



**islington college**  
(इस्लिंग्टन कलेज)

**Module Code & Module Title**

**CS6P05NI Final Year Project**

**Assessment Weightage & Type**

**40% Final Year Report**

**Semester**

**2023 Autumn**

**PROJECT TITLE: KarKhana**

**Student Name: Nishan Shrestha**

**London Met ID: 21049546**

**College ID: np01cp4a210202**

**Internal Supervisor: Lekhnath Katuwal**

**External Supervisor: Shubhankar Sharma**

**Assignment Due Date: April 24, 2024**

**Assignment Submission Date: April 23, 2024**

**Word Count (Where Required): 8242**

*I confirm that I understand my coursework needs to be submitted online via Google 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.*

## Table of Contents

CHAPTER1: INTRODUCTION.....	1
1.1 INTRODUCTION TO THE TOPIC .....	1
Benefits for Users.....	1
Benefits for multi-vendors .....	1
1.2 CURRENT SCENARIO .....	2
1.2.1 CURRENT SCENARIO BASED ON RESEARCH.....	2
1.2.2 CURRENT SCENARIO BASED ON SURVEY.....	2
1.3 PROBLEM DOMAIN .....	2
1.3.1 PROBLEM DOMAIN LISTED FROM RESEARCH .....	2
1.3.2 PROBLEM DOMAIN LISTED FROM SURVEY.....	2
1.4 PROJECT AS A SOLUTION .....	3
1.5 AIMS AND OBJECTIVES .....	3
1.5.1 AIMS .....	3
1.5.2 OBJECTIVES.....	4
1.6 STRUCTURE OF THE REPORT.....	4
1.6.1 BACKGROUND .....	4
1.6.2 DEVELOPMENT .....	4
1.6.3 TESTING AND ANALYSIS OF PROJECT .....	5
1.6.4 CONCLUSION .....	5
CHAPTER 2: BACKGROUND .....	6
2.1 ABOUT THE END USERS .....	6
2.2 UNDERSTANDING OF THE PROJECT.....	6
2.2.1 OVERVIEW OF THE SYSTEM.....	6
2.2.3 TECHNOLOGY USED .....	7

2.3 FUNCTIONS AND FEATURES .....	11
2.4 REVIEW OF SIMILAR PROJECTS .....	12
2.4.1 SIMILAR PROJECTS.....	12
2.4.2 COMPARISON OF SIMILAR PROJECT .....	15
2.5 CRITICAL EVALUATION.....	16
CHAPTER 3: DEVELOPMENT .....	17
3.1 CONSIDERED METHOLOGIES.....	17
Waterfall Methodology .....	17
Scrum Methodology .....	18
Evolutionary Prototyping .....	19
3.2 SELECTED METHODOLOGIES .....	20
3.3 PHASES OF METHODOLOGY .....	22
3.3.1 Phases of RUP Methodology in Table .....	23
3.4 SURVEY RESULTS .....	24
3.4.1 Pre-Survey Result .....	24
3.4.1.1 Pre-Survey Analysis .....	29
3.4.2 Post Survey Result.....	30
3.4.2.1 Post Survey Analysis .....	34
3.5 Requirement Analysis.....	35
3.5.1 FUNCTIONAL REQUIREMENTS .....	35
3.5.1.1 REGISTRATION.....	35
3.5.1.2 Login.....	36
3.5.1.3 Add Deal to Cart .....	37
3.5.1.4 Buy Coupons .....	38
3.5.1.5 User Feedback .....	39

3.5.1.6 Payment .....	40
3.5.1.7 Search .....	41
3.5.1.8 Coupon Status.....	42
3.5.1.9 Add Deals .....	43
3.5.2 Non-Functional Requirements.....	44
3.5.2.1 Performance Requirements.....	44
3.5.2.2 Safety Requirements .....	45
3.6 DESIGN.....	46
3.6.1 Logo of the Application.....	46
3.6.2 System Design.....	47
3.6.2.1 Use Case Diagram .....	47
3.6.2.2 ER-Diagram.....	48
3.6.2.3 Activity Diagram.....	49
3.6.2.4 Sequence Diagram .....	50
3.6.2.5 Flowchart.....	51
3.6.2.6 System Architecture Diagram .....	52
3.6.2.7 System Overview Diagram .....	53
3.6.3 Wireframe.....	54
3.6.3.1 Customer.....	54
I. Welcome Page .....	54
II. Customer Login Page.....	55
III. Customer Registration Page .....	56
IV. Customer Homepage .....	57
3.6.3.2 Vendor .....	58
V. Vendor Login Page.....	58

VI.	Vendor Registration Page .....	59
VII.	Vendor Homepage .....	60
3.6.4	UI Design .....	61
3.6.4.1	Customer .....	61
I.	Welcome Page .....	61
II.	Customer Login Page.....	62
III.	Customer Registration Page .....	63
IV.	Customer Homepage .....	64
3.6.4.2	Vendor Pages .....	65
I.	Vendor Login Page.....	65
II.	Vendor Registration Page .....	66
III.	Vendor Homepage .....	67
3.6.5	Feature Design.....	68
3.6.5.1	Sequence Diagram .....	68
3.6.5.1.1	Sequence Diagram for Login .....	68
3.6.5.1.2	Sequence Diagram for Register .....	69
3.6.5.1.3	Sequence Diagram for Set Current Location .....	70
3.6.5.1.4	Sequence Diagram for Add to Cart .....	71
3.6.5.1.5	Sequence Diagram for Payment .....	72
3.6.5.1.6	Sequence Diagram for Generate Coupon.....	73
3.6.5.1.7	Sequence Diagram for Confirm Coupon .....	74
3.6.5.1.8	Sequence Diagram for Share Coupon .....	75
3.6.5.1.9	Sequence Diagram For Add Deals.....	76
3.6.5.2	Flowchart .....	77
3.6.5.2.1	Flowchart for Login.....	77

3.6.5.2.2 Flowchart for User Registration.....	78
3.6.5.2.3 Flowchart for Vendor Registration.....	79
3.6.5.2.4 Flowchart for Set Current Location.....	80
3.6.5.2.5 Flowchart for Add to Cart .....	81
3.6.5.2.6 Flowchart for Payment .....	82
3.6.5.2.7 Flowchart for Coupon Generate .....	83
3.6.5.2.8 Flowchart for Confirm Coupon .....	84
3.6.5.2.9 Flowchart for Share Coupon .....	85
3.6.5.2.10 Flowchart for Add Deals.....	86
3.6.5.3 Activity Diagram.....	87
3.6.5.3.1 Activity Diagram for Login .....	87
3.6.5.3.2 Activity Diagram for User Registration .....	88
3.6.5.3.3 Activity Diagram for Vendor Registration .....	89
3.6.5.3.4 Activity Diagram for Set Current Location .....	90
3.6.5.3.5 Activity Diagram for Add to Cart.....	91
3.6.5.3.6 Activity Diagram for Payment.....	92
3.6.5.3.7 Activity Diagram for Generate Coupon.....	93
3.6.5.3.8 Activity Diagram for Confirm Coupon .....	94
3.6.5.3.9 Activity Diagram for Share Coupon .....	95
3.6.5.3.10 Activity Diagram for Add Deals .....	96
3.6.5.4 DFD .....	97
3.6.5.4.1 DFD for Login.....	97
3.6.5.4.2 DFD for User Registration.....	98
3.6.5.4.3 DFD for Vendor Registration.....	99
3.6.5.4.4 DFD for Set Current Location.....	100

3.6.5.4.5 DFD for Add to Cart .....	101
10.6 DFD for Payment .....	102
3.6.5.4.7 DFD for Coupon Generate .....	103
3.6.5.4.8 DFD for Confirm Coupon .....	104
3.6.5.4.9 DFD for Share Coupon .....	105
3.6.5.4.10 DFD for Add Deals .....	106
3.7 Implementation .....	107
3.7.1 Inception.....	107
Final Gantt Chart .....	109
3.7.2 Elaboration .....	109
3.7.2.1 System Architecture Diagram .....	109
3.7.2.2 ERD .....	110
3.7.2.3 Use Case Diagram .....	111
3.7.2.4 Sequence Diagram.....	112
3.7.2.5 Flowchart.....	113
3.7.2.6 Activity Diagram.....	114
3.7.2.7 Wireframes .....	115
3.7.2.8 Prototypes .....	115
For Customer.....	115
For Vendor.....	133
3.7.3 Construction .....	141
3.7.3.1 Database Design .....	141
3.7.3.2 Application Endpoints .....	146
3.7.3.3 Mobile Application .....	147
Iteration 1 .....	147

Iteration 2 .....	160
Iteration 3 .....	181
3.7.4 Transition .....	207
CHAPTER 4: TESTING AND ANALYSIS.....	208
4.1 TEST PLAN .....	208
4.1.1 Unit Testing, Test Plan.....	208
For Customer.....	208
4.1.2 System Testing, Test Plan .....	213
4.2 UNIT TESTING.....	214
4.2.1 Registration Empty Field .....	214
4.2.2 Registration with invalid input.....	216
4.2.3 Registration Process .....	218
4.2.4 Token Invalid.....	220
4.2.5 Login Error .....	222
4.2.6 Login Successful .....	224
4.2.7 Invalid password length.....	227
4.2.8 Empty login field.....	229
4.2.9 Save current location in database.....	231
4.2.10 Show current location in homepage .....	233
4.2.11 Add to cart.....	235
4.2.12 Search success.....	242
4.2.13 Search with invalid deal title .....	246
4.2.14 Calculated coupon price.....	248
4.2.15 Send feedback .....	252
4.2.16 Send feedback field validation.....	257

4.2.17 Edit profile .....	259
4.2.18 Edit profile field validation.....	263
4.2.19 Purchase coupon .....	265
4.2.20 Purchase coupon failed.....	275
4.2.21 Active coupon share.....	281
4.2.22 Display active coupon .....	287
4.2.23 Display previous coupon .....	290
4.2.24 Current location recommended deals .....	294
4.2.25 Generated coupon displayed. ....	296
4.2.26 Logout .....	300
4.2.28 Post registration details .....	305
4.2.29 Registration field validation .....	309
4.2.30 Login error.....	312
4.2.31 Login success .....	314
4.2.32 Login field error .....	317
4.2.33 Add deal before uploading business profile.....	319
4.2.34 Upload business profile .....	322
4.2.35 Add deal field validation .....	325
4.2.36 Add deal success .....	328
4.2.37 Confirm coupon success.....	333
4.2.38 Display active coupon .....	338
4.2.39 Display previous coupon .....	342
4.2.40 Recently added deals.....	347
4.2.41 Logout .....	349
4.3 SYSTEM TESTING .....	354

4.3.1 Connectivity Testing .....	354
4.3.2 Compatibility testing .....	357
4.3.3 Application without online server.....	359
4.3.4 System walkthrough testing for customer .....	360
4.3.4 System walkthrough testing for vendor .....	385
4.4 Critical Analysis .....	403
4.4.1 Development with Methodology .....	403
4.4.2 Testing Analysis .....	403
CHAPTER 5: OVERALL CRITICAL ANALYSIS .....	404
CHAPTER 6: CONCLUSION .....	405
6.1 LEGAL, SOCIAL AND ETHICAL ISSUES .....	406
6.1.1 Legal Issues .....	406
6.1.2 Social Issues .....	406
6.1.3 Ethical Issues .....	407
6.2 ADVANTAGES .....	408
6.3 LIMITATIONS .....	409
6.4 FUTURE WORK .....	410
CHAPTER 7: BIBLIOGRAPHY .....	412
CHAPTER 8: APPENDIX .....	414
8.1 APPENDIX 1: PRE-SURVEY .....	414
8.1.1 PRE-SURVEY FORM .....	414
8.1.2 Sample of Filled Pre-Survey Forms .....	419
8.1.3 Pre-Survey Result .....	425
8.2 APPENDIX 2: POST-SURVEY .....	431
8.2.1 Post-Survey Form .....	431

---

8.2.2 SAMPLE OF FILLED POST-SURVEY FORMS .....	435
8.2.3 Post-Survey Result .....	440
8.3 APPENDIX 3: SAMPLE CODE.....	444
8.3.1 Sample Code of the UI.....	444
For Customers.....	444
User Login UI .....	444
User Registration UI.....	459
Forgot Password UI .....	471
Set User Location UI.....	476
User Homepage UI .....	482
User Cart UI.....	507
User Purchase UI.....	515
Profile UI .....	518
Deal UI.....	527
Active Coupon UI .....	535
Previous Coupon UI.....	542
For Vendor.....	549
Vendor Registration UI.....	549
Vendor Login UI .....	563
Vendor Homepage UI .....	575
Vendor Add Deals UI .....	589
Vendor Purchase UI.....	612
8.4 APPENDIX 4: DESIGNS .....	616
8.4.1 GANTT CHART.....	616
8.4.1.1 Final Gantt Chart .....	616

8.4.1.2 Gantt Chart Iteration .....	616
8.4.2 Use Case Diagram.....	617
8.4.2.1 Final Use Case Diagram.....	617
8.4.2.2 Initial Use Case Diagram.....	618
8.4.2.3 High Level Use Case Description .....	619
User Registration .....	619
Vendor Registration .....	619
User/Vendor Login .....	619
Browse Deals.....	620
Payment.....	620
View Purchased Coupons.....	620
Share Active Coupons .....	620
Customize User/Vendor Details .....	621
Customize Deal Details.....	621
Notify User .....	621
Upload Profile .....	621
Add Deals .....	622
View Coupons Details.....	622
Verify Active Coupons.....	622
8.4.3 ERD Iterations.....	623
8.4.3.1 Final ERD of the system .....	623
8.4.3.2 ERD Iteration .....	624
8.4.4 WIREFRAMES .....	625
Set Location Page .....	625
Search Page.....	626

Deal Details Page .....	627
Cart Page .....	628
Payment Page .....	629
Coupon Details Page.....	630
Profile Page.....	631
Edit Profile Page.....	632
Send Feedback Page .....	633
Share Page.....	634
Vendor Add Deals Page .....	635
Vendor Active Coupons Page.....	636
Vendor Active Coupon Details Page .....	637
Vendor Used Coupons Page .....	638
<b>8.5 APPENDIX: 5 SRS (SOFTWARE REQUIREMENTS SPECIFICATION) .....</b>	<b>639</b>
<b>8.5.1 PROJECT TITLE .....</b>	<b>639</b>
<b>8.5.2 CATEGORY .....</b>	<b>639</b>
<b>8.5.3 INTRODUCTION .....</b>	<b>639</b>
<b>8.5.3.1 Purpose.....</b>	<b>639</b>
<b>8.5.3.2 Targeted Audiences and Reading Suggestions .....</b>	<b>639</b>
<b>8.5.3.3 Project Scope.....</b>	<b>639</b>
<b>8.5.3.4 Existing System .....</b>	<b>640</b>
<b>8.5.3.5 Purposed System.....</b>	<b>640</b>
<b>8.5.4 Overall Description .....</b>	<b>641</b>
<b>8.5.4.1 System Perspective .....</b>	<b>642</b>
<b>8.5.4.2 System Features .....</b>	<b>642</b>
<b>8.5.5 USER CLASS AND CHARACTERISTICS.....</b>	<b>644</b>

---

8.5.6 ASSUMPTION AND DEPENDENCIES .....	645
8.5.7 FUNCTIONAL REQUIREMENTS.....	646
8.5.7.1 REGISTRATION .....	646
8.5.7.2 Login .....	647
8.5.7.3 Add Deal to Cart .....	648
8.5.7.4 Buy Coupons .....	649
8.5.7.5 User Feedback.....	650
8.5.7.6 Payment.....	651
8.5.7.7 Search.....	652
8.5.7.8 Coupon Status .....	653
8.5.7.9 Add Deals .....	654
8.5.8 Non-Functional Requirements .....	655
8.5.8.1 Performance Requirements .....	655
8.5.8.2 Safety Requirements.....	656
8.6 APPENDIX 6: USER MANUAL.....	657
8.6.1 Customer.....	658
8.6.2 For Vendor .....	667
8.7 APPENDIX 7: FUTURE WORK .....	670
8.6.1 READING FOR FUTURE WORK.....	670
Rating .....	670
QR generator.....	672
Chatbot.....	675
Another E-payment option implementation (E-Sewa).....	677

## Table of Tables

Table 1: Justification for not choosing waterfall methodology. ....	17
Table 2: Justification for not choosing waterfall methodology. ....	18
Table 3: Justification for choosing RUP Methodology. ....	22
Table 4: Justification for choosing RUP Methodology. ....	22
Table 5: Justification for choosing RUP Methodology. ....	22
Table 6: Functional Requirements of Register. ....	35
Table 7: Functional Requirements of Login. ....	36
Table 8: Functional Requirements of Add deals to cart. ....	37
Table 9: Functional Requirements of Buy Coupons. ....	38
Table 10: Functional Requirements of User Feedback. ....	39
Table 11: Functional Requirements of Payment. ....	40
Table 12: Functional Requirements of Search. ....	41
Table 13: Functional Requirements of Coupon Status. ....	42
Table 14: Functional Requirements of Add Deals. ....	43
Table 15: Performance Requirements. ....	44
Table 16: Safety Requirements. ....	45
Table 17: Test Plan for Unit Testing for Customer. ....	210
Table 18: Test Plan for Unit Testing for Vendor. ....	212
Table 19: Test Plan for System Testing. ....	213
Table 20: Registration field validation testing table. ....	214
Table 21: Registration with invalid input testing table. ....	216
Table 22: Registration success testing table. ....	218
Table 23: Token Invalid testing table. ....	220
Table 24: Login error testing table. ....	222
Table 25: Login success testing table. ....	224
Table 26: Invalid password length testing table. ....	227
Table 27: Empty login field testing table. ....	229
Table 28: Save current location in database testing table. ....	231
Table 29: Show current location in homepage testing table. ....	233
Table 30: Add to cart testing table. ....	235

Table 31: Search success testing table.....	242
Table 32: Invalid search testing table.....	246
Table 33: Calculated coupon price testing table.....	248
Table 34: Send feedback testing table.....	252
Table 35: Send feedback field validation testing table.....	257
Table 36: Edit profile testing table.....	259
Table 37: Edit profile field validation testing table.....	263
Table 38: Purchase coupon testing table.....	265
Table 39: Purchase coupon failed testing table.....	275
Table 40: Share coupon testing table.....	281
Table 41: Display active coupon testing table.....	287
Table 42: Display previous coupon testing table.....	290
Table 43: Current location recommendation testing table.....	294
Table 44: Generated coupon testing table.....	296
Table 45: Logout testing table.....	300
Table 46: Post registration details testing table.....	305
Table 47: Registration field validation testing table.....	309
Table 48: Login error testing table.....	312
Table 49: Login success testing table.....	314
Table 50: Login field validation testing table.....	317
Table 51: Adding deal before uploading business profile testing table.15.....	319
Table 52: Upload business profile testing table.....	322
Table 53: Add deal field validation testing table.....	325
Table 54: Add deal testing table.....	328
Table 55: Confirm coupon testing table.....	333
Table 56: Display active coupons testing table.....	338
Table 57: Display previous coupons testing table.....	342
Table 58: Recently added deal testing table.....	347
Table 59: Logout testing table.....	349
Table 60: Connectivity testing.....	354
Table 61: Compatibility Testing.....	357

---

Table 62: Application without online server testing.....	359
Table 63: System walkthrough testing table.....	360
Table 64: System walkthrough testing table for vendor.....	385
Table 65: High Level Use Case of User Registration.....	619
Table 66:High Level Use Case of Vendor Registration.....	619
Table 67: High Level Use Case of User/Vendor Login .....	619
Table 68: High Level Use Case of Browse Deals.....	620
Table 69: High Level Use Case of Payment.....	620
Table 70: High Level Use Case of View Purchased Coupons.....	620
Table 71: High Level Use Case of Share Active Coupons.....	620
Table 72: High Level Use Case of Customize User/Vendor Details.....	621
Table 73: High Level Use Case of Customize Deal Details.....	621
Table 74: High Level Use Case of Notify User .....	621
Table 75: High Level Use Case of Upload Profile. ....	621
Table 76: High Level Use Case of Add Deals. ....	622
Table 77: High Level Use Case of View Coupon Details.....	622
Table 78: High Level Use Case of Verify Active Coupons.....	622
Table 79: Final ERD of the system.....	623
Table 80: First Iteration of ERD of the system.....	624
Table 81: Functional Requirements of Register. ....	646
Table 82: Functional Requirements of Login.....	647
Table 83: Functional Requirements of Add deals to cart.....	648
Table 84: Functional Requirements of Buy Coupons. ....	649
Table 85: Functional Requirements of User Feedback. ....	650
Table 86: Functional Requirements of Payment. ....	651
Table 87: Functional Requirements of Search. ....	652
Table 88: Functional Requirements of Coupon Status. ....	653
Table 89: Functional Requirements of Add Deals. ....	654
Table 90: Performance Requirements.....	655
Table 91: Safety Requirements.....	656

---

## Table of Figures

Figure 1: Dart Logo (Google, 2019) .....	7
Figure 2: Python Logo (Wikipedia, 2008) .....	7
Figure 3: Django Logo (Django Community, 2023) .....	8
Figure 4: SQLite Logo (Google, 2013) .....	8
Figure 5: Khalti Logo (Khalti Digital Wallet, 2018).....	9
Figure 6: RegExp Logo (eneko, 2021) .....	9
Figure 7: Hive Logo (simc, 2023) .....	10
Figure 8: Similar Project-Slick deals.....	12
Figure 9: Similar Project-Groupon .....	13
Figure 10: Similar Project-Nearbuy .....	14
Figure 11: Comparison of the similar projects .....	15
Figure 12: RUP Methodology .....	21
Figure 13: Phases of RUP Methodology in table.....	23
Figure 14: Response 1 .....	24
Figure 15: Response 2 .....	24
Figure 16: Response 3 .....	25
Figure 17: Response 4 .....	25
Figure 18: Response 5 .....	26
Figure 19: Response 6 .....	26
Figure 20: Response 7 .....	27
Figure 21: Response 8 .....	27
Figure 22: Response 9 .....	28
Figure 23: Response 10 .....	28
Figure 24: Response 11 .....	29
Figure 25: post-Survey response 1 .....	30
Figure 26: post-Survey response 2 .....	31
Figure 27: post-Survey response 3 .....	31
Figure 28: post-Survey response 4 .....	32
Figure 29: post-Survey response 5 .....	32
Figure 30: post-Survey response 6 .....	33

Figure 31: post-Survey response 7 .....	33
Figure 32: post-Survey response 8 .....	34
Figure 33: KarKhana Logo .....	46
Figure 34: Use Case Diagram.....	47
Figure 35: Overall ERD of the System.....	48
Figure 36: Overall Activity Diagram of the system.....	49
Figure 37: Overall Sequence diagram of the system.....	50
Figure 38: Overall flowchart of the system.....	51
Figure 39: System Architecture Diagram of the system .....	52
Figure 40: System Overview Diagram of the system .....	53
Figure 41: Wireframe of Welcome Page.....	54
Figure 42: Wireframe of Customer Login Page .....	55
Figure 43: Wireframe of Customer registration Page .....	56
Figure 44: Wireframe of Customer Homepage.....	57
Figure 45: Wireframe of Vendor Login Page .....	58
Figure 46: Wireframe of Customer Registration Page.....	59
Figure 47: Wireframe of Vendor Homepage.....	60
Figure 48: UI Design of Welcome page.....	61
Figure 49: UI Design of Customer Login Page .....	62
Figure 50: UI Design of Customer Registration Page .....	63
Figure 51: UI Design of Customer Homepage .....	64
Figure 52: UI Design of Vendor Login Page.....	65
Figure 53: UI Design of Vendor Registration Page .....	66
Figure 54: UI Design of Vendor Homepage .....	67
Figure 55: Sequence Diagram for Login.....	68
Figure 56: Sequence Diagram for Register .....	69
Figure 57: Sequence Diagram for Set Current Location.....	70
Figure 58: Sequence Diagram for Add to cart .....	71
Figure 59: Sequence Diagram for Payment .....	72
Figure 60: Sequence Diagram for Generate coupon.....	73
Figure 61: Sequence Diagram for Confirm Coupon .....	74

Figure 62: Sequence Diagram for Share Coupon.....	75
Figure 63: Sequence Diagram for Add Deals.....	76
Figure 64: Flowchart for Login.....	77
Figure 65: Flowchart for User Registration.....	78
Figure 66: Flowchart for Vendor Registration.....	79
Figure 67: Flowchart for Set Current Location.....	80
Figure 68: Flowchart for add to cart.....	81
Figure 69: Flowchart for Payment.....	82
Figure 70: Flowchart for generate coupon.....	83
Figure 71: Flowchart for Confirm Coupon.....	84
Figure 72: Flowchart for Share Coupon.....	85
Figure 73: Flowchart for Add Deals.....	86
Figure 74: Activity Diagram for Login.....	87
Figure 75: Activity Diagram for User Registration.....	88
Figure 76: Activity Diagram for Vendor Registration.....	89
Figure 77: Activity Diagram for Set Current Location.....	90
Figure 78: Activity Diagram for Add to cart.....	91
Figure 79:Activity Diagram for Payment.....	92
Figure 80: Activity Diagram for Coupon Generate.....	93
Figure 81: Activity Diagram for Confirm Coupon.....	94
Figure 82: Activity Diagram for Share Coupon.....	95
Figure 83: Activity Diagram for Add Deals.....	96
Figure 84: Level 2 DFD of Login.....	97
Figure 85: level 2 DFD for User Registration.....	98
Figure 86: level 2 DFD for Vendor Registration.....	99
Figure 87: Level 2 DFD for Set Current Location.....	100
Figure 88: Level 2 DFD for Add to Cart.....	101
Figure 89: Level 2 DFD for Payment.....	102
Figure 90: Level 2 DFD for Coupon Generate.....	103
Figure 91: Level 2 DFD for Confirm Coupon.....	104
Figure 92: Level 2 DFD for Share Coupon.....	105

Figure 93: Level 2 DFD for Add Deals.....	106
Figure 94: Screenshots of Similar Projects. ....	107
Figure 95: Work Break down Structure. ....	108
Figure 96: Final Gantt Chart.....	109
Figure 97: System Architecture Diagram in Elaboration Phase.....	109
Figure 98: ERD Design in Elaboration Phase. ....	110
Figure 99: Use Case Diagram in Elaboration Phase.....	111
Figure 100: Sequence Diagram in Elaboration Phase.....	112
Figure 101: Overall Flowchart in Elaboration Phase. ....	113
Figure 102: Overall Activity Diagram in Elaboration Phase.....	114
Figure 103: Prototype of Welcome Page.....	115
Figure 104: Prototype of Login Page.....	116
Figure 105: Prototype of Registration Page. ....	117
Figure 106: Prototype of set current location page.....	118
Figure 107: Prototype of Homepage. ....	119
Figure 108: Prototype of Deal Details Page. ....	120
Figure 109: Prototype of Cart page. ....	121
Figure 110: Prototype of Settings Page.....	122
Figure 111: Prototype of Active Coupons Page. ....	123
Figure 112: Prototype of Previous Coupon Page.....	124
Figure 113: Prototype of Coupon Details Page. ....	125
Figure 114: Prototype of Share Page.....	126
Figure 115: Prototype of Payment Page. ....	127
Figure 116: Prototype of Send Feedback.....	128
Figure 117: Prototype of Edit Profile Page.....	129
Figure 118: Prototype of About Us Page.....	130
Figure 119: Prototype of Search Page. ....	131
Figure 120: Prototype of Forgot Password.....	132
Figure 121: Prototype of Vendor Login.....	133
Figure 122: Prototype of Vendor Registration. ....	134
Figure 123: Prototype of Vendor Home Page. ....	135

---

Figure 124: Prototype of Add Deals Page.....	136
Figure 125: Prototype of Settings Page.....	137
Figure 126: Prototype of Active Coupons.....	138
Figure 127: Prototype of Previous Coupon.....	139
Figure 128: Prototype of Coupon Details.....	140
Figure 129: Importing the libraries.....	141
Figure 130: Model for User.....	141
Figure 131: Model for Vendor Request for registration.....	142
Figure 132: Model of Vendor for uploading business profile.....	142
Figure 133: Model for Email notification.....	142
Figure 134: Model for Deals .....	143
Figure 135: Model for Cart .....	143
Figure 136: Model for Coupon rate.....	144
Figure 137: Model for Coupon.....	144
Figure 138: Model for Feedback.....	144
Figure 139: Model for Reviews and Ratings.....	145
Figure 140: Registering the database.....	145
Figure 141: Overall Endpoints of the System.....	146
Figure 142: Dart installation in Vs code.....	147
Figure 143: Running Flutter Doctor in Terminal.....	147
Figure 144: Python installation in Vs code.....	148
Figure 145: Django Project Created.....	148
Figure 146: Welcome Page UI.....	149
Figure 147: Customer Login Page UI.....	150
Figure 148: Login Code.....	151
Figure 149: Login API code for both type of users.....	152
Figure 150: User Registration UI.....	153
Figure 151: User Registration Code.....	154
Figure 152: User Registration API code.....	155
Figure 153: User Registration End Point.....	156
Figure 154: Settings UI.....	157

Figure 155: Forget Password UI.....	158
Figure 156: Forget Password Code.....	159
Figure 157: Set Current Location UI.....	160
Figure 158: Set Current Location Code.....	161
Figure 159: Set Current Location API.....	162
Figure 160: Set Current Location Endpoint .....	162
Figure 161: Vendor Login UI.....	163
Figure 162: Vendor Login Code.....	164
Figure 163: Vendor Registration UI.....	165
Figure 164: Vendor Registration Code.....	166
Figure 165: Vendor Registration Endpoint .....	166
Figure 166: Vendor Registration API.....	167
Figure 167: Vendor Homepage UI.....	168
Figure 168: Customer Homepage UI.....	169
Figure 169: Vendor Add Deals UI.....	170
Figure 170: Vendor Add Deals Code.....	171
Figure 171: Add Deal Endpoint .....	171
Figure 172: Add Deals API - 1.....	172
Figure 173: Add Deals API - 2.....	173
Figure 174: Search UI.....	174
Figure 175: Search Code.....	175
Figure 176: Search Code - 2 .....	176
Figure 177: Search Code - 3 .....	177
Figure 178: Seach API by Deal Title.....	178
Figure 179: Admin Send Email Verification.....	179
Figure 180: Recently added Deals UI.....	180
Figure 181: Generated Coupon UI.....	181
Figure 182: Generate Coupon Code .....	182
Figure 183: Generate Coupon API.....	183
Figure 184: Active Coupon UI.....	184
Figure 185: Active Coupon Code - 1 .....	185

Figure 186: Active Coupon History Code - 2 .....	186
Figure 187: Previous Coupon UI.....	187
Figure 188: Previous Coupon Code - 1.....	188
Figure 189: Previous Coupon Code - 2 .....	189
Figure 190: Notification in mail.....	190
Figure 191: Send notification Code .....	190
Figure 192: Send Notification API.....	191
Figure 193: Add to Cart UI.....	192
Figure 194: Cart code - 1.....	193
Figure 195: Cart Code - 2.....	194
Figure 196: Cart API.....	195
Figure 197: Redeem Coupon UI. ....	196
Figure 198: Redeem Coupon Code. ....	197
Figure 199: Redeem Coupon API. ....	198
Figure 200: Location Based Recommendation in Homepage. ....	199
Figure 201: Location based recommendation system code -1.....	200
Figure 202: Location based recommendation system code -2.....	201
Figure 203: Location based recommendation system code -3.....	202
Figure 204: Location based recommendation system API. ....	202
Figure 205: Feedback UI.....	203
Figure 206: Feedback Code.....	204
Figure 207: Feedback API.....	205
Figure 208: Edit profile UI.....	206
Figure 209: Post Survey in Transition Phase. ....	207
Figure 210: Documentation of the application. ....	207
Figure 211: Registration field Validation Testing. ....	215
Figure 212: Registration field test with invalid input. ....	217
Figure 213: Successful registration testing.....	219
Figure 214: Token Sent in mail. ....	220
Figure 215: Invalid Token Testing. ....	221
Figure 216: Invalid Email/Password Testing. ....	223

Figure 217: Login with valid credentials. ....	225
Figure 218: Logged in Successfully. ....	226
Figure 219: Invalid Password length testing.....	228
Figure 220: Empty Login Field Validation Testing. ....	230
Figure 221: Setting Location of Emulator. ....	231
Figure 222: Using Current Location Testing.....	232
Figure 223: Testing whether the location is set in database or not. ....	233
Figure 224: Show Current location in homepage testing.....	234
Figure 225: Add to Cart Testing -1. ....	236
Figure 226: Add to Cart Testing -2.....	237
Figure 227: Add to Cart Testing -3.....	238
Figure 228: Add to Cart Testing -4.....	239
Figure 229: Add to Cart Testing -5.....	240
Figure 230: Add to Cart Testing -5.....	241
Figure 231: Search testing -1.....	243
Figure 232: Search Testing -2.....	244
Figure 233: Search Testing -3.....	245
Figure 234: Invalid Deal Search Testing. ....	247
Figure 235: Calculated Coupon price Testing -1.....	249
Figure 236: Calculated Coupon Price Testing -2.....	250
Figure 237: Calculated Coupon Price Testing -3.....	251
Figure 238: Send Feedback Testing -1. ....	253
Figure 239: Send Feedback Testing -2. ....	254
Figure 240: Send Feedback Testing -3. ....	255
Figure 241: Send Feedback Testing -4. ....	256
Figure 242: Feedback Successfully Updated in Database. ....	257
Figure 243: Send Feedback Validation Testing.....	258
Figure 244: Testing Profile Testing -1. ....	260
Figure 245: Edit Profile Testing -2. ....	261
Figure 246: Edit Profile Testing -3. ....	262
Figure 247: Profile Successfully Updated in Database.....	263

---

Figure 248: Edit Profile Validation Testing .....	264
Figure 249: Coupon Purchase Testing -1.....	266
Figure 250: Coupon Purchase Testing -2.....	267
Figure 251: Coupon Purchase Testing -3.....	268
Figure 252: Coupon Purchase Testing -4.....	269
Figure 253: Coupon Purchase Testing -5.....	270
Figure 254: Coupon Purchase Testing -6.....	271
Figure 255: Code Received in SMS and Gmail.....	272
Figure 256: Coupon Purchase Testing -7.....	273
Figure 257: Coupon Purchased Successfully.....	274
Figure 258: Successful Coupon Purchase Notification in Gmail. ....	275
Figure 259: Coupon Purchase Failed Testing -1.....	276
Figure 260: Coupon Purchase Failed Testing -2.....	277
Figure 261: Coupon Purchase Failed Testing -3.....	278
Figure 262: Coupon Purchased Failed Testing -4.....	279
Figure 263: Coupon Purchased Failed Testing -5.....	280
Figure 264: Active Coupon Share Testing -1. ....	282
Figure 265: Active Coupon Share Testing -2. ....	283
Figure 266: Active Coupon Share Testing -3. ....	284
Figure 267: Active Coupon Share Testing -4. ....	285
Figure 268: Active Coupon Share Testing -5. ....	286
Figure 269: Coupon Successfully Received.....	287
Figure 270: Active Coupon Testing -1.....	288
Figure 271: Active Coupon Testing -2.....	289
Figure 272: Previous Coupon Testing -1.....	291
Figure 273: Previous Coupon Testing -2.....	292
Figure 274: Previous Coupon Testing -3.....	293
Figure 275: Current Location Recommendation Testing.....	295
Figure 276: Generate Coupon Display Testing -1.....	297
Figure 277: Generated Coupon Testing -2.....	298
Figure 278: Generated Coupon Testing -3.....	299

---

Figure 279: Logout Testing -1 .....	301
Figure 280: Logout Testing -2 .....	302
Figure 281: Logout Testing -3 .....	303
Figure 282: Logout Testing -4 .....	304
Figure 283: Registration testing -1 .....	306
Figure 284: Registration testing -2 .....	307
Figure 285: Registration testing -3 .....	308
Figure 286: Registration field validation testing -1.....	310
Figure 287: Registration field validation testing -2.....	311
Figure 288: Login Error. ....	313
Figure 289: Login Testing -1 .....	315
Figure 290: Logged in Successfully. ....	316
Figure 291: Login Validation testing.....	318
Figure 292: Add deal before business profile -1.....	320
Figure 293: Add deal before business profile -2.....	321
Figure 294: Upload profile testing -1 .....	323
Figure 295: Upload profile testing -2 .....	324
Figure 296: Add Deals Validation Testing -1.....	326
Figure 297: Add Deals Validation Testing -2.....	327
Figure 298: Add Deals Testing -1.....	329
Figure 299: Add Deals Testing -2.....	330
Figure 300: Add Deals Testing -3.....	331
Figure 301: Add Deals Testing -4.....	332
Figure 302: Confirm Coupon Testing -1.....	334
Figure 303: Confirm Coupon Testing -2.....	335
Figure 304: Confirm Coupon Testing -3.....	336
Figure 305: Confirm Coupon Testing -4.....	337
Figure 306: Display active coupons testing -1.....	339
Figure 307: Display active coupons testing -2.....	340
Figure 308: Display active coupons testing -4.....	341
Figure 309: Display previous coupons testing -1.....	343

Figure 310: Display previous coupons testing -2.....	344
Figure 311: Display previous coupons testing -3.....	345
Figure 312: Display previous coupons testing -4.....	346
Figure 313Recently added deal testing.....	348
Figure 314: Logout testing -1 .....	350
Figure 315: Logout testing -2 .....	351
Figure 316: Logout testing -3 .....	352
Figure 317: Logout testing -4 .....	353
Figure 318: Connectivity testing -1.....	355
Figure 319: Connectivity Testing -2.....	356
Figure 320: Compatibility test in Android phone.....	358
Figure 321: Application without online server testing -1 (Django Project not running).359	
Figure 322: Application without online server testing -2 (Error Occurred in flutter project). .....	360
Figure 323: System walkthrough testing -1.....	361
Figure 324: System walkthrough testing -2.....	362
Figure 325: System walkthrough testing -3.....	363
Figure 326: System walkthrough testing -4.....	364
Figure 327: System walkthrough testing -5.....	365
Figure 328: System walkthrough testing -6.....	366
Figure 329: System walkthrough testing -7.....	367
Figure 330: System walkthrough testing -8.....	368
Figure 331: System walkthrough testing -9.....	369
Figure 332: System walkthrough testing -10.....	370
Figure 333: System walkthrough testing -11.....	371
Figure 334: System walkthrough testing -12.....	372
Figure 335: System walkthrough testing -13.....	373
Figure 336: System walkthrough testing -14.....	374
Figure 337: System walkthrough testing -15.....	375
Figure 338: System walkthrough testing -16.....	376
Figure 339: System walkthrough testing -17.....	377

Figure 340: System walkthrough testing -18 .....	378
Figure 341: System walkthrough testing -19 .....	379
Figure 342: System walkthrough testing -20 .....	380
Figure 343: System walkthrough testing -21 .....	381
Figure 344: System walkthrough testing -22 .....	382
Figure 345: System walkthrough testing -23 .....	383
Figure 346: System walkthrough testing -24 .....	384
Figure 347: System walkthrough testing table for vendor -1 .....	386
Figure 348: System walkthrough testing table for vendor -2 .....	387
Figure 349: System walkthrough testing table for vendor -3 .....	388
Figure 350: System walkthrough testing table for vendor -4 .....	389
Figure 351: System walkthrough testing table for vendor -5 .....	390
Figure 352: System walkthrough testing table for vendor -6 .....	391
Figure 353: System walkthrough testing table for vendor -7 .....	392
Figure 354: System walkthrough testing table for vendor -8 .....	393
Figure 355: System walkthrough testing table for vendor -9 .....	394
Figure 356: System walkthrough testing table for vendor -10 .....	395
Figure 357: System walkthrough testing table for vendor -11 .....	396
Figure 358: System walkthrough testing table for vendor -12 .....	397
Figure 359: System walkthrough testing table for vendor -13 .....	398
Figure 360: System walkthrough testing table for vendor -14 .....	399
Figure 361: System walkthrough testing table for vendor -15 .....	400
Figure 362: System walkthrough testing table for vendor -16 .....	401
Figure 363: System walkthrough testing table for vendor -17 .....	402
Figure 364: Pre-Survey form question screenshot-1.....	414
Figure 365: Pre-Survey form question screenshot-2.....	415
Figure 366: Pre-Survey form question screenshot-3.....	416
Figure 367: Pre-Survey form question screenshot-4.....	417
Figure 368: Pre-Survey form question screenshot-5.....	418
Figure 369: Pre-Survey filled form screenshot-1.....	419
Figure 370: Pre-Survey filled form screenshot-2.....	420

Figure 371: Pre-Survey filled form screenshot-3.....	421
Figure 372: Pre-Survey filled form screenshot-4.....	422
Figure 373: Pre-Survey filled form screenshot-5.....	423
Figure 374: Pre-Survey filled form screenshot-6.....	424
Figure 375: Response 1.....	425
Figure 376: Response 2.....	426
Figure 377: Response 3.....	426
Figure 378: Response 4.....	427
Figure 379: Response 5.....	427
Figure 380: Response 6.....	428
Figure 381: Response 7 .....	428
Figure 382: Response 8.....	429
Figure 383: Response 9.....	429
Figure 384: Response 10.....	430
Figure 385: Response 11.....	430
Figure 386: post-Survey form screenshot-1.....	431
Figure 387: post-Survey form screenshot-2.....	432
Figure 388: post-Survey form screenshot-3.....	433
Figure 389: post-Survey form screenshot-4.....	434
Figure 390: post-Survey form screenshot-5.....	434
Figure 391: post-Survey filled form screenshot-1.....	435
Figure 392: post-Survey filled form screenshot-2.....	436
Figure 393: post-Survey filled form screenshot-3.....	437
Figure 394: post-Survey filled form screenshot-4.....	438
Figure 395: post-Survey filled form screenshot-5.....	439
Figure 396: post-Survey response 1 .....	440
Figure 397: post-Survey response 2 .....	440
Figure 398: post-Survey response 3 .....	441
Figure 399: post-Survey response 4 .....	441
Figure 400: post-Survey response 5 .....	442
Figure 401: post-Survey response 6 .....	442

---

Figure 402: post-Survey response 7 .....	443
Figure 403: post-Survey response 8 .....	443
Figure 404:Final Gantt Chart.....	616
Figure 405: First Iteration of Gantt Chart.....	616
Figure 406: Final Use Case Diagram. ....	617
Figure 407: Initial Use Case Diagram.....	618
Figure 408: Wireframe of Set Location Page .....	625
Figure 409: Wireframe of Search Page.....	626
Figure 410: Wireframe of Deal Details Page.....	627
Figure 411: Wireframe of Cart Page.....	628
Figure 412: Wireframe of Payment Page. ....	629
Figure 413: Wireframe of Coupon Details page. ....	630
Figure 414: Wireframe of Profile Page. ....	631
Figure 415: Wireframe of Edit Profile Page.....	632
Figure 416: Wireframe of Send Feedback Page. ....	633
Figure 417: Wireframe of share page.....	634
Figure 418: Wireframe of Vendor Add deals page. ....	635
Figure 419: Wireframe of Vendor Active Coupons Page.....	636
Figure 420: Wireframe of Vendor Active Coupon Details Page.....	637
Figure 421: Wireframe of Vendor Used Coupons Page. ....	638
Figure 422: System Perspective of Android Application.....	641
Figure 423: Customer manual Step-1. ....	658
Figure 424: Customer manual Step-2. ....	659
Figure 425: Customer manual Step-3. ....	660
Figure 426: Customer manual Step-4. ....	661
Figure 427: Customer manual Step-5. ....	662
Figure 428: Customer manual Step-6. ....	663
Figure 429: Customer manual Step-7. ....	664
Figure 430: Customer manual Step-8. ....	665
Figure 431: Customer manual Step-9. ....	666
Figure 432: Vendor manual Step-1. ....	667

Figure 433: Vendor manual Step-2 .....	668
Figure 434: Vendor manual Step-3. ....	669
Figure 435: Rating bar research.....	670
Figure 436: Rating Bar Research (II).....	671
Figure 437: QR generator (I) .....	672
Figure 438: QR generator (II).....	672
Figure 439: QR generator (III) .....	673
Figure 440: QR generator (IV).....	673
Figure 441: QR generator (V).....	674
Figure 442: Implementing Chatbot in flutter application research (I) .....	675
Figure 443: Implementing Chatbot in Flutter Application Research (II) .....	676
Figure 444: E-Sewa payment integration research (I) .....	677
Figure 445: E-Sewa payment integration research (II) .....	678

## CHAPTER1: INTRODUCTION

### 1.1 INTRODUCTION TO THE TOPIC

KarKhana is a mobile application that enables businesses to advertise customers discounts and offers. This app is developed to add value to Nepalese customers and businesses.

KarKhana is a platform where users sign up to discover and purchase deals, discounts, and offers from various registered businesses. Users can buy coupons through the app, but a valid mobile number is required. Businesses, including restaurants, spas, salons, and more, can advertise customizable promotions like off-hour discounts, happy hour specials, and combo offers on KarKhana. Join us to showcase your amazing deals.

#### Benefits for Users

- 
- i. Uncover fantastic offers from nearby establishments.
  - ii. Secure desired deals effortlessly with digital payment options.
  - iii. Easily redeem discounts by visiting the businesses directly.

#### Benefits for multi-vendors

- 
- i. Attract fresh clientele with enticing discounts.
  - ii. Enhance business during off-peak hours by advertising appealing happy hour specials.
  - iii. Increase customer reach by enlisting your business on the KarKhana platform.
  - iv. Craft flexible discount schemes with various timeframes, such as daily, weekly, monthly, or annually.

---

## 1.2 CURRENT SCENARIO

---

### 1.2.1 CURRENT SCENARIO BASED ON RESEARCH

As more people use their phones, they want an easy way to find and buy online deals. Businesses offer discounts to attract customers, but many people miss out because they don't know about the best deals. Businesses face tough competition, so offering discounts is crucial. People are always looking for good bargains to save money. Since the market for discounts is growing, KarKhana helps businesses and users connect, so they both benefit.

### 1.2.2 CURRENT SCENARIO BASED ON SURVEY

---

According to the survey, most of the people are interested in discounts while some were confused whether they want discounts and sales or not. Half of the respondents were unfamiliar with the deal's apps and found difficult to get discounts. Most of the people were searching for deals on the daily basis. Most of the respondents have bad experience while searching for deals. Based on the survey, it is very important to provide discounts and offers by the businesses so that they can attract customers.

---

## 1.3 PROBLEM DOMAIN

---

### 1.3.1 PROBLEM DOMAIN LISTED FROM RESEARCH

---

- i. Difficulty in growing business.
- ii. Negotiating deals and ensuring a mutually beneficial relationship can be complex (Bryant, 2023).
- iii. Building trust among users is crucial.
- iv. People's hectic schedules can disrupt the chances of being up to date on the new event and offers.

### 1.3.2 PROBLEM DOMAIN LISTED FROM SURVEY

---

- i. Difficulty in receiving discounts.
- ii. Unknown about the new deals offer near me or at other location.

## 1.4 PROJECT AS A SOLUTION

---

- i. **Become more reachable to user:** Our platform ensures seamless access for users, allowing them to purchase tickets and coupons for specific businesses at any time and from anywhere. This convenience empowers users to explore the incredible deals offered by businesses in their vicinity.
- ii. **Amplified discount opportunities:** In industries where negotiating discounts is challenging, such as healthcare, automotive services, and event tickets, our platform bridges the gap by offering discounted prices that are otherwise hard to come by.
- iii. **Bulk discount options for event enthusiasts:** For those seeking substantial quantities of goods with substantial discounts, our platform caters to their needs by providing attractive bulk purchase offers.
- iv. **Comprehensive event and offer notifications:** This platform keeps users informed about the latest events happening in their local area and beyond. Stay updated on exciting opportunities and experiences.
- v. **Effective business promotion:** Utilize our platform to expand your business's reach and popularity. By showcasing new events and offers, your business can gain a competitive edge, reaching a broader audience and outshining rivals in the market.

---

## 1.5 AIMS AND OBJECTIVES

---

### 1.5.1 AIMS

---

The main goal of this project is to develop a practical, trustworthy, and useful mobile application that will allow users to locate amazing deals nearby, broaden their reach to numerous neglected businesses, and save time by removing the need to physically visit each business to find these deals.

### 1.5.2 OBJECTIVES

---

Some of the objectives to meet the aim mentioned above are:

- i. Develop a user-friendly app for finding local deals.
- ii. Ensure accurate location tracking and reliable deal information.
- iii. Implement robust security measures for user data and transactions.
- iv. Focus on diverse businesses, promoting economic inclusivity.
- v. Optimize algorithms to efficiently filter and display relevant deals.
- vi. Enable notifications for timely access to nearby offers.
- vii. Save users time by eliminating the need to visit each business physically.
- viii. Conduct thorough testing to ensure app reliability.
- ix. Collaborate with local businesses for community engagement.
- x. Continuously gather feedback and iterate for user satisfaction.

---

## 1.6 STRUCTURE OF THE REPORT

---

### 1.6.1 BACKGROUND

The background clarifies the project's needs and describes the project and its target clients, allowing for a better understanding of the project. It is also beneficial to read about comparable projects and compare this project to similar projects to have a better knowledge of the features and components.

### 1.6.2 DEVELOPMENT

This section describes methodology and various part that were included while developing this project. The Rational Unified Process (RUP) approach was used, which is divided into four stages: inception, elaboration, construction, and transition. To better understand the demands of the consumers, a survey was performed, and both pre- and post-survey data were reviewed. Following a rigorous requirement analysis, the design phase was carried out using RUP-specified methodologies such as use cases, activity diagrams, and class diagrams. The implementation phase comprised the creation of essential features and architecture, as well as the inclusion of important screenshots in the development documentation. The RUP methodology guaranteed a systematic

development approach, while the survey and requirement analysis assured that the program fulfilled the demands of the users.

### **1.6.3 TESTING AND ANALYSIS OF PROJECT**

---

Testing and analysis are critical components of software development, and a detailed test strategy was developed for this project. This covered unit testing as well as system testing. The unit testing strategy concentrated on individual program components, with test cases specified for each. Similarly, the system testing strategy guaranteed that the entire system operated smoothly and satisfied the requirements set earlier in the process. Critical analysis was undertaken throughout the testing process to identify any flaws or potential areas for improvement. This enabled for a comprehensive assessment of the program and guaranteed the best quality end outcome.

### **1.6.4 CONCLUSION**

---

The project's conclusion emphasizes the legal, social, and ethical challenges that must be addressed before adopting the suggested solution. The benefits of the project include enhanced efficiency, cost savings, and better resource allocation, while the drawbacks include potential privacy problems and the need for more work to attain full functioning. Future development might concentrate on broadening the solution's use to other industries and overcoming any identified constraints. Overall, this research provides a viable answer to the problem at hand while acknowledging the significance of understanding the larger ramifications and potential implementation problems.

## CHAPTER 2: BACKGROUND

### 2.1 ABOUT THE END USERS

Customers, Vendors, and Admin are the end users, and all will be receiving benefits from the KarKhana application. Providing discounts is the main theme of this project. The main task was to find the people will use and like this app, The types of deals that user are more interested in and how the businesses can provide the specific deals and would the users buy and share all types of deals for their family and friends and enjoy.

According to the [survey](#), 89.67% people like the discounts and sales, half of the people were not familiar using this type of deal application and are happy to use these types of deals application. Most of the people find so difficult to get discounts on their favorite items due to their own personal reasons. 41.8% of the people search the discounted items and services daily, 31.3% weekly, 25.4% monthly and 1.5% people yearly which indicates people love to get discounts and sales. 59.7 % people say it is very important for the businesses to provide discounts to grow their business. 59.7% people like foods and wanted the foods in the best price. Half of the people strongly agree that this application will make the new deals/offers more reachable to them. 85.1% people say that they would love to buy and share all types of deals for your family and friends to enjoy too. About 76.1% people agree this application will help in growth of businesses by the offers/deals they provide even to their unreachable customers through this application.

### 2.2 UNDERSTANDING OF THE PROJECT

#### 2.2.1 OVERVIEW OF THE SYSTEM

The mobile application, "KarKhana," aims to benefit both businesses and customers. Customers can easily purchase or gift exclusive discount coupons for their favorite deals anytime, anywhere. Vendors use the platform to promote their businesses, while admins ensure the smooth operation of the application.

### 2.2.3 TECHNOLOGY USED

#### Programming Language Used

##### i. Dart:

Dart is the frontend language for the application, known for creating fast apps on various platforms. It's versatile, supporting web, mobile, desktop, and server applications (dart.dev, 2023). This language will be used for the front-end in my project.



Figure 1: Dart Logo (Google, 2019).

##### ii. Python:

Python is a user-friendly, high-level programming language with dynamic features. Its simple syntax makes it great for scripting and connecting components. Python prioritizes readability, reducing maintenance costs. This language will be used for the backend development in the project.



Figure 2: Python Logo (Wikipedia, 2008).

## **Framework**

### i. **Django:**

Django is a free, open-source Python web framework designed for fast and pragmatic web development. It aims to simplify the process, allowing developers to focus on building apps efficiently without starting from scratch. This framework will be used in the backend of the project.



*Figure 3: Django Logo (Django Community, 2023).*

### ii. **Django Rest Framework:**

The Django REST framework is a well-known toolkit for creating web APIs using Django. It has a full set of capabilities that make it simple to develop RESTful APIs, as well as handle serialization, authentication, and permission.

## **Database**

### i. **SQLite**

The project uses SQLite as its database, automatically created by the Django project. SQLite is a fast, reliable, and widely used C-language library for SQL databases. This is the database used in my project.



*Figure 4: SQLite Logo (Google, 2013).*

### **External APIs and Terminologies**

#### i. **Khali:**

Khali is a mobile wallet, payment gateway & service provider in Nepal. It allows users to pay for a range of services like basic utility payments, hotel bookings, movie and domestic flight tickets, events and many more. This API will be used for the payment.



Figure 5: Khali Logo (Khali Digital Wallet, 2018).

#### ii. **RegExp:**

Regular expressions (also known as regex or regexp) are strings of letters that define a text input matching method. When a regexp is applied to input text, the text is either accepted or denied depending on whether the regexp matches it.



Figure 6: RegEx Logo (eneko, 2021).

**iii. Hive:**

For flutter and dart apps, the Hive NoSQL database is speedy and convenient. If you require a plain key-value database with few relations that is also easy to use, Hive is quite beneficial. It will be used as temporary database.



Figure 7: Hive Logo (simc, 2023).

**iv. Others:**

Geolocator is used to gain access to the device's location, Future is a type of object that is used to represent a possible value or error that will become accessible in the future, Cubit is a simplified or subset of the BLoC design pattern that streamlines how we handle application state. These are the other terminologies used in the project.

## 2.3 FUNCTIONS AND FEATURES

---

- I. **Login:** Customer and vendor both type of users can login into this application for their own respective goal of using this application.
- II. **Register:** Both customer and vendor can register to Karkhana using mobile application only as registering of both type of user is made possible in this application.
- III. **Set current location:** Customer can set their location through one click of a button. Their current location is tracked and saved by the system.
- IV. **Location Based Recommendation System:** Customer will be displayed all the deals in their area along with all the other deals posted in the system by the vendors.
- V. **Add to cart:** Customers can save their favorite deals in their cart, and from where they can buy the coupon for those respective deals.
- VI. **Payment:** Online payment is also available to buy coupon. Khalti's Online payment service is used in this application.
- VII. **Coupon Generate:** After buying the deal, a coupon is generated with all the information of deal, customer, and vendor along with calculated coupon price. This generated coupon is used by the customer to utilize the deal they bought from the KarKhana application.
- VIII. **Confirm Coupon:** The vendor will be required to validate the usage of the coupon when the customer wants to use it.
- IX. **Share coupon:** Customers can share their active coupons to their friends and family.
- X. **Add Deals:** Deals can be added by vendors from within the application.

## 2.4 REVIEW OF SIMILAR PROJECTS

### 2.4.1 SIMILAR PROJECTS

#### i. Slick Deals

The Slick deals Android app enables users to browse and search for deals, create deal notifications, and engage with the Slick deals' community. Users may vote and comment on offers as well as submit their own deals on the app, which has a barcode scanner for pricing comparisons.

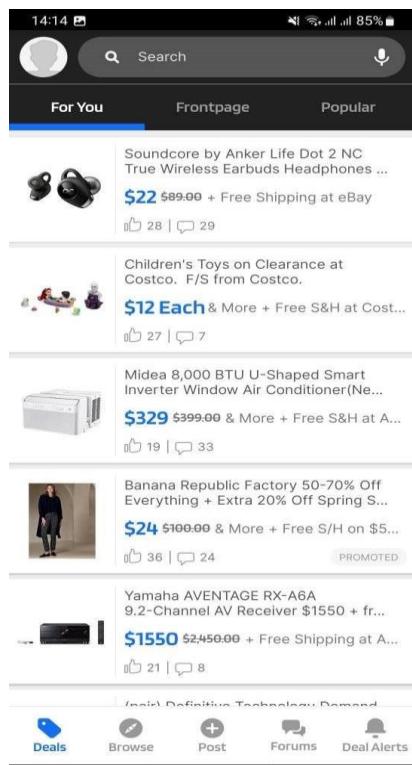


Figure 8: Similar Project-Slick deals.

## ii. Groupon

Users may explore and buy discounts from local businesses, travel offers, and other goods and services using the user-friendly and well-liked Groupon Android app. Users may quickly find fresh offers and buy them immediately from their mobile devices using a credit card or PayPal thanks to customized deal suggestions based on their location and past purchases. Users of the program may also track their vouchers, sign up for push notifications for new and expired discounts, and share bargains with friends and family via social media, email, or text message.



Figure 9: Similar Project-Groupon

### iii. Nearbuy

A well-known Android software called Nearbuy makes it possible for customers and nearby businesses to find and interact with one another on a hyper-local platform. Users of the app may find fresh spots to eat, unwind, and have fun in their city or adjacent locations. In 100,000+ different locations throughout 35+ cities and 18+ categories, the app provides deals and discounts from 50,000 retailers. Directly from the app, users may buy offers and coupons for the activities they want to have.

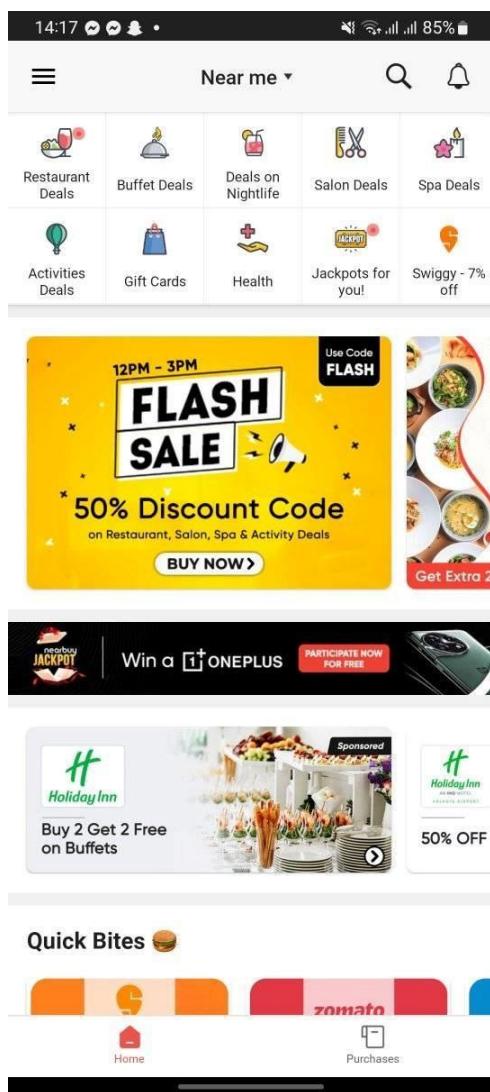


Figure 10: Similar Project-Nearbuy

## 2.4.2 COMPARISON OF SIMILAR PROJECT

**KarKhana Comparison Report**

Features	KarKhana	Slick Deals	Groupon	NearBuy
Current user location tracker	✓	✗	✓	✓
online payment	✓	✗	✓	✓
Location based recommendation	✓	✗	✓	✓
Feedback	✓	✓	✓	✓
Notification	✓	✓	✓	✓
Generate Coupon	✓	✗	✓	✓
Coupon history	✓	✗	✓	✓
Share coupon	✓	✗	✓	✓
Share Deals	✓	✓	✓	✓
Vendor section in mobile application	✓	✗	✗	✗
Add deals from mobile application	✓	✓	✗	✗

Legend: ✓ Always, ✗ Not Available

**CREATED BY**  
Nishan Shrestha

**SOURCES**  
<https://slickdeals.net/>  
<https://www.groupon.com/>  
<https://www.nearbuy.com/>

Made with Visme

Figure 11: Comparison of the similar projects.

## 2.5 CRITICAL EVALUATION

---

To offer and acquire bargains, sellers and customers can use the smartphone application KarKhana. It contains several capabilities like online payments, search integration, and location-based promotions that are comparable to the other three applications we've spoken about. There are, however, certain additional distinctions that make it unique. KarKhana has the benefit of being created with both suppliers and customers in mind. As a result, businesses can readily advertise their special offers, making it simple for customers to discover and buy them. Customers may uncover exclusive bargains that might not be offered on other applications thanks to this, while sellers may find it easier to contact new clients.

The fact that KarKhana is a relatively new app and might not have the same user base and reputation as some of the other applications we've explored is one possible drawback. This can make it more difficult for businesses to reach a large audience and for consumers to locate a variety of offers. The well-known applications Groupon, Slick deals, and NearBuy all have substantial user bases and provide a broad variety of deals. Similar services like deal notifications, search integration, and location-based discounts are available, and they are known for providing excellent deals from reliable suppliers. Because of this, they could be more appealing to clients searching for reputable and reputable deal providers.

Overall, the decision of which app to use may be influenced by a number of variables, including the kinds of discounts provided, the user interface, and the app's level of credibility and repute. Customers seeking for unusual bargains and suppliers trying to broaden their consumer base may both find KarKhana to be a useful option. Customers might also want to think about utilizing reputable apps with a larger selection of offers, such as Groupon, Slick deals, and NearBuy. The success of any mobile application that offers bargains ultimately hinges on the value of the deals, how simple it is to use, and how reliable the platform is.

## CHAPTER 3: DEVELOPMENT

### 3.1 CONSIDERED METHODOLOGIES

- i. Waterfall Methodology
- ii. Scrum Methodology
- iii. Evolutionary Prototyping

#### Waterfall Methodology

Some advantages that made me considered the waterfall methodology are:

- i. It is well-established and traditional approach to software development.
- ii. Its structure is linear and sequential, making it simple to comprehend and manage.
- iii. It emphasizes on detailed documentation, which aids in understanding the requirements and system design.
- iv. It is appropriate for projects with well-defined needs and in a steady environment.

Reasons for not choosing this methodology ae described out below:

- i. It is unsuitable for projects with changing or growing needs because it lacks flexibility.
- ii. When testing and feedback occur late in the development process, it can cause delays and cost overruns.
- iii. It does not put an importance on client participation and satisfaction, which may result in a product that does not satisfy their requirements.

<b>Scenario</b>	Since, it is a final year project which has to be reviewed to supervisor every week.
<b>Justification</b>	But in Waterfall methodology, every step of software development cycle (SDLC) can only be followed one after another.

*Table 1: Justification for not choosing waterfall methodology.*

<b>Scenario</b>	It has a fixed deadline and test should be completed in time.
<b>Justification</b>	Since, testing is carried only in the end of the development in this model. So, Bugs couldn't be caught until the development is completed which can affect the completion of the project in the deadline.

*Table 2: Justification for not choosing waterfall methodology.*

## Scrum Methodology

Some advantages that made me considered the Scrum methodology are:

- i. It is a software development paradigm that fosters flexibility and adaptation.
- ii. It promotes teamwork and communication among team members, resulting in a more productive work environment.
- iii. It offers incremental software feature delivery, allowing for faster feedback and time-to-market.
- iv. It works well for projects with changing requirements and in dynamic situations.

Reasons for not choosing this methodology are described out below:

- i. It may not be appropriate for large and complicated projects since it might cause coordination and management issues.
- ii. It necessitates the availability and affordability of a highly qualified and experienced crew.
- iii. If not handled appropriately, it can lead to scope creep, resulting in a product that is not in line with the original requirements.

<b>Scenario</b>	Since, it is a final year project which has well-defined and fixed requirements.
<b>Justification</b>	But Scrum focuses on adaptability and embraces changes.

*Table 3: Justification for not choosing Scrum methodology.*

<b>Scenario</b>	It has a fixed deadline.
<b>Justification</b>	The fixed timeline doesn't allow for the iterative and incremental nature of Scrum.

*Table 4: Justification for not choosing Scrum Methodology.*

<b>Scenario</b>	Since, It is a individual project and the person should fulfill all the responsibilities of all required person.
<b>Justification</b>	The methodology is not suitable for a single person because it needs a crew for the project.

*Table 5: Justification for not choosing Scrum Methodology.*

## Evolutionary Prototyping

Some advantages that made me considered the Evolutionary prototyping are:

- i. It is an iterative process in which workable prototypes of the software to be developed are created.
- ii. It enables the investigation and testing of various design concepts and user interfaces.
- iii. It allows for more regular and immediate feedback from stakeholders, resulting in a more user-centered design.
- iv. It is appropriate for projects with ambiguous or shifting needs, as well as those where the end-users are not clearly specified.

Reasons for not choosing this methodology are described out below:

- i. It may not be appropriate for projects with stringent time and budget limits, since it might result in an endless timetable and cost increases.
- ii. It necessitates the availability and affordability of a highly qualified and experienced crew.
- iii. It may not be appropriate for projects with fixed needs since it focuses on finding requirements through iterative prototype development. (Martinez, 2023)

<b>Scenario</b>	It has a fixed deadline.
<b>Justification</b>	Since, this model is dependent on the feedback, the project might not be completed in the provided deadline.

*Table 6: Justification for not choosing Prototype model.*

<b>Scenario</b>	The project needs proper resources.
<b>Justification</b>	Since, this model requires huge number of resources as prototype. An individual will lack resources.

*Table 7: Justification for not choosing Prototype model.*

<b>Scenario</b>	The project needs proper scheduling.
<b>Justification</b>	Since, this model is dependent on the feedback and multiple prototypes should be created, the project will not be able to properly scheduled.

*Table 8: Justification for not choosing Prototype model.*


---

### 3.2 SELECTED METHODOLOGIES

For this project, I have chosen Rational Unified Process (RUP) methodology. It is an object-oriented software development process. It is often referred to as the Unified Process Model. Rational Corporation built it, and it is planned and documented using UML (Unified Modeling Language). This procedure is part of the IBM Rational Method Composer (RMC) product.

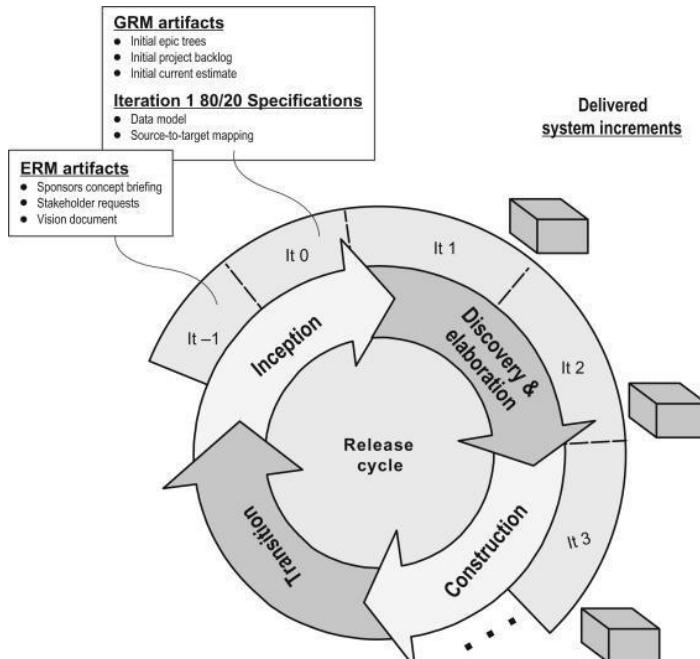


Figure 12: RUP Methodology.

The reason behind selecting RUP methodology are as follows:

- i. **Flexibility:** It is a versatile framework that can be tailored to various project kinds and sizes. It provides a framework for managing complexity and hazards while retaining agility and flexibility.
- ii. **Emphasis on Architecture:** It emphasizes software architecture, ensuring that the system is intended to be scalable, maintainable, and responsive to changing needs. This is critical for projects that require long-term upkeep and upgrades.
- iii. **Collaboration and Communication:** It encourages cooperation and communication among the development team and stakeholders. This guarantees that the product satisfies the criteria and expectations of the stakeholders.
- iv. **Risk Management:** Its approach places a heavy emphasis on risk management, ensuring that possible hazards are recognized early in the development process and mitigation techniques are implemented.

<b>Scenario</b>	The project needs thorough planning and structured approach.
<b>Justification</b>	RUP helps to plan the project in smaller steps, making it easier to handle the project planning.

*Table 3: Justification for choosing RUP Methodology.*

<b>Scenario</b>	Things might change in the development phase.
<b>Justification</b>	RUP allows to change the plans if necessary.

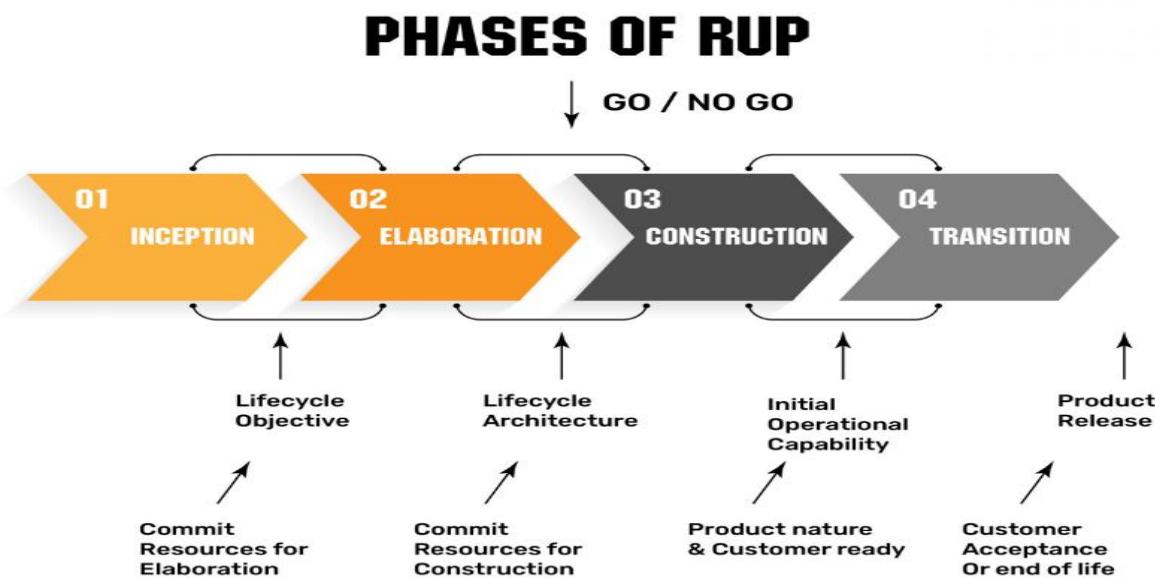
*Table 4: Justification for choosing RUP Methodology.*

<b>Scenario</b>	The project needs well defined artifacts to guide the development process.
<b>Justification</b>	RUP relies on a set of well-defined artifacts to guide the development process. This ensures that there is clear documentation at each phase of the project, aiding in understanding, collaboration, and future maintenance.

*Table 5: Justification for choosing RUP Methodology.*

### 3.3 PHASES OF METHODOLOGY

The Rational Unified Process (RUP) is a software development process that is divided into four stages, each with its own set of tasks and goals.



The following are the phases in my project:

- **Inception:**

The project needs are determined and specified at this phase, which includes project definition, planning, and resource procurement. [How is it used in my project?](#)

- **Elaboration:**

This phase involves analyzing the issue to be solved, evaluating the development timetable, and designing the system architecture. The design is implemented using Balsamic Wireframes, and other UML diagrams are also done at this phase. [How is it used in my project?](#)

- **Construction:**

Both frontend, backend and various testing are done in this phase. With that the system is created during this phase utilizing specialized technologies such as Python, Django, Django Rest Framework, Flutter, and Dart. With the resources obtained, the project plan is improved, and testing and prototyping are carried out. [How is it used in my project?](#)

- **Transition:**

The last step involves beta testing, user feedback, and system documentation. Additional enhancements are made based on user input, and the system is eventually released. Documents of the entire project is submitted. [How is it used in my project?](#)

### 3.3.1 Phases of RUP Methodology in Table

S. N	Tasks	Estimated days	Start Date	End Date
1	Inception	29 Days	17/11/2023	15/12/2023
2	Elaboration	10 Days	16/12/2023	25/12/2023
3	Construction	108 Days	25/12/2023	10/04/2024
4	Transition	14 Days	11/04/2024	24/04/2024

Figure 13: Phases of RUP Methodology in table.

