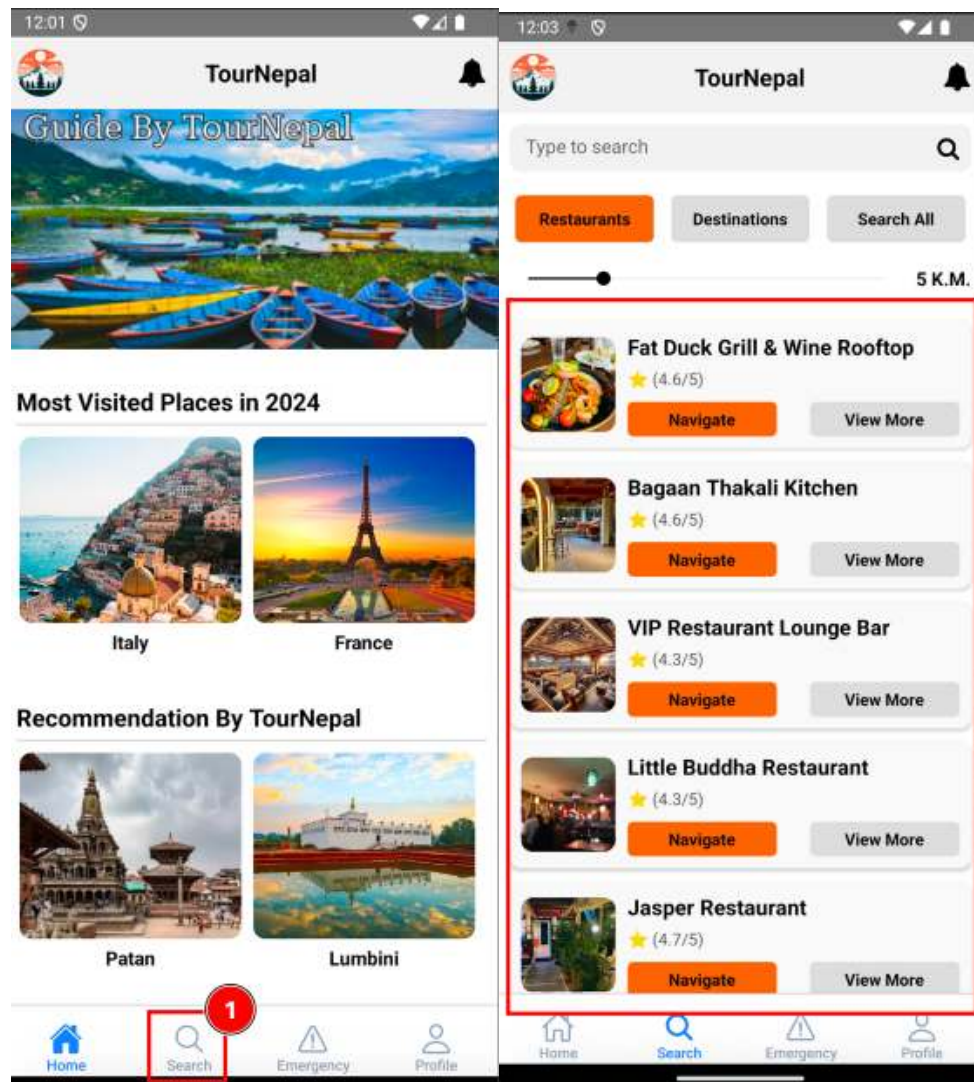


Figure 203: Default destinations search

2) SYS-13 Refreshing radius slider to update restaurants

Objective	To test if user is able to get newer restaurants when changing radius slider.
Action	1) Navigated to search section from the bottom navigation bar. 2) Changed radius slider, reducing radius to 3K.M
Expected result	User should be able to view restaurants near user's device location within 3K.M of radius.
Actual result	User was able to view restaurants near user's device location within 3K.M of radius.
Conclusion	Test was successful.

Table 82: Refreshing radius slider to update restaurants

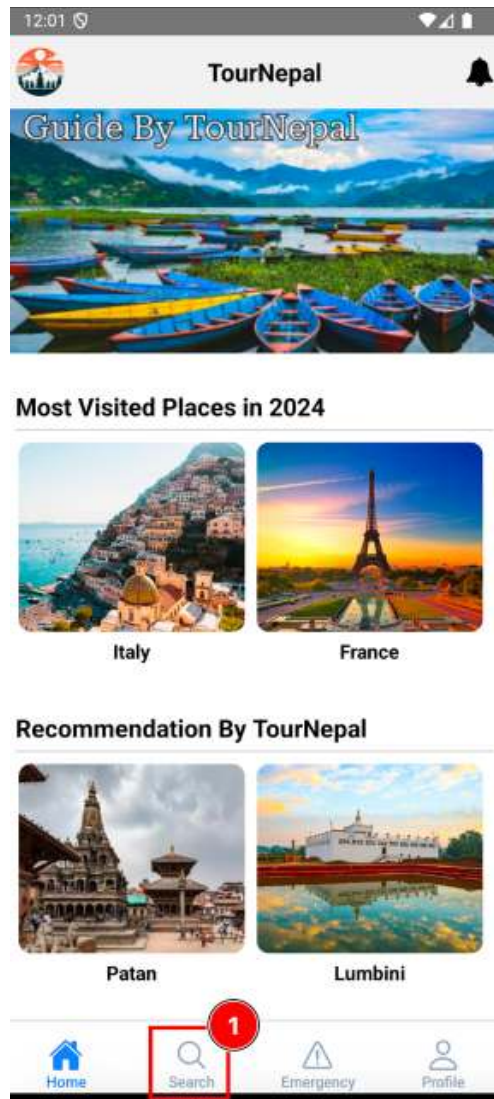


Figure 204: Navigating to search section

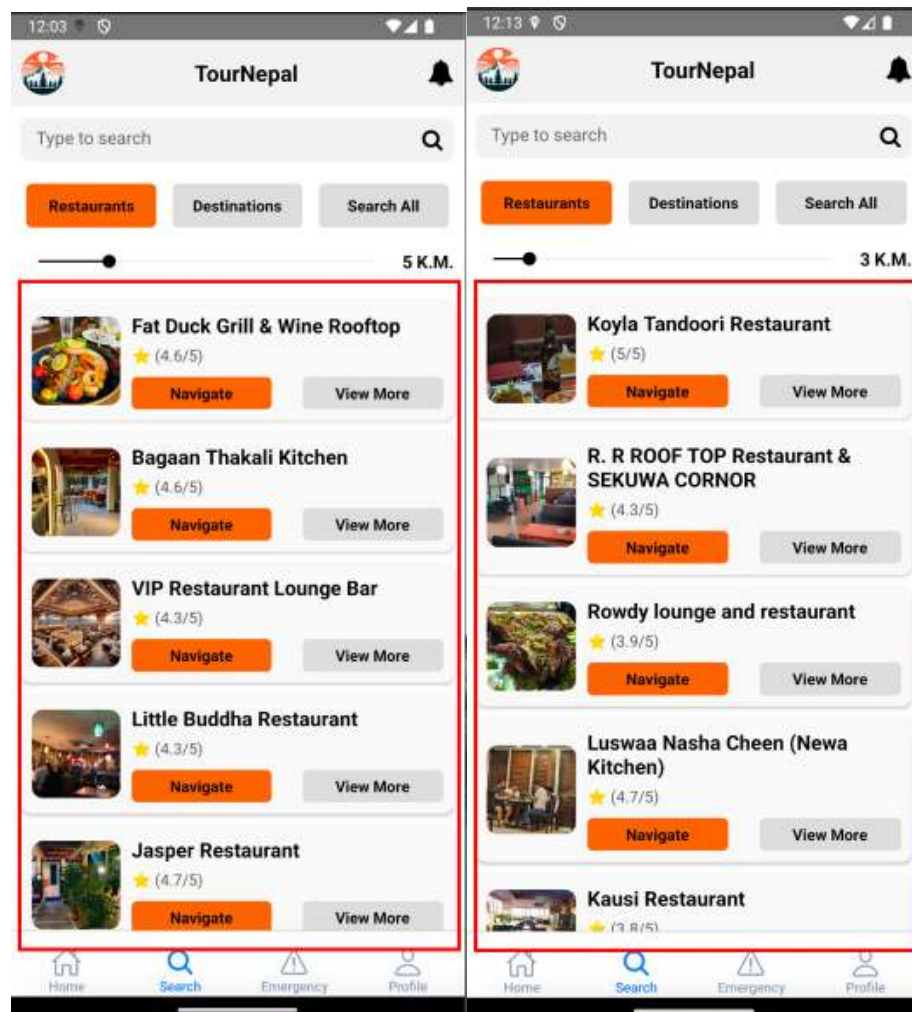


Figure 205: Restaurants filtering with radius

3) SYS-14 Destinations fetching by radius

Objective	To test if user is able to view destinations when changing destination type
Action	1) Navigated to search section from the bottom navigation bar. 2) Change the type to destinations.
Expected result	User should be able to view destinations near user's device location within 5K.M of radius.
Actual result	User was able to view destinations near user's device location within 5K.M of radius.
Conclusion	Test was successful.

Table 83: Destinations fetching by radius

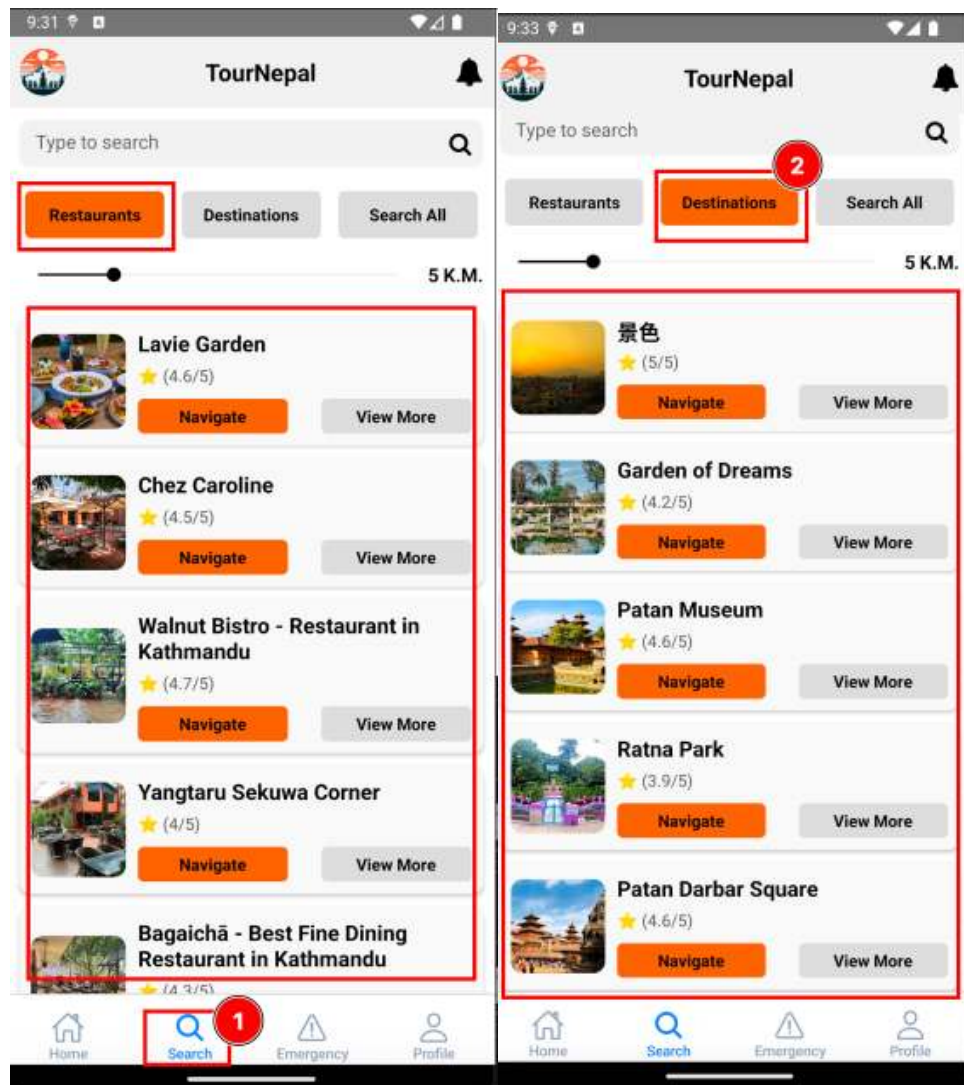


Figure 206: Destinations fetching by radius

4) SYS-15 Refreshing radius slider to update restaurants

Objective	To test if user is able to get newer destinations when changing radius slider.
Action	1) Navigated to search section from the bottom navigation bar. 2) Changed the type to destinations. 3) Adjusted the radius slider to view new destination
Expected result	User should be able to view destinations near user's device location within 8K.M of radius.
Actual result	User was able to view destinations near user's device location within 8K.M of radius.
Conclusion	Test was successful.

Table 84: Refreshing radius slider to update restaurants

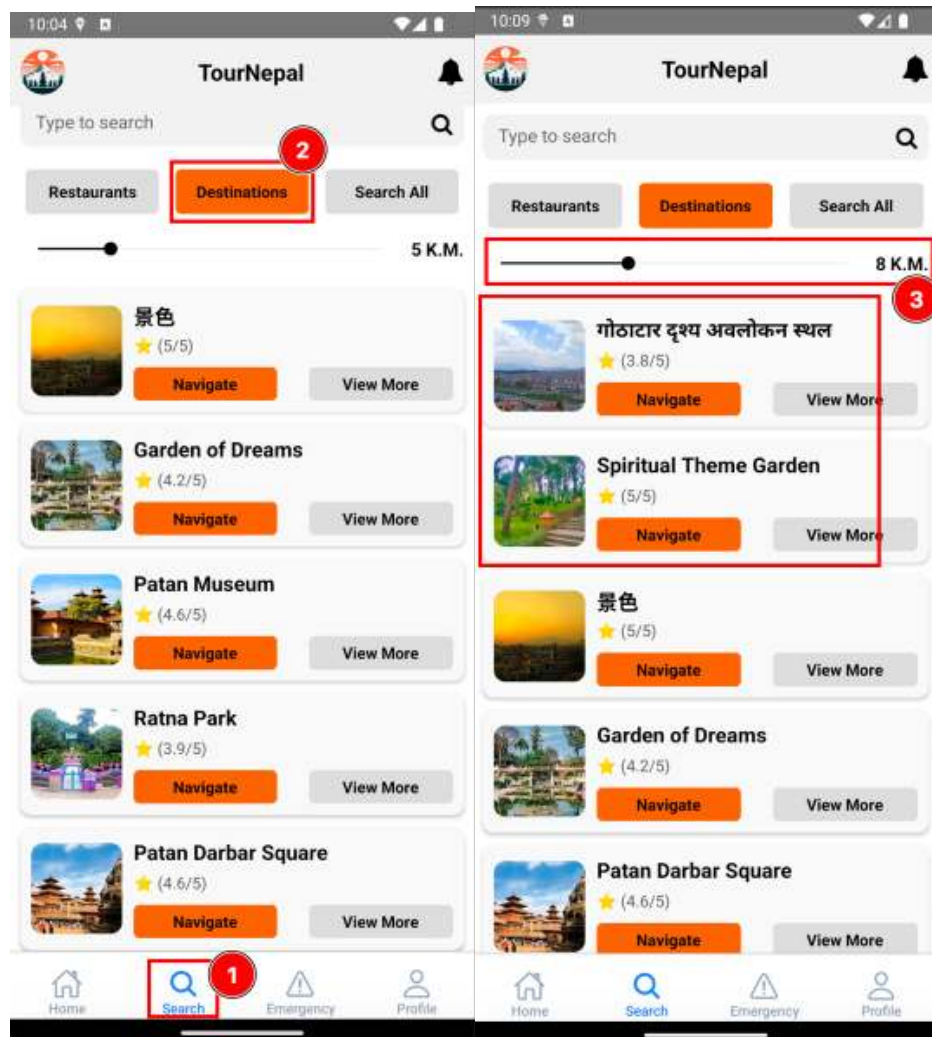


Figure 207: Refreshing radius slider to update restaurants

5) SYS-16 Search any place feature

Objective	To test if user is able to search any places without location and radius restriction.
Action	1) Navigated to search section from the bottom navigation bar. 2) Changed the type to Search All. 3) Searched place name and pressed search
Expected result	User should be able to view results for searched place.
Actual result	User was able to view results for searched place.
Conclusion	Test was successful.

Table 85: Search any place

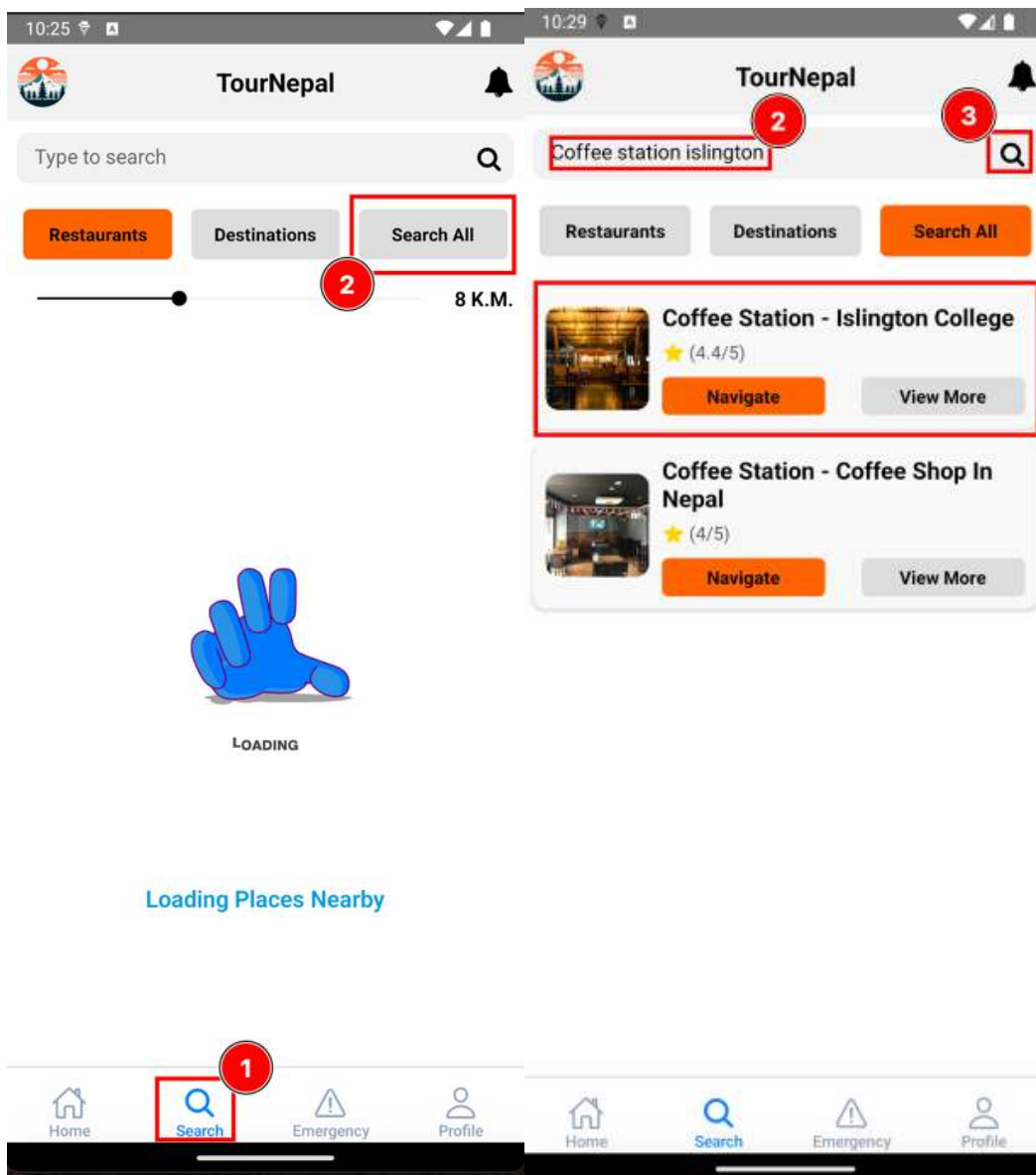


Figure 208: Search any place feature

6) SYS-17 Search filter for fetched location

Objective	To test if user is able to search through filtered location and destination type.
Action	1) Navigated to search section from the bottom navigation bar. 2) Searched for a place in fetched location.
Expected result	User should be able to view results according to search if available.
Actual result	User was able to view results according to search.
Conclusion	Test was successful.

Table 86: Search filter for fetched location

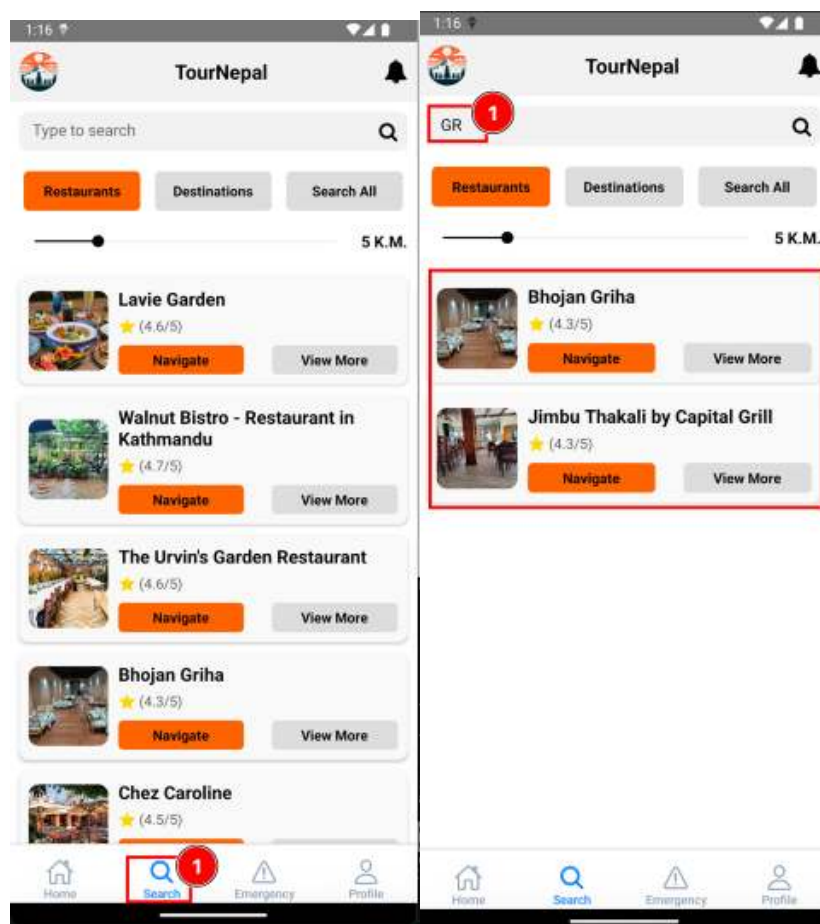


Figure 209: Search filter for fetched location

7) SYS-18 Navigating to google maps via application

Objective	To test if user can navigate via google map of selected location.
Action	1) Navigated to search section from the bottom navigation bar. 2) Clicked on navigate for redirection. 3) Revisited the application. 4) Clicked on view more of any place. 5) Clicked on navigate
Expected result	User should be redirected to google maps with exact location pinned both ways.
Actual result	User was redirected to google maps with exact location pinned both ways.
Conclusion	Test was successful.

Table 87:Navigating to google maps via application

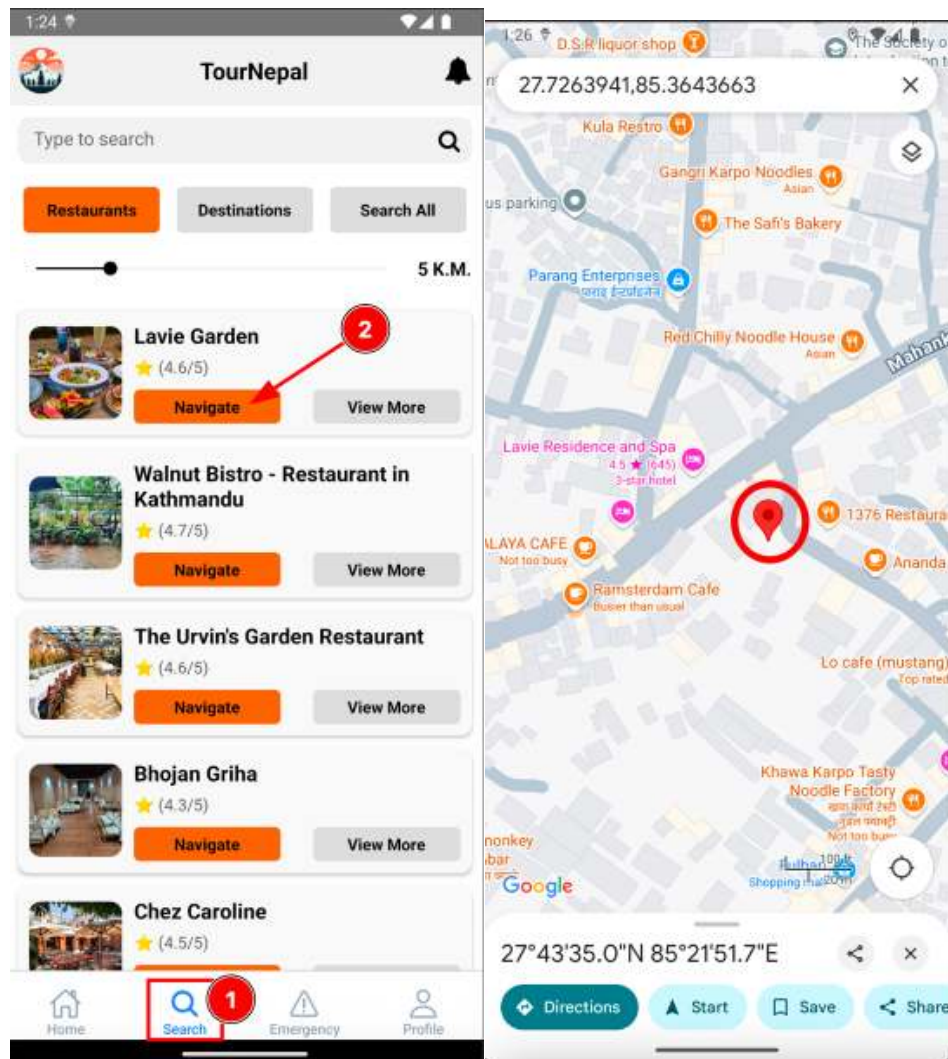


Figure 210: Navigating via main search screen

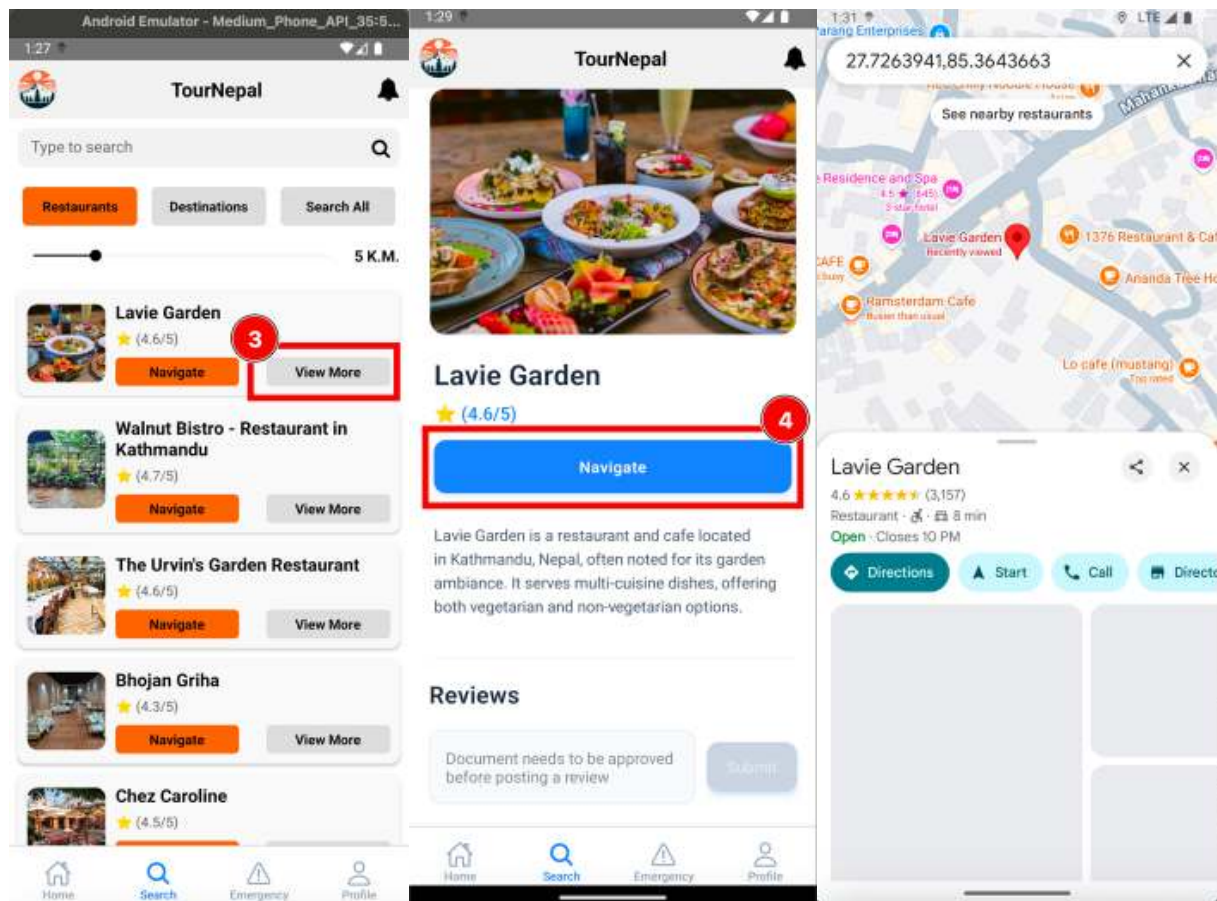


Figure 211: Navigating to location via description page

8) SYS-19 Description of restaurants

Objective	To test if user can view description of selected restaurants.
Action	1) Navigated to search section from the bottom navigation bar. 2) Chose destination type as Search All. 3) Searched Aangan and clicked on view more.
Expected result	User should be able to view the description of the restaurant, if well-known.
Actual result	User was able to view. The description of the restaurants.
Conclusion	Test was successful.

Table 88: Description of restaurants

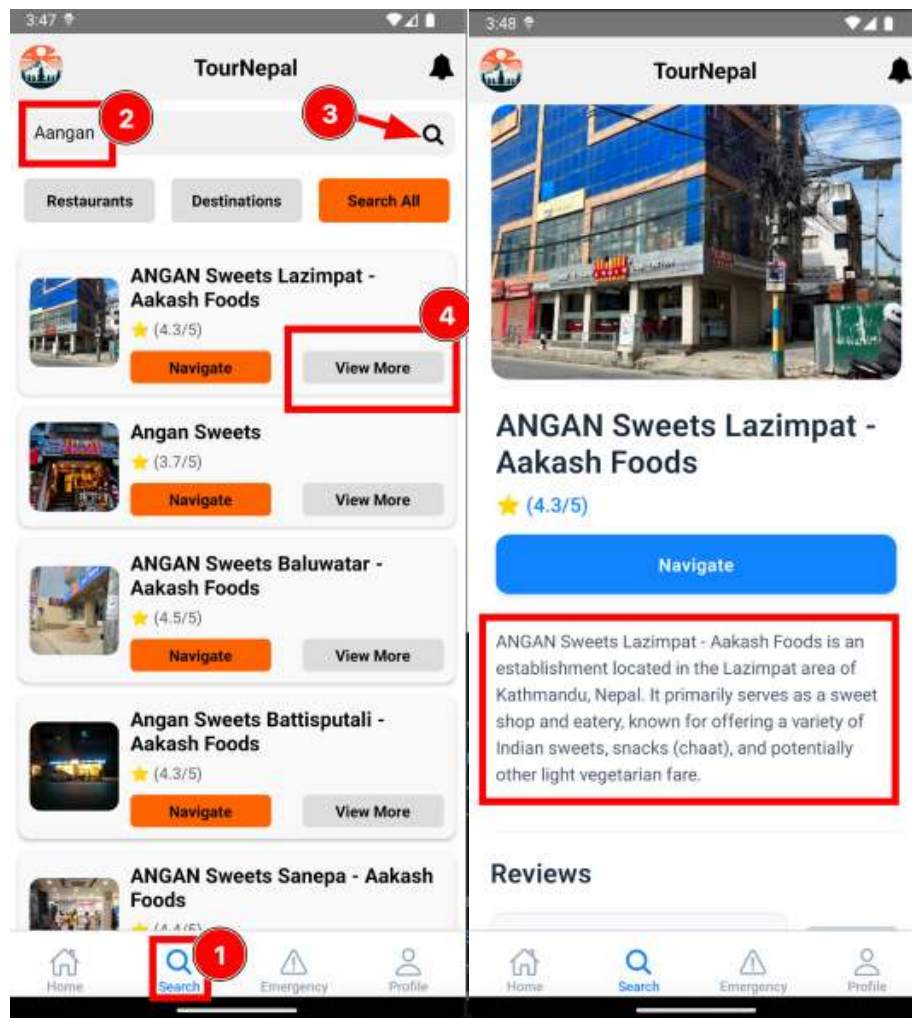


Figure 212: Description of restaurants

9) SYS-20 Description for less popular restaurants

Objective	To test if user can view description of selected less-popular restaurants.
Action	1) Navigated to search section from the bottom navigation bar. 2) Clicked on view more for unknown restaurant name.
Expected result	A proper message should be shown if description is unavailable for the place.
Actual result	A proper message was shown for less popular place.
Conclusion	Test was successful.