## 3.4 SURVEY RESULTS

### 3.4.1 Pre-Survey Result

The purpose of the [pre-survey](#) was to assist us determine the amount of interest in our proposed system, as well as any potential issues or impediments that may occur throughout the development process. We also intended to obtain a deeper grasp of our target audience's wants and preferences so that we could design our system to match their expectations and provide unique value. Responses from various people was recorded and are shown below:

#### i. Response 1

Do you like discounts and sales?

 Copy

68 responses

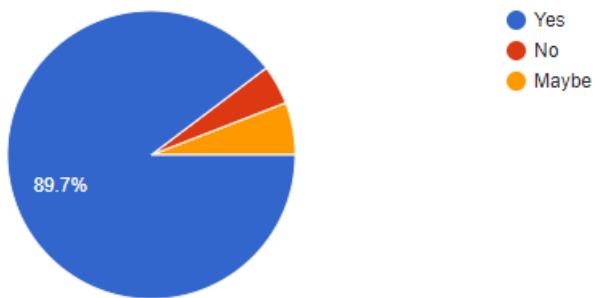


Figure 14: Response 1

#### ii. Response 2

Are you familiar using this type of deals related application?

 Copy

68 responses

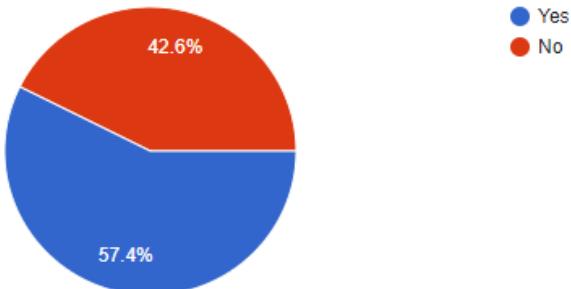


Figure 15: Response 2.

### iii. Response 3

On what scale do you rate the difficulty of getting discounts on your favourite items or services? [!\[\]\(5ce670b29500345973e564b628718938\_img.jpg\) Copy](#)

68 responses

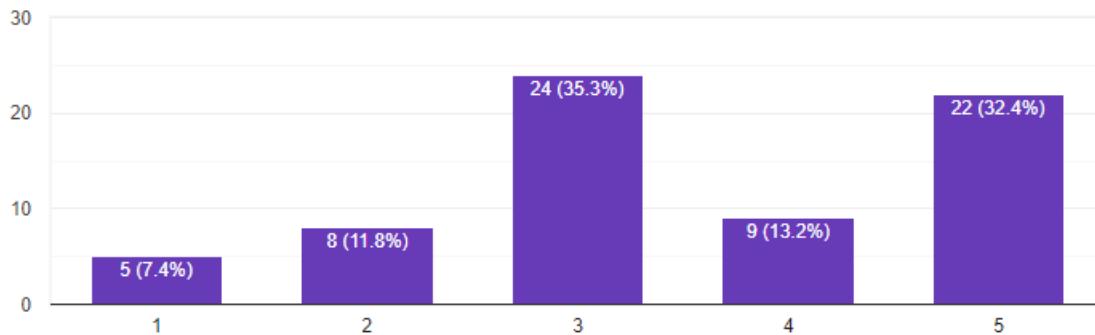


Figure 16: Response 3.

### iv. Response 4

How often do you search for discounted item and services? [!\[\]\(fb4b48dd1f8d628e36dd63ec18d60e31\_img.jpg\) Copy](#)

68 responses

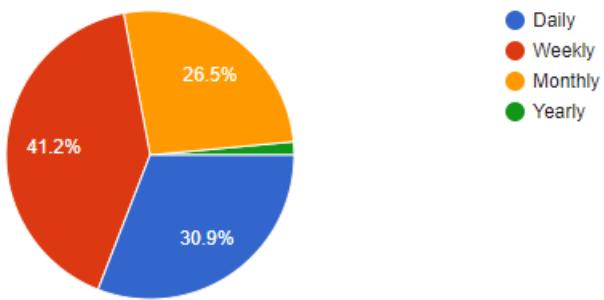


Figure 17: Response 4.

#### v. Response 5

What kind of experience did you have while searching for deals?

 Copy

68 responses

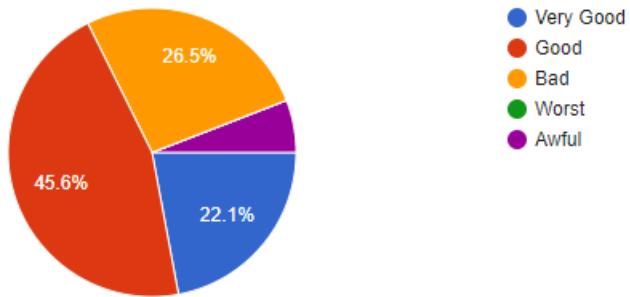


Figure 18: Response 5.

#### vi. Response 6

How important is to give discounts to customers for any type of businesses?

 Copy

68 responses

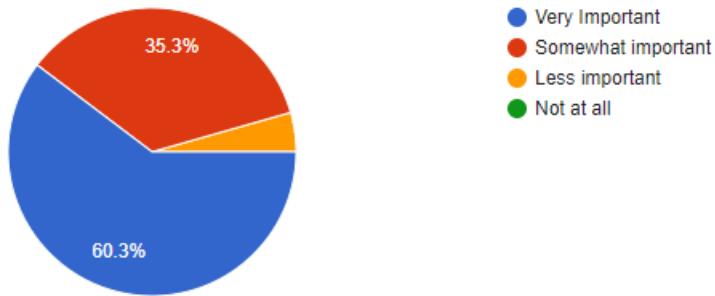


Figure 19: Response 6.

### vii. Response 7

What kind of deals are you more interested in?

 Copy

68 responses



Figure 20: Response 7

### viii. Response 8

Do you agree this application will make the new deals/offers either near or far location more reachable to you?

 Copy

68 responses

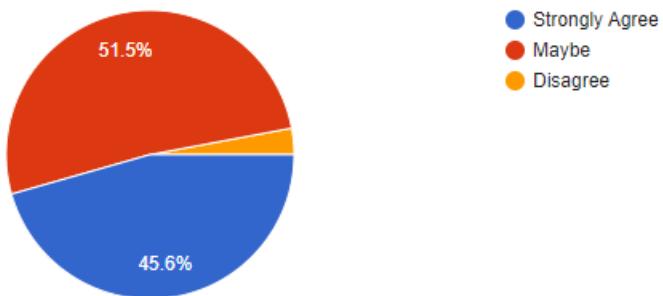


Figure 21: Response 8.

### ix. Response 9

How often do you get notified about any new deals offered near you or at other location?

 Copy

68 responses

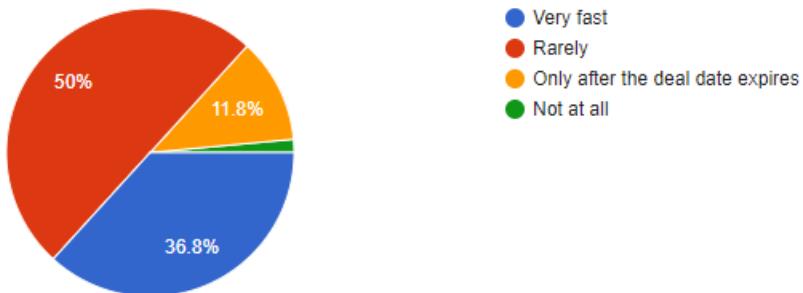


Figure 22: Response 9.

### x. Response 10

Would you like to buy and share all types of deals for your family and friends to enjoy too?

 Copy

68 responses

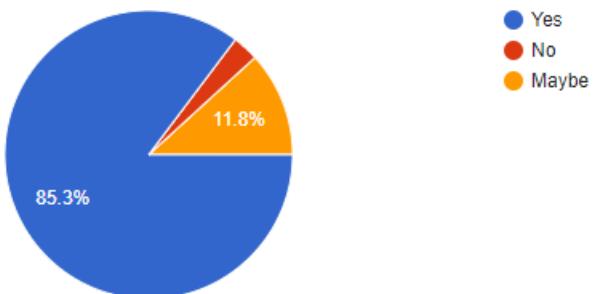


Figure 23: Response 10.

### xi. Response 11

Do you think this application will help in growth of businesses by the offers/deals they provide even to their unreachable customers through this application?

 Copy

68 responses

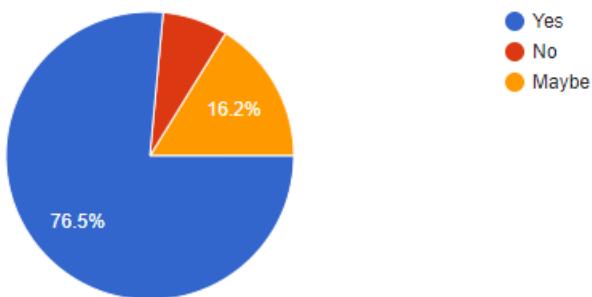


Figure 24: Response 11.

#### 3.4.1.1 Pre-Survey Analysis

As the pre-survey was completed, all the responses in each question were reviewed and the project was developed further accordingly. First respondents were asked about their interest in deals and deals related application to get the overview of client's problem related to deals. As number of them responded that they search for discounts daily so that was the main focus of this application to contain every type of deal and customer doesn't even have to work to find deals. To make the experience of the customers better lot of features like sharing coupon, near me recommendation, search bar etc. were planned. As every type of deals are valid in this application so respondents were as about the most interested deals so the deals in the application could be categories according to that. As people were not get notify or have knowledge about deals near them and to make it possible, near me recommendation system was also main focus of the application.

### 3.4.2 Post Survey Result

The [post-survey](#) was done to obtain feedback from our users and acquire a better idea of how we can enhance the system in the future. The post-survey also indicated areas where the system might be improved in the future. Several users, for example, proposed that we enhance the search capabilities and deal structure, and we are actively working to solve these issues. We are devoted to listening to our users and improving "KarKhana" to ensure that it fulfills their requirements and expectations.

#### I. Response 1

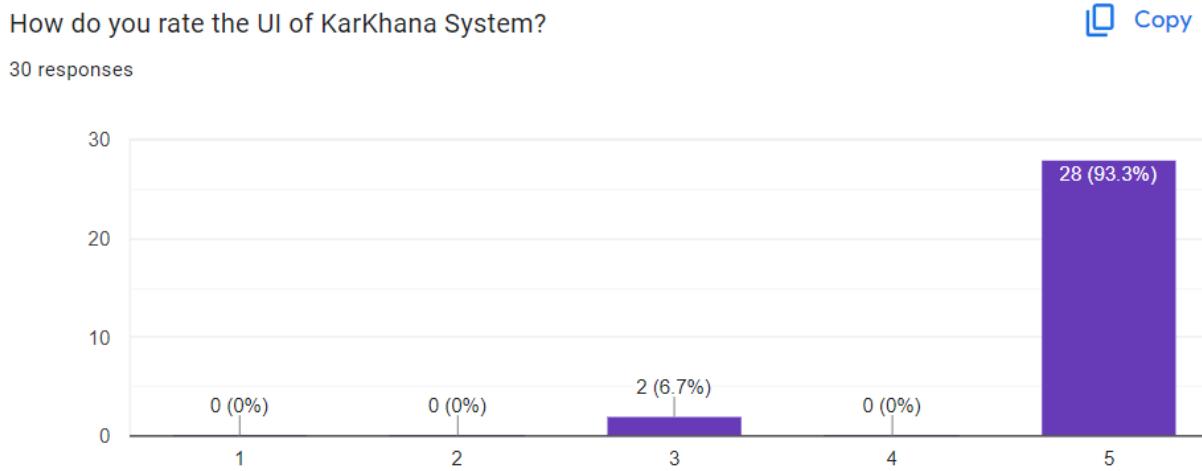


Figure 25: post-Survey response 1.

## II. Response 2

How satisfied were you with the Khalti payment option for buying deal coupon?

 Copy

30 responses

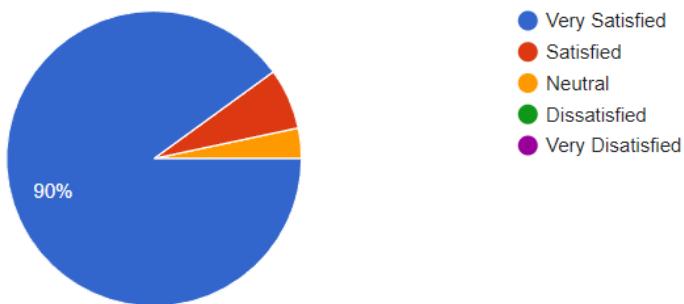


Figure 26: post-Survey response 2.

## III. Response 3

Did you find Coupon Sharing feature helpful to share deals with your friends and families?

 Copy

30 responses

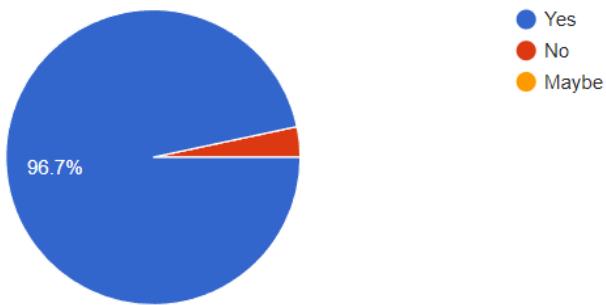


Figure 27: post-Survey response 3.

#### IV. Response 4

Did you find Add to cart feature helpful?

 Copy

30 responses

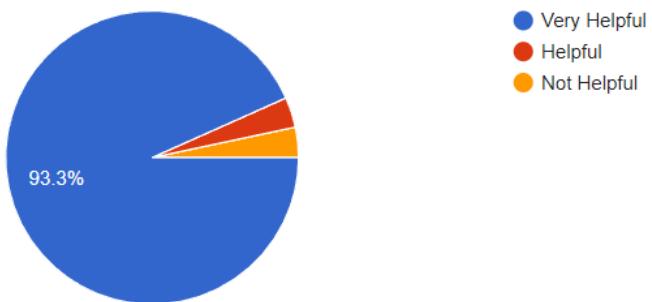


Figure 28: post-Survey response 4.

#### V. Response 5

How easy was it to search deals on KarKhana?

 Copy

30 responses

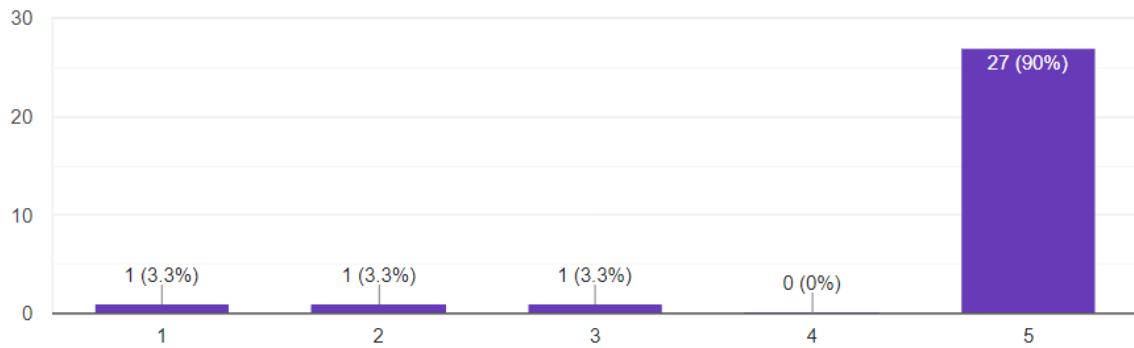


Figure 29: post-Survey response 5.

## VI. Response 6

How easy is to post deals on KarKhana?

 Copy

30 responses

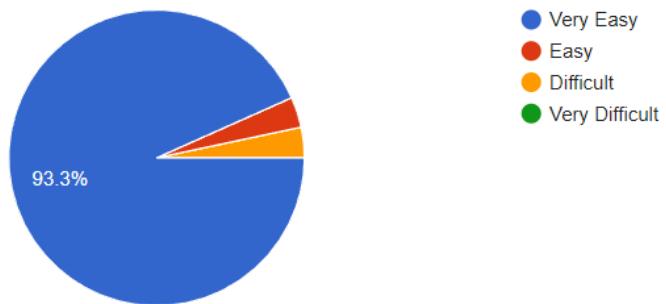


Figure 30: post-Survey response 6.

## VII. Response 7

How satisfied are you with the KarKhana Mobile application?

 Copy

30 responses

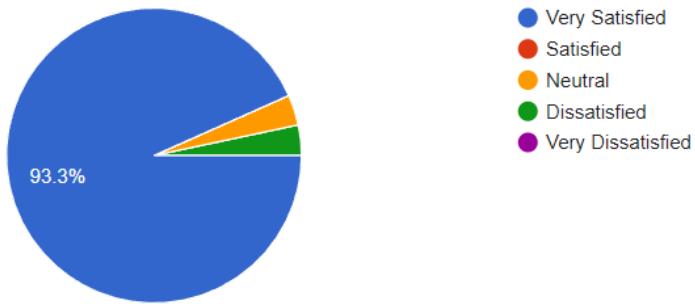


Figure 31: post-Survey response 7.

### VIII. Response 8

Would you Recommend our Deal purchasing platform to others?

 Copy

30 responses

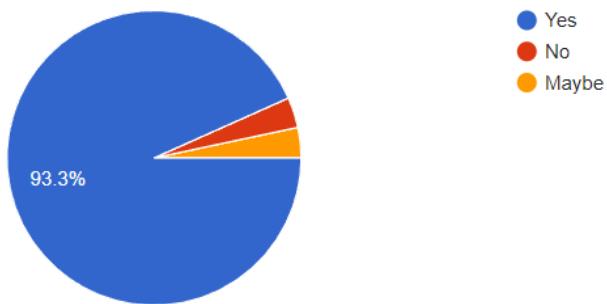


Figure 32: post-Survey response 8.

#### 3.4.2.1 Post Survey Analysis

As the post-survey was completed, all the responses in each question were reviewed and the project had made updates ready for future accordingly. UI of the application was displayed to respondents and which most of them loved it. Khalti was implemented for online payment process, everyone had a great experience with it. Some features like sharing of coupons, add to cart, search bar etc. were found very helpful by the customers. Also, vendors can add deals in the application which was very easy to use according to the responses from the survey. From some of the feedback, future updates were determined like features of chat (messaging), review and ratings, enhanced location-based recommendation system etc.

The post-survey form for my system is described in detail in appendix section. Click here for more on post-survey form: post-Survey.

---

### 3.5 Requirement Analysis

---

#### 3.5.1 FUNCTIONAL REQUIREMENTS

---

##### 3.5.1.1 REGISTRATION

Req. ID	Requirement Description	
FR.01	Sys. Req.	System Requirement
	ID	
	SR.01	Users can view their respective register form.
	SR.02	The system will determine whether or not the required section was entered.
	SR.03	Authentication of the entered details of the user will be verified by the system
	SR.04	For customer, a token is sent by the system to the mail provided by the customer while registering to verify the email. For vendor, entered details are sent to admin for further process
	SR.05	the system will display a message depending on whether the user registration was successful or not,

Table 6: Functional Requirements of Register.

### 3.5.1.2 Login

Req. ID	Requirement Description	
	Sys. Req.	System Requirement
	ID	
FR.02	Both types of users can login into this KarKhana application	
	<b>SR.01</b>	Users can view their respective login form.
	<b>SR.02</b>	The system will validate whether or not the required section was entered.
	<b>SR.03</b>	System must check if the provided details are valid or not.
	<b>SR.04</b>	the system will display a message depending on whether the user registration was successful or not.
	<b>SR.05</b>	After login success customer should be navigated to set location page and vendor should navigate straight to homepage.

Table 7: Functional Requirements of Login.

### 3.5.1.3 Add Deal to Cart

Req. ID	Requirement Description	
FR.03	Sys.	System Requirement
	Req. ID	
	<b>SR.01</b>	Customer can browse and select desired deal.
	<b>SR.02</b>	After customer confirms to add the deal to cart, System should display a success message
	<b>SR.03</b>	System must display the added deal into the respective customer's cart.

Table 8: Functional Requirements of Add deals to cart

### 3.5.1.4 Buy Coupons

Req. ID	Requirement Description	
	Sys. Req. ID	System Requirement
FR.04	Customer can buy the coupon of the deal added to their cart.	
	SR.01	Customer can view the deals added to cart in their cart page.
	SR.02	When the customer selects the deal from cart and proceeds to buy then system generates and displays the coupon details.
	SR.03	System calculates the coupon rate and also displays that in the coupon details
	SR.04	If customer clicks on buy option, then Khalti's interface is displayed by the system.
	SR.05	After successful Khalti procedure, system displays the success message that the coupon purchase is successful.
	SR.06	Along with success message, system also sends email notification about the successful purchase of the coupon to the customer.

Table 9: Functional Requirements of Buy Coupons.

### 3.5.1.5 User Feedback

Req. ID	Requirement Description	
FR.05	Customer can send feedback to the system	
Sys. Req. ID	System Requirement	
<b>SR.01</b>	Customer can view the feedback page.	
<b>SR.02</b>	System checks the field authentication if customer is trying to send empty message or not.	
<b>SR.03</b>	When customer clicks the send button, system gets the email, customer id and customer name of the logged in customer.	
<b>SR.04</b>	System stores the feedback along with customer details in the database for admin to review.	

Table 10: Functional Requirements of User Feedback.

### 3.5.1.6 Payment

Req. ID	Requirement Description	
FR.06	<b>Sys. Req. ID</b> <b>System Requirement</b>	
	<b>SR.01</b>	Customer can pay via Khalti.
	<b>SR.02</b>	System displays the Khalti interface to customer and proceed for further payment procedure.
	<b>SR.03</b>	After Khalti procedure if it's a success, then success message must be displayed
	<b>SR.04</b>	In case of not successful payment, system displays the failed message.

Table 11: Functional Requirements of Payment.

### 3.5.1.7 Search

Req. ID	Requirement Description	
	Sys. Req. ID	System Requirement
<b>FR.07</b>		Customer can search for their desired deals and their purchased coupon. Vendor can also search for their coupon purchased by the customers.
	<b>SR.01</b>	Customer can search for their desired deals by the deal title and also can search for their purchased coupon by the coupon id in purchases page.
	<b>SR.02</b>	Vendor can also search for the coupon of their business bought by the customers.
	<b>SR.03</b>	The system will check if the provided deal title or coupon Id is available or not.
	<b>SR.04</b>	In case of searched data not available, “no data found” message should be displayed by the system.
	<b>SR.05</b>	If the deal with the provided title or coupon with the provided id is available then the system should display all the matching deals or coupons to the customer

Table 12: Functional Requirements of Search.

### 3.5.1.8 Coupon Status

Req. ID	Requirement Description	
FR.08	Sys. Req.	System Requirement
	ID	
	<b>SR.01</b>	After purchasing a coupon, system must display it as Active coupon in active section of the purchases page.
	<b>SR.02</b>	Along with that, system also updates respective vendor side and displays the purchased coupon of their respective deals.
	<b>SR.03</b>	When the customer wants to redeem the coupon, vendor confirms the coupon then again system updates the status of the coupon in both side of the user.
	<b>SR.04</b>	After redemption success, system sends email notification to user about the redemption.
	<b>SR.05</b>	Now the system displays the redeemed coupon in previous coupon section of purchases page of both user side.

Table 13: Functional Requirements of Coupon Status.

### 3.5.1.9 Add Deals

Req. ID	Requirement Description	
FR.09	Sys. Req. ID	System Requirement
	<b>SR.01</b>	Vendor should enter deals detail in add deal page.
	<b>SR.02</b>	System validates the fields whether valid details are entered or not.
	<b>SR.03</b>	System checks if the vendor has uploaded their business profile or not.
	<b>SR.04</b>	If profile is uploaded then system should post the deal to the application which is then displayed to customers.
	<b>SR.05</b>	If profile is not uploaded then the system should ask the vendor to upload the profile first.

Table 14: Functional Requirements of Add Deals.

### 3.5.2 Non-Functional Requirements

#### 3.5.2.1 Performance Requirements

Req. ID	Requirement Description	Priority
<b>PR.01</b>	It should take at most of 5 to 10 seconds to launch the application.	Could
<b>PR.02</b>	Before everything, if user is log in then homepage should be displayed else welcome screen should be display which ask if user want to login as customer or vendor.	Should
<b>PR.03</b>	This application should display welcome page, if the user is not logged in.	Should
<b>PR.04</b>	This application should display the location page for customer and homepage for vendor at fist after successfully log in.	Should
<b>PR.05</b>	All the features will be functional on the android application for all the users	Should

Table 15: Performance Requirements.

### 3.5.2.2 Safety Requirements

Req. ID	Requirement Description	Priority
<b>SR.01</b>	Permission is asked by the application when it needs to track the user's current location.	Should
<b>SR.02</b>	There are no advertisements of anything or anyone displayed in this application	Should
<b>SR.03</b>	Non-requirement software should not be installed without the access of information about the user.	Should
<b>SR.04</b>	Application shall be made free from any attacks	Could
<b>SR.05</b>	Once used coupon cannot be re-used, application should recognize and update the coupon status after every redemption.	Should

Table 16: Safety Requirements.

### 3.6 DESIGN

---

Design is an essential component of software development since it lays the framework for the entire undertaking. A thorough and detailed strategy for the software's architecture and functioning must be made before development can begin. This will make it easier for the development team to comprehend the project's needs, features, and procedures, which will make it more efficient. UML diagrams may be used to produce a high-level design for the entire project as well as specific designs for each of its aspects. This strategy makes sure that everyone working on the project is on the same page regarding its goals and intended audience, which eventually results in a good output.

#### 3.6.1 Logo of the Application

---



Figure 33: KarKhana Logo.

### 3.6.2 System Design

#### 3.6.2.1 Use Case Diagram

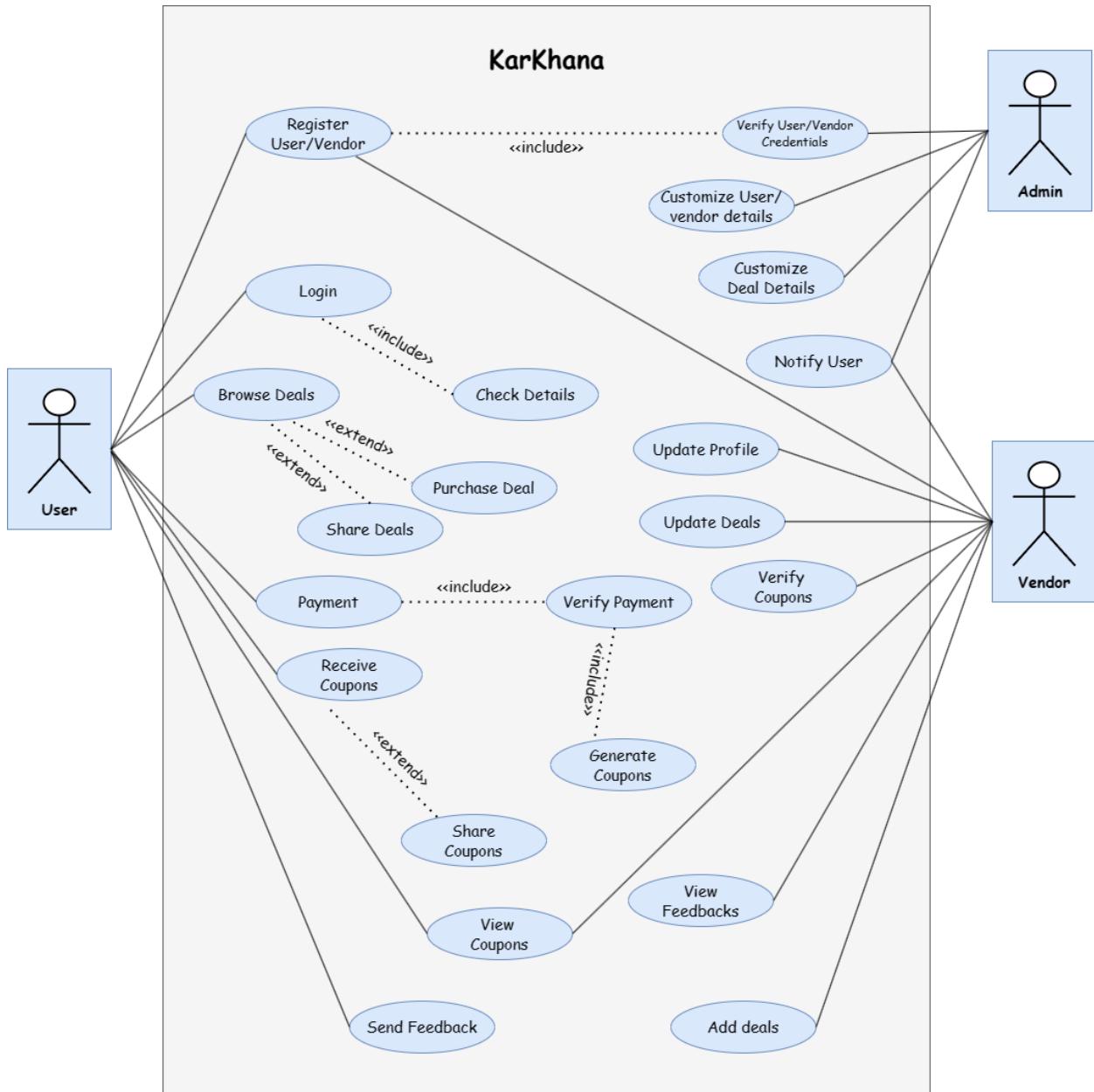


Figure 34: Use Case Diagram.

### 3.6.2.2 ER-Diagram

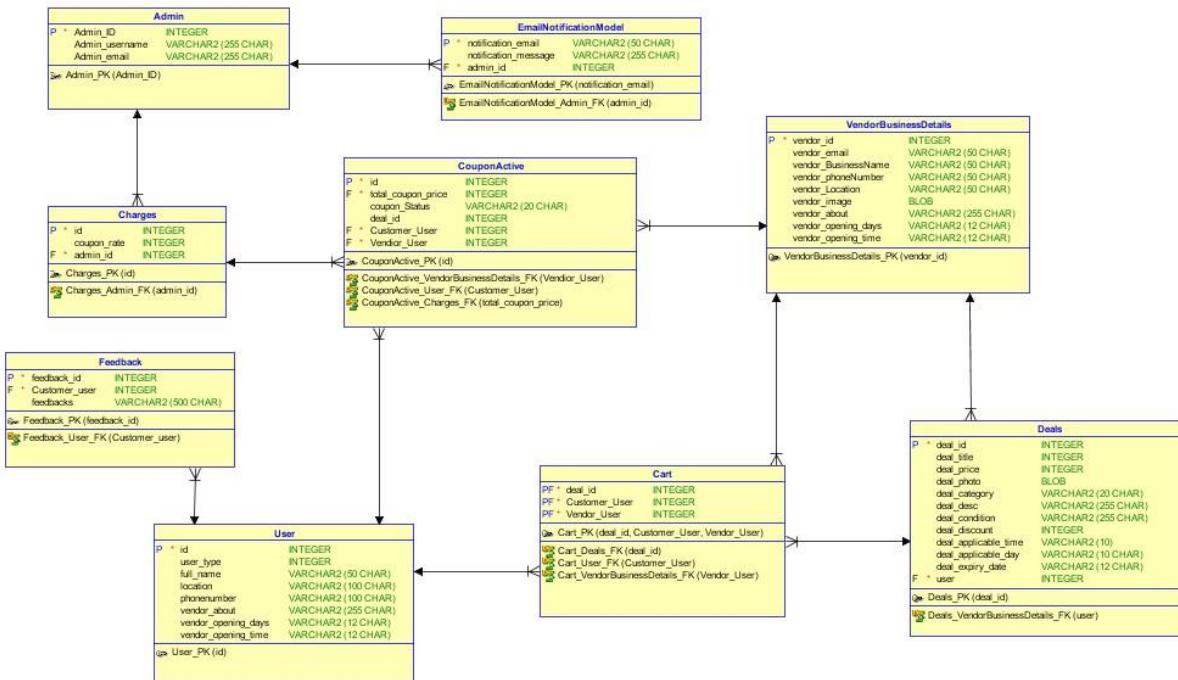


Figure 35: Overall ERD of the System.

### 3.6.2.3 Activity Diagram

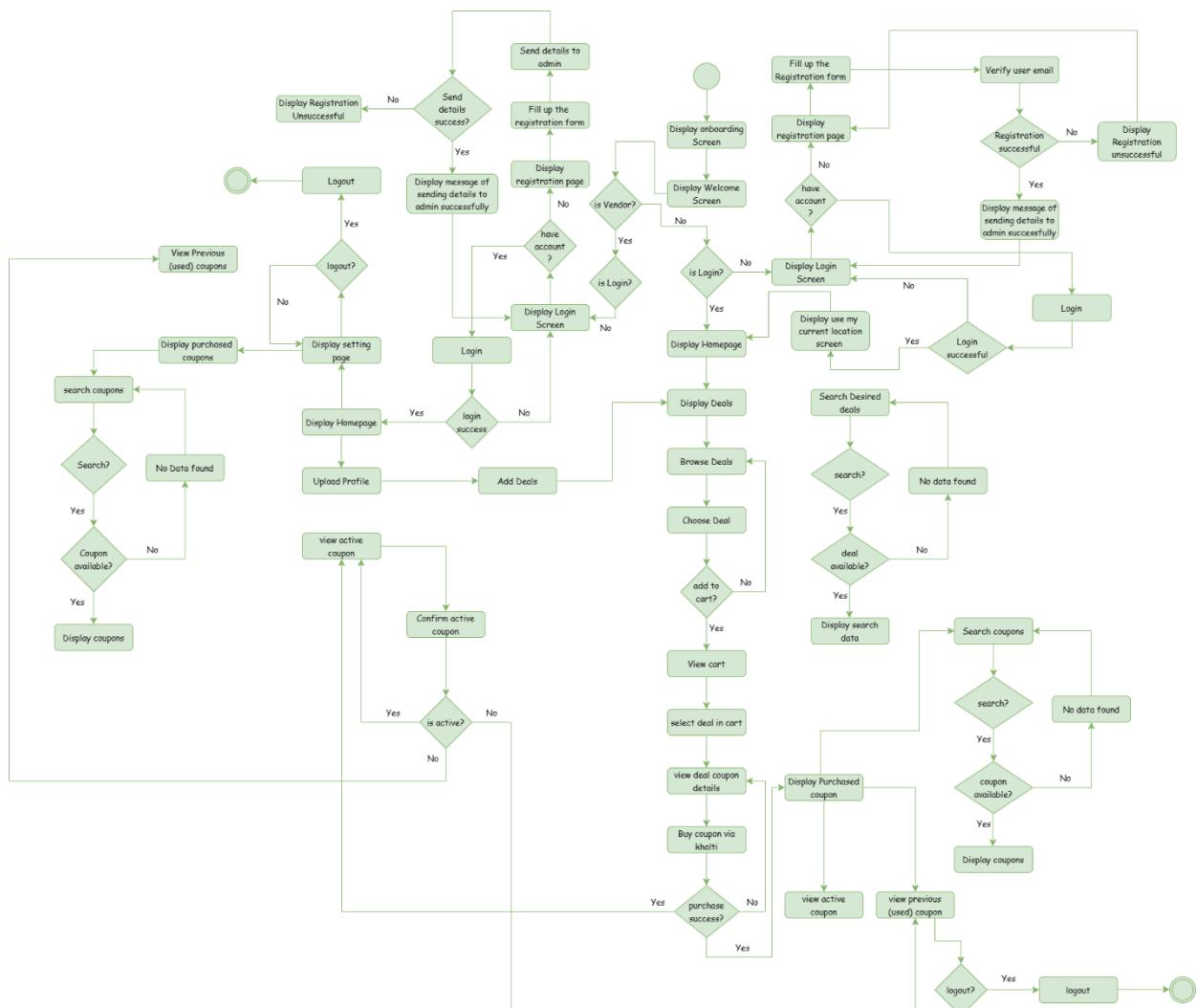


Figure 36: Overall Activity Diagram of the system.

### 3.6.2.4 Sequence Diagram

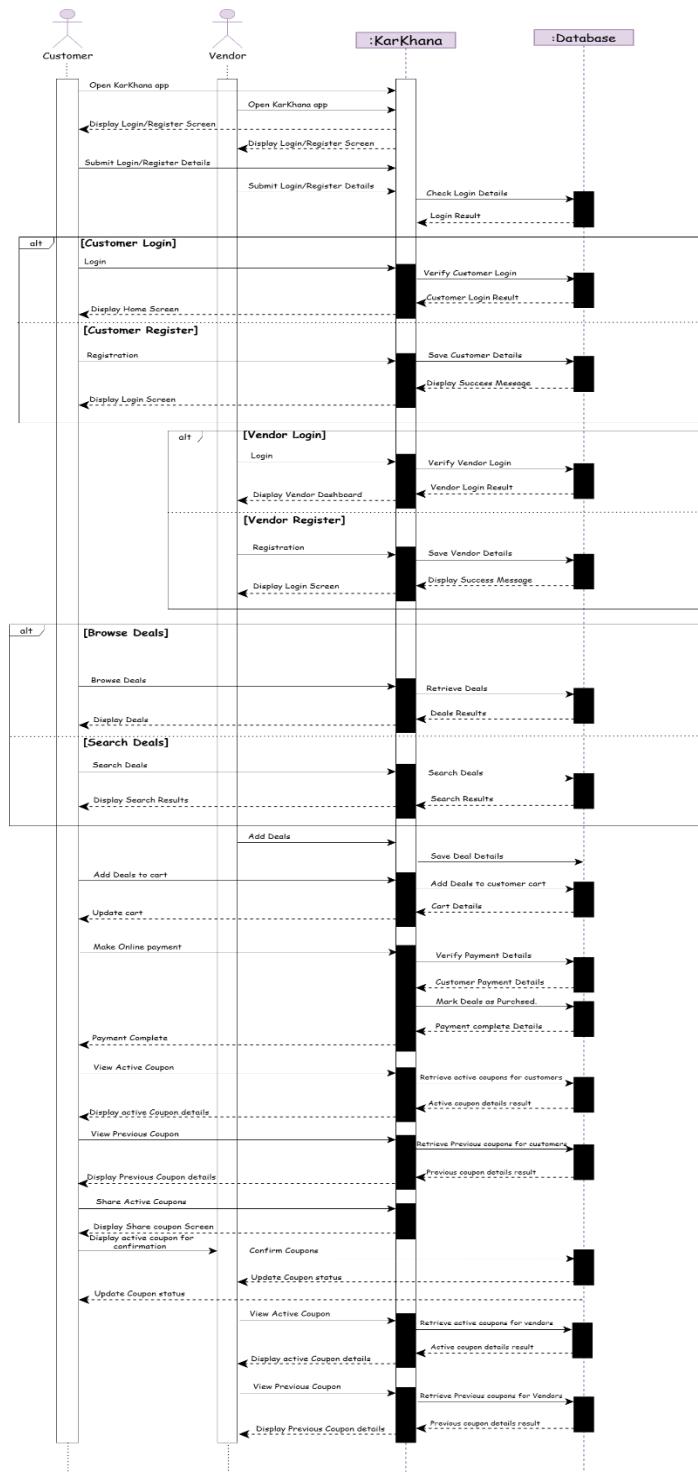
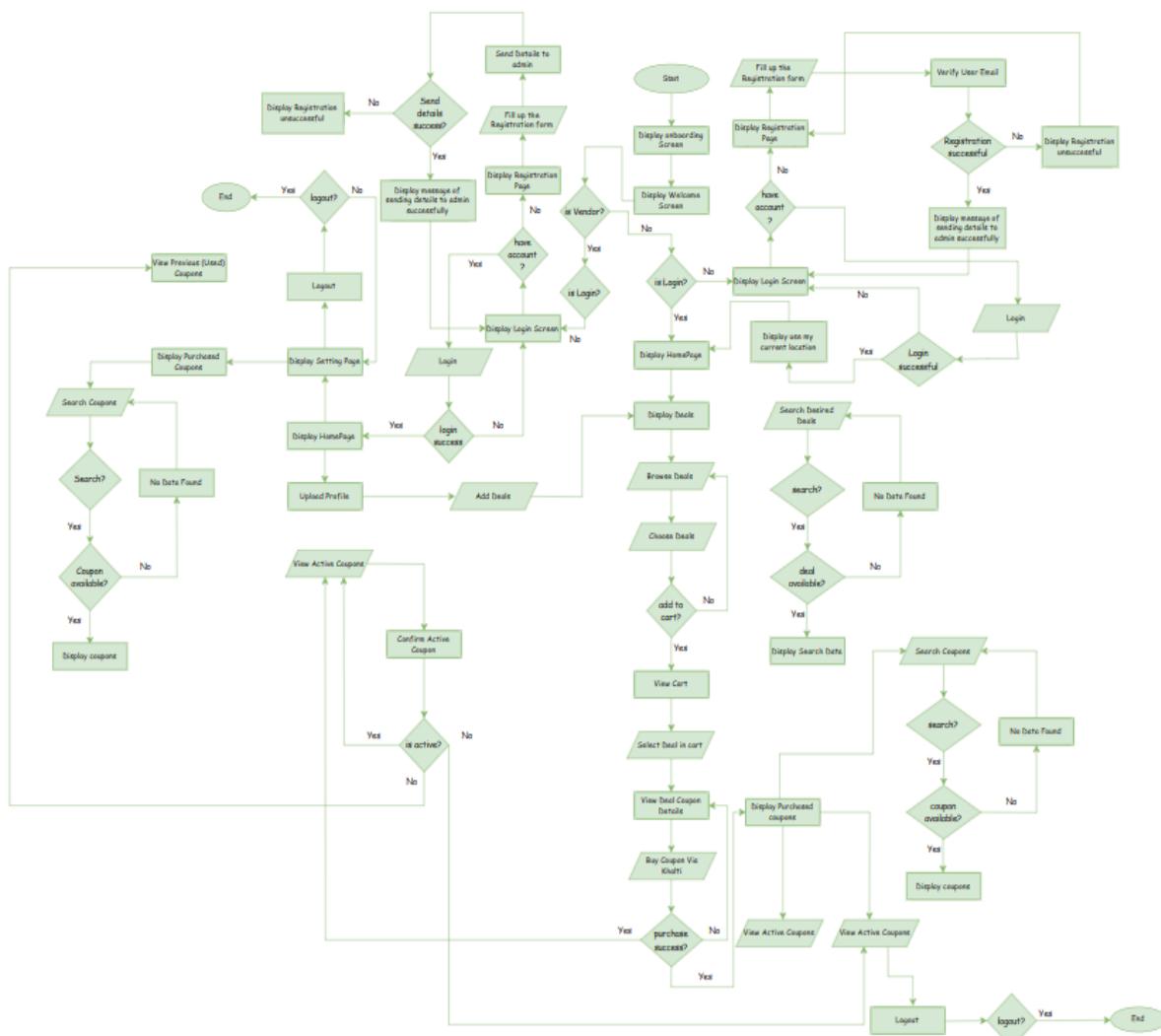


Figure 37: Overall Sequence diagram of the system.

### **3.6.2.5 Flowchart**



*Figure 38: Overall flowchart of the system.*

### 3.6.2.6 System Architecture Diagram

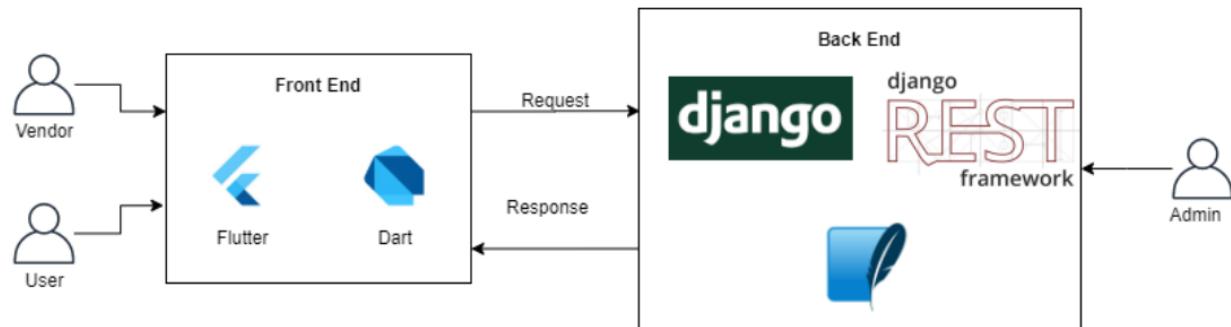


Figure 39: System Architecture Diagram of the system.

### 3.6.2.7 System Overview Diagram

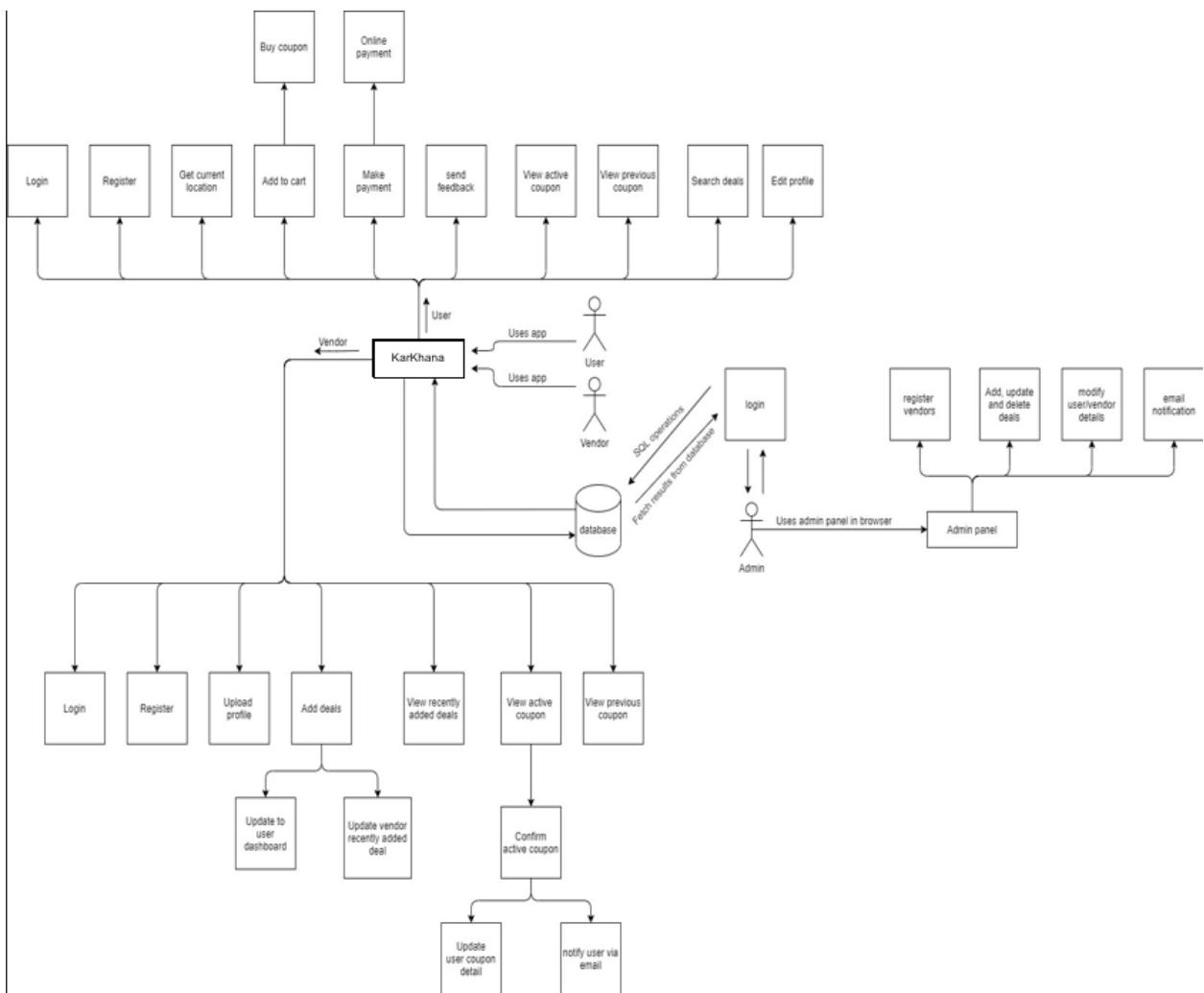


Figure 40: System Overview Diagram of the system.

### 3.6.3 Wireframe

The designs you received are called wireframes (sometimes called wires, mockups, or mocks). A wireframe is a schematic or blueprint that may assist you, your programmers, and designers in thinking about and communicating about the structure of the software or website you're developing. The same screen can be designed in a variety of ways, but only a handful of them will effectively convey your message and result in an easy-to-use program or website. The most crucial aspect of software design is establishing a decent interface structure. Doing this work now, before writing any code and before finalizing the visual design, can save you a lot of time and hard adjustment work later.

Some of the wireframes of the project for users are shown below:

#### 3.6.3.1 Customer

##### I. Welcome Page



Figure 41: Wireframe of Welcome Page.

## II. Customer Login Page



Figure 42: Wireframe of Customer Login Page.

### III. Customer Registration Page

The wireframe for the Customer registration page is titled "KarKhana" and "Registration". It features a large "X" icon in the center with the text "Be a member and Empower your choices" below it. The form includes fields for "Name", "Email", "Password", and "Confirm Password", each with a clear button. A "Register" button is at the bottom.

<input type="text"/>	Name	
<input type="text"/>	Email	
<input type="password"/>	Password	<input type="button" value="X"/>
<input type="password"/>	Confirm Password	<input type="button" value="X"/>
<input type="button" value="Register"/>		

Figure 43: Wireframe of Customer registration Page.

#### IV. Customer Homepage

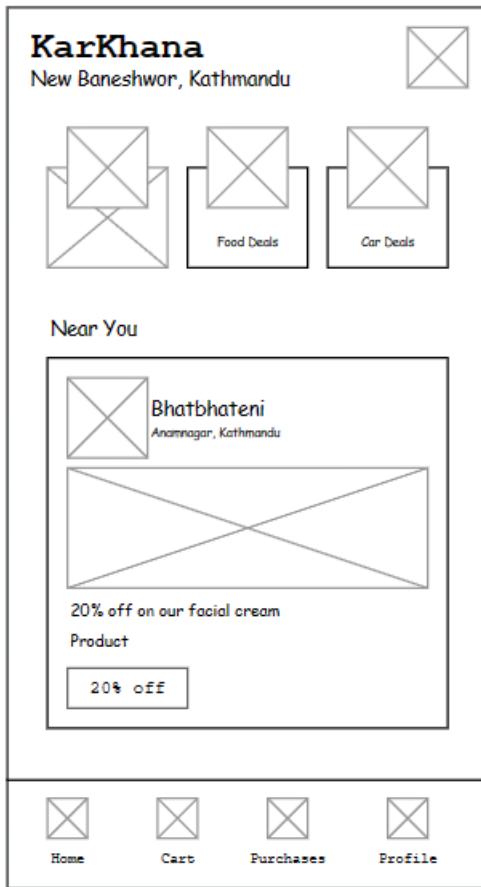


Figure 44: Wireframe of Customer Homepage.

### 3.6.3.2 Vendor

#### V. Vendor Login Page

The wireframe shows a login page for a business account. At the top, there is a logo placeholder with an 'X' icon. Below it, the brand name "KarKhana" is displayed in a large, bold, sans-serif font. A sub-instruction "Login into your Business account" is present above a large rectangular input field containing a large 'X' symbol. Below this field is the slogan "Empower Your Business with us". The main form area contains two input fields: one for "Email" and one for "Password", each with an "X" icon. To the right of the password field is a small "Forgot Password?" link. Below these fields is a large "Log in" button. At the bottom of the form, there is a link for users who don't have a business account: "Don't have business account? Register".

Figure 45: Wireframe of Vendor Login Page.

## VI. Vendor Registration Page

The wireframe shows a registration form for 'KarKhana'. At the top right is a close button (X). Below it is the brand name 'KarKhana' in bold. Underneath is the tagline 'Register Your Business'. A large rectangular input field contains a large 'X' symbol. Below this is the slogan 'Empowering Choices, Connecting Vendors'. The form consists of five input fields: 'Name', 'Email', 'Phone No.', 'Location' (with a clear button to its right), and a final 'Register' button at the bottom.

Figure 46: Wireframe of Customer Registration Page.

## VII. Vendor Homepage

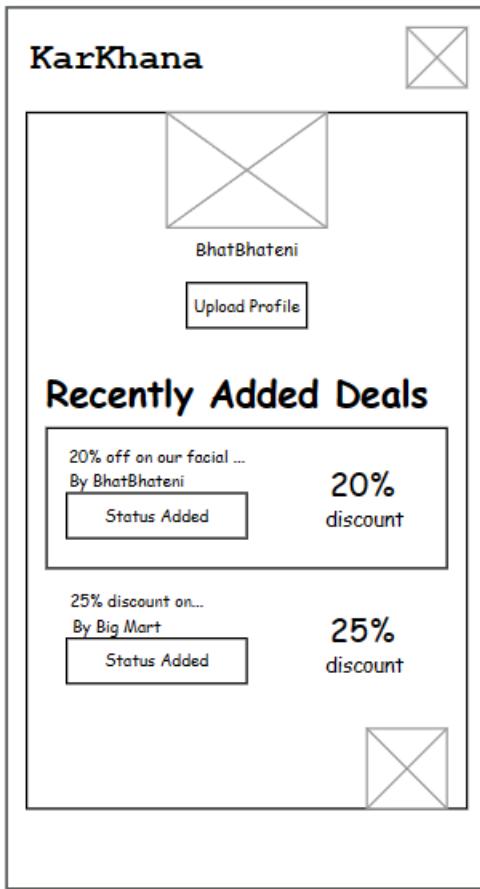


Figure 47: Wireframe of Vendor Homepage.

### 3.6.4 UI Design

#### 3.6.4.1 Customer

##### I. Welcome Page

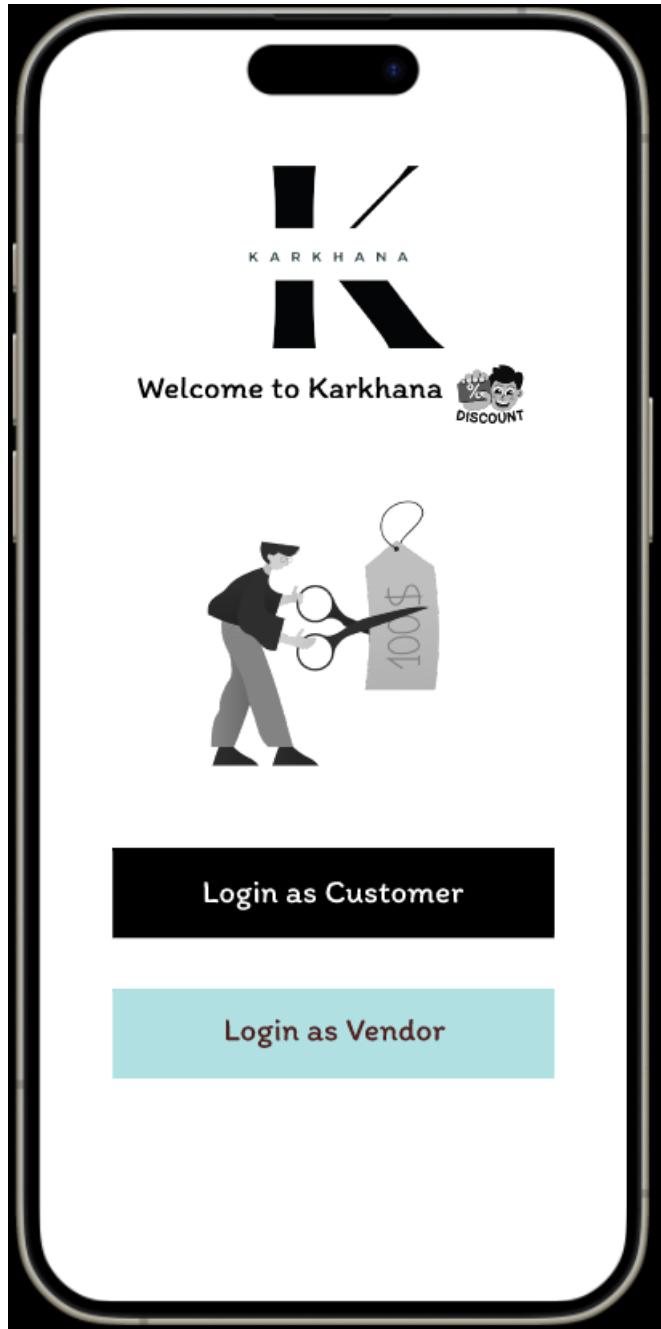


Figure 48: UI Design of Welcome page.

## II. Customer Login Page

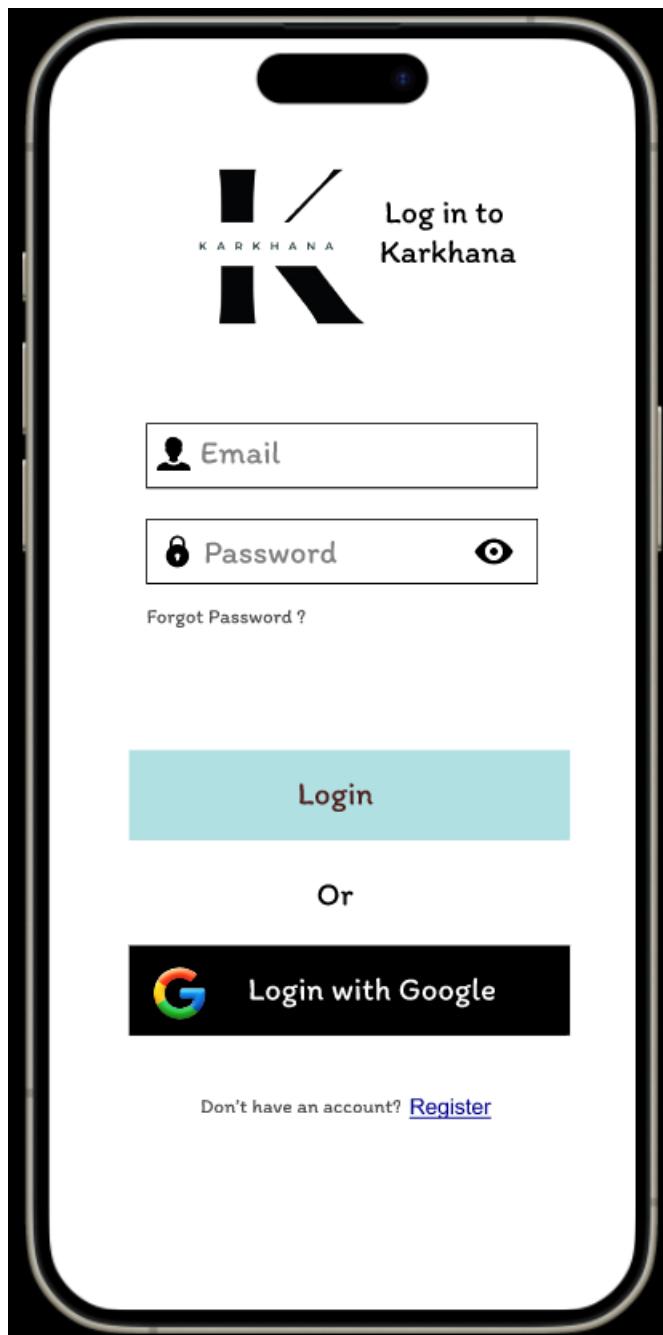


Figure 49: UI Design of Customer Login Page.

### III. Customer Registration Page



Figure 50: UI Design of Customer Registration Page.

#### IV. Customer Homepage

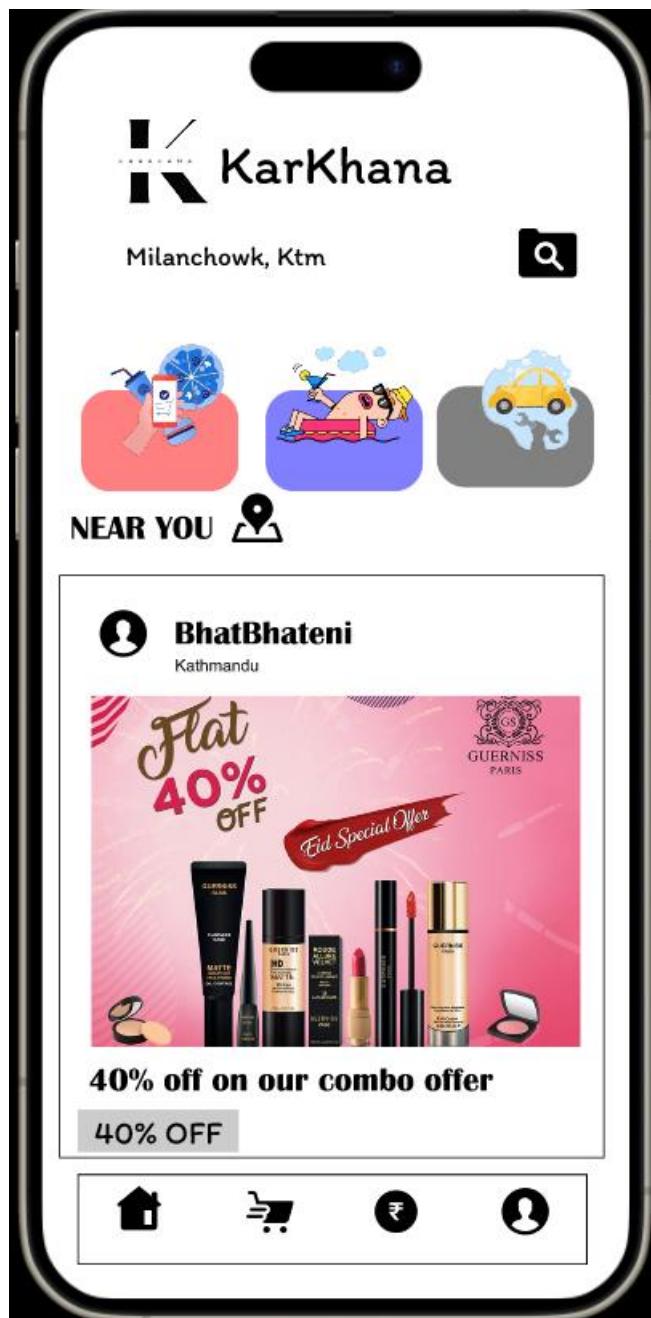


Figure 51: UI Design of Customer Homepage.

### 3.6.4.2 Vendor Pages

#### I. Vendor Login Page

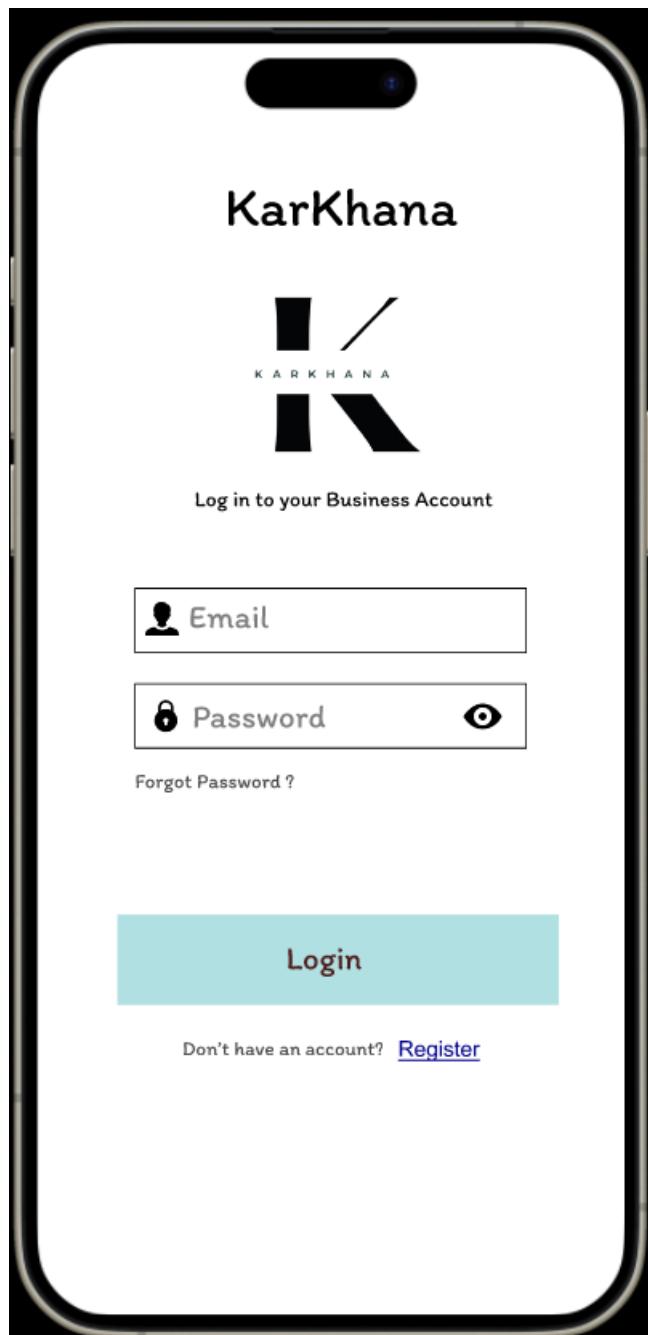


Figure 52: UI Design of Vendor Login Page.

## II. Vendor Registration Page

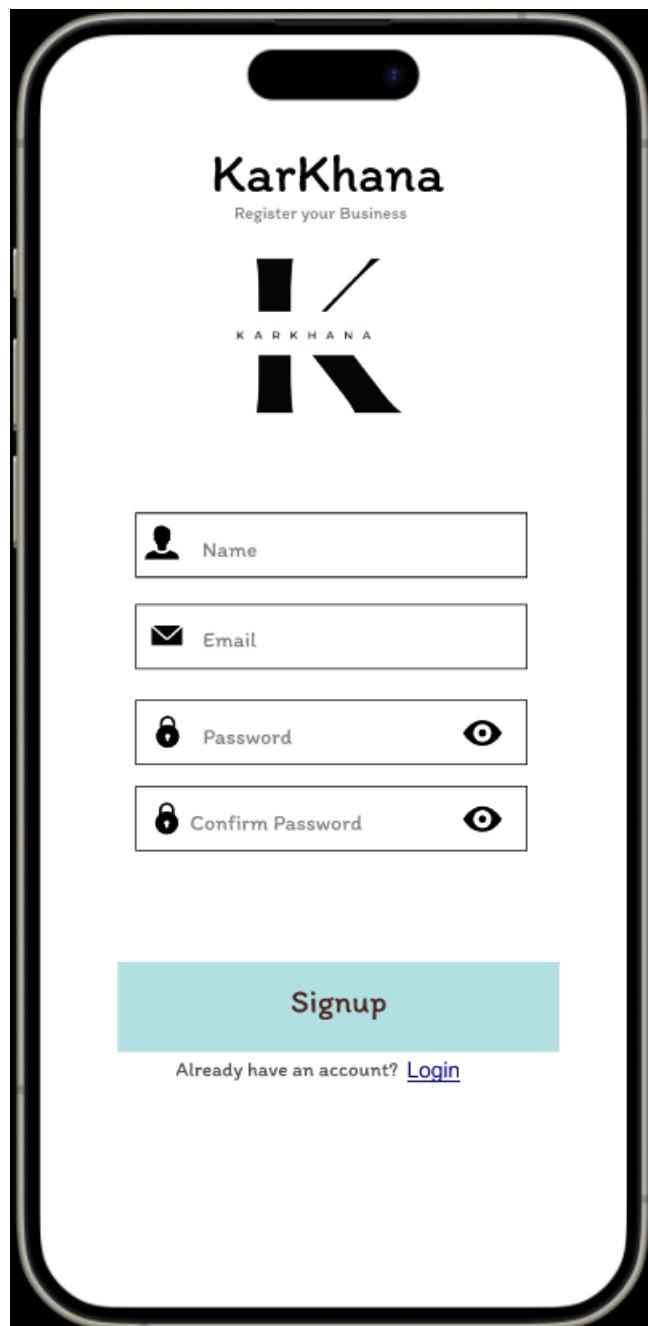


Figure 53: UI Design of Vendor Registration Page.

### III. Vendor Homepage

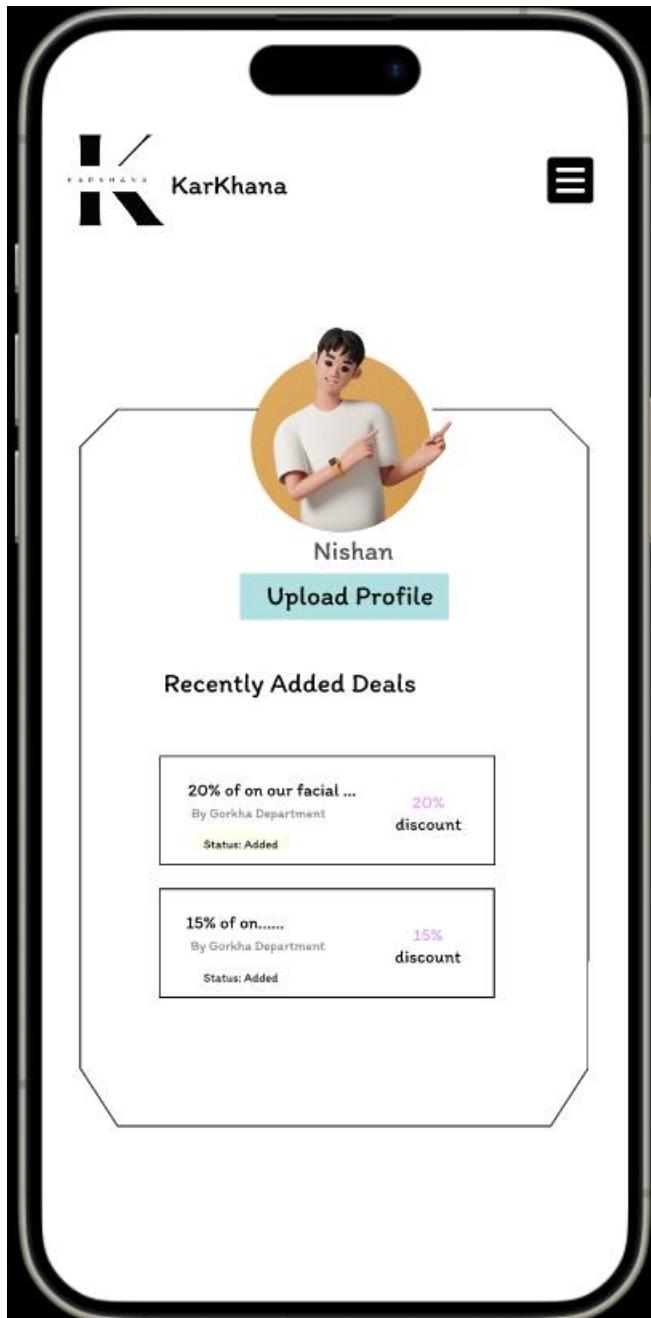


Figure 54: UI Design of Vendor Homepage.

### 3.6.5 Feature Design

#### 3.6.5.1 Sequence Diagram

##### 3.6.5.1.1 Sequence Diagram for Login

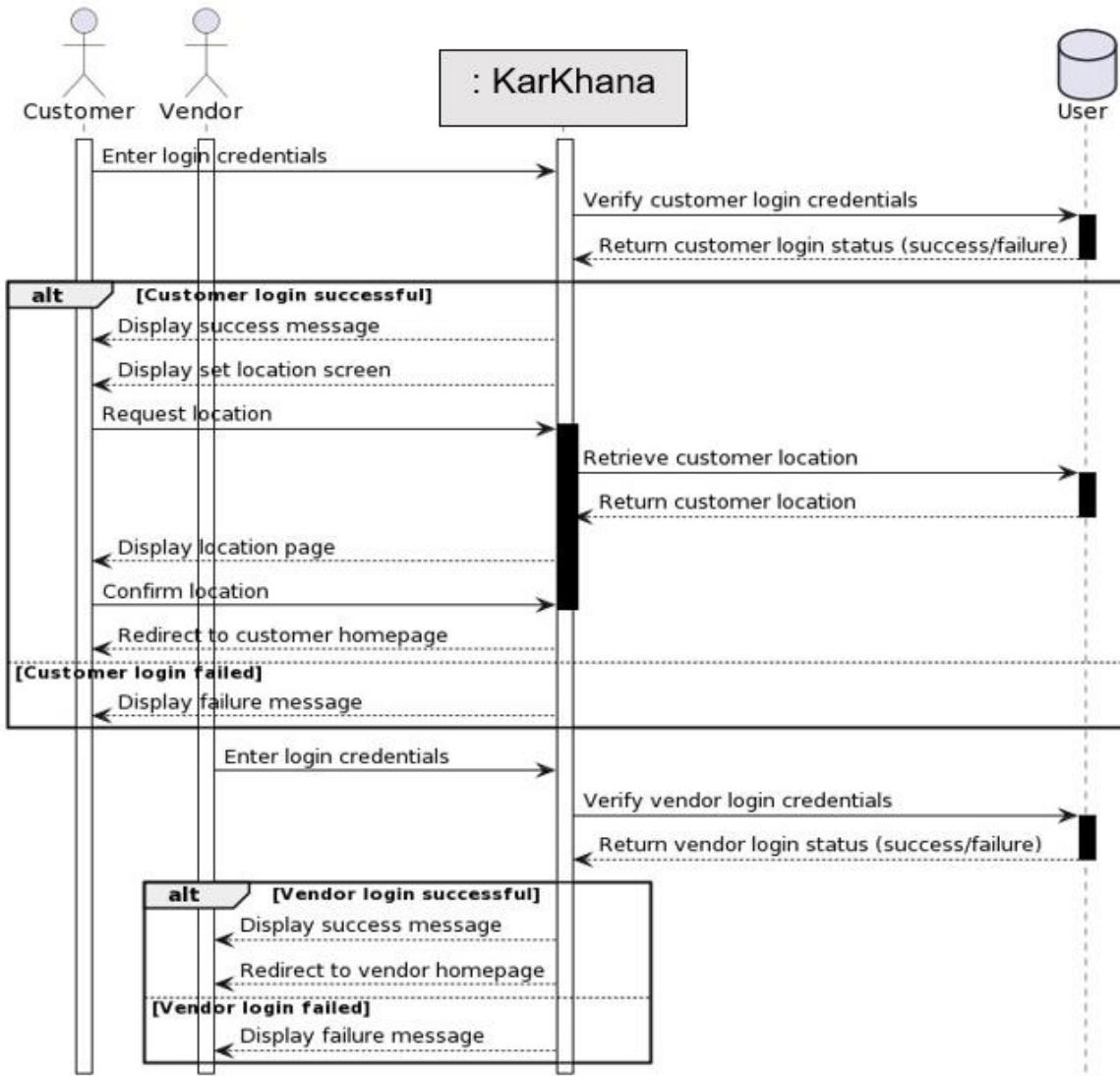


Figure 55: Sequence Diagram for Login

### 3.6.5.1.2 Sequence Diagram for Register

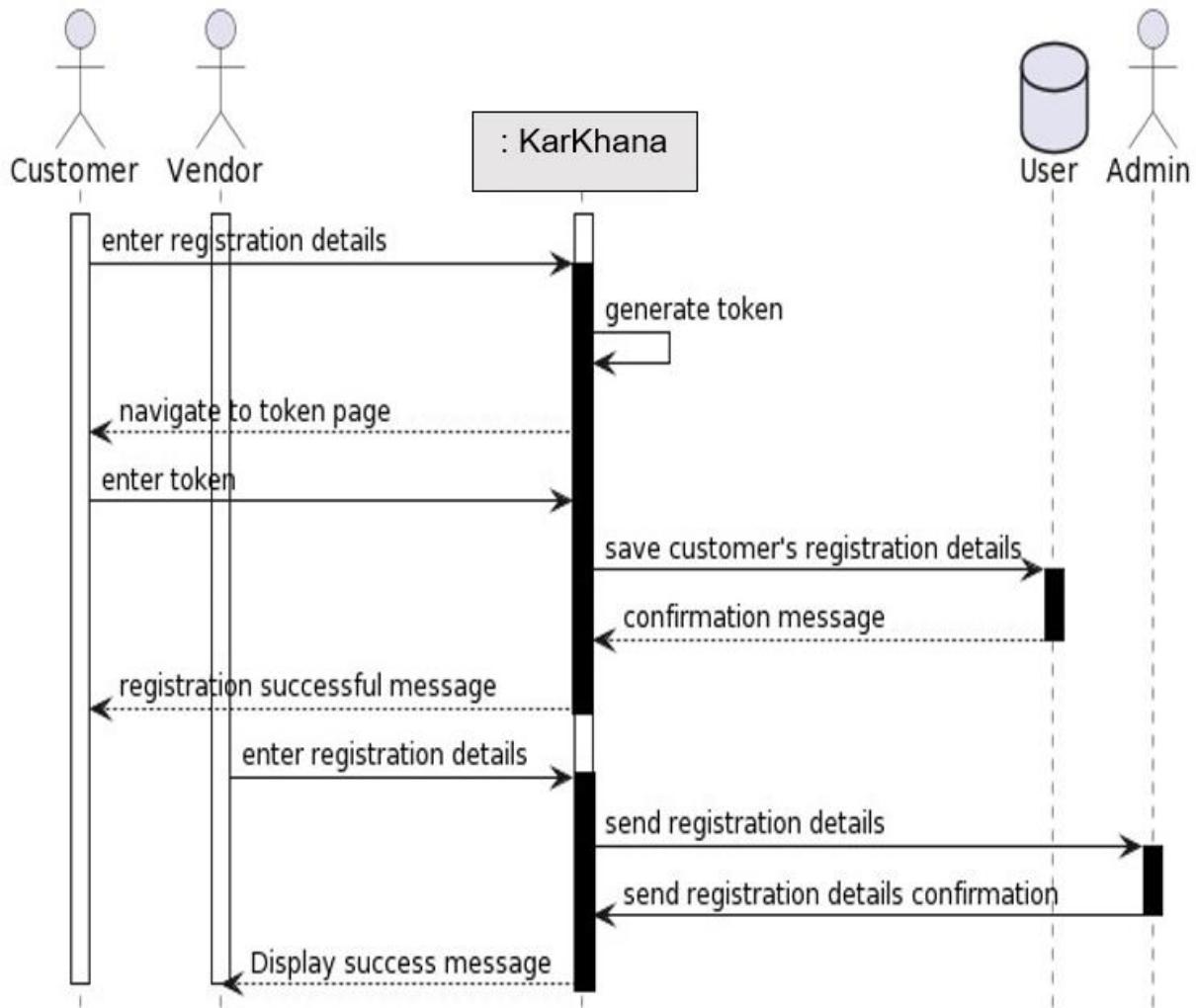


Figure 56: Sequence Diagram for Register.

### 3.6.5.1.3 Sequence Diagram for Set Current Location

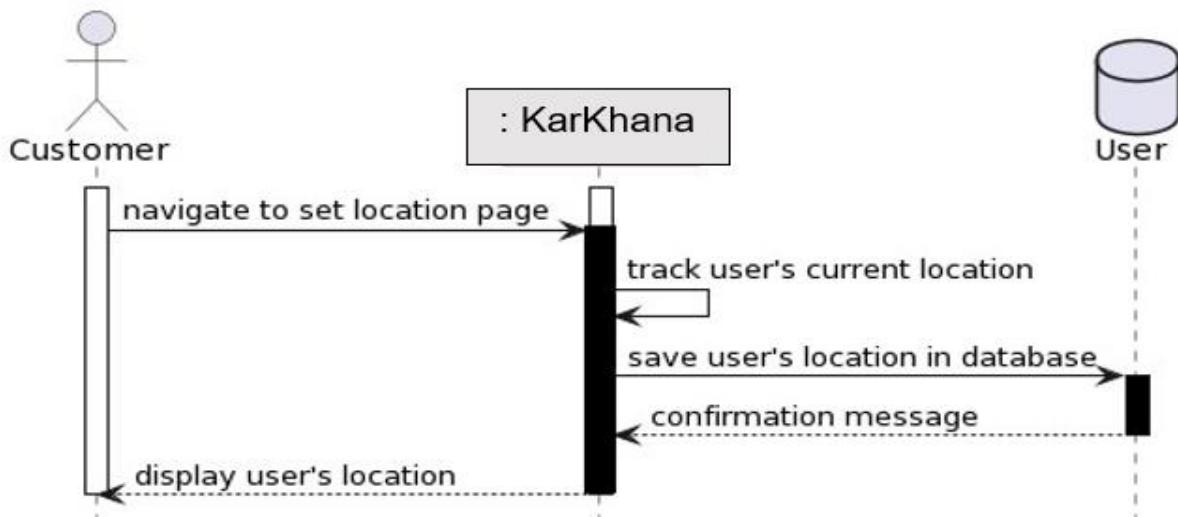


Figure 57: Sequence Diagram for Set Current Location.

### 3.6.5.1.4 Sequence Diagram for Add to Cart

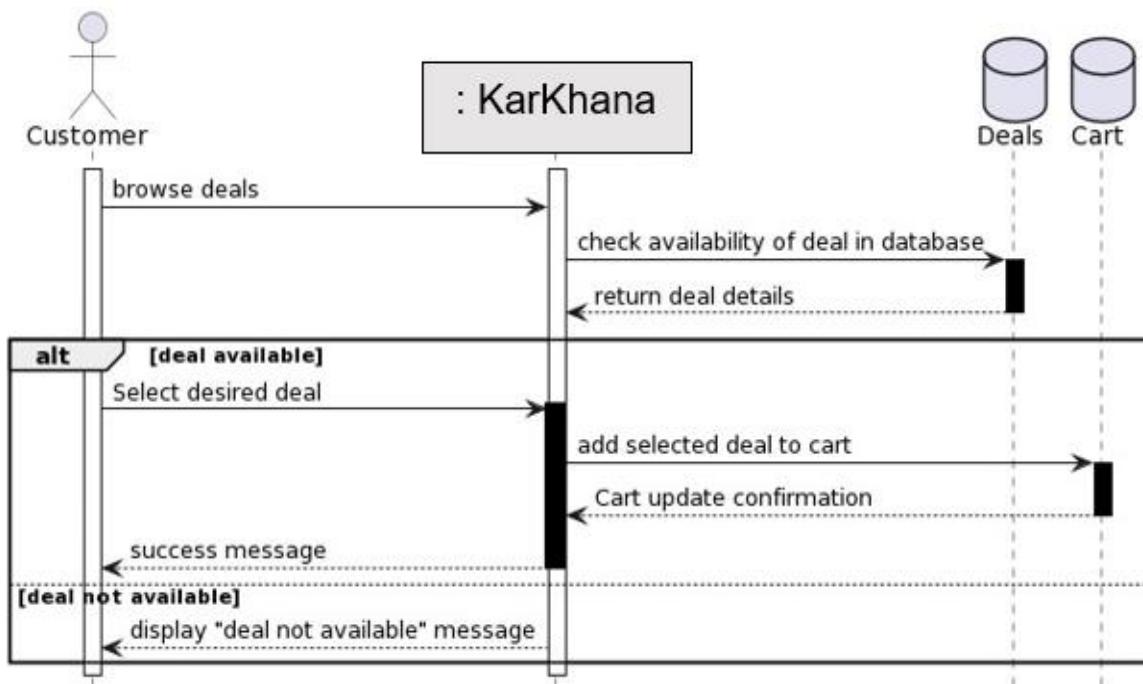


Figure 58: Sequence Diagram for Add to cart.

### 3.6.5.1.5 Sequence Diagram for Payment

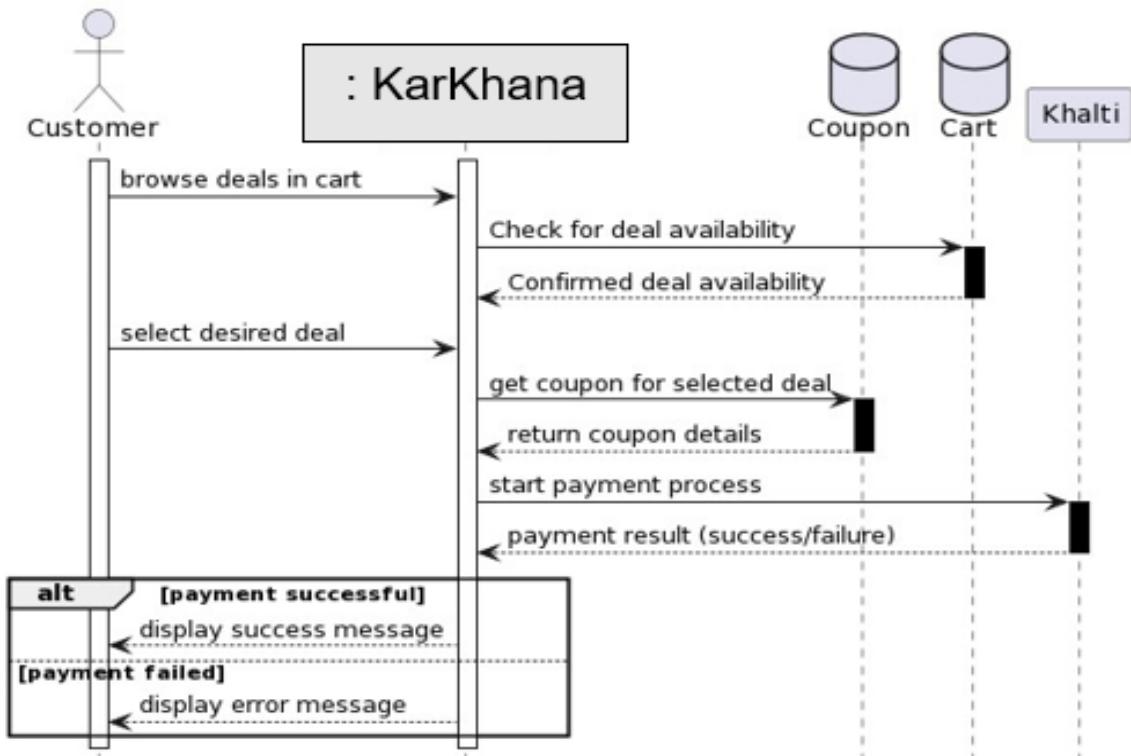


Figure 59: Sequence Diagram for Payment.

### 3.6.5.1.6 Sequence Diagram for Generate Coupon

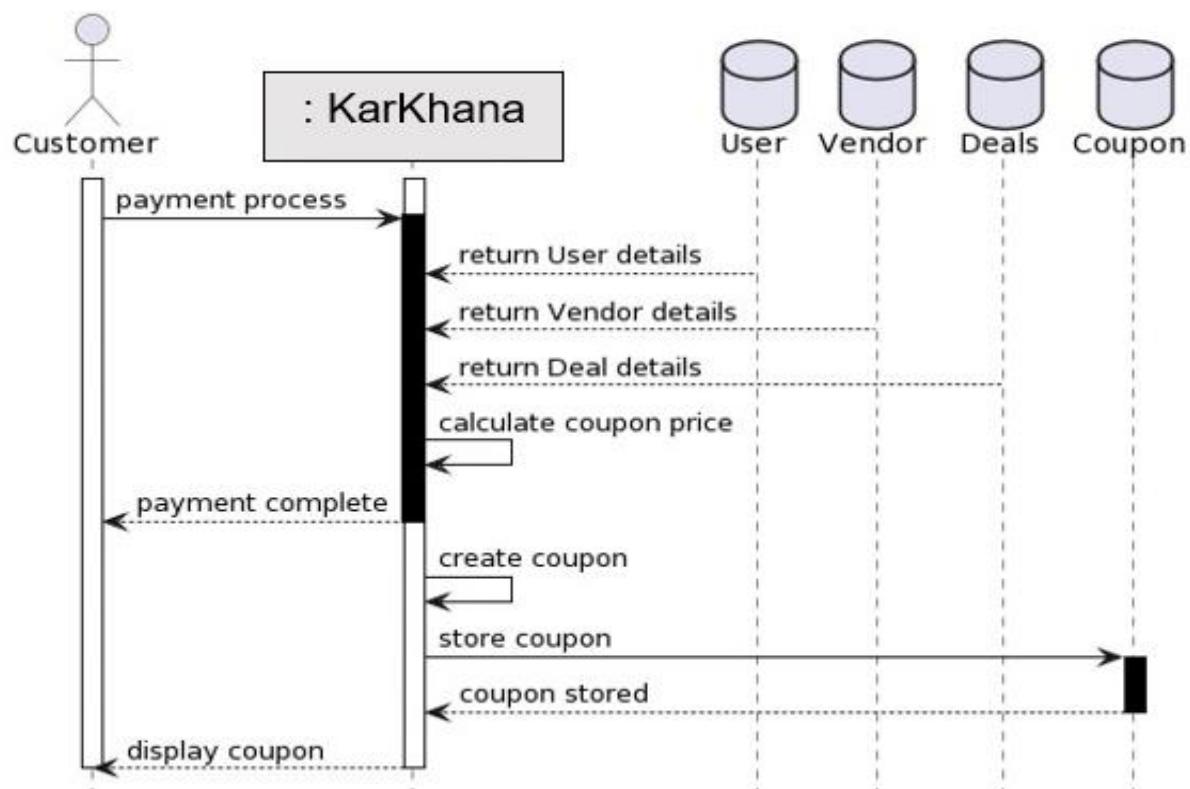


Figure 60: Sequence Diagram for Generate coupon.

### 3.6.5.1.7 Sequence Diagram for Confirm Coupon

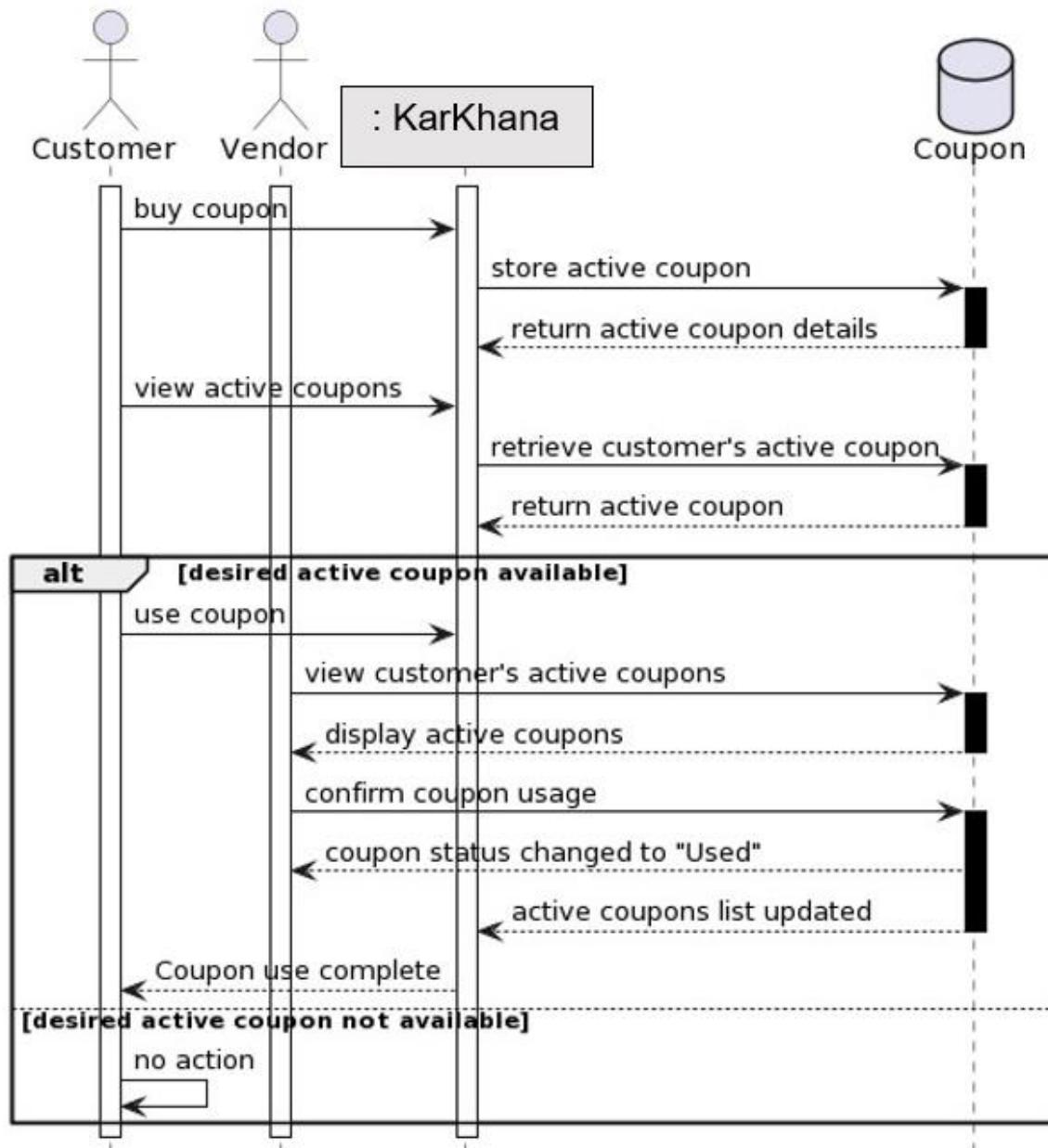


Figure 61: Sequence Diagram for Confirm Coupon.

### 3.6.5.1.8 Sequence Diagram for Share Coupon

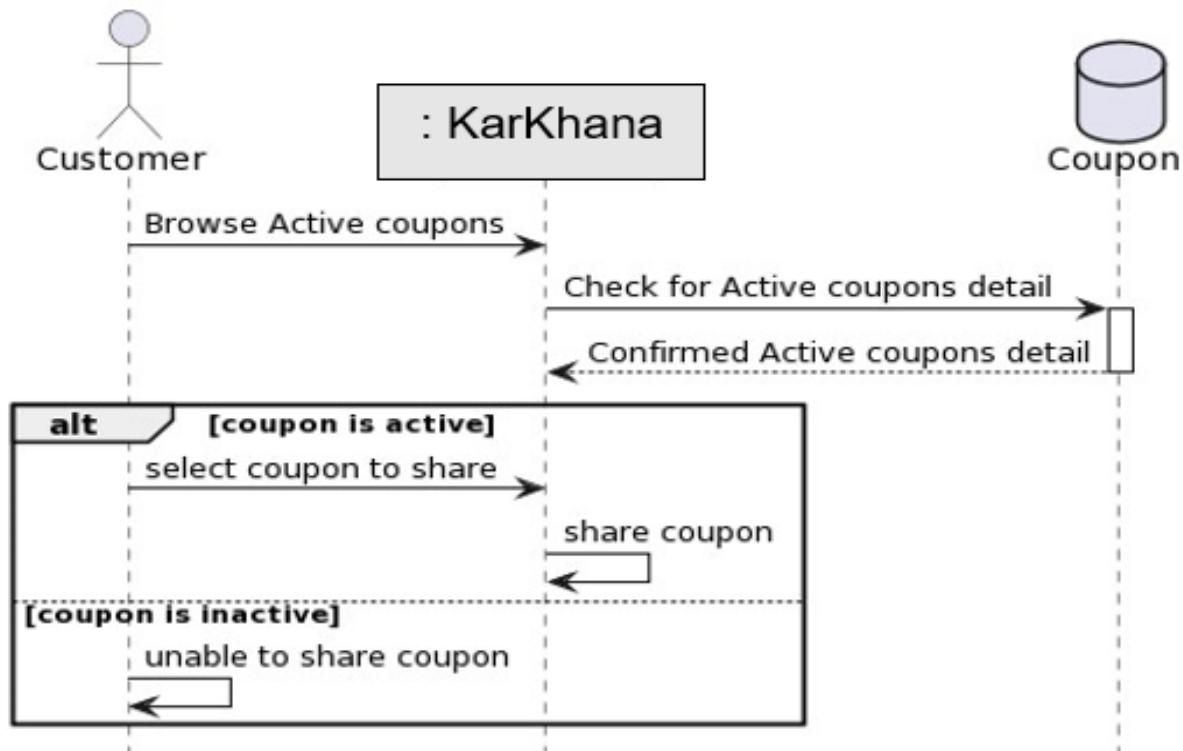


Figure 62: Sequence Diagram for Share Coupon.

### 3.6.5.1.9 Sequence Diagram For Add Deals

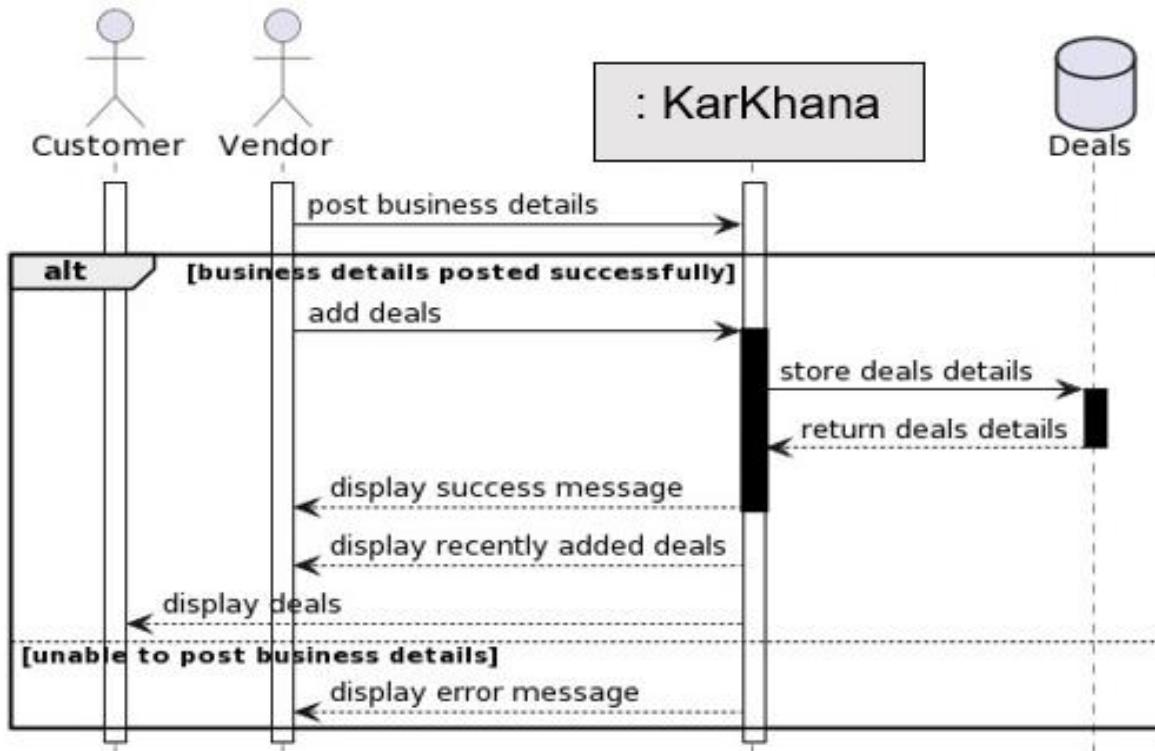


Figure 63: Sequence Diagram for Add Deals.

### 3.6.5.2 Flowchart

#### 3.6.5.2.1 Flowchart for Login

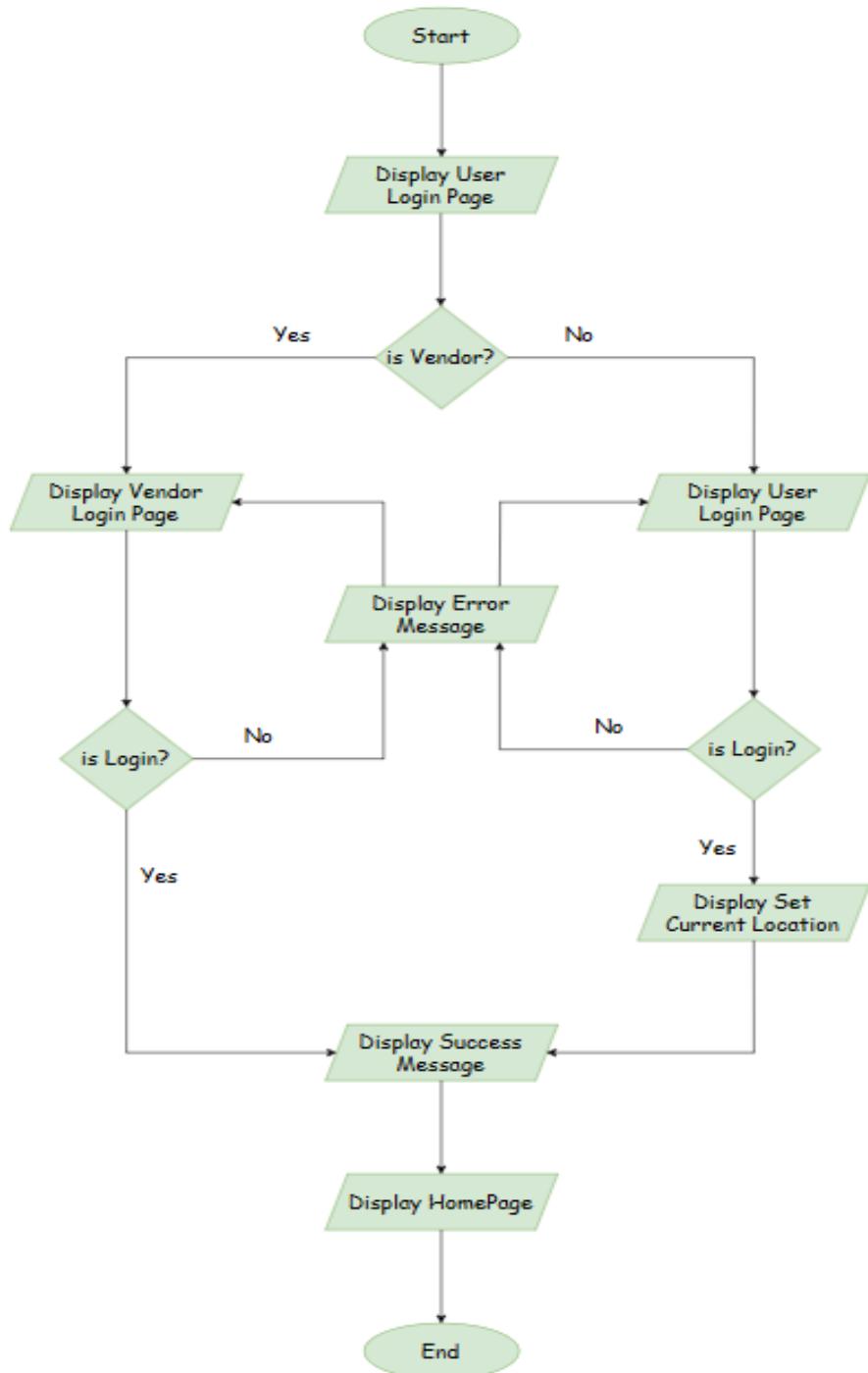


Figure 64: Flowchart for Login.

### 3.6.5.2.2 Flowchart for User Registration

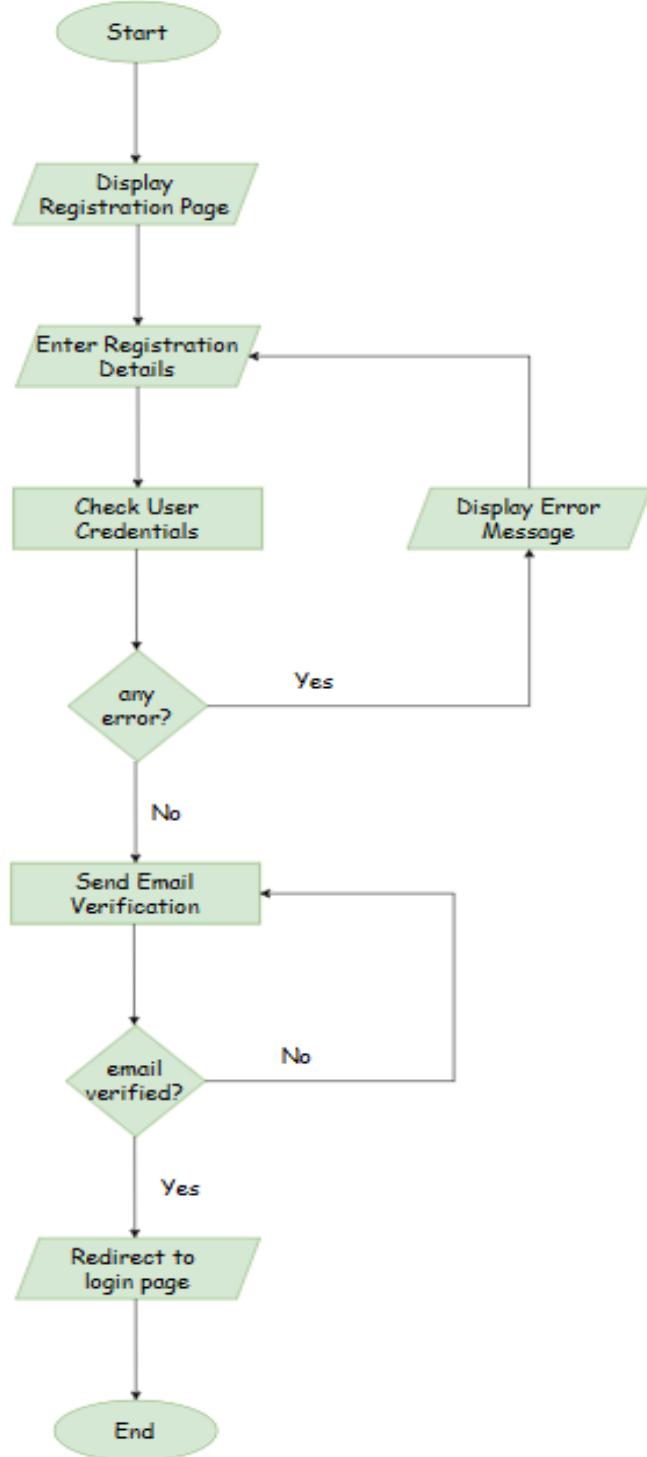


Figure 65: Flowchart for User Registration.

### 3.6.5.2.3 Flowchart for Vendor Registration

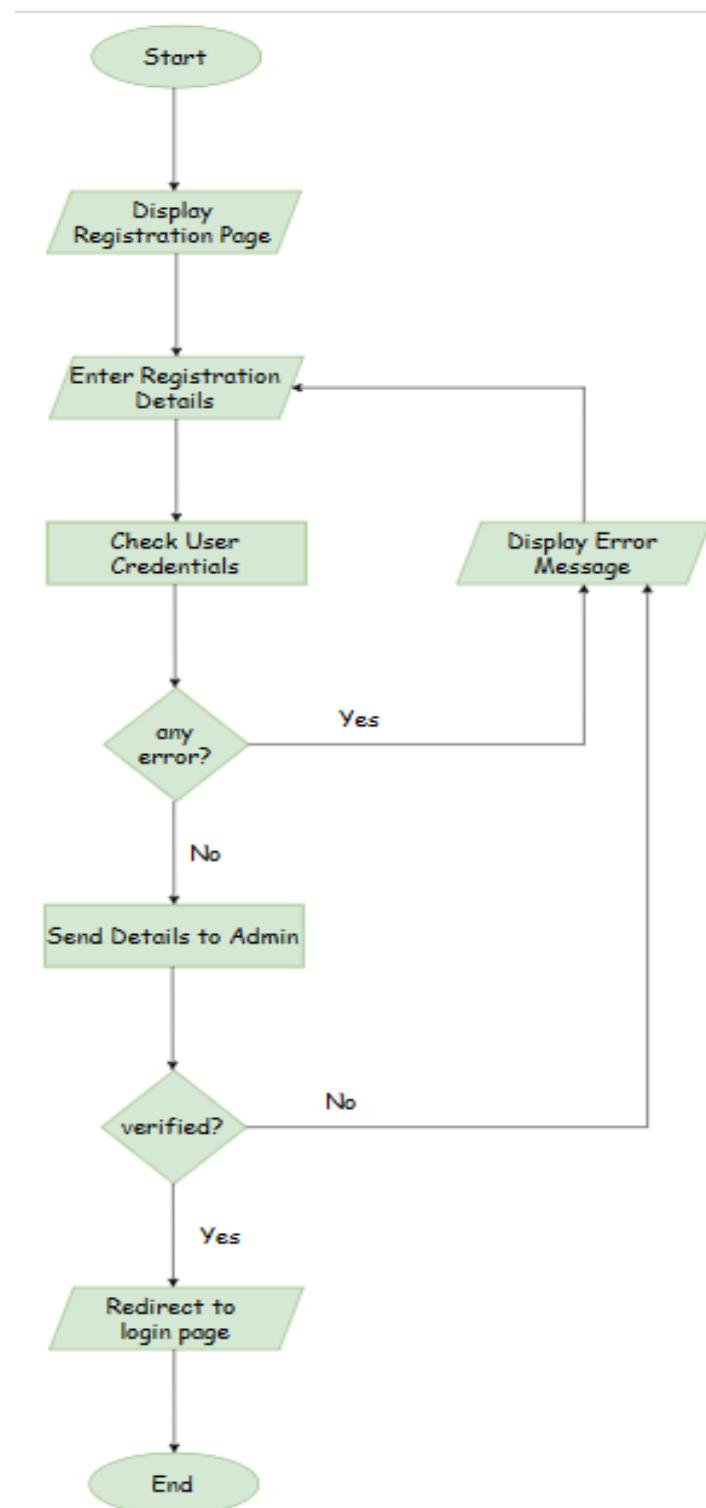


Figure 66: Flowchart for Vendor Registration.

### 3.6.5.2.4 Flowchart for Set Current Location

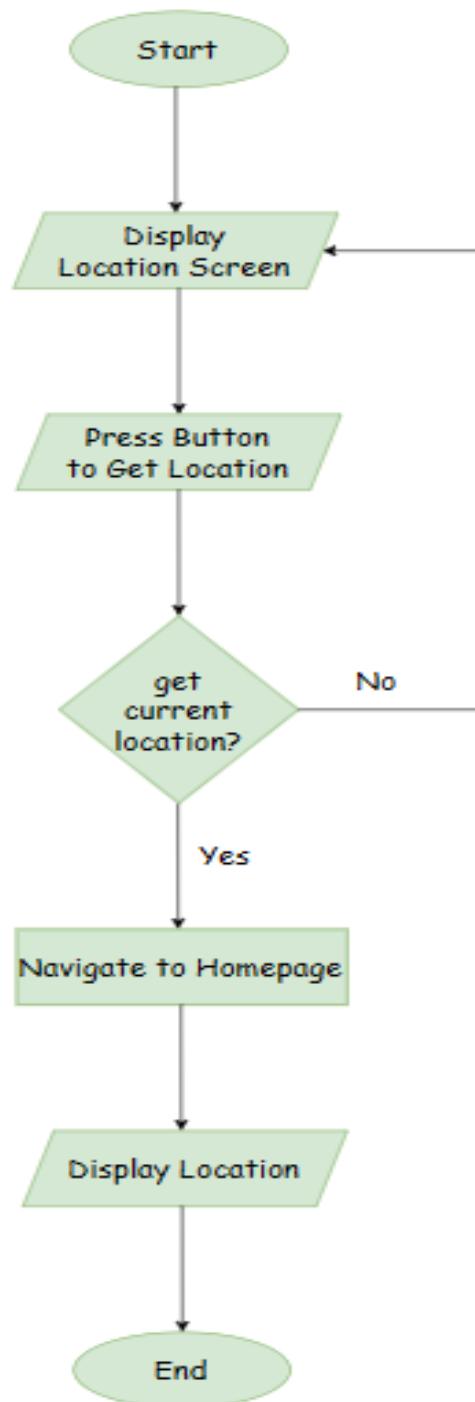


Figure 67: Flowchart for Set Current Location.

### 3.6.5.2.5 Flowchart for Add to Cart

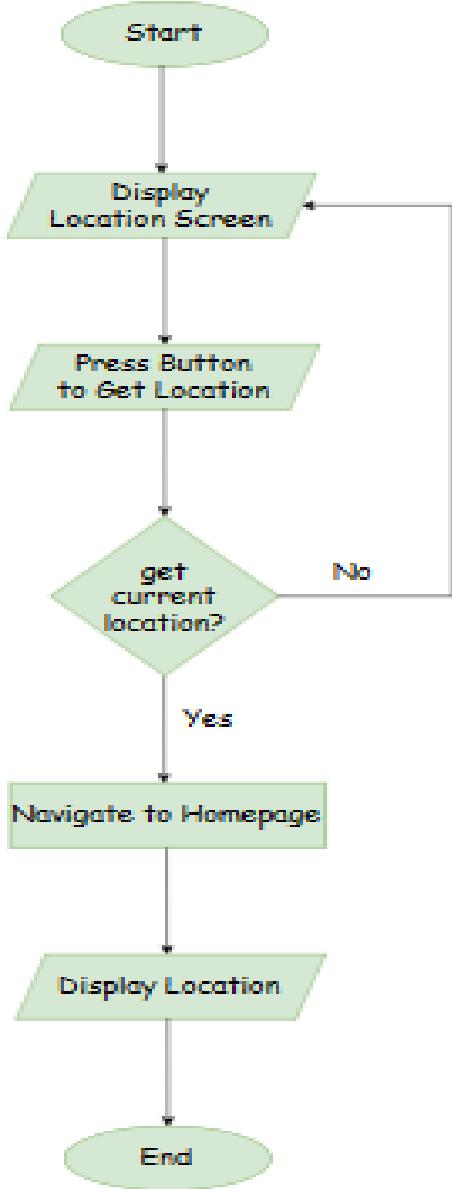


Figure 68: Flowchart for add to cart.

### 3.6.5.2.6 Flowchart for Payment

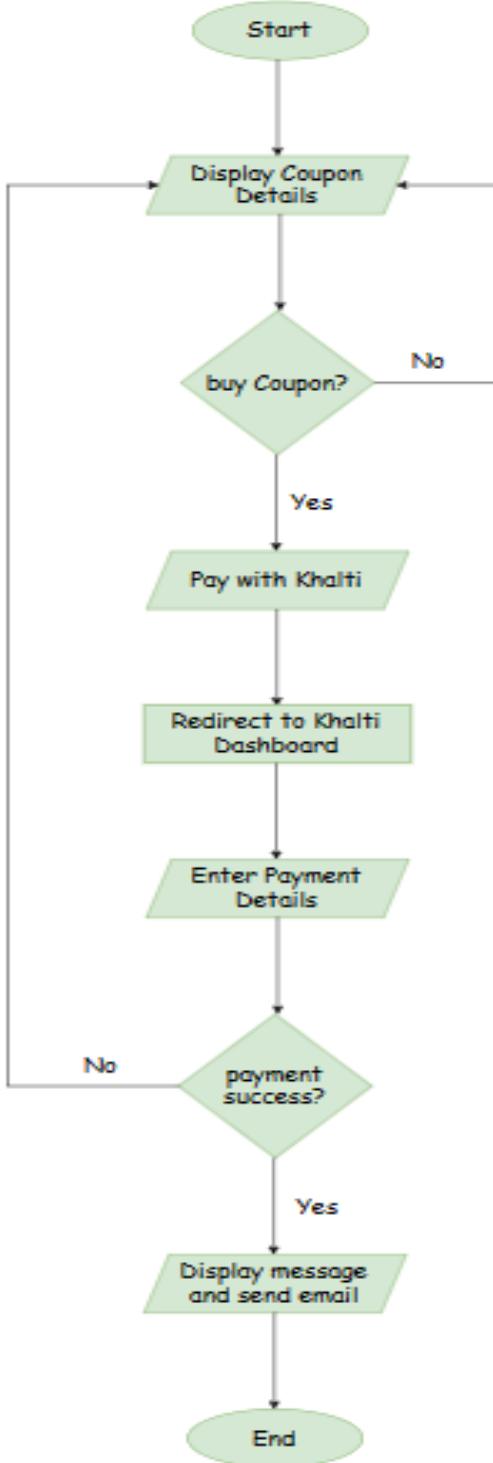


Figure 69: Flowchart for Payment.

### 3.6.5.2.7 Flowchart for Coupon Generate

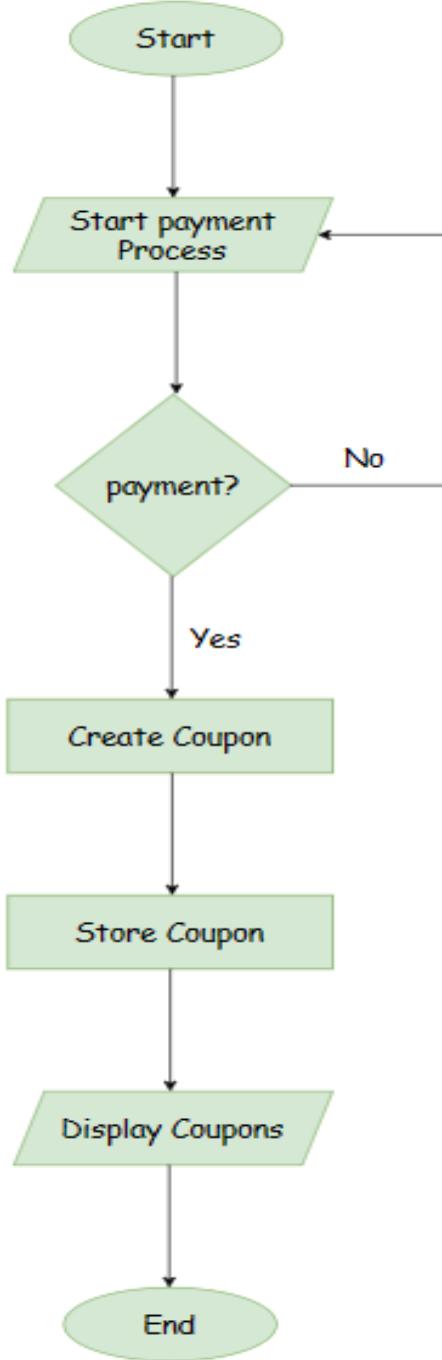


Figure 70: Flowchart for generate coupon.

### 3.6.5.2.8 Flowchart for Confirm Coupon

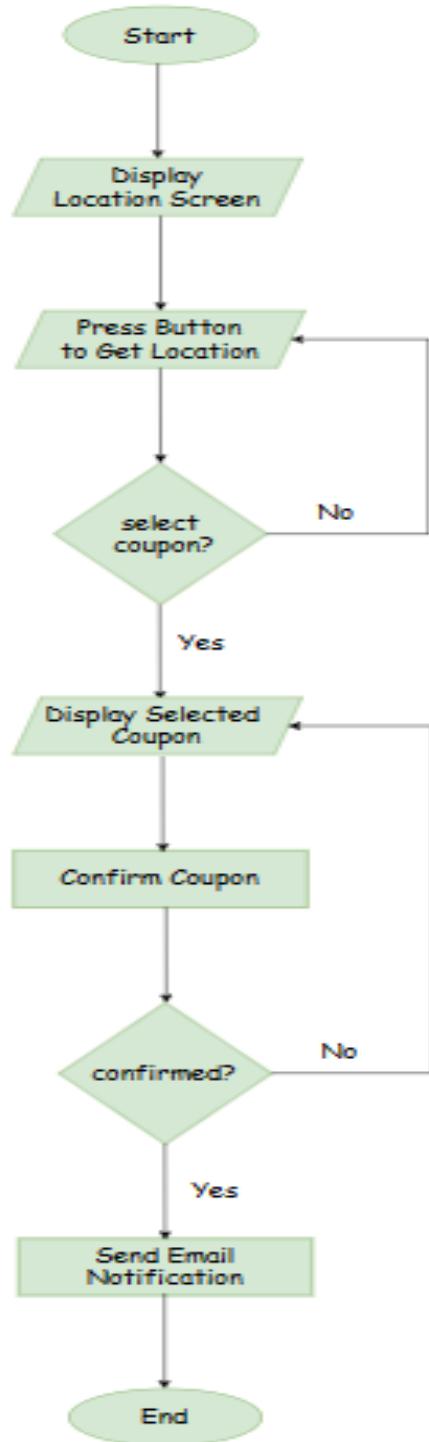


Figure 71: Flowchart for Confirm Coupon.

### 3.6.5.2.9 Flowchart for Share Coupon

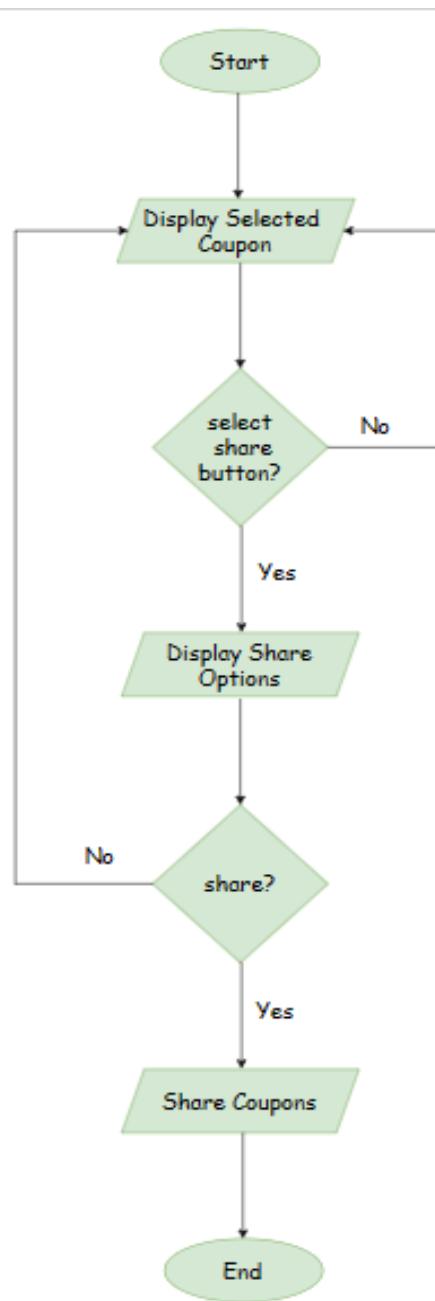


Figure 72: Flowchart for Share Coupon.

### 3.6.5.2.10 Flowchart for Add Deals

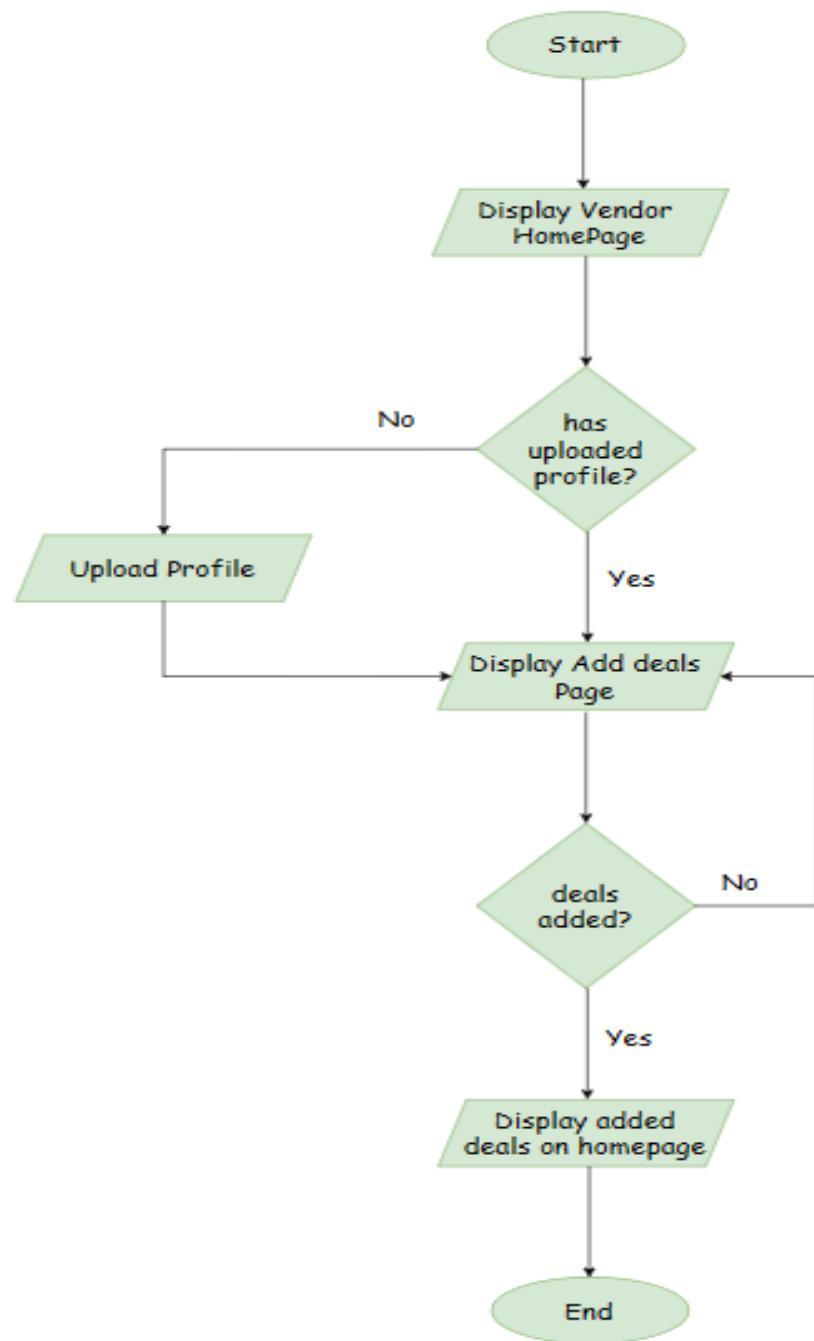


Figure 73: Flowchart for Add Deals.

### 3.6.5.3 Activity Diagram

#### 3.6.5.3.1 Activity Diagram for Login

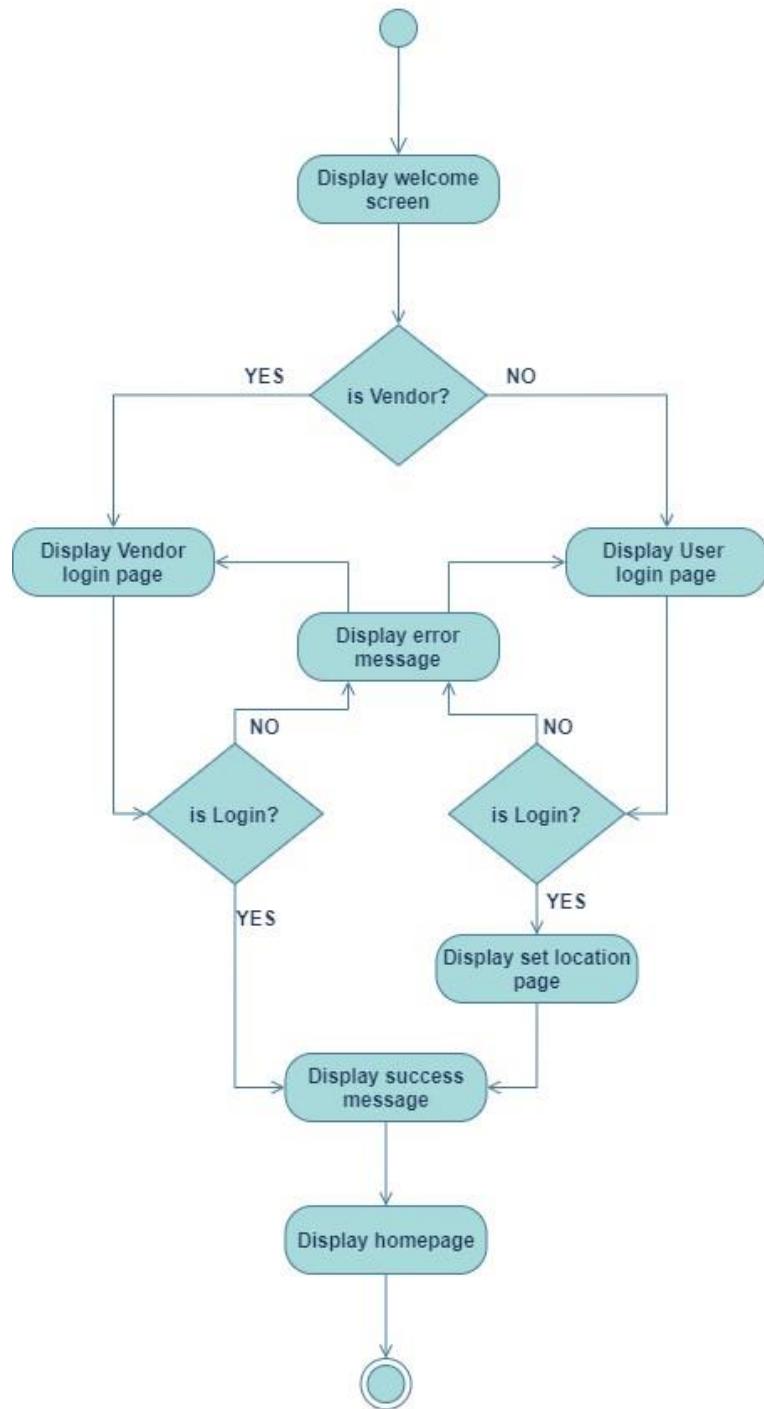


Figure 74: Activity Diagram for Login.

### 3.6.5.3.2 Activity Diagram for User Registration

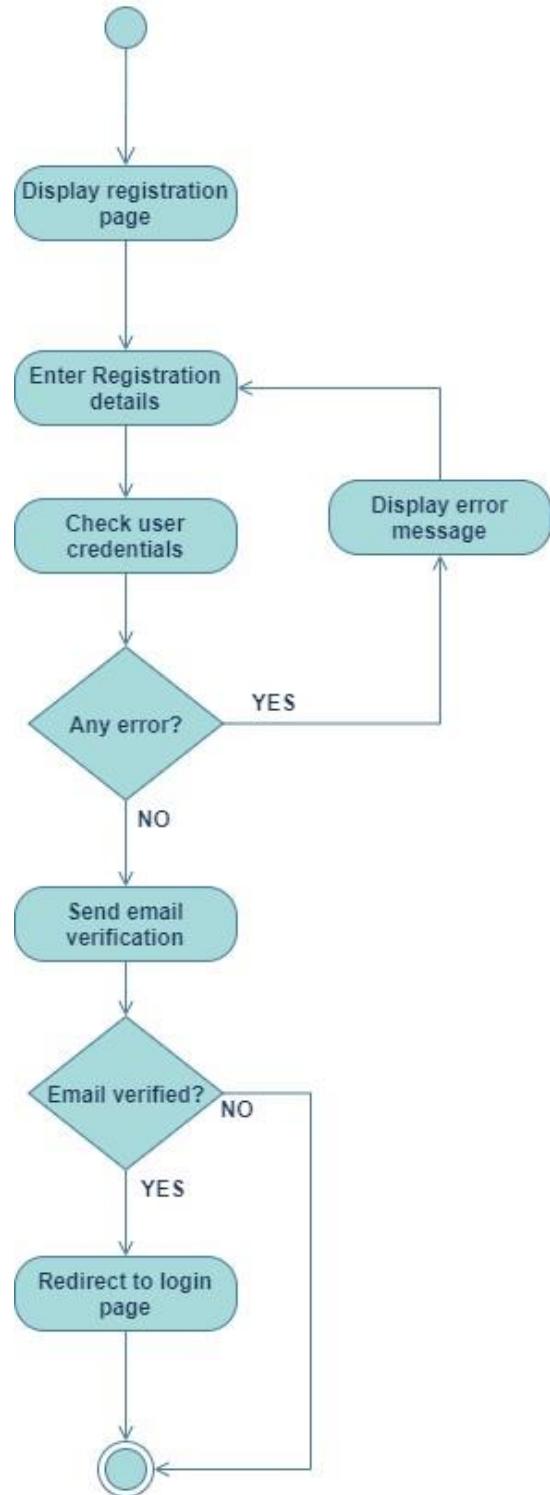


Figure 75: Activity Diagram for User Registration.

### 3.6.5.3.3 Activity Diagram for Vendor Registration

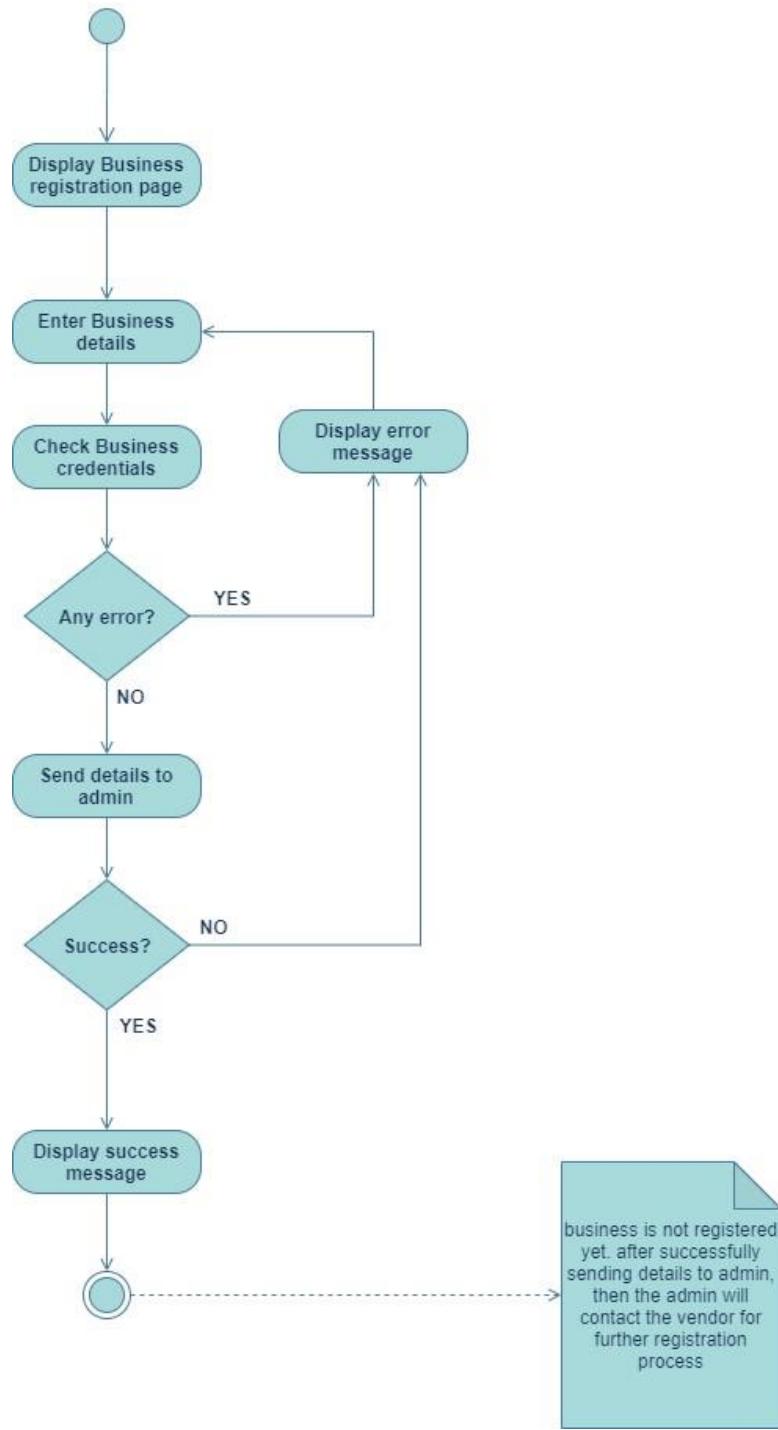


Figure 76: Activity Diagram for Vendor Registration.

### 3.6.5.3.4 Activity Diagram for Set Current Location

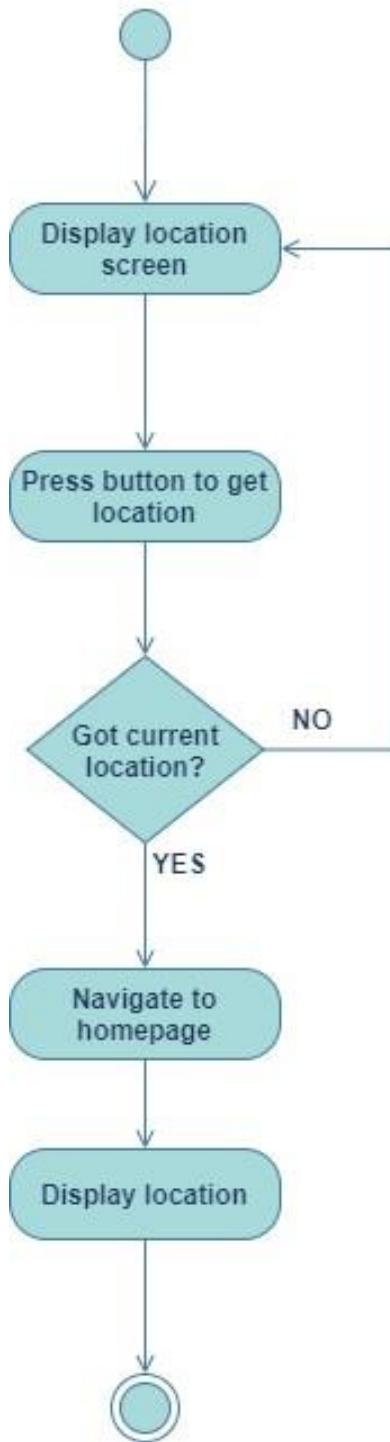


Figure 77: Activity Diagram for Set Current Location.

### 3.6.5.3.5 Activity Diagram for Add to Cart

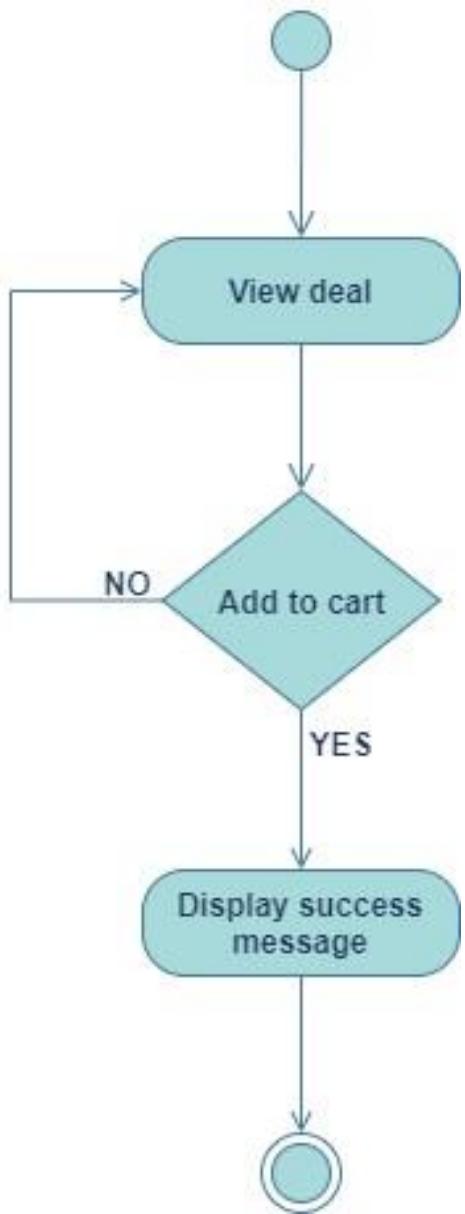


Figure 78: Activity Diagram for Add to cart.

### 3.6.5.3.6 Activity Diagram for Payment

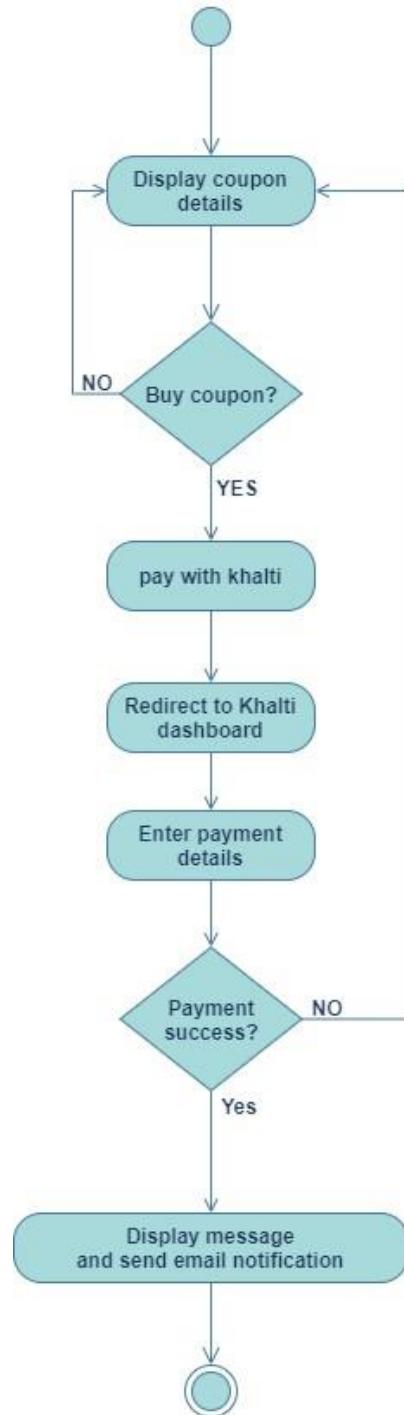


Figure 79:Activity Diagram for Payment.

### 3.6.5.3.7 Activity Diagram for Generate Coupon

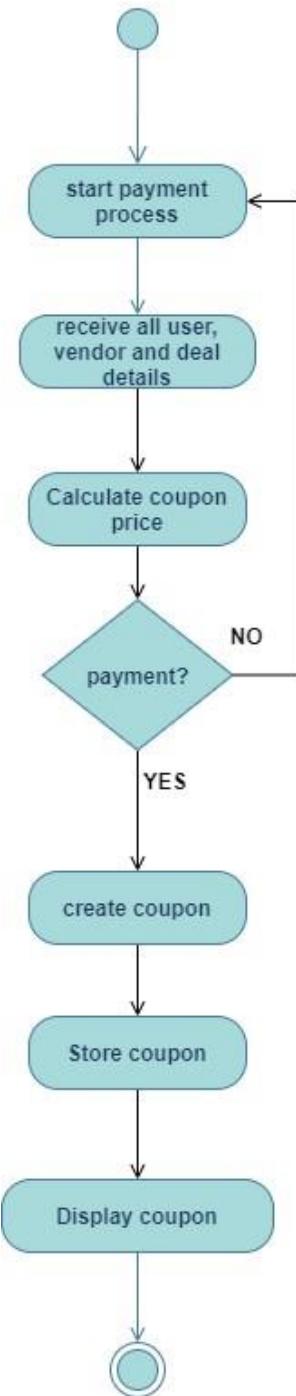


Figure 80: Activity Diagram for Coupon Generate.

### 3.6.5.3.8 Activity Diagram for Confirm Coupon

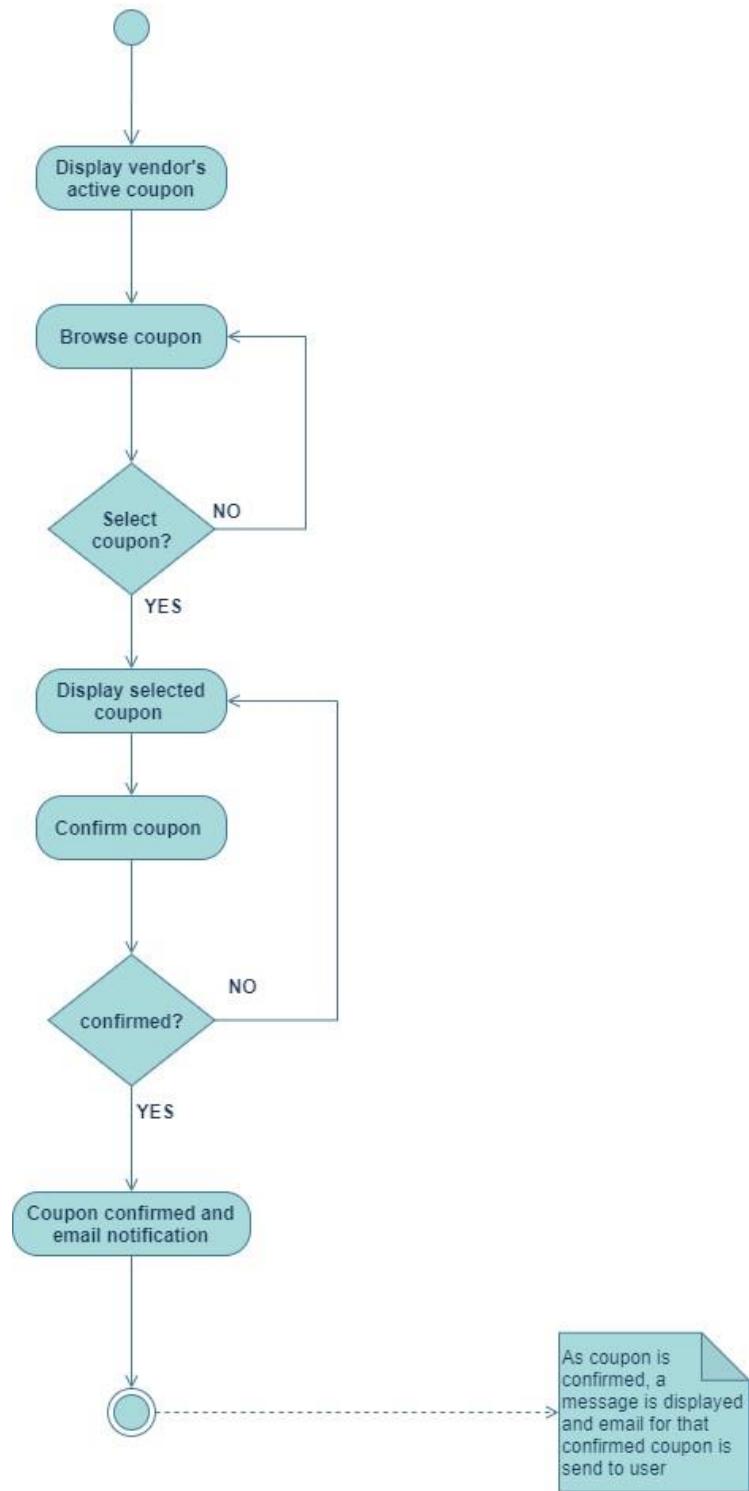


Figure 81: Activity Diagram for Confirm Coupon.