Table 89: Description for less popular restaurants

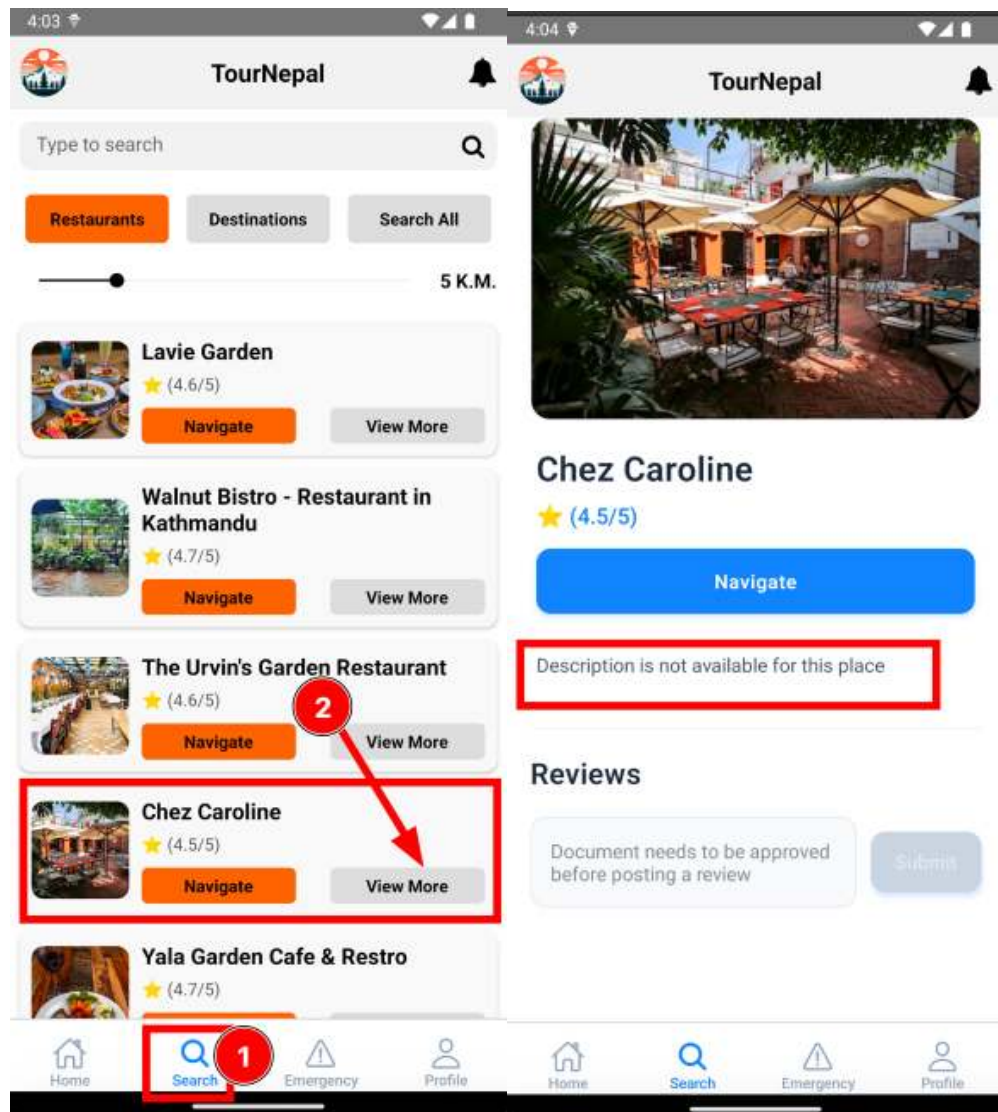


Figure 213: Description of less popular restaurants.

10) SYS-21 Description for destinations

Objective	To test if user can view description of selected destination.
Action	1) Navigated to search section from the bottom navigation bar. 2) Switched destination type to destinations. 3) Clicked on view more for a destination.
Expected result	Description of the selected destination should be shown.
Actual result	Description of the selected destination was shown.
Conclusion	Test was successful.

Table 90: Description for destinations

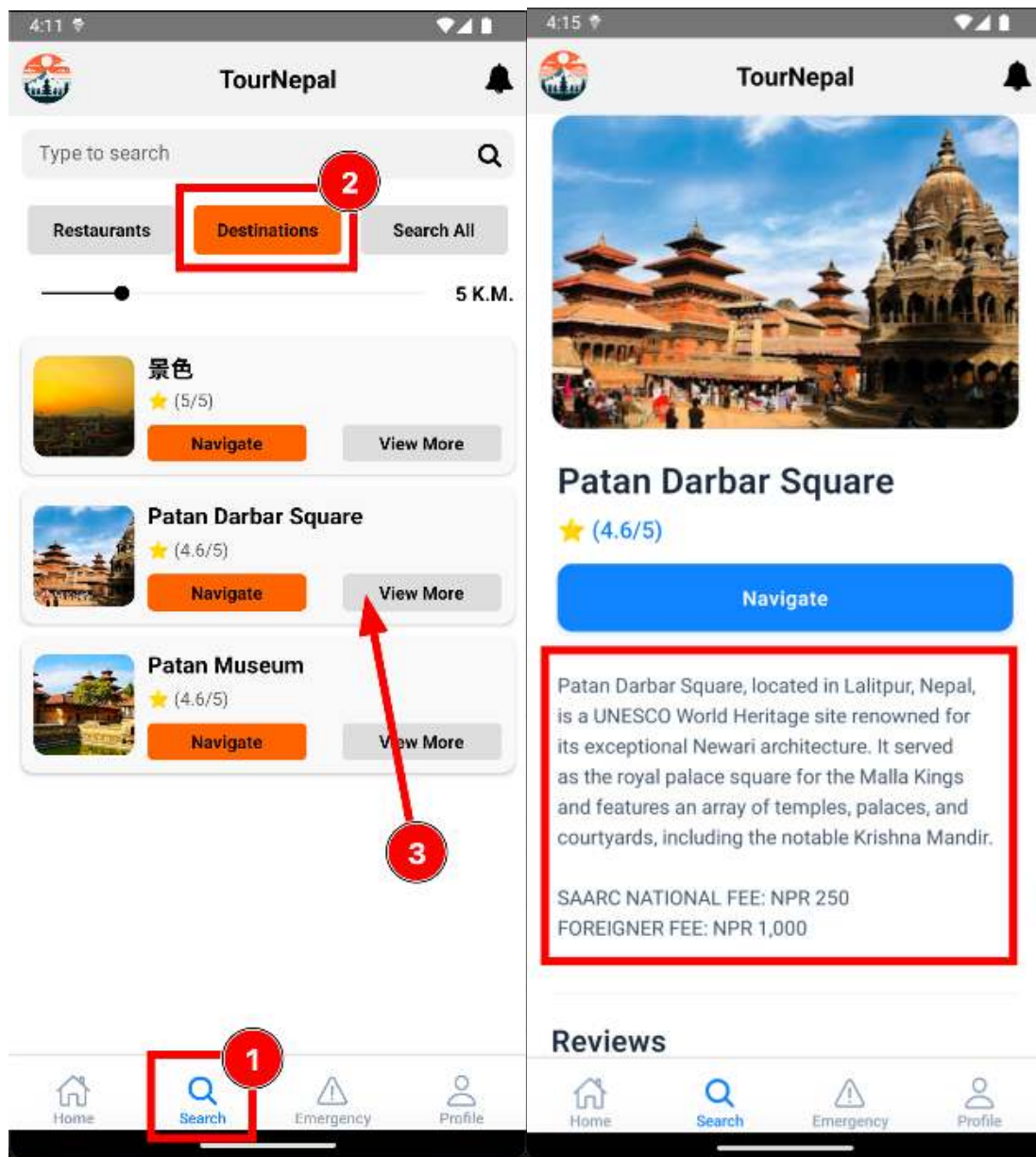


Figure 214: Description for destinations

4.4.3. Profile and reviews testing

1) SYS-22 Reuploading documents

Objective	To test if user can re-upload documents after verification rejected.
Action	1) Navigated to profile section from the bottom navigation bar. 2) Clicked on upload passport and visa button. 3) Captured dummy images 4) Clicked on resubmit.
Expected result	User's document should be resubmitted and verification status should change from Rejected to Pending Verification.
Actual result	User's document was resubmitted and verification status was changed from Rejected to Pending Verification.
Conclusion	Test was successful.

Table 91: Reuploading documents

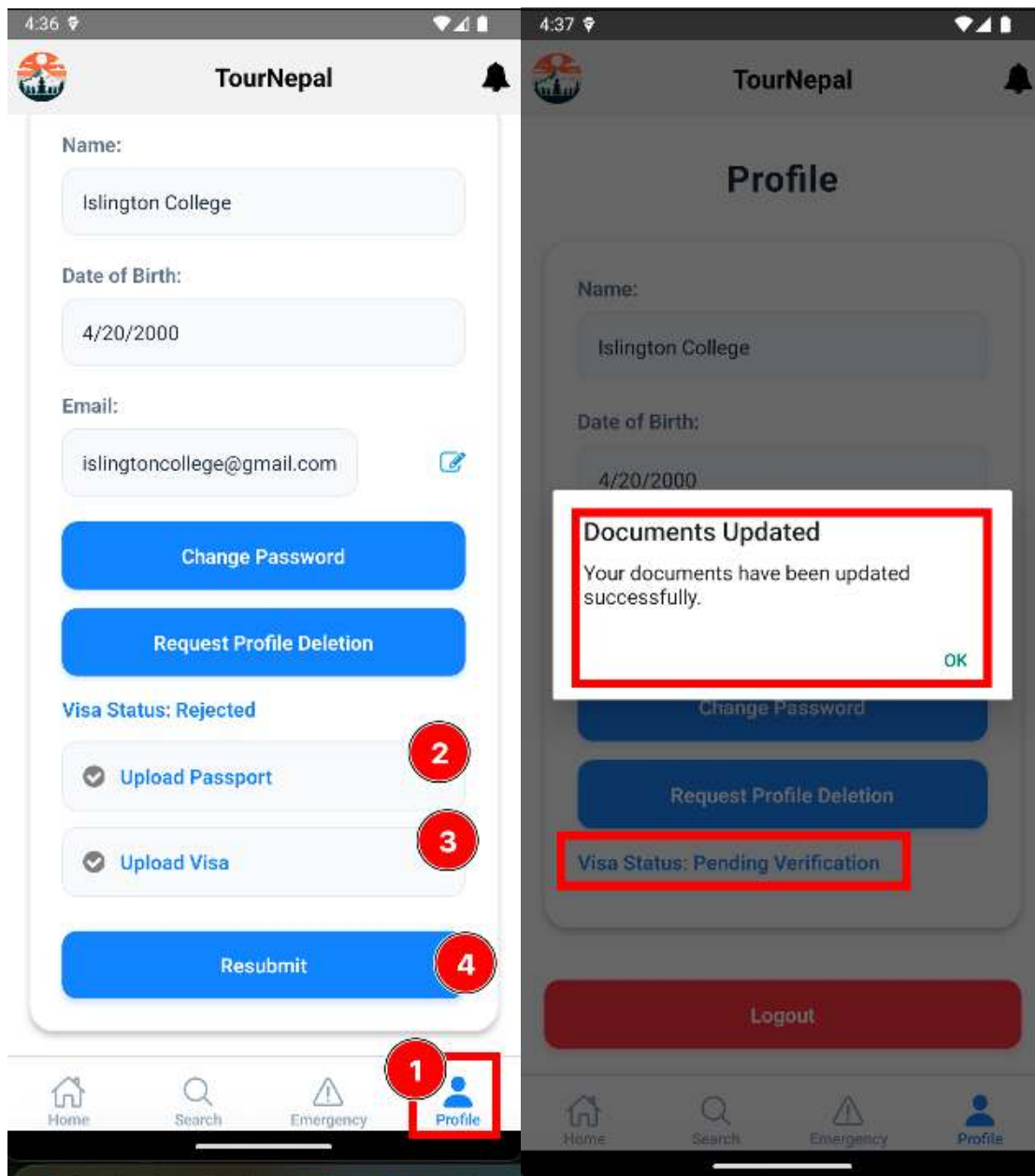


Figure 215: Reuploading documents

2) SYS-23 Verified user leaving a review

Objective	To test if verified user can leave a review in any place.
Action	1) Logged in from a verified user account. 2) Navigated to search section from the bottom navigation bar. 3) Changed destination type to Search All and searched Coffee Station Islington 4) Left a review.
Expected result	User should be able to leave a review successfully.
Actual result	User was able to leave a review successfully.
Conclusion	Test was successful.

Table 92: Verified user leaving a review

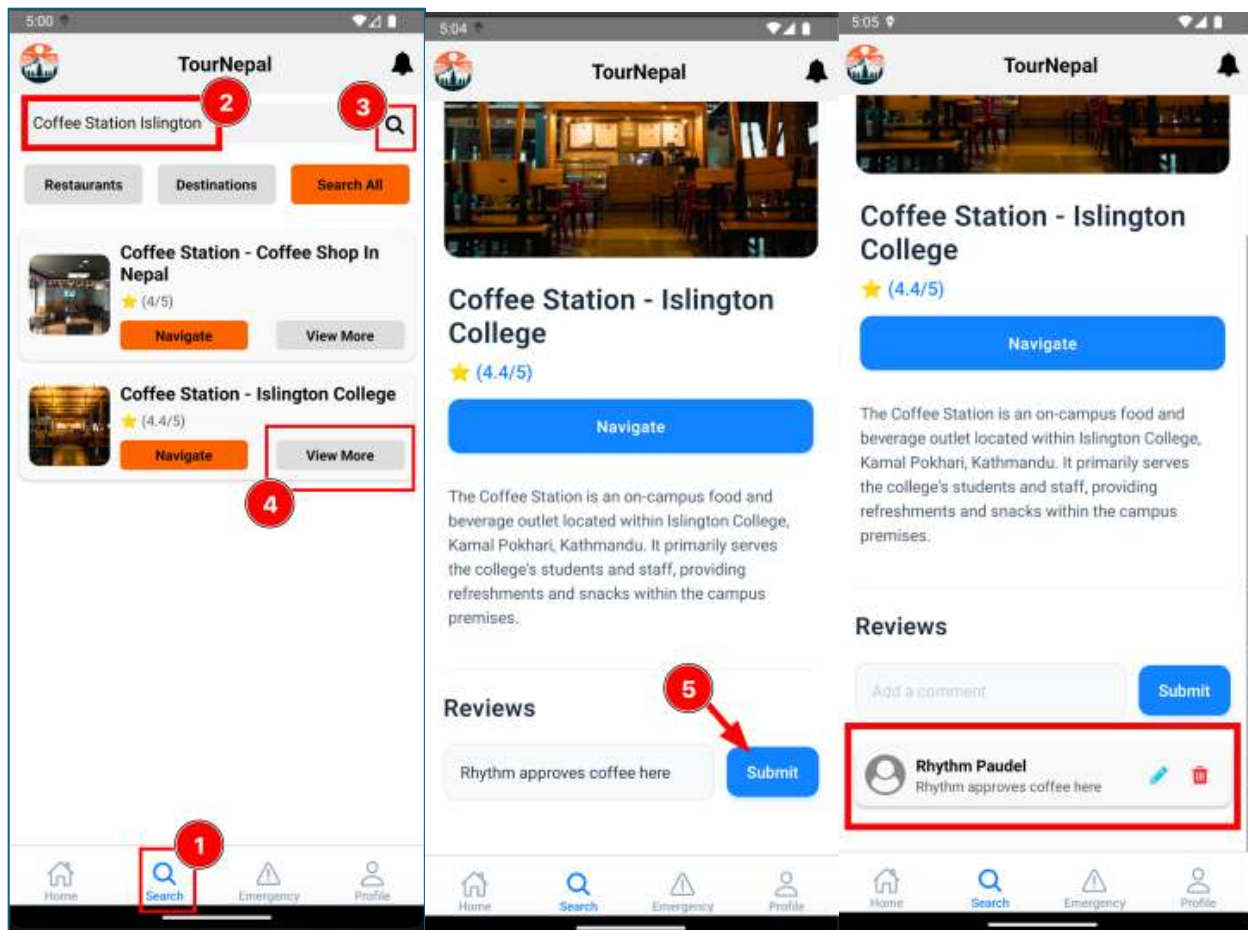


Figure 216: Verified user leaving a review

3) SYS-24 Unverified user leaving a review

Objective	To test if unverified user can leave a review in any place.
Action	1) Logged in from a unverified user account. 2) Navigated to search section from the bottom navigation bar. 3) Changed destination type to Search All and searched Coffee Station Islington 4) Added a review.
Expected result	User should not be able to leave a review.
Actual result	User was not able to leave a review.
Conclusion	Test was successful.

Table 93: Unverified user leaving a review

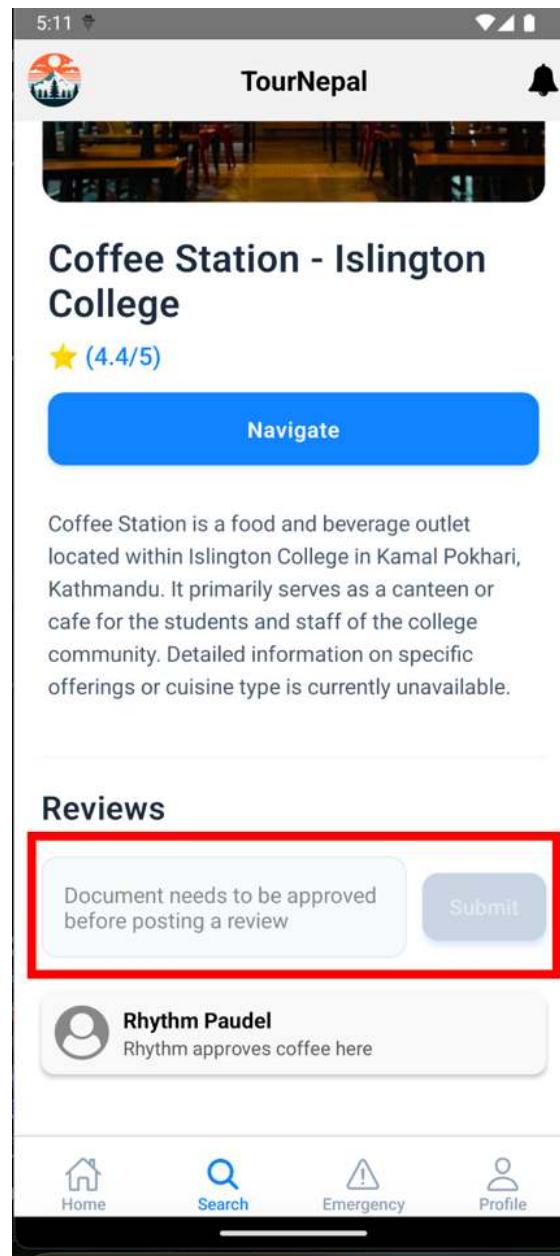


Figure 217: Unverified user trying to post a review

4.4.4. Admin tasks testing

1) SYS-25 Admin login test

Objective	To test if admin is able to login with valid credentials.
Action	1) Logged in using valid credentials
Expected result	Admin should be redirected to homepage.
Actual result	Admin was redirected to homepage.
Conclusion	Test was successful.

Table 94: Admin login test

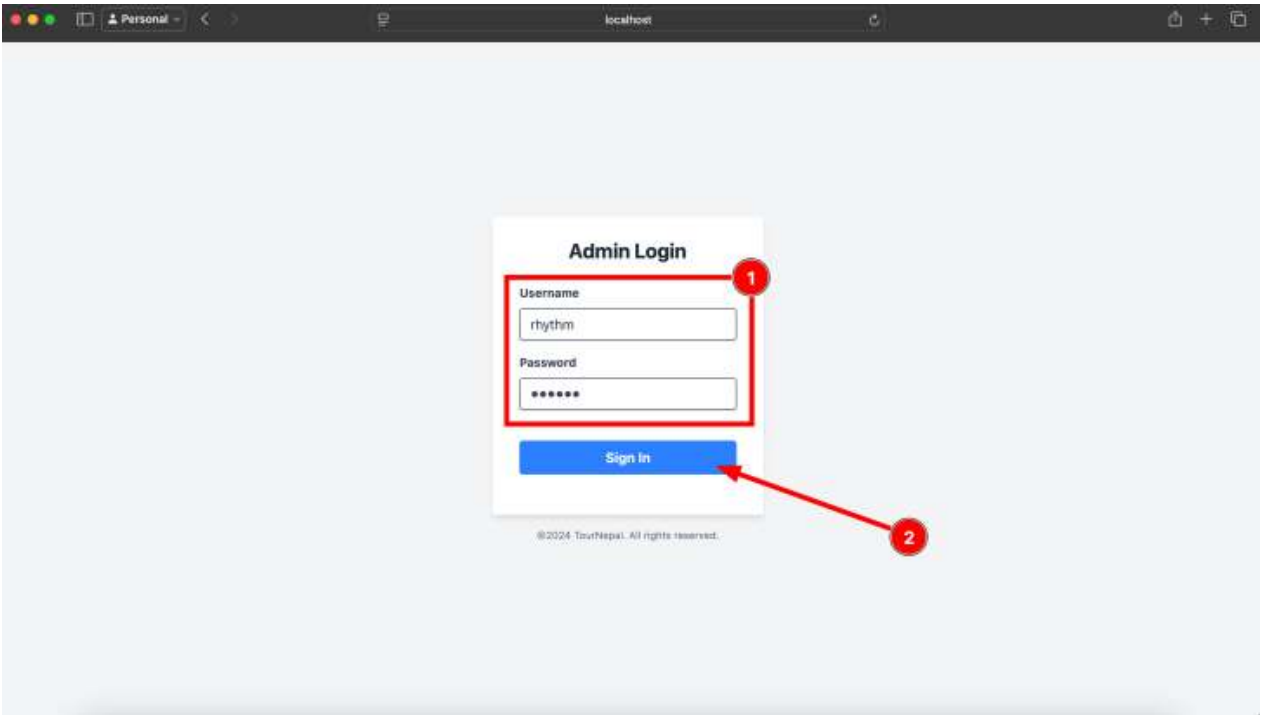


Figure 218: Admin logging in

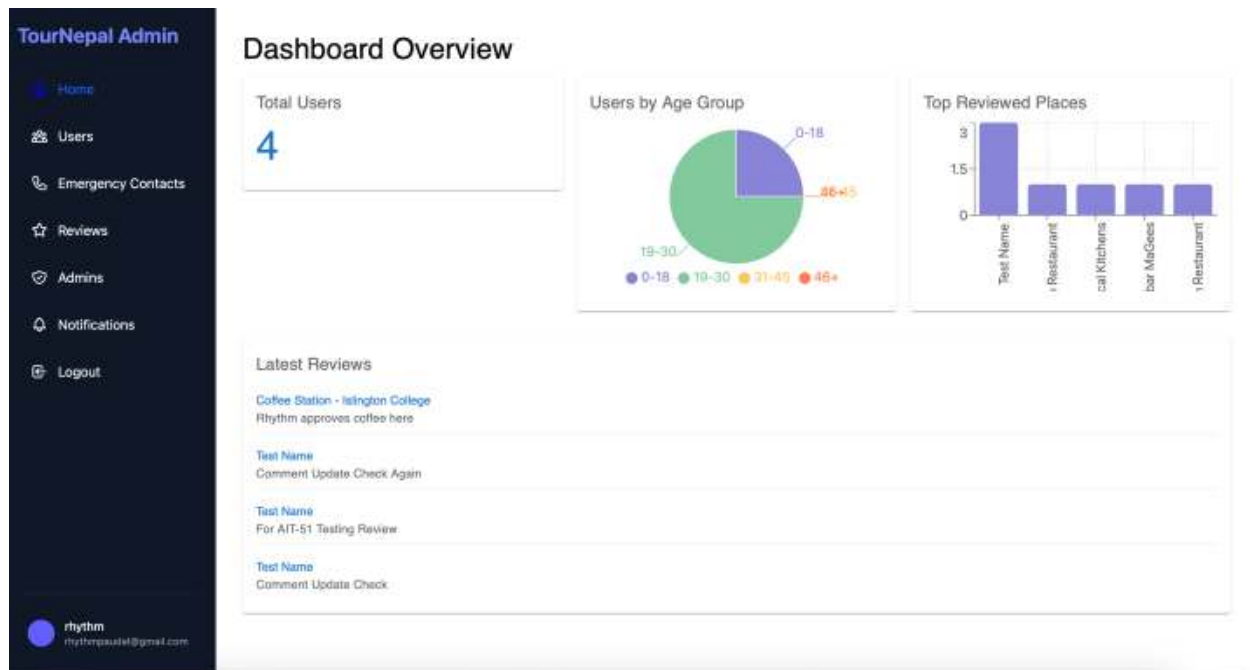


Figure 219: Admin dashboard

2) SYS-26 Admin adding user

Objective	To test if admin can add a new user for mobile application.
Action	1) Navigated to users section from the sidebar. 2) Clicked on add user. 3) Filled up all the required details. 4) Clicked on add user
Expected result	A new user should be added
Actual result	A new user was added.
Conclusion	Test was successful.

Table 95: Admin adding user

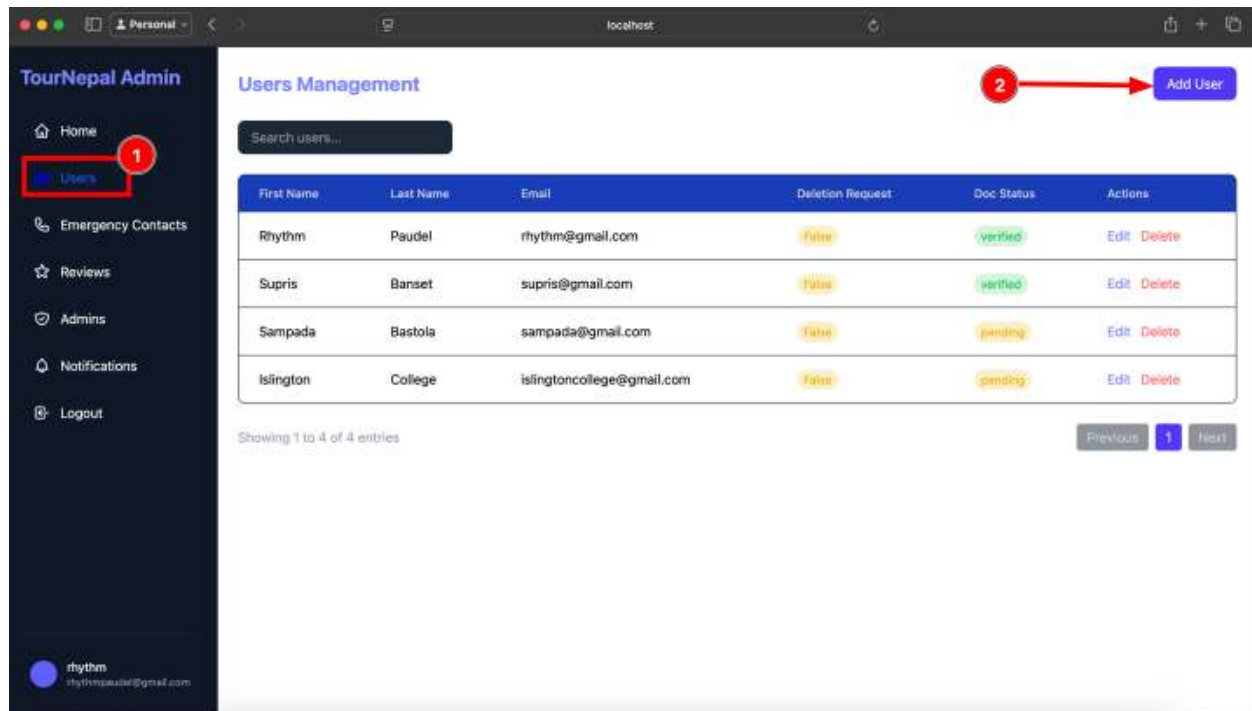


Figure 220: Navigating to add user section

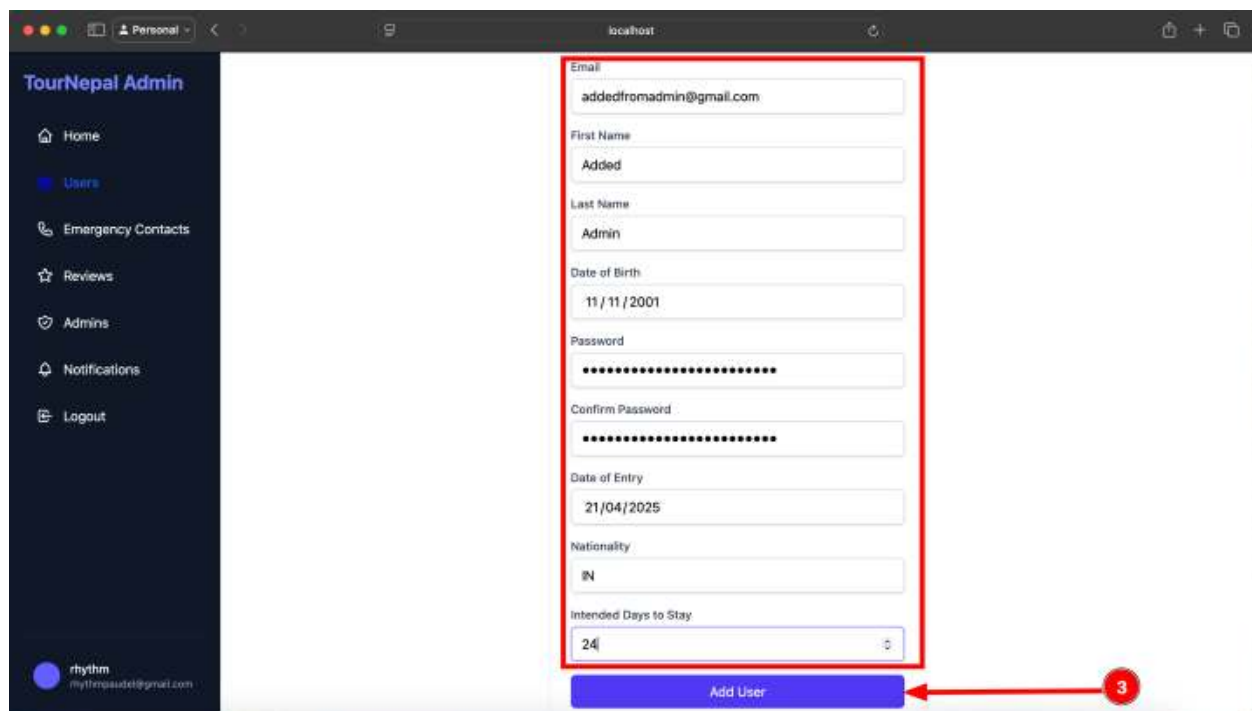


Figure 221: Creating a new user

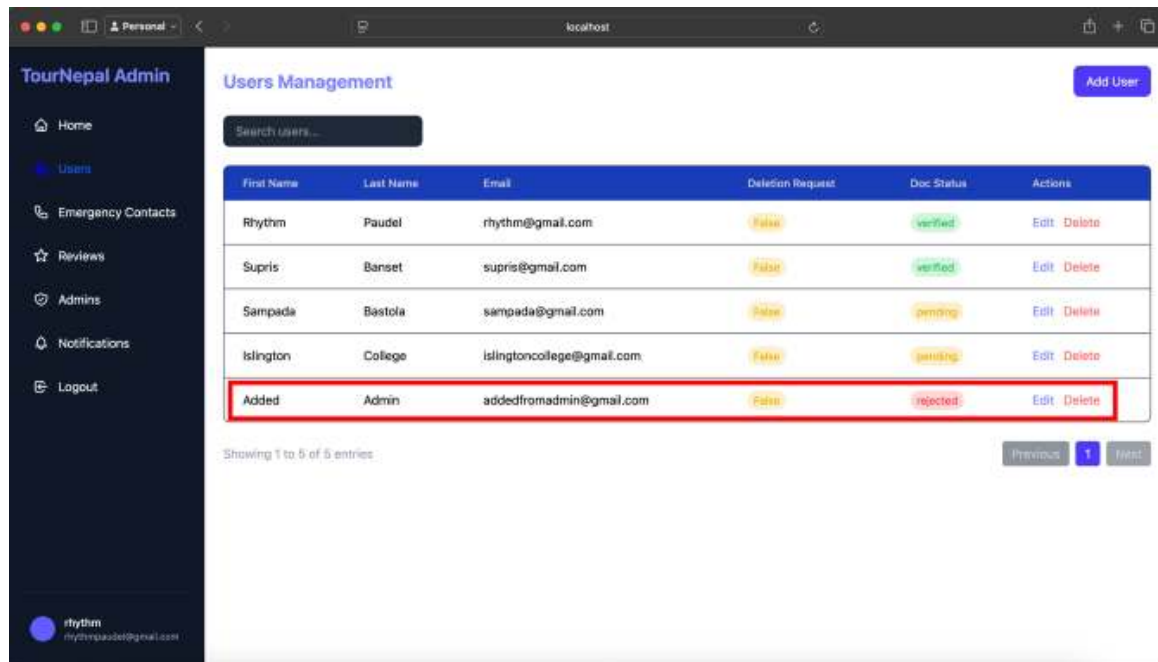


Figure 222: Addition of new user

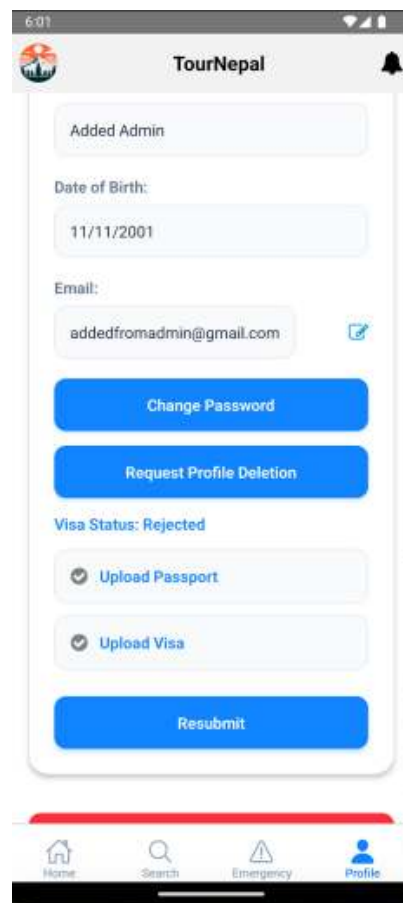


Figure 223: Profile view of added user on Mobile app

3) SYS-27 Verifying user status

Objective	To test if admin can verify/reject user document status.
Action	1) Navigated to users section from the sidebar. 2) Clicked on edit. 3) Changed the status of user from rejected/pending to verified 4) Clicked saved.
Expected result	User status should be updated to verified.
Actual result	User status was updated to verified.
Conclusion	Test was successful.

Table 96: Verifying user status

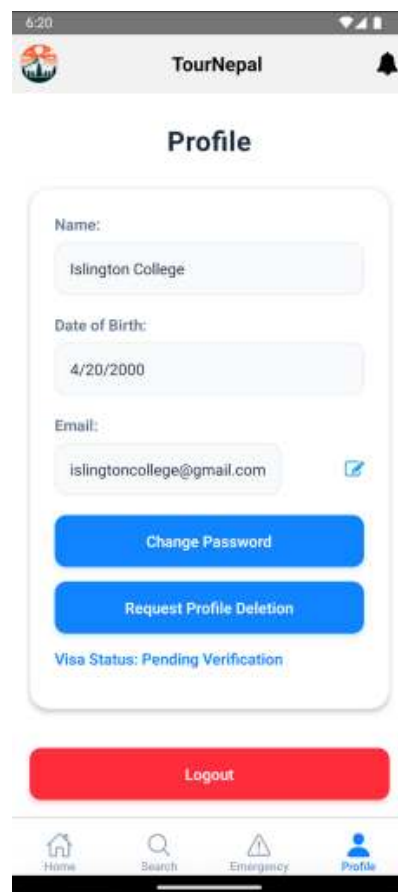


Figure 224: User status before verification

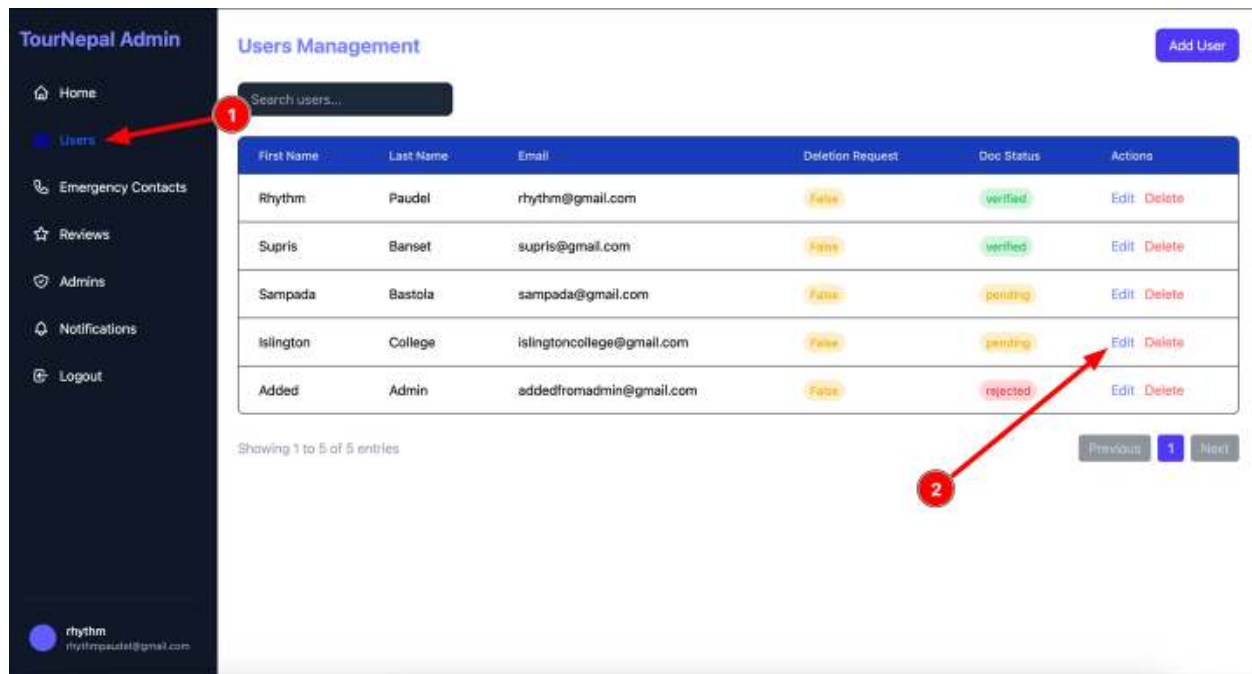


Figure 225: Verifying user(1)

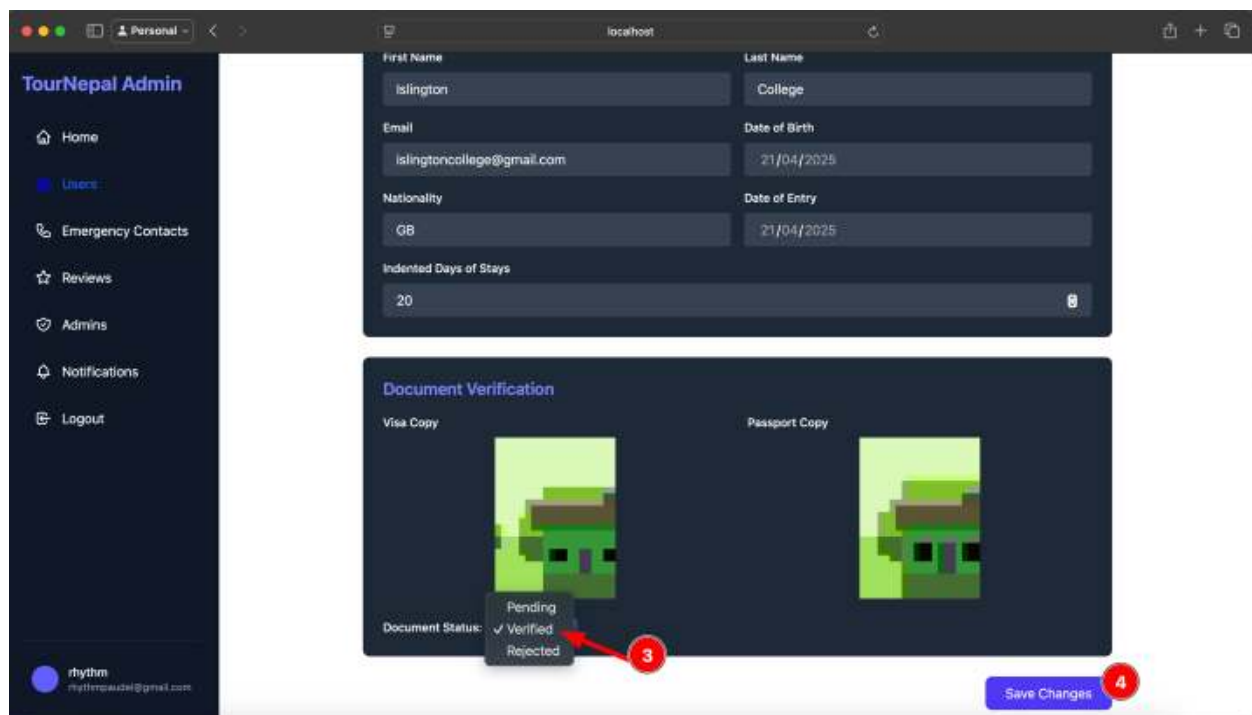
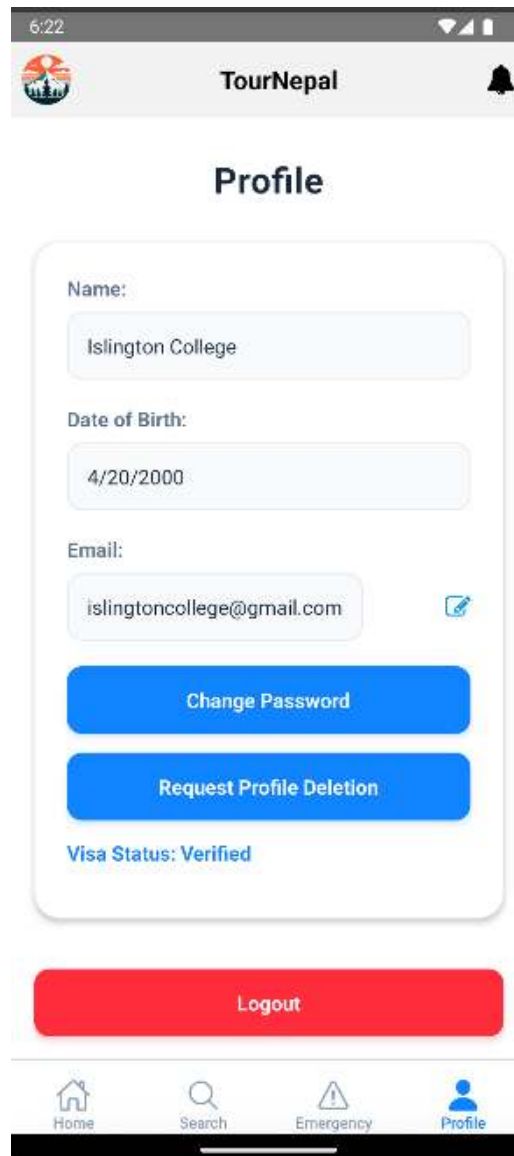


Figure 226: Verifying user(2)



The screenshot displays the 'Profile' page of the TourNepal mobile application. At the top, the status bar shows the time 6:22 and signal icons. The app header includes the TourNepal logo, the name 'TourNepal', and a notification bell icon. The main title 'Profile' is centered. Below it, a white card contains the following information: 'Name:' with the value 'Islington College', 'Date of Birth:' with '4/20/2000', and 'Email:' with 'islingtoncollege@gmail.com' and an edit icon. Two blue buttons, 'Change Password' and 'Request Profile Deletion', are positioned below the email field. The text 'Visa Status: Verified' is displayed in blue. A large red 'Logout' button is at the bottom of the card. The bottom navigation bar features four icons: Home, Search, Emergency, and Profile (which is highlighted).

6:22

TourNepal

Profile

Name:
Islington College

Date of Birth:
4/20/2000

Email:
islingtoncollege@gmail.com

Change Password

Request Profile Deletion

Visa Status: Verified

Logout

Home Search Emergency Profile

Figure 227: Verified user status

4) SYS-28 Casting notifications

Objective	To test if admin can send notification to specific/ all users.
Action	1) Navigated to notifications section from the sidebar. 2) Added title and body 3) Sent to all users. 4) Added title and body. 5) Sent to specific user only.
Expected result	User status should be updated to verified.
Actual result	User status was updated to verified.
Conclusion	Test was successful.

Figure 228: Notification casting

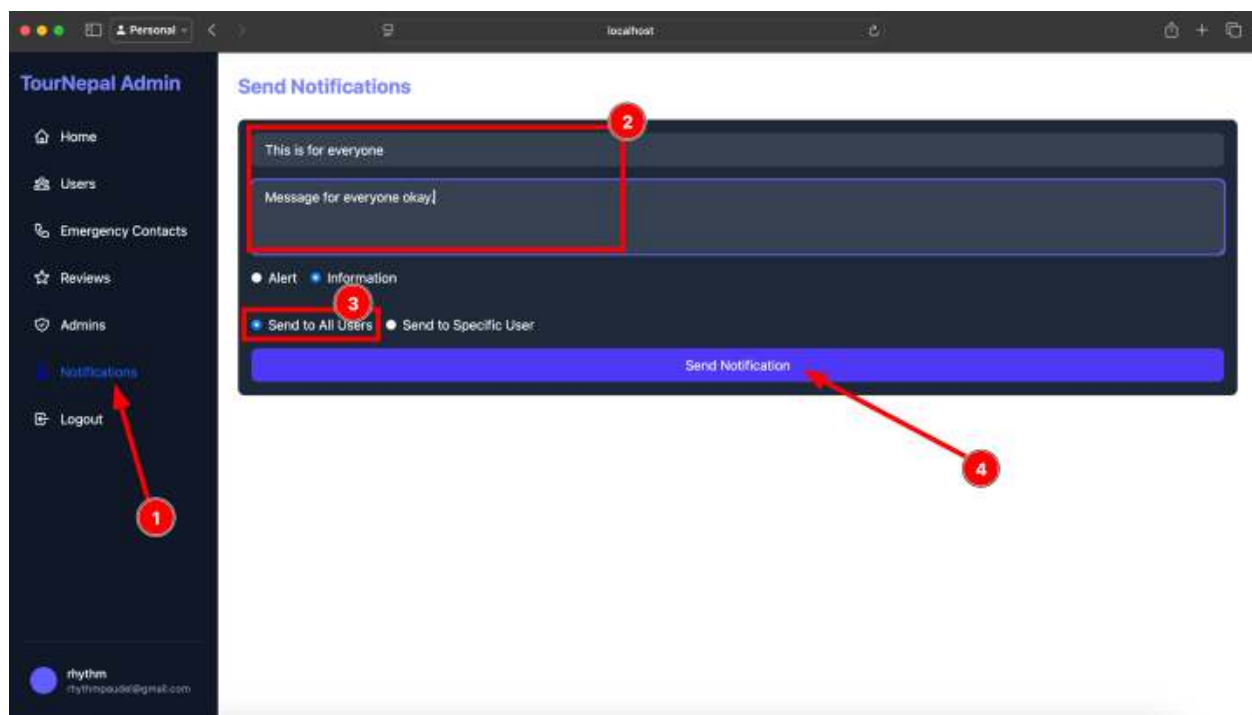


Figure 229: Casting notification to all users

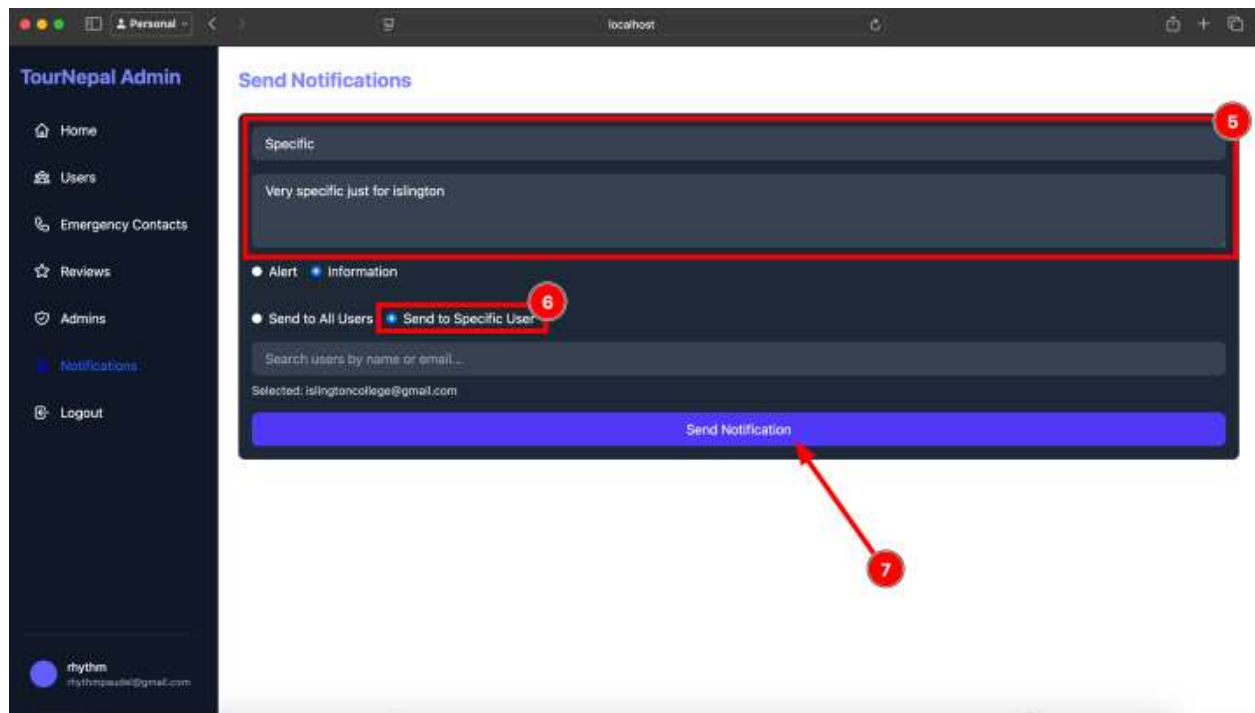


Figure 230: Casting notification to specific user

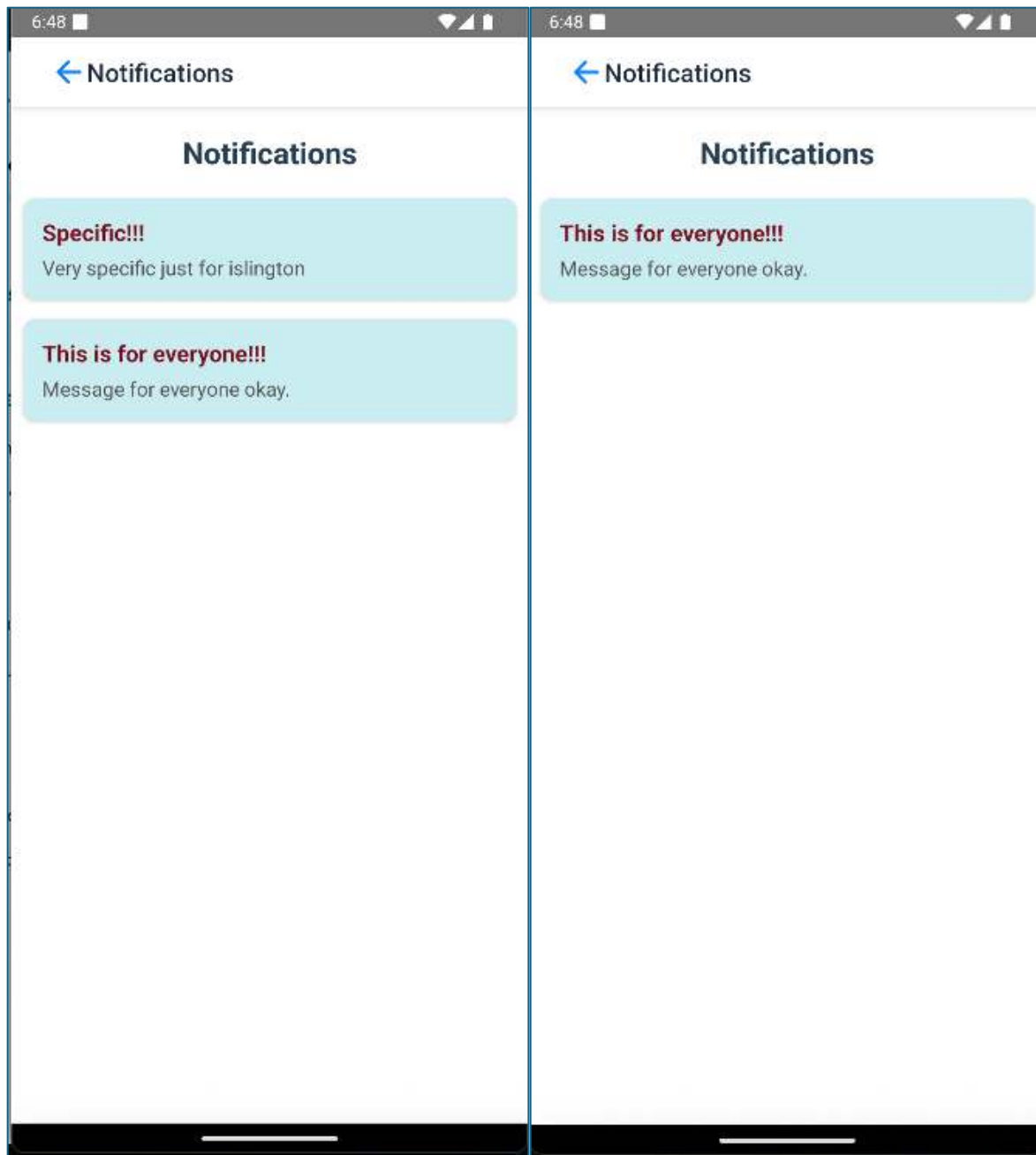


Figure 231: Notification list on different accounts

4.5. Critical analysis

The testing phases played a crucial role in the development. Several types of testing were performed, that ensured the application worked properly and handled errors in an efficient way. Three main types of testing were performed, which almost covered all the aspects of the application. Repeated testing was ignored, like review deletion from UI and API, as it contained repetitive procedures.

For the most part, unit testing, Jest, was used, which is used to mock several controllers and test logic. Similarly, for API integration, Postman/Thunder-Client were switched according to the features these API test applications provided. Some unexpected events were countered during the testing procedure; however, they were solved without altering the required outcome.

In conclusion, these tests played an important part in stabilizing the application. Additionally, unexpected logical and syntax errors that could have been encountered after the deployment of the application were handled before deployment, enhancing the experience of the application.

Chapter 5. Conclusion

The TourNepal, an astounding application, was designed to deliver a tourist-driven platform that could help tourists visiting Nepal in a couple of ways. The application primarily focuses on the main problems a tourist has to face in Nepal or in any country most often like having to deal with non-relevant reviews of a place. The technologies used for the development of the application was chosen very wisely. The used technologies are very active and widely used making it easier to update, upgrade or configure errors.

5.1. Legal, Social and Ethical Issues

5.1.1. Legal issues

The purpose of this application is to help tourists during their stay in Nepal, it might introduce various legal issues. One of the main concerns is data and privacy protection. The application collects confidential information of tourists, like their visa and passport details, to verify that the users are tourists, and if that information gets leaked or misused, it will impact the app's reputation and could lead to legal problems.

According to the Privacy Act, 2075, if any data leaks or unauthorized handling has been performed of any personal information, the government could charge up-to Rs.30,000 and imprisons the related person up-to 3 years (lawcommission.gov.np, 2018). [\(Additional details on legal issues can be found in Appendix G\)](#)

5.1.2. Social Issues

In addition to the legal issues, there are some social issues that can arise with this application. The app focuses on preventing scams and mistreatment of tourists visiting Nepal which could create a negative and a bad reputation of Nepal and Nepali people. This could also show that Nepali people are not trustworthy, and Nepal is not safe for tourists, leading to misunderstanding and social issues. [\(Additional details on social issues can be found in Appendix G\)](#)

5.1.3. Ethical Issues

Lastly, there are also some ethical issues that come up with this app. One of the main ones is how the user's data are used because the app collects sensitive information from the user. This app could face some ethical issues if the data collected are not secured and used fairly. For example, if tourists don't fully understand why their visa details are needed, it raises concerns about the app's ethics. Therefore, the app should be responsible to clearly explain what's being collected and used in its privacy policy. [\(Additional details on ethical issues can be found in Appendix G\)](#)

5.2. ADVANTAGES

The TourNepal application comes with a lot of benefits, making it an easy choice for tourists in Nepal. Some of the advantages of the application are listed below:

- i) The application provides a list of restaurants or destinations around their actual location.
- ii) The application lets the user decide, the radius of the destination, user is comfortable to travel.
- iii) The application provides list of all emergency contacts in a single place, where user will not have to go through internet every time for an emergency case for every situation.
- iv) The user using the application will be updated about the events, disaster or any type or disruption that might take place when user travel within, or outside of the valley.
- v) For any place unlike google maps, the reviews will be authentic and will only be relevant to tourists.
- vi) Admin will have the control over the use of application, making application genuine, trustworthy and in-some cases helpful for providing information, if required by government, in case of emergency.
- vii) The development was done thoughtfully, where application is open for modification without altering the existing functions.

5.3. Limitations

- i) Application is dependent on internet, therefore if there is not proper availability of internet, the application will not work properly.
- ii) In the current context, the application is working with Google Maps API; hence, if Google Maps API has some technical issue, the issue will be reflected in the application as well.
- iii) The application is not currently designed to support large scale workloads, however on demand the environments can be switched according to the need.
- iv) Application does not automatically identify suspicious activities, which makes application filtering manual by admins.

5.4. Future work

- i) Offline support for basic functionality will improve the user experience.
- ii) Adding functionality where user can directly book hotels, for restaurants for dining.
- iii) B2B portal for adding custom restaurants, hotels or destinations, rather than relying on google maps API.
- iv) In-app cab booking system for easy access of long/short route travel for users.
- v) Famous sites and events managing system designed solely for tourists.
- vi) In-app e-sim purchasing facility and activation service in collaboration with carrier services.

Chapter 6. References

CollegesNp, 2023. *Overcoming the Challenges Confronting the Nepali Tourism Industry*. [Online]

Available at: <https://www.collegenp.com/article/overcoming-the-challenges-confronting-the-nepali-tourism-industry/>

[Accessed 18 09 2024].

Gautam, B. P., n.d. *Tourism and Economic Growth in Nepal*. [Online]

Available at: [https://www.nrb.org.np/contents/uploads/2019/12/NRB Economic Review-Vol 23-2 October 20112 Tourism and Economic Growth in NepalBishnu-Prasad-Gautam-Ph.D..pdf](https://www.nrb.org.np/contents/uploads/2019/12/NRB_Economic_Review-Vol_23-2_October_20112_Tourism_and_Economic_Growth_in_NepalBishnu-Prasad-Gautam-Ph.D..pdf)

[Accessed 19 March 2025].

Nepal Republic Media Limited, 2025. *Nepal's tourism bounces back in 2024, arrival number recorded over 1.147 million in a year - myRepublica - The New York Times Partner, Latest news of Nepal in English, Latest News Articles | Republica*. [Online]

Available at: <https://myrepublica.nagariknetwork.com/news/nepals-tourism-bounces-back-in-2024-arrival-number-recorded-over-1147-milli...-67741e276fc76.html>

[Accessed 19 March 2025].

Dhungana, S., 2019. *Foreigners report high number of theft and fraud cases with tourist police*. [Online]

Available at: <https://kathmandupost.com/national/2019/10/24/foreigners-report-high-number-of-theft-and-fraud-cases-with-tourist-police>

[Accessed 11 November 2024].

Karki, J., 2023. *Possible Scams to be Aware of When Traveling to Nepal*. [Online]

Available at: <https://luxuryholidaynepal.com/blog/possible-scams-in-nepal>

[Accessed 18 09 2024].

Mosaic Adventure, 2024. *19 Possible Tourist Scams you Should AVOID in Nepal in 2024..* [Online]

Available at: <https://mosaicadventure.com/tourist-scams-you-should-avoid-in-nepal/>

[Accessed 18 09 2024].

Tripadvisor LLC, 2023. *US Press Center | About Tripadvisor*. [Online] Available at: <https://tripadvisor.mediaroom.com/us-about-us> [Accessed 5 January 2025].

Condor Limited, 2025. *100+ TripAdvisor Statistics 2024*. [Online] Available at: <https://www.condorferries.co.uk/tripadvisor-statistics#:~:text=TripAdvisor%20ranked%20no.,nearing%20900%20million%20registered%20users>.

[Accessed 5 January 2025].

Tripadvisor LLC, 2025. *Travel better with the new Tripadvisor app.* [Online] Available at: <https://www.tripadvisor.com/app> [Accessed 5 January 2025].

Google Play, 2025. *Culture Trip: Book Travel - Apps on Google Play*. [Online] Available at: https://play.google.com/store/apps/details?id=culturetrip.com&hl=en_GB [Accessed 5 January 2025].

Google Play, 2024. *Visit Nepal Official Guide - Apps on google play*. [Online] Available at: <https://play.google.com/store/apps/details?id=com.pas.visitnepal&hl=en> [Accessed 5 January 2025].

Kumar, G. & Bhatia, P. K., 2012. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*. [Online] Available at: https://www.researchgate.net/profile/Gaurav-Kumar-175/publication/255707851_Impact_of_Agile_Methodology_on_Software_Development_Process/links/00b49520489442e12d000000/Impact-of-Agile-Methodology-on-Software-Development-Process.pdf

[Accessed 1 October 2024].

AWS, 2024. *What is Scrum? - Scrum Methodology Explained - AWS*. [Online] Available at: <https://aws.amazon.com/what-is/scrum/> [Accessed 19 9 2024].

Kouassi, J.-P. K., 2024. *The Agile Scrum Methodology simplified*. [Online] Available at: <https://www.linkedin.com/pulse/agile-scrum-methodology-simplified-jean-paul-k-kouassi-vwu4e>

[Accessed 12 November 2024].

OpenJs Foundation, 2024. *Node.js --- The V8 JavaScript Engine*. [Online] Available at: <https://nodejs.org/en/learn/getting-started/the-v8-javascript-engine>

[Accessed 1 October 2024].

Amazon Web Services, Inc., 2025. *What is Architecture Diagramming? - Software & System Architecture Diagramming Explained - AWS*. [Online]

Available at: <https://aws.amazon.com/what-is/architecture-diagramming/#:~:text=System%20architecture%20diagrams%20provide%20a,a%20system's%20structure%20and%20architecture.>

[Accessed 29 March 2025].

IBM, 2021. *Logical data models - IBM Documentation*. [Online]

Available at: <https://www.ibm.com/docs/en/ida/9.1?topic=modeling-logical-data-models>

[Accessed 29 March 2025].

Osis, J. & Donins, U., 2017. *Topological UML Modeling*. s.l.:Elsevier.

lawcommission.gov.np, 2018. *The-Privacy-Act-2075-2018*. [Online]

Available at: <https://repository.lawcommission.gov.np/en/wp-content/uploads/2019/07/The-Privacy-Act-2075-2018.pdf>

[Accessed 21 April 2025].

Google, 2025. *Google Maps Platform Terms Of Service | Google Cloud*. [Online]

Available at: <https://cloud.google.com/maps-platform/terms>

[Accessed 21 April 2025].