### 3.6.5.3.9 Activity Diagram for Share Coupon

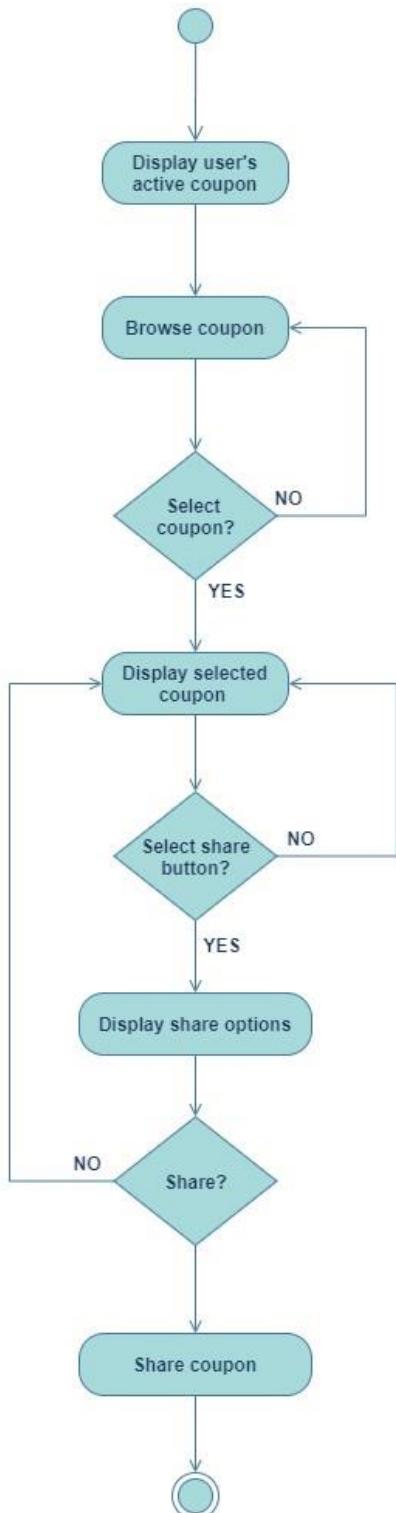


Figure 82: Activity Diagram for Share Coupon.

### 3.6.5.3.10 Activity Diagram for Add Deals

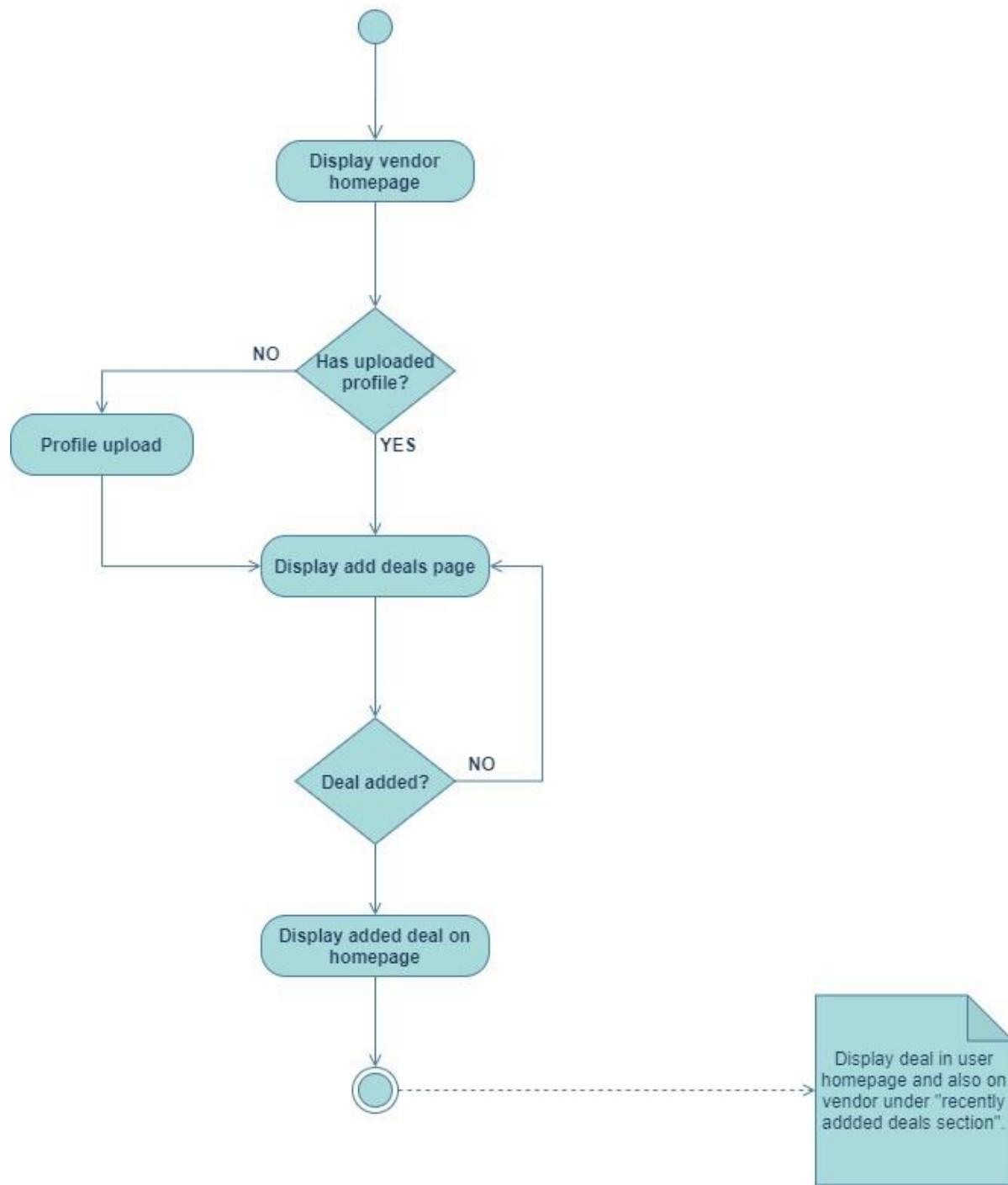


Figure 83: Activity Diagram for Add Deals.

### 3.6.5.4 DFD

#### 3.6.5.4.1 DFD for Login

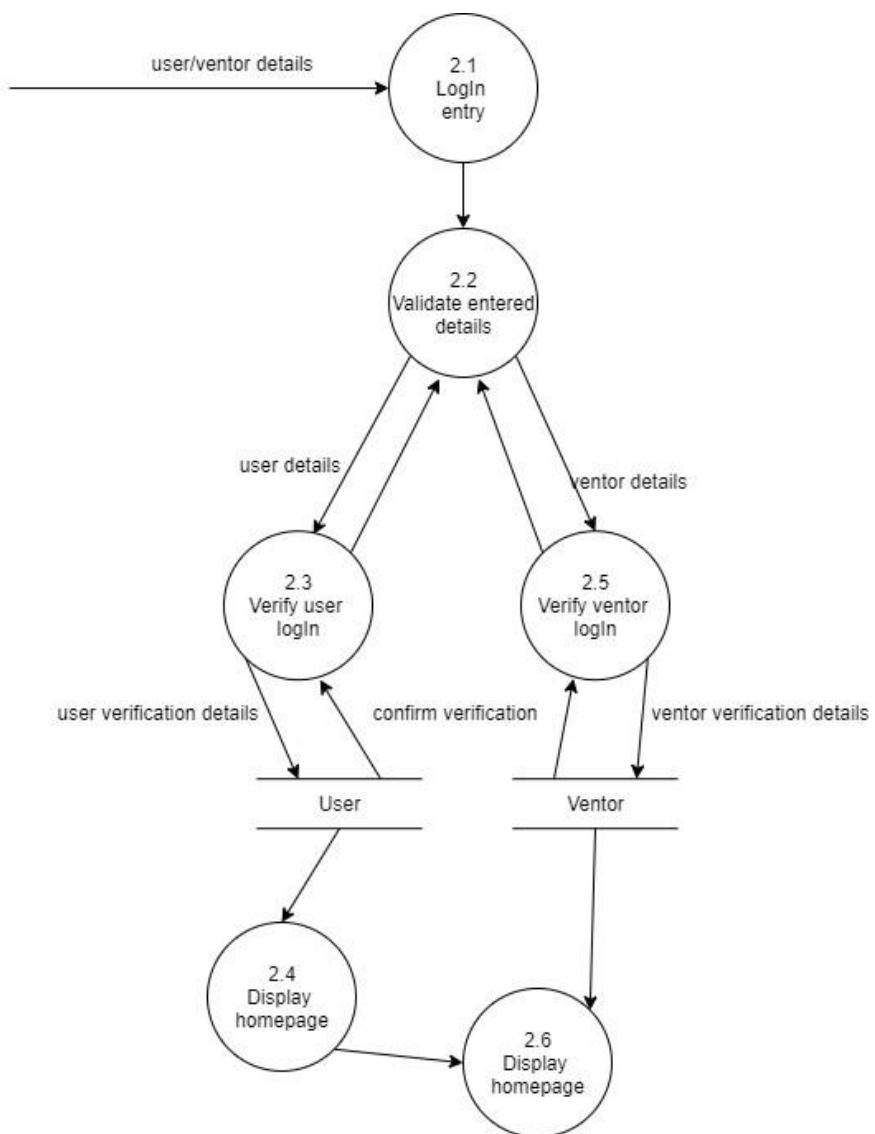


Figure 84: Level 2 DFD of Login.

### 3.6.5.4.2 DFD for User Registration

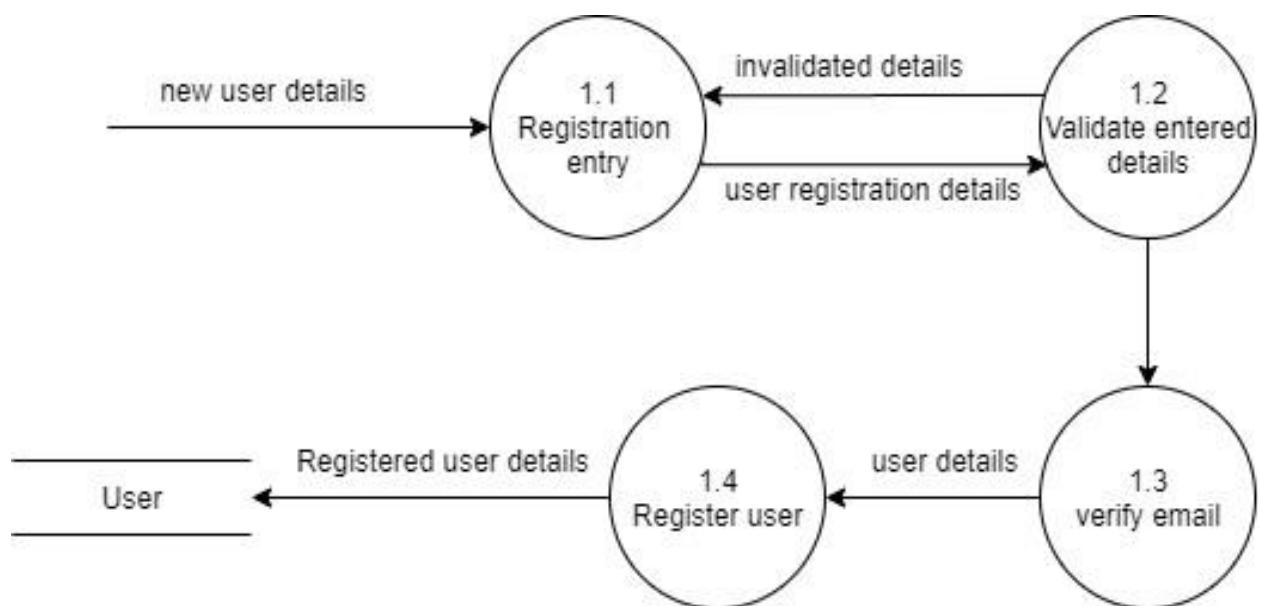


Figure 85: level 2 DFD for User Registration.

### 3.6.5.4.3 DFD for Vendor Registration

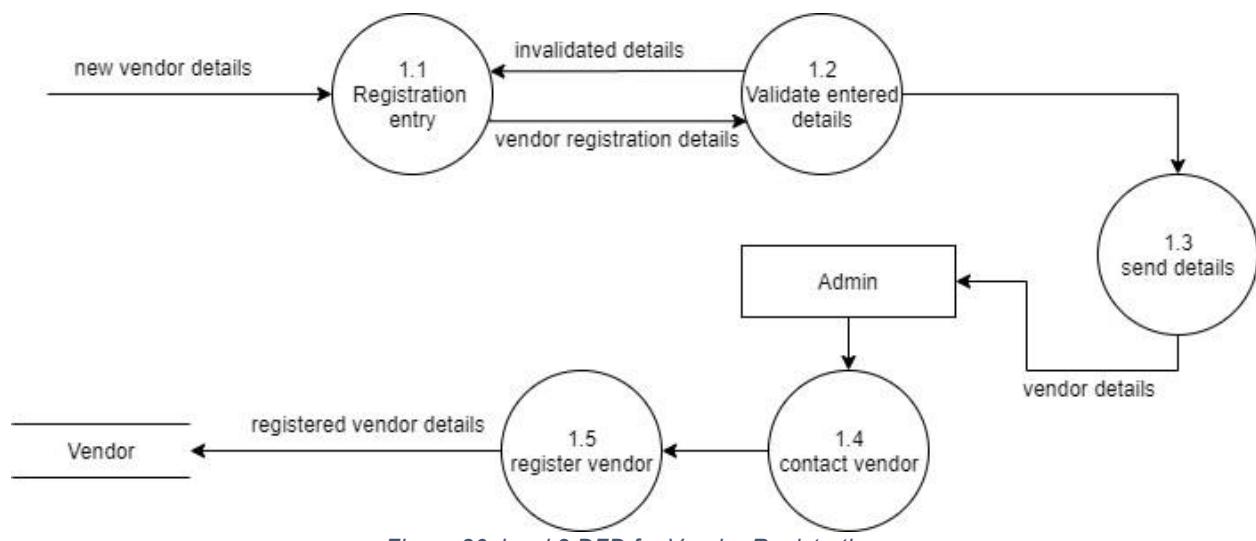


Figure 86: level 2 DFD for Vendor Registration.

### 3.6.5.4.4 DFD for Set Current Location

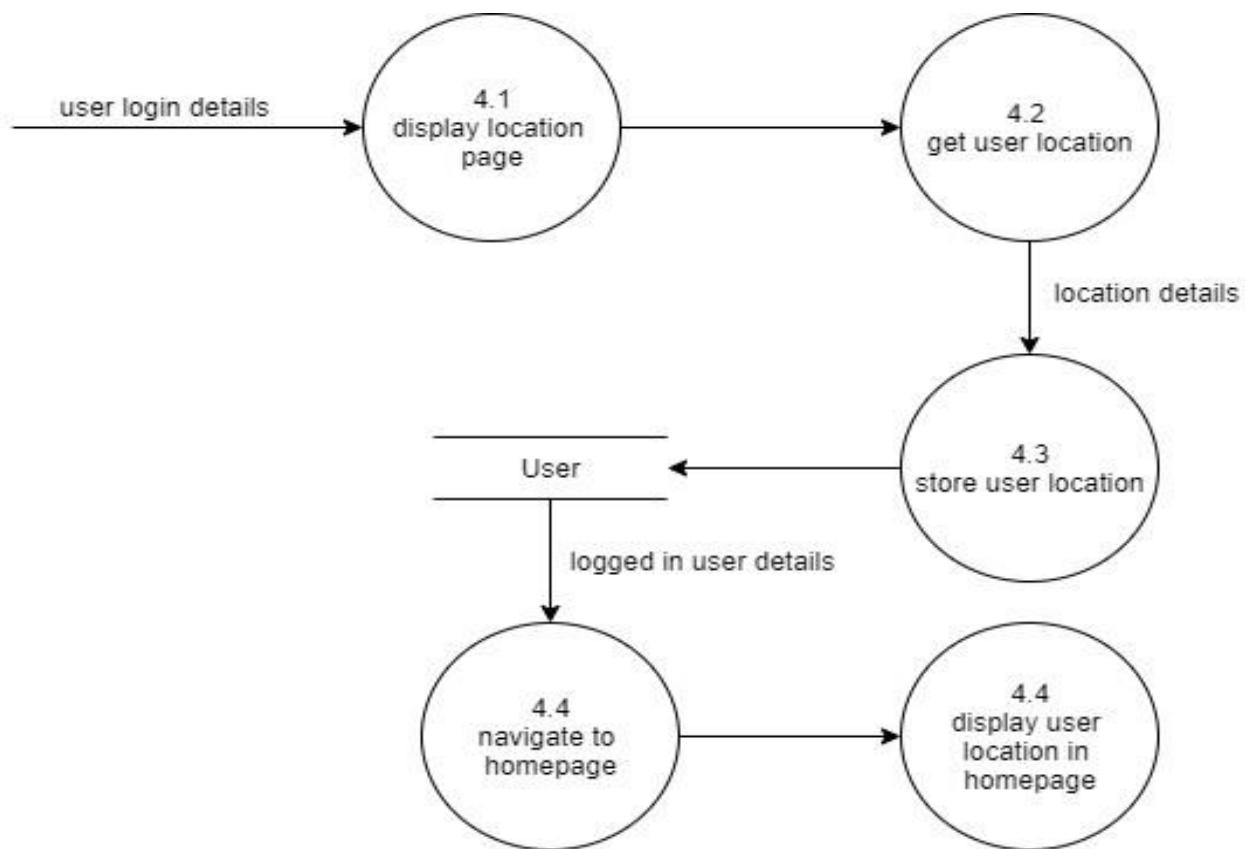


Figure 87: Level 2 DFD for Set Current Location.

### 3.6.5.4.5 DFD for Add to Cart

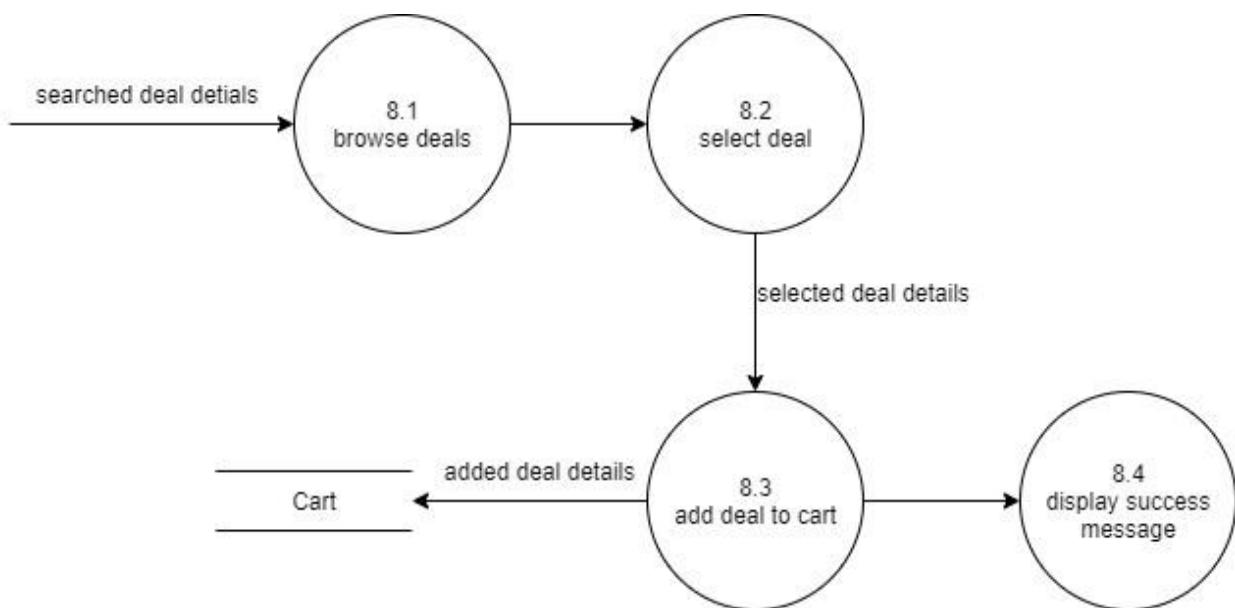


Figure 88: Level 2 DFD for Add to Cart.

## 10.6 DFD for Payment

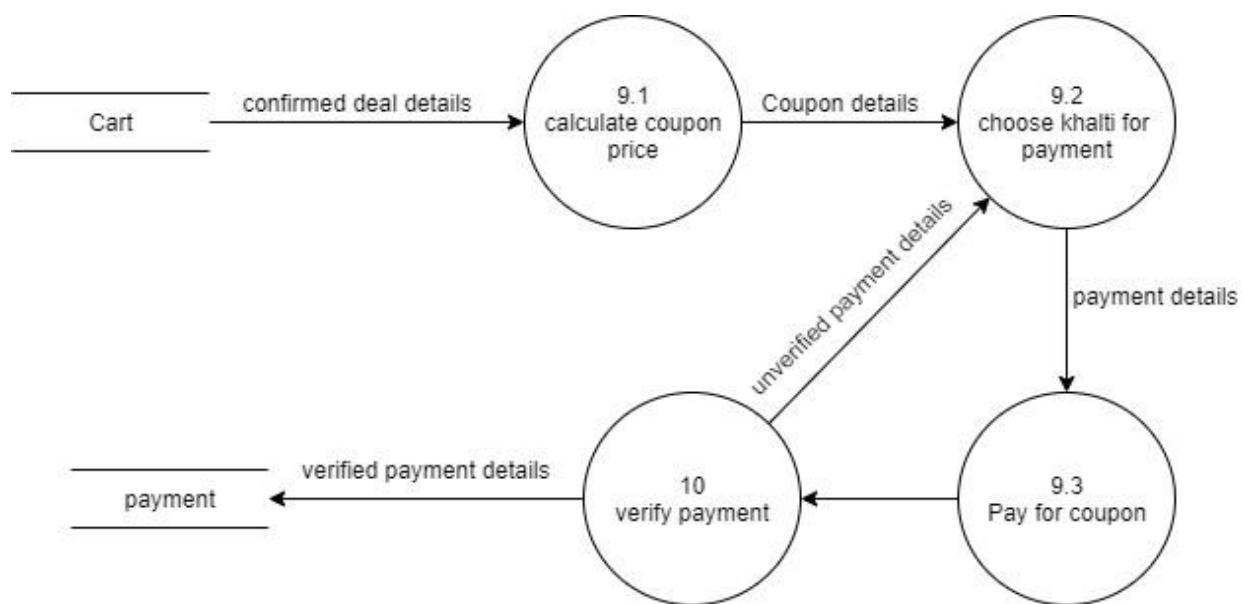


Figure 89: Level 2 DFD for Payment.

### 3.6.5.4.7 DFD for Coupon Generate

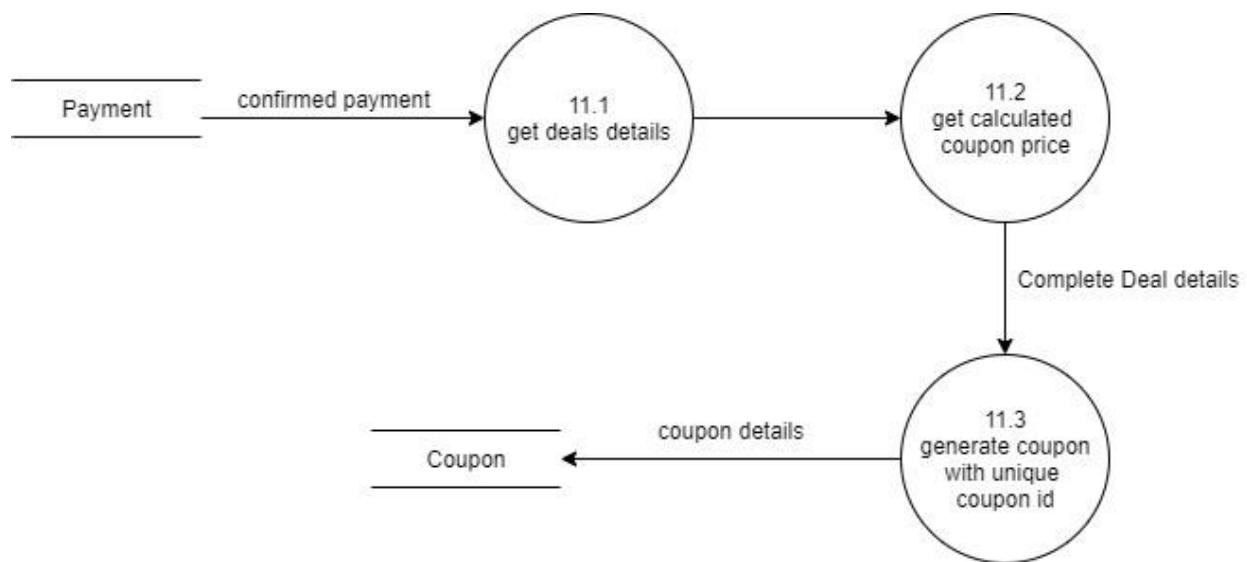


Figure 90: Level 2 DFD for Coupon Generate.

### 3.6.5.4.8 DFD for Confirm Coupon

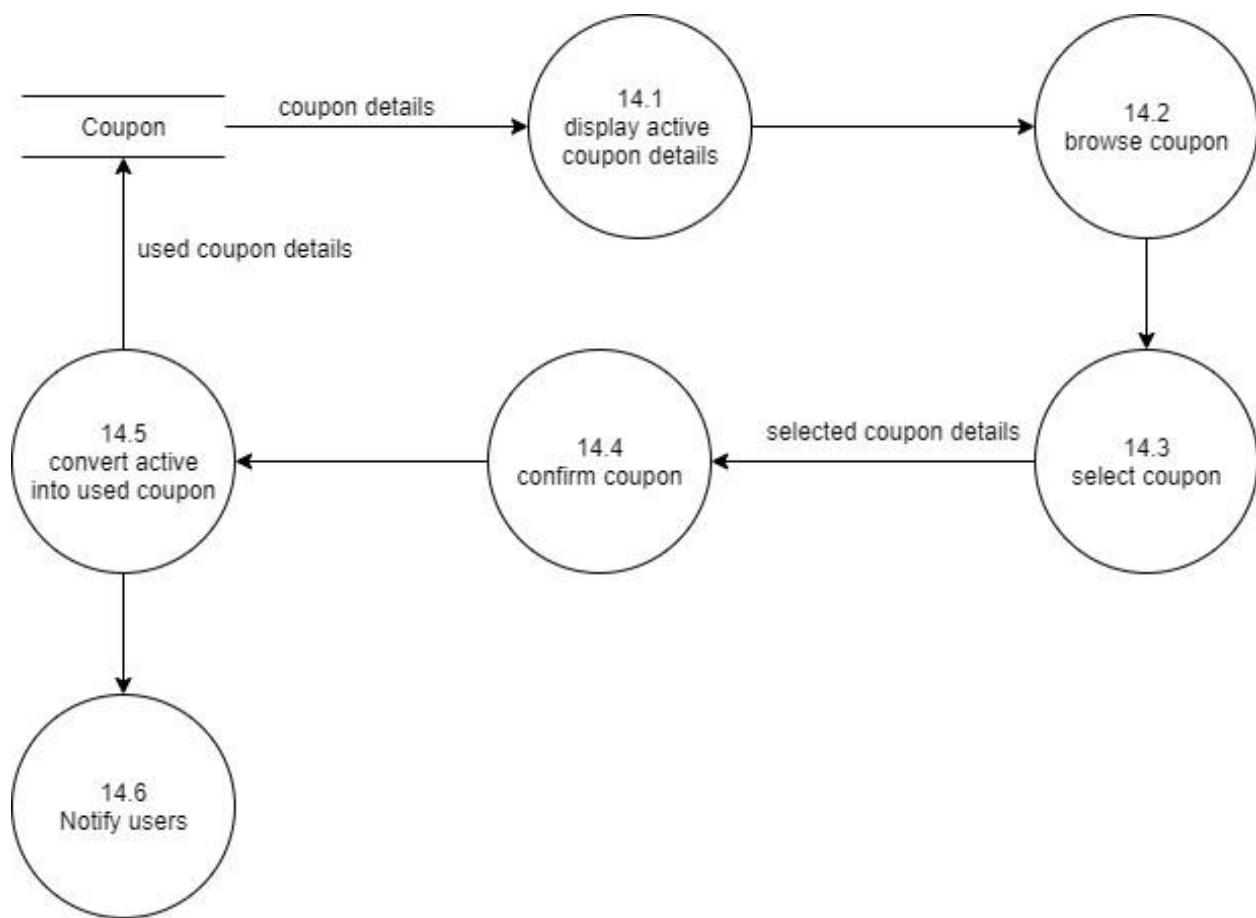


Figure 91: Level 2 DFD for Confirm Coupon

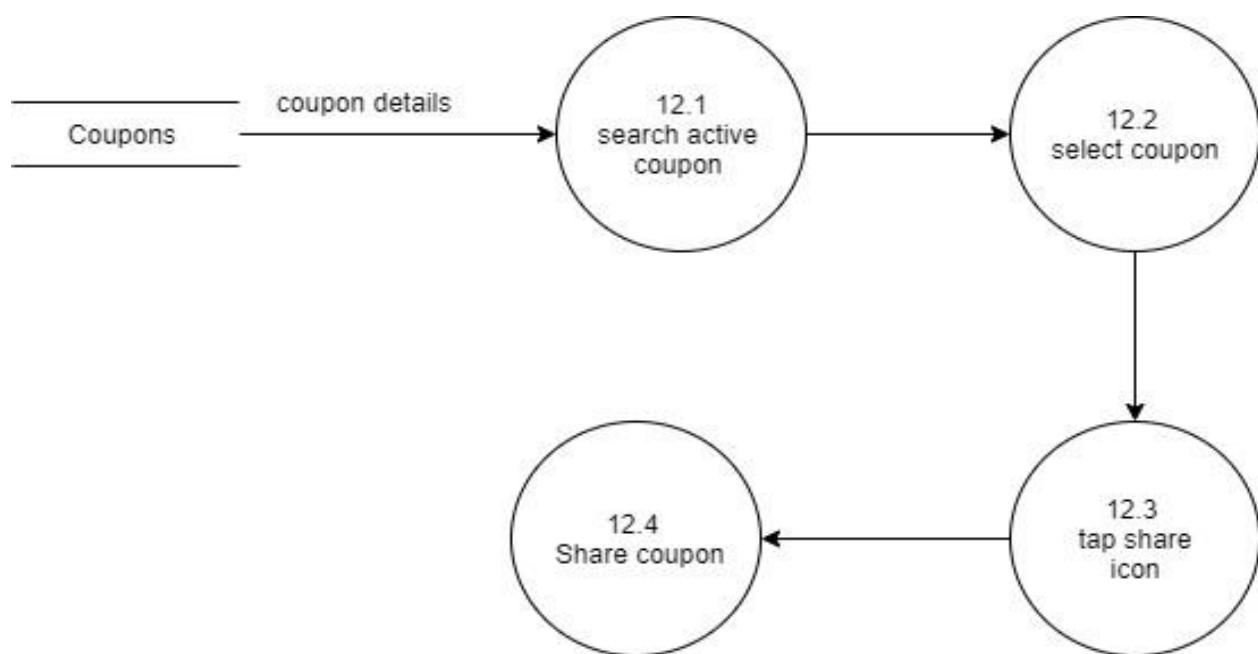
**3.6.5.4.9 DFD for Share Coupon**

Figure 92: Level 2 DFD for Share Coupon.

### 3.6.5.4.10 DFD for Add Deals

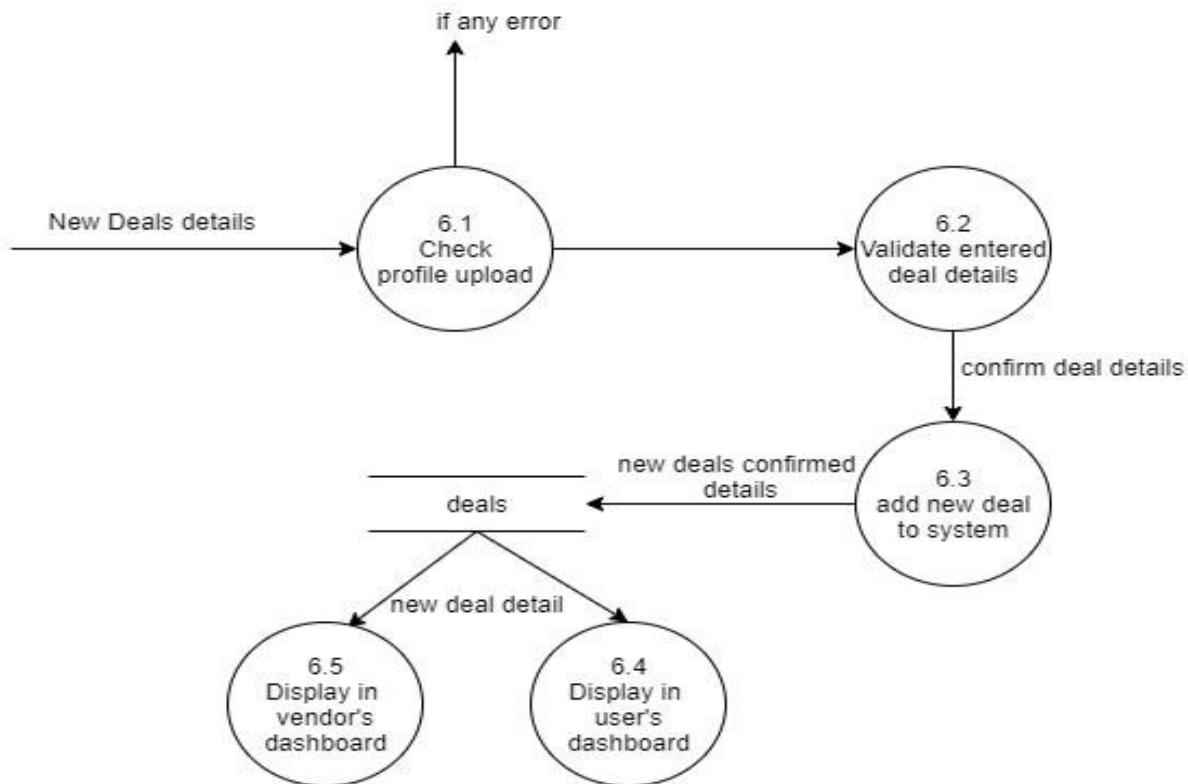


Figure 93: Level 2 DFD for Add Deals.

## 3.7 Implementation

This project was done according to the chosen methodology i.e., RUP that consist of 4 phases inception, elaboration, construction, and transition.

### 3.7.1 Inception

The first step of the RUP process, known as Inception, focuses on identifying the project's scope, goals, and viability. This phase required a lot of research because it established the overall project concept. The present market environment for this type of application, as well as the issue domain, were examined, and it was decided how this application will operate as a solution to the existing difficulties. This phase also included the development of main objectives and goals, as well as a clear vision for the project.

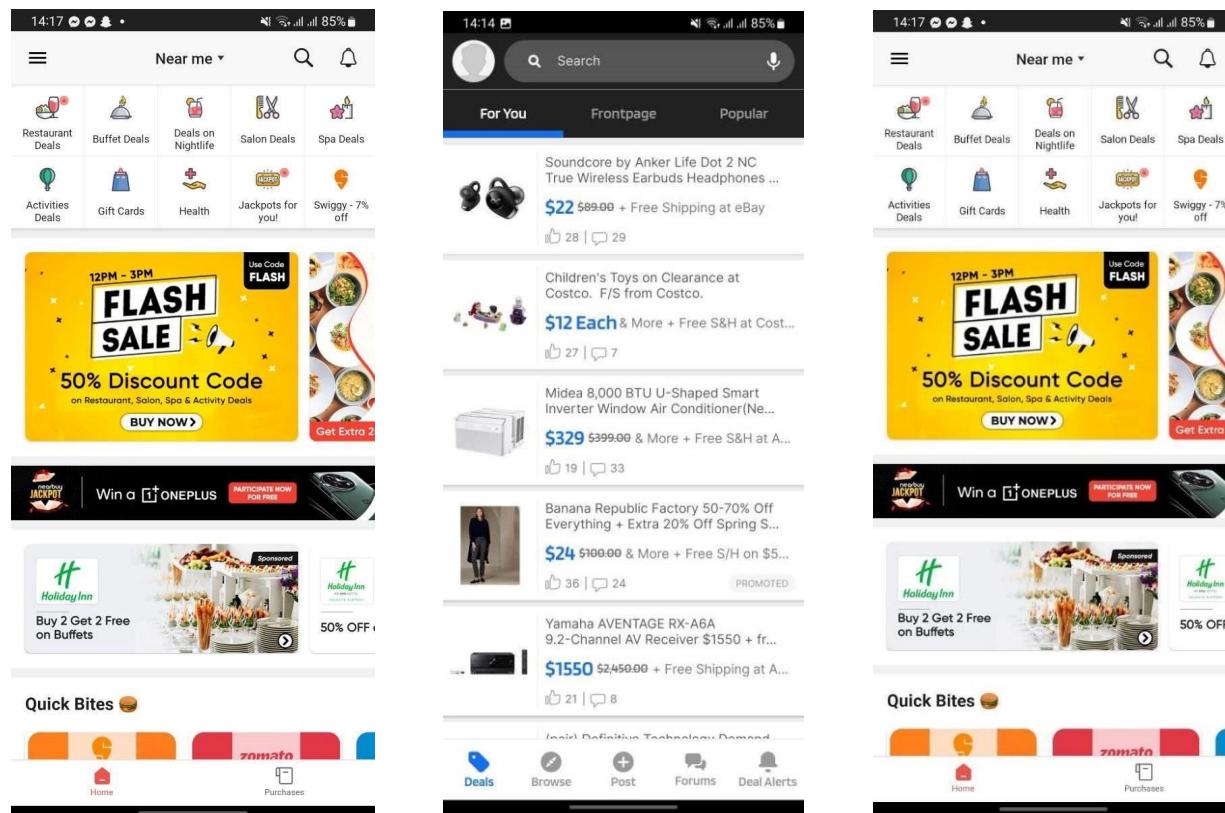


Figure 94: Screenshots of Similar Projects.

Lot of [similar apps](#) were first researched and compared for more discrete overall concept of the project.

As project planning was completed and finalized in order to proceed to the next phases, a work breakdown structure was developed for the project to make continued development on the project more effective, as time is a significant issue for completion. This makes it easier to arrange the duties and keep everything on time.

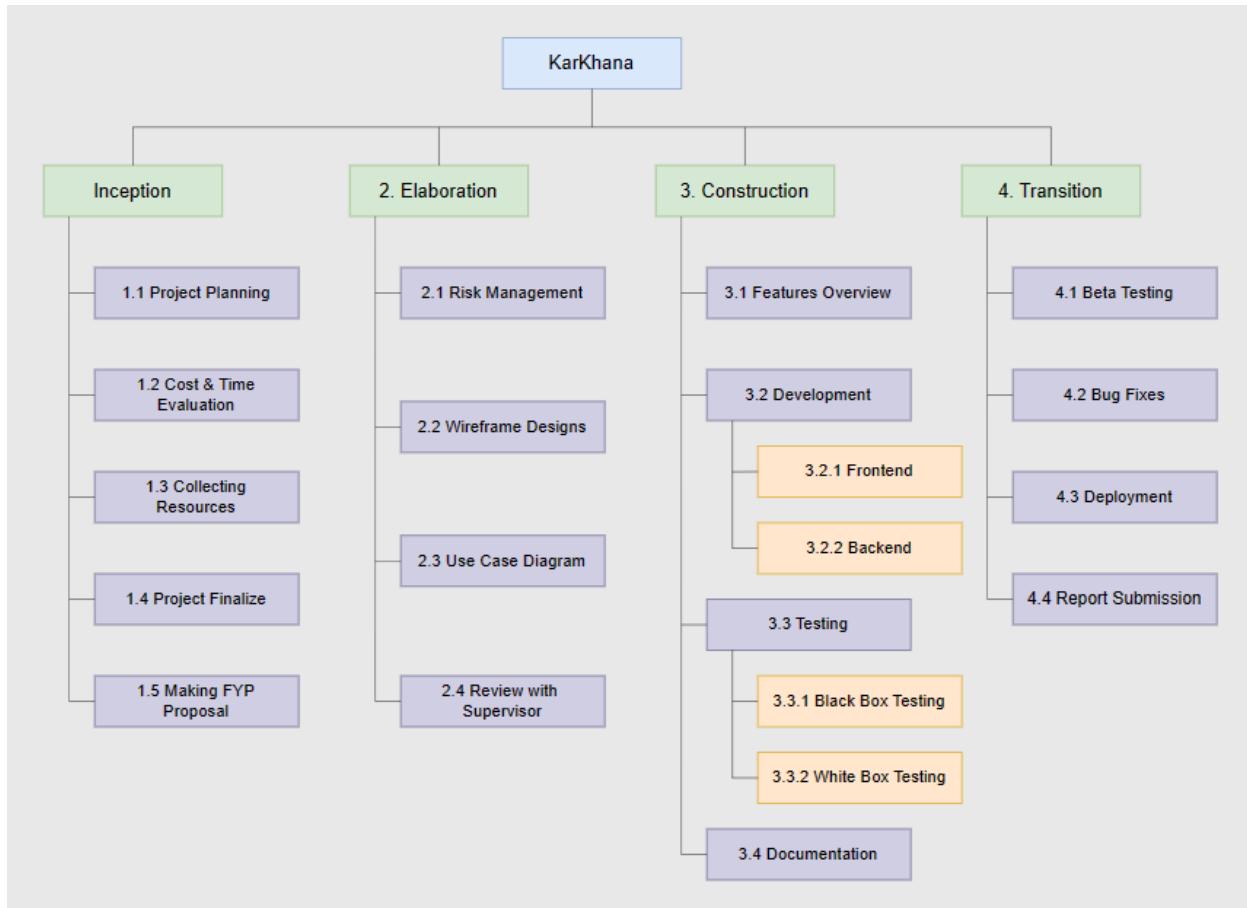


Figure 95: Work Break down Structure.

Along with how the project will be completed after breaking down each component, a Gantt chart was created to provide an overview of how the project will advance in the future, including additional specifics such as dates and days for initiation and completion. It provides a clearer perspective of the project's completion route.

## Final Gantt Chart

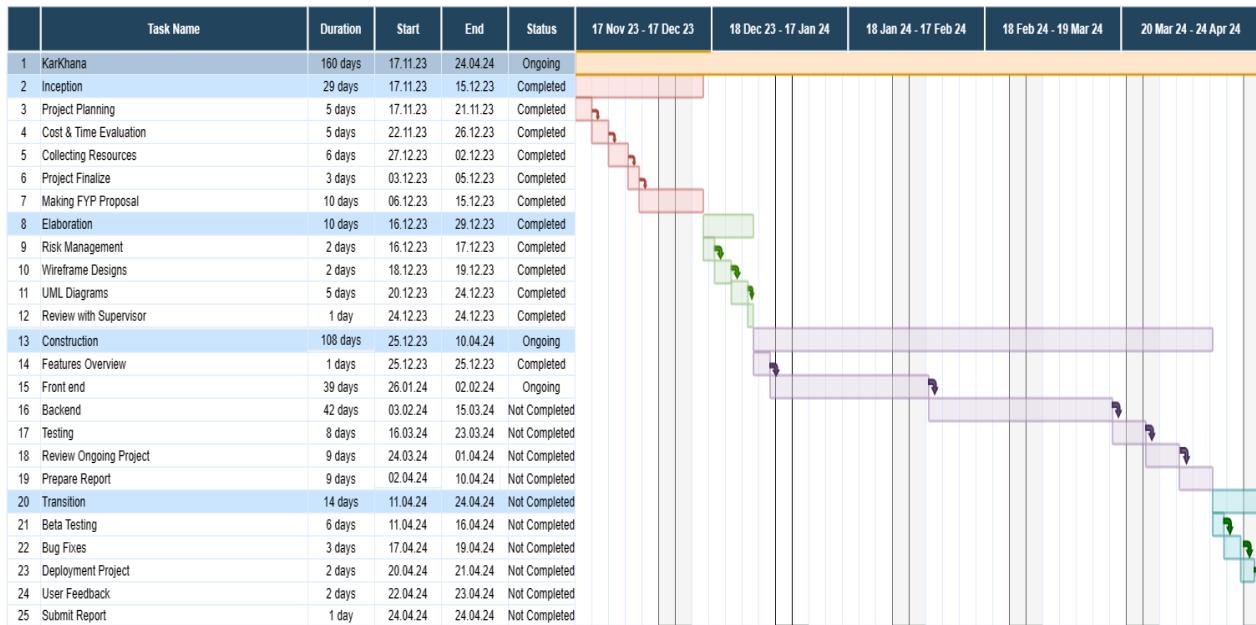


Figure 96: Final Gantt Chart.

### 3.7.2 Elaboration

In this phase, all the requirements and idea for the system is analyzed and finalized before the development starts. All the designs like UML, wireframes, prototypes, system architecture and other diagrams for the system were completed in this phase.

#### 3.7.2.1 System Architecture Diagram

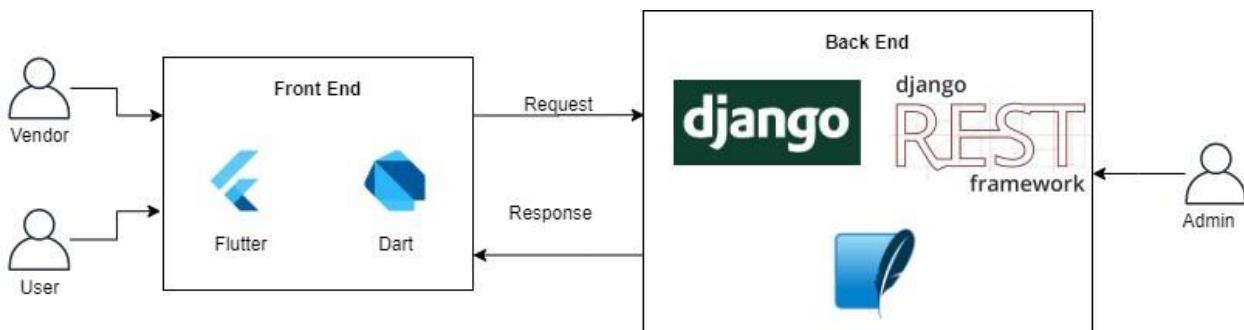


Figure 97: System Architecture Diagram in Elaboration Phase.

The system consists of three end users i.e., user (Customer), Vendors and Admin where user and vendors can register and login into system through mobile application. Admin will be handling all system data from web of Django admin panel.

### 3.7.2.2 ERD

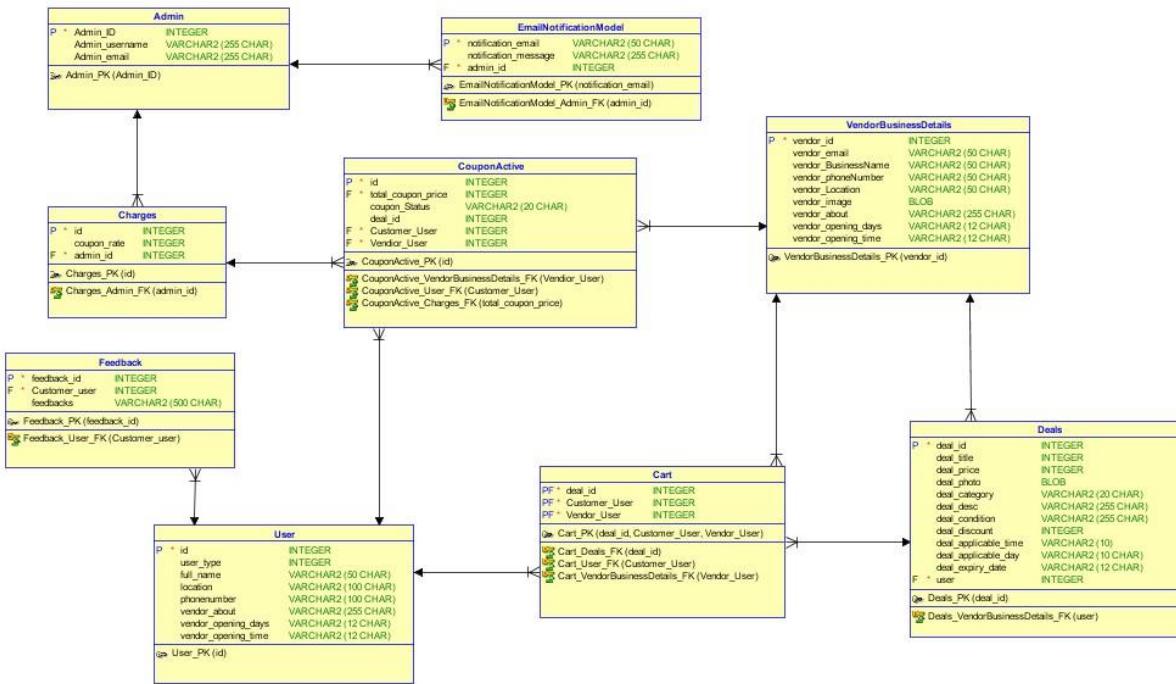


Figure 98: ERD Design in Elaboration Phase.

### 3.7.2.3 Use Case Diagram

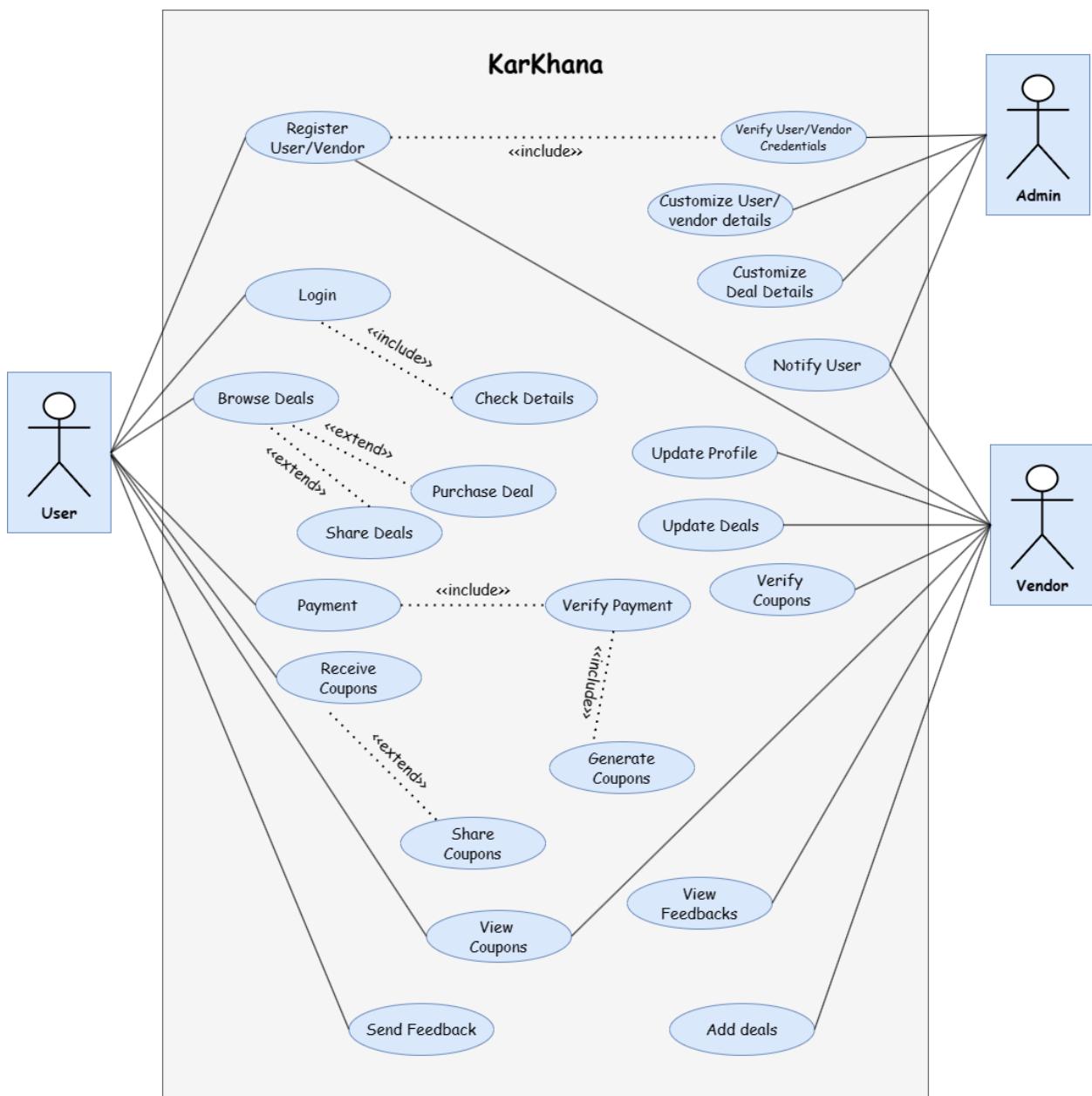


Figure 99: Use Case Diagram in Elaboration Phase.

### 3.7.2.4 Sequence Diagram

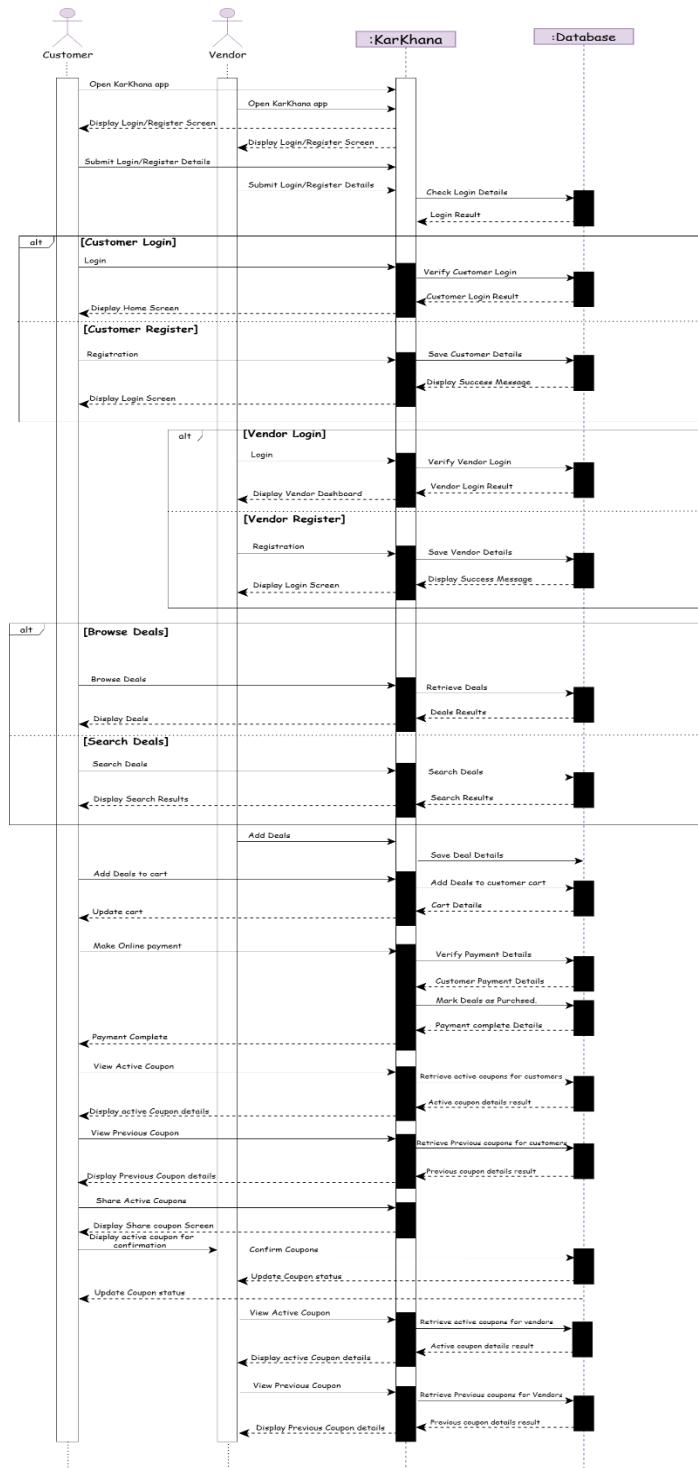


Figure 100: Sequence Diagram in Elaboration Phase.

### 3.7.2.5 Flowchart

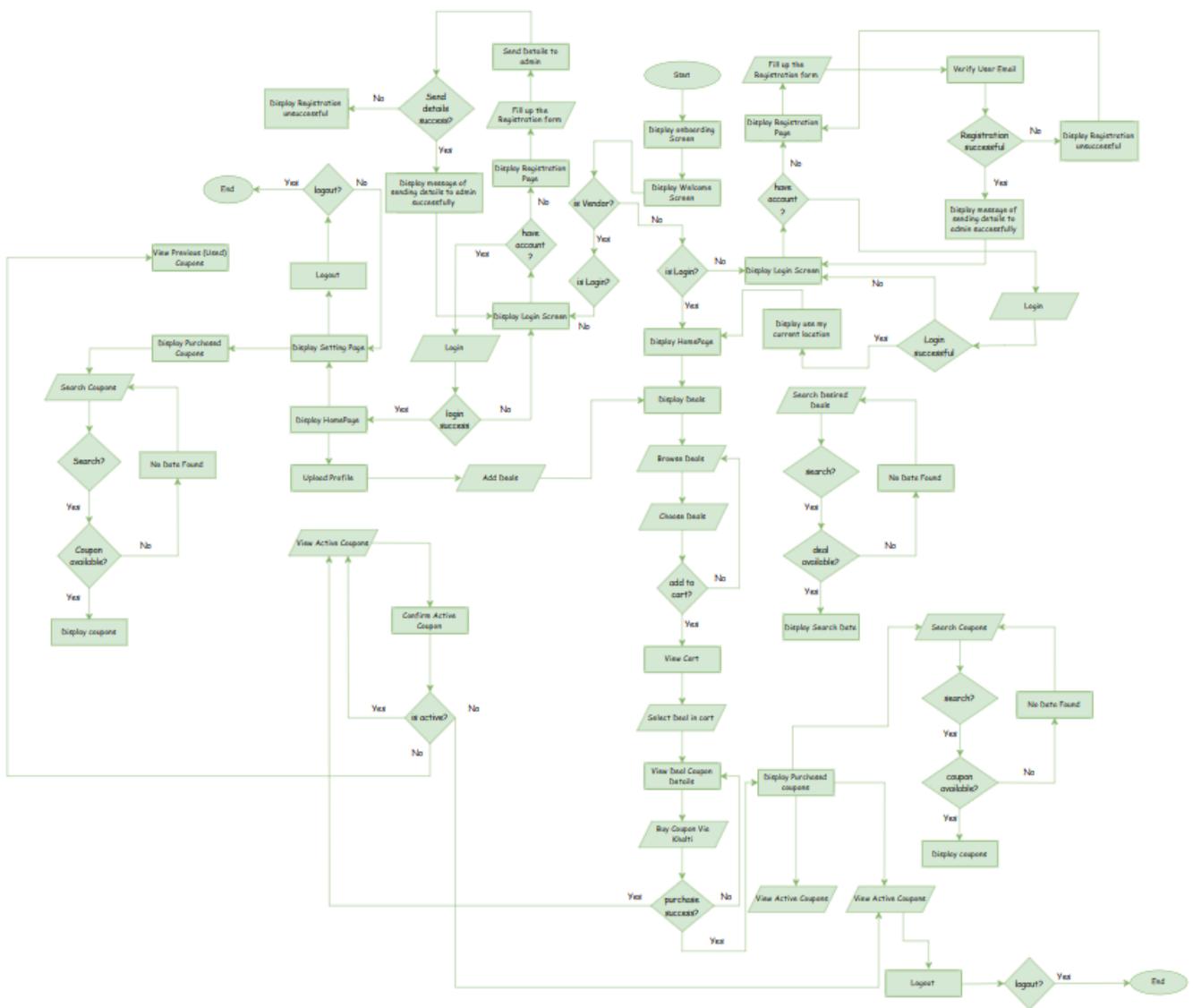


Figure 101: Overall Flowchart in Elaboration Phase.

### 3.7.2.6 Activity Diagram

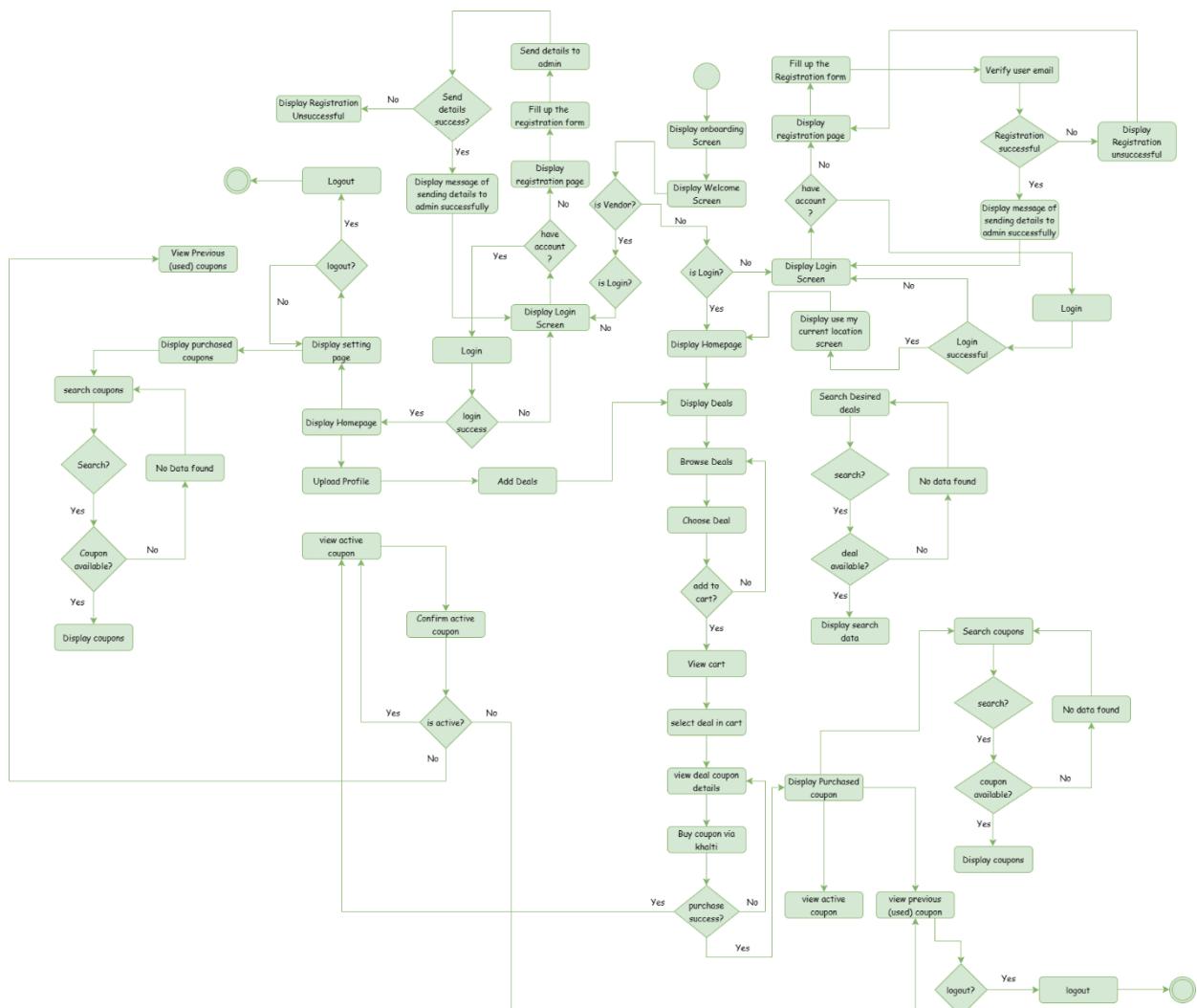


Figure 102: Overall Activity Diagram in Elaboration Phase.

### 3.7.2.7 Wireframes

Wireframes were also designed in this phase. This helped to see the UI of the system in early phase on how the system is going to be and how it should look which made the further work in development part easier. To see the complete wireframes of the system you can see the section [wireframes](#).

### 3.7.2.8 Prototypes

Before spending time and money on development, you may test your ideas with users and explain the rationale behind a feature or the overall design concept using a prototype (Wikipedia, 2024). Basically, it is a finalized form of wireframes. Below are some prototypes finalized for the system.

#### For Customer

- i. Welcome Page

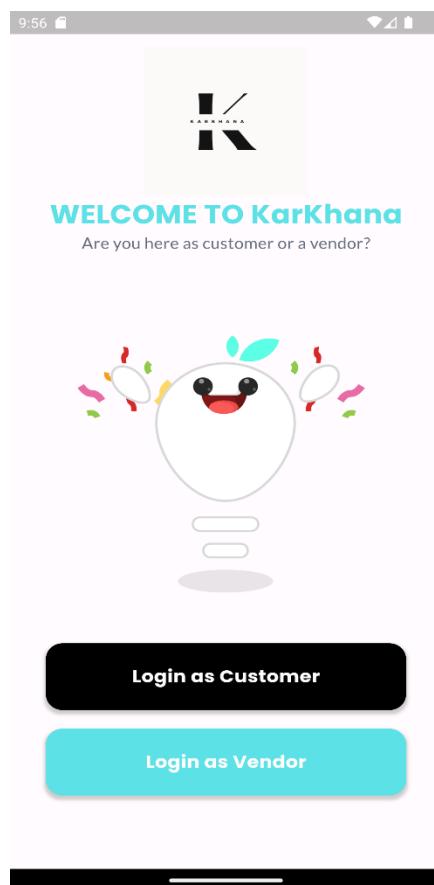


Figure 103: Prototype of Welcome Page.

## ii. Login

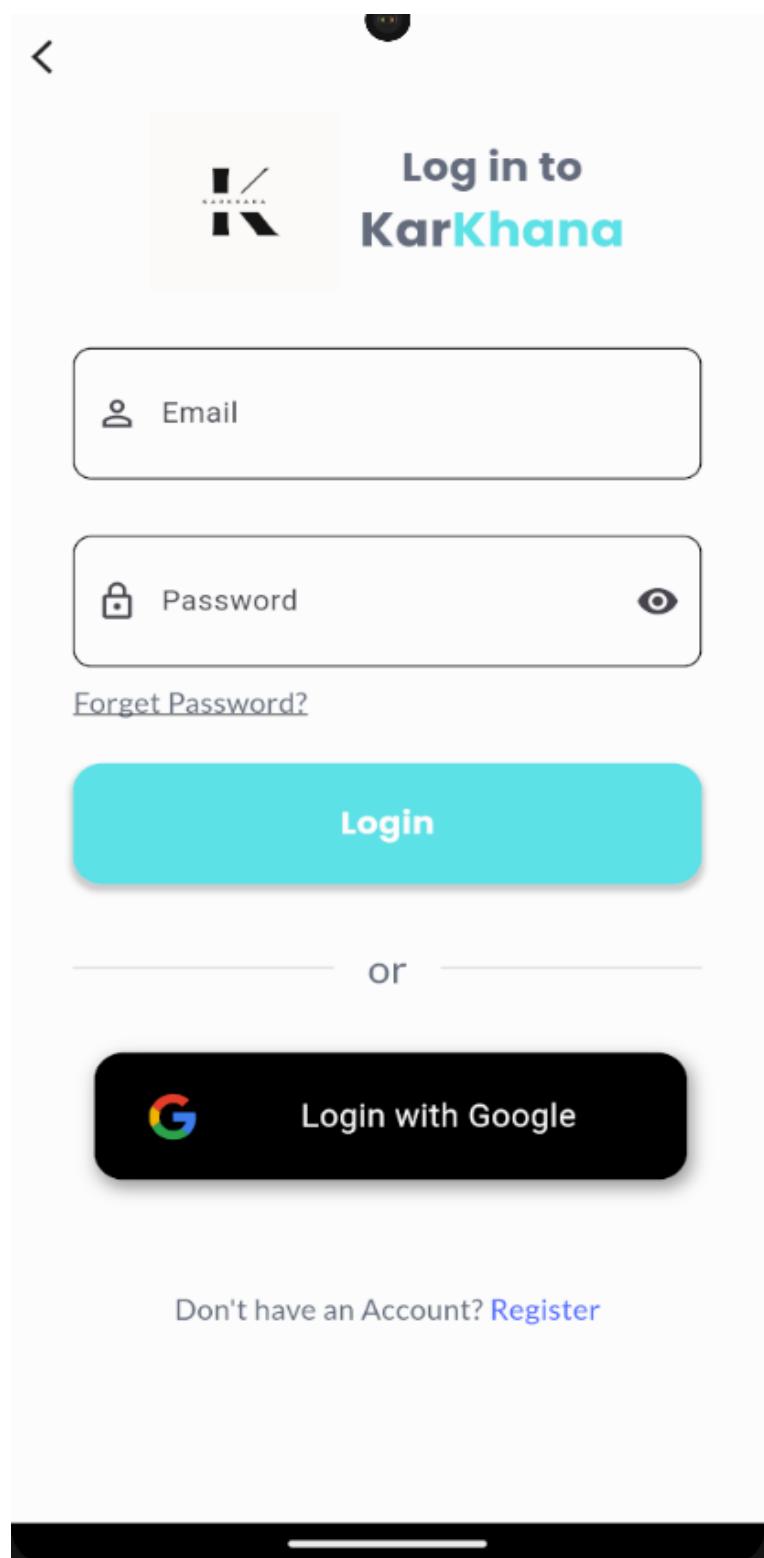


Figure 104: Prototype of Login Page.

iii. Registration Page

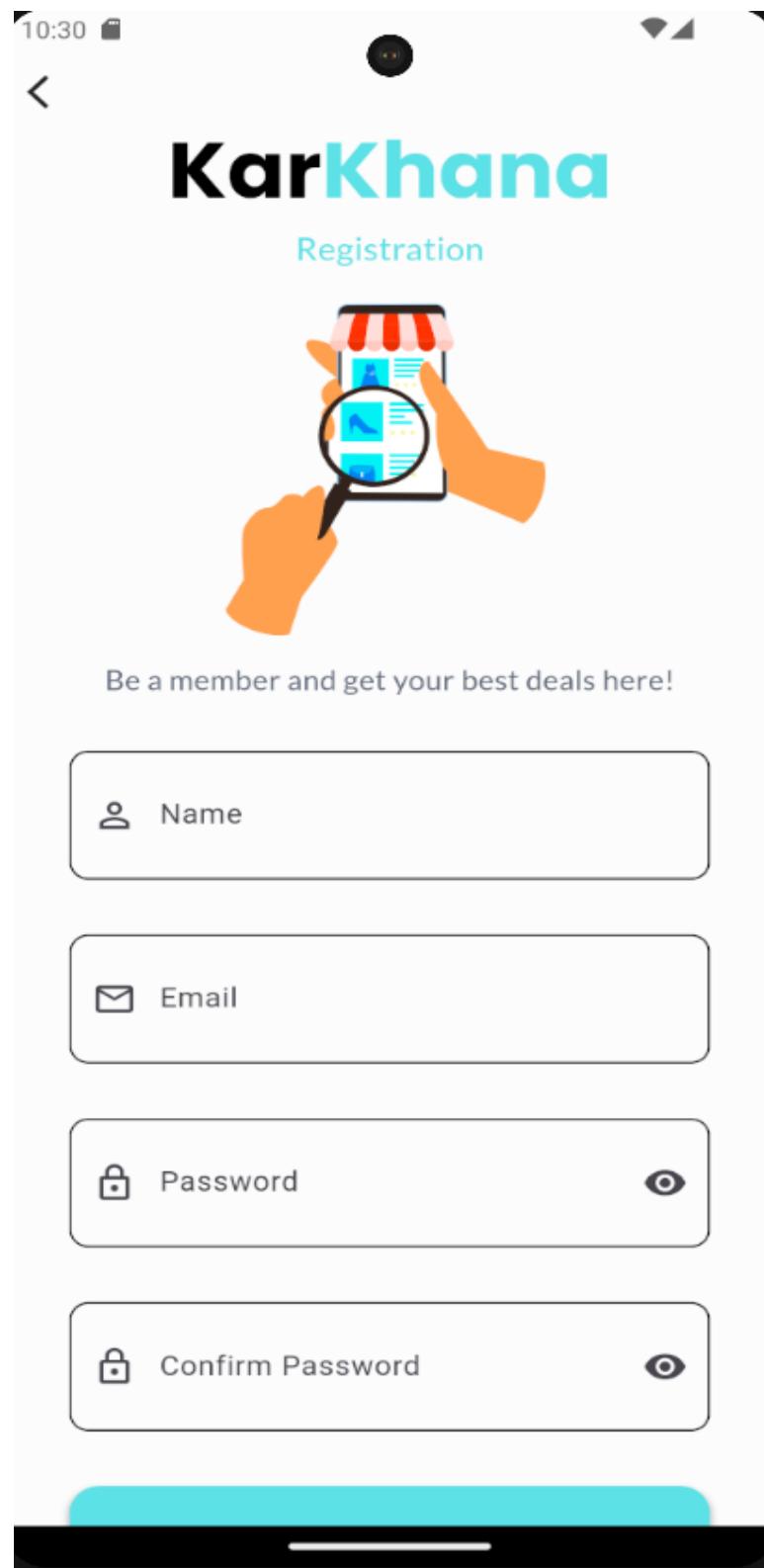


Figure 105: Prototype of Registration Page.

iv. Set Current Location

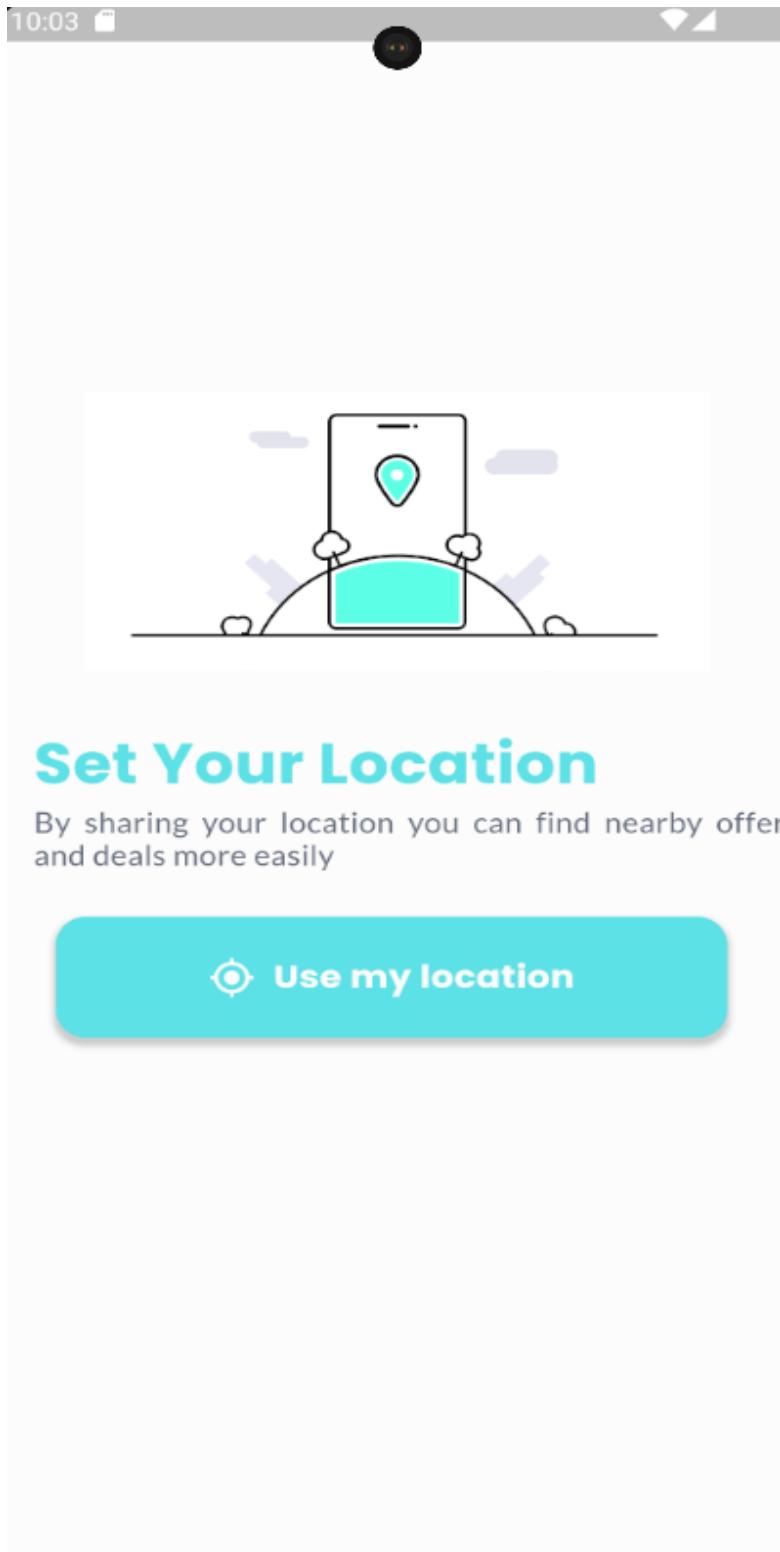


Figure 106: Prototype of set current location page.