Jacobs, M., Casey, L. & Kaim, E., 2022. *What is Git? - Azure DevOps | Microsoft Learn*.

[Online]

Available at: <https://learn.microsoft.com/en-us/devops/develop/git/what-is-git>

[Accessed 26 April 2025].

Amazon Web Services, Inc., 2025. *What is JavaScript? - JavaScript (JS) Explained - AWS*. [Online]

Available at: <https://aws.amazon.com/what-is/javascript/>
[Accessed 26 April 2025].

Sutherland, K. S. & J., 2020. *The Scrum Guide*. [Online]
Available at: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf>
[Accessed 12 November 2024].

Atlassian, 2024. *What is scrum and how to get started*. [Online]
Available at: <https://www.atlassian.com/agile/scrum>
[Accessed 13 November 2024].

Hoory, L. & Bottorff, C., 2024. *What is Waterfall Methodology - Forbes Advisor*. [Online]
Available at: <https://www.forbes.com/advisor/business/what-is-waterfall-methodology/>
[Accessed 18 November 2024].

Indeed Editorial Team, 2024. *A Complete Guide to the Waterfall Methodology | Indeed.com*. [Online]
Available at: <https://www.indeed.com/career-advice/career-development/waterfall-methodology>
[Accessed 18 November 2024].

Wakode, R. B., Raut, L. P. & Talmale, P., 2015. Overview on Kanban Methodology and its Implementation. *IJSRD - International Journal for Scientific Research & Development*, 3(02), pp. 2321-0613.

TopSquad, 2023. *Kanban Framework: Your Guide to Agile Efficiency and Simplicity*. [Online]
Available at: <https://www.linkedin.com/pulse/kanban-framework-your-guide-agile-efficiency-simplicity>
[Accessed 18 November 2024].

Ganney, P. S., Pisharody, S. & Claridge, E., 2020. Chapter 9 - Software engineering. In: A. Taktak, P. S. Ganney, D. Long & R. G. Axell, eds. *Clinical Engineering*. s.l.:Academic Press, pp. 131-168.

Sitesbay, 2024. *What is Spiral Model in SDLC - Software Engineering Tutorial*. [Online] Available at: https://www.sitesbay.com/software-engineering/se-spiral-model#google_vignette
[Accessed 18 November 2024].

Chapter 7. Appendix

7.1. Appendix A: Phases of methodology

7.1.1. Initiation phase

In the initial phase, the topic for development was selected, taking into consideration the current context problem. After the selection of the project, scopes, objectives, and requirements were gathered in surface level for a better understanding of the application requirements. This was performed with discussion with stake-holders. Feasibility of the project is performed in this phase.

7.1.2. Sprint planning phase

In Scrum, the sprint planning phase is one of the most important phases, where at the beginning of every sprint, a planning/meeting phase was conducted. This ensured the project was being developed as per plan and that changes were needed to make. This also made sure the project was within scope.

7.1.3. Execution phase

This phase includes the development of the project. Multiple broken steps were developed in an iterative manner, which includes stand-up meetings for making everything up-to-date and updated. Core features of the application, like user authentication, Google map integration, admin panel functionalities, and notifications features, were developed and improved in this phase.

7.1.4. Review Phase

There are several steps performed in each sprint. It is necessary to understand if there have been changes, or anything is not according to the sprint plan. To overcome this, a review phase was held that ensured the collection of feedback through stakeholders. This phase also makes sure if any further improvements are needed to be carried out in next sprint.

7.1.5. Sprint Restrospective Phase

In this phase, a discussion is carried out on what went well, challenges faced, and improvements made. The review phase also concludes if any changes are needed to be added in the next sprint.

7.1.6. Final Testing and Deployment Phase

This is the final phase of the scrum methodology. In this phase, a comprehensive testing is performed. Several testing techniques, like unit testing and system testing were performed. If any bugs and error are found, it was solved in the testing phase. After the testing was performed, and application was ready to deploy, deployment process was begun. Similarly, project was closely monitored and evaluated for future scalability.

7.2. Appendix B: Selected methodology

Scrum Methodology has been increasing due to its flexibility and adaptability, which is one of the major reasons for this project to work on the Scrum framework. Being able to manage sprints, it will be easy to make continuous improvements to the project. (Sutherland, 2020). This methodology stands out compared to other methodologies in trend. Another popular methodology Kanban lacks the iteration where there could be an improvement. This indicates that there needs to be a very definite and defined version of the features. (Atlassian, 2024).

Scrum is a management framework that will help the members to work on a common goal with proper communication and collaboration that will hold multiple meetings with the team (AWS, 2024). In Scrum, tasks are divided into various sprints. Each sprint consists of a single task that needs to be completed in the given timeframe. Multiple iterations are also performed to meet the requirements of the project.

Due to Scrum's iterative approach, it allows for frequent feedback and improvement in each sprint, which ensures a proper delivery of features required. Scrum methodology's ability to adapt to changing requirements and ensure a timely delivery of the features makes it the best-suited methodology for this application.

7.3. Appendix C: Considered methodology

7.3.1. Waterfall Methodology

Waterfall methodology is a linear project management methodology that has a well-defined start and end. This methodology is best suited for the projects that has clear objectives and are highly predictable (Hoory & Bottorff, 2024). Since this project needs a constant change the waterfall methodology was rejected.

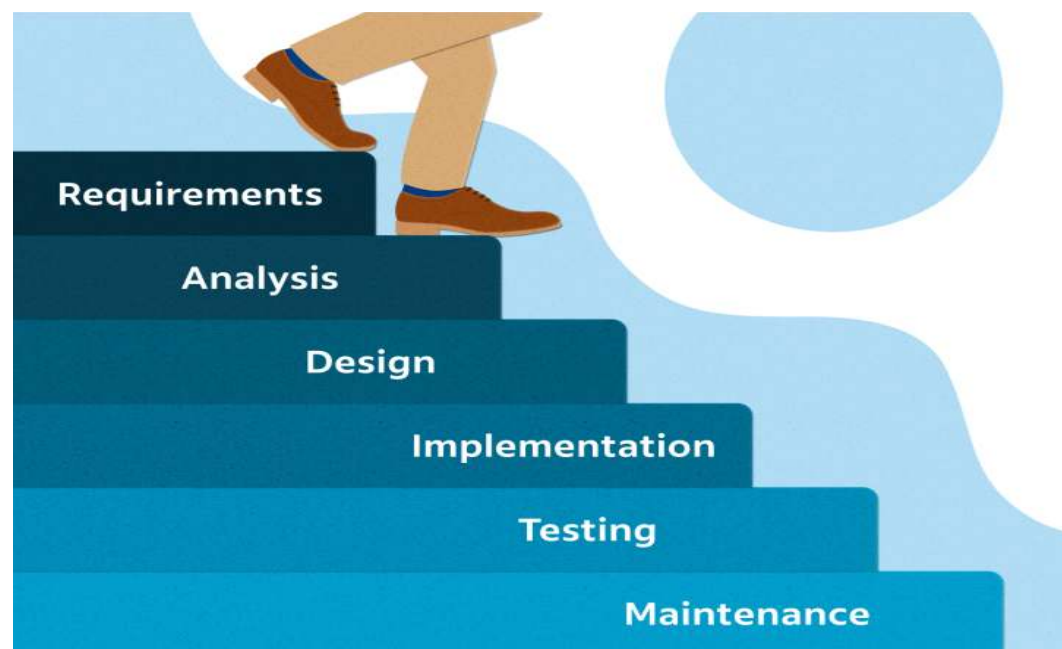


Figure 232: Waterfall methodology

(Indeed Editorial Team, 2024)

7.3.2. Kanban Methodology

Kanban methodology is a project management method where cards are used knowing and updating the upcoming, completed and current task. It ensures the right amount of work is being done at the right time, also known as closed-loop process (Wakode, et al., 2015).



Figure 233: Kanban methodology.

(TopSquad, 2023)

7.3.3. Spiral Methodology

Spiral Methodology is a combination of waterfall and prototyping model. This methodology works in an incrementing order. For each iteration a working prototype is prepared and is updated with user feedback (Ganney, et al., 2020).

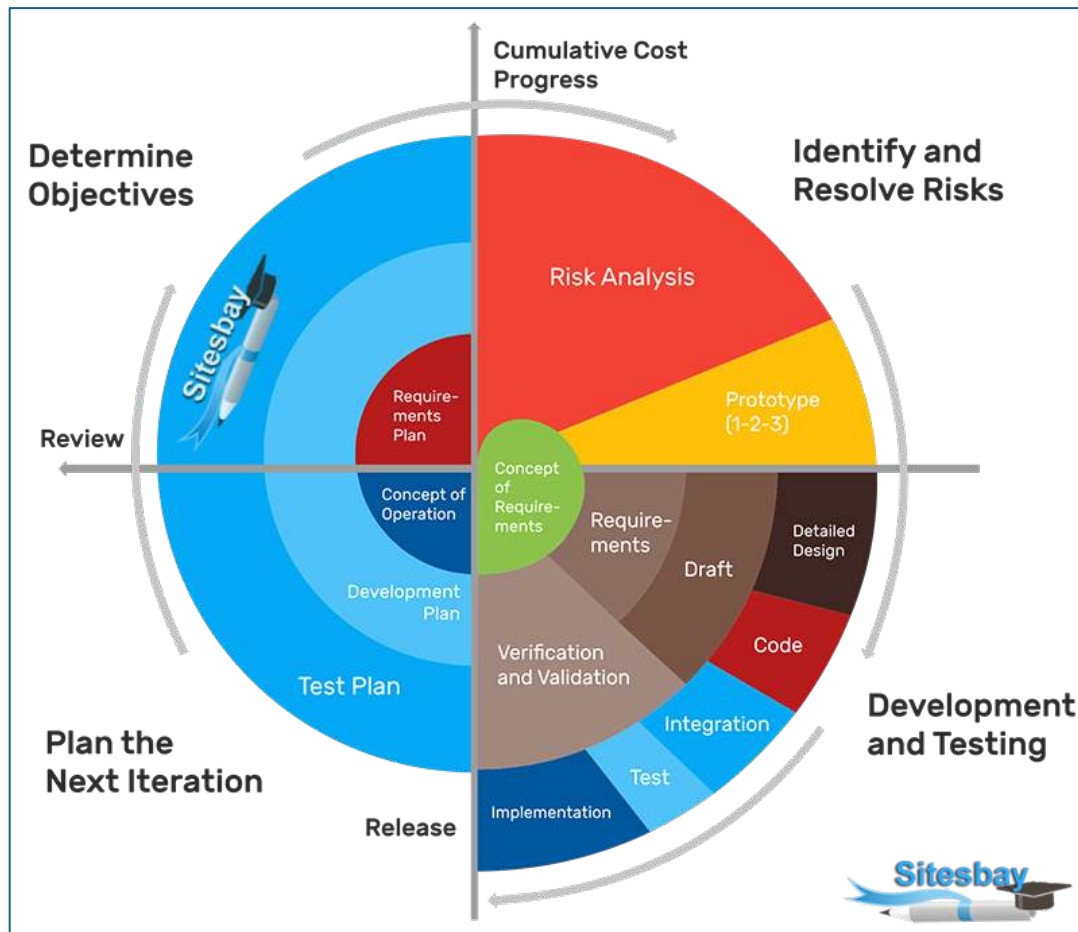


Figure 234: Spiral methodology.

(Sitesbay, 2024)

7.4. Appendix D: Requirement analysis

1. System: Windows OS or Mac OS
2. Text Editor/ IDE: Visual Studio Code

Visual Studio Code is lightweight text editor software that supports the tech stack mentioned in this project well. Many other text editors or IDE were taken into consideration like android studio. However, different code platform requires different setup configurations. Since visual studio code is very popular amongst most developers. Many extensions and packages are also available on visual studio which will make the development process easier.

3. Postman/Thunder-client: Postman and thunder-client are highly useful for testing out API during backend development. These tools not only help to check endpoints but also provides a various option for manipulating data sent to the backend like headers or cookie. Extension for thunder-client in VS Code was used, however; for postman the application was used.
4. Git: Git is highly required for this project development for version controlling purposes to prevent any major errors that might occur when developing (Jacobs, et al., 2022). This means the git, for every commit or for every permanent change made, it creates a track, which is highly useful for rollback purpose if any unexpected errors are encountered.
5. Frontend Development: JavaScript, React-Native, React.

JavaScript was mostly popular for building logic in client-side for websites. Every logic handling part like dynamic data changing that happens in any websites is due to the JavaScript code behind it (Amazon Web Services, Inc., 2025). However, with time JavaScript became popular for server-side development as well. Ever since JS became popular amongst developers and the community grew big.

React was developed by Meta (formerly known as Facebook) for web-app development. React replaces traditional way of web-development, where pages are also developed on JavaScript.

React-Native is used for the development of this project for mobile application, as it helps in cross-platform development, and is entirely based on JavaScript also developed by Meta.

6. Backend Development: Express.js, Node.js

For the simplification of development, JavaScript is used for the backend as well. Similarly, node.js is built on a JavaScript engine that compiles the code directly into machine-native code (OpenJs Foundation, 2024).

Express JS is popular for routing purpose in backend. It handles all the routing related functionalities like handling post, get, put, or delete requests.

7. MongoDB: Since this project may require constant changes in features that might alter the data structure, the NoSQL-based database MongoDB is chosen.

7.5. Appendix E: Logical data model

Logical data model is used to define the schema of the organization rather than defining the database. This is not dependent on the database; however, it is similar as it is a collection of entities and attributes. Later, the logical data model is converted into a physical data model (IBM, 2021).

7.6. Appendix F: Client description and requirements

Client Description: Tripe R, a travels and tour company, recently coming into market for exposure and working on travel sector of Nepal has been chosen to be the client of this application. Mrs. Ambika Gyawali, the managing director of this company wanted a similar application that could be beneficial for the tourists here in Nepal.

Mrs. Gyawali had been actively seeking a developer for similar application for it to be provided to their customers. The meeting with the client made it clear that this project would fit the best as per his requirements and needs.

Client Requirements:

- The application should let any users to be registered and logged in.
- The users of the application should have their data properly encrypted during verification, login or registration process.
- The users should be restricted to perform actions if they are not fully verified.
- The application should be made simple and minimal for avoiding confusion regarding functions from within the app.
- In case of system failure, the crash should be handled properly by the system.
- The user should have the ability to delete the profile if approved by the admin.
- User should be recommended with nearby places of preferred distance, along with a search feature.
- An admin panel should be created for handling all the user related activities.

7.7. Appendix G: Post survey result

These are the insights of the feedback collected:

- All the participants overall had a good experience with the application.
- All the participants expected the application to be beneficial.
- Some of the participants struggled with the registration step as it was lengthy.
- According to the survey, the participants were also confident with the accuracy of the results.
- Almost all the survey participants were satisfied with the user-friendliness of the application.
- Survey participants responded in support of TourNepal being significantly better than other travel companions in the current context of tourists coming to visit Nepal.
- Overall, the participants shared a positive view and experience of the application.
- The participants also believed the application to be more useful and could be improved further.

7.7.1. Questions and responses

I) Overall application experience

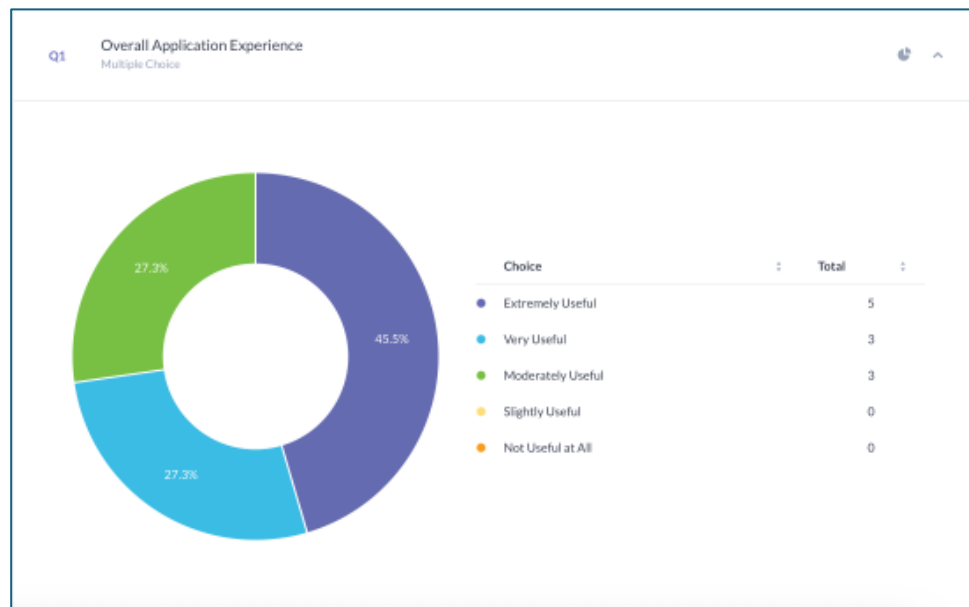


Figure 235: Overall application experience (survey response)

II) Which features are most relevant to user's daily use?

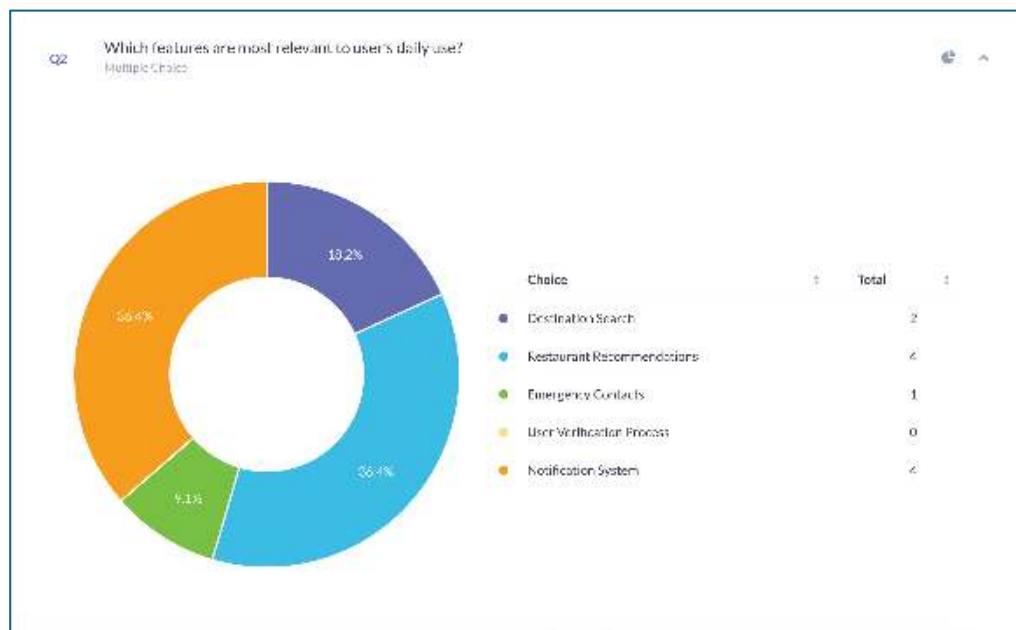


Figure 236: Most relevant features (survey response)

III) How effectively can you use the app's search features?

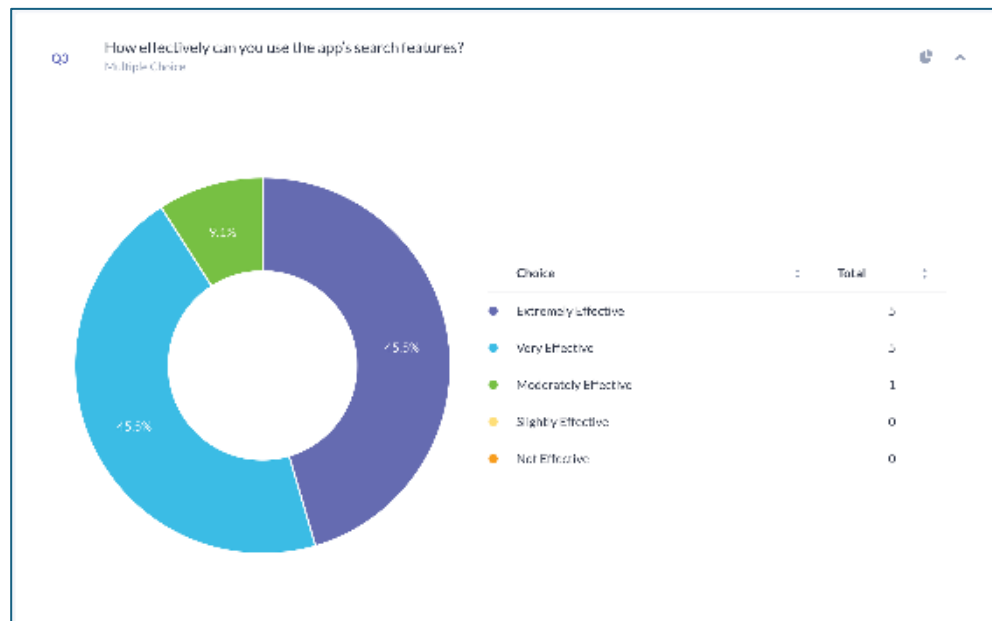


Figure 237: Search feature (survey response)

IV) How confident are you in the accuracy of information provided by the app?

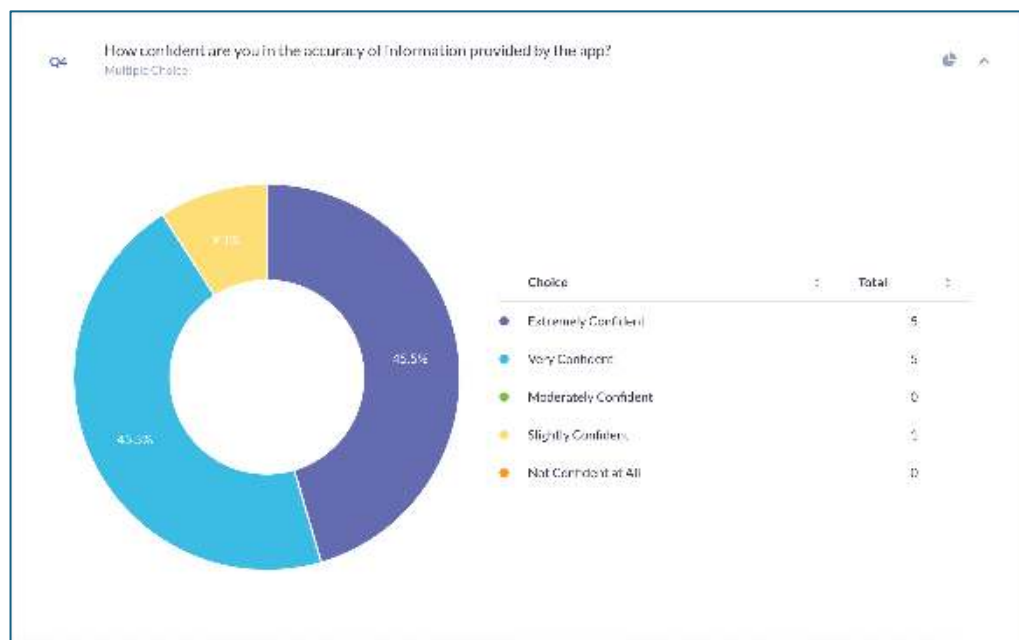


Figure 238: Application accuracy (survey response)

V) From your professional perspective, how user-friendly is the application?

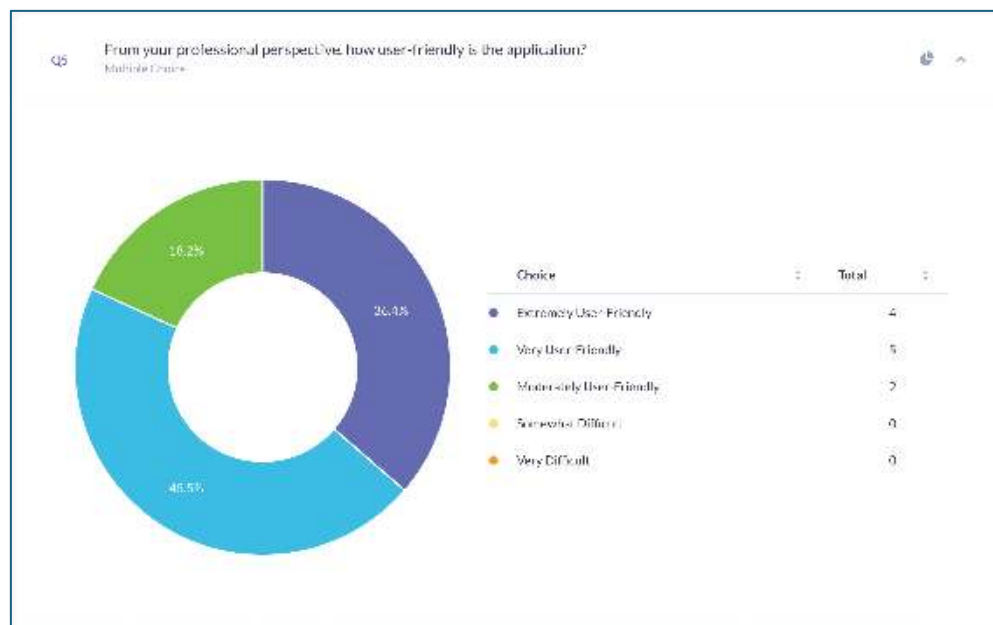


Figure 239: User-friendliness (survey response)

VI) How does this app compare to other travel support tools you've used? (In context of tourist visiting in Nepal)

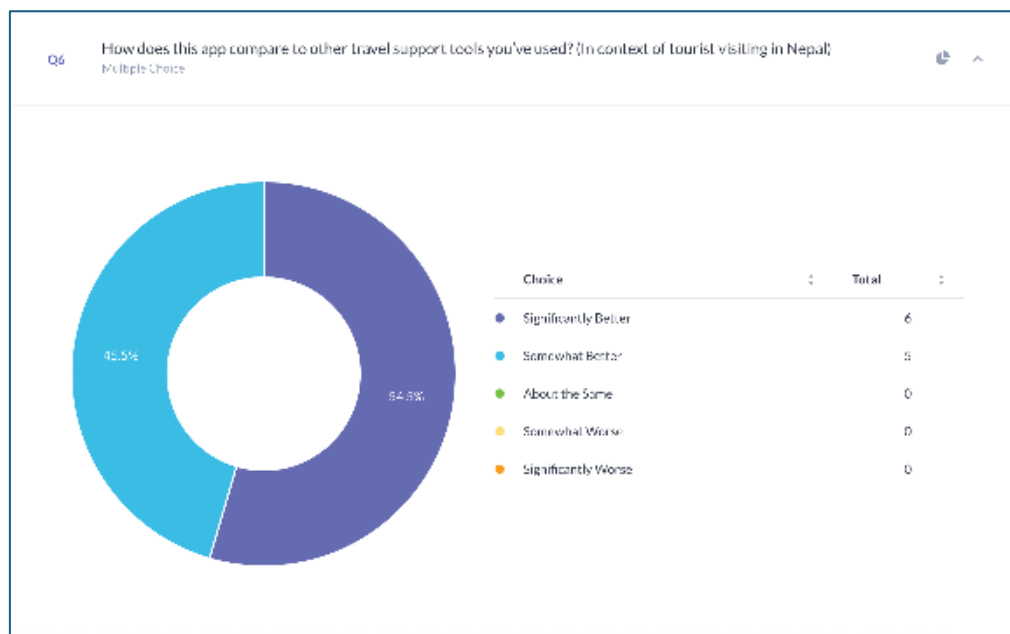


Figure 240: Similar application comparison (survey response)

VII) How useful are the app's notification features for your work?

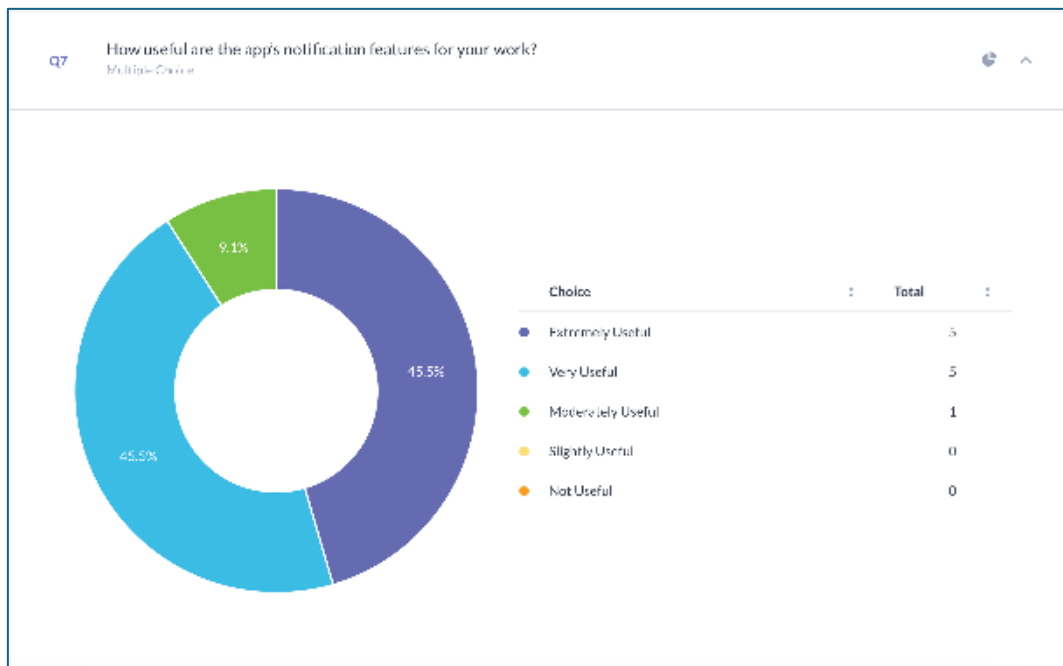


Figure 241: Application usefulness (survey response)

VIII) How much potential do you see for this app in expanding travel services?

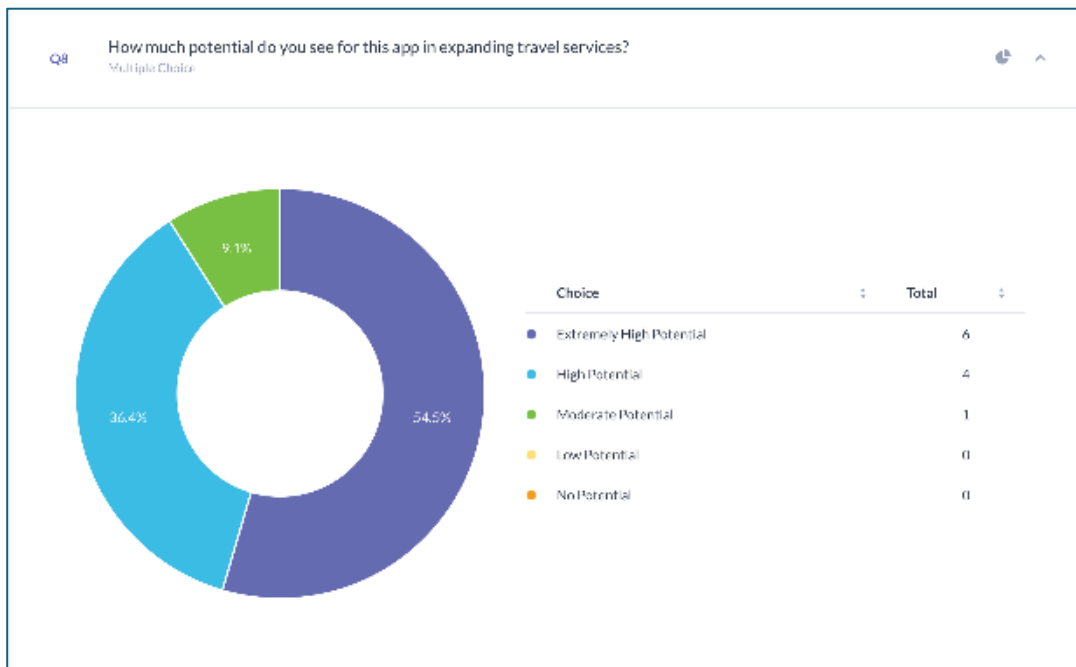


Figure 242: Potentiality of application (survey response)

IX) How easy was the registration and document upload process?