## v. Home page

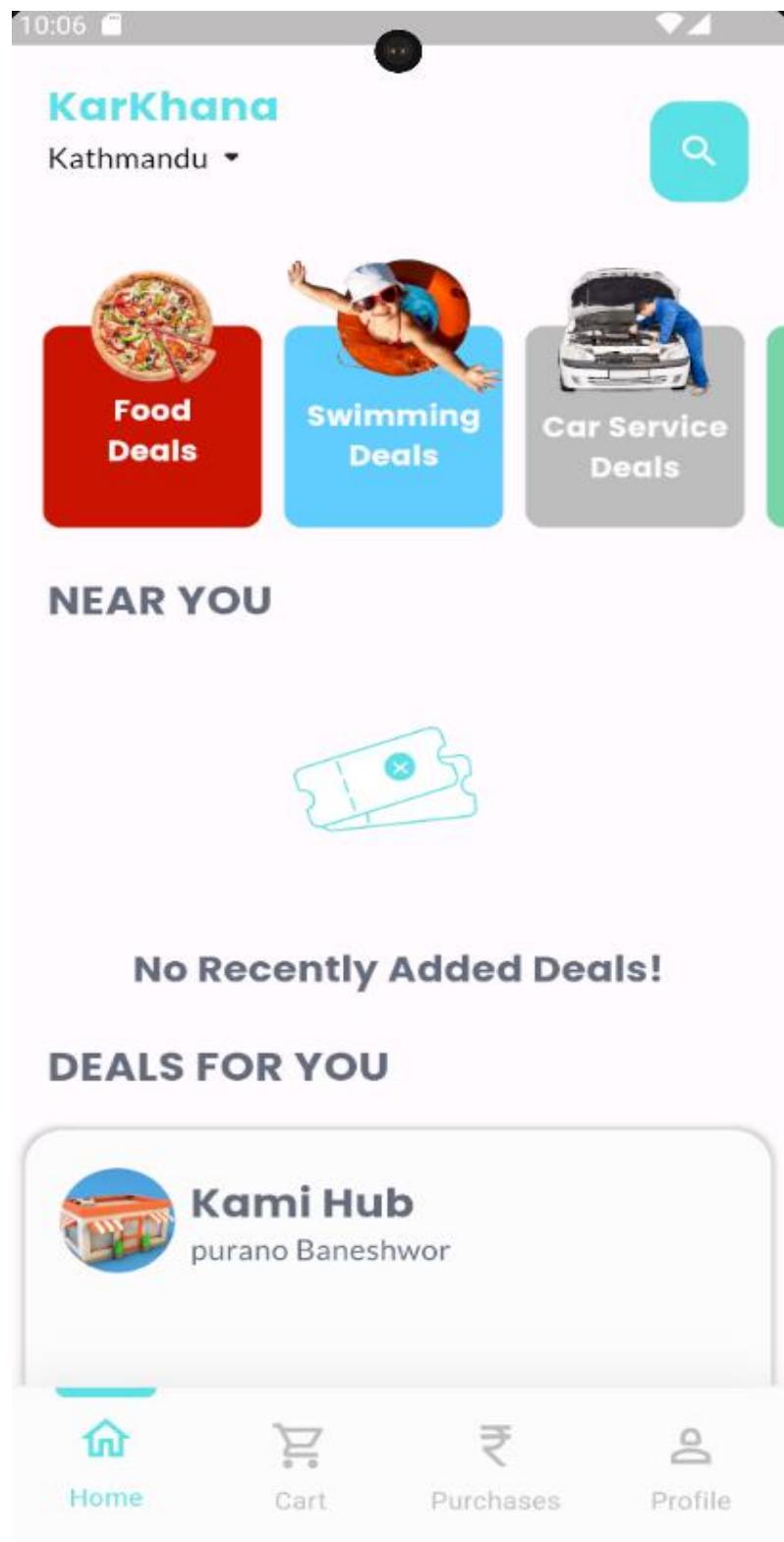


Figure 107: Prototype of Homepage.

## vi. Deal Details

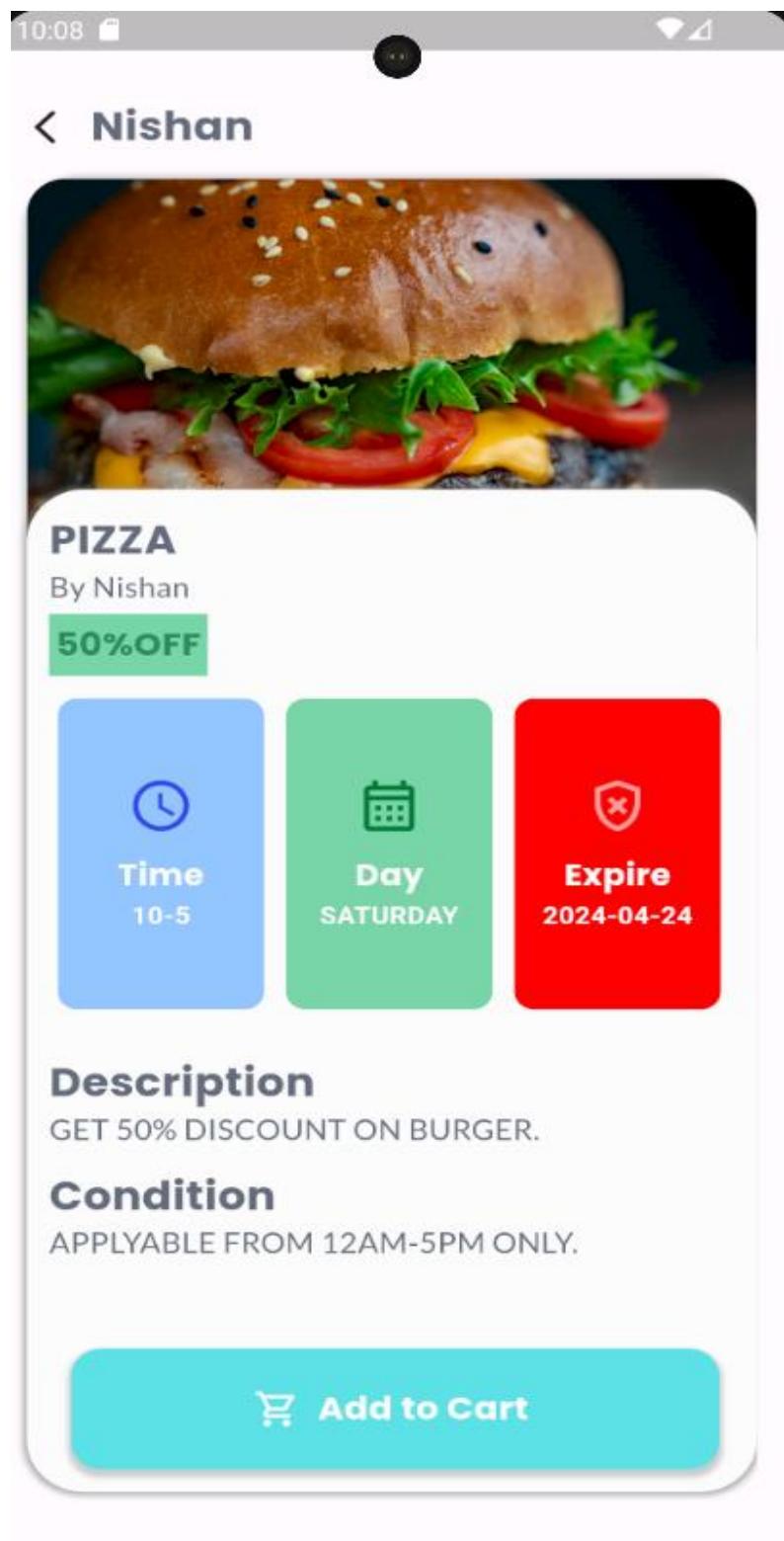


Figure 108: Prototype of Deal Details Page.

## vii. Cart

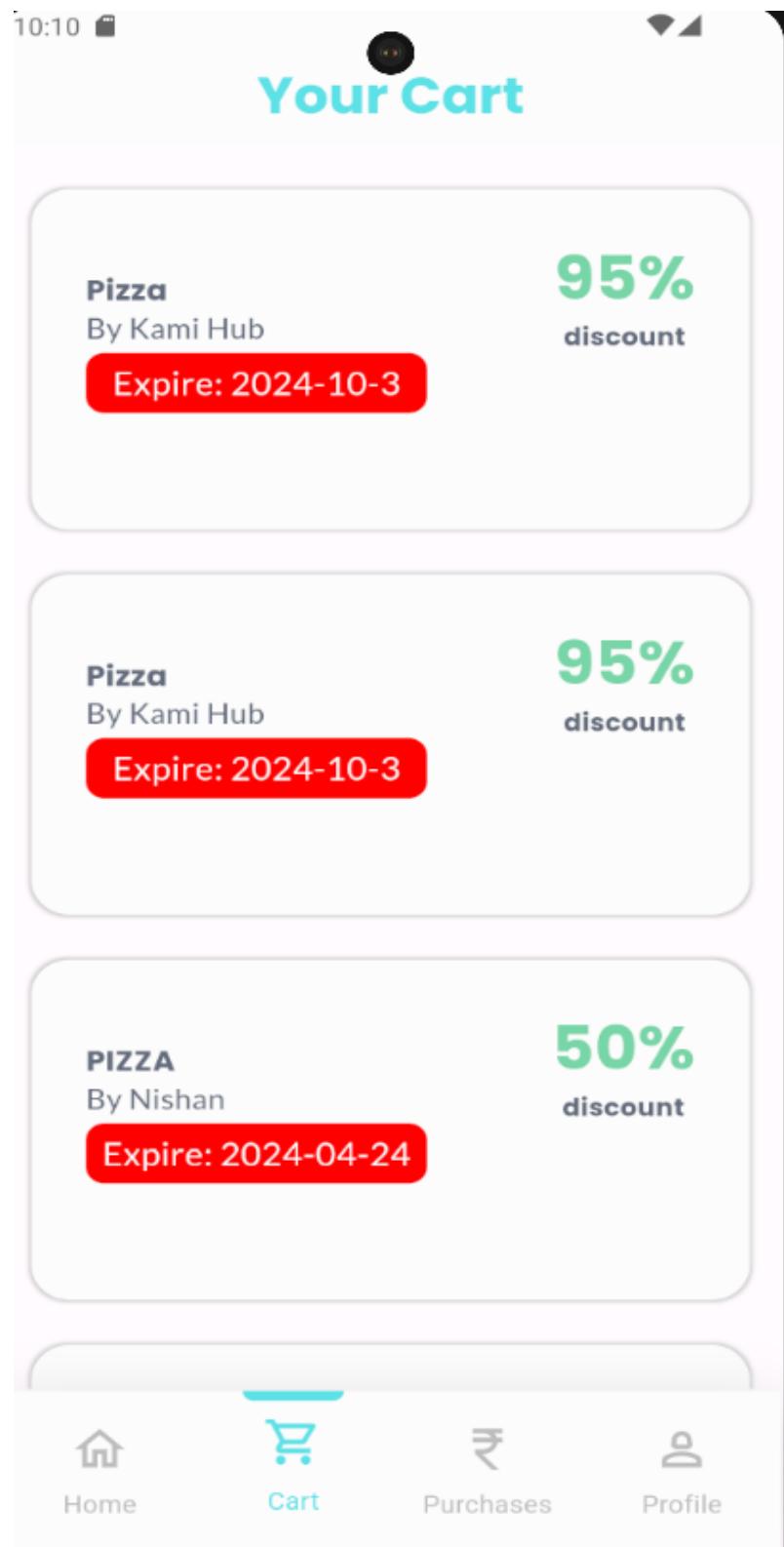


Figure 109: Prototype of Cart page.

## viii. Settings

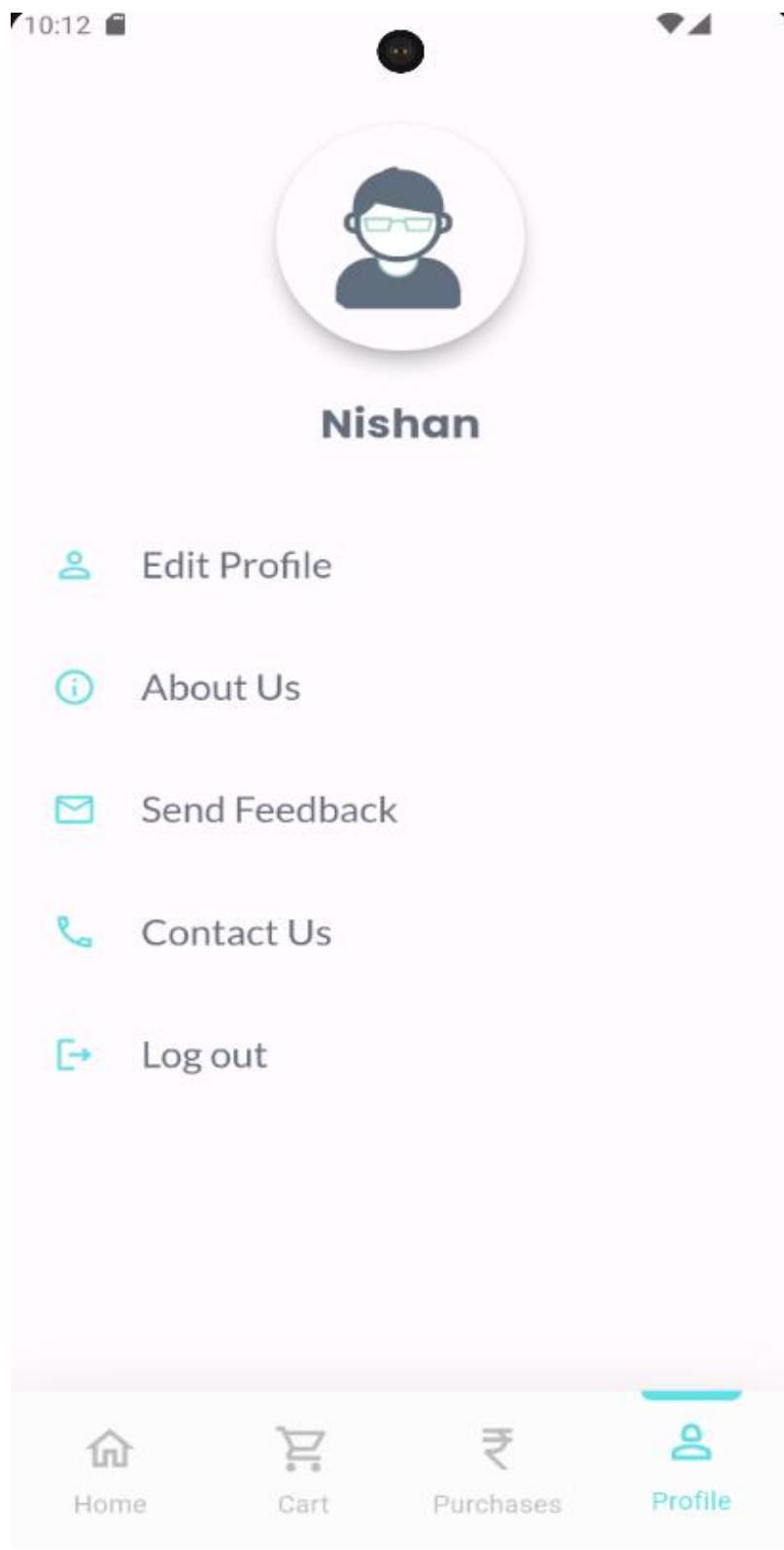


Figure 110: Prototype of Settings Page.

## ix. Active Coupons Page

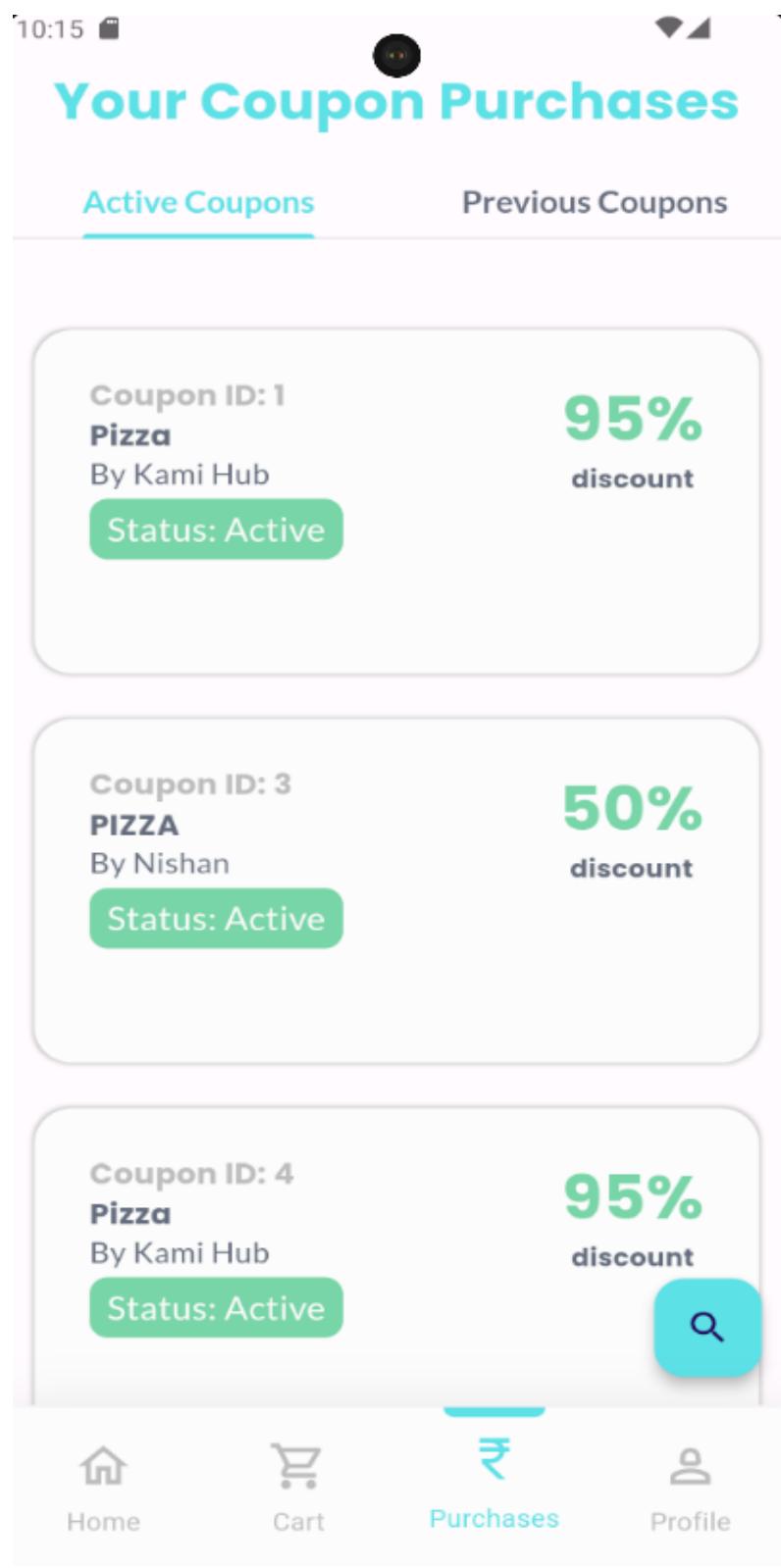


Figure 111: Prototype of Active Coupons Page.

## x. Previous Coupon Page

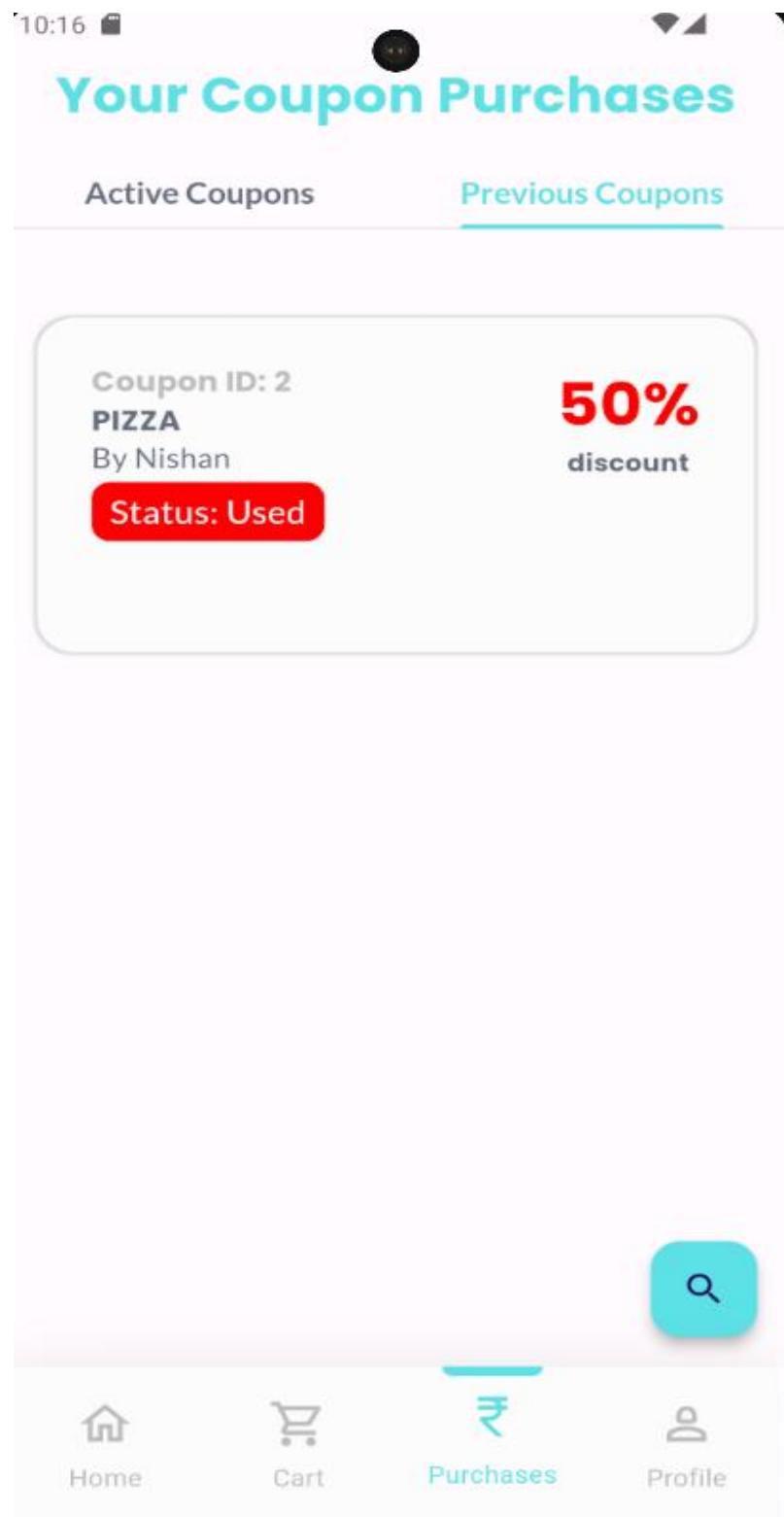


Figure 112: Prototype of Previous Coupon Page.

## xi. Coupon Details Page

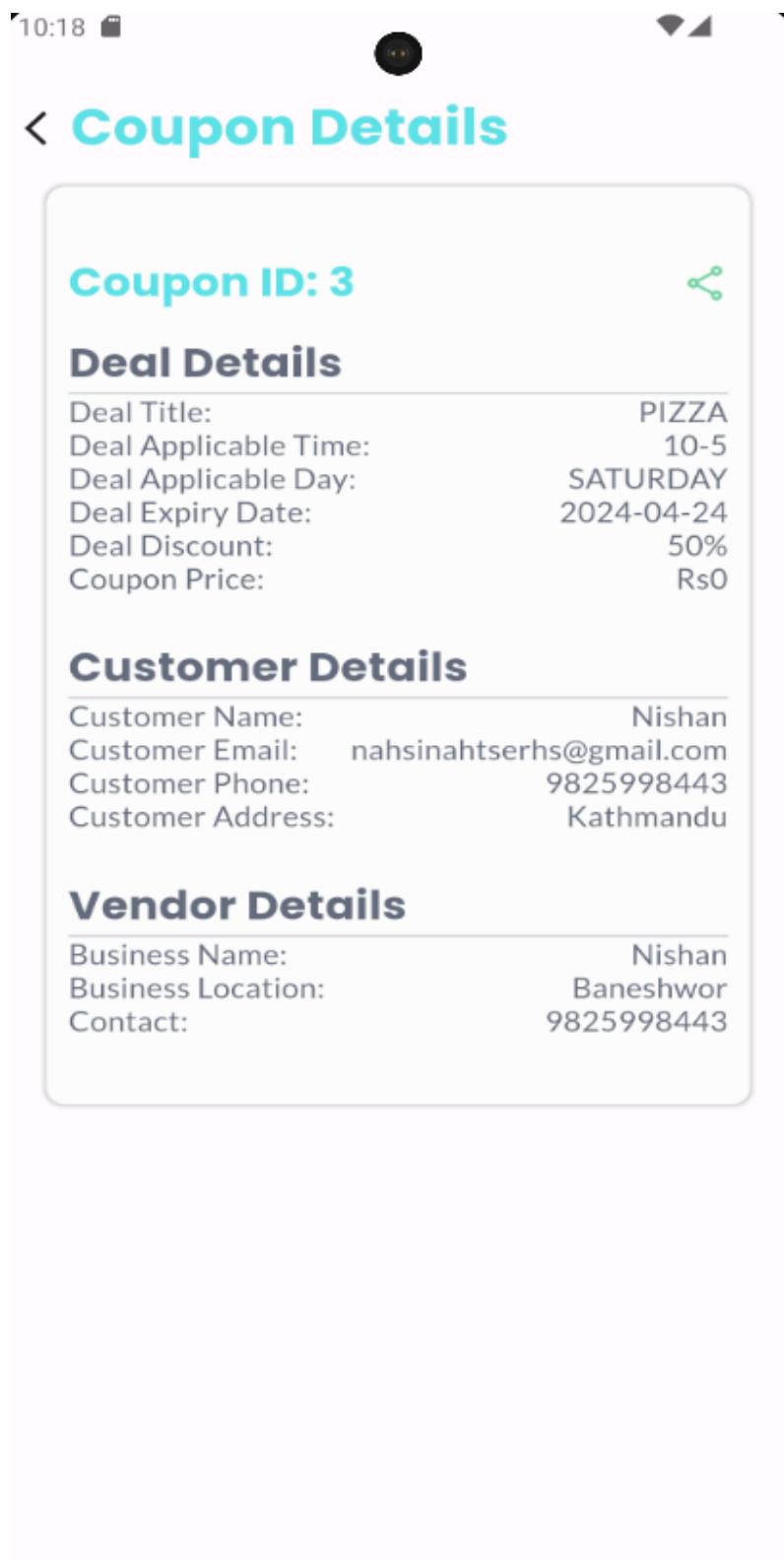


Figure 113: Prototype of Coupon Details Page.

## xii. Share

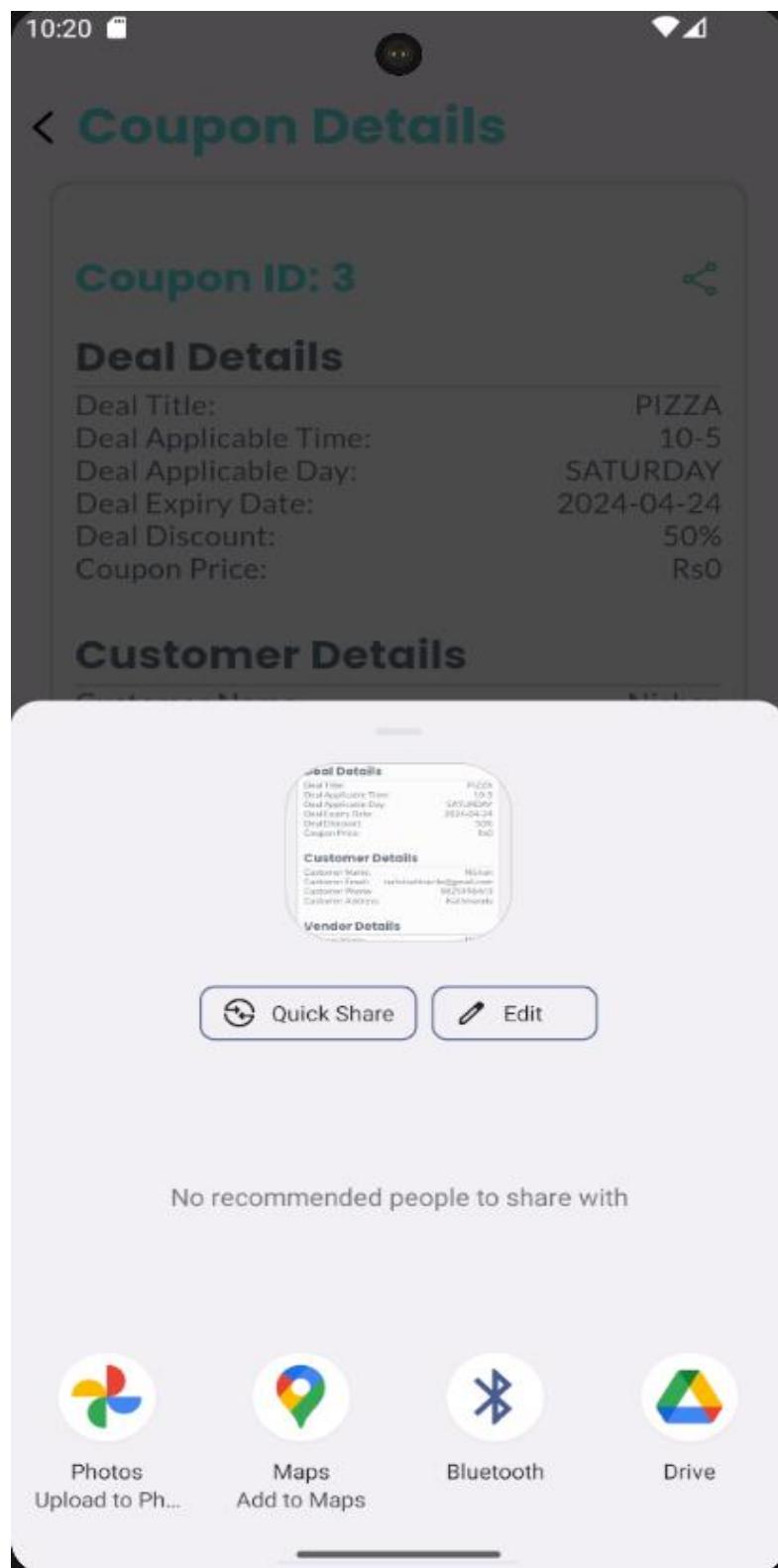


Figure 114: Prototype of Share Page.

## xiii. Payment

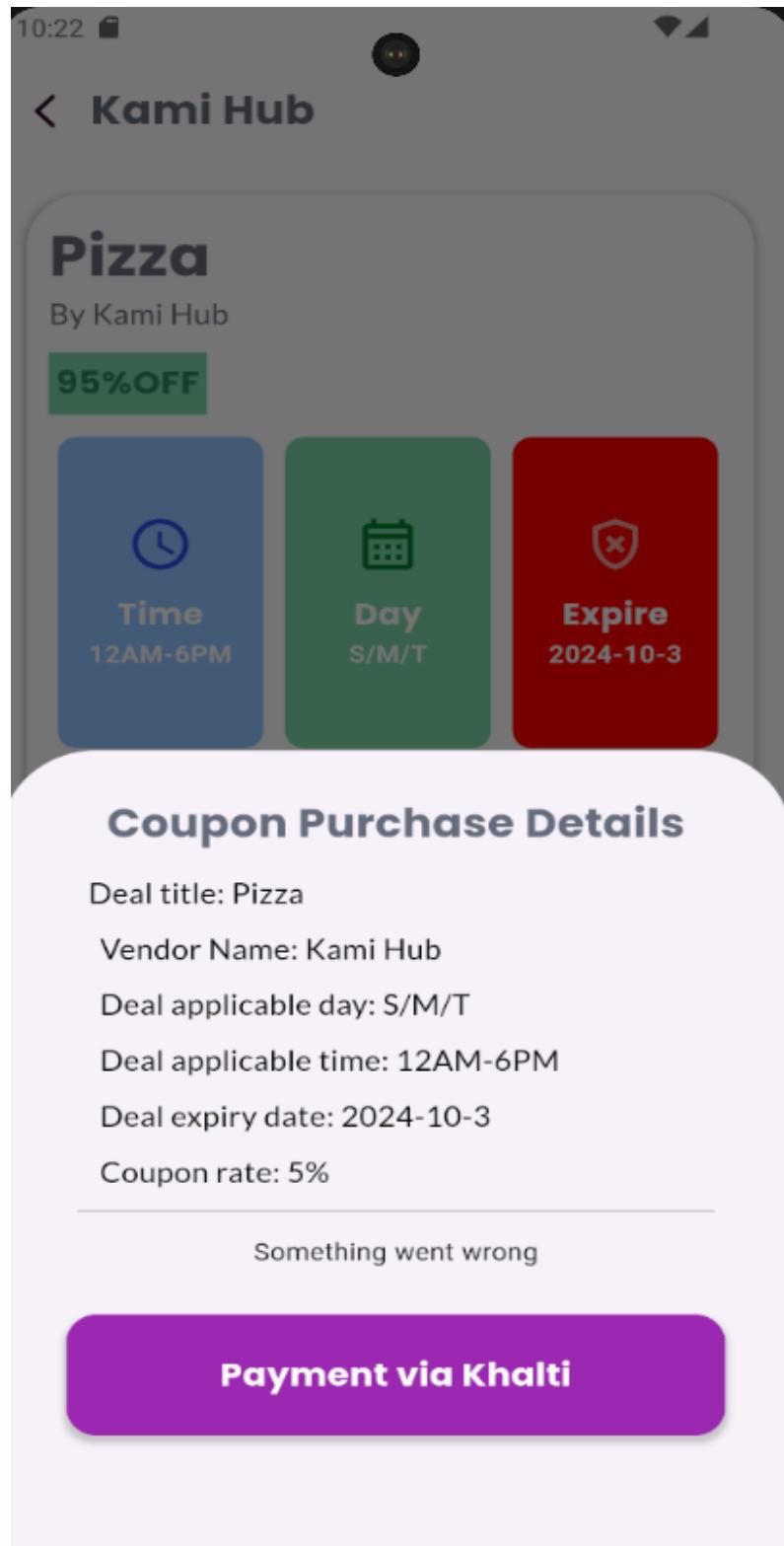


Figure 115: Prototype of Payment Page.

## xiv. Send Feedback

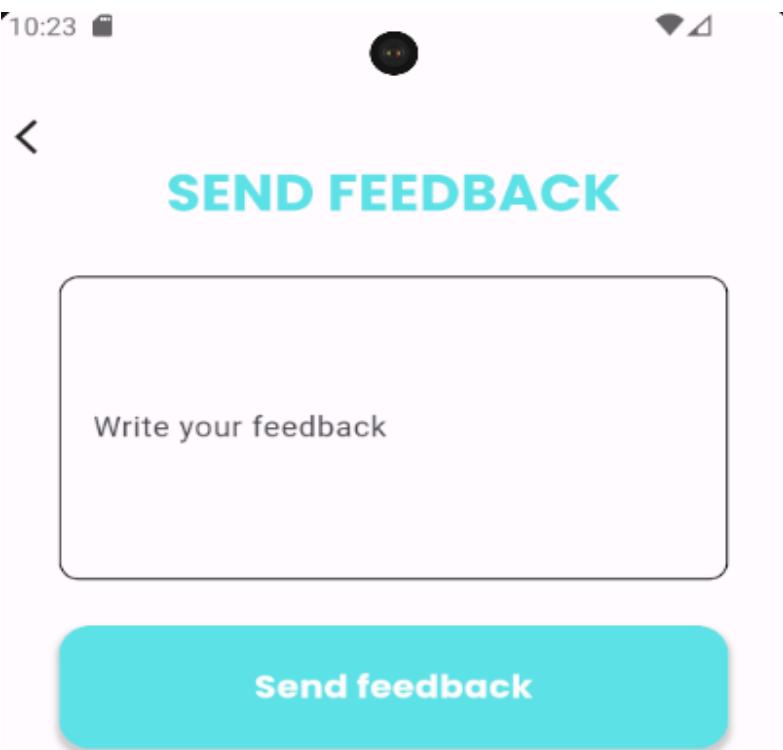


Figure 116: Prototype of Send Feedback.

## xv. Edit Profile

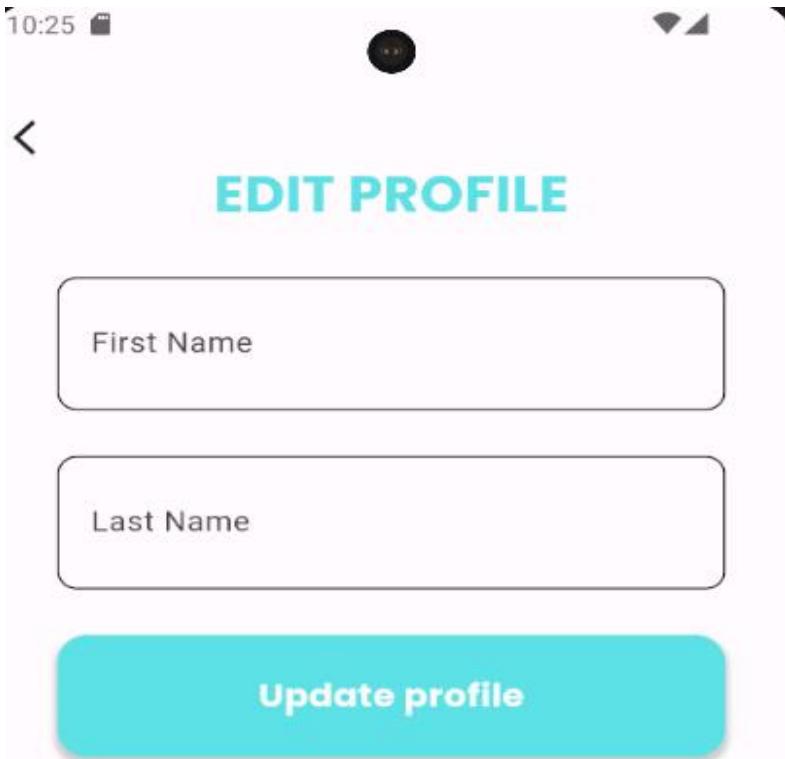


Figure 117: Prototype of Edit Profile Page.

## xvi. About Us

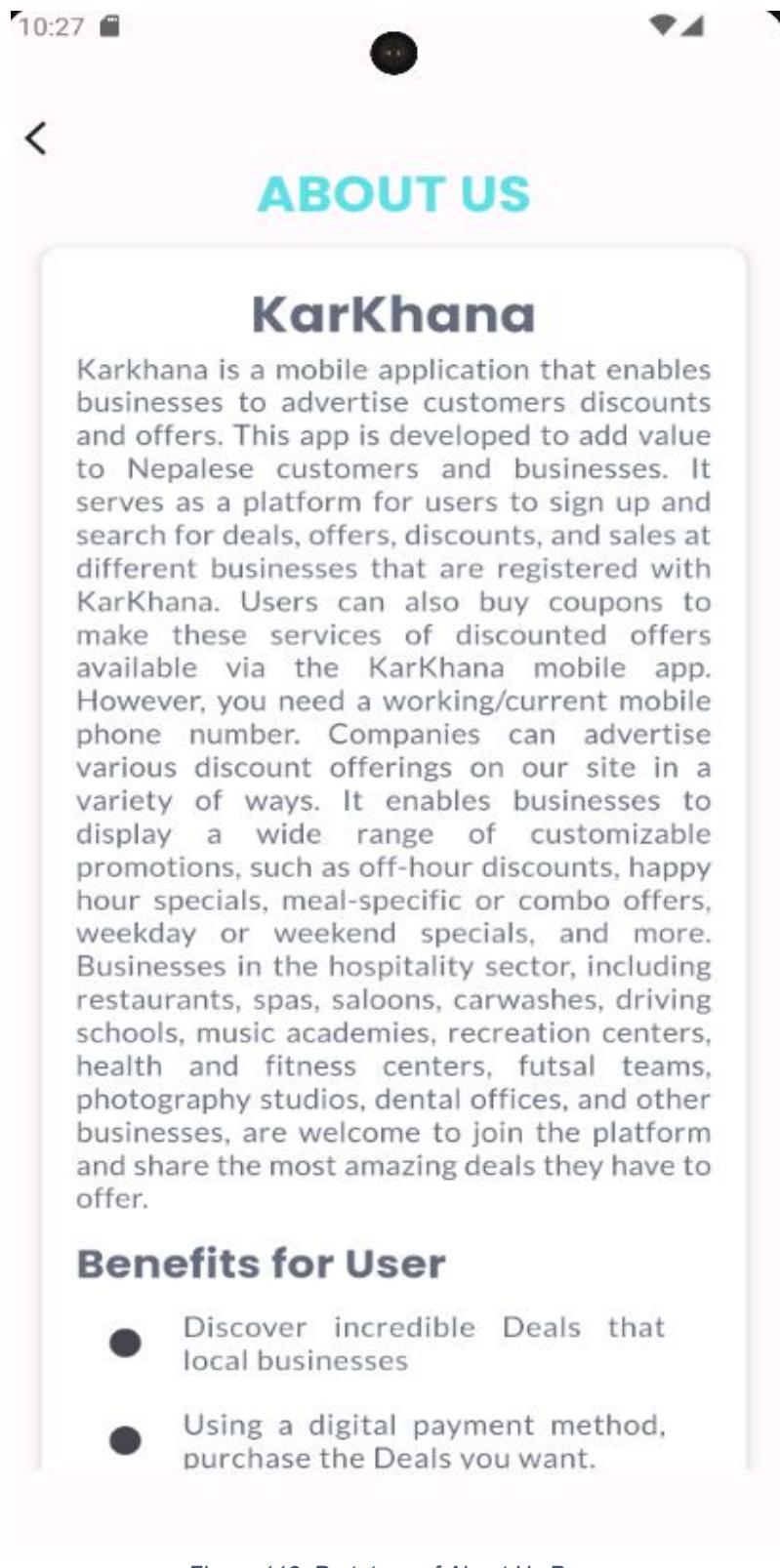


Figure 118: Prototype of About Us Page.

## xvii. Search

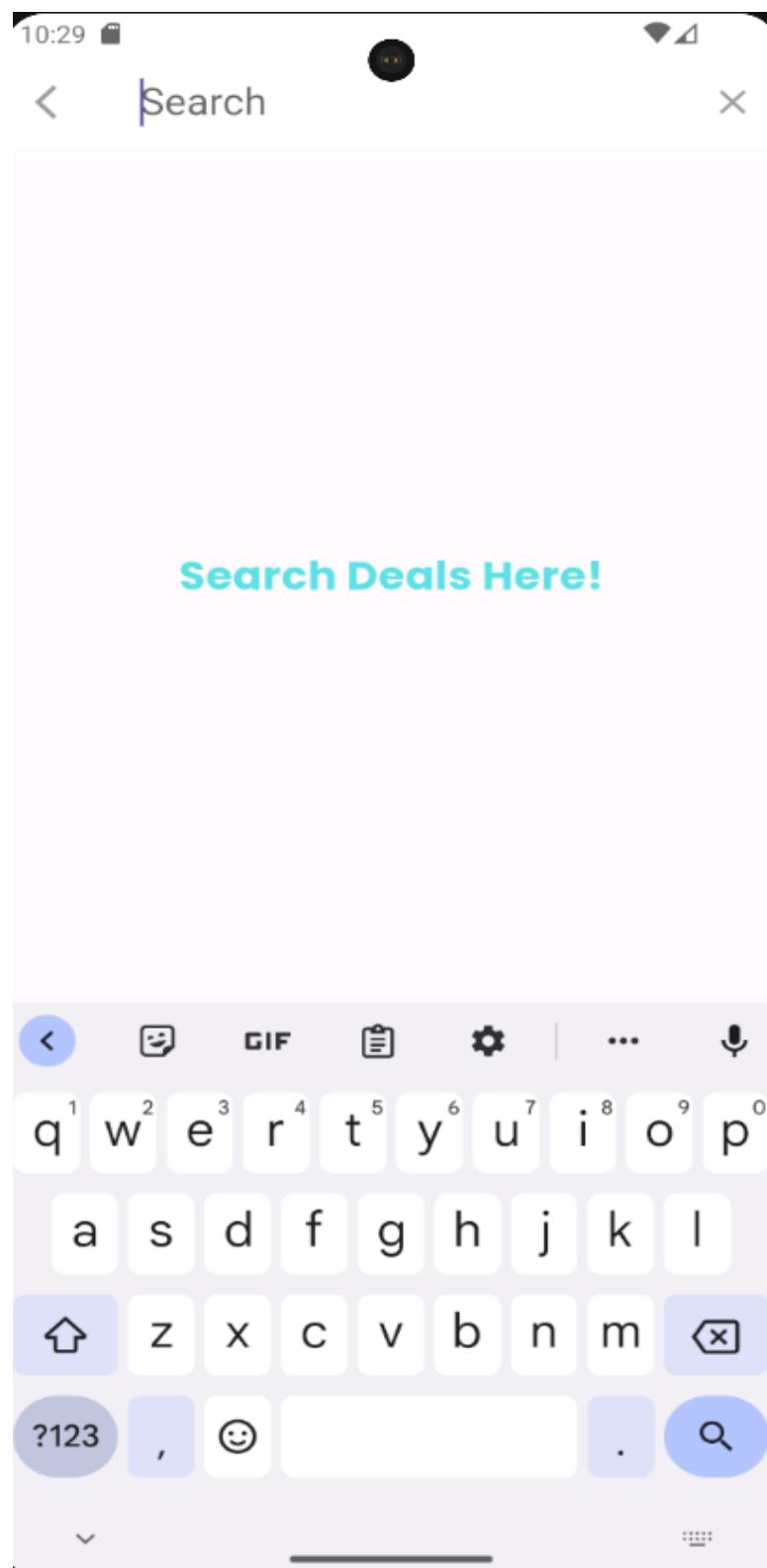


Figure 119: Prototype of Search Page.

## xviii. Forgot Password

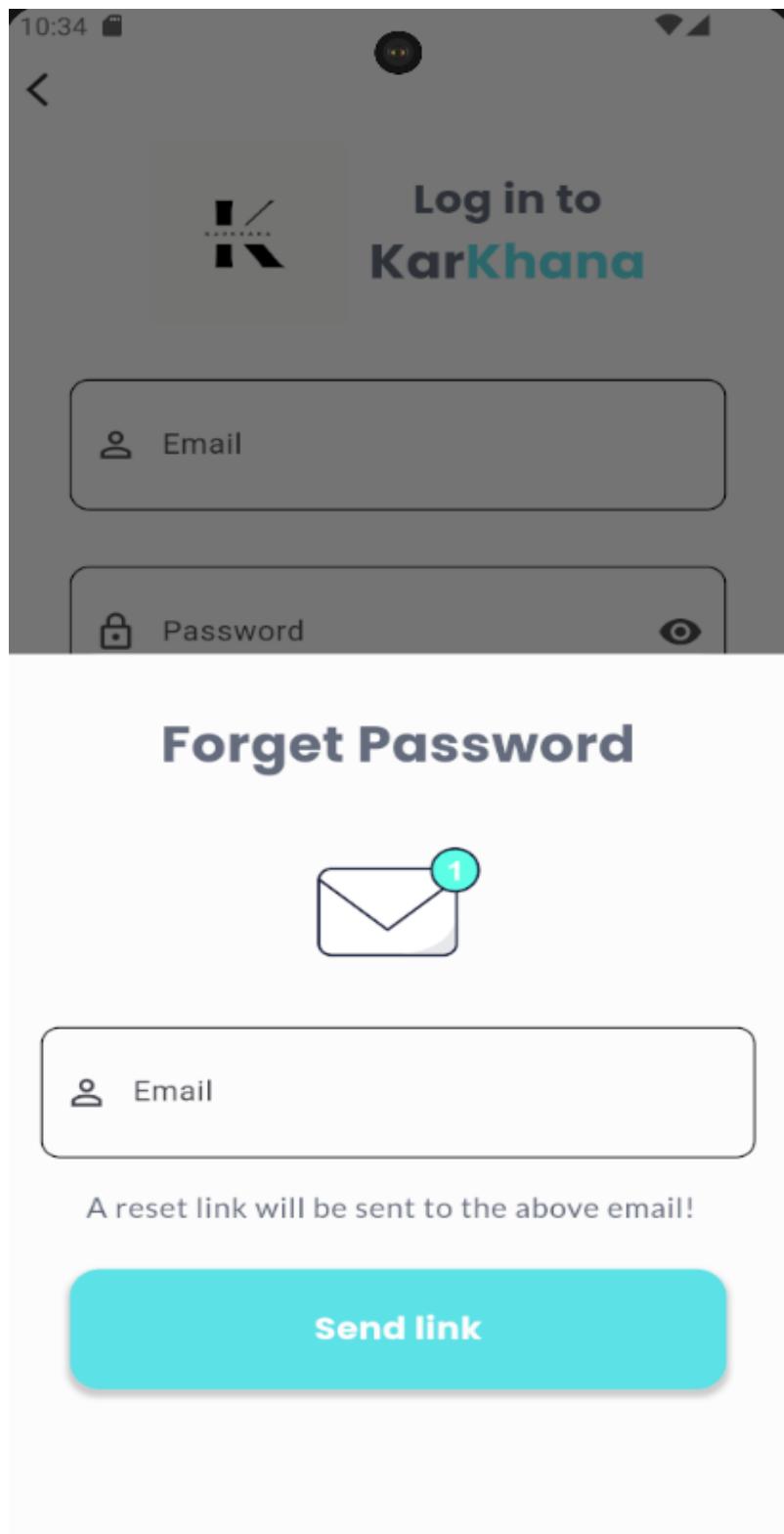


Figure 120: Prototype of Forgot Password.

**For Vendor**

- i. Login

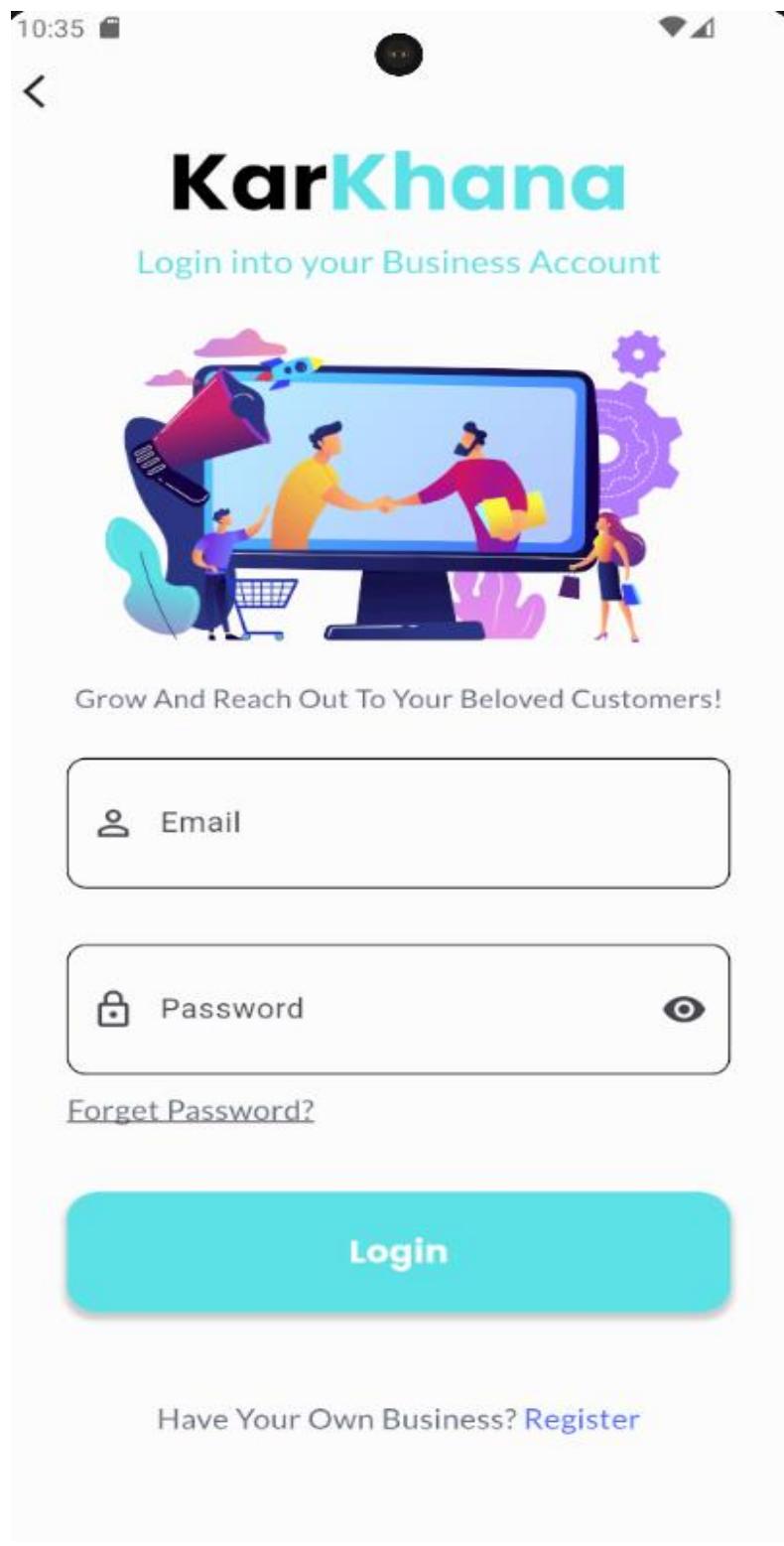


Figure 121: Prototype of Vendor Login.

## ii. Registration Page

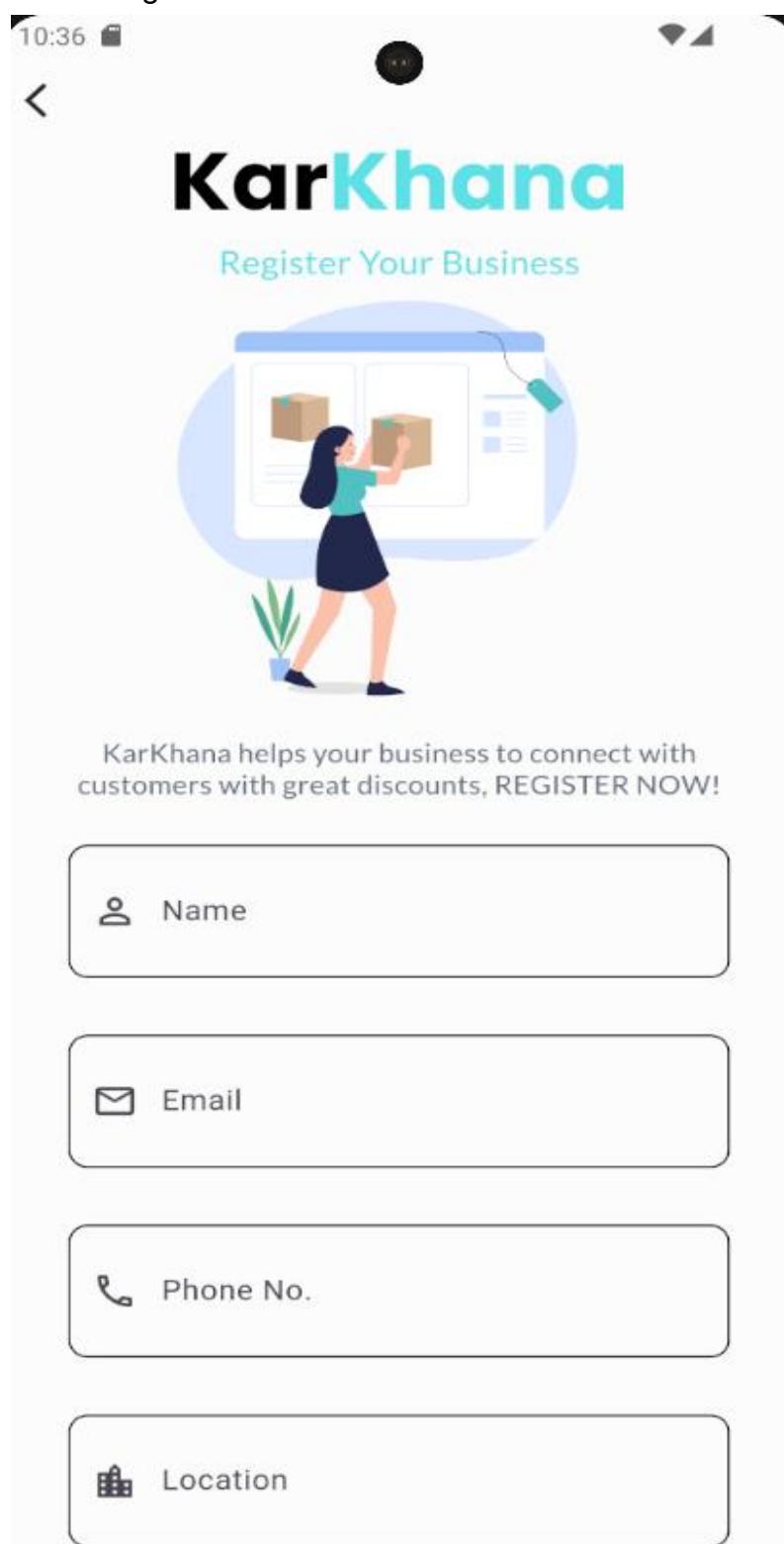


Figure 122: Prototype of Vendor Registration.

## iii. Home Page

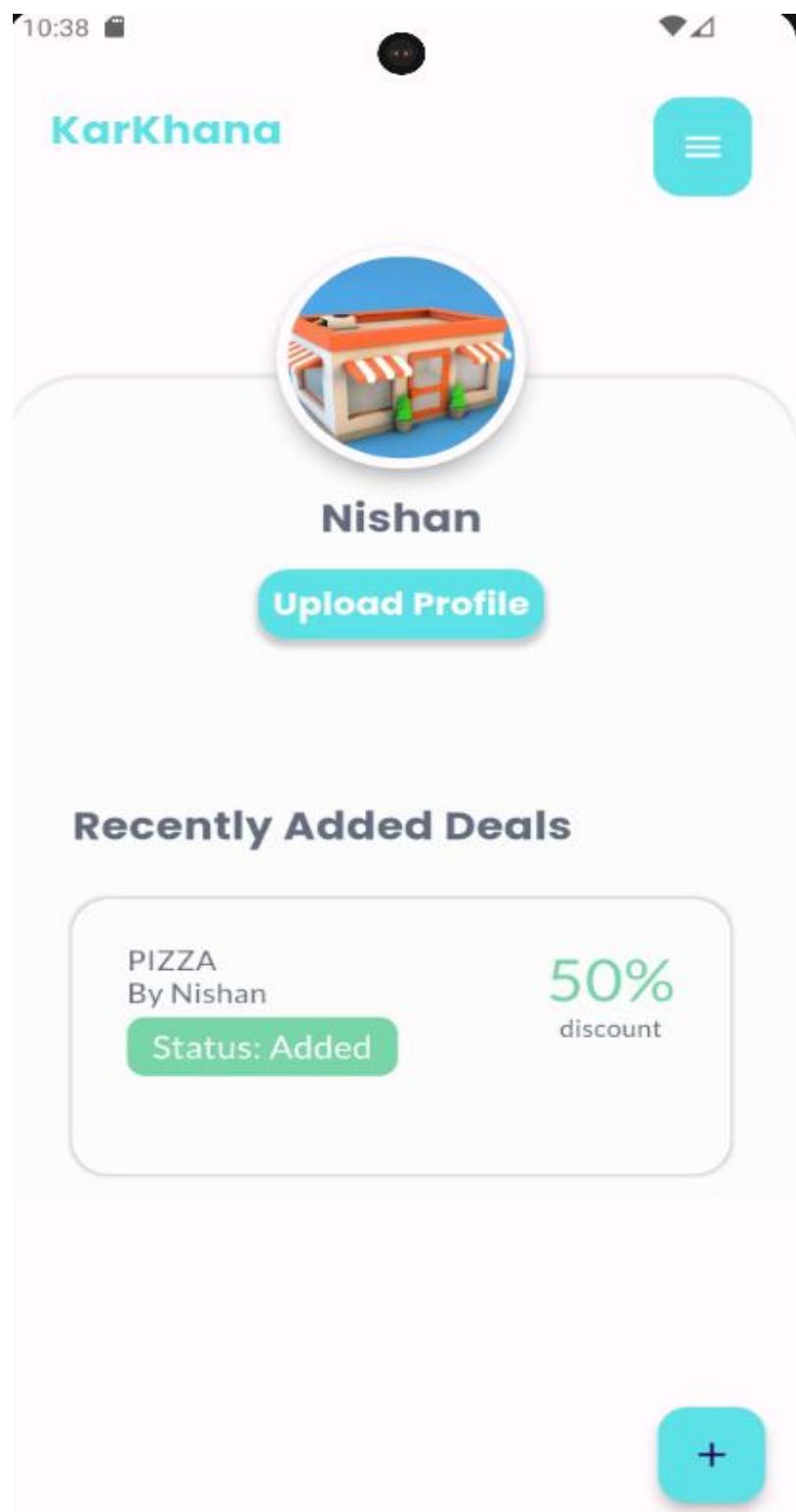


Figure 123: Prototype of Vendor Home Page.

## iv. Add Deals Page

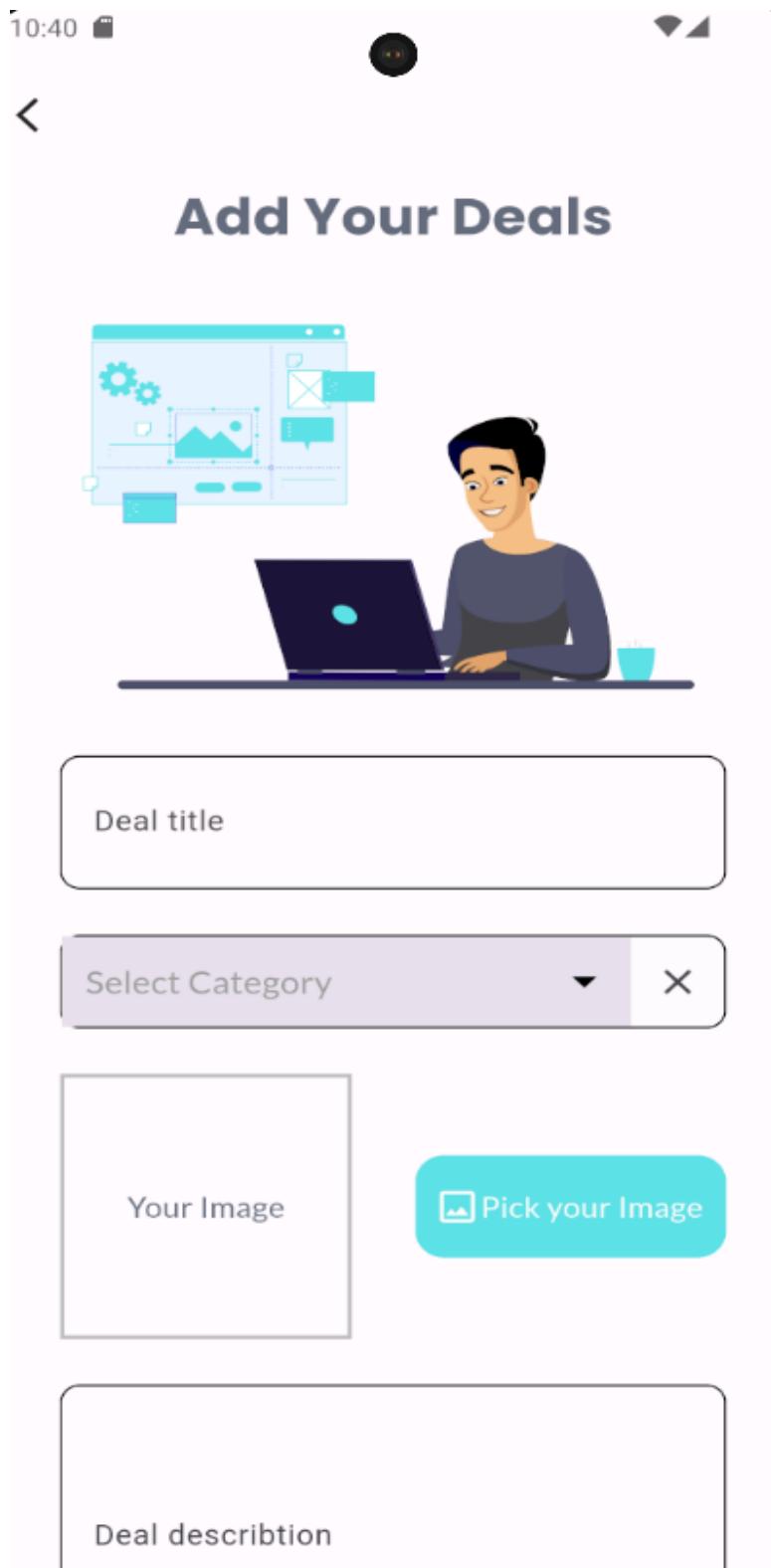


Figure 124: Prototype of Add Deals Page.

## v. Settings Page



Figure 125: Prototype of Settings Page.

## vi. Active Customers

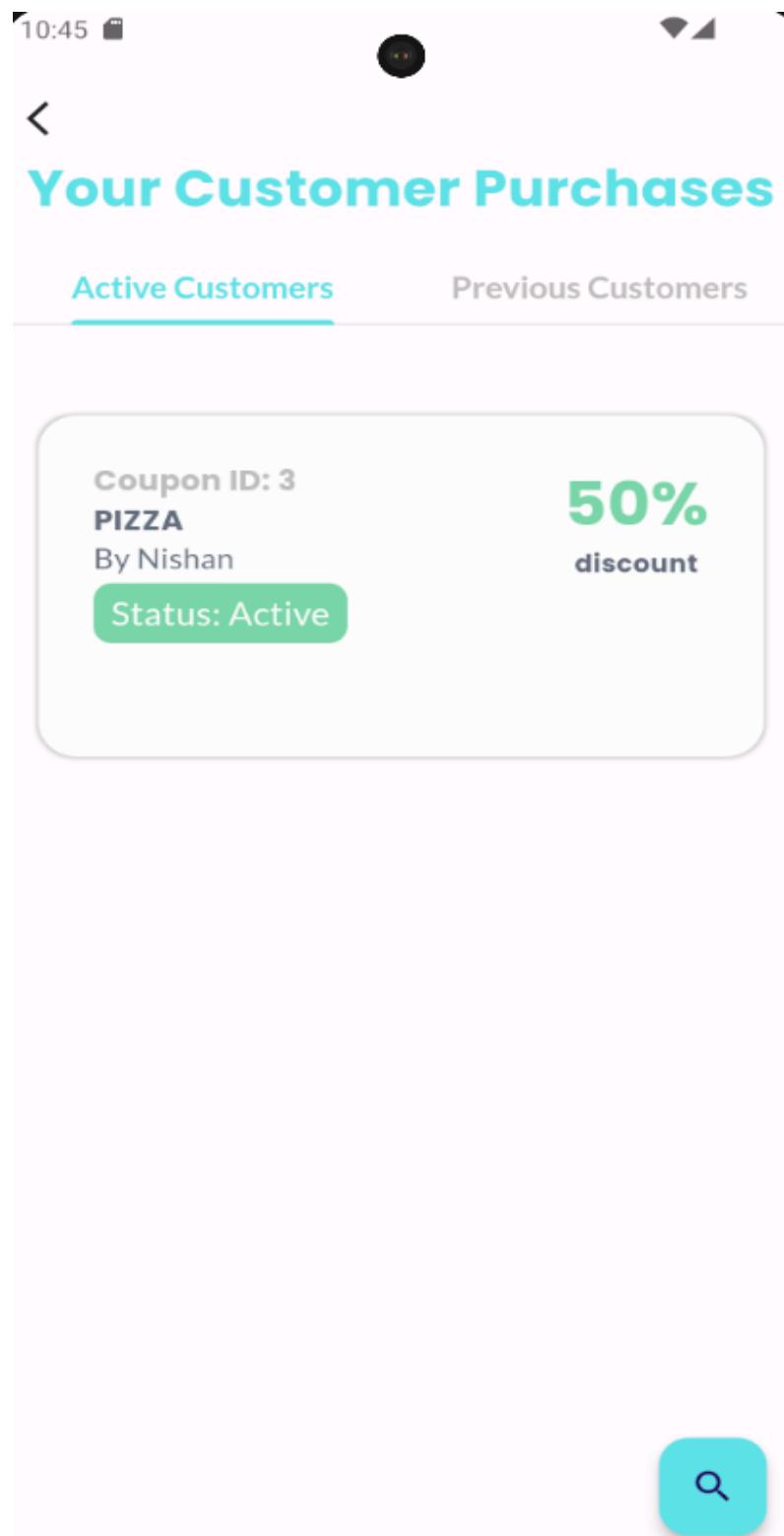


Figure 126: Prototype of Active Coupons.