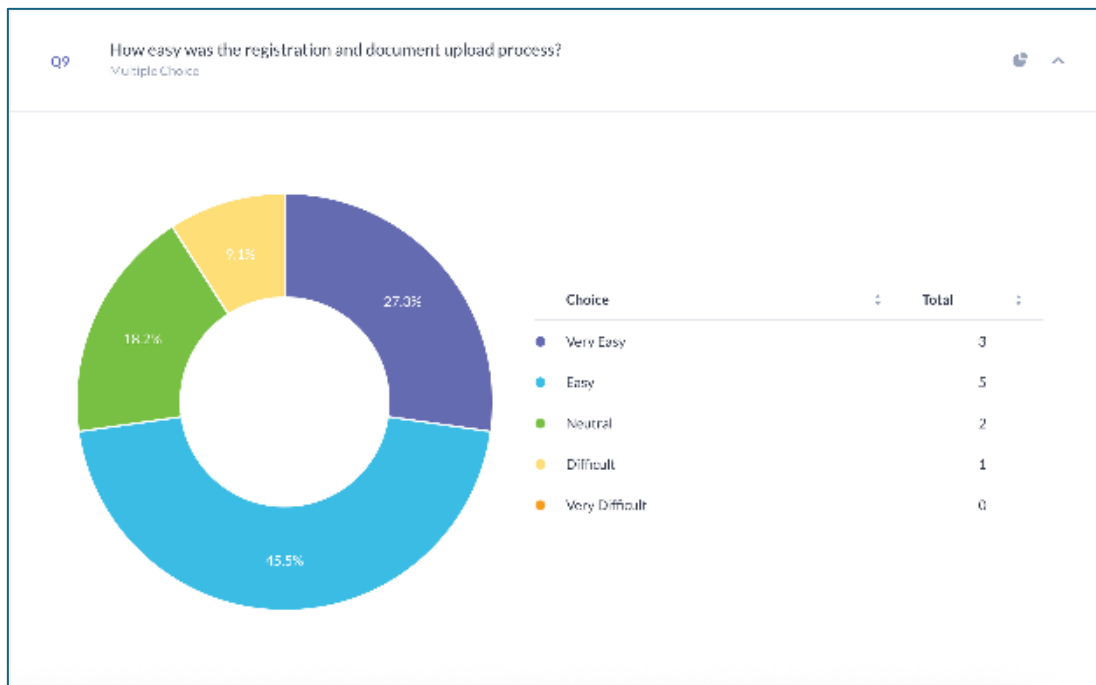


Figure 243: Registration process (survey response)

X) How effective did you find the destination and restaurant search functionality?

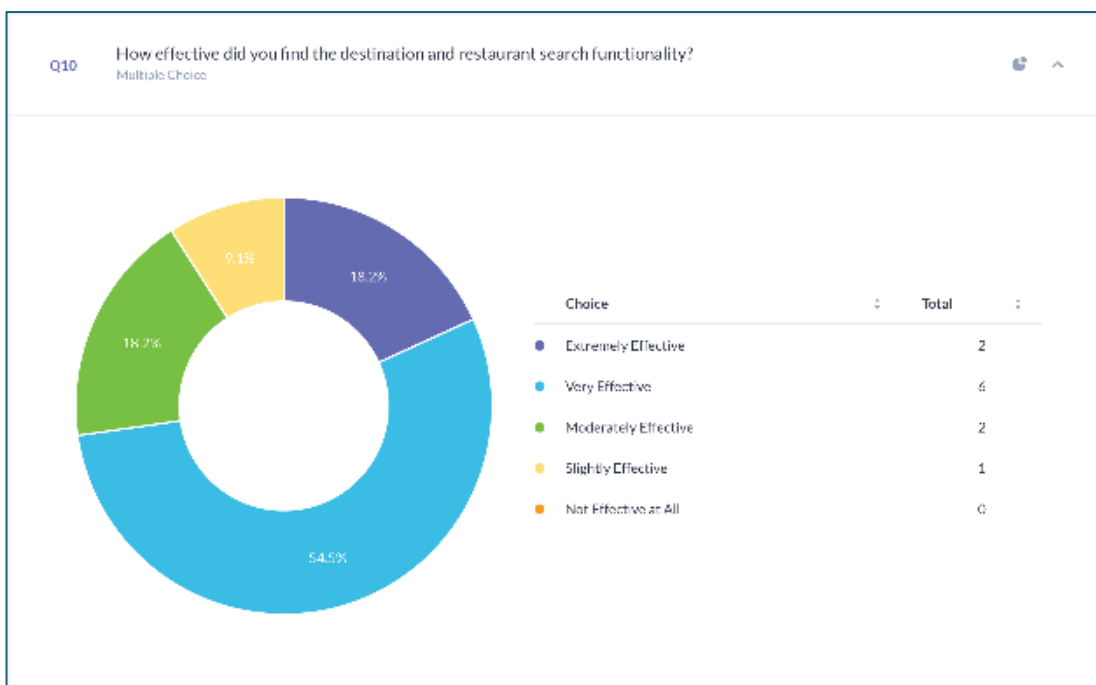


Figure 244: Search functionality effectiveness (survey response)

XI) How effective is the document-verified review system?

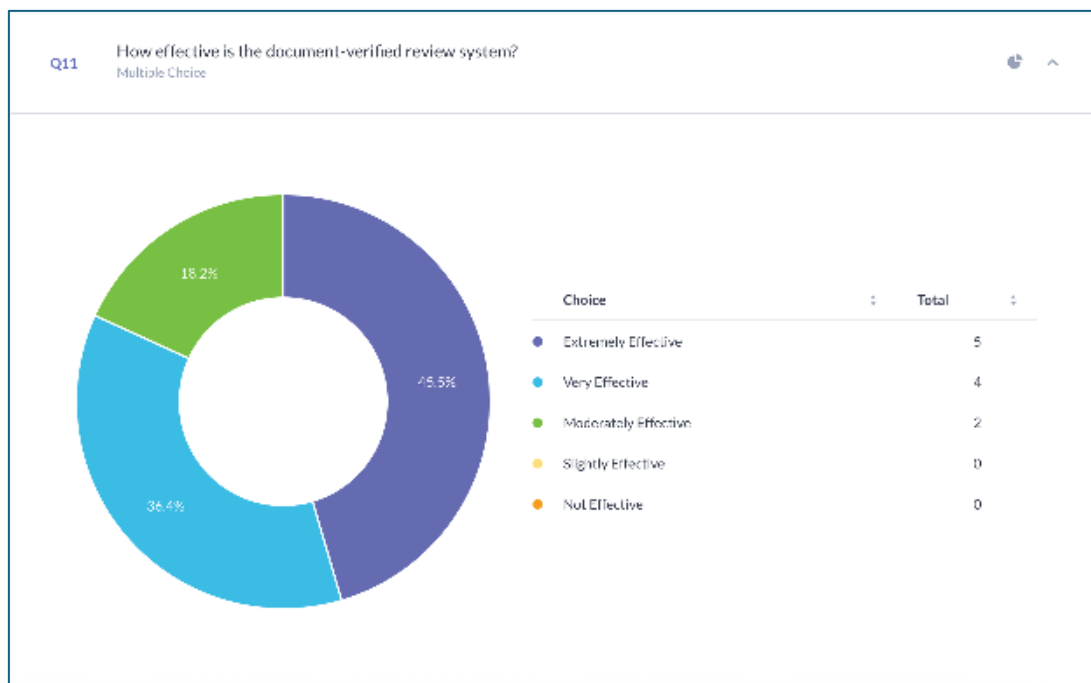


Figure 245: Review system effectiveness (survey response)

XII) Additional comments or specific insights about the TourNepal application:

Q12 Additional comments or specific insights about the TourNepal application:
Essay

Date	Answers
Mar 28	The reason I put somewhat better in-comparison to other applications is because of missing features, like hotel booking.
Mar 28	The application has very high potential, and can be made much much better.
Mar 27	The application is great so far. However, some limitations could be found in the application.

Figure 246: Additional comments (survey response)

7.8. Appendix G: Legal, Social and Ethical Issues

7.8.1. Legal issues

Another legal concern is how the app uses services like Google Maps. The app uses Google Maps for navigation, which makes it easy for tourists to directly open the directions to the destination they want to visit. While this feature is very helpful, the app has to follow Google's terms of service. If the app breaks Google's rules, such as by going over the free API usage limits or not crediting them properly, then it could result in legal issues like getting banned from using their services and facing high charges for the misuse (Google, 2025).

Additionally, this app also comes with the concerns about discrimination against people. For example, if someone's visa gets rejected based on their nationality, visa type, or other reasons, it might be seen as unfair and raise a concern about discrimination leading to a bad reputation of the app as well as causing someone to file a lawsuit against the app.

7.8.2. Social issues

Another social issue of this app related to accessibility and the digital knowledge of the users. Some tourists who visit Nepal are not always young, good with technology, and carry a smartphone with data. Older tourists, those who come from places with less technological knowledge, and those who don't have data on their smartphones, might have a hard time using the app or may not be able to use the app at all. This means that although the app is developed to help people, it might only be helpful to a specific group of people.

Additionally, the app suggests popular tourist destinations, restaurants, and services, which is one of the most important features of this app. However, this could lead small and local businesses to be left out. For example, there are many small businesses in Nepal that don't have online reviews or websites, and if tourists only go to the top restaurants and places suggested by the app, it might harm the smaller businesses that only rely on foot traffic. This could also stop tourists from connecting with Nepali people, exchanging cultures, learning from each other and experiencing the authentic experience.

7.8.3. Ethical issues

Since the app provides recommendations for destinations and restaurant it might look there is a paid promotions instead of suggestions based on fair criteria. If businesses can pay the app to be featured on it without informing the user, it might raise ethical concerns and biases. Also, putting profit over accuracy can damage the app's reputation. Furthermore, only featuring businesses that are popular among tourists can be unfair to small local businesses. Instead of only suggesting popular destinations, suggesting diverse destinations can not only enhance the tourists' experience in Nepal but also be fair to all kinds of businesses and destinations. Finally, although English is an international language, many tourists who visit Nepal might not speak English and if the app has its privacy policy in a language users don't understand then they aren't informed about how their data is being used which also raises ethical issues.

7.9. Appendix H: Designs

7.9.1. Gantt Chart

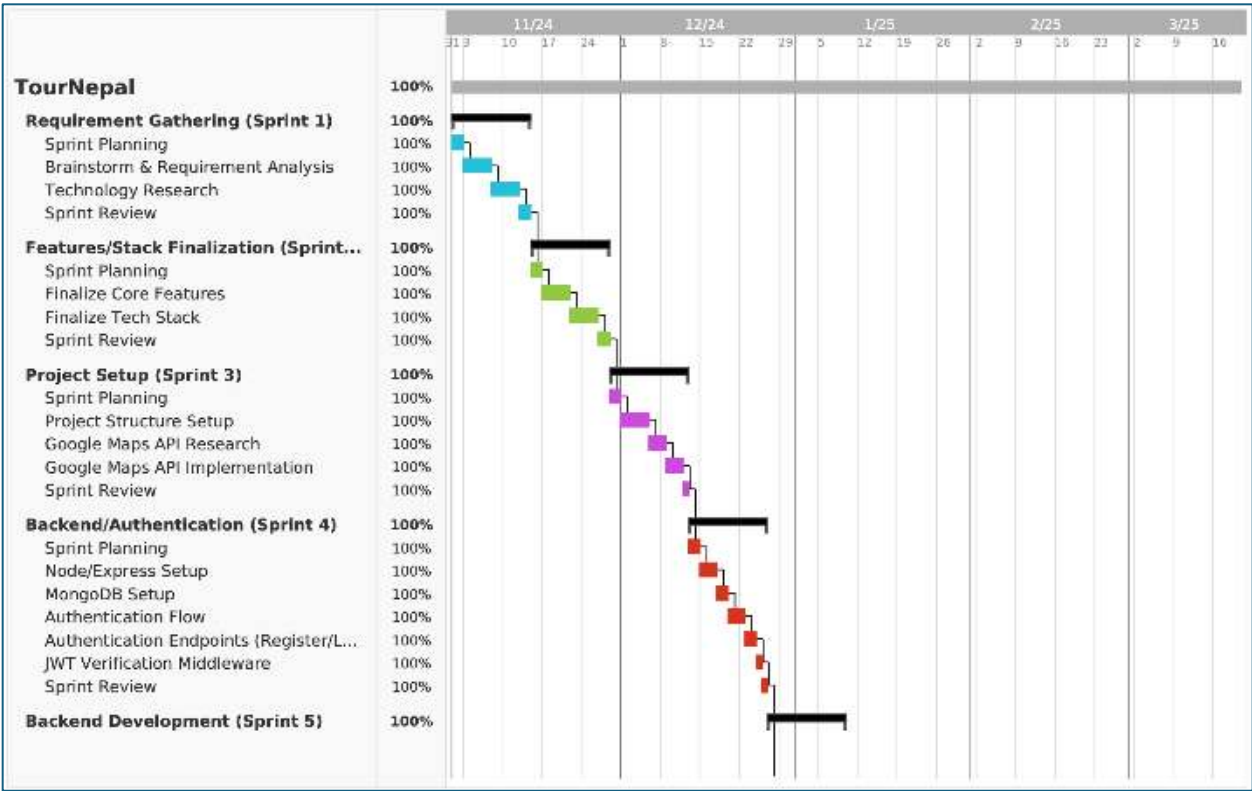


Figure 247: Gantt chart (1)

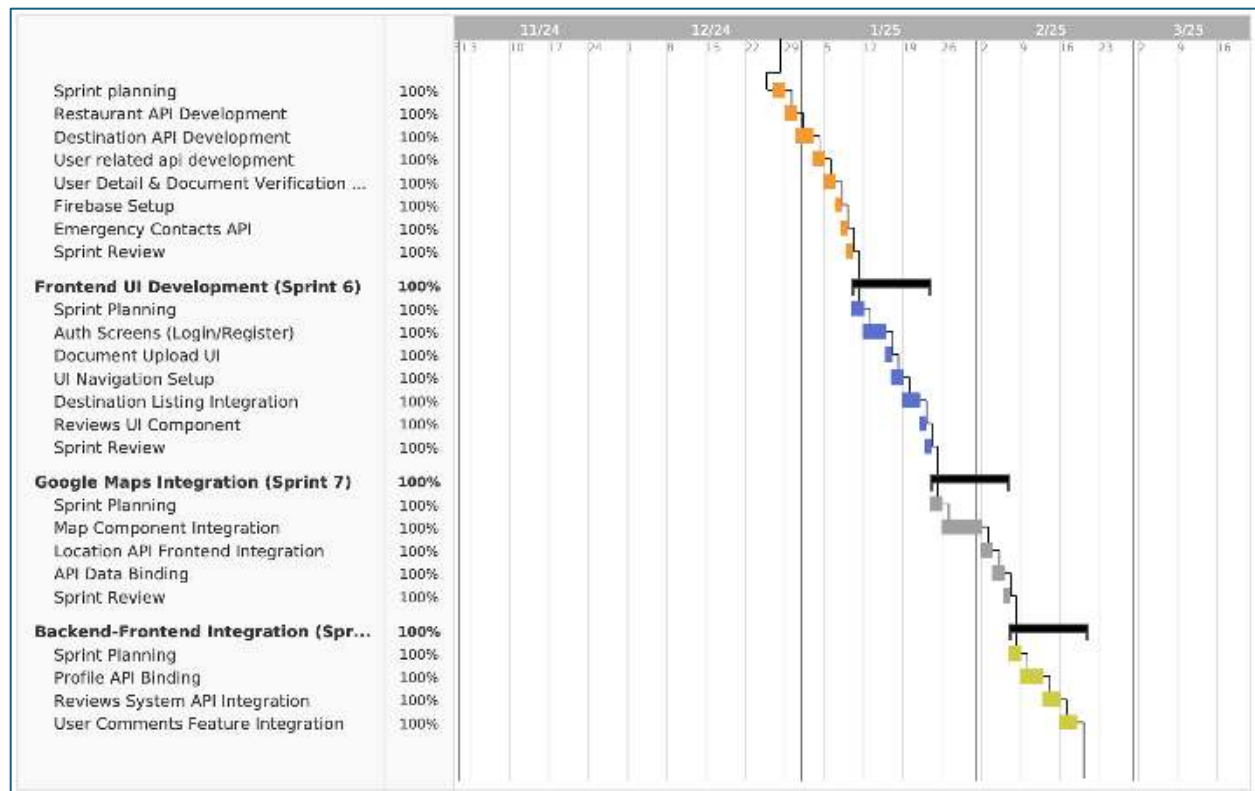


Figure 248: Gantt chart (2)

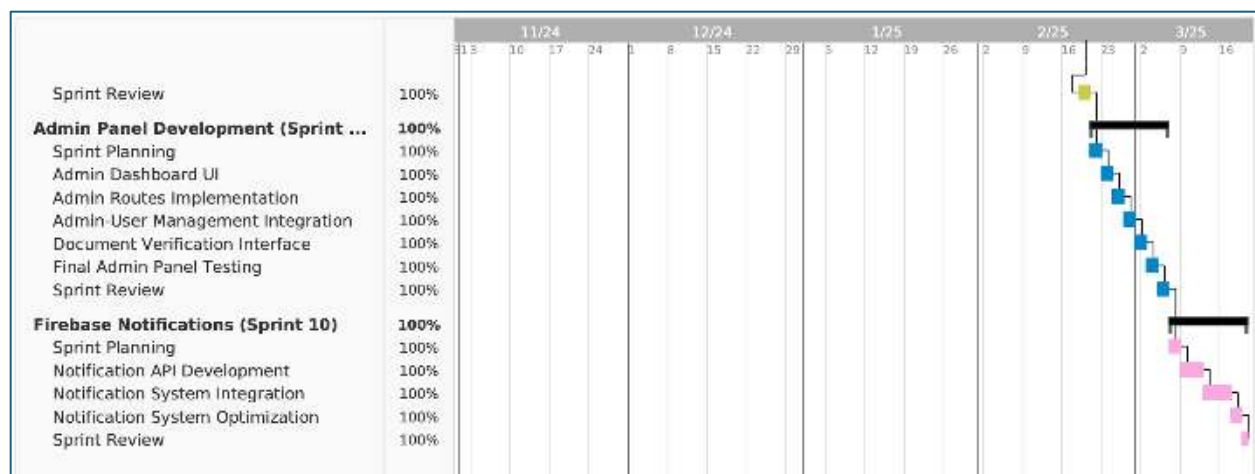


Figure 249:: Gantt chart (3)

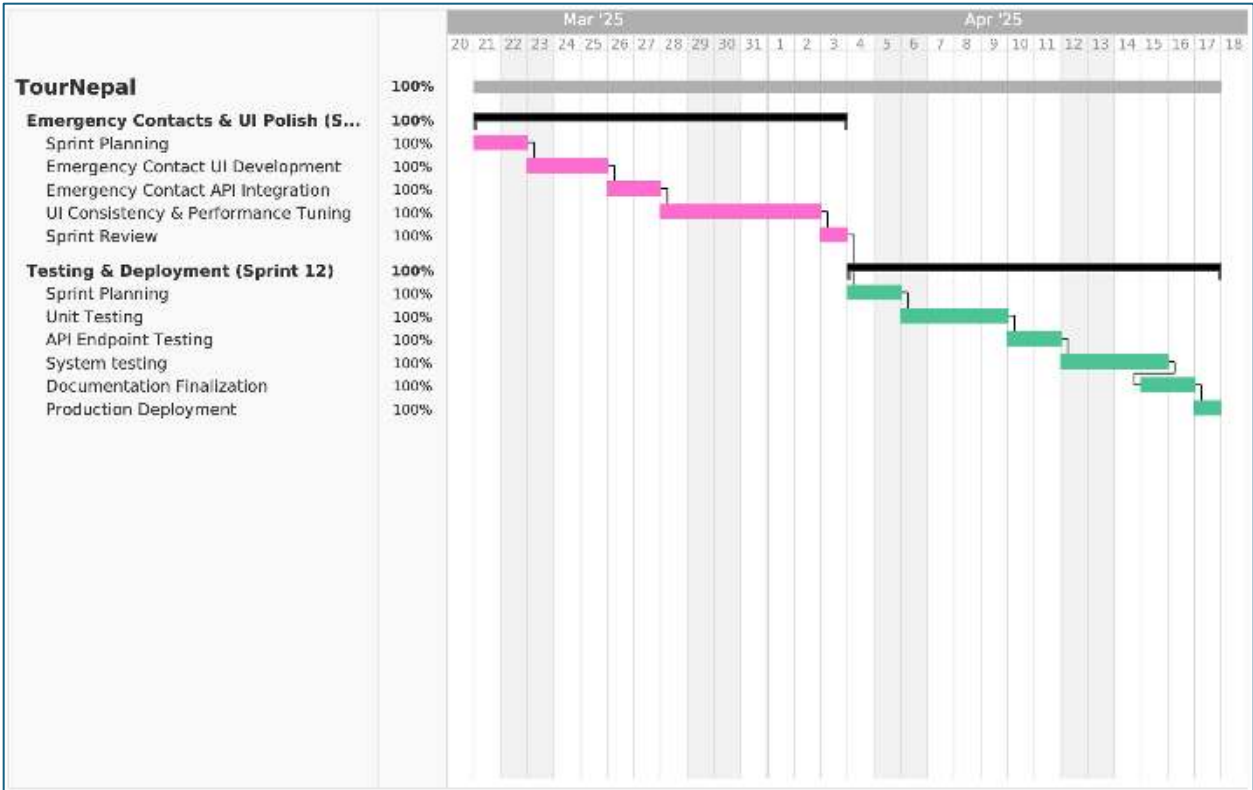


Figure 250: Gantt chart (4)

7.9.2. Work Breakdown Structure (WBS)

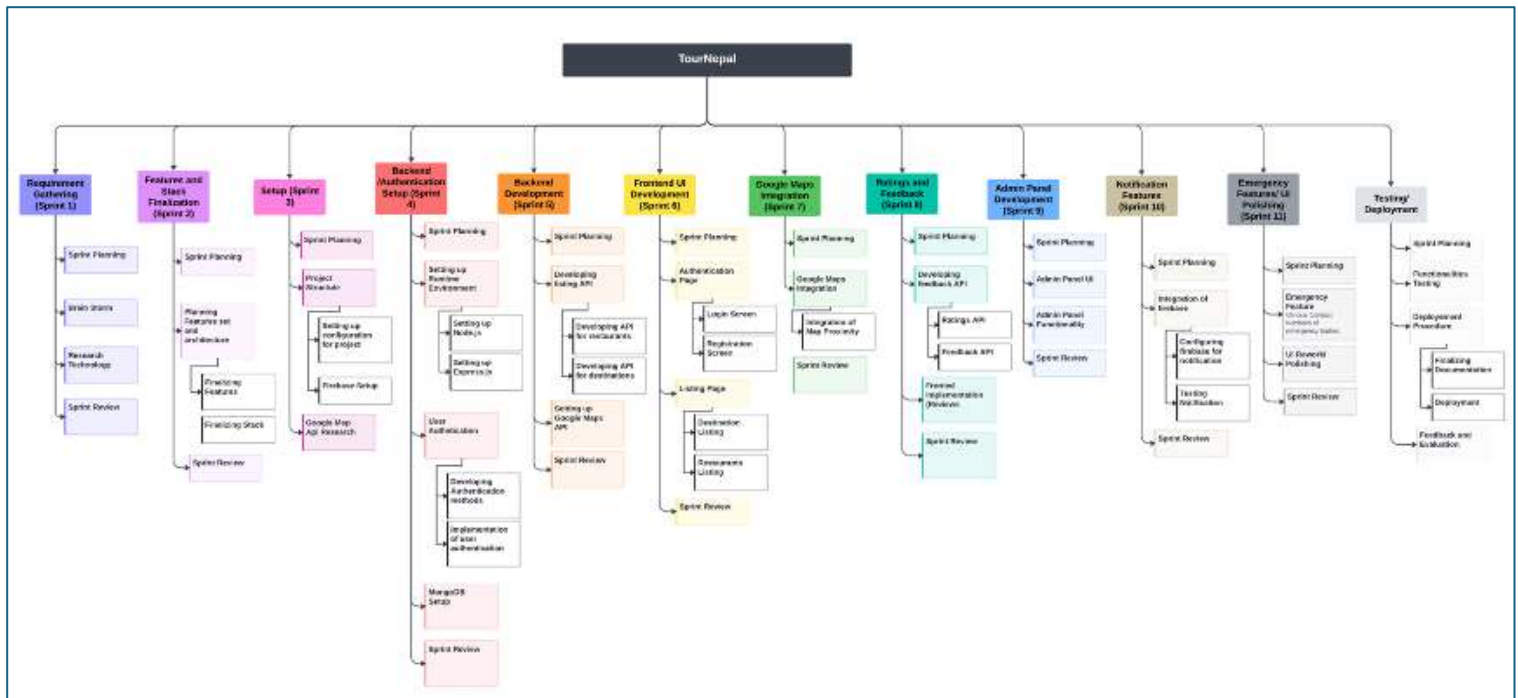


Figure 251: Work breakdown structure (WBS)

7.9.3. Use Case

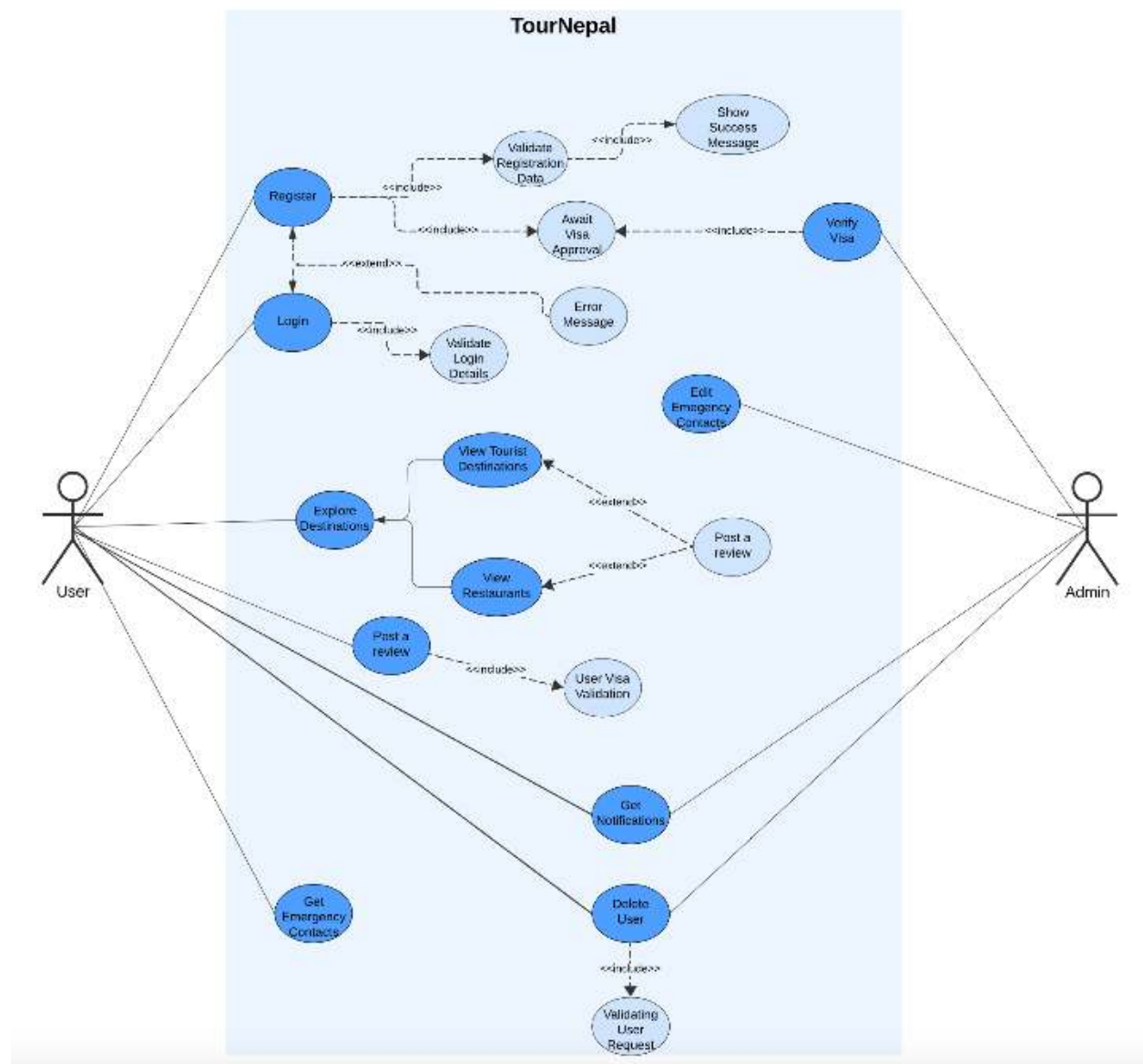


Figure 252: Use case diagram

7.9.4. Data-flow diagram

Since the data flow and logic are not complex of this application level one and above are not necessary for data flow diagram (DFD).

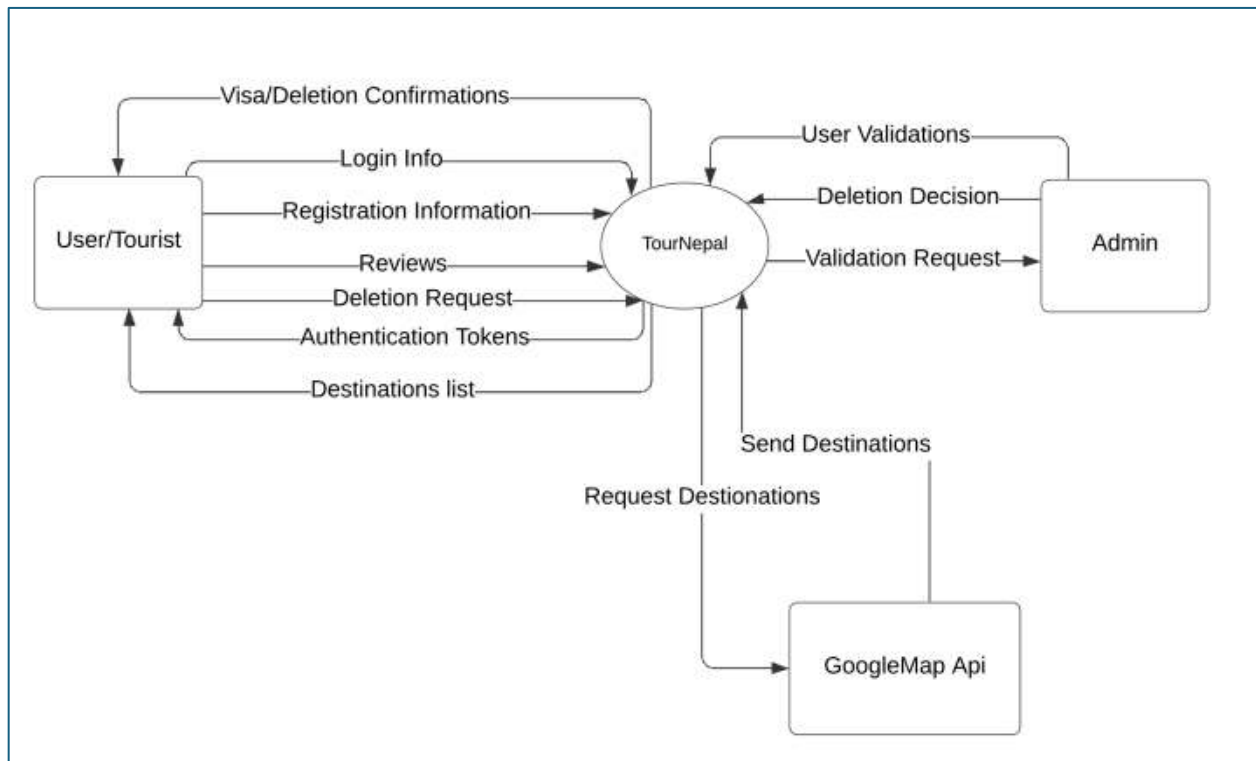


Figure 253: Dataflow diagram

7.9.5. System flowchart

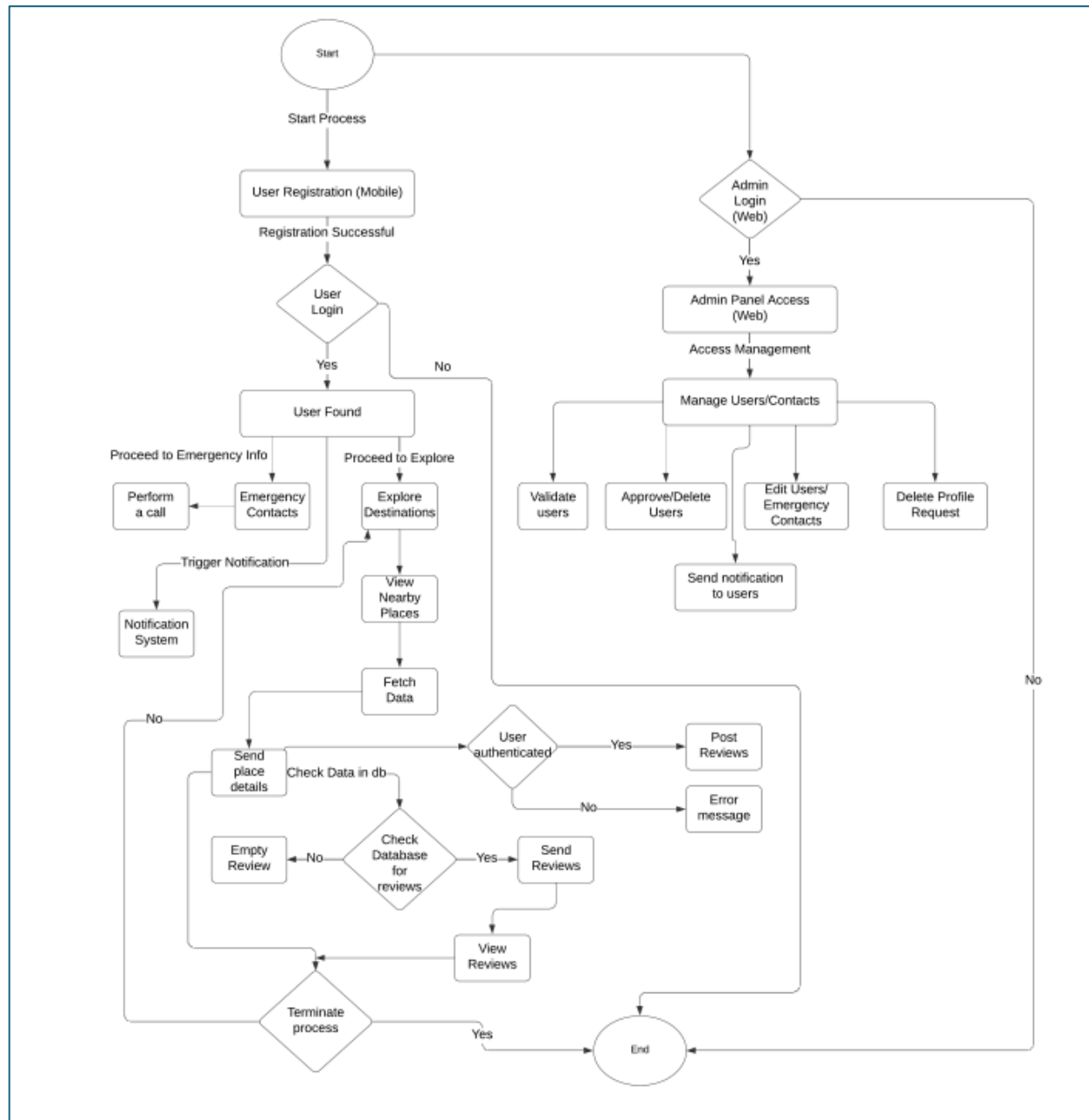


Figure 254: System flowchart

7.9.6. Sequence diagram

(Sequence diagrams if not available can be found in 3.5.6)

I) Login

The login process is kept simple. After the successful registration, the user access token and refresh token are assigned to the user. Access token expiration is kept short, which will be used to perform actions that require verification in the backend. The refresh token is used to generate a new access token when the old one is expired. Both the tokens are kept in the user's device storage secured.