## vii. Previous Coupon

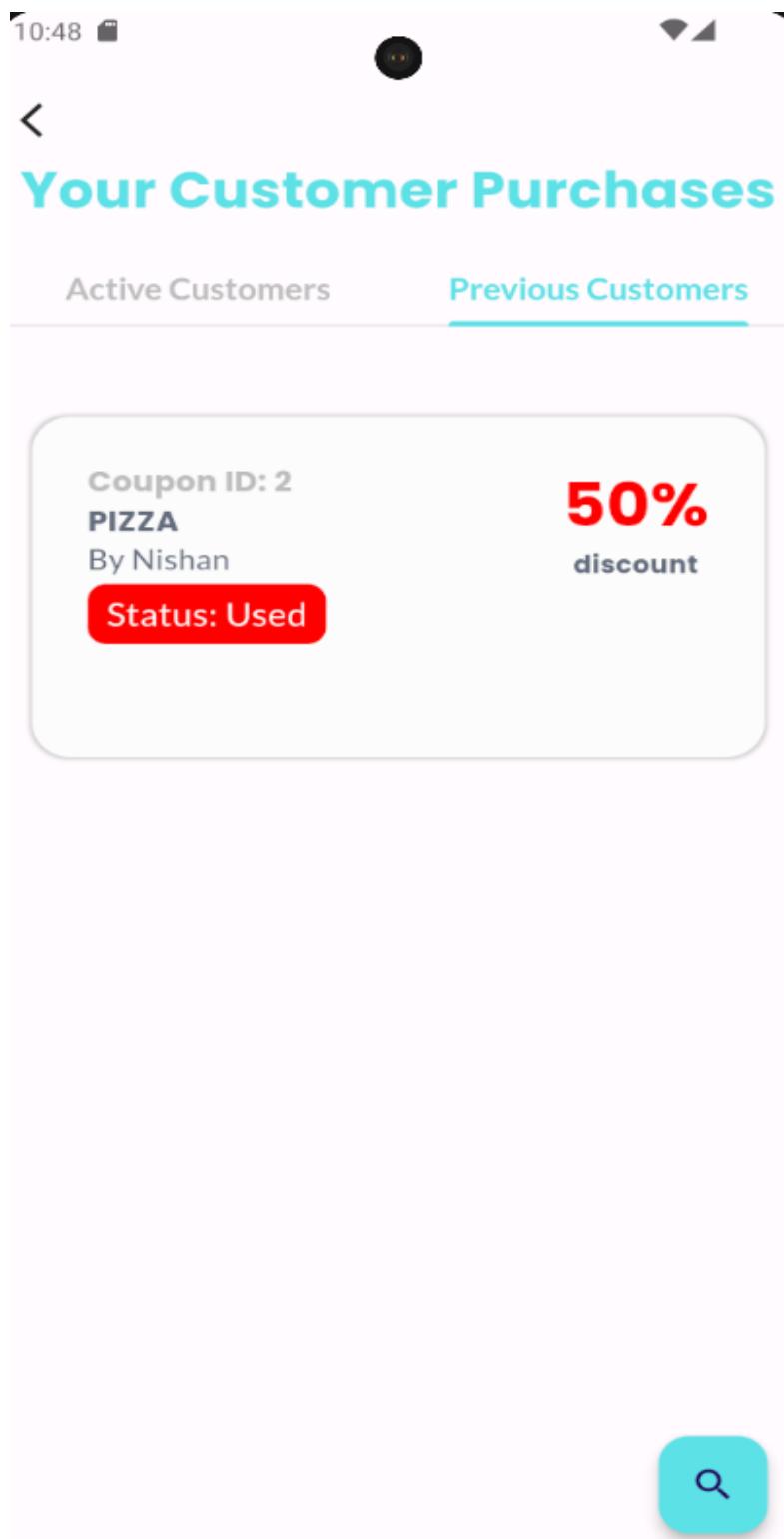


Figure 127: Prototype of Previous Coupon.

## viii. Coupon Details

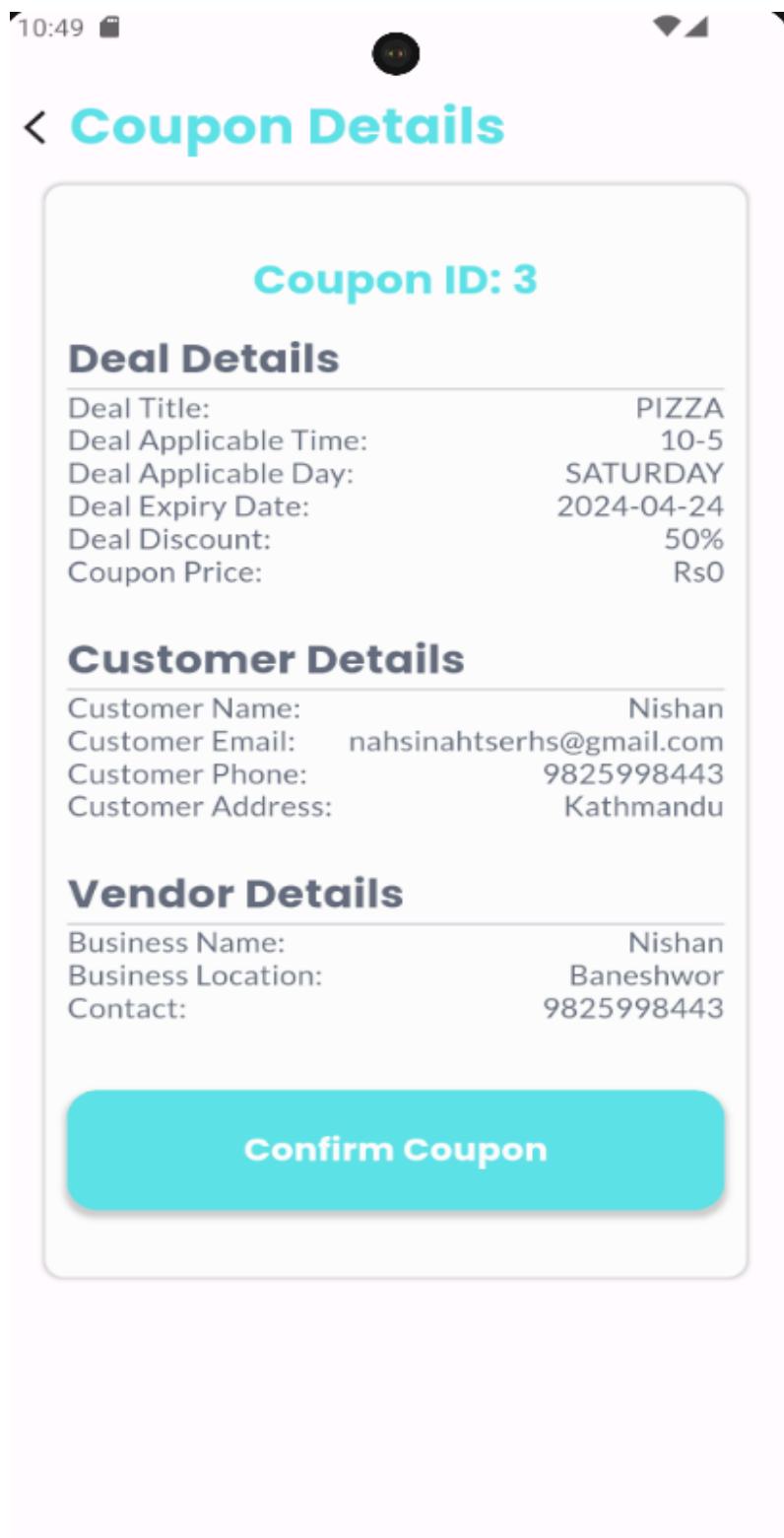


Figure 128: Prototype of Coupon Details.

### 3.7.3 Construction

#### 3.7.3.1 Database Design

A database is a planned grouping of material that has been arranged and is often kept electronically in a computer system. A database management system (DBMS) often has control over a database. The term "database system," which is sometimes abbreviated to "database," refers to the combination of the data, the DBMS, and the applications that are connected to it.

The ERD from the design phase is used to produce the models for the database of the KarKhana application (Oracle, 2023).

```
from django.db import models
from django.contrib.auth.models import AbstractUser, Group, Permission
```

Figure 129: Importing the libraries.

```
class User(AbstractUser):
    type = ((1, 'Customer'),
            (2, 'Vendor'))
    user_type = models.IntegerField(choices=type, default= 1)
    username = models.CharField(max_length=55, unique=False)
    full_name = models.CharField(max_length=50, blank=True)
    location = models.CharField(max_length=100, blank=True)
    phonenumer = models.CharField(max_length=100, blank=True)
    vendor_about = models.TextField(blank=True)
    vendor_opening_days = models.CharField(max_length=12, blank=True)
    vendor_opening_time = models.CharField(max_length=12, blank=True)
    groups = models.ManyToManyField(Group, related_name='karkhana_users')
    user_permissions = models.ManyToManyField(Permission, related_name='karkhana_users_permissions')

    def __str__(self):
        return self.full_name
```

Figure 130: Model for User.

```

class VendorRequest(models.Model):
    vendor_email = models.CharField(max_length=50, blank=True)
    vendor_BusinessName = models.CharField(max_length=50, blank=True)
    vendor_phoneNumber = models.CharField(max_length=50, blank=True)
    vendor_Location = models.CharField(max_length=50, blank=True)
    vendor_about = models.TextField(blank=True)
    vendor_opening_days = models.CharField(max_length=12, blank=True)
    vendor_opening_time = models.CharField(max_length=12, blank=True)

    def __str__(self):
        return self.vendor_BusinessName

```

Figure 131: Model for Vendor Request for registration.

```

class VendorBusinessDetails(models.Model):
    vendor_email = models.CharField(max_length=50, blank=True)
    vendor_BusinessName = models.CharField(max_length=50, blank=True)
    vendor_phoneNumber = models.CharField(max_length=50, blank=True)
    vendor_Location = models.CharField(max_length=50, blank=True)
    vendor_id = models.IntegerField(primary_key=True, default="0000")
    vendor_image = models.ImageField(null = True, blank=True)
    vendor_about = models.TextField(blank=True)
    vendor_opening_days = models.CharField(max_length=12, blank=True)
    vendor_opening_time = models.CharField(max_length=12, blank=True)

    def __str__(self):
        return self.vendor_BusinessName

```

Figure 132: Model of Vendor for uploading business profile.

```

class EmailNotificationModel(models.Model):
    notification_email = models.CharField(max_length=50, blank=True)
    notification_message = models.CharField(max_length=255, blank=True)

    def __str__(self):
        return self.notification_email

```

Figure 133: Model for Email notification.

```

class Deals(models.Model):
    user = models.ForeignKey(VendorBusinessDetails, on_delete=models.CASCADE)
    BusinessName = models.CharField(max_length=50, blank=True)
    Business_ProfileImage = models.ImageField(null= True, blank= True)
    BusinessLocation = models.CharField(max_length=50, blank=True, null=True)
    Business_Phone = models.CharField(max_length = 50, blank=True)
    deal_title = models.CharField(max_length=70)
    deal_price = models.IntegerField(blank= True, default= 0)
    deal_photo = models.ImageField(null= True, blank=True)
    deal_category = models.CharField(max_length=20)
    deal_desc = models.TextField()
    deal_condition = models.TextField(blank=True)
    deal_discount = models.IntegerField(blank= True, default= 0)
    deal_applicable_time = models.CharField(max_length=10)
    deal_applicable_day = models.CharField(max_length=10)
    deal_expiry_date = models.CharField(max_length=12)

    def __str__(self):
        return self.deal_title

```

Figure 134: Model for Deals

```

class Cart(models.Model):
    deal_title = models.CharField(max_length=70)
    deal_price = models.IntegerField(blank= True, default= 0)
    deal_photo = models.ImageField(null= True, blank=True)
    deal_category = models.CharField(max_length=20)
    deal_desc = models.TextField()
    deal_condition = models.TextField(blank=True)
    deal_discount = models.IntegerField(blank= True, default= 0)
    deal_applicable_time = models.CharField(max_length=10)
    deal_applicable_day = models.CharField(max_length=10)
    deal_expiry_date = models.CharField(max_length=12)
    Customer_User = models.ForeignKey(User, on_delete=models.CASCADE)
    Vendor_User = models.ForeignKey(VendorBusinessDetails, on_delete=models.CASCADE)
    VendorBusiness_Name = models.CharField(max_length = 50, blank=True)
    VendorBusiness_Location = models.CharField(max_length = 50, blank=True)
    VendorBusiness_ProfileImage = models.ImageField(null= True, blank= True)
    Vendor_Phone = models.CharField(max_length = 50, blank=True)

    def __str__(self):
        return self.deal_title

```

Figure 135: Model for Cart.

```
class Charges(models.Model):
    coupon_rate = models.IntegerField(blank=True, default=0)
```

Figure 136: Model for Coupon rate.

```
class CouponActive(models.Model):
    deal_title = models.CharField(max_length=70, blank=True)
    deal_photo = models.ImageField(null=True, blank=True)
    deal_discount = models.IntegerField(blank=True, default=0)
    deal_applicable_time = models.CharField(max_length=10, blank=True)
    deal_applicable_day = models.CharField(max_length=10, blank=True)
    deal_expiry_date = models.CharField(max_length=12, blank=True)
    Customer_User = models.ForeignKey(User, on_delete=models.CASCADE)
    Vendor_User = models.ForeignKey(VendorBusinessDetails, on_delete=models.CASCADE)
    VendorBusiness_Name = models.CharField(max_length=50, blank=True)
    VendorBusiness_Location = models.CharField(max_length=50, blank=True)
    Vendor_Phone = models.CharField(max_length=50, blank=True)
    Total_Coupon_Price = models.IntegerField(blank=True, default=0)
    Customer_Name = models.CharField(max_length=50, blank=True)
    Customer_Address = models.CharField(max_length=100, blank=True)
    Customer_Phonenumber = models.CharField(max_length=50, blank=True)
    Customer_Email = models.CharField(max_length=50, blank=True)
    Status = (
        ('Active', 'ACTIVE'),
        ('Used', 'USED'),
    )
    coupon_Status = models.CharField(max_length=20, choices=Status, default='Active')

    def __str__(self):
        return self.Customer_Name
```

Figure 137: Model for Coupon.

```
class Feedback(models.Model):
    Customer_user = models.ForeignKey(User, null=True, blank=True, on_delete=models.CASCADE)
    customer_id = models.IntegerField(blank=False, null=True)
    Customer_Name = models.CharField(max_length=70, blank=True)
    Customer_Email = models.CharField(max_length=70, blank=True)
    Feedbacks = models.CharField(max_length=500, blank=True)

    def __str__(self):
        return self.Customer_Name
```

Figure 138: Model for Feedback.

```

class ReviewsRatings(models.Model):
    Business_ID = models.ForeignKey(VendorBusinessDetails, on_delete=models.CASCADE)
    Business_Name = models.CharField(max_length=70, blank=True)
    Customer_Name = models.CharField(max_length=70, blank=True)
    Rating = models.FloatField(blank=True, default=0)
    Review = models.CharField(max_length=500, blank=True)

    def __str__(self):
        return self.Business_Name

```

*Figure 139: Model for Reviews and Ratings.*

```

from django.contrib import admin
from . import models

# username = admin, pw = admin
#registering models
admin.site.register(models.User)
admin.site.register(models.VendorRequest)
admin.site.register(models.Cart)
admin.site.register(models.Charges)
admin.site.register(models.VendorBusinessDetails)
admin.site.register(models.Feedback)
admin.site.register(models.ReviewsRatings)
@admin.register(models.Deals)
class DealsAdmin(admin.ModelAdmin):
    list_display=['deal_title', 'deal_category']

admin.site.register(models.CouponActive)
admin.site.register([models.EmailNotificationModel])

```

*Figure 140: Registering the database.*

### **3.7.3.2 Application Endpoints**

There are lots of endpoints used with the help of Django rest framework. Below are some images of endpoints that are used for this KarKhana application.

### i. Overall Endpoints

In the below image overall endpoints used for the application are shown.

```
serializers.py M views.py M models.py M admin.py M jazzmin.py U settings.py M urls.py

brisk_app1 > urls.py > ...
You, 1 minute ago | 1 author (You)
1 from dj_rest_auth.registration.views import VerifyEmailView
2 from django.contrib import admin
3 from django.urls import path,include
4 from brisk_app1.views import CartDelete, CartGetCreate, CartListByUser, ChargesView, ChargesViewAll, CouponListByUser, CouponListByVendor, CouponUpdate, CouponItemsViews, UserCoupon, VendorCoupon
5
6
7
8 urlpatterns = [
9     path('auth/',include('dj_rest_auth.urls')),
10    path('auth/registration/',include('dj_rest_auth.registration.urls')),
11    path('auth/confirm-email/', VerifyEmailView.as_view(), name='account_email_verification_sent'),
12
13
14    # for registration of vendor and customer
15    path('registration/customer/', CustomerRegistrationView.as_view(), name = 'Customer Register' ),
16    path('registration/vendor/', VendorRegistrationView.as_view(), name = 'Vendor Register' ),
17
18
19
20    # to post vendor details for registration process
21    path('unverifiedVendor/',vendorForm.as_view(), name='Unverified Vendors'),
22
23
24    # to send notification to specific user
25    path('send-email-notification/',SendEmailNotification.as_view()),
26
27
28    # for vendor details
29    path("VendorBusinessDetails/", TodoGetCreate.as_view()),
30    path("VendorBusinessDetails/<int:pk>", TodoUpdateDelete.as_view()),
31
32
33    # for deals details
34    path("deals/", DealsView.as_view()),
35    path("deals/<str:vendor>/", DealsGCUView.as_view()),
36
37
38    # for add to cart
39    path("Cart/", CartGetCreate.as_view()),
40    path("CartDelete/<int:pk>/", CartDelete.as_view()),
41    path("Cart/<str:cart>/", CartListByUser.as_view(), name='deals-list-by-vendor'),
42
43
44    # for coupon charge calculation
45    path('Charges/', ChargesViewAll.as_view()),
46    path('Charges/<int:pk>/', ChargesView.as_view()),
47    # path('auth/google/', GoogleLogin.as_view(), name='google_login')
48
49
50    # coupon
51    path('Coupon/', CouponItemsViews.as_view()),
52    path('Coupon/<int:pk>', CouponUpdate.as_view()),
53    path('UserCoupon/<str:user>', CouponListByUser.as_view()),
54    path('VendorCoupon/<str:vendor>', CouponListByVendor.as_view()),
55
56
57 ]
```

*Figure 141: Overall Endpoints of the System.*

### 3.7.3.3 Mobile Application

#### Iteration 1

##### i. Dart and Django installation

###### Dart

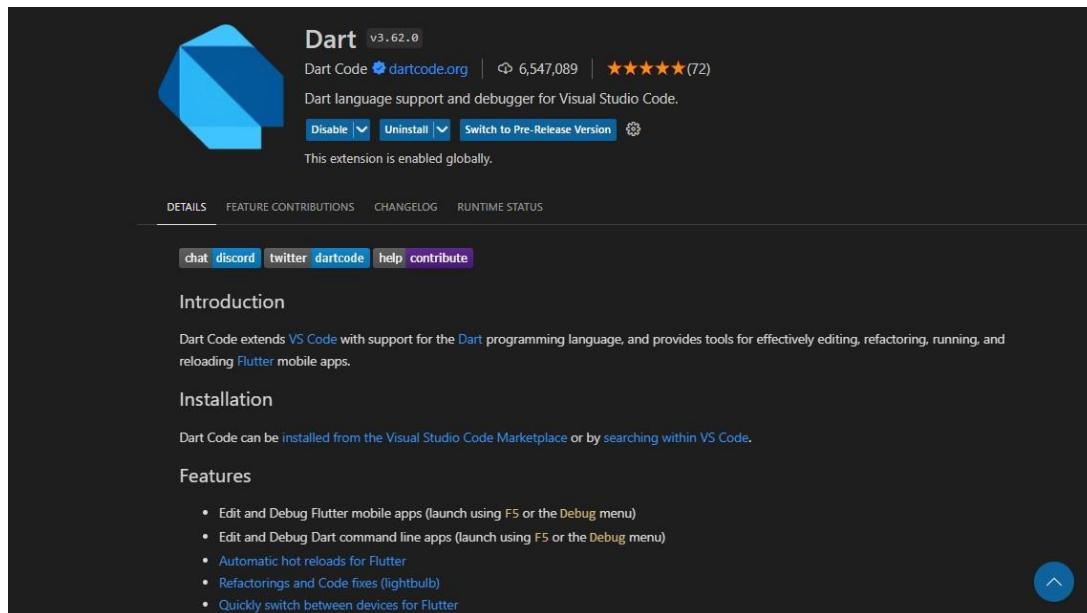


Figure 142: Dart installation in Vs code.

```
PS G:\FYP\KarKhana\KarKhana> flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.19.3, on Microsoft Windows [Version
    10.0.22631.3447], locale en-US)
[✓] Windows Version (Installed version of Windows is version 10 or higher)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Chrome - develop for the web
[!] Visual Studio - develop Windows apps (Visual Studio Community 2022 17.8.2)
    X Visual Studio is missing necessary components. Please re-run the Visual
        Studio installer for the "Desktop development with C++" workload, and
        include these components:
        MSVC v142 - VS 2019 C++ x64/x86 build tools
            - If there are multiple build tool versions available, install the
                latest
        C++ CMake tools for Windows
        Windows 10 SDK
[✓] Android Studio (version 2023.2)
[✓] VS Code (version 1.87.2)
[✓] Connected device (4 available)
[✓] Network resources

! Doctor found issues in 1 category.
PS G:\FYP\KarKhana\KarKhana>
```

Figure 143: Running Flutter Doctor in Terminal.

# Django



*Figure 144: Python installation in Vs code.*

The screenshot shows a Windows desktop environment with the Visual Studio Code (VS Code) application open. The title bar reads "KarKhana\_dj". The left sidebar displays the "EXPLORER" view, showing the project structure:

- OPEN EDITORS
- KARKHANA\_DJ
  - karkhana\_app
  - karkhana\_config
    - \_\_pycache\_\_
    - \_\_init\_\_.py
    - asgi.py
    - settings.py
    - urls.py
    - wsgi.py
  - karkhana\_venv
  - media
  - templates\account\email
    - email\_confirmation\_message.txt
    - email\_confirmation\_subject.txt
  - venv
  - db.sqlite3
  - manage.py

The "settings.py" file is currently selected and open in the main editor area. The code content is as follows:

```
1  import os
2  from pathlib import Path
3
4  # Build paths inside the project like this: BASE_DIR / 'subdir'.
5  BASE_DIR = Path(__file__).resolve().parent.parent
6
7
8  # Quick-start development settings - unsuitable for production
9  # See https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/
10
11 # SECURITY WARNING: keep the secret key used in production secret!
12 SECRET_KEY = 'django-insecure-#*1j@34984fb14aa-jt*at82bn*2t$zxbwz)30^6z=es'
13
14 # SECURITY WARNING: don't run with debug turned on in production!
15 DEBUG = True
16
17 ALLOWED_HOSTS = ["*"]
18
19
20 # Application definition
21
22
23 INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'karkhana_app',
]
```

The bottom status bar indicates the current working directory is "PS G:\FYP\KarKhana\KarKhana\_dj>".

Ln 59, Col 53 Sp

## ii. Welcome Page UI

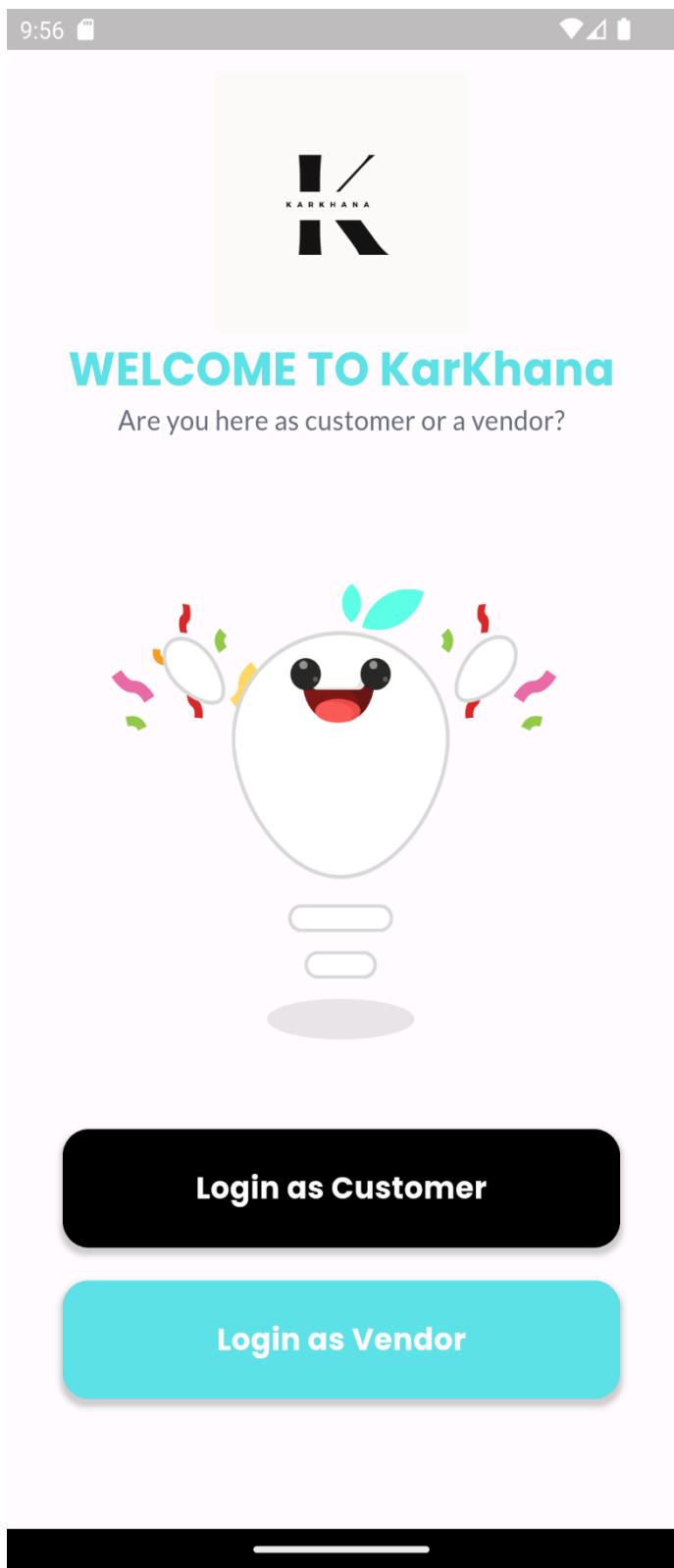


Figure 146: Welcome Page UI.

## iii. Customer Login Page

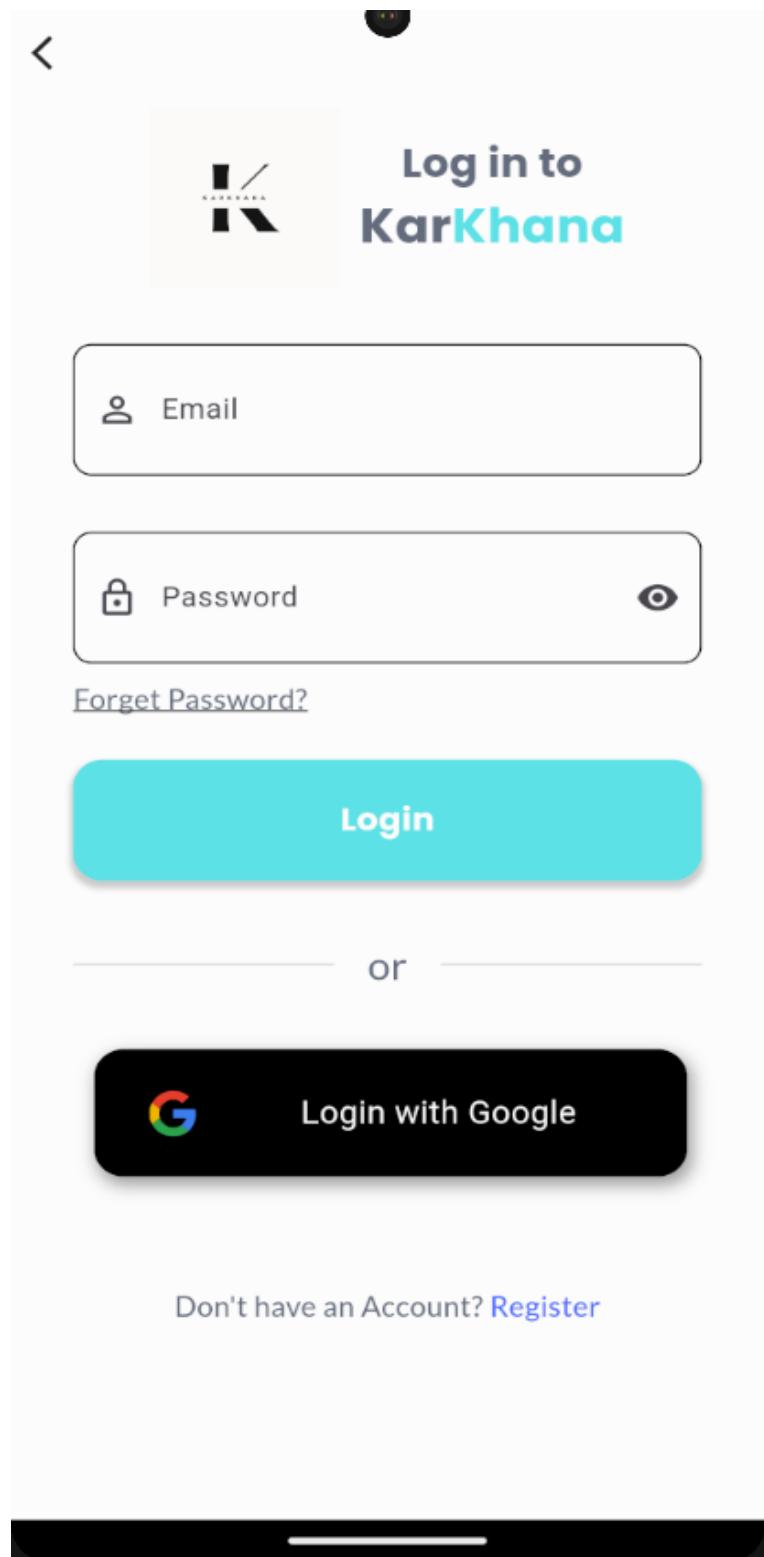
UI

Figure 147: Customer Login Page UI.

## Code

```
    ButtonContainer(
      text: 'Login',
      onClick: () async {
        final isValidForm =
            _formKey.currentState!.validate();
        if (isValidForm) {
          var authResponse =
              await userAuth(_UserEmail.text, _UserPw.text);
          if (authResponse.runtimeType == String) {
            // ignore: use_build_context_synchronously
            showDialog(
              context: context,
              builder: (BuildContext context) =>
                  AlertDialog(
                    title: const Text('Invalid Input'),
                    content: Text(
                      authResponse,
                    ), // Text
                    actions: [
                      TextButton(
                        onPressed: () {
                          Navigator.pop(context);
                        },
                        child: const Text('Close')) // TextButton
                    ],
                  )); // AlertDialog
          } else if (authResponse.runtimeType == User) {
            User user = authResponse;
            // ignore: use_build_context_synchronously
            context.read<UserCubit>().emit(user);
            if (user.is_user == 1) {
              ScaffoldMessenger.of(context)
                  .showSnackBar(const SnackBar(
                    backgroundColor: Colors.green,
                    content: Text(
                      "Logged in",
                      textAlign: TextAlign.center,
                    )));
            } // Text // SnackBar
            Navigator.of(context).pushReplacement(
                MaterialPageRoute(
                  builder: (context) => askLocation())); // MaterialPageRoute
          } else {
            await logOutUser(user.token!);
            showDialog(
              context: context,
              builder: (BuildContext context) =>
                  AlertDialog(
                    title: const Text('Invalid Input'),
                    content: const Text(
                      'Unable to log in with provided credentials.'), // Text
                    actions: [
                      TextButton(
                        onPressed: () async {
                          Navigator.pop(context);
                        },
                        child: const Text('Close')), // TextButton
                    ],
                  )); // AlertDialog
          }
        }
      },
      buttonColor: ColorsOn.secondaryColor,
      buttonBorderColor: ColorsOn.secondaryColor,
    ); // ButtonContainer
```

Figure 148: Login Code.

User API

```
Future<dynamic> userAuth(String email, String password) async {
  Map body = {"email": email, "password": password};
  var url = Uri.parse("$baseUrl/accounts/auth/login/");
  var res = await http.post(
    url,
    body: body,
  );

  try {
    if (res.statusCode == 200) {
      Map json = jsonDecode(res.body);
      String token = json['key'];
      var box = await Hive.openBox(tokenBox);
      box.put('token', token);
      User? user = await getUser(token);
      return user;
    } else {
      Map json = jsonDecode(res.body);
      if (json.containsKey("email")) {
        return json["email"][0];
      }
      if (json.containsKey("password")) {
        return json["password"][0];
      }
      if (json.containsKey("non_field_errors")) {
        return json["non_field_errors"][0];
      }
    }
  } catch (error) {
    return error;
  }
}
```

Figure 149: Login API code for both type of users.

## iv. User Registration Page

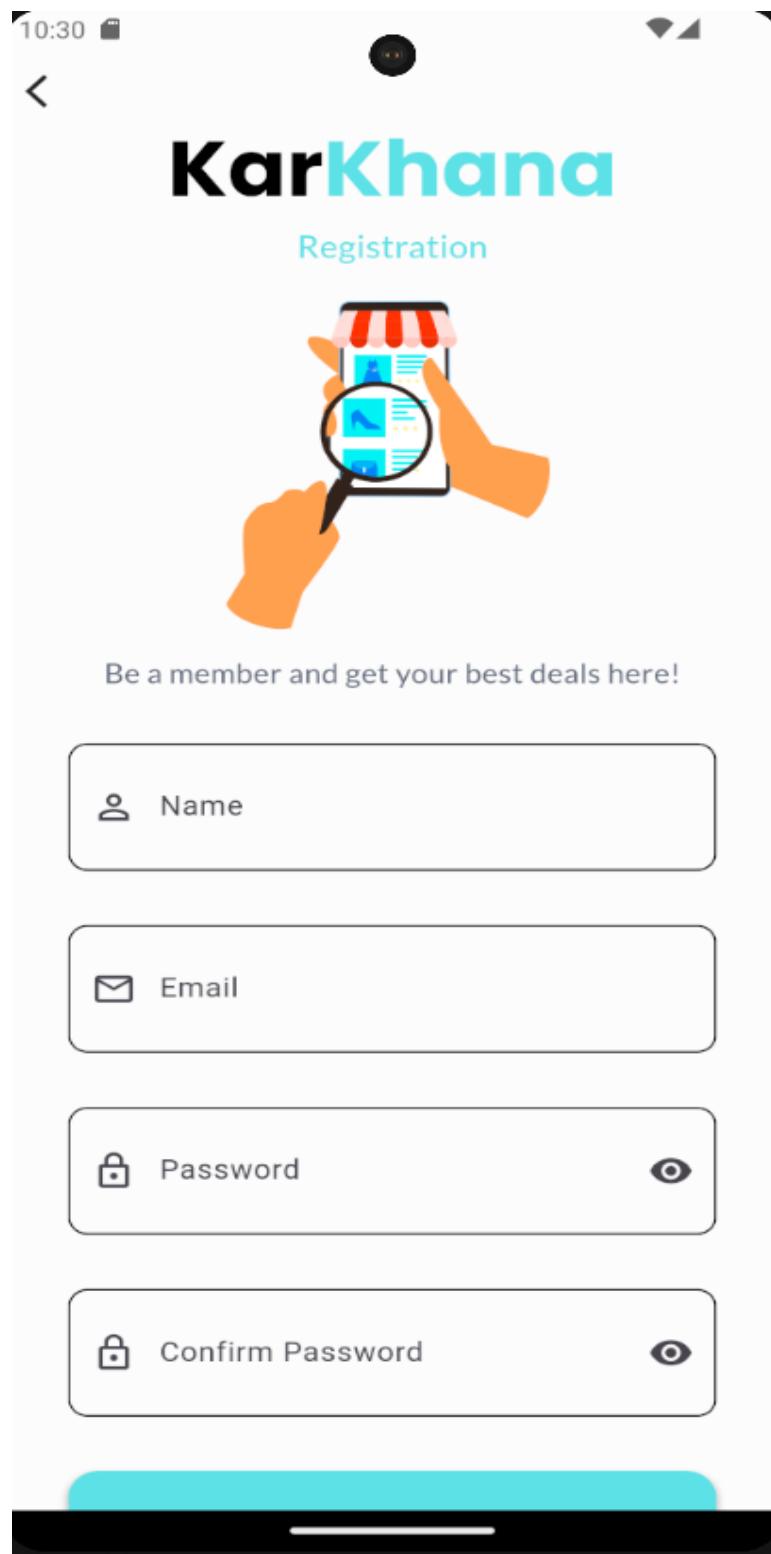
UI

Figure 150: User Registration UI.

Code

```
//> // Signup
  ButtonContainer(
    | butborderColor: ColorsOn.secondaryColor,
    | text: 'Signup',
    | butColor: ColorsOn.secondaryColor,
    | onClick: () async {
    |   final isValidForm =
    |     _formKey.currentState!.validate();
    |   if (isValidForm) {
    |     var authResponse = await registerUser(
    |       _UserName.text,
    |       _UserEmail.text,
    |       _UserPw.text,
    |       _UserPwConfirm.text,
    |     );
    |
    |     if (authResponse.runtimeType == String) {
    |       // ignore: use_build_context_synchronously
    |       showDialog(
    |         context: context,
    |         builder: (BuildContext context) =>
    |           AlertDialog(
    |             title: const Text('Invalid Input'),
    |             content: Text(
    |               authResponse,
    |             ), // Text
    |             actions: [
    |               TextButton(
    |                 onPressed: () {
    |                   Navigator.pop(context);
    |                 },
    |                 child: const Text('Close')) // TextButton
    |               ],
    |             ); // AlertDialog
    |       } else {
    |         Navigator.of(context).push(
    |           MaterialPageRoute(builder: (context) {
    |             return LoadingSplashScreen();
    |           }), // MaterialPageRoute
    |         );
    |       }
    |     }
    |   },
    | ), // ButtonContainer
```

Figure 151: User Registration Code.

API

```
Future<dynamic> registerUser(
    String full_name,
    String email,
    String password,
    String confirm_password,
) async {
    Map<String, dynamic> data = {
        "full_name": full_name,
        "email": email,
        "password1": password,
        "password2": confirm_password,
    };

    var url = Uri.parse("$baseUrl/accounts/registration/customer/");
    var res = await http.post(
        url,
        body: data,
    );

    if (res.statusCode == 200 || res.statusCode == 201) {
        Map json = jsonDecode(res.body);

        if (json.containsKey("key")) {
            String token = json["key"];
            var box = await Hive.openBox(tokenBox);
            box.put("token", token);
            var a = await getUser(token);
            if (a != null) {
                User user = a;
                return user;
            } else {
                return null;
            }
        }
    } else if (res.statusCode == 400) {
        Map json = jsonDecode(res.body);
        if (json.containsKey("email")) {
            return json["email"][0];
        } else if (json.containsKey("password")) {
            return json["password"][0];
        } else if (json.containsKey("non_field_errors")) {
            return json["non_field_errors"][0];
        }
    } else {
        return null;
    }
}
```

Figure 152: User Registration API code.

### End Point

The screenshot shows a browser window titled "Customer Registration - Django" with the URL "localhost:8000/accounts/registration/customer/". The page is titled "Customer Registration" and features a "OPTIONS" button. A "Raw data" tab is selected, showing the following JSON response:

```
HTTP 405 Method Not Allowed
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "detail": "Method \"GET\" not allowed."
}
```

Below the raw data, there is an "HTML form" tab containing five input fields: "Username", "Email", "Password1", "Password2", and "Full name".

Figure 153: User Registration End Point.

## v. Settings Page

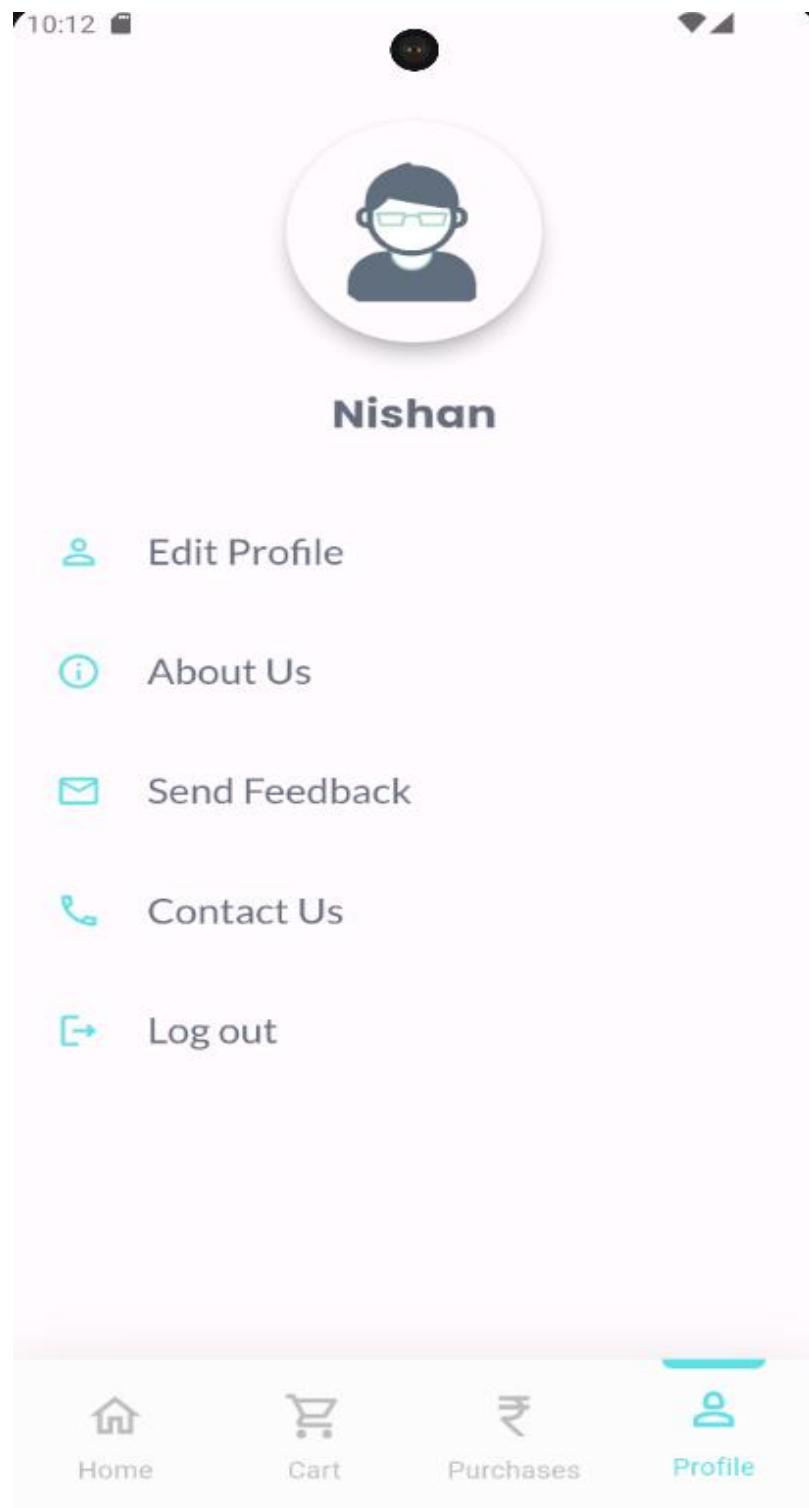
UI

Figure 154: Settings UI.

## vi. Forget Password

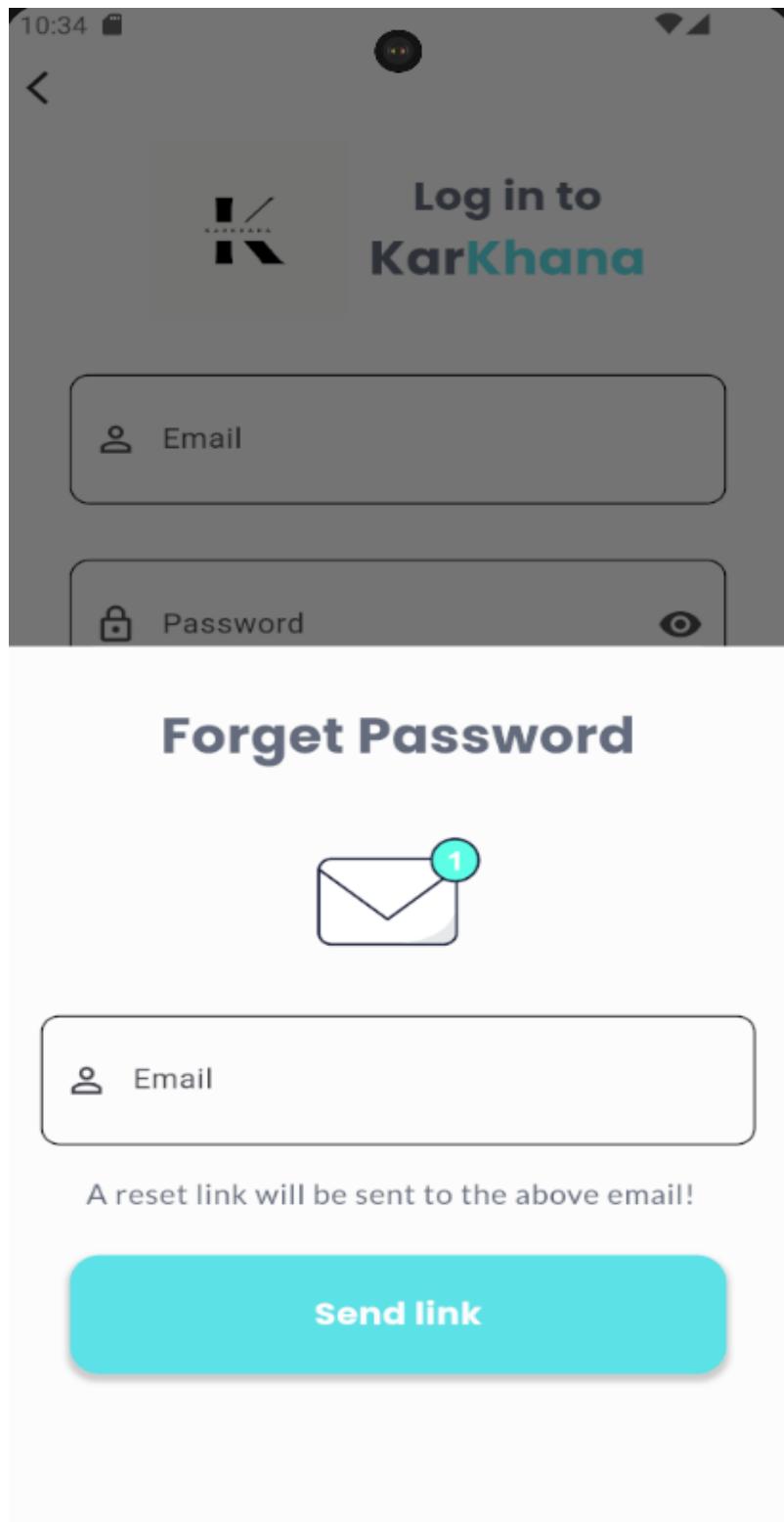
UI

Figure 155: Forget Password UI.

## Code

*Figure 156: Forget Password Code.*

**Iteration 2**

- i. Set Current Location

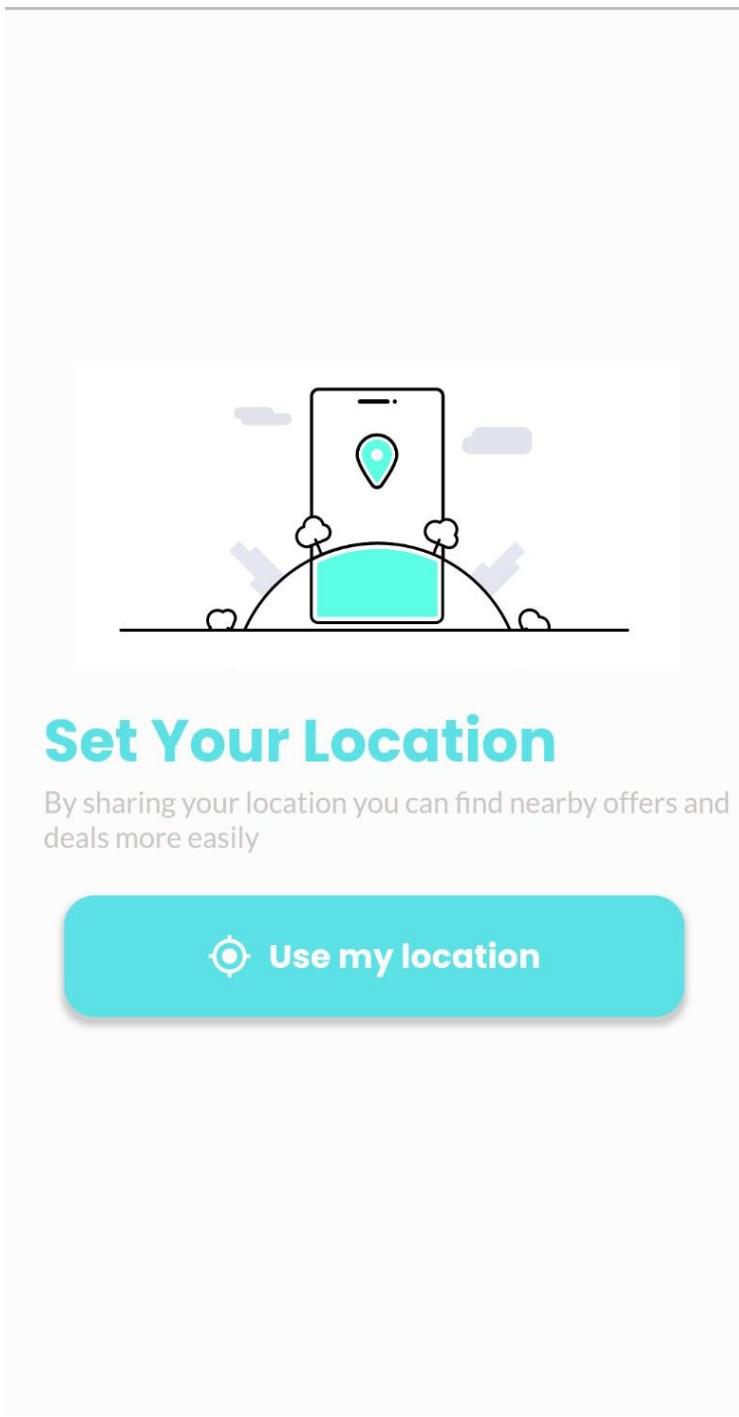
UI

Figure 157: Set Current Location UI.

## Code

```
Future<Position> getCurrentLocationUser() async {
    bool serviceEnabled;
    LocationPermission permission;
    serviceEnabled = await Geolocator.isLocationServiceEnabled();
    if (!serviceEnabled) {
        await Geolocator.openLocationSettings();
        return Future.error('Location services are disabled');
    }

    permission = await Geolocator.checkPermission();
    if (permission == LocationPermission.denied) {
        permission = await Geolocator.requestPermission();
        if (permission == LocationPermission.denied) {
            return Future.error('Location permissions are denied');
        }
    }

    if (permission == LocationPermission.deniedForever) {
        return Future.error(
            'Location permission are permanently denied, we cannot request');
    }

    return await Geolocator.getCurrentPosition();
}

Future<void> getUserAddress(Position position) async {
    List<Placemark> placemarks =
        await placemarkFromCoordinates(position.latitude, position.longitude);
    // ignore: avoid_print
    print(placemarks);

    // ignore: unused_local_variable
    Placemark place = placemarks[0];

    currentLocation = '${place.locality}';
    // "${place.locality}, ${place.subAdministrativeArea},${place.administrativeArea},${place.postalCode}";
    setState(() {});
    var box = await Hive.openBox(tokenBox);
    var keybox = box.get("token");
    await addLocation(keybox, currentLocation);
    Navigator.of(context).pushReplacement(PageTransition(
        type: PageTransitionType.bottomToTop,
        child: btmNavigationBar(), You, 1 second ago * Uncommitted changes
    )); // PageTransition
    // Navigator.of(context).pushReplacement(
    //     MaterialPageRoute(builder: (context) => LoadingSplashScreen2()));
}
```

Figure 158: Set Current Location Code.

## API

```
//adding location to user information
Future<dynamic> addLocation(String key, String address) async {
    Map body = {"location": address};
    var url = Uri.parse("$baseUrl/accounts/auth/user/");
    var res = await http.put(url,
        headers: {'Authorization': 'Token ${key}'}, body: body);

    print(res.body);
    print(res.statusCode);
}

//OSSCOUT API
```

Figure 159: Set Current Location API.

## Endpoint



Figure 160: Set Current Location Endpoint.

## ii. Vendor Login

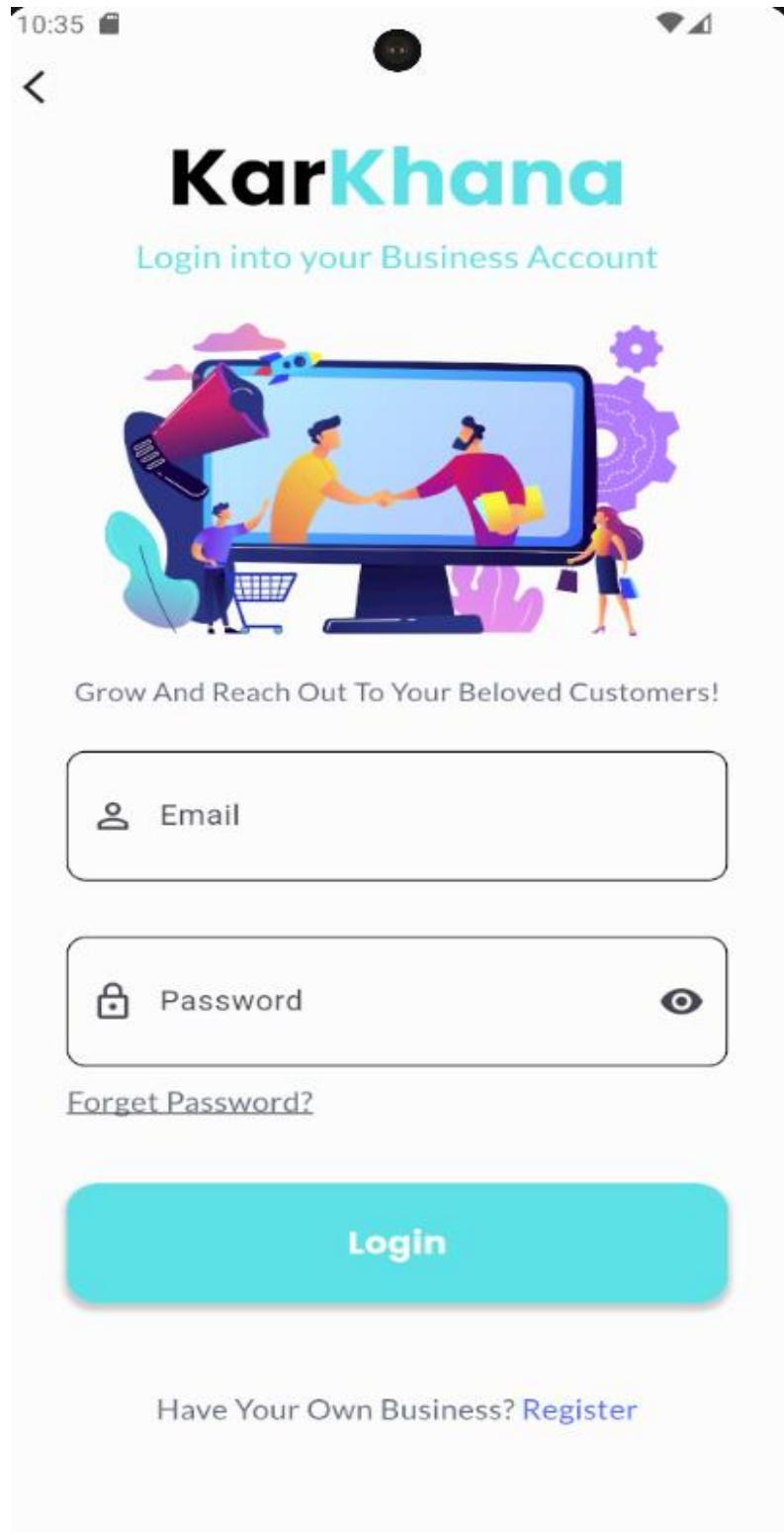
UI

Figure 161: Vendor Login UI.

## Code

```
  ButtonContainer(
    text: 'Login',
    onClick: () async {
      final isValidForm =
        _formKey.currentState!.validate();
      if (isValidForm) {
        var authResponse =
          await userAuth(_UserEmail.text, _UserPw.text);
        if (authResponse.runtimeType == String) {
          // ignore: use_build_context_synchronously
          showDialog(
            context: context,
            builder: (BuildContext context) =>
              AlertDialog(
                title: const Text('Invalid Input'),
                content: Text(
                  authResponse,
                ), // Text
                actions: [
                  TextButton(
                    onPressed: () {
                      Navigator.pop(context);
                    },
                    child: const Text('Close')) // TextButton
                ],
              )); // AlertDialog
        } else if (authResponse.runtimeType == User) {
          User user = authResponse;
          // ignore: use_build_context_synchronously
          context.read
```

Figure 162: Vendor Login Code.

iii. Vendor Registration

UI

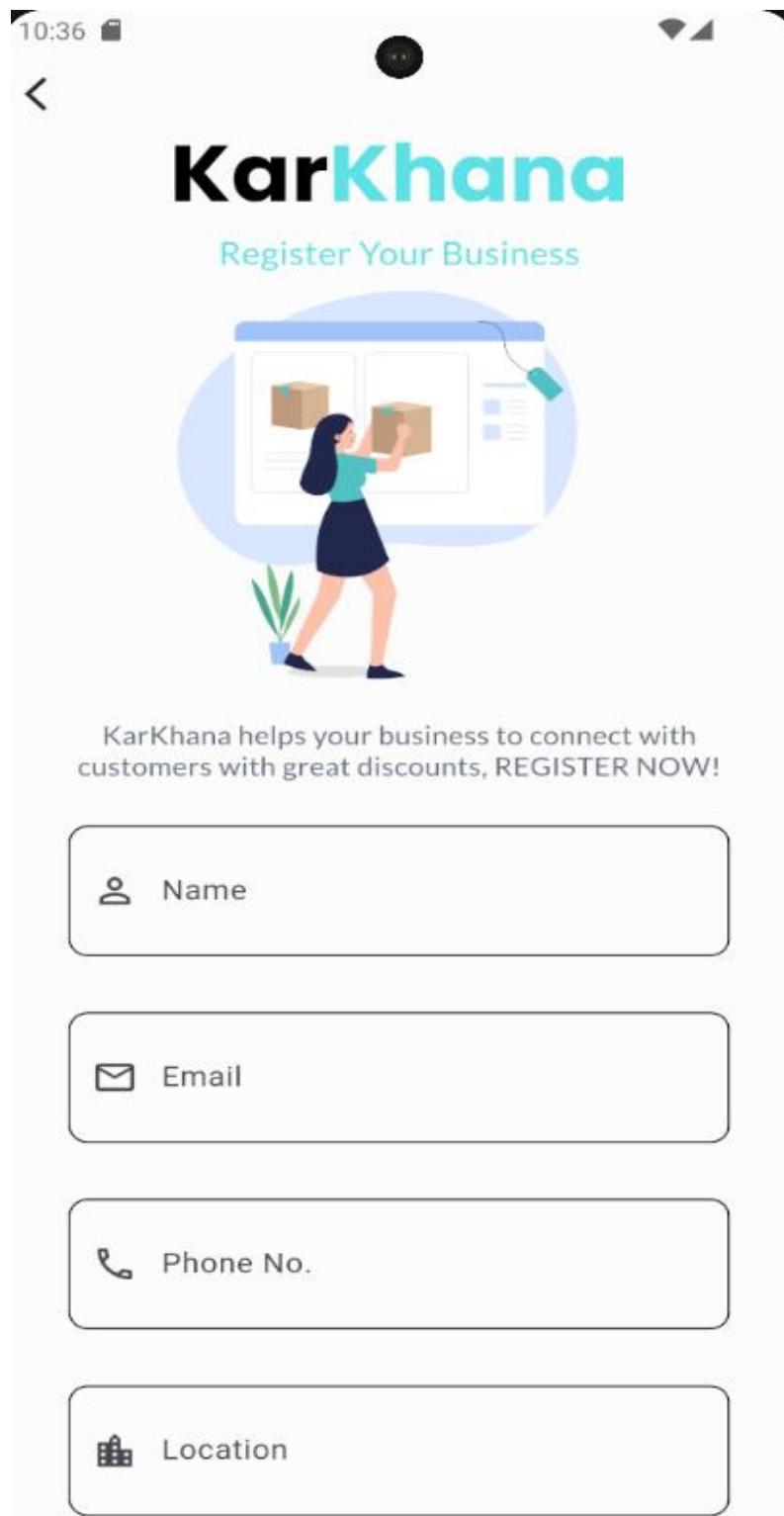


Figure 163: Vendor Registration UI.

## Code

```

ButtonContainer(
  buttonColor: ColorsOn.secondaryColor,
  text: 'Signup',
  buttonColor: ColorsOn.secondaryColor,
  onClick: () => {
    final isValidForm =
      _formKey.currentState.validate();
    if (isValidForm) {
      var authResponse = await VendorRequestRegister(
        _VendorName.text,
        _VendorEmail.text,
        _VendorPhone.text,
        _VendorLocation.text,
        _VendorAbout.text,
        _VendorOpeningDays.text,
        _VendorOpeningTime.text,
        You, 1 second ago * Uncommitted changes
      );
      // ignore: use_build_context_synchronously
      showDialog(
        context: context,
        builder: (BuildContext context) =>
          AlertDialog(
            title: BigText(
              text: 'Registration Submitted'),
            content: const Text(
              "Your Business data has successfully submitted in BriskDeals. We will be in contact with you in few days for further registration process. Please don't lose your contact information that you have just submitted!",
            ),
            actions: [
              TextButton(
                onPressed: () {
                  Navigator.pop(context);
                },
                child: const Text('Close'),
              ),
            ],
          );
    }
  },
), // ButtonContainer

```

Figure 164: Vendor Registration Code.

## Endpoint

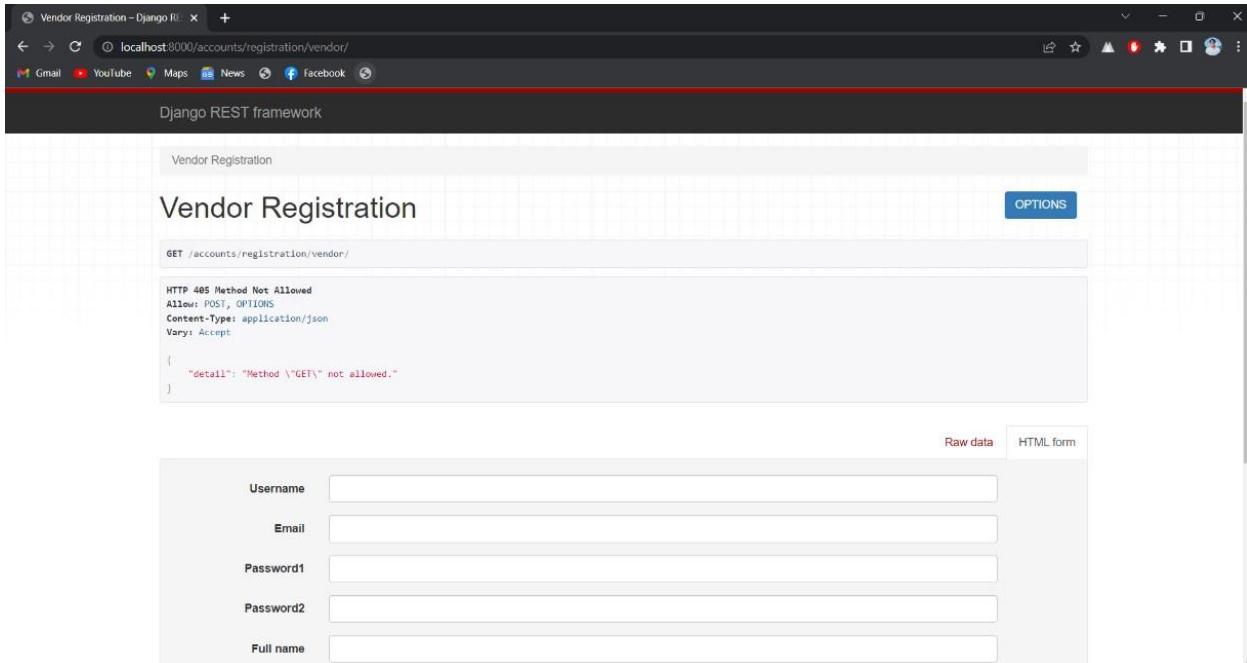


Figure 165: Vendor Registration Endpoint.

API

```
//Vendor details send to admin for registration process
Future<dynamic> VendorRequestRegister(
    String business_name,
    String email,
    String phone,
    String location,
    String about,
    String opening_days,
    String opening_time,
) async {
    Map<String, dynamic> data = {
        "vendor_email": email,
        "vendor_BusinessName": business_name,
        "vendor_phoneNumber": phone,
        "vendor_Location": location,
        "vendor_about": about,
        "vendor_opening_days": opening_days,
        "vendor_opening_time": opening_time,
    };
    var url = Uri.parse("$baseUrl/accounts/unverifiedVendor/");
    var res = await http.post(url, body: data);
    print(res.body);
}
```

Figure 166: Vendor Registration API.

## iv. Vendor Home Page

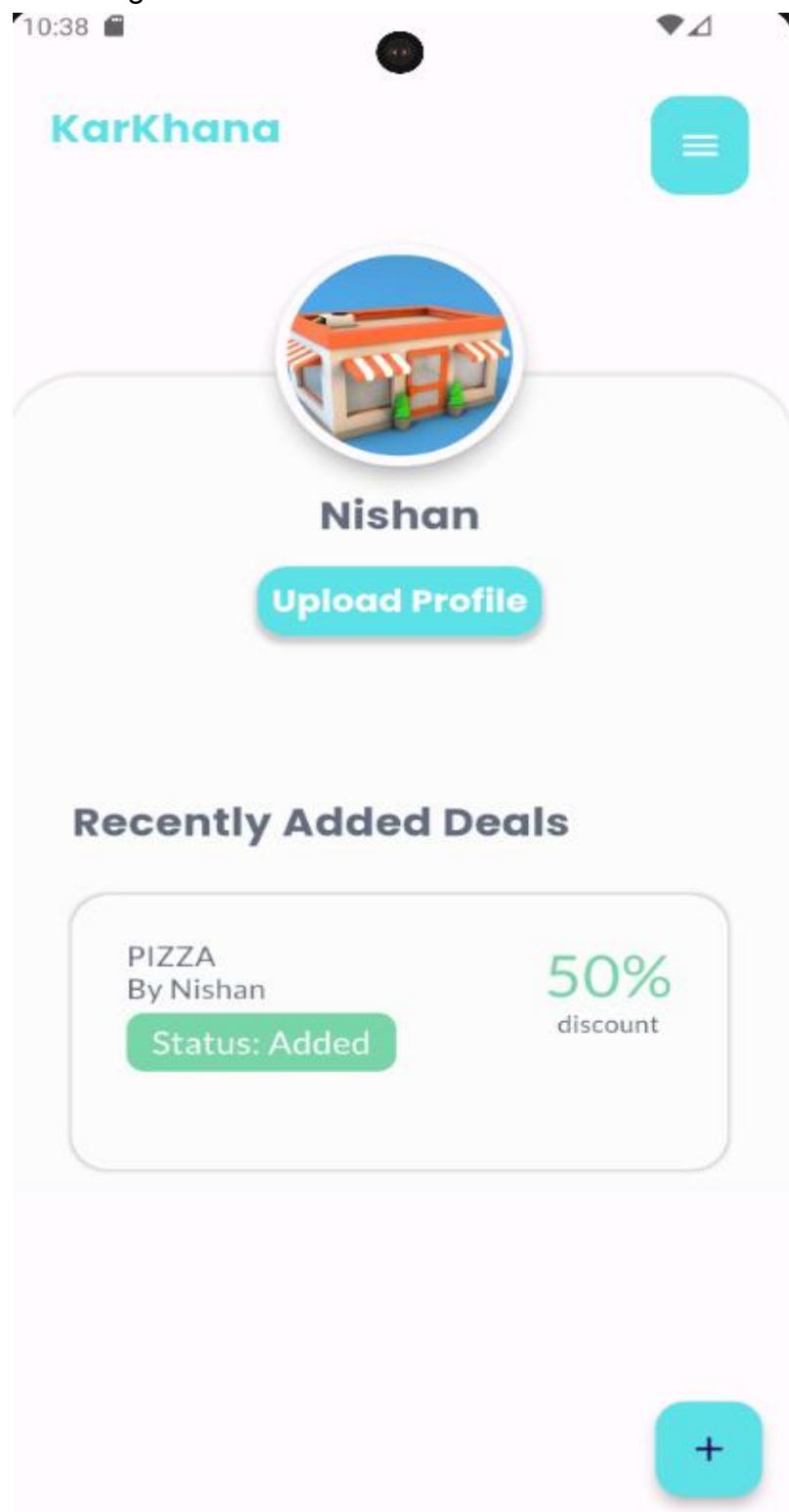


Figure 167: Vendor Homepage UI.

## v. Customer Home Page

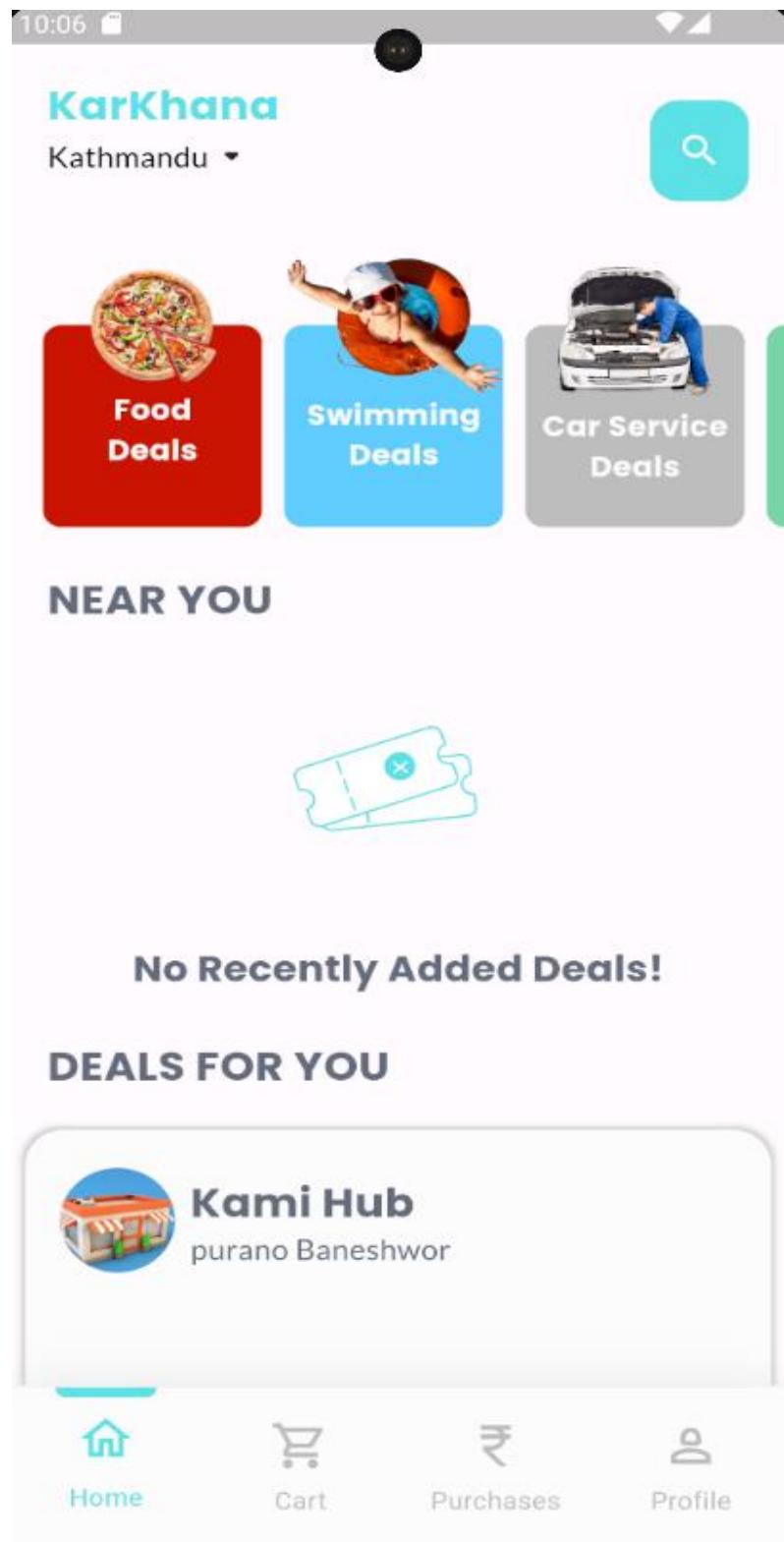


Figure 168: Customer Homepage UI.

## vi. Vendor Add Deals

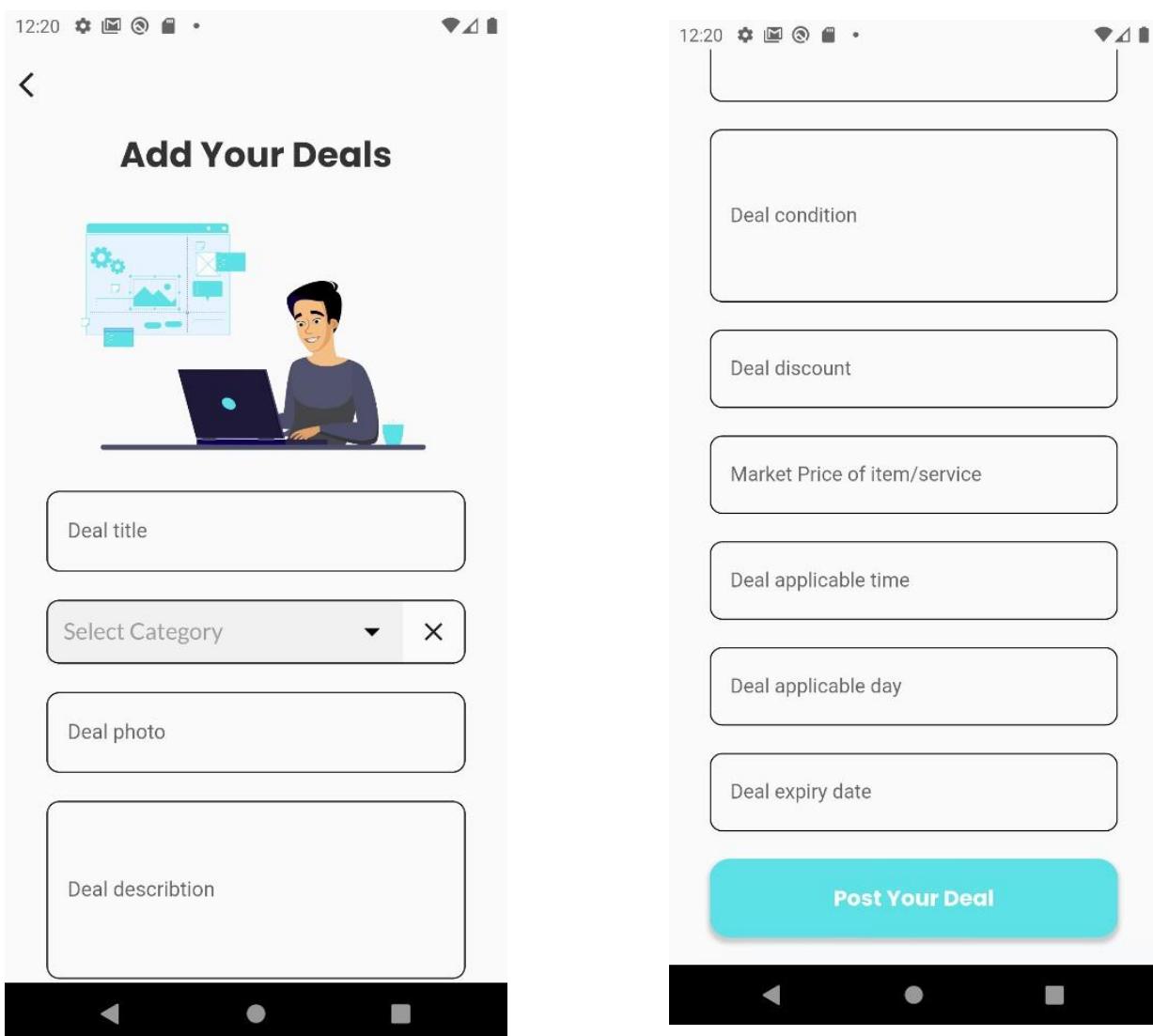


Figure 169: Vendor Add Deals UI.

Code

```

    ButtonContainer(
      butborderColor: ColorsOn.secondaryColor,
      text: "Post Your Deal",
      butColor: ColorsOn.secondaryColor,
      onClick: () {
        int price = int.parse(_dealprice.text);
        int discount = int.parse(_dealdiscount.text);
        postDeals(
          BusinessName,
          BusinessLocation,
          _dealttitle.text,
          price,
          _dealcategory.text,
          _dealdesc.text,
          _dealcondition.text,
          discount,
          _dealapplicabletime.text,
          _dealapplicableday.text,
          _dealexpirydate.text,
          pk,
        );
      } // ButtonContainer
)

```

Figure 170: Vendor Add Deals Code.

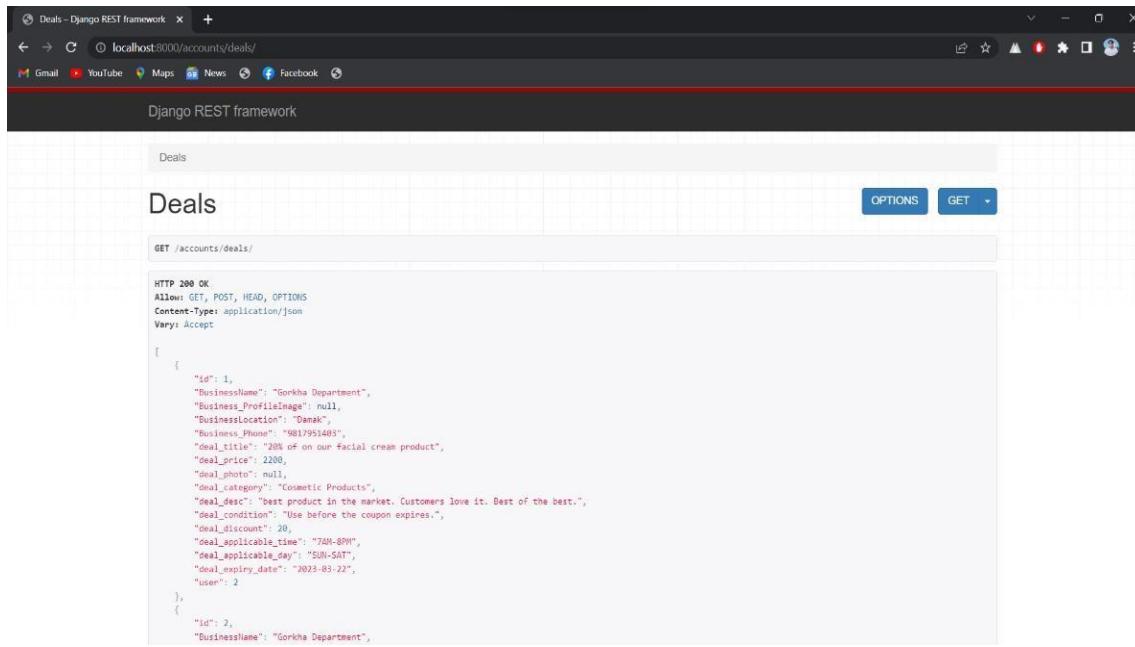
End point

Figure 171: Add Deal Endpoint.

## API

```
Future<dynamic> postDeals(
    String BusinessName,
    String BusinessLocation,
    String deal_title,
    int deal_price,
    String deal_category,
    String deal_desc,
    String deal_condition,
    int deal_discount,
    String deal_applicable_time,
    String deal_applicable_day,
    String deal_expiry_date,
    int user,
) async {
    var url = Uri.parse("$baseUrl/accounts/deals/");
    var res = await http.post(url,
        headers: <String, String>{
            'Content-Type': 'application/json; charset=UTF-8',
        },
        body: jsonEncode(<String, dynamic>{
            "BusinessName": BusinessName,
            "BusinessLocation": BusinessLocation,
            "BusinessPhone": BusinessPhone,
            "deal_title": deal_title,
            // "deal_photo": null,
            "deal_price": deal_price,
            "deal_category": deal_category,
            "deal_desc": deal_desc,
            "deal_condition": deal_condition,
            "deal_discount": deal_discount,
            "deal_applicable_time": deal_applicable_time,
            "deal_applicable_day": deal_applicable_day,
            "deal_expiry_date": deal_expiry_date,
            "user": user,
        }));
    if (res.statusCode == 200 || res.statusCode == 201) {
        setState(() {
            _dealttitle.clear();
            _dealprice.clear();
            _dealcategory.clear();
            _dealdesc.clear();
            _dealcondition.clear();
            _dealdiscount.clear();
            _dealapplicablename.clear();
            _dealapplicableday.clear();
            _dealexpirydate.clear();
        });
        // ignore: use_build_context_synchronously
        return showDialog(
            context: context,
            builder: (BuildContext context) => AlertDialog(
                ...
            )
        );
    }
}
```

Figure 172: Add Deals API - 1.

```
        return showDialog(
            context: context,
            builder: (BuildContext context) => AlertDialog(
                title: LargeText(text: 'Deal Success'),
                content: const Text(
                    "Deal has been successfully added in KarKhana system."),
                ), // Text
                actions: [
                    TextButton(
                        onPressed: () {
                            Navigator.pop(context);
                        },
                        child: const Text('Close')) // TextButton
                    ],
                )), // AlertDialog
        } else {
            // ignore: use_build_context_synchronously
            return showDialog(
                context: context,
                builder: (BuildContext context) => AlertDialog(
                    title: LargeText(text: 'Deal Failed'),
                    content: const Text(
                        "Deal was not added into KarKhana system as you have not uploaded your profile. Upload your business profile first."),
                    ), // Text
                    actions: [
                        TextButton(
                            onPressed: () {
                                Navigator.pop(context);
                            },
                            child: const Text('Close')) // TextButton
                        ],
                    )), // AlertDialog
    }
}
```

Figure 173: Add Deals API - 2.

## vii. Deals Search Bar

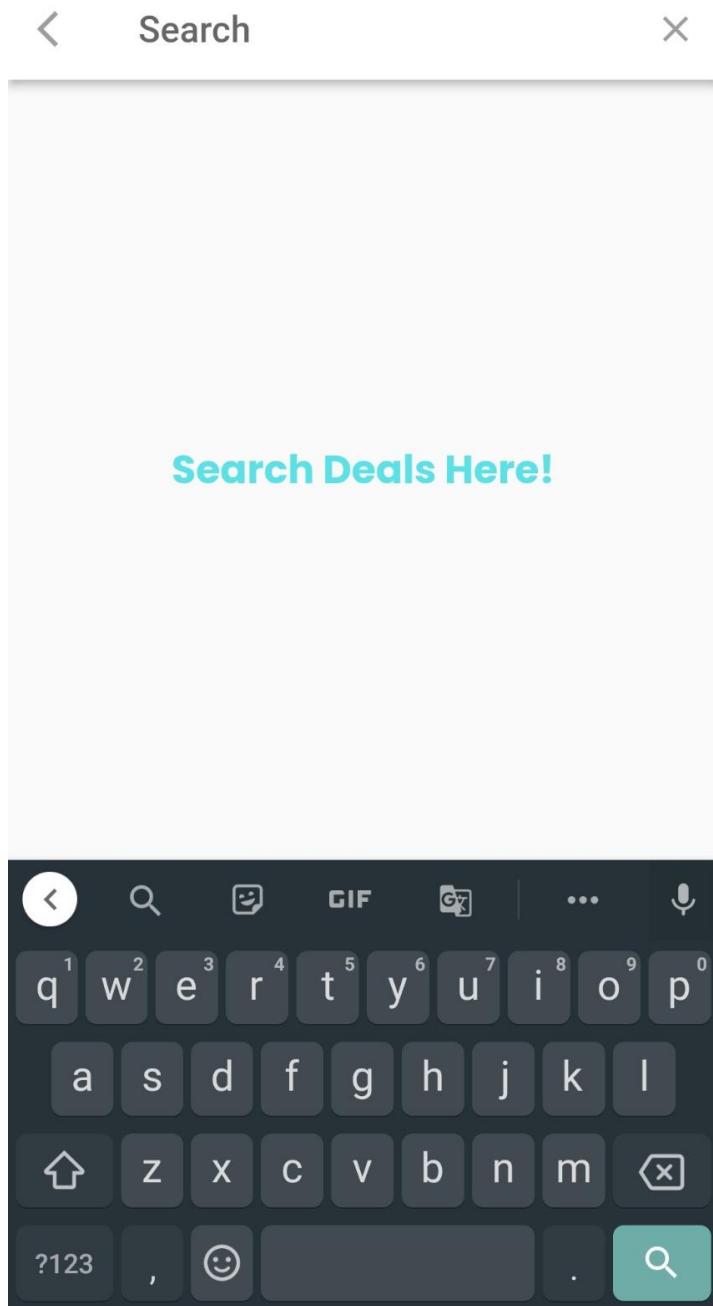


Figure 174: Search UI.

## Code

```

3   class SearchUser extends SearchDelegate {
4     @override
5     List<Widget>? buildActions(BuildContext context) {
6       return [
7         IconButton(
8           onPressed: () {
9             query = "";
10            },
11            icon: Icon(
12              Icons.close,
13            ), // Icon
14          ), // IconButton
15        ],
16      ];
17    }
18
19    @override
20    Widget? buildLeading(BuildContext context) {
21      return IconButton(
22        icon: Icon(Icons.arrow_back_ios),
23        onPressed: () {
24          Navigator.pop(context);
25        },
26      ); // IconButton
27    }
28
29    FetchUser _dealList = FetchUser();
30
31    @override
32    Widget buildResults(BuildContext context) {
33      return SingleChildScrollView(
34        child: Padding(
35          padding: EdgeInsets.only(left: 20.w, right: 20.w),
36          child: Center(
37            child: FutureBuilder(
38              future: _dealList.getUserList(query: query),
39              builder: (BuildContext context, AsyncSnapshot snapshot) {
40                if (snapshot.connectionState == ConnectionState.waiting) {
41                  return const Center(child: CircularProgressIndicator());
42                } else if (snapshot.hasError) {
43                  return Column(
44                    mainAxisAlignment: MainAxisAlignment.center,
45                    children: [
46                      Text('Error: ${snapshot.error}'),
47                      ],
48                    ); // Column
49                } else {
50                  if (snapshot.hasData) {
51                    if (snapshot.hasData && snapshot.data.isEmpty) {
52                      return Column(
53                        children: [
54                          SizedBox(height: 15.h),
55                          Row(
56                            children: [
57                              SizedBox(width: 10.w),
58                              Expanded(
59                                child: BigText(
60                                  size: 16,
61                                  maxLine: 2,
62                                  text:
63                                    "Your Search Results for '${query}'"), // BigText // Expanded
64                                ],
65                              ), // Row
66                            Padding(
67                              padding: EdgeInsets.symmetric(vertical: 100),
68                              child: Column(
69                                children: [
70                                  Lottie.asset("assets/lottie/search_empty.json"),
71                                ],
72                              ),
73                            ),
74                          ],
75                        );
76                      }
77                    }
78                  }
79                }
80              );
81            }
82          );
83        );
84      );
85    }
86
87    @override
88    void onQueryChanged(String query) {
89      _dealList.getUserList(query: query);
90    }
91  }

```

Figure 175: Search Code.

```

69   Lottie.asset("assets/lottie/search_empty.json"),
70   BigText(
71     text: "No Deals Found!",
72   ), // BigText
73   BigText(text: "Search Other Deals."),
74 ],
75 ), // Column
76 ) // Padding
77 ],
78 ); // Column
79 } else {
80   return Column(
81   children: [
82     SizedBox(height: 15.h),
83     Row(
84       children: [
85         SizedBox(width: 10.w),
86         Expanded(
87           child: BigText(
88             size: 16,
89             maxLine: 2,
90             text:
91               "Your Search Results for '${query}'"), // BigText // Expanded
92         ],
93       ), // Row
94     SizedBox(height: 15.h),
95     Center(
96       child: Column(
97         children: snapshot.data.map<Widget>((e) {
98           String VendorBusiness_Name = "${e.Business_name}";
99           String VendorBusiness_Location =
100             "${e.Business_location}";
101           String Deals_Title = "${e.Deals_title}";
102           String Deals_Photo = "${e.Deals_photo}";
103           int Deals_Price = e.Deals_price;
104           String Deals_Desc = "${e.Deals_desc}";
105           String Deal_category = "${e.Deals_category}";
106           String Deals_condition = "${e.Deals_condition}";
107           int Deals_Discount = e.Deals_discount;
108           String Deals_applicable_time =
109             "${e.Deals_applicable_time}";
110           String Deals_applicable_day =
111             "${e.Deals_applicable_day}";
112           String Deals_expiry_date =
113             "${e.Deals_expiry_date}";
114           int Deal_user = e.Deals_user;
115
116           return HomeDealContainer(
117             VendorBusiness_Name: VendorBusiness_Name,
118             VendorBusiness_Location:
119               VendorBusiness_Location,
120             Deals_Title: Deals_Title,
121             // Deals_Photo: Deals_Photo,
122             Deals_Price: Deals_Price,
123             Deals_Desc: Deals_Desc,
124             Deals_category: Deal_category,
125             Deals_condition: Deals_condition,
126             Deals_Discount: Deals_Discount,
127             Deals_applicable_time: Deals_applicable_time,
128             Deals_applicable_day: Deals_applicable_day,
129             Deals_expiry_date: Deals_expiry_date,
130             Deal_user: Deal_user); // HomeDealContainer
131           }).toList(),
132         ),
133       ), // Column
134     SizedBox(height: 20.h),
135   ],
136 );

```

Figure 176: Search Code - 2.

*Figure 177: Search Code - 3.*

API

```
// Search integration by deals title
You, 21 minutes ago | 1 author (You)
class FetchUser {
    var data = [];
    List<Deals> results = [];
    String api = '$baseUrl/accounts/deals/';
    Future<List<Deals>> getUserList({String? query}) async {
        var url = Uri.parse(api);
        var res = await http.get(url);
        try {
            if (res.statusCode == 200) {
                data = json.decode(res.body);
                results = data.map((e) => Deals.fromJson(e)).toList();
                if (query != null) {
                    results = results
                        .where((element) => element.Deals_title!
                            .toLowerCase()
                            .contains(query.toLowerCase()))
                        .toList();
                }
            } else {
                print('api error');
            }
        } on Exception catch (e) {
            print('error: $e');
        }
        return results;
    }
}
```

Figure 178: Search API by Deal Title.

## viii. Admin Send Email Notification

Add email notification model

Notification email:

Notification message:

**SAVE** **Save and add another** **Save and continue editing**

Figure 179: Admin Send Email Verification.

## ix. Vendor Recently Added Deals

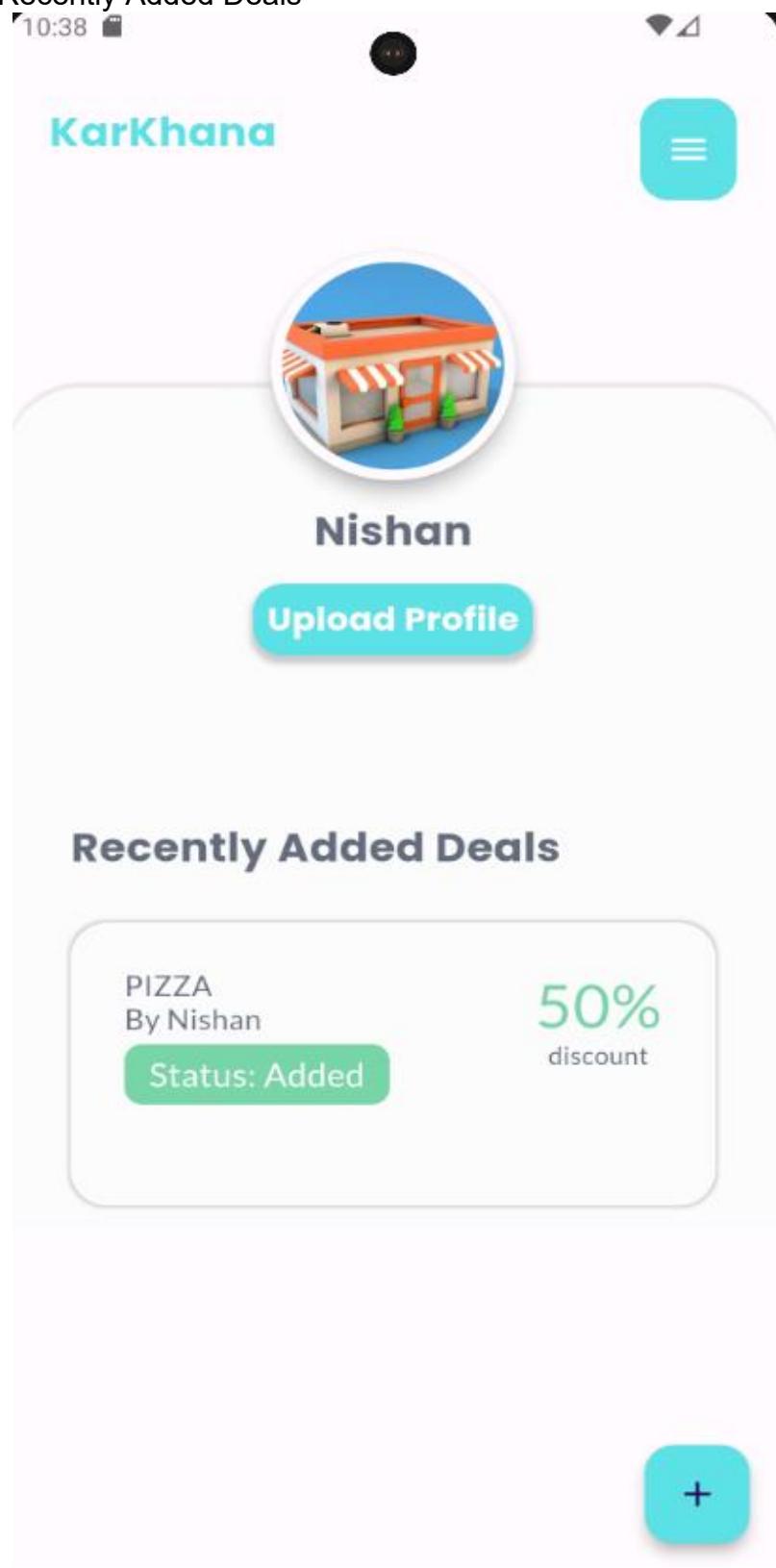


Figure 180: Recently added Deals UI.

**Iteration 3**

- i. Generated Coupon

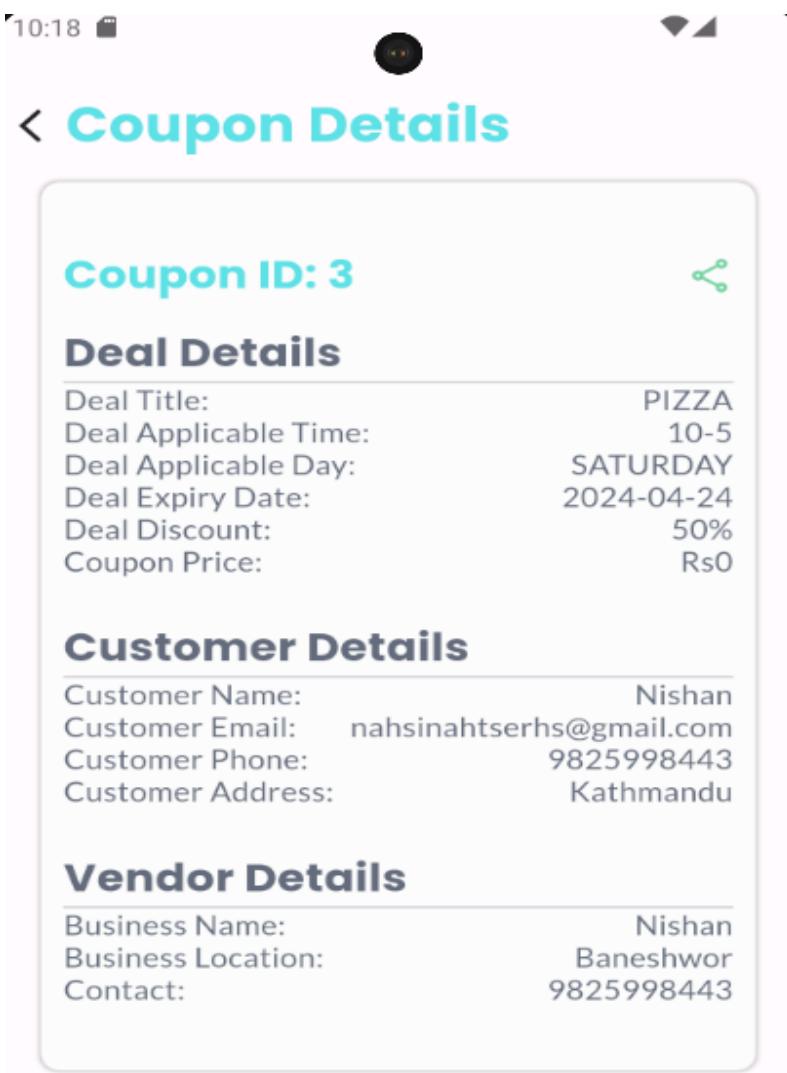
UI

Figure 181: Generated Coupon UI.

Code

```
        context: context,
        builder: (context) {
          return AlertDialog(
            title: BigText(text: "Payment successful"),
            content: NormalText(
              text: "Your payment for " +
                  "${widget.Deals_Title}" +
                  " was successful.",
              maxline: 2,
            ), // NormalText
            actions: [
              SimpleDialogOption(
                child: const Text("Close"),
                onPressed: () {
                  Navigator.pop(context);
                  setState(() {
                    //referenceID = success.idx;
                  });
                },
              ),
            ) // SimpleDialogOption
          ],
        ); // AlertDialog
      },
    );
    await postCouponDetails(
      widget.Deals_Title,
      // widget.Deals_Photo,
      widget.Deals_Discount,
      widget.Deals_applicable_time,
      widget.Deals_applicable_day,
      widget.Deals_expiry_date,
      user_id,
      widget.Vendor_User,
      widget.VendorBusinessName,
      widget.Business_location,
      widget.Business_Phone,
      total_coupon_charge,
      user_name,
      user_location,
      user_phone,
      user_email,
      true,
    );
  );
}
```

Figure 182: Generate Coupon Code.

## API

```
Future<dynamic> postCouponDetails(
    String deal_title,
    // String deal_photo,
    int deal_discount,
    String deal_applicable_time,
    String deal_applicable_day,
    String deal_expiry_date,
    int Customer_user,
    int Vendor_user,
    String VendorBusiness_Name,
    String VendorBusiness_Location,
    String Vendor_Phone,
    int Total_Coupon_Price,
    String Customer_Name,
    String Customer_Address,
    String Customer_Phonenumber,
    String Customer_Email,
    bool Payment_Complete,
) async {
    var url = Uri.parse("$baseUrl/accounts/Coupon/");
    String coupon_Status = "Active";
    var res = await http.post(url,
        headers: <String, String>{
            'Content-Type': 'application/json; charset=UTF-8',
        },
        body: jsonEncode(<String, dynamic>{
            "deal_title": deal_title,
            // "deal_photo": deal_photo,
            "deal_discount": deal_discount,
            "deal_applicable_time": deal_applicable_time,
            "deal_applicable_day": deal_applicable_day,
            "deal_expiry_date": deal_expiry_date,
            "Customer_User": Customer_user,
            "Vendor_User": Vendor_user,
            "VendorBusiness_Name": VendorBusiness_Name,
            "VendorBusiness_Location": VendorBusiness_Location,
            "Vendor_Phone": Vendor_Phone,
            "Total_Coupon_Price": Total_Coupon_Price,
            "Customer_Name": Customer_Name,
            "Customer_Address": Customer_Address,
            "Customer_Phonenumber": Customer_Phonenumber,
            "Customer_Email": Customer_Email,
            "coupon_Status": coupon_Status,
            "Payment_Complete": Payment_Complete,
        }));
    print(res.body);
    print(res.statusCode);

    if (res.statusCode == 200 || res.statusCode == 201) {
        Fluttertoast.showToast(
            msg: "You have purchased this coupon",
            fontSize: 18,
            gravity: ToastGravity.BOTTOM,
            backgroundColor: Colors.green,
            textColor: Colors.white);
    } else {
        Fluttertoast.showToast(
            msg: "Sorry! Something went wrong",
            fontSize: 18,
            gravity: ToastGravity.BOTTOM,
            backgroundColor: Colors.red,
            textColor: Colors.white);
        print("Sorry");
    }
}
```

Figure 183: Generate Coupon API.

## ii. Active Coupon History

UI

Figure 184: Active Coupon UI.

## Code

```

child: FutureBuilder(
  future: GetCouponDetails(id),
  builder: (BuildContext context, AsyncSnapshot snapshot) {
    if (snapshot.connectionState == ConnectionState.waiting) {
      return const Center(child: CircularProgressIndicator());
    } else if (snapshot.hasError) {
      return const Center(
        child: Text("Something went wrong"),
      ); // Center
    } else {
      if (snapshot.hasData) {
        if (snapshot.data && snapshot.data.isEmpty) {
          return Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              SizedBox(
                height: 50.h,
              ), // SizedBox
              Center(
                child: Lottie.asset(
                  'assets/lottie/noDealsFound.json',
                  height: 200.h,
                  width: 200.w,
                ),
              ), // Center
              BigText(
                text: "No Deals In Cart",
                size: 18,
              ), // BigText
            ],
          ); // Column
        } else {
          return Column(
            children: [
              Center(
                child: Column(
                  children: snapshot.data.map<Widget>((e) {
                    int couponId = int.parse("${e.couponID}");
                    String Deals_title = "${e.deal_title}";
                    int Deals_discount = int.parse("${e.deal_discount}");
                    String Deals_applicable_time =
                      "${e.deal_applicable_time}";
                    String Deals_applicable_day =
                      "${e.deal_applicable_day}";
                    String Deals_expiry_date = "${e.deal_expiry_date}";
                    int Customer_User = e.Customer_user;
                    int Deals_user = e.Vendor_user;

                    String Business_name = "${e.VendorBusiness_Name}";
                    String Business_location =
                      "${e.VendorBusiness_Location}";
                    String Business_Phone = "${e.Vendor_Phone}";
                    int Total_coupon_price =
                      int.parse("${e.Total_Coupon_Price}");
                    String CustomerName = "${e.Customer_Name}";
                    String CustomerEmail = "${e.Customer_Email}";
                    String CustomerLocation = "${e.Customer_Address}";
                    String CustomerPhone = "${e.Customer_Phonenumbers}";
                    String coupon_Status = "${e.coupon_Status}";
                    String Payment_Complete = "${e.Payment_Complete}";
                    // String Deals_photo = "${e.Deals_photo}";
                    if (coupon_Status == "Active") {
                      return GestureDetector(
                        onTap: () {
                          ...
                        }
                      );
                    }
                  })
                )
              )
            ],
          );
        }
      }
    }
  }
);

```

Figure 185: Active Coupon Code - 1.

```

    // String Deals_photo = >(e.Deals_photo);
    if (coupon_Status == "Active") {
      return GestureDetector(
        onTap: () {
          Navigator.of(context).push(PageTransition(
            type: PageTransitionType.bottomToTop,
            child: activeCouponDeal(
              couponID: couponId,
              DealTitle: Deals_title,
              DealTime: Deals_applicable_time,
              DealDay: Deals_applicable_day,
              DealExpire: Deals_expiry_date,
              DealDiscount: Deals_discount,
              CouponPrice: Total_coupon_price,
              CustomerName: CustomerName,
              CustomerEmail: CustomerEmail,
              CustomerPhone: CustomerPhone,
              CustomerAddress: CustomerLocation,
              BusinessName: Business_name,
              BusinessLocation: Business_location,
              BusinessContact: Business_Phone,
              isActiveCustomer: true,
            )));
        }, // activeCouponDeal // PageTransition
      ),
      child: smallContainer(
        dealtitle: Deals_title,
        businessName: Business_name,
        discount: Deals_discount,
        frontLabel: "Status",
        newLabel: coupon_Status,
        smallboxColor: ColorsOn.greenColor,
        smallboxWidght: 0.28,
        couponID: couponId,
      )); // smallContainer // GestureDetector
    } else {
      return Container();
    }
  ).toList(), // Column
),
), // Center
],
); // Column
}
else {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      SizedBox(
        height: 50.h,
      ), // SizedBox
      Center(
        child: Lottie.asset(
          'assets/lottie/noDealsFound.json',
          height: 200.h,
          width: 200.w,
        ),
      ), // Center
      BigText(
        text: "No Deals In Cart",
        size: 18,
      ), // BigText
    ],
  );
}
),
); // FutureBuilder

```

Figure 186: Active Coupon History Code - 2.

iii. Previous Coupon History

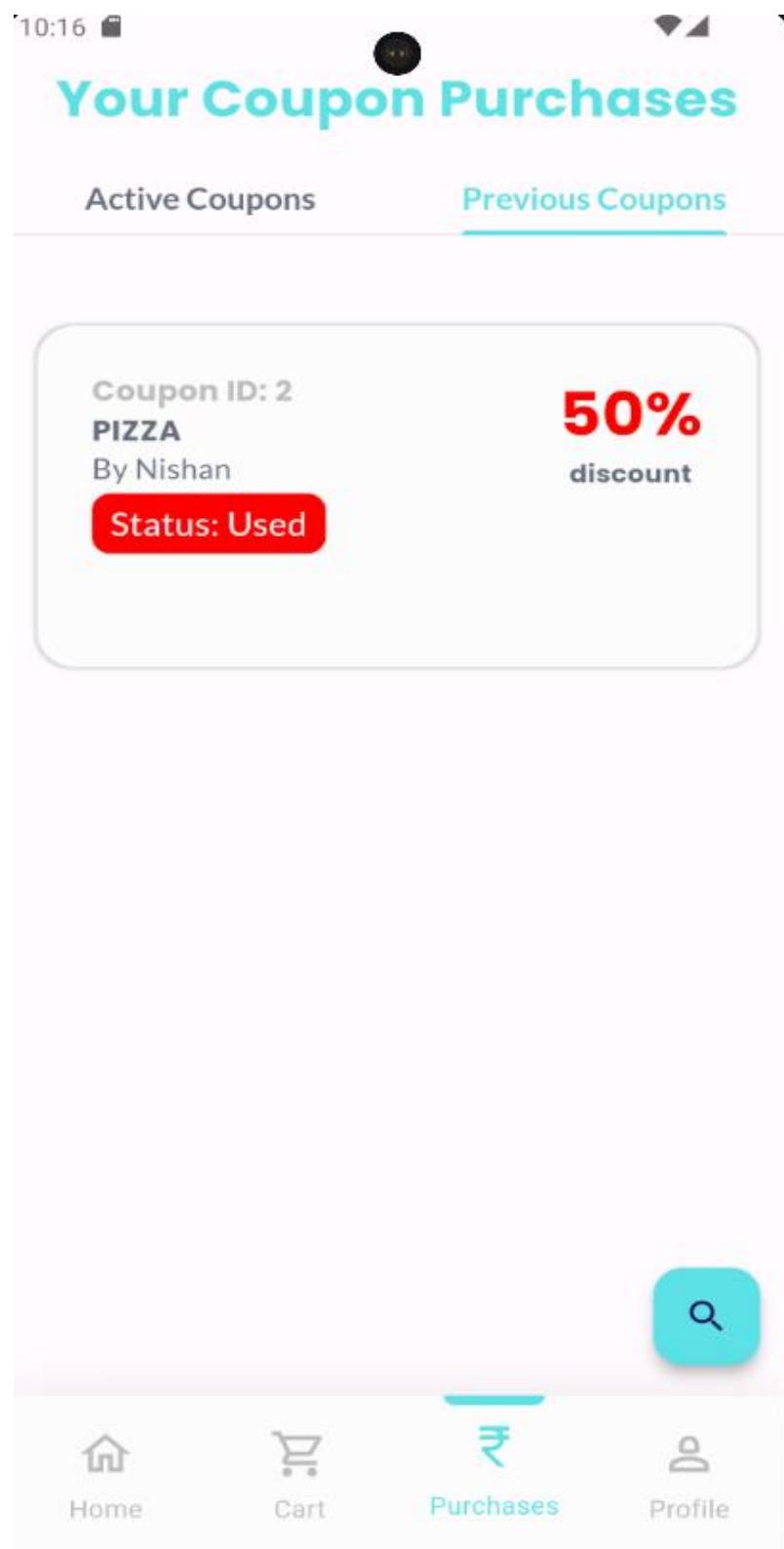


Figure 187: Previous Coupon UI.

## Code

```
child: FutureBuilder<
  future: widget.isVendor
    ? GetVendorCouponDetails(widget.id)
    : GetCouponDetails(widget.id),
builder: (BuildContext context, AsyncSnapshot snapshot) {
  if (snapshot.connectionState == ConnectionState.waiting) {
    return const Center(child: CircularProgressIndicator());
  } else if (snapshot.hasError) {
    return const Center(
      child: Text("Something went wrong"),
    ); // Center
  } else {
    if (snapshot.hasData) {
      if (snapshot.data && snapshot.data.isEmpty) {
        return Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            SizedBox(
              height: 50.h,
            ), // SizedBox
            Center(
              child: Lottie.asset(
                'assets/lottie/noDealsFound.json',
                height: 200.h,
                width: 200.w,
              ),
            ), // Center
            BigText(
              text: "No Deals In Cart",
              size: 18,
            ), // BigText
          ],
        ); // Column
      } else {
        return Column(
          children: [
            Center(
              child: Column(
                children: snapshot.data.map<Widget>((e) {
                  int couponId = int.parse("${e.couponID}");
                  String Deals_title = "${e.deal_title}";
                  int Deals_discount = int.parse("${e.deal_discount}");
                  String Deals_applicable_time =
                    "${e.deal_applicable_time}";
                  String Deals_applicable_day =
                    "${e.deal_applicable_day}";
                  String Deals_expiry_date = "${e.deal_expiry_date}";
                  int Customer_User = e.Customer_user;
                  int Deals_user = e.Vendor_user;

                  String Business_name = "${e.VendorBusiness_Name}";
                  String Business_location =
                    "${e.VendorBusiness_Location}";
                  String Business_Phone = "${e.Vendor_Phone}";
                  int Total_coupon_price =
                    int.parse("${e.Total_Coupon_Price}");
                  String CustomerName = "${e.Customer_Name}";
                  String CustomerEmail = "${e.Customer_Email}";
                  String Customerlocation = "${e.Customer_Address}";
                  String CustomerPhone = "${e.Customer_Phonenumer}";
                  String coupon_Status = "${e.coupon_Status}";
                  String Payment_Complete = "${e.Payment_Complete}";
                  // String Deals_photo = "${e.Deals_photo}";

                  if (coupon_Status == "Used") {
                    return GestureDetector(

```

*Figure 188: Previous Coupon Code - 1.*

```
        if (coupon_Status == "Used") {
            return GestureDetector(
                onTap: () {
                    Navigator.of(context).push(PageTransition(
                        type: PageTransitionType.bottomToTop,
                        child: activeCouponDeal(
                            couponID: couponId,
                            DealTitle: Deals_title,
                            DealTime: Deals_applicable_time,
                            DealDay: Deals_applicable_day,
                            DealExpire: Deals_expiry_date,
                            DealDiscount: Deals_discount,
                            CouponPrice: Total_coupon_price,
                            CustomerName: CustomerName,
                            CustomerEmail: CustomerEmail,
                            CustomerPhone: CustomerPhone,
                            CustomerAddress: CustomerLocation,
                            BusinessName: Business_name,
                            BusinessLocation: Business_location,
                            BusinessContact: Business_Phone))); // activeCouponDeal // PageTransition
                },
                child: smallContainer(
                    dealTitle: Deals_title,
                    businessName: Business_name,
                    discount: Deals_discount,
                    frontLabel: "Status",
                    newLabel: coupon_Status,
                    discountColor: ColorsOn.errorColor,
                    smallboxWidth: 0.26,
                    couponID: couponId,
                )); // smallContainer // GestureDetector
            } else {
                return Container();
            }
        }).toList(), // Column
    ), // Center
),
); // Column
}
} else {
return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
        SizedBox(
            height: 50.h,
        ), // SizedBox
        Center(
            child: Lottie.asset(
                'assets/lottie/noDealsFound.json',
                height: 200.h,
                width: 200.w,
            ),
        ), // Center
        BigText(
            text: "No Deals In Cart",
            size: 18,
        ), // BigText
    ],
); // Column
}
),
), // FutureBuilder
```

Figure 189: Previous Coupon Code - 2.

iv. Purchased and Confirmed Notification

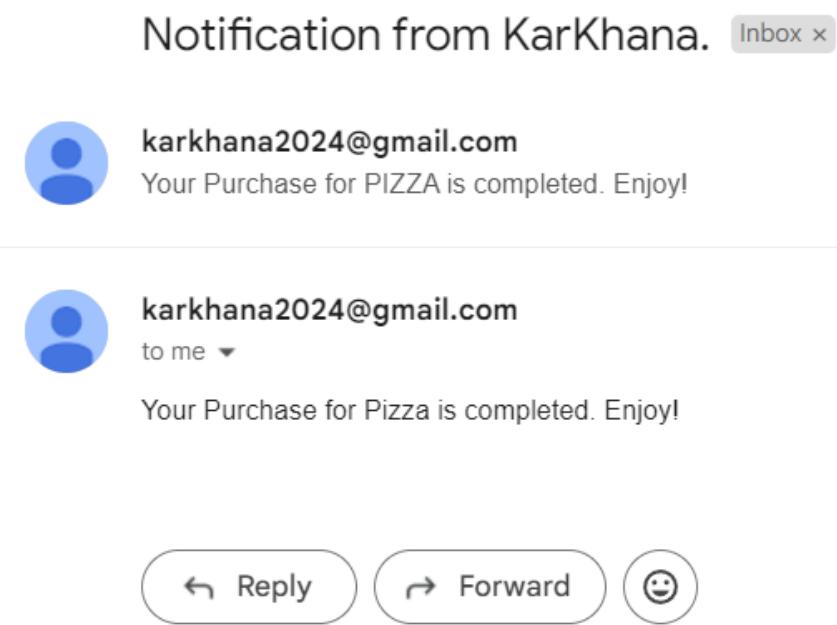


Figure 190: Notification in mail.

Code

```
sendPurchaseEmail(  
    "Your Purchase for " +  
    "${widget.Deals_Title}" +  
    "is completed. Enjoy!",  
    user.email!);  
    setState(() {});
```

Figure 191: Send notification Code.

## API

```
Future<dynamic> sendPurchaseEmail(String message, String email) async {
  Map<String, dynamic> data = {
    "notification_email": email,
    "notification_message": message,
  };
  var api = Uri.parse('$baseUrl/accounts/send-email-notification/');
  var res = await http.post(api, body: data);
  if (res.statusCode == 200 || res.statusCode == 201) {
    return 1;
  } else {
    return Fluttertoast.showToast(
      msg: "Sorry! Something went wrong while sending you email",
      fontSize: 18,
      gravity: ToastGravity.BOTTOM,
      backgroundColor: Colors.red,
      textColor: Colors.white);
  }
}
```

Figure 192: Send Notification API.

## v. Add to Cart

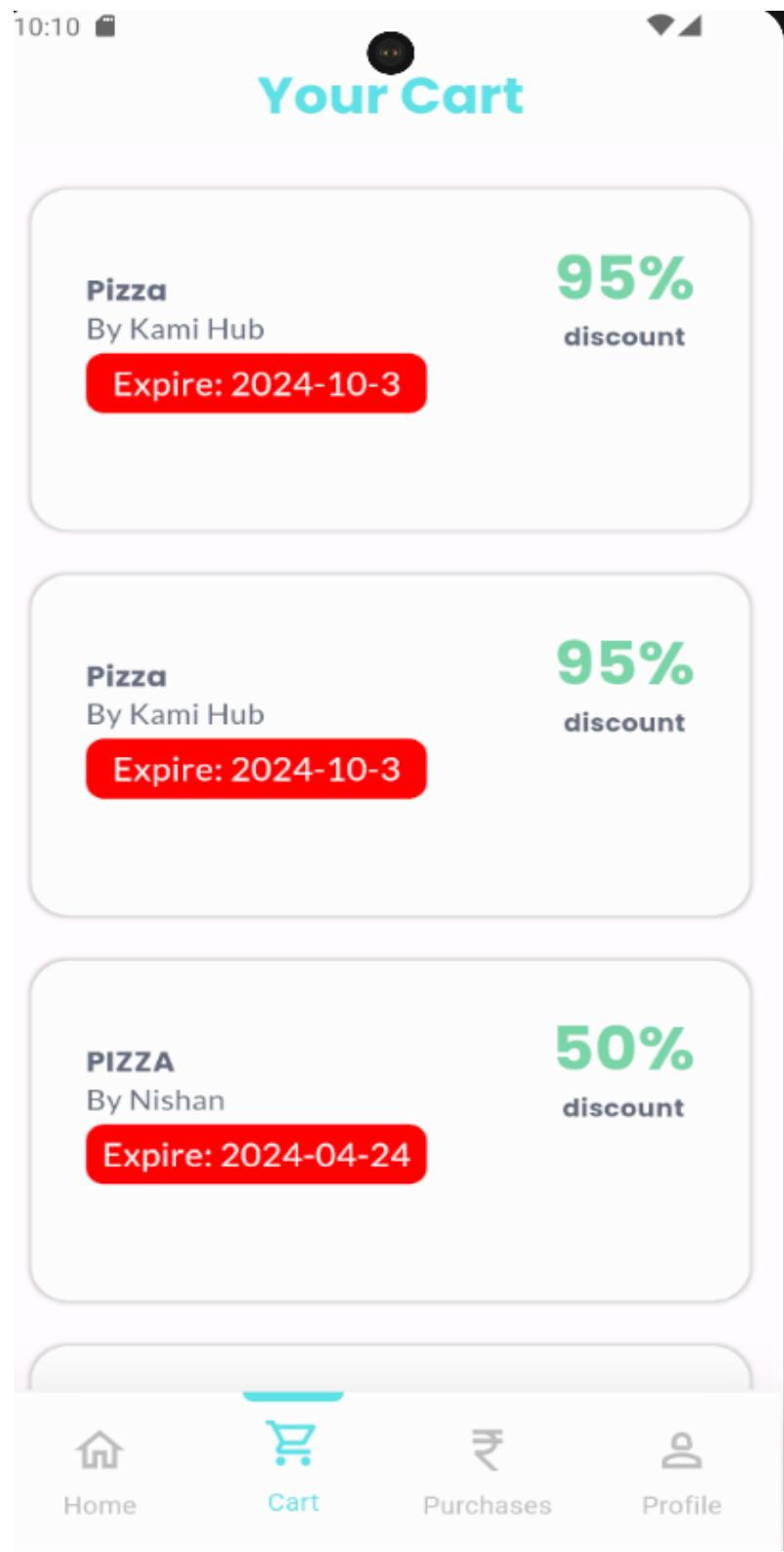
UI

Figure 193: Add to Cart UI.

## Code

```

child: FutureBuilder(
    future: GetCart(id),
    builder: (BuildContext context, AsyncSnapshot snapshot) {
        if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
        } else if (snapshot.hasError) {
            return const Center(
                child: Text("Something went wrong"),
            ); // Center
        } else {
            if (snapshot.hasData) {
                if (snapshot.hasData && snapshot.data.isEmpty) {
                    return Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                            SizedBox(
                                height: 50.h,
                            ), // SizedBox
                            Center(
                                child: Lottie.asset(
                                    'assets/lottie/noDealsFound.json',
                                    height: 200.h,
                                    width: 200.w,
                                ),
                            ), // Center
                            BigText(
                                text: "No Deals In Cart",
                                size: 18,
                            ), // BigText
                            ],
                );
            } else {
                return Column(
                    children: [
                        Center(
                            child: Column(
                                children: snapshot.data.map<Widget>((e) {
                                    int Deals_Id = e.Deals_Id;
                                    String Deals_title = "${e.Deals_title}";
                                    int Deals_price = e.Deals_price;
                                    // String Deals_photo = "${e.Deals_photo}";
                                    String Deals_category = "${e.Deals_category}";
                                    String Deals_desc = "${e.Deals_desc}";
                                    String Deals_condition = "${e.Deals_condition}";
                                    String Business_Phone = "${e.Vendor_Phone}";
                                    int Deals_discount =
                                        int.parse("${e.Deals_discount}");
                                    String Deals_applicable_time =
                                        "${e.Deals_applicable_time}";
                                    String Deals_applicable_day =
                                        "${e.Deals_applicable_day}";
                                    String Deals_expiry_date =
                                        "${e.Deals_expiry_date}";
                                    String Business_name = "${e.Business_name}";
                                    String Business_location =
                                        "${e.Business_location}";
                                    // String Business_photo = "${e.Business_photo}";
                                    int Customer_User = e.Customer_User;
                                    int Deals_user = e.Vendor_User;
                                })
                            )
                        ),
                    ],
                );
            }
        }
    },
);
}

```

Figure 194: Cart code - 1.

```
        return GestureDetector(
            onTap: () {
                Navigator.of(context).push(PageTransition(
                    type: PageTransitionType.bottomToTop,
                    child: CartDealsDetails(
                        VendorBusinessName: Business_name,
                        // Business_photo: Business_photo,
                        Business_location: Business_location,
                        Business_Phone: Business_Phone,
                        Deals_Title: Deals_title,
                        Deals_Price: Deals_price,
                        // Deals_Photo: Deals_photo,
                        Deals_Desc: Deals_desc,
                        Deals_Category: Deals_category,
                        Deals_condition: Deals_condition,
                        Deals_Discount: Deals_discount,
                        Deals_applicable_time:
                            Deals_applicable_time,
                        Deals_applicable_day:
                            Deals_applicable_day,
                        Deals_expiry_date: Deals_expiry_date,
                        Customer_User: Customer_User,
                        Vendor_User: Deals_user,
                    )));
            }
        );
    },
    child: smallContainer(
        dealTitle: Deals_title,
        businessName: Business_name,
        discount: Deals_discount,
        frontLabel: "Expire",
        newLabel: Deals_expiry_date,
        smallboxWidth: 0.38,
        iscart: true,
    ));
); // smallContainer // GestureDetector
).toList(), // Column
), // Center
),
); // Column
)
}
} else {
return Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
SizedBox(
height: 50.h,
), // SizedBox
Center(
child: Lottie.asset(
'assets/lottie/noDealsFound.json',
height: 200.h,
width: 200.w,
),
), // Center
BigText(
text: "No Deals In Cart",
size: 18,
), // BigText
],
); // Column
}
}
```

Figure 195: Cart Code - 2.

## API

```
// sends data of the deal to cart
Future<dynamic> postCart(
    String BusinessName,
    // String BusinessPhoto,
    String BusinessLocation,
    String deal_Title,
    int deal_price,
    String deal_category,
    String deal_desc,
    String deal_condition,
    int deal_discount,
    String deal_applicable_time,
    String deal_applicable_day,
    String deal_expiry_date,
    int Customer_User,
    int Vendor_User,
    // String Deal_photo,
    String Vendor_Phone,
) async {
    var url = Uri.parse("$baseUrl/accounts/Cart/");
    var res = await http.post(url,
        headers: <String, String>{
            'Content-Type': 'application/json; charset=UTF-8',
        },
        body: jsonEncode(<String, dynamic>{
            "deal_title": deal_Title,
            "deal_price": deal_price,
            // "deal_photo": null,
            "deal_category": deal_category,
            "deal_desc": deal_desc,
            "deal_condition": deal_condition,
            "deal_discount": deal_discount,
            "deal_applicable_time": deal_applicable_time,
            "deal_applicable_day": deal_applicable_day,
            "deal_expiry_date": deal_expiry_date,
            "VendorBusiness_Name": BusinessName,
            "VendorBusiness_Location": BusinessLocation,
            // "VendorBusiness_ProfileImage": BusinessPhoto,
            "Customer_User": Customer_User,
            "Vendor_User": Vendor_User,
            "Vendor_Phone": Vendor_Phone,
        }));
    print(res.body);
    print(res.statusCode);

    if (res.statusCode == 200 || res.statusCode == 201) {
        return Fluttertoast.showToast(
            msg: "Item has been added to your Cart",
            fontSize: 18,
            gravity: ToastGravity.BOTTOM,
            backgroundColor: Colors.green,
            textColor: Colors.white);
    } else {
        return Fluttertoast.showToast(
            msg: "Sorry! Something went wrong",
            fontSize: 18,
            gravity: ToastGravity.BOTTOM,
            backgroundColor: Colors.red,
            textColor: Colors.white);
    }
}
```

Figure 196: Cart API.

## vi. Redeem Coupon

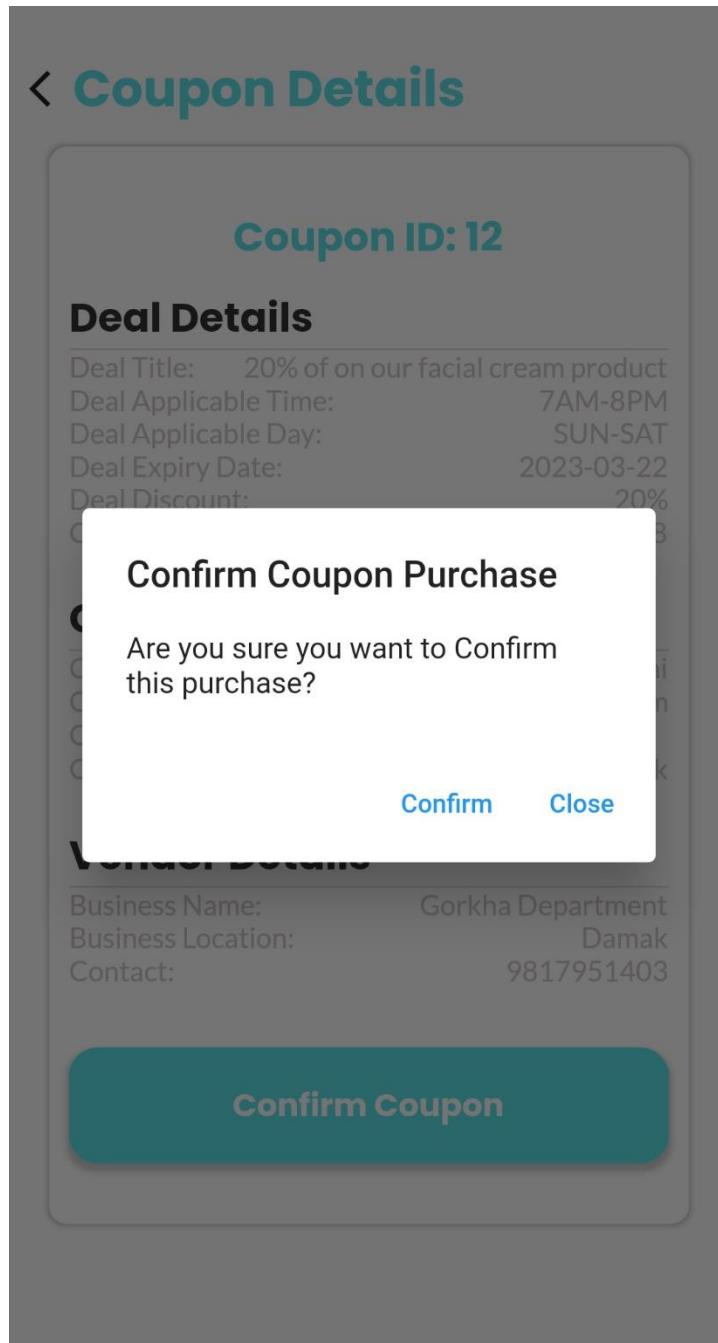
UI

Figure 197: Redeem Coupon UI.

## Code

```
  ButtonContainer(
    butborderColor: ColorsOn.secondaryColor,
    text: "Confirm Coupon",
    butColor: ColorsOn.secondaryColor,
    onClick: () =>
      showDialog(
        context: context,
        builder: (BuildContext context) =>
          AlertDialog(
            title: const Text(
              'Confirm Coupon Purchase'), // Text
            content: const Text(
              'Are you sure you want to Confirm this purchase?'), // Text
            actions: [
              TextButton(
                onPressed: () {
                  confirmPurchase(
                    widget.couponID,
                    widget.customerID,
                    widget.vendorID,
                    "Used",
                    widget
                      .CustomerEmail,
                    "Your Coupon for deal " +
                    "${widget.DealTitle}" +
                    " has been used."),
                },
                Navigator.of(context)
                  .pushReplacement(
                    PageTransition(
                      type: PageTransitionType
                        .bottomToTop,
                      child:
                        VendorDealPurchased(),
                    ), // PageTransition
                  );
                },
                child: const Text(
                  'Confirm')), // Text // TextButton
              TextButton(
                onPressed: () {
                  Navigator.pop(
                    context);
                },
                child:
                  const Text('Close')) // TextButton
            ],
          )); // AlertDialog
    }) // ButtonContainer
```

Figure 198: Redeem Coupon Code.

## API

```
// API
Future<dynamic> confirmPurchase(
    int id,
    int customer,
    int vendor,
    String status,
    String Email,
    String Message,
) async {
    String ID = id.toString();
    String CustomerID = customer.toString();
    String VendorID = vendor.toString();
    Map body = {
        "Customer_User": CustomerID,
        "Vendor_User": VendorID,
        "coupon_Status": status,
    };
    var url = Uri.parse("$baseUrl/accounts/Coupon/$ID");
    var res = await http.put(url, body: body);

    print(res.body);
    print(res.statusCode);
    if (res.statusCode == 200 || res.statusCode == 201) {
        Fluttertoast.showToast(
            msg: "The coupon has been confirmed",
            fontSize: 18,
            gravity: ToastGravity.BOTTOM,
            backgroundColor: Colors.green,
            textColor: Colors.white);
        GetCouponDetails(vendor);
        sendPurchaseEmail(Message, Email);

        setState(() {});
    } else {
        Fluttertoast.showToast(
            msg: "Sorry! Something went wrong",
            fontSize: 18,
            gravity: ToastGravity.BOTTOM,
            backgroundColor: Colors.red,
            textColor: Colors.white);
        print("Sorry");
    }
}
```

Figure 199: Redeem Coupon API.

## vii. Location Based Recommendation System

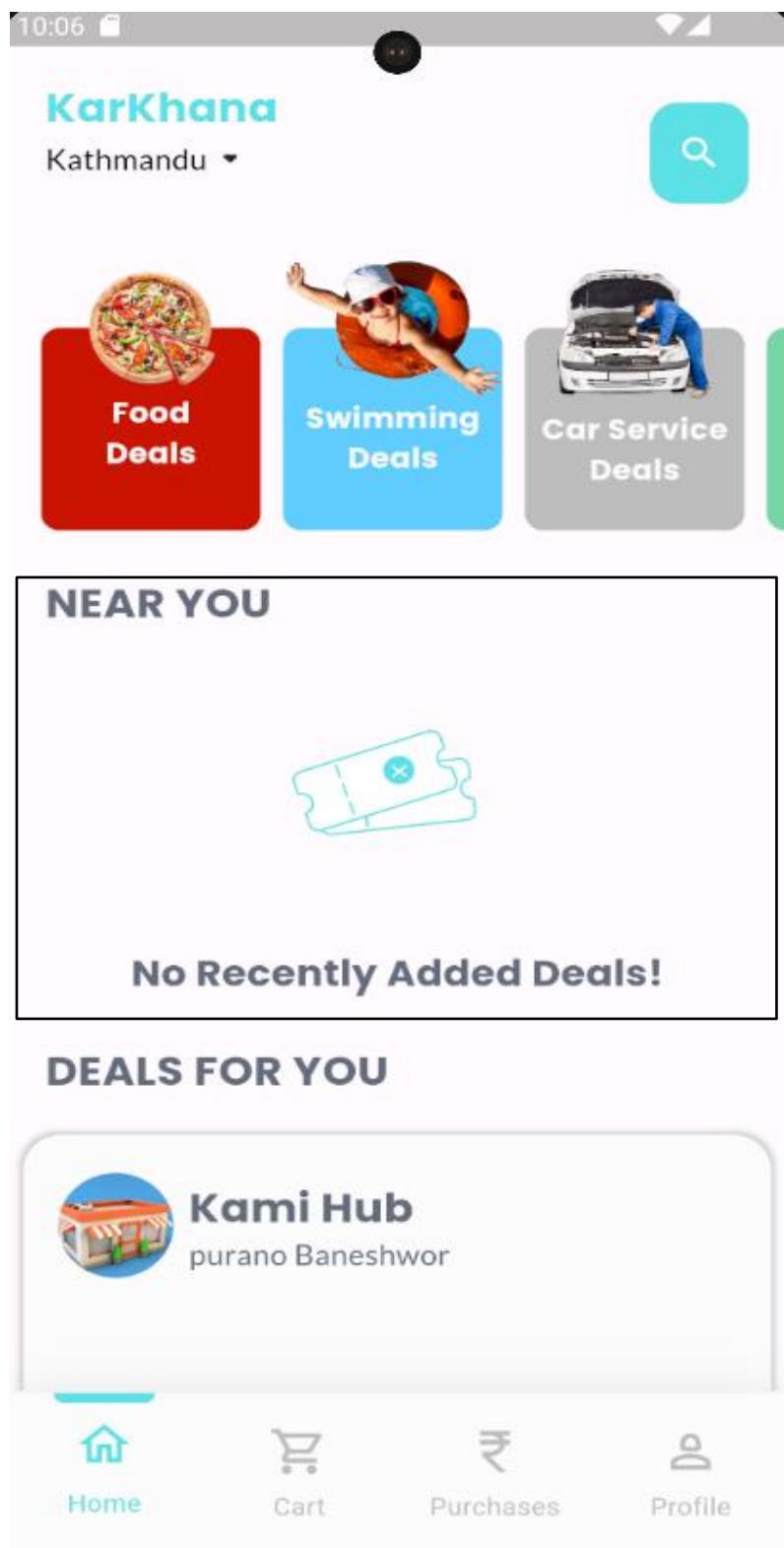
UI

Figure 200: Location Based Recommendation in Homepage.

## Code

Figure 201: Location based recommendation system code -1.

```
, // SIZEDBOX
    Center(
      child: SingleChildScrollView(
        scrollDirection: Axis.horizontal,
        child: Row(
          children: snapshot.data
            .map<Widget>((e) {
              String VendorBusiness_Name =
                "${e.Business_name}";
              String VendorBusiness_Location =
                "${e.Business_location}";
              // String VendorBusiness_Image =
              //   "${e.Business_photo}";
              String Deals_Title =
                "${e.Deals_title}";
              // String Deals_Photo =
              //   "${e.Deals_photo}";
              int Deals_Price = e.Deals_price;
              String Deals_Desc =
                "${e.Deals_desc}";
              String Deals_category =
                "${e.Deals_category}";
              String Deals_condition =
                "${e.Deals_condition}";
              int Deals_Discount =
                e.Deals_discount;
              String Deals_applicable_time =
                "${e.Deals_applicable_time}";
              String Deals_applicable_day =
                "${e.Deals_applicable_day}";
              String Deals_expiry_date =
                "${e.Deals_expiry_date}";
              int Deal_user = e.Deals_user;
              String Vendor_Phone =
                "${e.Business_Phone}";
              return GestureDetector(
                onTap: () {
                  Navigator.of(context).push(
                    PageTransition(
                      type: PageTransitionType
                        .rightToLeftWithFade,
                      child: scrollDealdetail(
                        VendorBusinessName:
                          VendorBusiness_Name,
                        VendorBusinessLocation:
                          VendorBusiness_Location,
                        Business_Phone:
                          Vendor_Phone,
                        // VendorBusinessPhoto:
                        //   VendorBusiness_Image,
                        Deals_Title:
                          Deals_Title,
                        Deals_Price:
                          Deals_Price,
                        // Deals_Photo: Deals_Photo,
                        Deals_Desc:
                          Deals_Desc,
                        Deals_category:
                          Deals_category);
                }
              );
            })
        )));
  
```

Figure 202: Location based recommendation system code -2.

```
// VendorBusiness_Imgae,
Deals_Title: Deals_Title,
Deals_Price: Deals_Price,
// Deals_Photo: Deals_Photo,
Deals_Desc: Deals_Desc,
Deals_category:
    Deals_category,
Deals_condition:
    Deals_condition,
Deals_Discount:
    Deals_Discount,
Deals_applicable_time:
    Deals_applicable_time,
Deals_applicable_day:
    Deals_applicable_day,
Deals_expiry_date:
    Deals_expiry_date,
Deal_user: Deal_user,
), // NearYouDeals
), // Padding
); // GestureDetector
[]).toList(), // Row
), // SingleChildScrollView
), // Center
SizedBox(
width: 50.h,
) // SizedBox
],
); // Column
}
} else {
return Padding(
padding: EdgeInsets.symmetric(vertical: 18),
child: Column(
children: [
Lottie.asset(
"assets/lottie/search_empty.json",
),
BigText(
text: "No Deals Found!",
), // BigText
BigText(text: "Search Other Deals."),
],
), // Column
); // Padding
}
}
}, // FutureBuilder
```

Figure 203: Location based recommendation system code -3.

## API

```
You, 2 hours ago | 1 author (You)
class FetchLocationDeals {
    var data = [];
    List<Deals> results = [];
    String api = '$baseUrl/accounts/deals/';
    Future<List<Deals>> getLocationDeals({String? query}) async {
        var url = Uri.parse(api);
        var res = await http.get(url);
        try {
            if (res.statusCode == 200) {
                data = json.decode(res.body);
                results = data.map((e) => Deals.fromJson(e)).toList();
                if (query != null) {
                    results = results
                        .where((element) => element.Business_location!
                            .toLowerCase()
                            .contains(query.toLowerCase()))
                        .toList();
                }
            } else {
                print('api error');
            }
        } on Exception catch (e) {
            print('error: $e');
        }
        return results;
    }
}
```

*Figure 204: Location based recommendation system API.*

## viii. Feedback

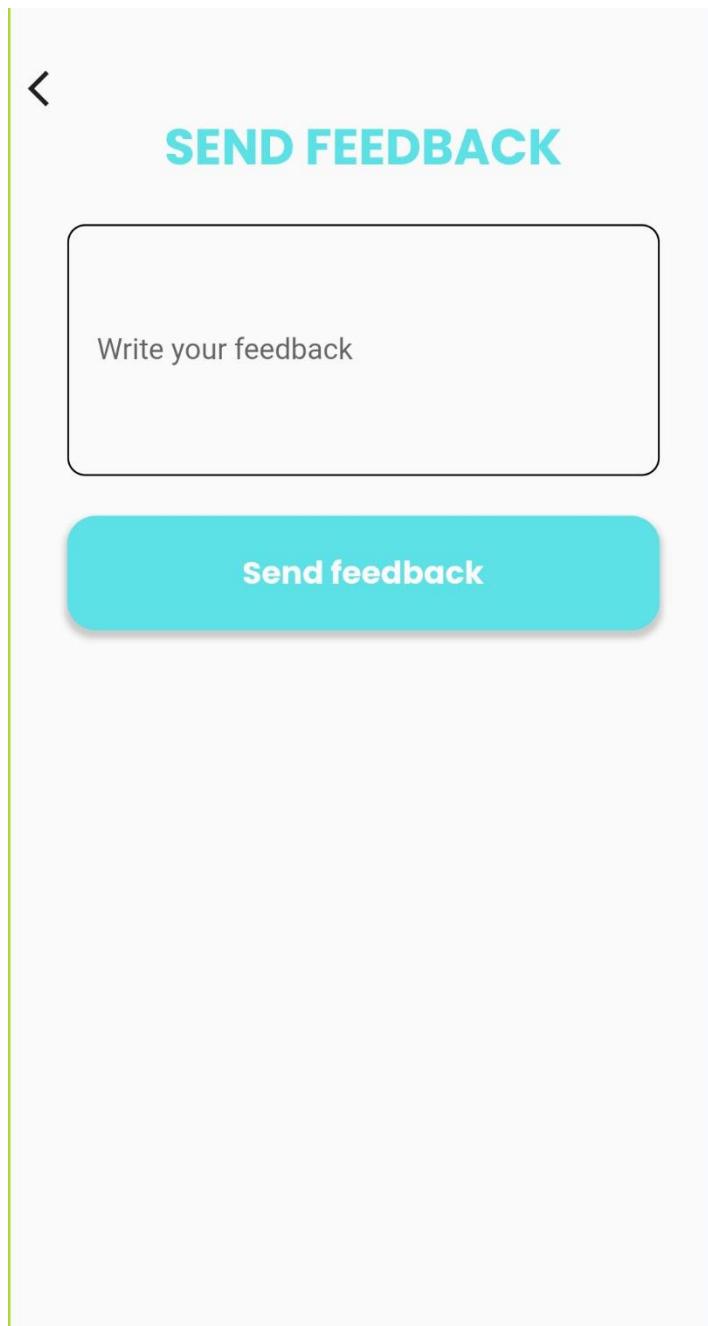
UI

Figure 205: Feedback UI.

Code

```
ButtonContainer(
    butborderColor: ColorsOn.secondaryColor,
    text: "Send feedback",
    butColor: ColorsOn.secondaryColor,
    onClick: () async {
        final isValidForm =
            _formKey.currentState!.validate();
        if (isValidForm) {
            await sendFeedback(id, id, userMessage.text,
                "${user.email}", "${user.name}");
            // ignore: use_build_context_synchronously
            showDialog(
                context: context,
                builder: (context) {
                    return AlertDialog(
                        title: BigText(
                            text: "Send Feedback",
                            color: ColorsOn.secondaryColor,
                        ), // BigText
                        content: NormalText(
                            text:
                                "Your feedback was recorded successfully.",
                            maxLine: 2,
                        ), // NormalText
                        actions: [
                            SimpleDialogOption(
                                child: const Text("Close"),
                                onPressed: () {
                                    Navigator.pop(context);
                                },
                            ) // SimpleDialogOption
                        ],
                    ); // AlertDialog
                });
            setState(() {
                userMessage.clear();
            });
        }
    },
);
```

Figure 206: Feedback Code.

## API

```
Future<dynamic> sendFeedback(int userMain, int id, String message, String email,
    String customerName) async {
  String ID = id.toString();
  String userMAIN = userMain.toString();
  Map<String, dynamic> data = {
    "customer_id": ID,
    "Customer_Name": customerName,
    "Customer_Email": email,
    "Feedbacks": message,
    "Customer_user": userMAIN,
  };
  var api = Uri.parse('$baseUrl/accounts/Feedback/');
  var res = await http.post(api, body: data);
  if (res.statusCode == 200 || res.statusCode == 201) {
    return 1;
  } else {
    return Fluttertoast.showToast(
      msg: "Sorry! Something went wrong while sending you email",
      fontSize: 18,
      gravity: ToastGravity.BOTTOM,
      backgroundColor: Colors.red,
      textColor: Colors.white);
  }
}
```

Figure 207: Feedback API.