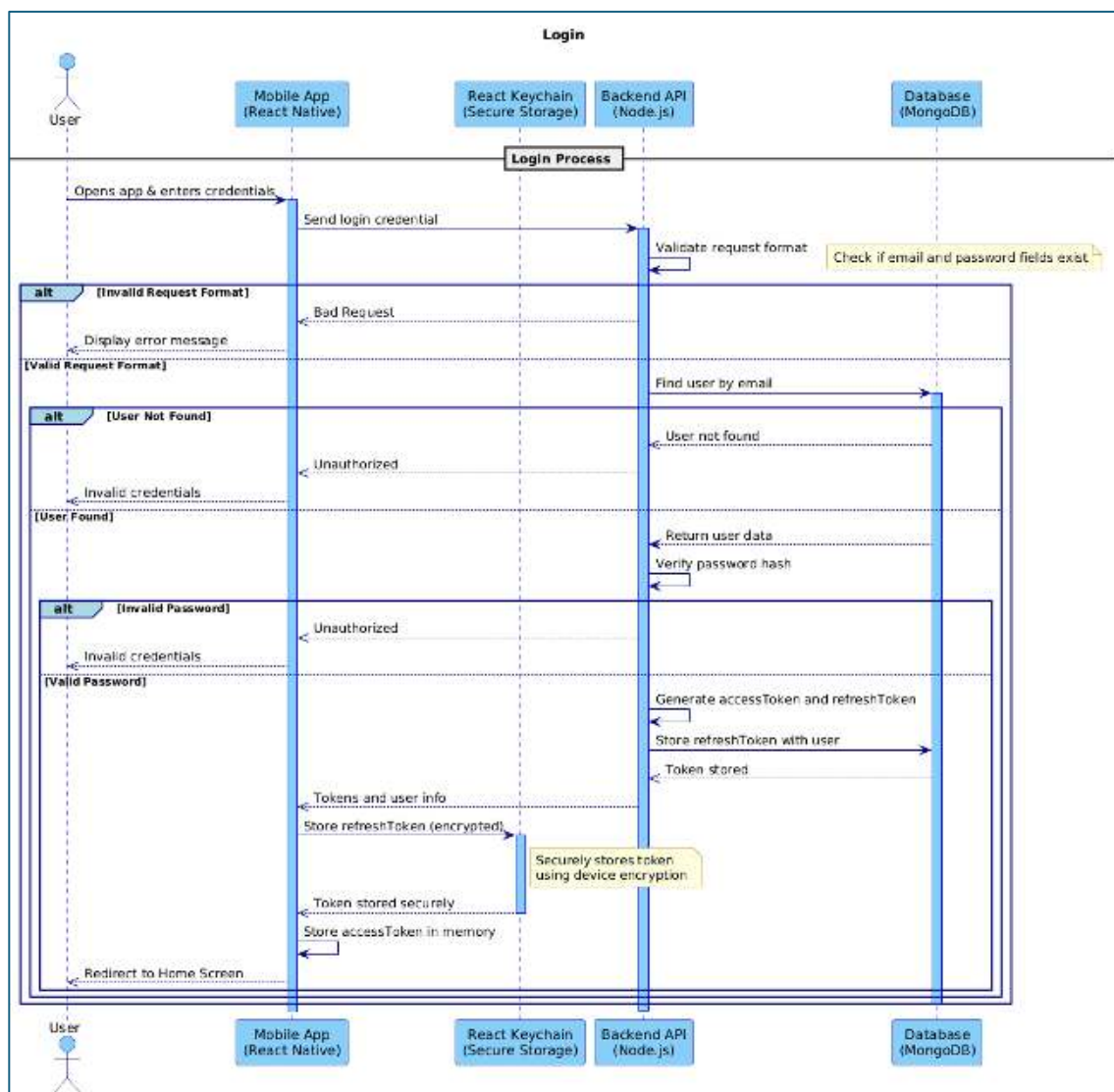


Figure 255: Login process sequence diagram

II) Registration

User can be registered in two different ways. One being user registration via application and the other being admin user creation process.

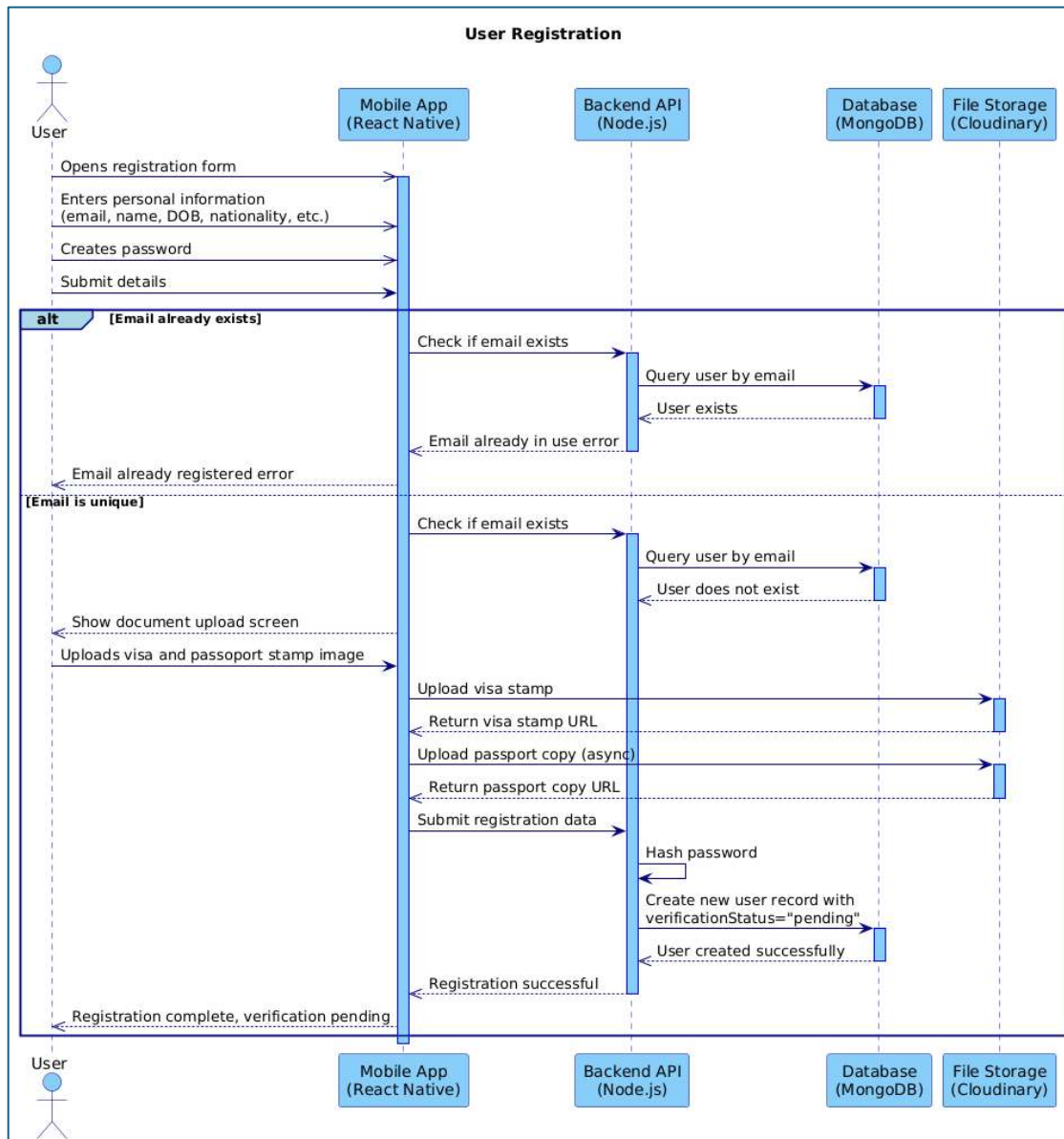


Figure 256: Registration process sequence diagram

During initial registration, all the fields are filled out with passed values, but some fields like `verificationStatus`, `deletionRequest`, and `notificationList` are set by the system. This will make sure to set the user into default state, and if any parameters of those are passed it make any change.

```
23     try {
24       //encrypt the password
25       const hashedPwd = await bcrypt.hash(password, 10);
26
27       //create and store the new user
28       const result = await User.create({
29         "email": email,
30         "password": hashedPwd,
31         "firstname":firstname,
32         "lastname":lastname,
33         "dateOfBirth": dateOfBirth,
34         "visaStamp": visaStamp,
35         "passportCopy":passportCopy,
36         "verificationStatus":"pending" ,
37         "dateOfEntry":dateOfEntry,
38         "nationality":nationality,
39         "intendedDays":intendedDays,
40         "deletionRequest":false,
41         "notificationList":[]
42       });
43
44       console.log(result);
45
46       res.status(201).json({ 'success': `New user ${firstname} created!` });
47     } catch (err) {
48       res.status(500).json({ 'message': err.message });
49       console.log(err.message)
50     }
51   }
```

Figure 257: User registration (code snippet)

However, this is not the case when an admin is creating a new user; other fields like visaStamp and passportCopy are also custom set by the system. For verificationStatus, it is set to rejected, as it is required to be uploaded by the user, which can be done through the profile section after the user logs in.

```
40     if (existingUser) return res.sendStatus(409); //Conflict
41
42     const hashedPwd = await bcrypt.hash(password, 10);
43     const result = await User.create({
44         email: email,
45         password: hashedPwd,
46         firstname: firstname,
47         lastname: lastname,
48         dateOfBirth: dateOfBirth,
49         visaStamp: "none",
50         passportCopy: "none",
51         verificationStatus: "rejected",
52         dateOfEntry: dateOfEntry,
53         nationality: nationality,
54         intendedDays: intendedDays,
55         deletionRequest: false,
56         notificationToken:"",
57         notificationList:[],
58     });
59     console.log(result);
60     res.status(201).json({ success: `New user ${firstname} created!` });
61 } catch (err) {
62     res.status(500).json({ message: err.message });
63     console.log(err.message);
64 }
```

Figure 258: User registration by admin (code snippet)

III) Authentication flow

Before performing any action that requires verification, several steps are performed in order to verify the authentication process. Every time a verification is needed for any action, the access token from the mobile application storage is retrieved and sent along with request headers and payload. If the access token is valid, a response is sent; however, if the access token is expired, a refresh token is accessed from the mobile storage and sent for a new access token request. After getting a new access token from the server, it is replaced with the old access token in the storage, and the request is re-requested.

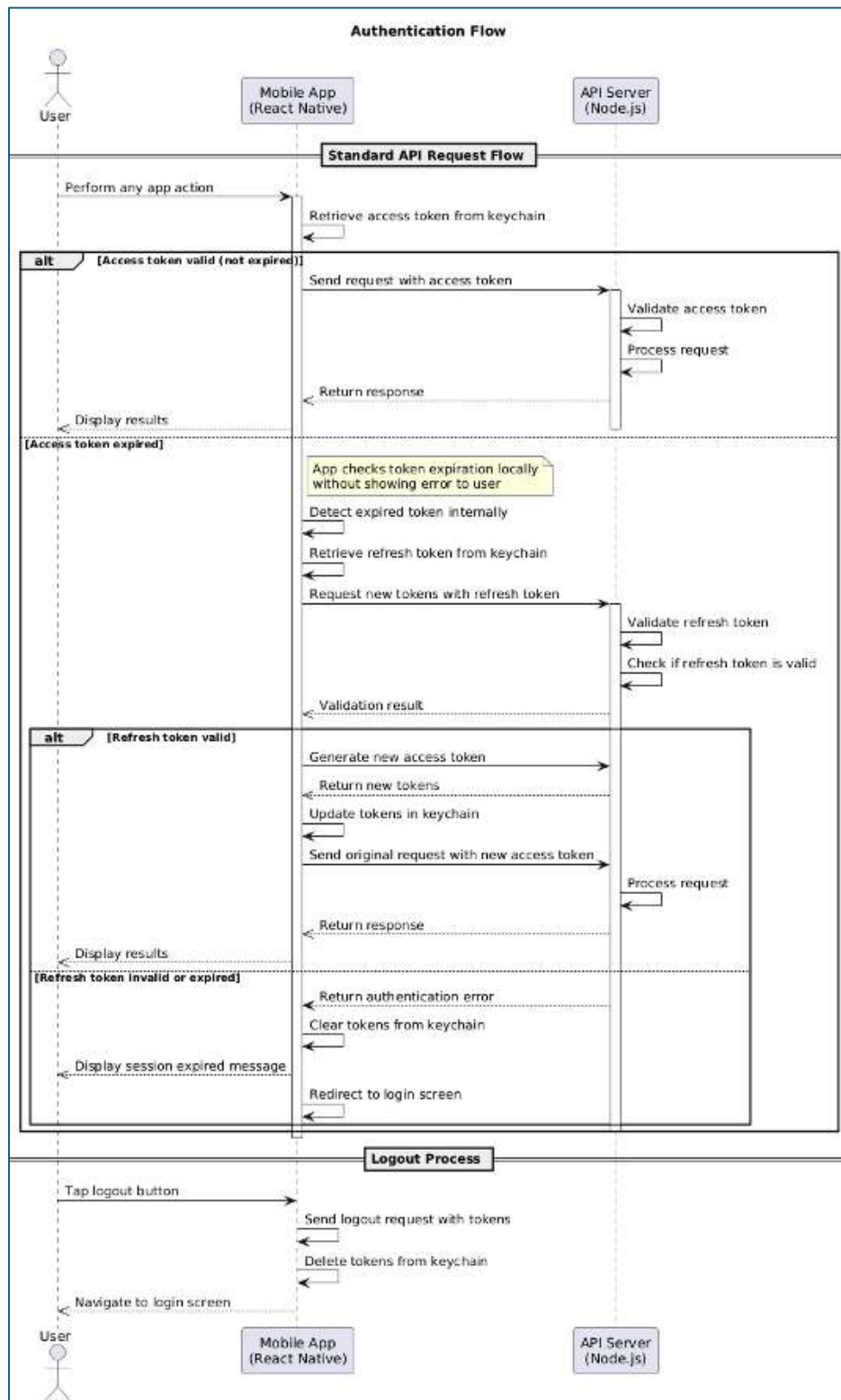
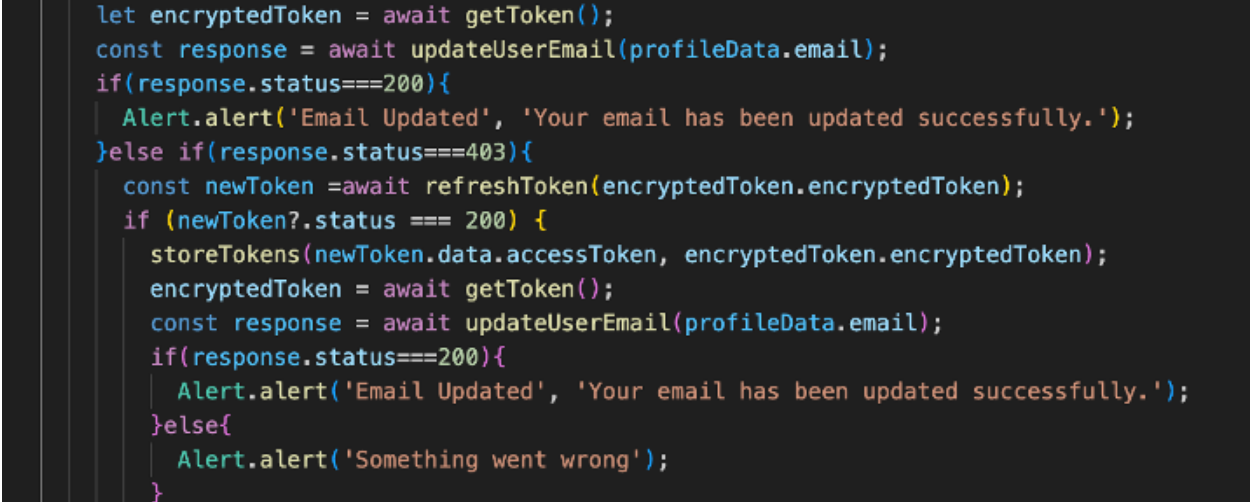


Figure 259: Authentication flow sequence diagram

```
let encryptedToken = await getToken();
const response = await updateUserEmail(profileData.email);
if(response.status===200){
  Alert.alert('Email Updated', 'Your email has been updated successfully.');
```



```
}else if(response.status===403){
  const newToken =await refreshToken(encryptedToken.encryptedToken);
  if (newToken?.status === 200) {
    storeTokens(newToken.data.accessToken, encryptedToken.encryptedToken);
    encryptedToken = await getToken();
    const response = await updateUserEmail(profileData.email);
    if(response.status===200){
      Alert.alert('Email Updated', 'Your email has been updated successfully.');
```

```
}else{
  Alert.alert('Something went wrong');
```

```
}
```

Figure 260: Authentication flow code snippet

IV) Explore popular destinations

When user navigates to search section auto-search for restaurants within 5 K.m. is executed. User can change the destination types and, based on the set radius searched results are refreshed. Location of the user is saved and changed in global context. With help of the temporary saved location, nearby destinations are fetched.

If location is denied a request for location permission is also asked to the user.

```

useEffect(() => {
  Windsurf: Refactor | Explain | Generate JSDoc | X
  const getPlaces = async () => {

    setIsLoading(true);
    const places = await nearbyPlaces(radius * 1000, currentSelection, currentLocation);

    setIsLoading(false);
    setAllPlaces(places);
    setFilteredPlaces(places);
  };

  if (locationPermission && currentSelection !== 'all') {
    getPlaces();
  } else if (!locationPermission) {
    Alert.alert(
      'Location Permission Required',
      'Please enable location permission to get nearby places',
      [
        {
          text: 'Open Settings',
          onPress: () => Linking.sendIntent('android.settings.LOCATION_SOURCE_SETTINGS')
        },
        {
          text: 'Cancel',
          style: 'cancel'
        }
      ]
    );
  }
}, [radius, currentSelection, currentLocation]);

```

Figure 261: Location fetching code snippet

V) Request profile deletion

When a user requests a profile deletion, the `userDeletionRequest` status changes from false to true. The admin looks after the process for user account deletion. If the user request is granted, the true parameter is sent and the account is deleted from the database; however, if the request is rejected, the `deletionRequest` status changes back to false.

VI) Admin document approval

User needs to submit their document in order to get their user status verified for leaving reviews in any place. User is able to submit their documents via two different places,

during registration, and from the profile screen. If user documents are rejected by the user or if the user is created by an admin, they are required to upload their valid documents from the profile screen.

VII) Posting a review

Users are able to post a review on any place they travel. However, before posting a review, several checks are performed. In the mobile application, if the user verification status is verified, then only the user is able to leave a review. Similarly, an additional check is performed in the backend as well. When adding a review, the user verification status is checked from the database as well. This way, the user will not be able to post a review as the comment box will be disabled. Since the data stored on the frontend side is vulnerable, additional checks are performed in the backend.

7.9.7. Activity diagram

(Activity diagram for main features can be found in 3.5.7).

I) Login

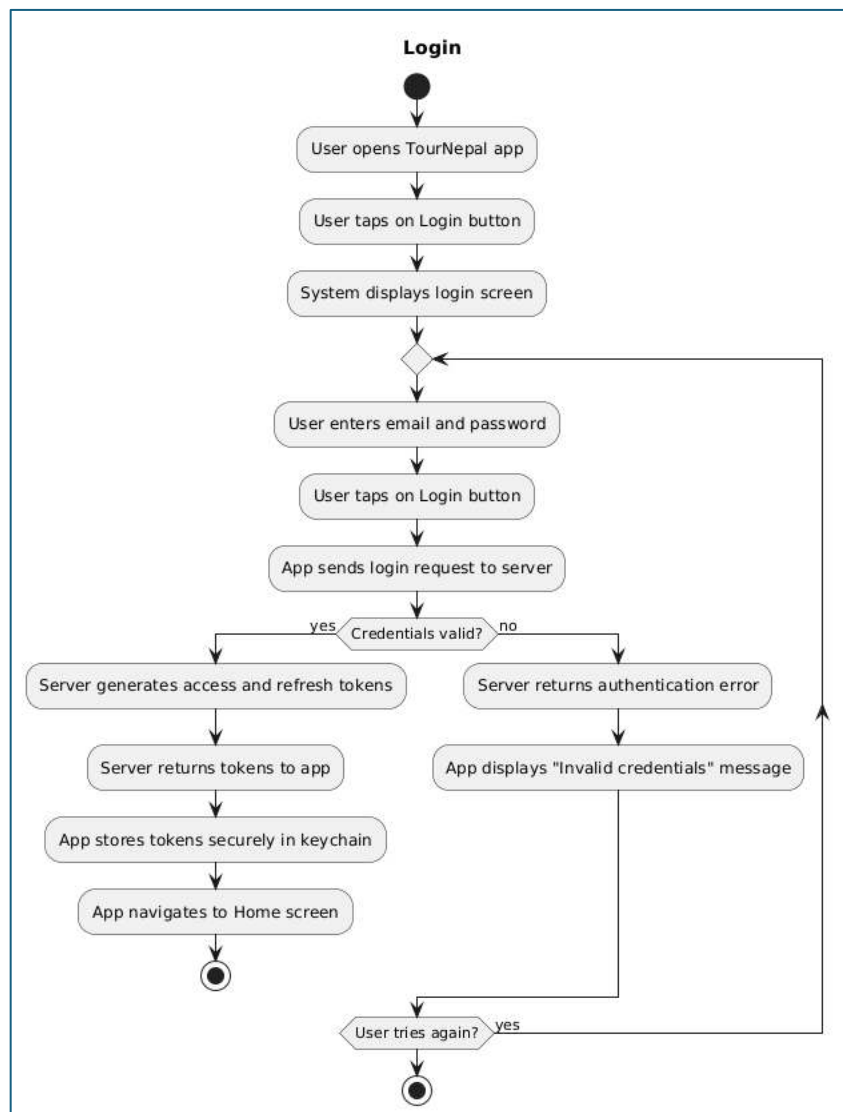


Figure 262: Login activity diagram

II) Registration

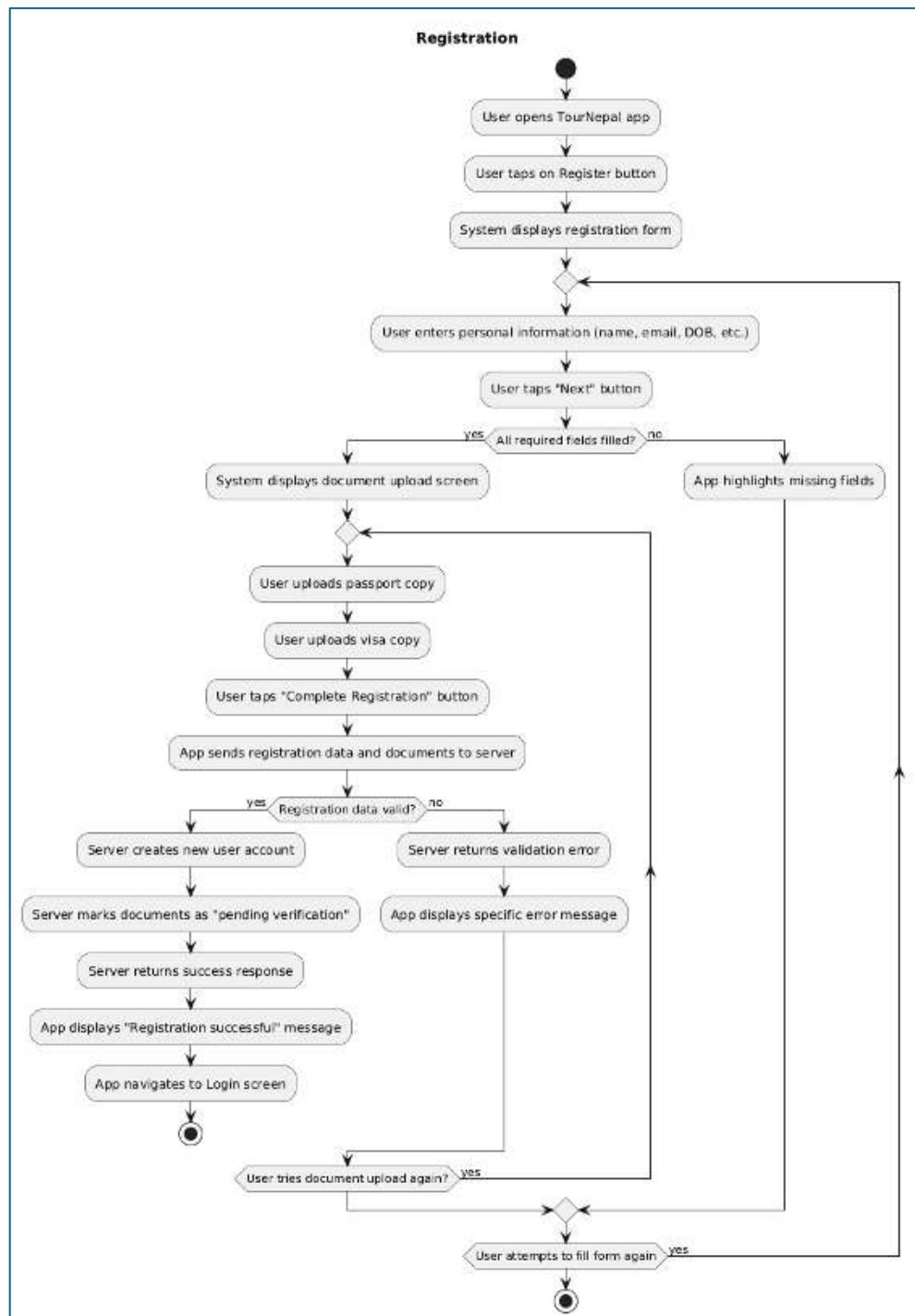


Figure 263: Registration activity diagram

III) Authentication flow

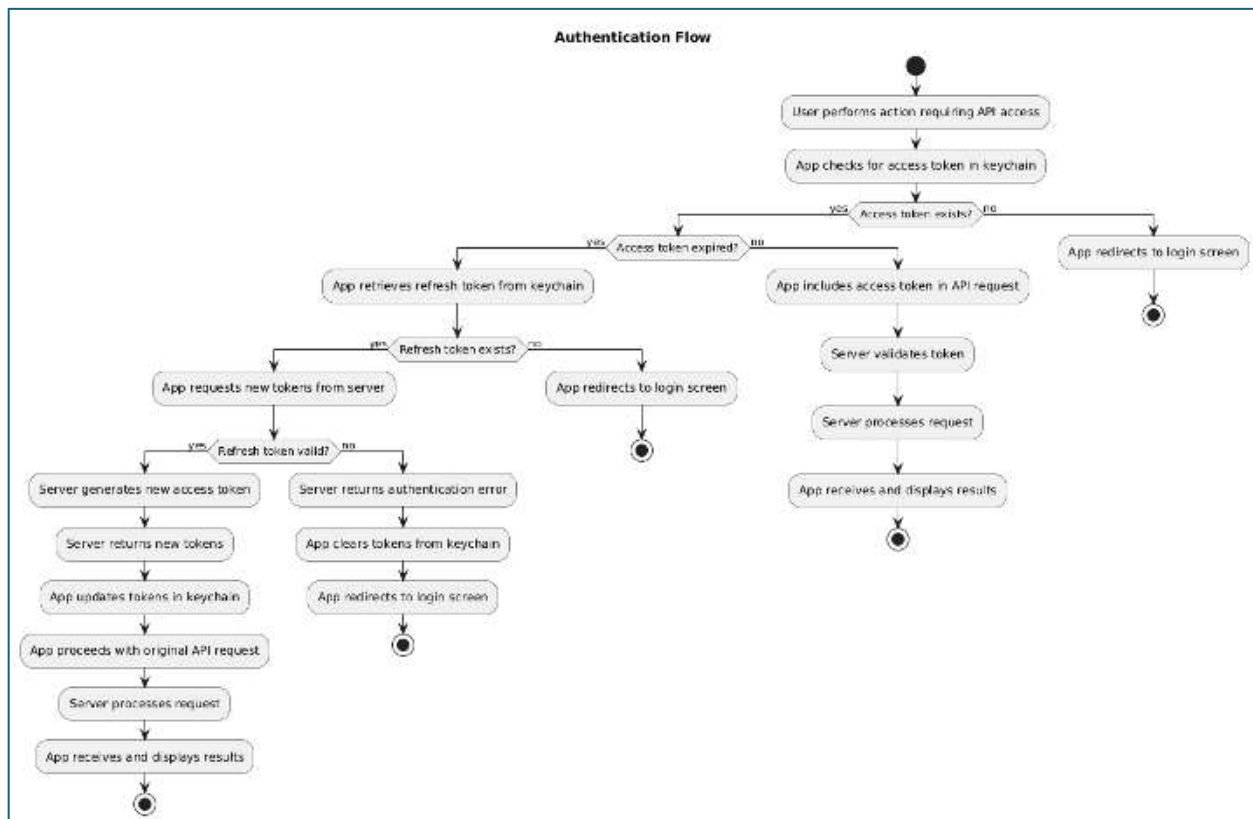


Figure 264: Authentication flow activity diagram

7.9.8. Wireframes

I) Login and registration

 TourNepal Email Password Don't have an account? Sign Up Login	 TourNepal Email Password Re-Enter Password Next Existing User? Login	 TourNepal Full Name Date of Birth Upload Passport Copy Upload Visa Stamp Sign Up
--	---	--

Figure 265: Login and registration wireframe

II) Mobile application screens

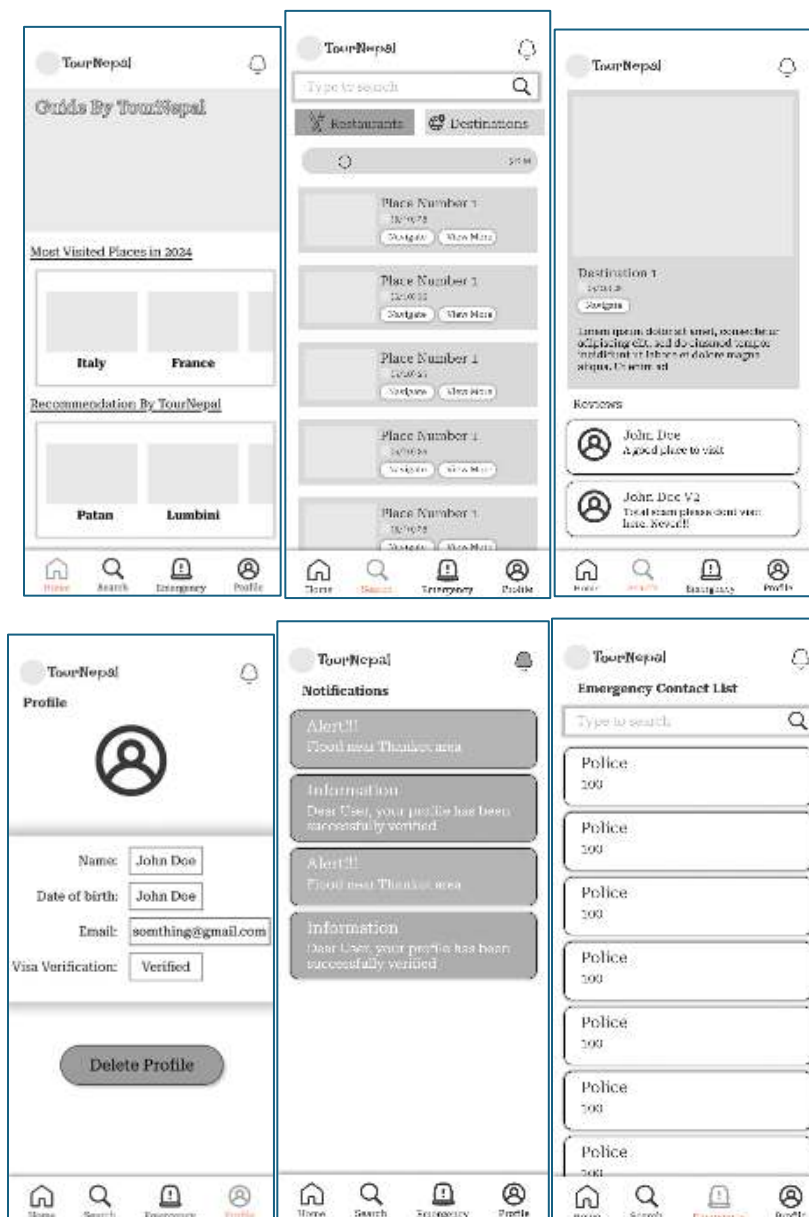


Figure 266: Mobile screen wireframes

III) Admin panel

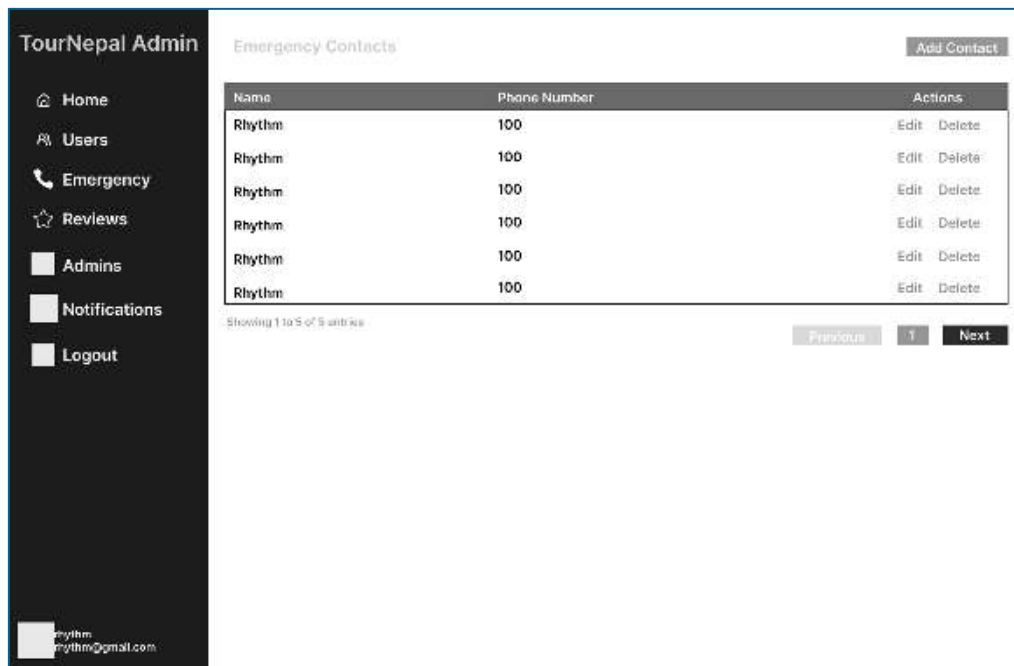


Figure 267: Emergency contacts wireframe (admin)

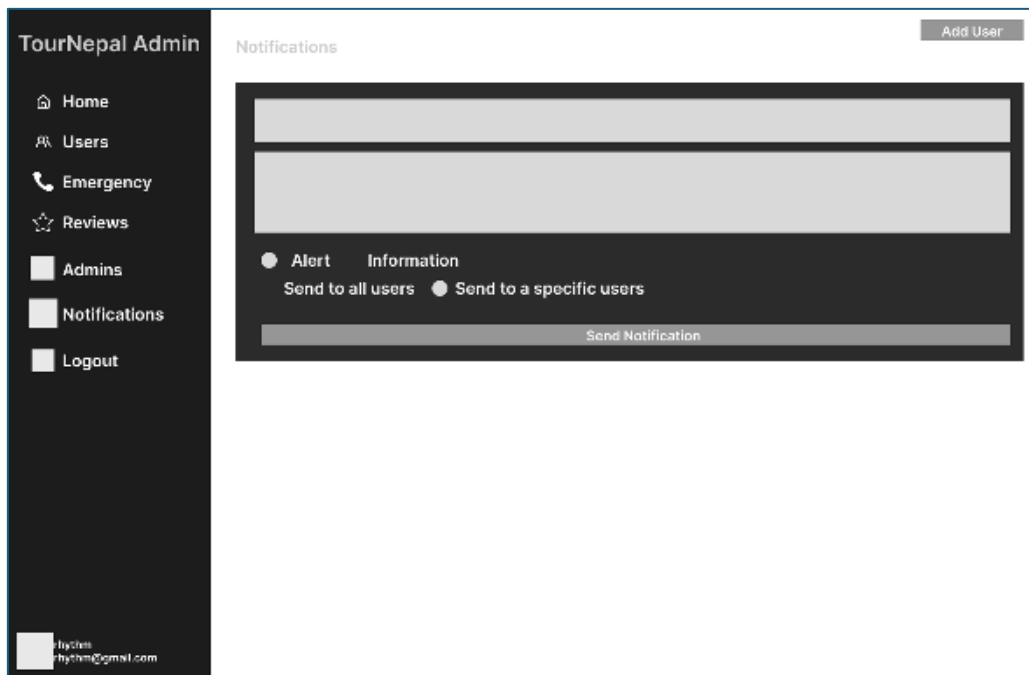


Figure 268: Notifications screen wireframe (admin)

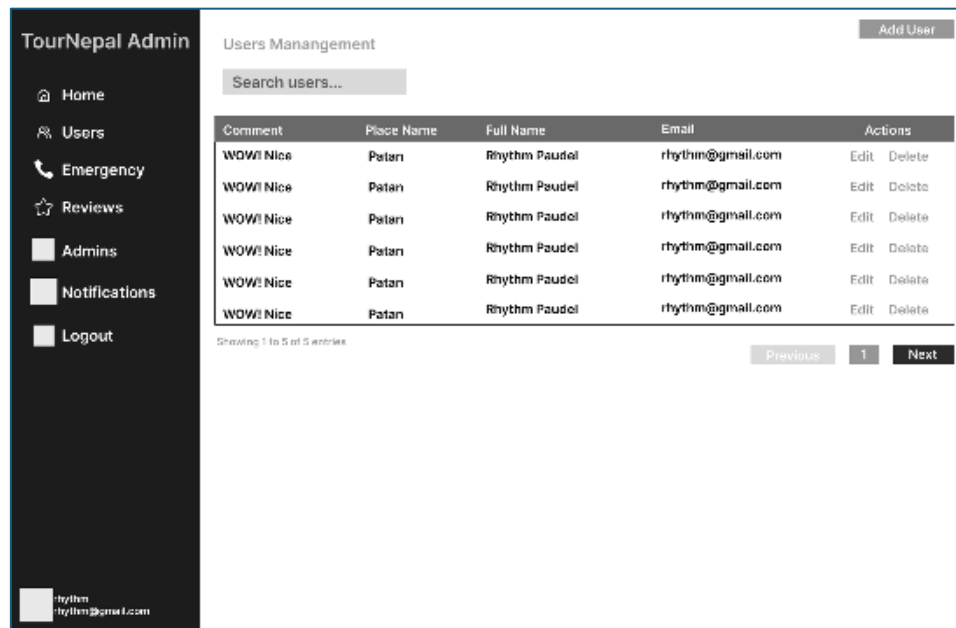


Figure 269: Review management screen wireframe (admin)

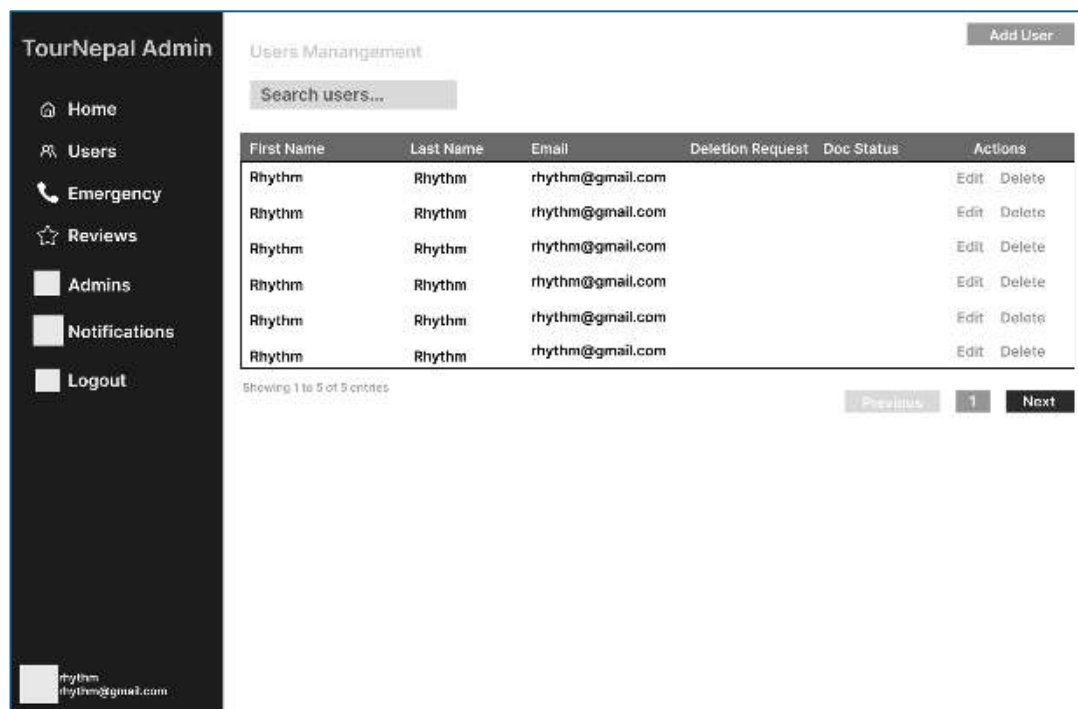


Figure 270: User management screen wireframe (admin)

7.9.9. User Intefrace (UI)

I) Mobile screens

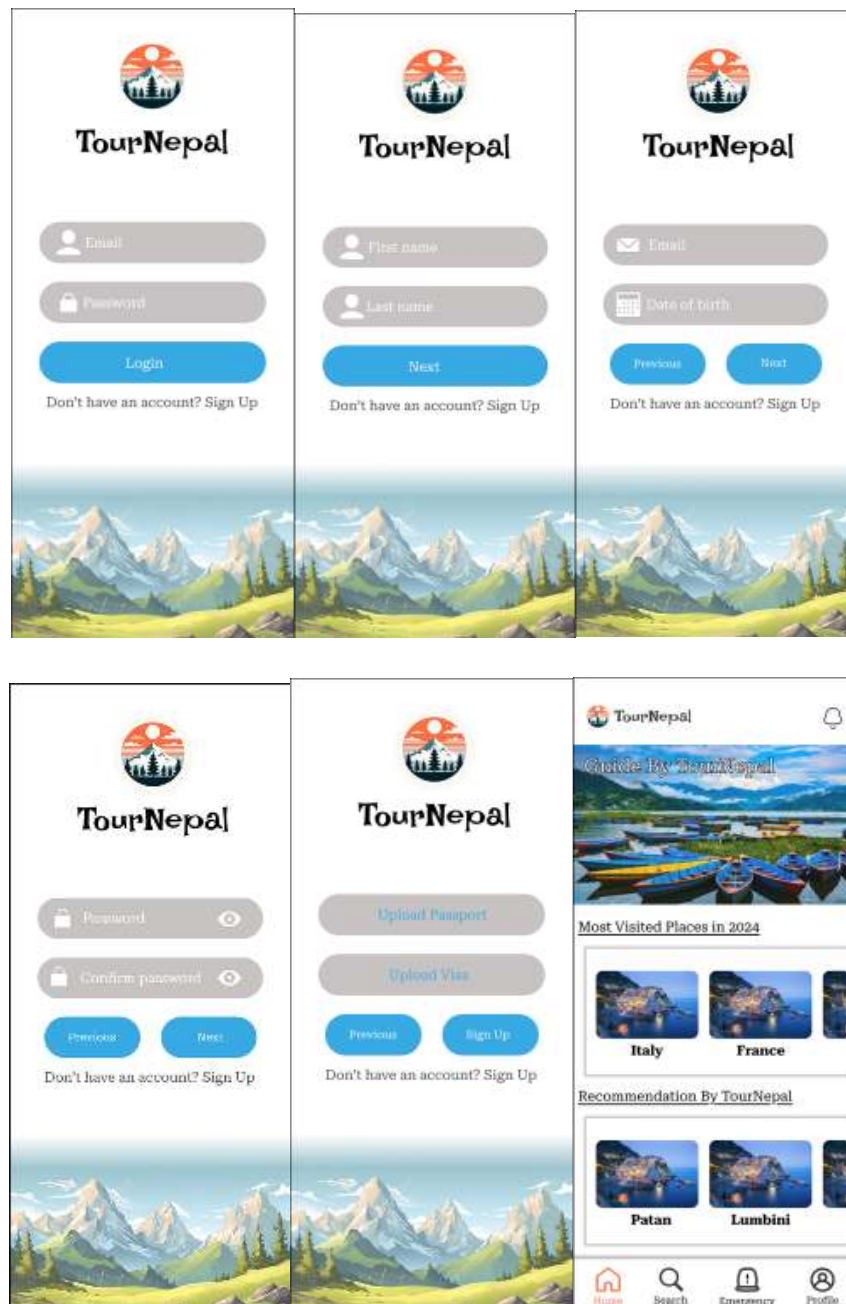


Figure 271: Registration, login and home screen

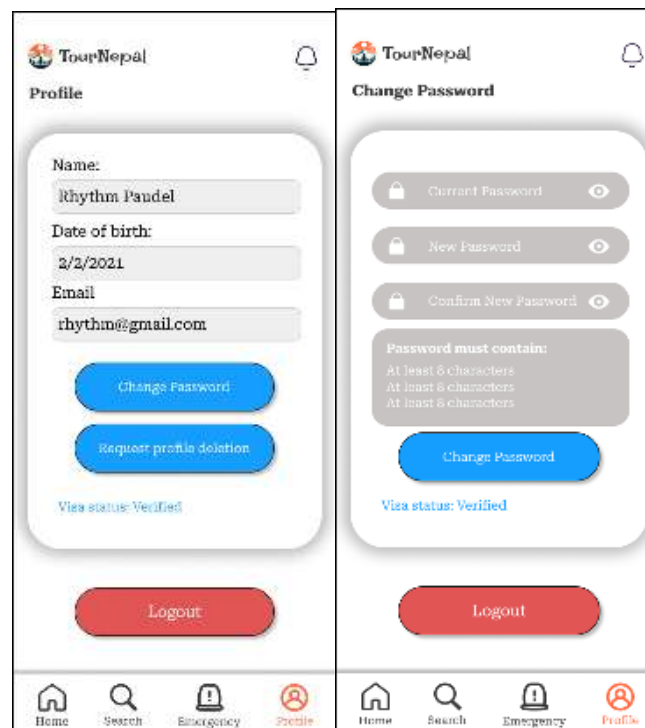


Figure 272: Profile screen

II) Web-app admin panel screen

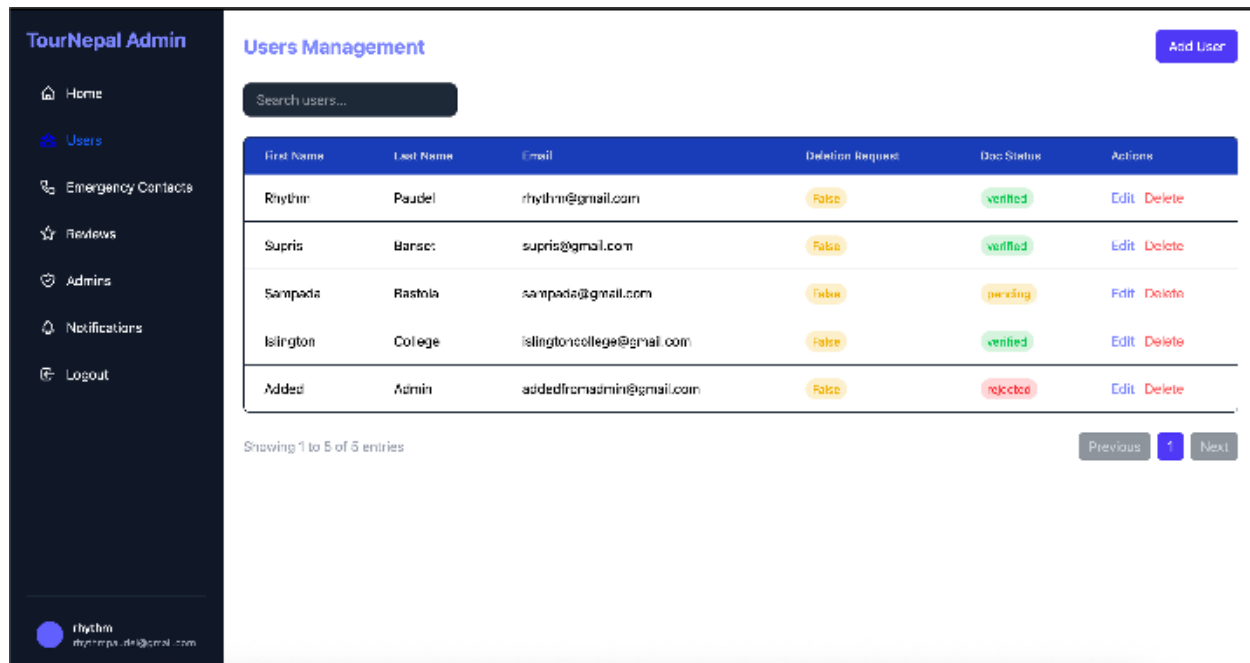


Figure 273: Users page (admin panel UI)

The screenshot displays the 'Personal Information' section of the user detail page. The left sidebar contains navigation links: Home, Users, Emergency Contacts, Reviews, Admins, Notifications, and Logout. The main content area shows the following details:

- First Name:** Supriya
- Last Name:** Banskot
- Email:** supriya@gmail.com
- Date of Birth:** 26/04/2025
- Nationality:** Bann
- Date of Entry:** 26/04/2025
- Indented Days of Stay:** 30

Below the personal information is the 'Document Verification' section, which includes two image uploads: 'Visa Copy' and 'Passport Copy'. Both images show a hand holding a document. The 'Document Status' is 'Verified'.

Figure 274: User detail (admin panel UI)

The screenshot displays the 'Emergency Contacts' section of the admin panel. The left sidebar is the same as in Figure 274. The main content area shows a table of emergency contacts with the following data:

Name	Phone Number	Actions
Police	100	Edit Delete
Ambulance	103	Edit Delete
A/T-36 Testing	11225344	Edit Delete

An 'Add Contact' button is located in the top right corner of the section.

Figure 275: Emergency contacts (admin panel UI)

TourNepal Admin

- Home
- Users
- Emergency Contacts
- Reviews**
- Admins
- Notifications
- Logout

Reviews Management

Search reviews...

Comment	Place Name	Full Name	Email	Actions
Hello world	Dina's Poolside Restaurant	Rhythm Paudel	rhythm@gmail.com	Delete
Idk something new	Local Kitchens	Rhythm Paudel	rhythm@gmail.com	Delete
Anko ma new try	Hibar McGass	Rhythm Paudel	rhythm@gmail.com	Delete
Yo ta Mexican ho	Mexican Restaurant	Supris Banset	supris@gmail.com	Delete
Orhh	Himalayan Food, Mart, Restaurant & Futsal Center	Rhythm Paudel	rhythm@gmail.com	Delete

Showing 1 to 5 of 10 entries

Previous 1 Next

rhythm
rhythm.paudel@gmail.com

Figure 276: Review management (admin panel UI)

TourNepal Admin

- Home
- Users
- Emergency Contacts
- Reviews
- Admins
- Notifications**
- Logout

Send Notifications

Notification Title

Notification Body

☒ Alert ☐ Information

☒ Send to All Users ☐ Send to Specific User

Send Notification

rhythm
rhythm.paudel@gmail.com

Figure 277: Notification screen (admin panel UI)

7.10. Appendix I: Code logic

7.10.1. Server setup logic

The following code snippet shows the basic logic on routes that have been set up for the backend server. All the routes after `verifyRefreshToken` require a verification check for any operations. All the routes before that can be accessed by anyone. Similarly, for posting a comment, another layer of security operation is performed before any actions are performed within the `/postComments` route.

```
33 app.use('/', require('./routes/root'));
34 app.use('/register', require('./routes/register'));
35 app.use('/login', require('./routes/login'));
36 app.use('/refresh', require('./routes/refresh'));
37 app.use('/logout', require('./routes/logout'));
38 app.use('/emergencycontacts', require('./routes/emergencyContacts'));
39 app.use('/location', require('./routes/api/location'));
40 app.use('/getReviews', require('./routes/api/getReviews'));
41 app.use('/send-notification', require('./routes/api/firebaseNotification'));
42 app.use('/admin', require('./routes/admin')); //handles all the admin related routes
43
44 //for verification of refresh token
45 app.use('/verifyJWT', require('./routes/verifyJWT'));
46
47 //using verification of user before letting them view screens
48 app.use(verifyRefreshToken) //this is named refreshToken, this is used for verification of access token
49 app.use('/userDetail', require('./routes/api/userDetail'));
50 app.use('/reUploadDocs', require('./routes/reuploadDocs'));
51 app.use(verifyUserVerification)
52 app.use('/postComments', require('./routes/postComments'));
53
```

Figure 278: Server setup (code snippet)

7.10.2. Persistent login logic

```
14  const AuthenticationProvider = ({children}) => {
91  useEffect(() => {
92
93      Windsurf: Refactor | Explain | Generate JSDoc | X
94      const checkToken = async () => {
95          setIsLoading(true);
96          try{
97              const tokens = await getToken();
98
99              if (tokens && tokens.encryptedToken && tokens.accessToken) {
100
101                  const response = await checkUserToken(tokens.encryptedToken); //waiting for response from the backend
102
103                  if (response?.status === 200) {
104                      let accessToken = decodedToken(tokens.accessToken);
105                      const details = await getUserDetail();
106                      if(details?.status===200){
107                          setCurrUser(details.data); //setting the user session if the token is validated
108
109
110                          setIsAuthenticated(true);
111                      }else{
112                          setIsAuthenticated(false);
113                      }
114
115                      await saveUserLocation();
116                  }
117                  //if the refresh token is expired then everything else is set to false and user has to re login
118                  else {
119                      setIsAuthenticated(false);
120                  }
121              } else {
122                  setIsLoading(false);
123                  setIsAuthenticated(false);
124              }
125          }
```

Figure 279: Persistent user login

7.10.3. Token storage

```
4 export const storeTokens = async (accessToken, encryptedToken) => {
5   try {
6     //for storing access token and encrypted token in dictionary
7     const authTokens = {accessToken, encryptedToken};
8     await Keychain.setGenericPassword('auth', JSON.stringify(authTokens), {
9       service: 'TourNepalTokens',
10    });
11  } catch (e) { ...
12  }
13 }
14 };
15
16 //for retrieving tokens in mobile device
17 Windsurf: Refactor | Explain | X
18 export const getToken = async () => {
19   try {
20     const authTokens = await Keychain.getGenericPassword({
21       service: 'TourNepalTokens',
22     });
23     if (authTokens) {
24       return JSON.parse(authTokens.password);
25     }
26     return "1st";
27  } catch (e) { ...
28  }
29 }
30 };
31
32 //for clearing tokens stored in mobile device
33 Windsurf: Refactor | Explain | X
34 export const clearToken = async () => {
35   try {
36     await Keychain.resetGenericPassword({service: 'TourNepalTokens'});
37  } catch (e) { ...
38  }
39 }
40 };
```

Figure 280: Token Storage code snippet

7.10.4. Review addition logic

```

<View style={styles.commentSection}>
  <TextInput
    style={[styles.commentInput, !verified && styles.disabledInput]}
    placeholder={verified ? "Add a comment" : "Document needs to be approved before posting a review"}
    value={comment}
    onChangeText={setComment}
    editable={verified}
    placeholderTextColor={!verified ? '#888' : '#ccc'}
  />

```

Figure 281: Review box disable code snippet

```

14  const DescriptionScreen = () => {
72    const handleComment = async () => {
73      let encryptedToken = await getToken();
74      Alert.alert(
75        'Add a review',
76        'Confirm your review before posting',
77        [
78          {
79            text: 'Cancel',
80            style: 'cancel',
81          },
82          {
83            text: 'Save',
84            onPress: async () => {
85              const response = await addComment(comment, place.id, place.name);
86              if (response?.status === 201) {
87                Alert.alert('Success', 'Comment was added successfully.');

```

Figure 282: Adding a comment (frontend)

```
Windsurf: Refactor | Explain | Generate JSDoc | X
const verifyUserVerification = async (req, res, next) => {
  try {
    const { email } = req;

    const userDB = await User.findOne({ email }).exec();

    if (!userDB) return res.sendStatus(404); //user not found

    if (userDB.verificationStatus !== "verified")
      return res
        .status(403)
        .json({ message: "Sorry the user is not verified" }); //forbidden to post comment

    req.User = userDB;
    next();
  } catch (e) {
    res.status(404).json({ message: e.message });
  }
};

module.exports = { verifyUserVerification };
```

Figure 283: User verification before posting a review code snippet

```
3   const postCommentsController = {
4     postComments: async (req, res) => {
12      if (!location || !commentBody || !name)
13        return res
14          .status(400)
15          .json({
16            message: "Location, and Comment Body are required.",
17          });
18
19      let existingReview = await Reviews.findOne({
20        "location.longitude": location.longitude,
21        "location.latitude": location.latitude,
22      });
23      try {
24
25
26        if (existingReview) {
27
28 >      existingReview.reviews.push({ ...
33      });
34
35      await existingReview.save();
36
37      commentID = existingReview.reviews.slice(-1)[0]._id;
38      text = existingReview.reviews.slice(-1)[0].text;
39      date = existingReview.reviews.slice(-1)[0].date;
40 >      review = { ...
47      };
48    } else {
49      const result = await Reviews.create({
50 >      location: { ...
53      },
```

Figure 284: Adding review code snippet

7.10.5. Notification sending logic

```
5  const Notifications = () => {
41  const handleSendNotification = async () => {
47      const notificationData = {
48          title: formState.title,
49          body: formState.body,
50          tokens: selectedUser,
51          messageType: formState.messageType
52      };
53
54      try {
55          const response = await sendNotifications(notificationData, user.accessToken);
56          if (response?.status === 200) {
57              alert('Notification sent successfully!');
58          } else if (response?.status === 403) {
59              const newToken = await getAccessToken();
60              if (newToken?.status === 200) {
61                  setUser(newToken.data);
62                  const retryResponse = await sendNotifications(notificationData, newToken.data.accessToken);
63                  if (retryResponse?.status === 200) {
64                      alert('Notification sent successfully!');
65                  }
66              } else if (newToken.status === 403) {
67                  setUser(null);
68              }
69          } else {
70              alert("Some error occurred");
71          }
72      } catch (error) {
73          alert("Error sending notification: " + error.message);
74      }
75
76      setFormState({ title: '', body: '', recipientType: 'all', specificUser: '', messageType: 'Information' });
77  };
78  }
```

Figure 285: Notification handling admin panel code snippet

```
11 const sendNotifications = async(req, res) => {
15   // Check if tokens is an array and if title and body are provided
16   if(!tokens || !Array.isArray(tokens) || tokens.length === 0 || !title || !body) {
17     return res.status(400).json({message:"Invalid request: tokens array, title, and body are required"});
18   }
19
20
21   // For multiple devices
22   let successCount = 0;
23   let failureCount = 0;
24   const failures = [];
25
26   for (const user of tokens) {
27     const { token, email } = user;
28     try {
29 >     const message = { -
35       };
36
37     const response = await admin.messaging().send(message);
38     const notificationMessage = `${title}: ${body}`;
39     const notificationObject = {
40       title: title,
41       message: body,
42       messagetype: messagetype
43     };
44     console.log("Sent successfully to", token, response);
45     successCount++;
46     await User.findOneAndUpdate(
47       {email},
48       { $push: { notificationList: notificationObject } },
49       { new: true }
50     );
```

Figure 286: Notification handling code snippet

7.10.6. Nearby place code logic

```

8  const getNearbyDestinations = async (latitude, longitude, radius, destinationType) => {
9
10   let lat = parseFloat(latitude)
11   let lon = parseFloat(longitude)
12   let rad = parseFloat(radius)
13   let destinations = `https://maps.googleapis.com/maps/api/place/nearbysearch/json?keyword=s(destinationType)&location=${lat},${lon}&radius=s(rad)&t
14
15
16
17   try {
18     if (!isNaN(rad)) {
19
20       destinations = `https://maps.googleapis.com/maps/api/place/textsearch/json?query=${radius}+in+Nepal&region=np&key=${process.env.GOOGLE_MAPS_API}
21     }
22
23     const response = await axios.get(destinations);
24     if (response.data.results) {
25       const placesDict = {};
26
27       for (const place of response.data.results) {
28
29         //response.data.results.forEach this for each wont work for asynchronous method
30         let latitude = place.geometry.location.lat;
31         let longitude = place.geometry.location.lng;
32
33         // Only add places that have a photo reference
34         let photo_aaa =
35           place.photos && place.photos.length ? await getPhoto(place) : false;
36
37         placesDict.push({
38           name: place.name,
39           //description: descriptionOfPlace,
40           location: { latitude: latitude, longitude: longitude },
41           rating: place.rating,
42           photoRef: photo_aaa,
43         });
44       }
45
46       return placesDict;
47     }
48   }
49 }

```

Figure 287: Nearby place fetching backend code snippet

```

35  useEffect(() => {
36      Windsurf: Refactor | Explain | Generate JSDoc | X
37      const getPlaces = async () => {
38
39          setIsLoading(true);
40          const places = await nearbyPlaces(radius * 1000, currentSelection, currentLocation);
41
42          setIsLoading(false);
43          setAllPlaces(places);
44          setFilteredPlaces(places);
45      };
46
47      if (locationPermission && currentSelection !== 'all') {
48          getPlaces();
49      } else if (!locationPermission) {
50          Alert.alert(
51              'Location Permission Required',
52              'Please enable location permission to get nearby places',
53              [
54                  {
55                      text: 'Open Settings',
56                      onPress: () => Linking.sendIntent('android.settings.LOCATION_SOURCE_SETTINGS')
57                  },
58                  {
59                      text: 'Cancel',
60                      style: 'cancel'
61                  }
62              ]
63          );
64      }
65      ], [radius, currentSelection, currentLocation]);

```

Figure 288: Nearby place search mobile app code snippet

```

const handleSearch = async () => {
    if (currentSelection === 'all' && searchInput.trim() && locationPermission) {
        setIsLoading(true);
        try {

            const places = await nearbyPlaces(searchInput, currentSelection, currentLocation);

            setAllPlaces(places);
            setFilteredPlaces(places);
        } catch (error) {
            console.error(error);
        } finally {
            setIsLoading(false);
        }
    }
}

```

Figure 289: Selective search code snippet

7.10.7. Description generation

```
useEffect(()=>{
  const descriptionOfPlace = async () => {
    setIsLoading(true)
    const response = await getDescription(place.name);
    if (response?.status === 200) {
      setDescription(response.data.description);
    }else{
      setDescription("Description for this place is not available");
    }
    setIsLoading(false)
  }
  descriptionOfPlace();
}, [])
```

Figure 290: Description generation mobile app code snippet

```
const getDescription = async (placeName) => {
  const apiKey = process.env.GOOGLE_GEMINI_API_KEY;
  const genAI = new GoogleGenerativeAI(apiKey);

  const model = genAI.getGenerativeModel({
    model: "gemini-2.0-pro-exp-02-05",
  });

  const generationConfig = {
    temperature: 0.7,
    topP: 0.95,
    topK: 64,
    maxOutputTokens: 8192,
    responseMimeType: "text/plain",
  };

  Windsurf: Refactor | Explain | Generate JSDoc | X
> async function run() { ...
  }

  return await run();
};

module.exports = {returnDescription}
```

Figure 291: Description generation Gemini code snippet

7.11. Appendix I: Software Requirement Specification (SRS)

7.11.1. Introduction

The TourNepal application is designed for tourists in Nepal to boost their experience while their stay. The application will provide a list of places like restaurants, and destinations nearby the users making them easier and tourist friendly interface to interact unlike other place suggestion applications. The application also provide user with list of emergency contacts in case of emergency.

The system will also include an admin panel which is web based, that lets admin to handle the activity of the users like validating user's visa or deleting the user if needed. Overall, the system aims to deliver a proper working software that will help the tourist navigation more convenient, and more tourist oriented.

7.11.2. Intended Audiences and Reading Suggestions

The SRS document is intended for a group of people working together to build this project.

The team contains:

- I) Development Team: For understanding the functional and non-functional requirements of the application
- II) Project Managers: For quality control purpose and making sure the development process is aligned with the goals.
- III) QA: For testing various cases with proper understanding of the functions.
- IV) Stakeholders: Likewise, project managers, for monitoring the requirements alignment with goals.

The SRS could be read in a following way so that the readers could understand the documentation in a detailed way.

- I) Readers can start with summary for a brief understanding of the system.
- II) Referring to the functional requirement for understanding the flow of the system.

- III) After the functional requirement is cleared out, reader should understand about non-functional requirement like understanding performance and security of the application.

7.11.3. Project Scope

In this section of this documentation, the features and deliverables taking the scope into considerations will be explained thoroughly. The TourNepal will focus on making the tourist visit enhanced. This will be a mobile-based application for the user and web-based application for admin. This application is planned to increase the satisfaction level among tourists by making the application a review-based destinations places integrating with map.

The project will include:

- I) A mobile application with main functionalities like, navigating places, feedback and emergency contact accessibility.
- II) Google maps integration for fetching places nearby.
- III) An admin panel for managing users and their validations.

Certain things will be excluded such as:

- I) Offline map functionalities will not be included.
- II) Restaurants reservation system will also be excluded.
- III) A description about restaurants showing in the nearby restaurants.

Limitations:

TourNepal will be entirely dependent on internet, meaning if the internet is not available none of the application would work as expected.

7.11.4. System features

- **SF1: Authentication**
- **Description:** This authentication feature make sure only registered user can access the application in a secured way. Authentication feature also makes sure only verified user can perform certain access like leaving feedback on destinations list.
- **SF2: Exploration of destinations**
- **Description:** This feature will let the user let go through a list of destinations nearby also with description for popular destinations.
- **SF3: Nearby Places**
- **Description:** This feature will use google map Api to fetch places which then forwards it to the previous feature of exploration of destinations. User can view and navigate to these places directly from the application.
- **SF4: User Feedback**
- **Description:** This feature will let the user/tourist leave feedback for the destinations. Here only verified user can leave a review but everyone can view.
- **SF5: Emergency Contact Feature**
- **Description:** This feature will let user access a list of emergency contact numbers ensuring the safety of the user.
- **SF6: Notifications**

- **Description:** This feature will inform the user about ongoing events or alert user for unexpected circumstances like natural disaster via firebase.

7.11.5. User class and characteristics

The following are the classes of types of users and roles and operations they are entitled to do.

UC1: Admin

UC1.1	Can log into the system.
UC1.2	Can perform user CRUD operations
UC1.3	Can validate user documents, like visa stamp
UC1.4	Can approve or reject, profile delete request by the user.
UC1.5	Can manage emergency contact list.

Table 97: User class (admin)

UC2: Tourist (User)

UC2.1	Can register and login into the system.
UC2.2	Can view various destinations like restaurants or famous tourist attractions.
UC2.3	Can view nearby places that uses Google Map API
UC2.4	Can leave a review for a place (restaurants or destinations).
UC2.5	Can access phone numbers of emergency bodies
UC2.6	Can receive notification for events, updates or emergencies.

Table 98: User class (user)

7.11.6. Operating Environment

- **OE1:** The application is developed primarily for mobile and some functionality on web application.
- **OE2:** This application will work on android devices
- **OE3:** The application will handle the screen sizes across devices and be responsive in webapp as well.
- **OE4:** The entire backend operation will be handled with Node.js framework and MongoDB as a cloud database provider.
- **OE5:** The application will be integrated with external APIs such as google map for destinations fetching and firebase for notifications.

7.11.7. Design and Implementation Constraint

- **CO1: Data Dependency:** The system is dependent on location's information, user information and emergency contact address from the database.
- **CO2: Scalability:** The application should be able to handle large number of user's data and increasing number of destinations.
- **CO3: Authentication:** User must be assigned with access token and refresh token for security. Only the verified user should be allowed to leave a feedback/review on a place.
- **CO4: Resource Constraints:** System may face some problem due to limited computational power. The application is fully dependent on internet so, internet access is needed for the functionalities.

7.11.8. Assumption and dependencies

Assumptions

- **AS1:** User will have a consistent internet when using the application.
- **AS2:** Tourist will provide a visa, and a passport copy for verification purpose.
- **AS3.** Admin will be active regarding visa validation process, and user's data manipulation process.
- **AS4:** External APIs that includes google map Api, or firebase will be operational

Dependencies

- **DP1:** Application is dependent on google maps Api for location fetch.
- **DP2:** Application is dependent on firebase by google for pushing notification.
- **DP3:** MongoDB will fetch the almost entire data to the backend that mainly user's information details.

7.11.9. Functional Requirements

Rrg.ID	Requirement Desc	Priority	Complexity
FR01	The system shall allow a user to register with credentials like personal details, visa stamp and passport copy.	High	Medium
	System Requirements		
	The system will perform the validation process for verifying user's email.		
	The system shall encrypt the sensitive fields like password.		
	The system shall generate a secure access and refresh token.		
	The system shall inform the user about successful or error during registration.		
	The system shall look for the duplicate entries during the registration.		
FR02	The system shall limit the functionality on the app, letting non-verified user not have full access to the functions.	High	Medium
	System Requirements		
	The system shall store the user's detail securely in database.		
	The system shall verify the documents provided in order to give feedback in destinations.		

		The system shall alert or inform the user if their visa verification was denied.												
		The system shall let the user to submit their documents again.												
FR03	The system will be able to list the destinations that may be either tourist attractions or restaurants. <table><tr><td colspan="2">System Requirements</td></tr><tr><td></td><td>The system will fetch places reviews if any from the database.</td></tr><tr><td></td><td>The system shall let the user to view reviews, and description of place (if available).</td></tr><tr><td></td><td>The system shall let the user know the price of places for tourist attraction if applicable.</td></tr><tr><td></td><td>The system shall let the user search through places near them within the selected radius.</td></tr></table>		System Requirements			The system will fetch places reviews if any from the database.		The system shall let the user to view reviews, and description of place (if available).		The system shall let the user know the price of places for tourist attraction if applicable.		The system shall let the user search through places near them within the selected radius.	High	Medium
System Requirements														
	The system will fetch places reviews if any from the database.													
	The system shall let the user to view reviews, and description of place (if available).													
	The system shall let the user know the price of places for tourist attraction if applicable.													
	The system shall let the user search through places near them within the selected radius.													
FR04	This feature will fetch destinations dynamically using Google Maps API. <table><tr><td colspan="2">System Requirements</td></tr><tr><td></td><td>The system shall access the user's current location.</td></tr><tr><td></td><td>The system shall send a list of places nearby the user.</td></tr><tr><td></td><td>The system shall also display a navigation button to navigate via google maps.</td></tr></table>		System Requirements			The system shall access the user's current location.		The system shall send a list of places nearby the user.		The system shall also display a navigation button to navigate via google maps.	High	High		
System Requirements														
	The system shall access the user's current location.													
	The system shall send a list of places nearby the user.													
	The system shall also display a navigation button to navigate via google maps.													

		The system shall secure the user's location during the process.		
FR05	This feature shall let the user access the emergency contact bodies in a list view.		High	Low
	System Requirements			
		The system shall store the list of emergency contact numbers in database.		
		The system shall let the user view the emergency contacts within the app.		
		The system shall display the emergency contact in useful ranking order.		
		The system shall update the emergency contacts number by admin in admin panel.		
FR06	The system shall notify the users via notification section regarding updates of their profile, emergency alerts, or general information		Medium	Medium
	System Requirements			
		The system shall push notifications if there are any emergency updates like flood or landslides.		
		The system shall notify the user if their visa has been validated or not.		

		The system shall inform user if user's profile deletion has been declined or approved.		
--	--	--	--	--

Table 99: Functional requirements

7.11.10. External Interfaces Requirements

User Interfaces

The system will provide a friendly user interface that will be responsive to the mobile screens. There also will be an admin panel which will be used to manage users, and emergency contacts. The mobile app will have support of login, register. It will consist of a home screen that will help user to navigate to different places. It will also have a separate section that will list down all the emergency contacts. In each destination, there will be a feedback or review section. A profile section that will consists of user details will also be available. Additionally, in admin panel the main section will be divided into two categories, one being user and other being emergency contacts. Both sections in admin panel will support editing feature.

Hardware Interfaces

- The users of this application will need to have a smartphone having a working android OS.
- Admin should have a desktop or laptop for the compatibility of this system.

Software Interfaces

- Google Maps api for location wise data fetching
- Firebase for pushing notifications to the user
- Node.js, Express.js for handling backend operations
- MongoDB for storing data and retrieving when in need.

Communication Interfaces

- The system will use HTTP/S for making a communication secured between frontend and backend data sending process.

- The system will also need a stable connection of internet for data retrieval from database or third-party API.

7.11.11. Non-functional Requirements

Req.ID	Requirement Description	Priority	Complexity
PR.01	The application should be able to operate efficiently under normal workload	Should	Medium
PR.02	The application should not be requiring running tasks that are not currently in use.	Should	Medium
PR.03	The application should make sure that each screen state should be loaded within 5 seconds	Should	Medium
PR.04	The application should be responsive to different screen sizes	Should	Medium
PR.05	The backend should be able to handle up	Must	High

	to 100 users at once in the beginning.		
--	--	--	--

Table 100: Non-functional Requirements

7.11.12. Safety Requirements

Req.ID	Requirement Description	Priority	Complexity
SR.01	The login credentials should be properly hashed while saving it in database	Must	High
SR.02	The application should not let normal users access admin functionality	Must	High
SR.03	The connection between the application server should be secured and well encrypted if the data contains sensitive info.	Must	High

Table 101: Safety Requirements

7.12. Appendix J: Screenshot of the system

I) Login screen



Figure 292: Login screen (mobile)

II) Signup screen

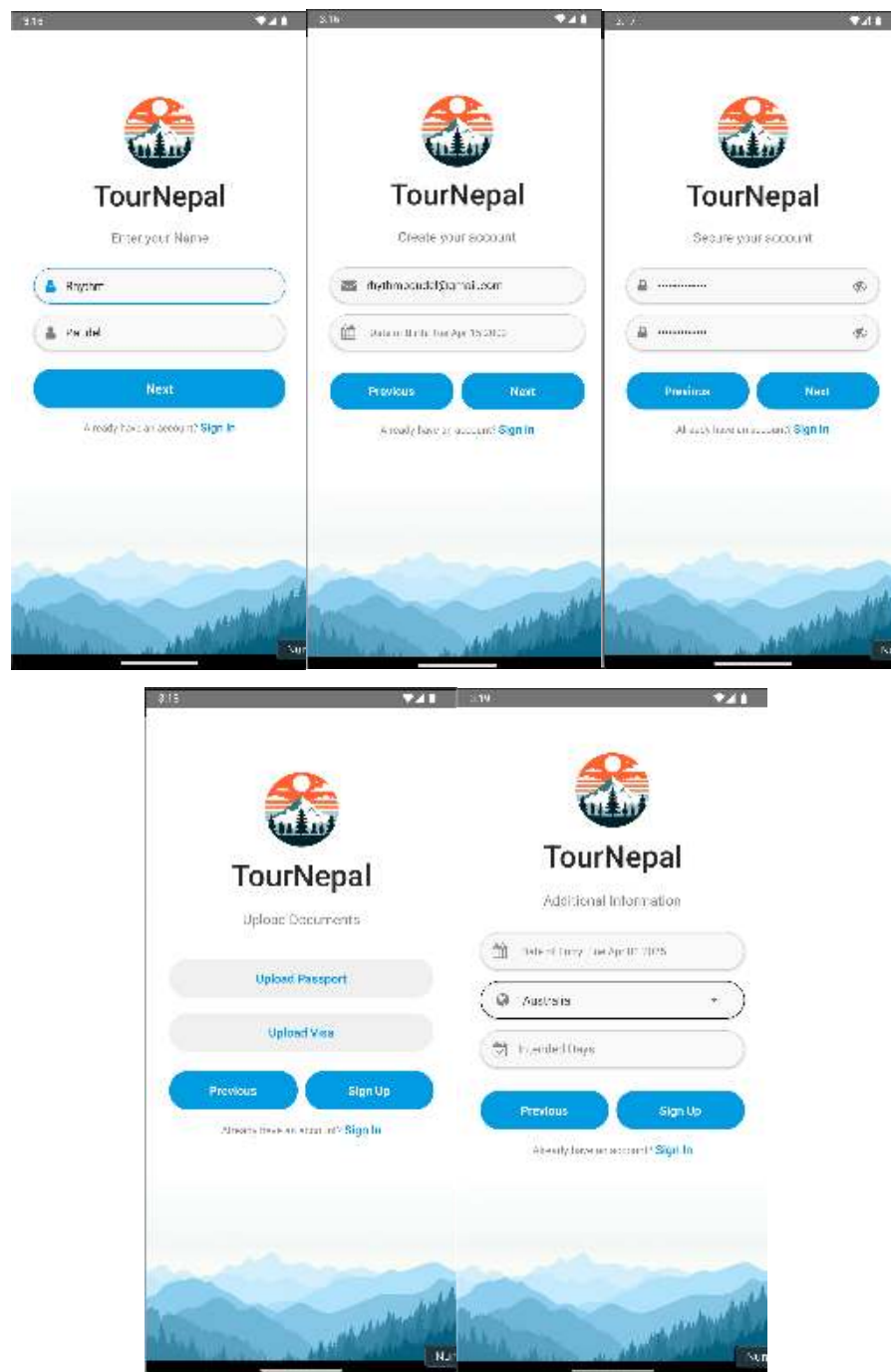


Figure 293: Signup screen (mobile)

III) Main screens

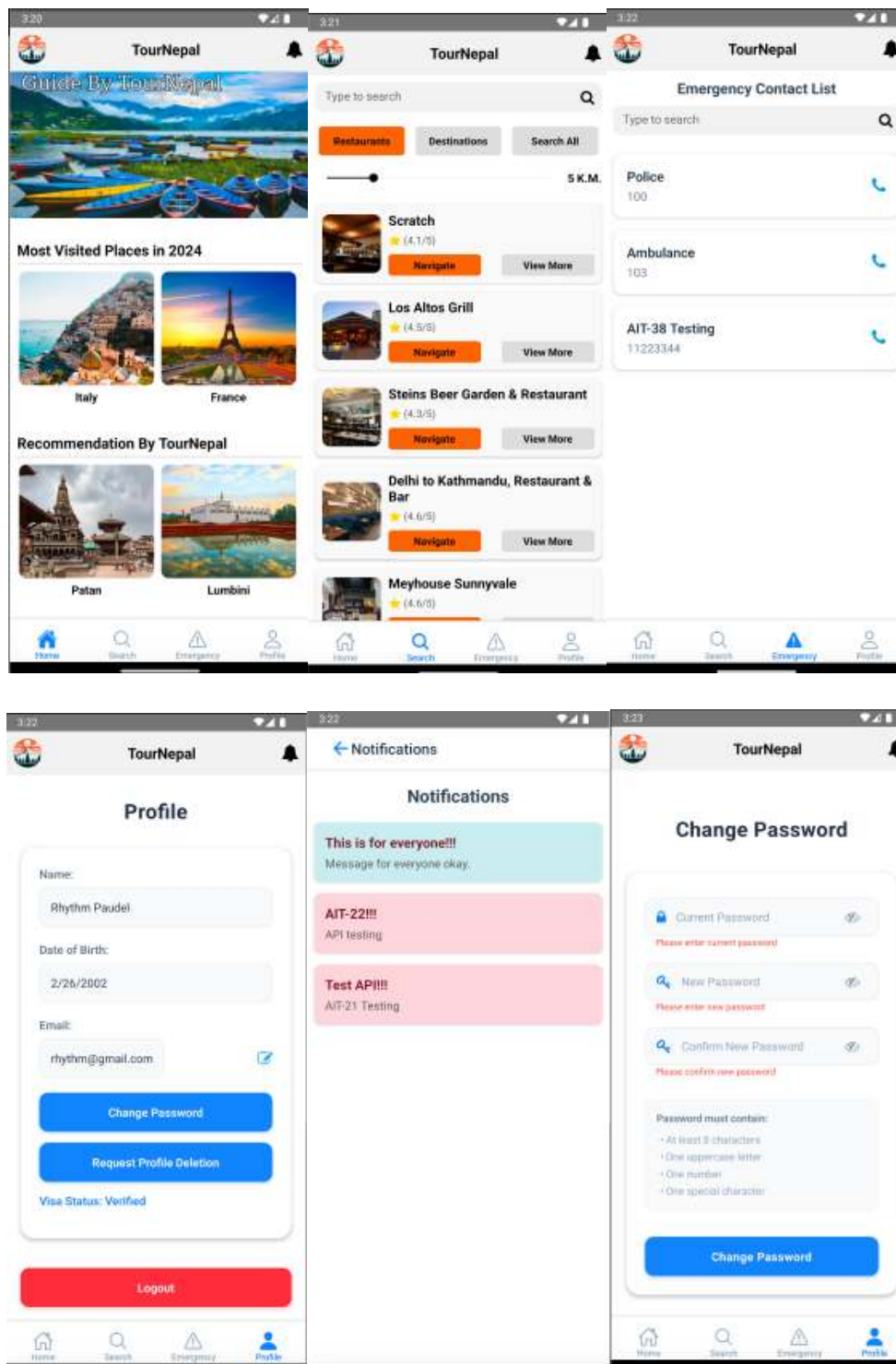


Figure 294: Main screens (mobile)

IV) Admin Panel

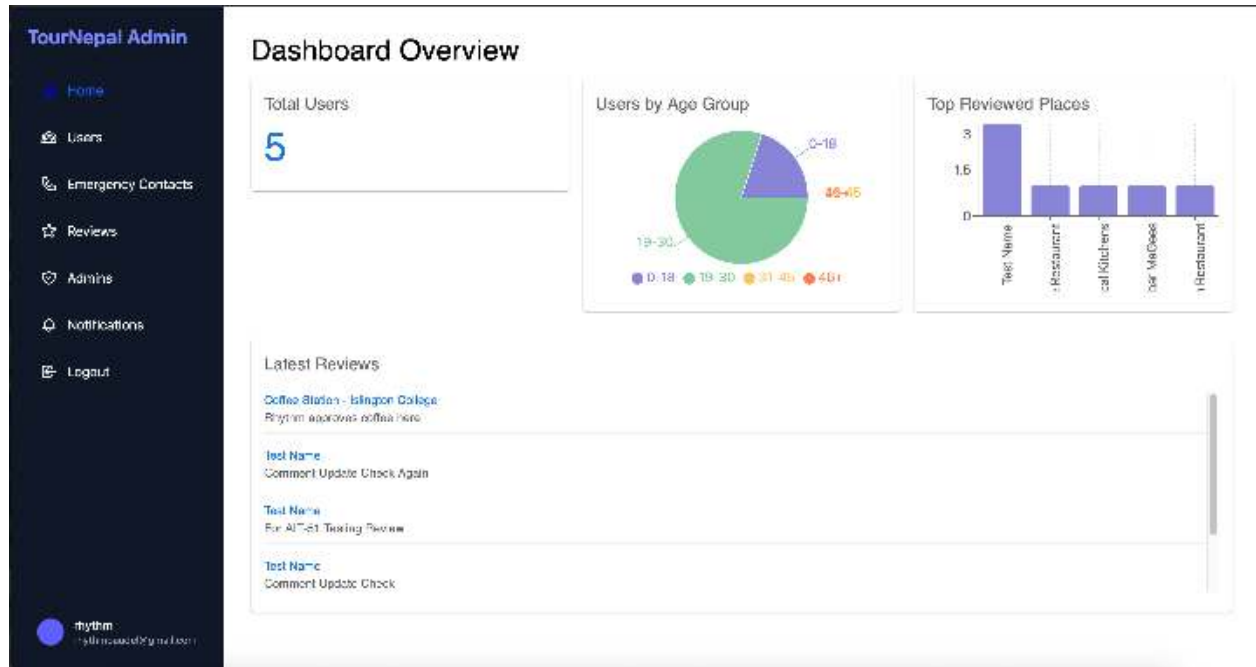


Figure 295: Home screen (admin)

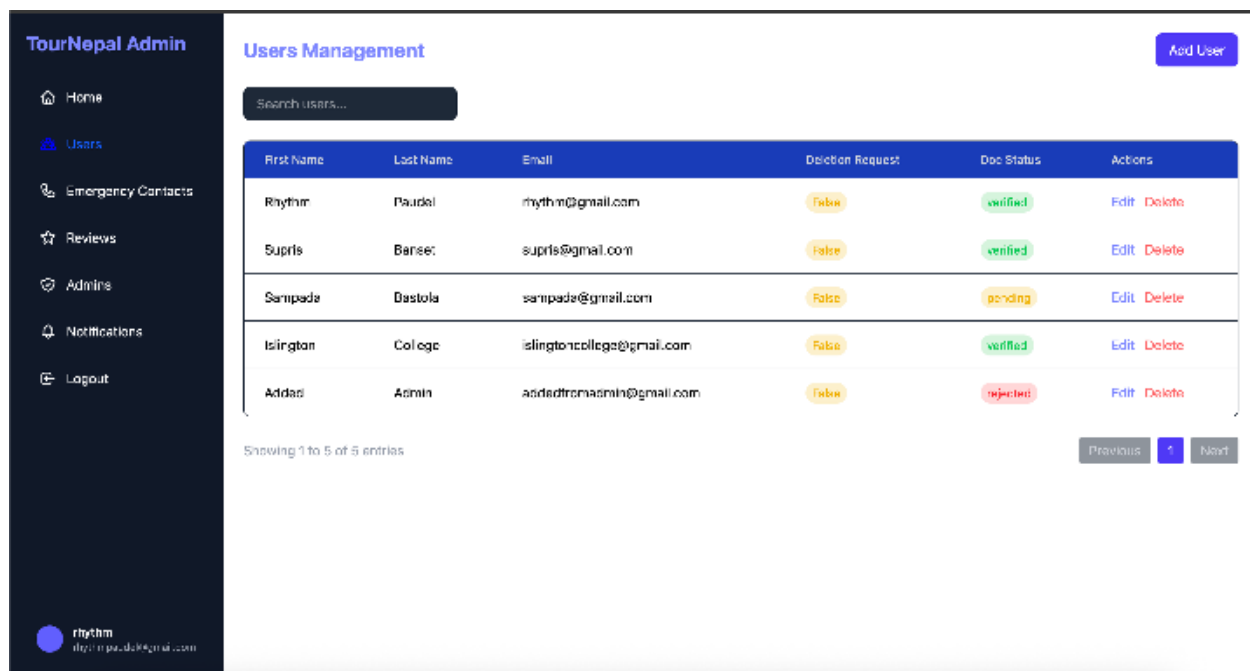


Figure 296: Users Management page (admin)

TourNepal Admin

- Home
- Users**
- Emergency Contacts
- Reviews
- Admins
- Notifications
- Logout

Edit User [Back to Users](#)

Personal Information

First Name: Last Name:

Email: Date of Birth:

Nationality: Date of Entry:

Intended Days of Stay:

Document Verification

Visa Copy:  Passport Copy: 

Document Status:

[Save Changes](#)

Figure 297: Edit user page (admin)

TourNepal Admin

- Home
- Users**
- Emergency Contacts
- Reviews
- Admins
- Notifications
- Logout

Add User [Back to Users](#)

Email:

First Name:

Last Name:

Date of Birth:

Password:

Confirm Password:

Date of Entry:

Nationality:

Intended Days of Stay:

[Add User](#)

Figure 298: Add user page (admin)

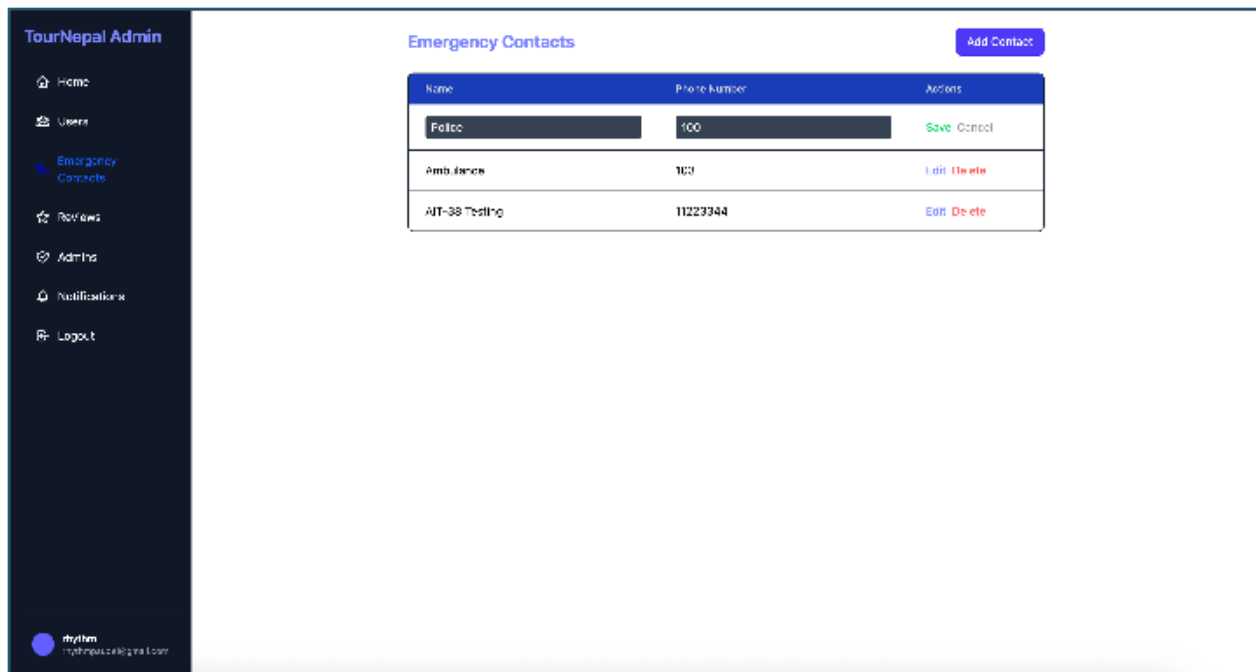


Figure 299: Emergency contacts page (admin)

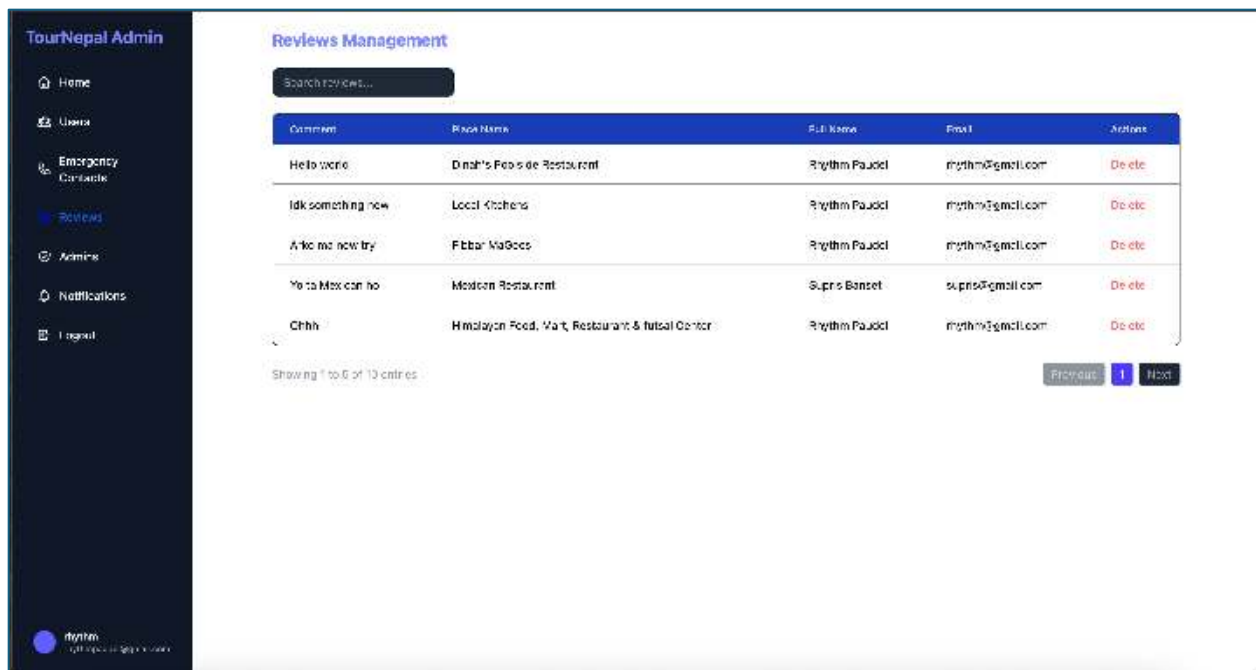


Figure 300: Review management page (admin)

TourNepal Admin

- Home
- Users
- Emergency Contacts
- Reviews
- Admins
- Notifications**
- Logout

Send Notifications

Notification Title

Notification Body

☐ Alert ☒ Information

☒ Send to All Users ☐ Send to Specific User

Send Notification

rhythm studio | info@rhythmstudio.com

Figure 301: Notification handling page (admin)

7.13. Appendix K: Client Approval Letter

Ambika Gyawali
Triple R Tours and Travel
Mahalaxmi-1, Imadol, Lalitpur
December 21, 2024

Rhythm Paudel
TourNepal
Islington College
Kamalpokhari
Kathmandu

Subject: Permission Granted for the role of Client.

Dear Rhythm Paudel:

I have reviewed your request for the role of client, and I am pleased to inform you that I have agreed to be your client for the project of the application. Taking reference from our recent verbal meeting I am willing to provide the requirements and the necessary information that I need in the (FYP Project) application.

If you have any questions, or if I can be of further service to you, please call me or arrange a meeting for which we can discuss the issues and present the progress of the application development till date.

Sincerely,



Ambika Gyawali

Rhythm Paudel
TourNepal

Signature

Table 102: Client approval letter

7.14. Appendix L: Development links

7.13.1. Trello board Link

→

<https://trello.com/invite/b/676d5a03a9715a276e956a59/ATTI8643ad38ea26590d503d3a4e72e80a74875B8FEE/tournepal-fyp>

7.13.2. GitHub repository link

→ <https://github.com/rhythm-paudel/finalyearproject.git>