## ix. Edit Profile

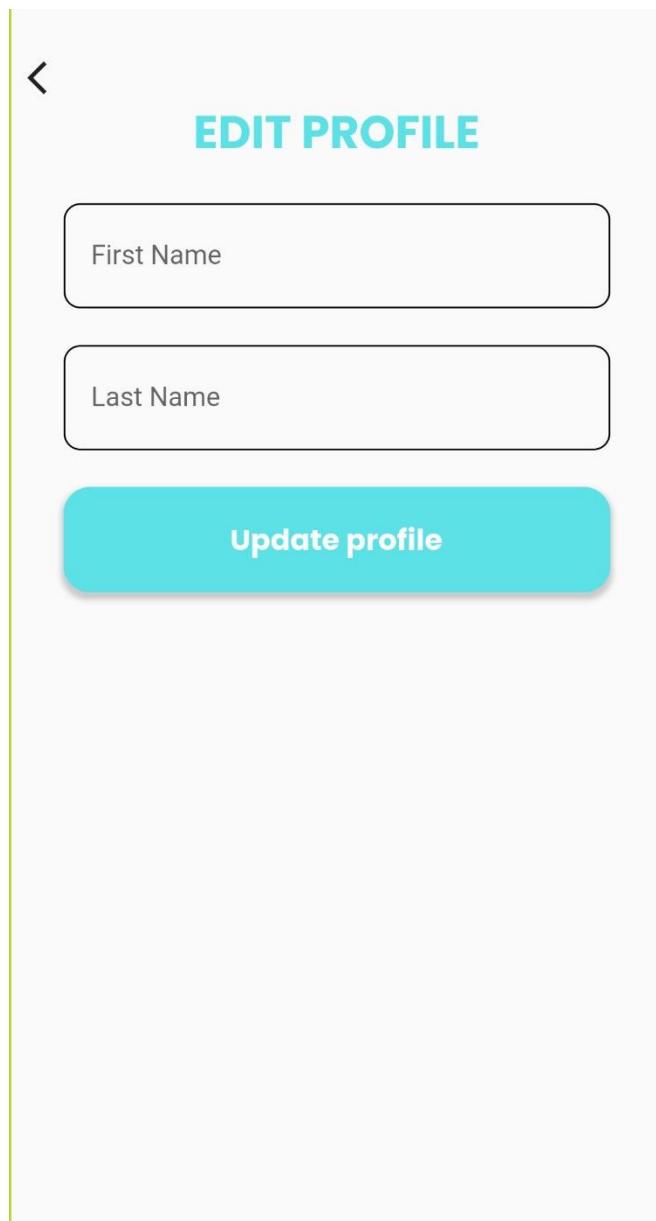


Figure 208: Edit profile UI.

### 3.7.4 Transition

This concludes the process of development. The program is made available to the public or clients at this phase. The product may be updated or modified in response to user input. It is the deployment procedure.

Post survey of the application was done in this phase.

The screenshot shows a survey titled "KarKhana". The survey introduction states: "KarKhana is a mobile application for people and businesses in Nepal which enables businesses to showcase their discounts and offers to customers/users. Users can easily sign up to buy deals from places like restaurants and salons. The app is super easy to use - just enter your phone number to get started. KarKhana helps users find awesome deals and lets businesses promote special offers. It's all about making a great marketplace for deals that benefits both customers and businesses in Nepal." Below the introduction is a question field labeled "Name \*". A "Short-answer text" input box is present. Another question field is labeled "How do you rate the UI of KarKhana System? \*". A rating scale from 1 to 5 is provided, with "Very Bad" at 1 and "Very Good" at 5. The interface includes standard survey navigation buttons like "Questions", "Responses 27", and "Settings".

Figure 209: Post Survey in Transition Phase.

And with the feedback provided by respondents through this survey, the application will be further update to be better and more favorable to both users and vendors. And with this, a complete documentation of the report was also created that described all the working mechanism of the application.

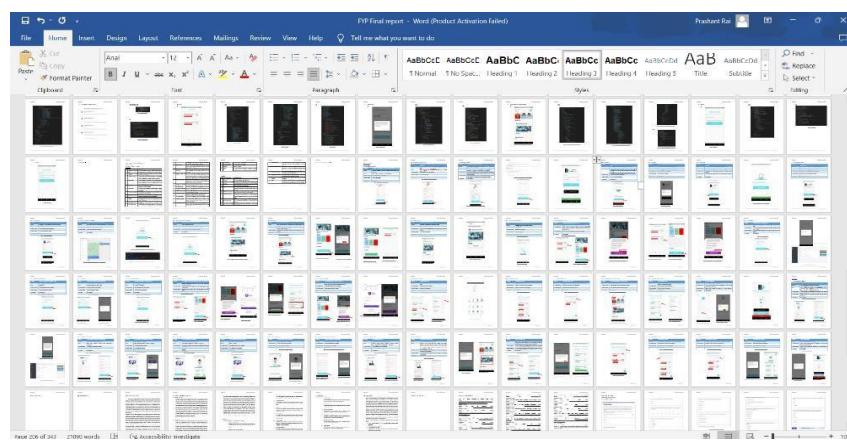


Figure 210: Documentation of the application.

## CHAPTER 4: TESTING AND ANALYSIS

### 4.1 TEST PLAN

#### 4.1.1 Unit Testing, Test Plan

##### For Customer

S.N.	Test Case Objective	Expected outcome
1.	Registration empty field error	As a user, I should not be able to register with empty fields in registration page. And field validation error should be displayed.
2.	Registration with invalid information.	As a vendor, I should get field validation error while I enter invalid information while registering
3.	Registration Success	As a user, when I enter my registration details, it should navigate to my token page where email verification will be completed. After successfully validating, I should be navigated to login page.
4.	Invalid Token	As a user, an email would be sent to me for verification which will contain a token that is used in the token page in application for further registration process. So, when I entered the wrong token, system should display error on textfield.
5.	login error	As a user, when I try to login with wrong credentials, a dialog box should pop up with error message.
6.	login success	As a user, I should be navigated to set location page if I entered the current login credentials. And a success message should also be displayed.
7.	Invalid Password length	As a user, when I enter password with length more than 8 then textfield error should be displayed
8.	Empty login field error	As a user, I should not be able to login with empty fields in login page. And field validation error should be displayed.

<b>9.</b>	Save current location in database	As a user, I will press the set my current location button in the set location page and when I do that, my current location should be stored in the database.
<b>10.</b>	Show current location in home screen	As a user, my current location should be displayed in my homepage.
<b>11.</b>	Add to cart success	As a user, when I select a deal and want to add it to my cart then first, a confirmation box is displayed if I want to add that deal in my cart or not. If I want to then I confirm, and that deal is added in my cart. Also, a success message is displayed.
<b>12.</b>	Search success	As a user, I will browse my desired deals in search bar. When I enter a word then if any deal related to that word should be displayed in the result page.
<b>13.</b>	Search with invalid deal title success	As a user, I will enter random words which are not similar to any deals in database, this should display 'no deals found' in the result page.
<b>14.</b>	Calculate coupon price success	As a user, I will buy multiple coupons from the cart. But before buying, the coupon details should be displayed along with the distinct calculated coupon price for different deals. (Coupon price should vary on each coupon of different deal depending upon discounts and price it is providing.)
<b>15.</b>	Send feedback success	As a user, I will send feedback to the system through send feedback page. A success message should be displayed and feedback should be posted in admin panel.
<b>16.</b>	Feedback field validation success	As a user, when I try to send empty feedback then field validation error should be displayed.
<b>17.</b>	Edit profile success	As a user, I should get a success message when I edit my profile.

<b>18.</b>	Edit profile field validation success	As a user, a field validation error should be displayed if I edit profile with empty fields.
<b>19.</b>	Purchase coupon success	As a user, I should be displayed with success message after the successful purchase of a coupon.
<b>20.</b>	Purchase coupon failed	As a user, I should be displayed with error/failure message if the purchase is incomplete or fails for further processes.
<b>21.</b>	Active coupon share	As a user, I should be able to share my active coupons. A sharing option page should be displayed.
<b>22.</b>	Display active coupon	As a user, my active coupons should be displayed in active coupon section of my purchases page after the successful purchase of the coupon.
<b>23.</b>	Display previous coupon	As a user, my previous coupons should be displayed in previous coupon section of my purchases page after the successful usage or redeem of the coupon.
<b>24.</b>	Current location recommended deals	As a user, I should be recommended with the deals according to my current location in homepage.
<b>25.</b>	Generated coupon displayed	As a user, after generating coupon, I should be able to view the details of the coupon.
<b>26.</b>	Logout	As a user, if I press the logout button, a confirmation box should be displayed and if I confirmed then I should be navigated to welcome screen with also displaying logout success message.

Table 17: Test Plan for Unit Testing for Customer.

**For Vendor**

S.N.	Test Case Objective	Expected outcome
1.	Post registration details Success	As a vendor, I should be displayed with success message when I post the registration details successfully.
2.	Registration Field validation success	As a vendor, I should not be able to register with empty fields in registration page. And field validation error should be displayed.
3.	Login error	As a vendor, when I try to login with wrong credentials, a dialog box should pop up with error message.
4.	Login success	As a vendor, I should be navigated to homepage if I entered the current login credentials. And a success message should also be displayed.
5.	Empty login field error	As a vendor, I should not be able to login with empty fields in login page. And field validation error should be displayed.
6.	Upload business profile	As a vendor, I should be displayed with success message when I upload my business to the system. And also, my business profile should be displayed in customer side with along with my added deals.
7.	Add deals before uploading business profile error	As a vendor, I should be displayed with a error message when I try to add deals before uploading my business profile to the system.
8.	Add deals field validation error	As a vendor, I should get field validation error when I add deal with empty field.

<b>9.</b>	Add deals success	As a vendor, I should get a success message when I add deals to the system. And the deal should be displayed in the customer side too.
<b>10.</b>	Confirm coupon success	As a vendor, to redeem or for usage of the coupon, I should be able to verify it. From the active coupon in my purchase section, I select a coupon and press confirm button. After that the status of the coupon should be changed from 'Active' to 'Used'.
<b>11.</b>	Display active coupons	As a vendor, I should be displayed with active coupons of my deal after customers successfully purchases it.
<b>12.</b>	Display previous coupons	As a vendor, I should be displayed the previous coupon used by customer, after I confirm and verify the coupon.
<b>13.</b>	Recently added deals displayed	As a vendor, I should be displayed my recently added deals in my homepage.
<b>14.</b>	Logout	As a vendor, if I press the logout button, a confirmation box should be displayed and if I confirmed then I should be navigated to welcome screen with also displaying logout success message.

Table 18: Test Plan for Unit Testing for Vendor.

#### 4.1.2 System Testing, Test Plan

S.N.	Test Case Objective	Expected outcome
1.	Connectivity test	As user, I will turn off internet (WIFI) from my device which should stop my application and prompt me with network unavailable message.
2.	Compatibility test	As user, I will use this application in mobile application, also in android tablet which should not show any error or any incompatibility.
3.	Running Application without online server	Try to run the application without running the Django project with has all the database structure and endpoints.
4.	Whole System walkthrough test for customer	As a Customer, I should not have any problem from logging in into the system to logging out of the system.
5.	Whole System walkthrough test for vendor	As a vendor, I should not have any problem from logging in into the system to logging out of the system.

Table 19: Test Plan for System Testing.

---

## 4.2 UNIT TESTING

---

### For Customer

#### 4.2.1 Registration Empty Field

<b>objective</b>	To get an error message if provided information are empty while registering.
<b>Actions</b>	Enter empty information in the field in the registration page and press signup button.
<b>Input:</b>	Name: " ", Email: " ", password: " ", confirm password: " "
<b>Expected results</b>	Field validation error must be displayed
<b>Obtained Results</b>	The error was displayed
<b>Conclusion</b>	Test Successful.

Table 20: Registration field validation testing table.

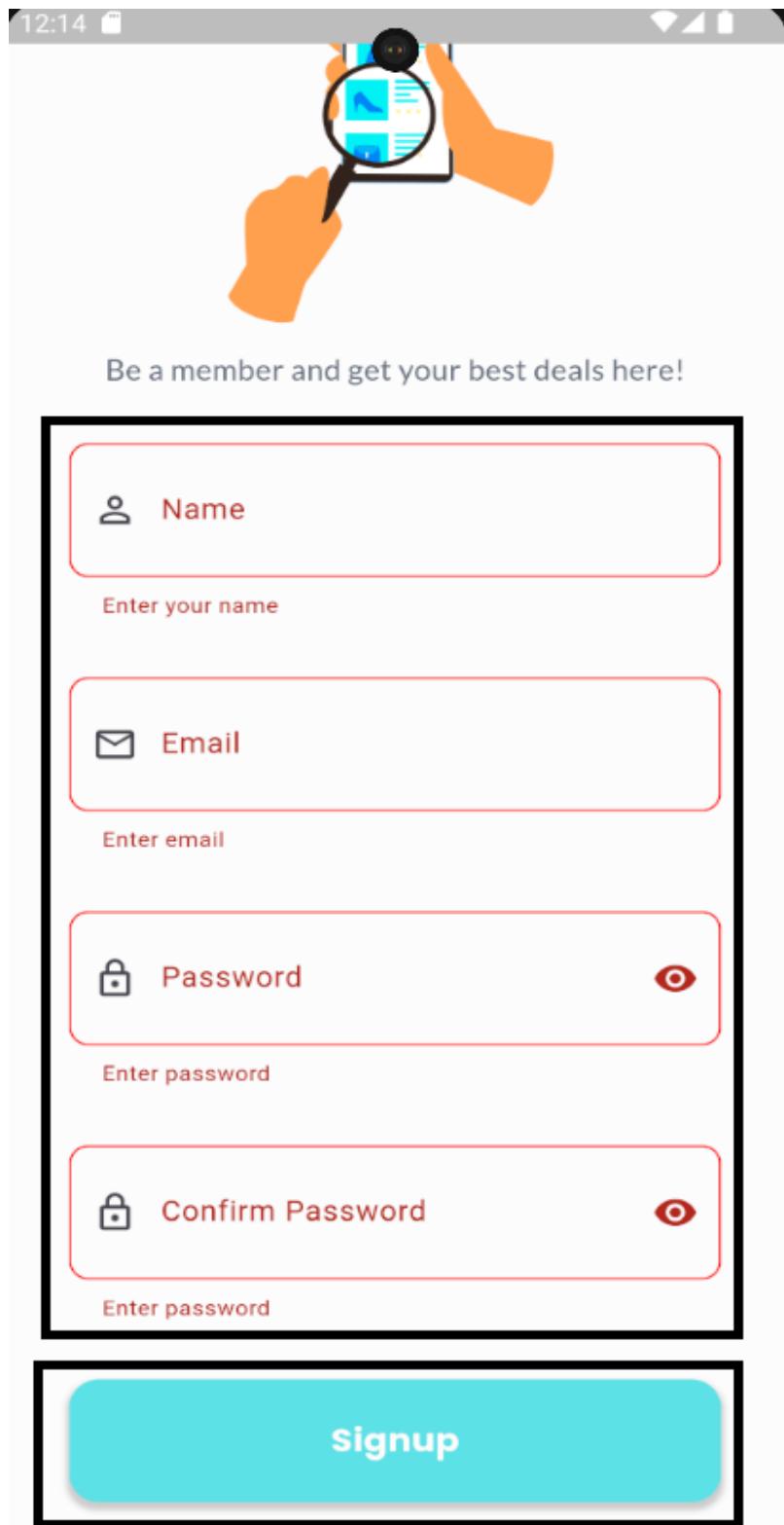


Figure 211: Registration field Validation Testing.

#### 4.2.2 Registration with invalid input

<b>objective</b>	To get an error message if provided information are invalid while registering.
<b>Actions</b>	Enter invalid information in the field in the registration page and press signup button
<b>Input:</b>	Name: "Ram rai", Email: "xxxxxxxxxxxx", password: "1234 ", confirm password: "abcd "
<b>Expected results</b>	Field validation error must be displayed
<b>Obtained Results</b>	The error was displayed
<b>Conclusion</b>	Test Successful.

Table 21: Registration with invalid input testing table.

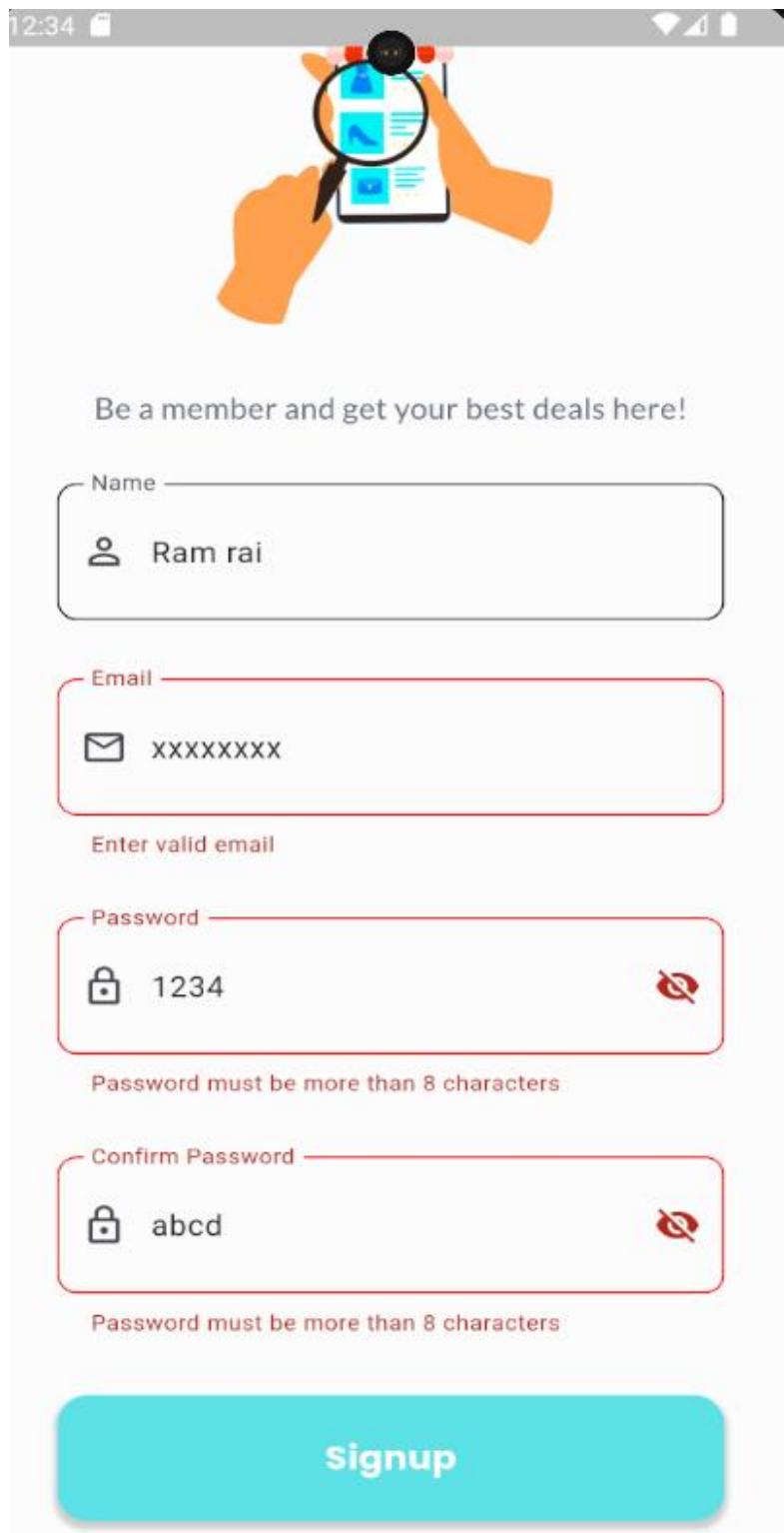


Figure 212: Registration field test with invalid input.

#### 4.2.3 Registration Process

objective	To register user in KarKhana system.
<b>Actions</b>	Enter registration details and enter signup button. Check email and use token and enter confirm button.
<b>Input:</b>	Name: "Bromish", Email: " <a href="mailto:np01cp4a20202@islingtoncollege.edu.np">np01cp4a20202@islingtoncollege.edu.np</a> ", password: "bromish@5270 ", confirm password: "bromish@5270 "
<b>Expected results</b>	User should be navigated to confirm email page then after validating token, user should be navigated to login screen and registration success message should be displayed
<b>Obtained Results</b>	User was navigated to confirm email page then after validating token, user was navigated to login page and success message was displayed
<b>Conclusion</b>	Test Successful.

Table 22: Registration success testing table.

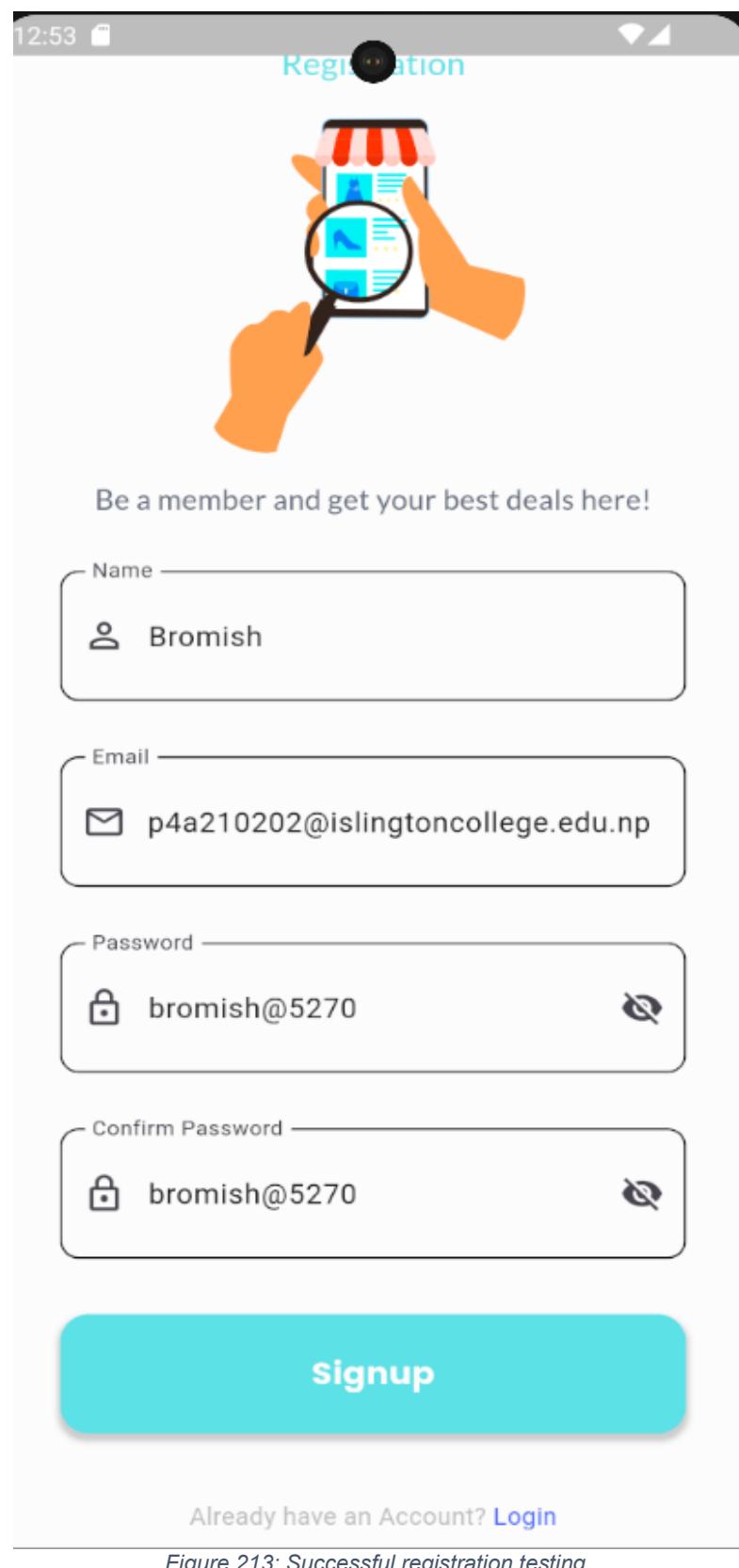


Figure 213: Successful registration testing.

#### 4.2.4 Token Invalid

<b>objective</b>	To get error message when wrong/invalid token is entered in token page
<b>Actions</b>	Enter wrong token code in token field and press confirm button.
<b>Input:</b>	<p>Actual Token:</p> <p>MTU:1ry5V4:LLWxQo1kpR3fjfG4E7_emmc8OMgNn8JxZhTj6bfyU0</p> <p>Used Token:</p> <p>MTI:1pmbWk:zCyVGoM1EV3wCea5sLIDZce6DEJq3Rc0LhfH5yBadD</p>
<b>Expected results</b>	Key error message should be displayed
<b>Obtained Results</b>	Key error message was displayed
<b>Conclusion</b>	Test Successful.

Table 23: Token Invalid testing table.

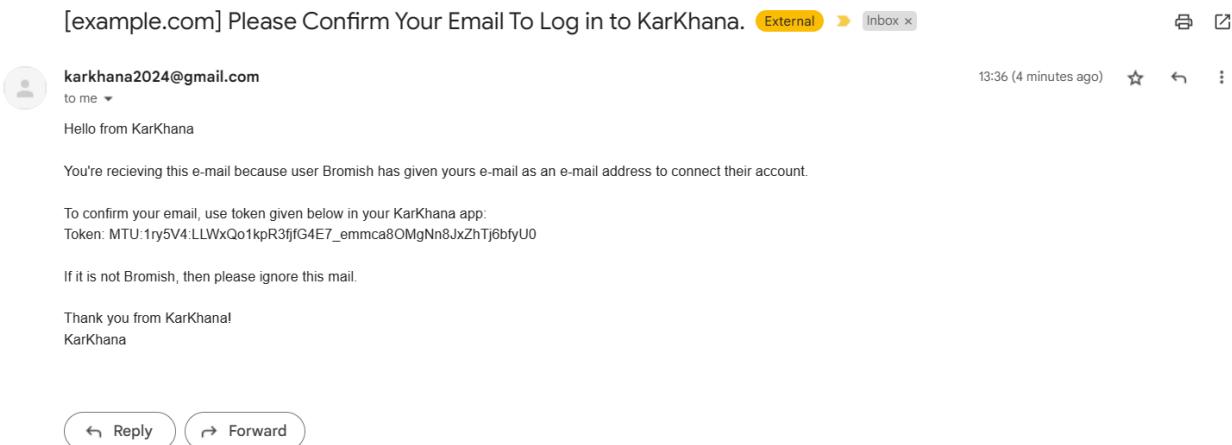


Figure 214: Token Sent in mail.

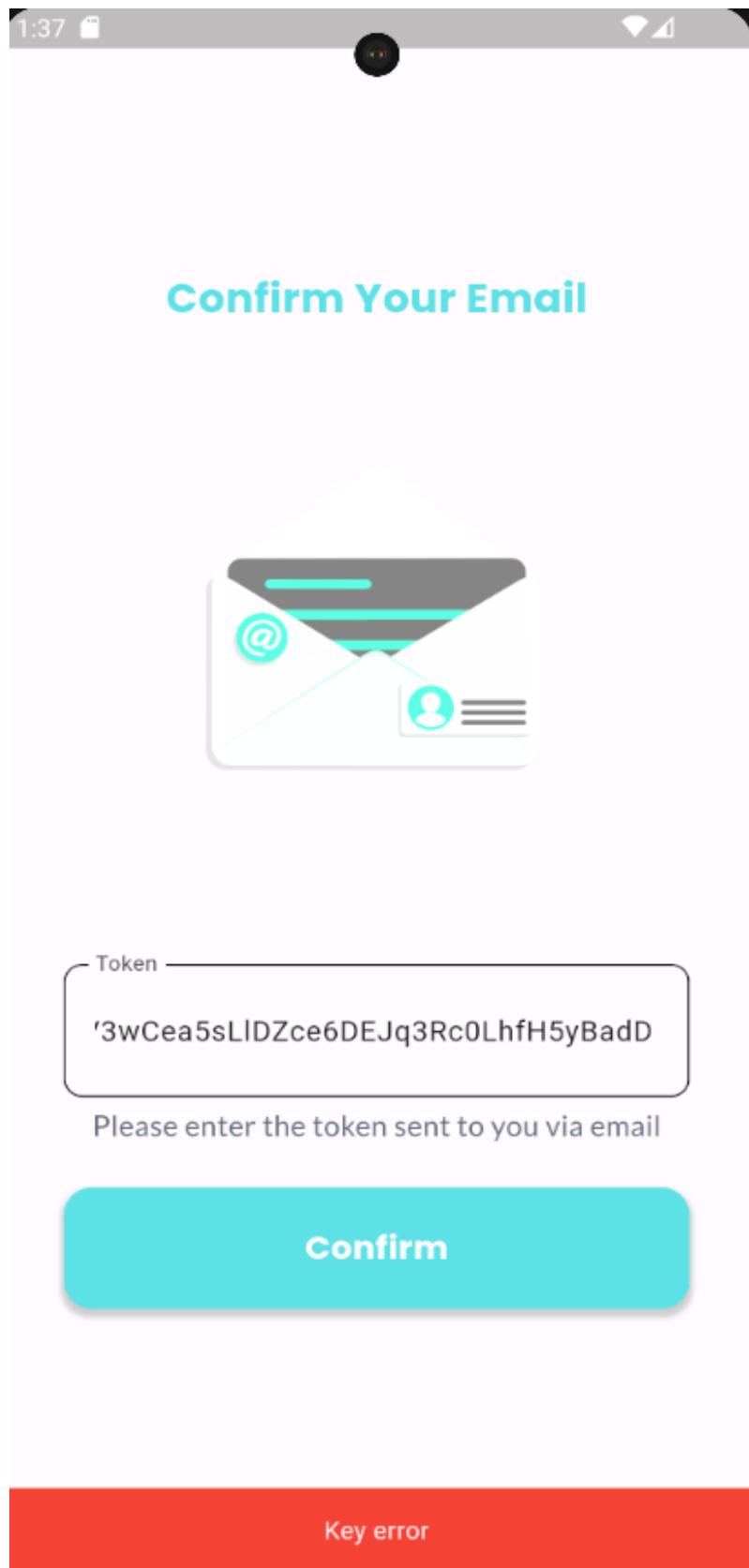


Figure 215: Invalid Token Testing.

#### 4.2.5 Login Error

<b>objective</b>	To get error message when invalid information is used while logging in.
<b>Actions</b>	Enter invalid Email/Password. And press login button.
<b>Input:</b>	Email: "mail@gmail.com", Password: "123psn@123"
<b>Expected results</b>	Error message should be displayed
<b>Obtained Results</b>	Error message was displayed
<b>Conclusion</b>	Test Successful.

Table 24: Login error testing table.

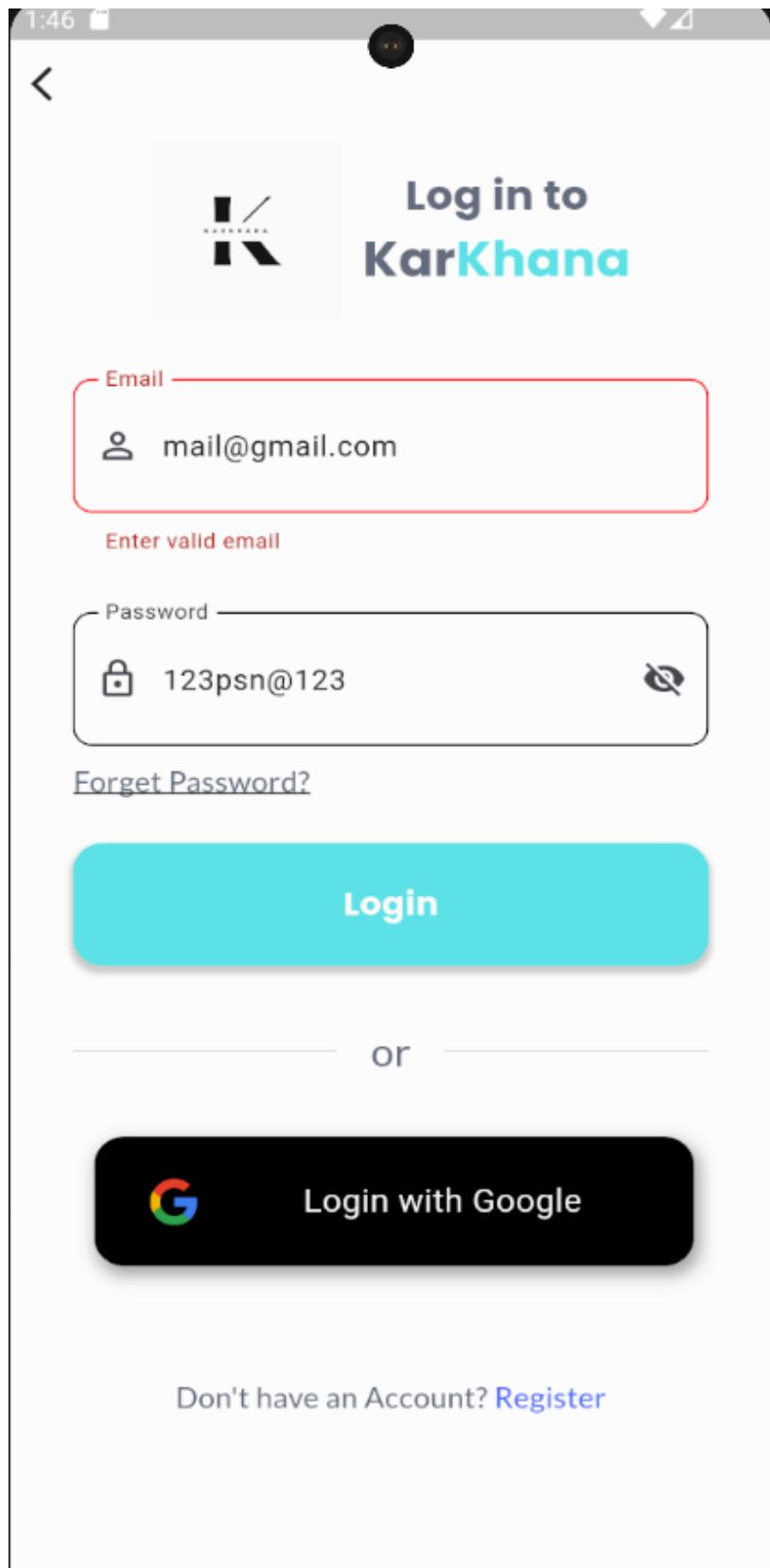


Figure 216: Invalid Email/Password Testing.

#### 4.2.6 Login Successful

<b>objective</b>	To navigate to location page and get a login success message after logging in.
<b>Actions</b>	Enter Login information in login page and press login button.
<b>Input:</b>	Email: "mail.nettv0212@gmail.com", Password: "12345@psn"
<b>Expected results</b>	Location page should be navigated, and login success message should be displayed.
<b>Obtained Results</b>	Location page was navigated, and login success message was displayed.
<b>Conclusion</b>	Test Successful.

Table 25: Login success testing table.

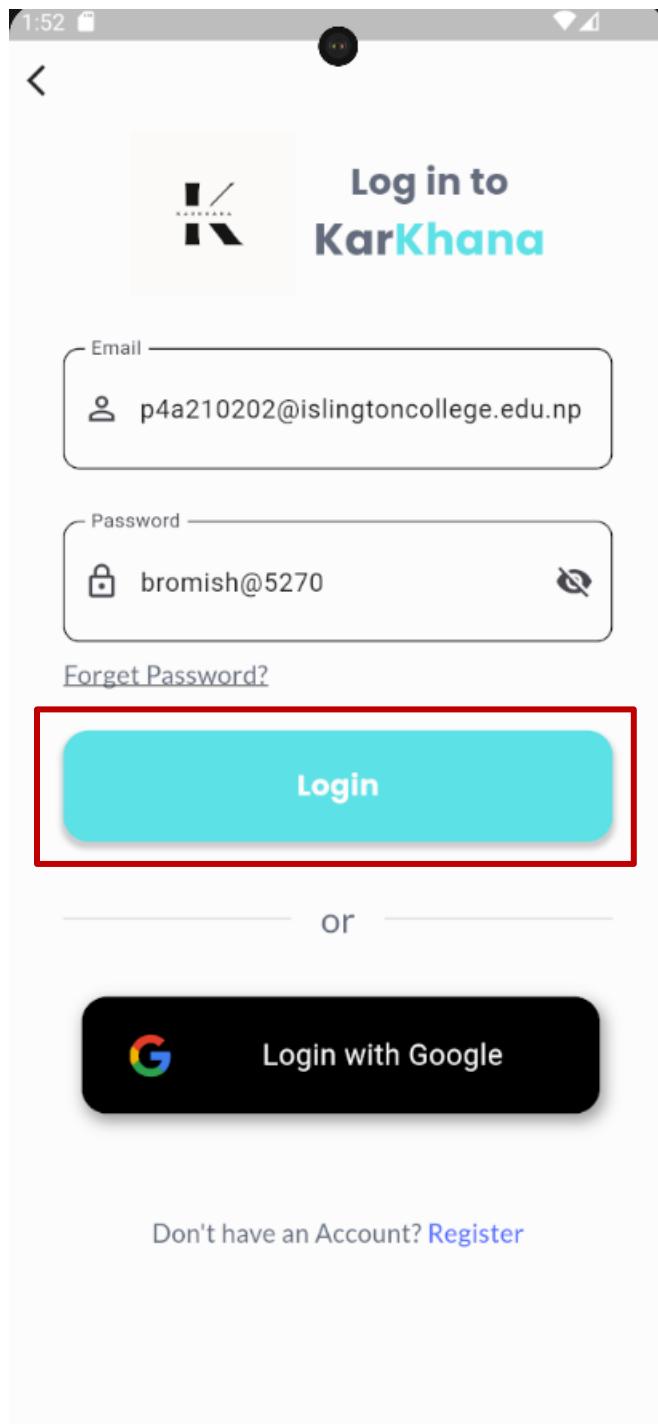


Figure 217: Login with valid credentials.

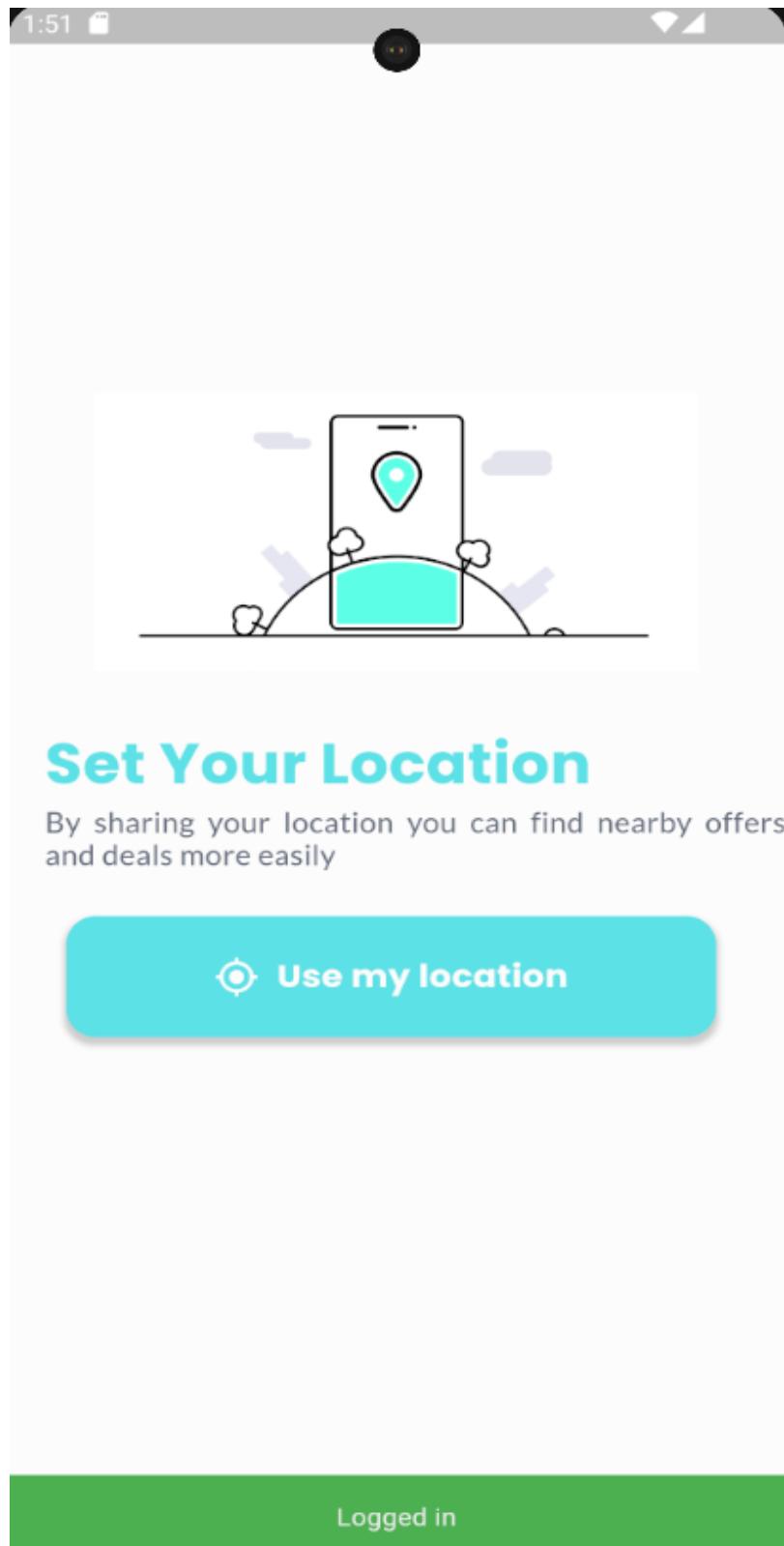


Figure 218: Logged in Successfully.

#### 4.2.7 Invalid password length

<b>objective</b>	To get error message when password with length less than 8 is used while logging in.
<b>Actions</b>	Enter password with length less than 8 and press login button.
<b>Input:</b>	Email: "np01cp4a210202@islingtoncollege.edu.np ", Password: "123"
<b>Expected results</b>	Password field error message should be displayed
<b>Obtained Results</b>	Password field error message was displayed
<b>Conclusion</b>	Test Successful.

Table 26: Invalid password length testing table.

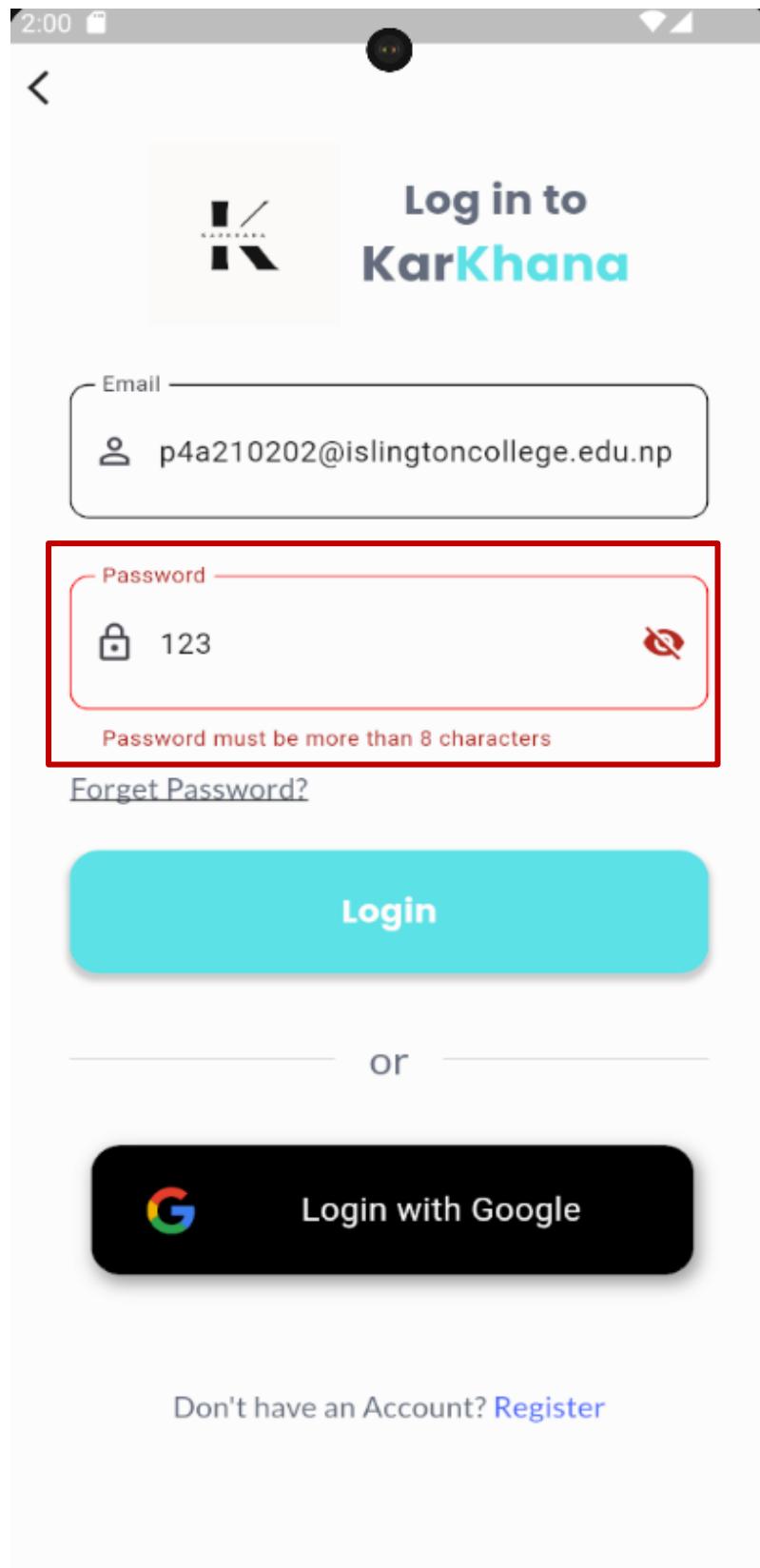


Figure 219: Invalid Password length testing.

#### 4.2.8 Empty login field

<b>objective</b>	To get error message when empty field was used while logging in.
<b>Actions</b>	Enter nothing in Email/Password field and press login button.
<b>Input:</b>	Email: “ ”; Password: “ ”
<b>Expected results</b>	Field Error message should be displayed
<b>Obtained Results</b>	Field Error message was displayed
<b>Conclusion</b>	Test Successful.

Table 27: Empty login field testing table.

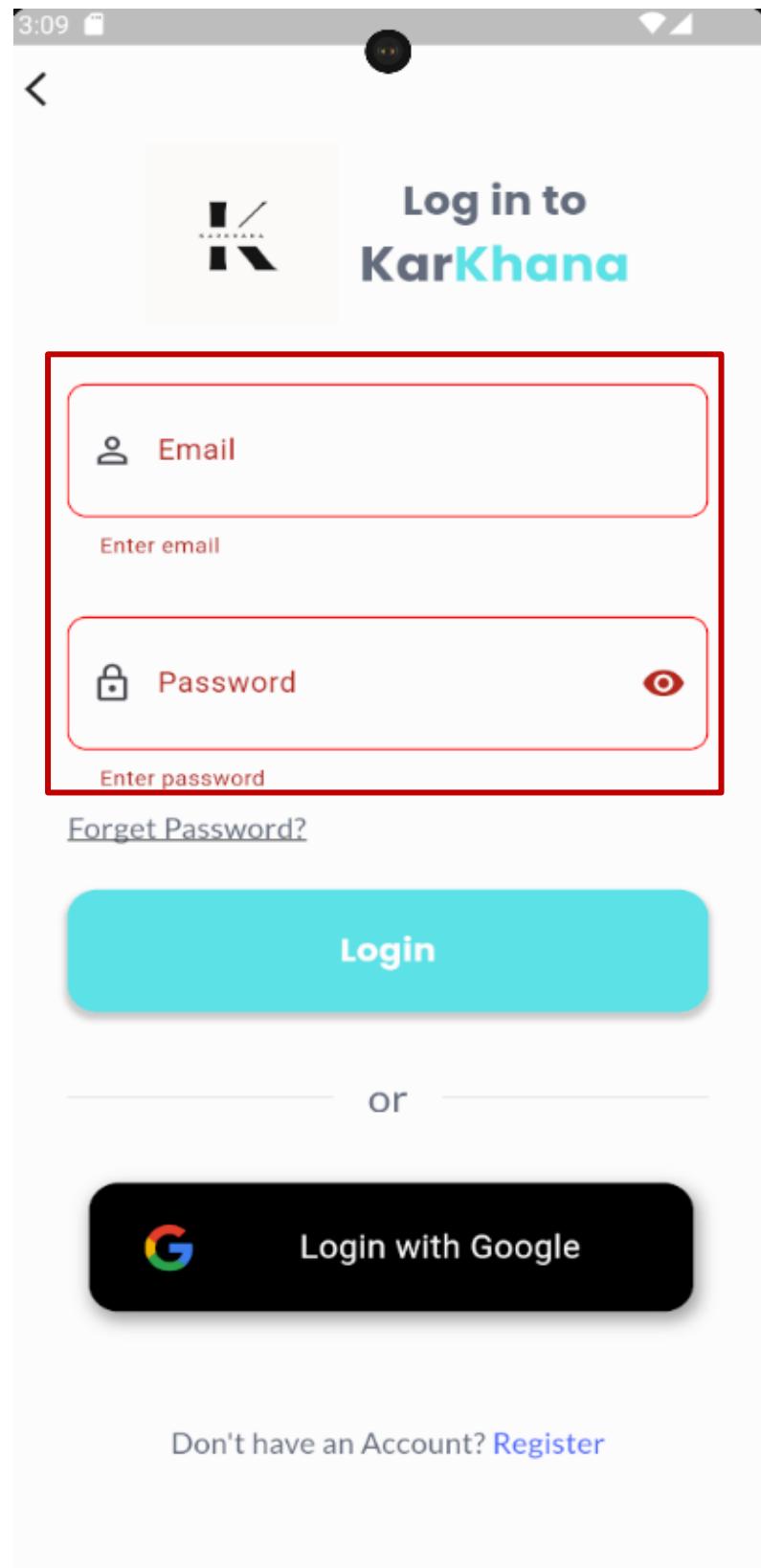


Figure 220: Empty Login Field Validation Testing.

#### 4.2.9 Save current location in database.

objective	To save current location of user in database.
Actions	Check current location in mobile device and press set current location button.
Expected results	Current location should be saved in database.
Obtained Results	Current location was saved in database.
Conclusion	Test Successful.

Table 28: Save current location in database testing table.

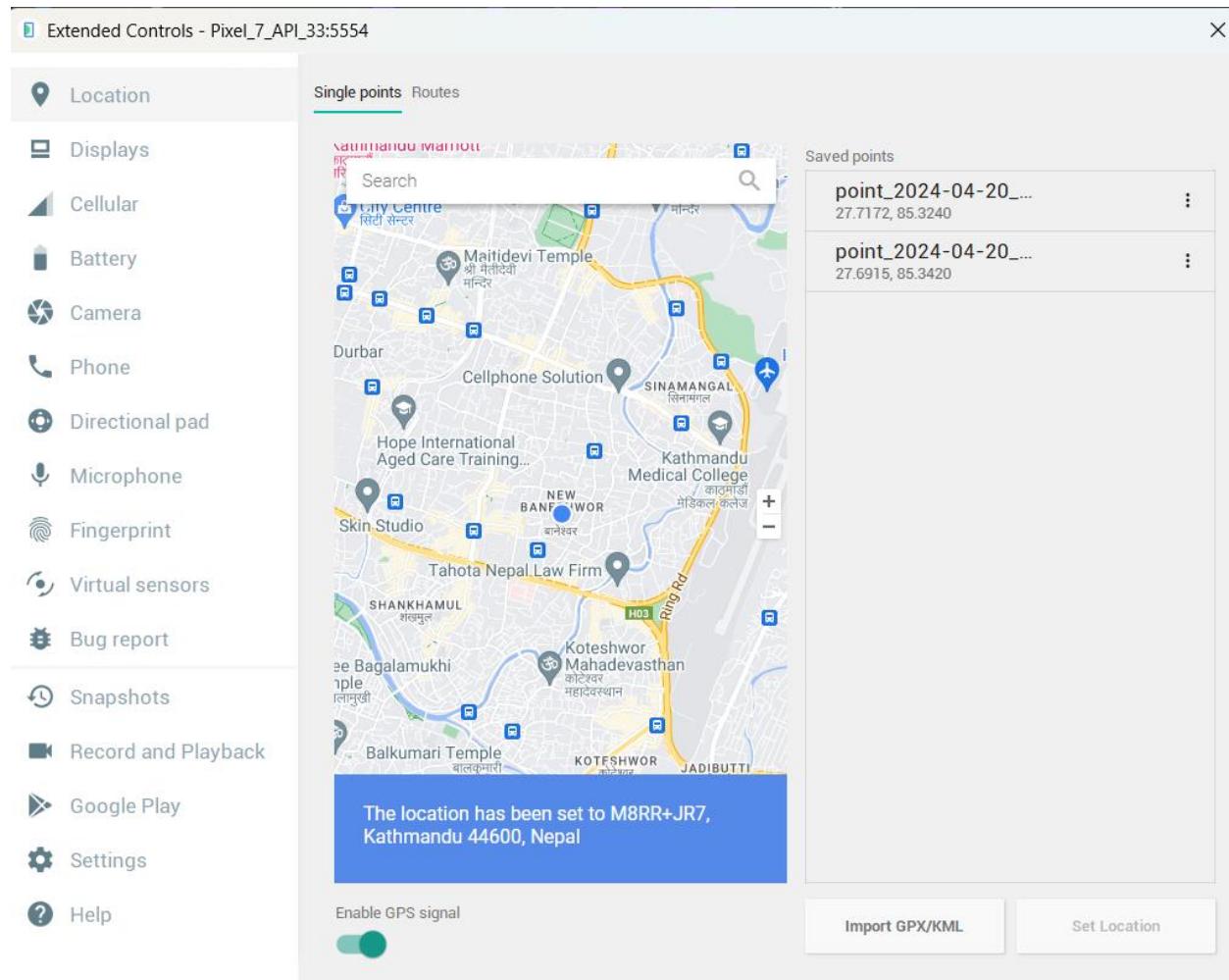


Figure 221: Setting Location of Emulator.

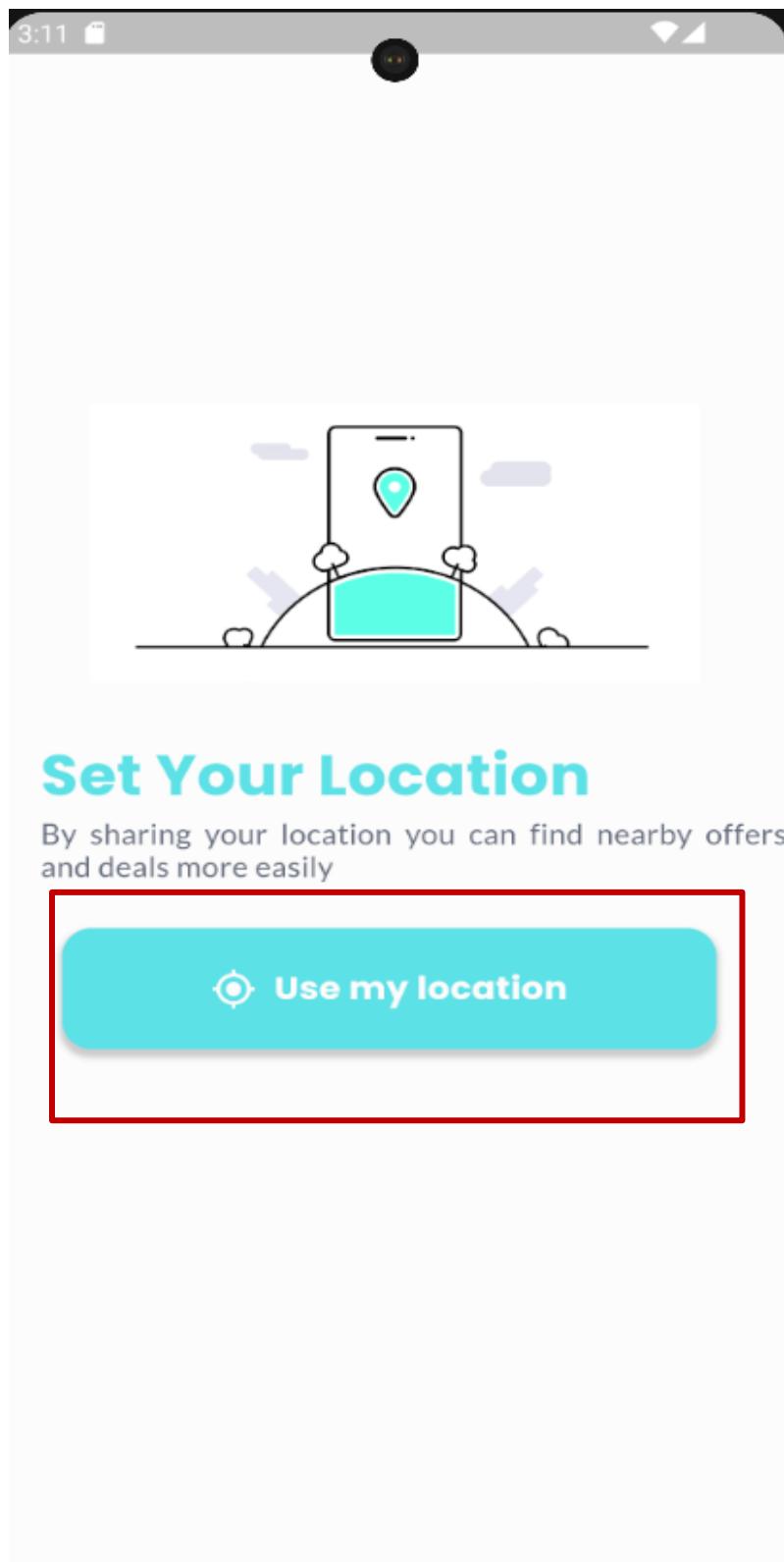


Figure 222: Using Current Location Testing.

User type: Customer

Username:

Full name: Bromish

Location: Kathmandu

Phonenumber:

Figure 223: Testing whether the location is set in database or not.

#### 4.2.10 Show current location in homepage

objective	To show current location of user in homepage.
<b>Actions</b>	Press set current location button.
<b>Expected results</b>	Current location of user should be displayed in homepage.
<b>Obtained Results</b>	Current location of user was displayed in homepage
<b>Conclusion</b>	Test Successful.

Table 29: Show current location in homepage testing table.

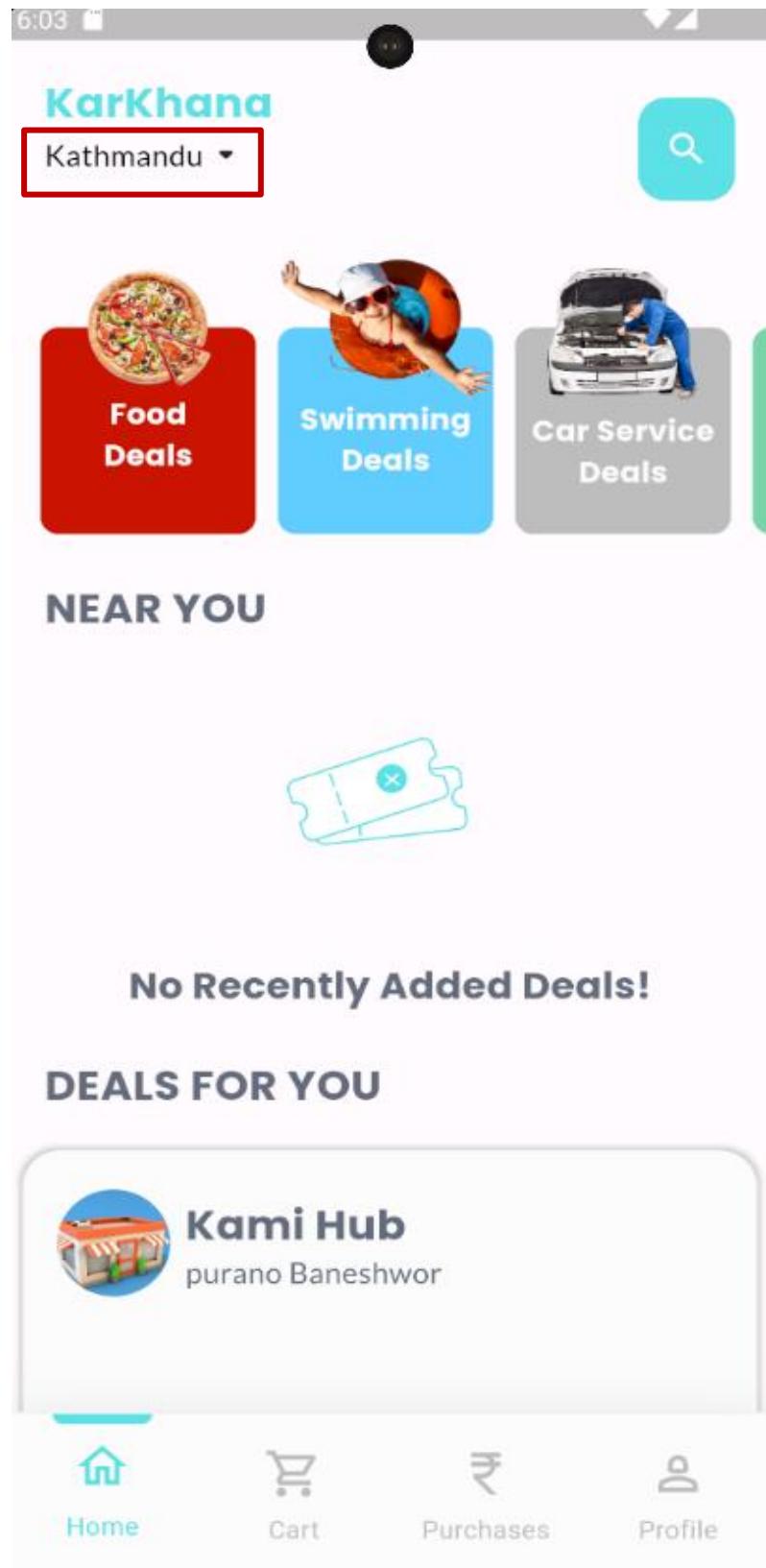


Figure 224: Show Current location in homepage testing.

**4.2.11 Add to cart**

objective	To show deals added in cart.
<b>Actions</b>	Select deal then press add to cart button. When a dialog box pops up then press yes.
<b>Expected results</b>	Success message should be displayed. Deal added to cart should be displayed in user's cart page.
<b>Obtained Results</b>	Success message was displayed. Deal added to cart was displayed in user's cart page.
<b>Conclusion</b>	Test Successful.

*Table 30: Add to cart testing table.*

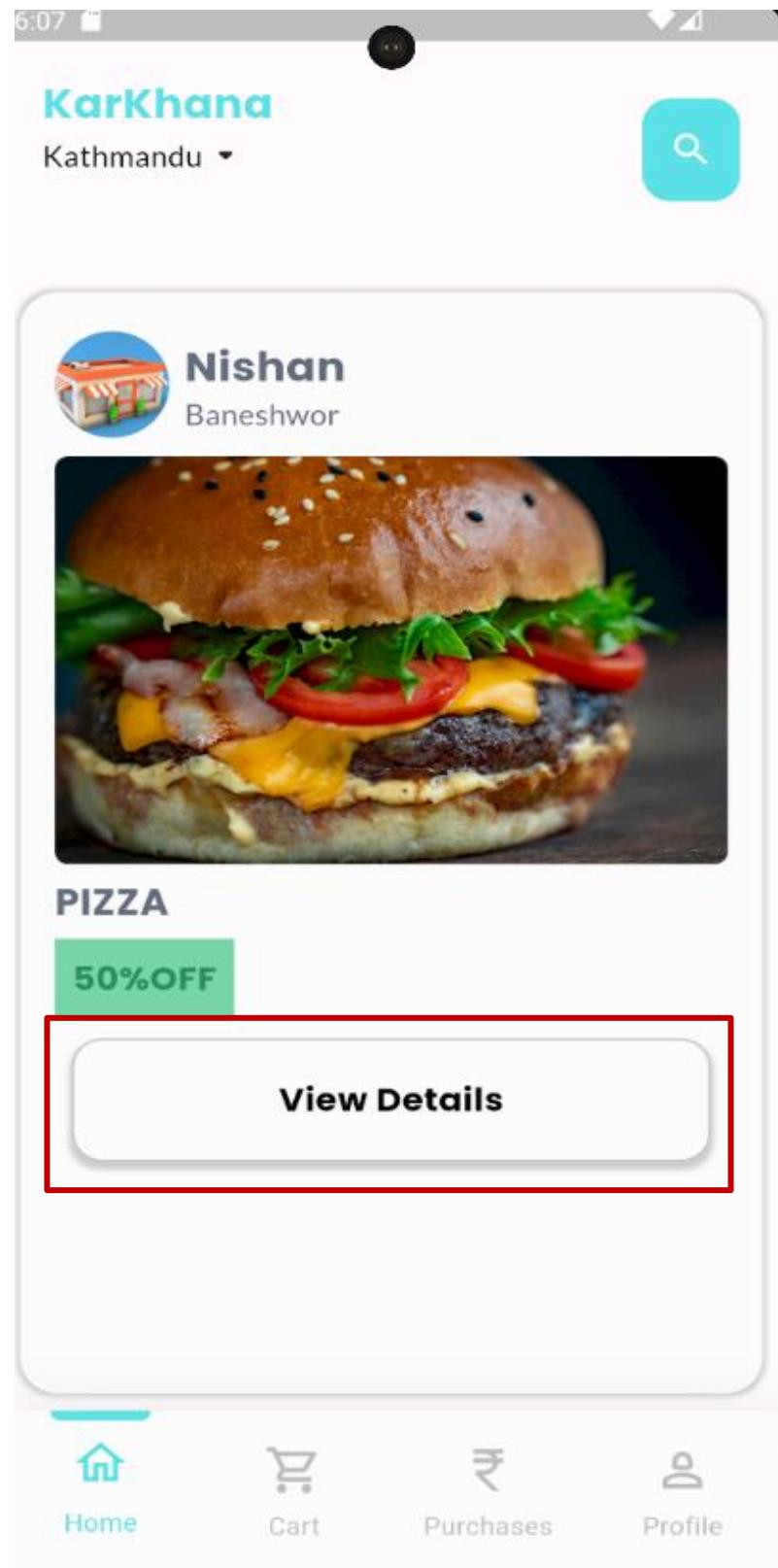


Figure 225: Add to Cart Testing -1.

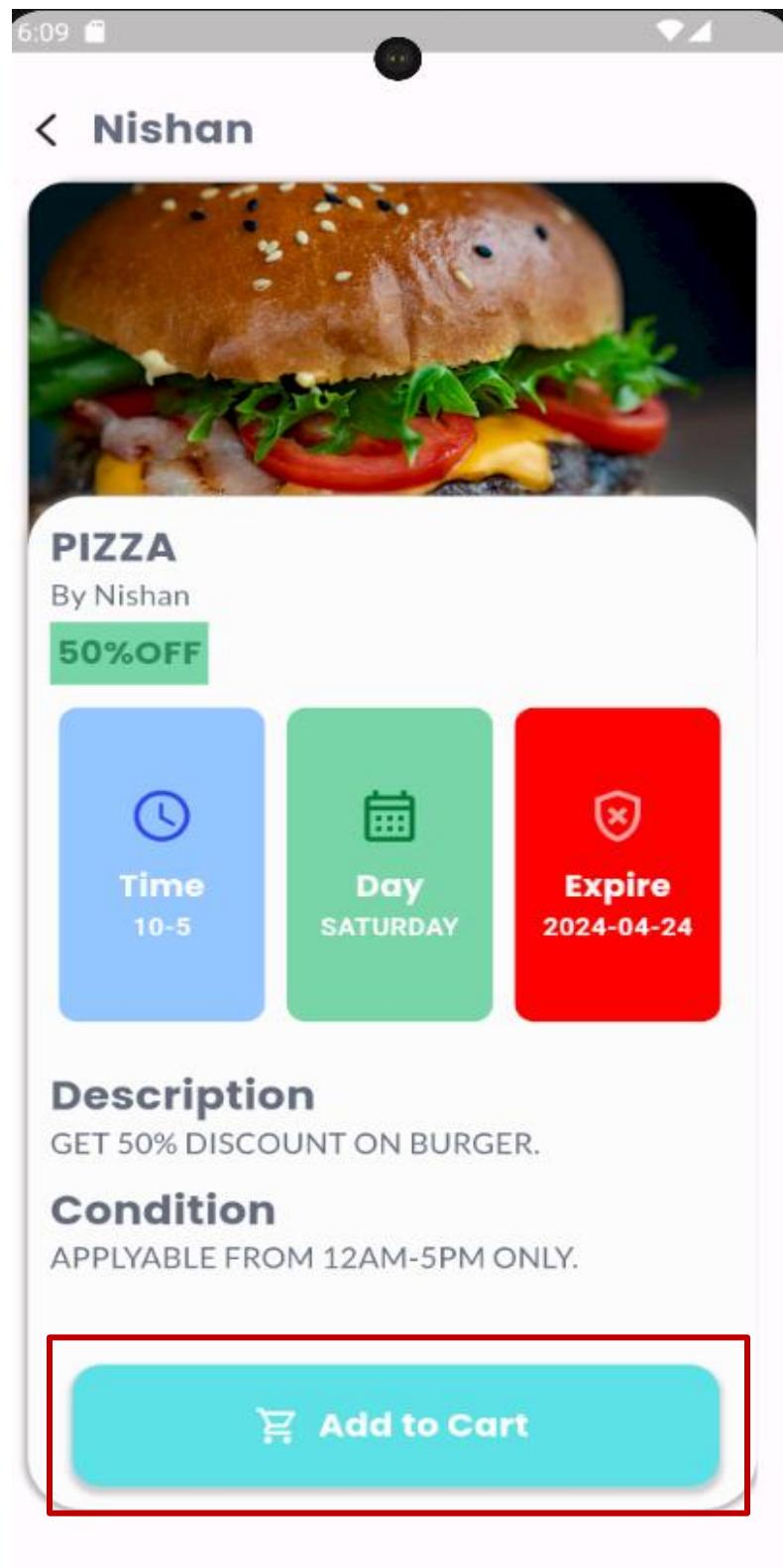


Figure 226: Add to Cart Testing -2.

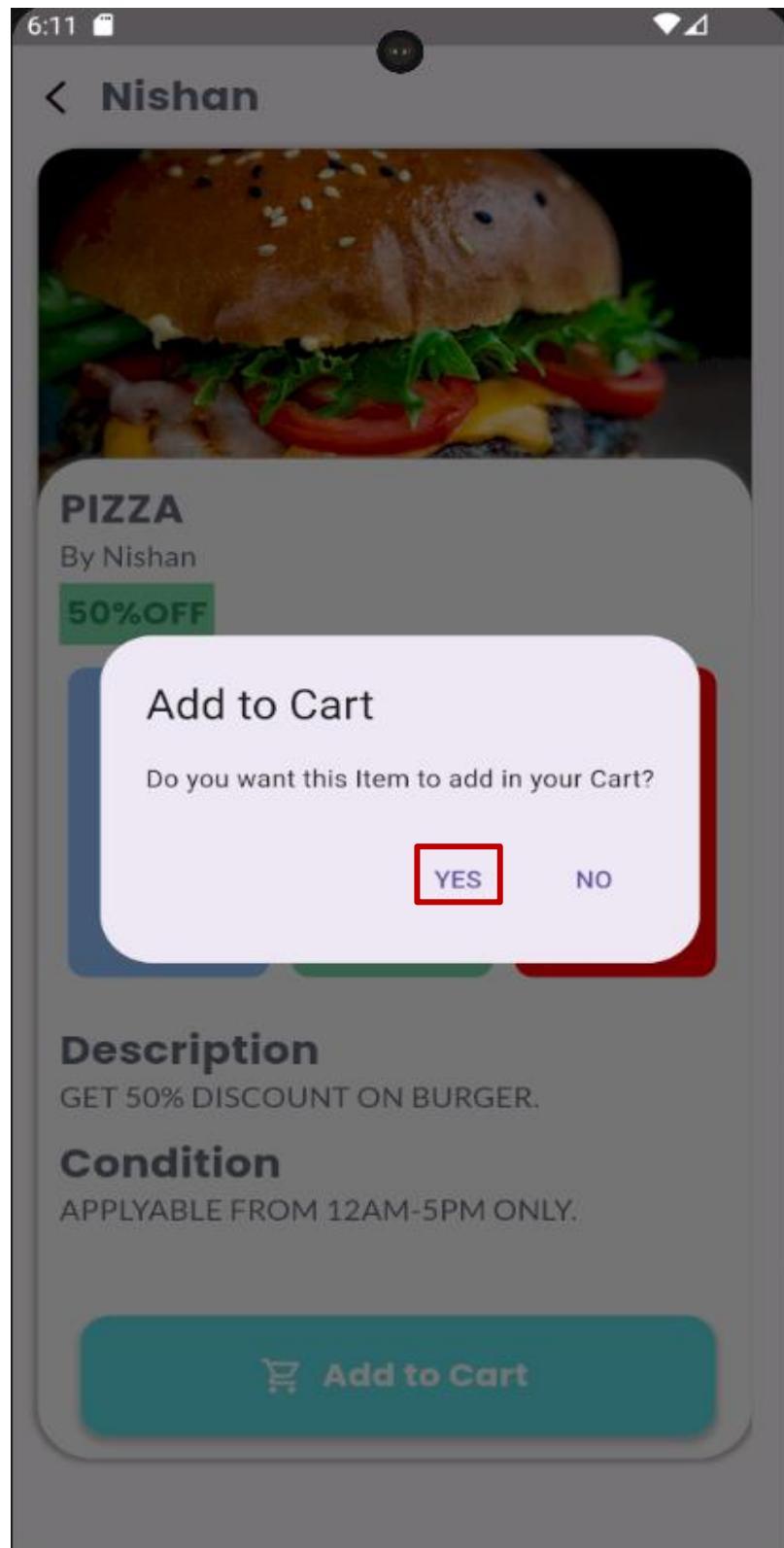


Figure 227: Add to Cart Testing -3.

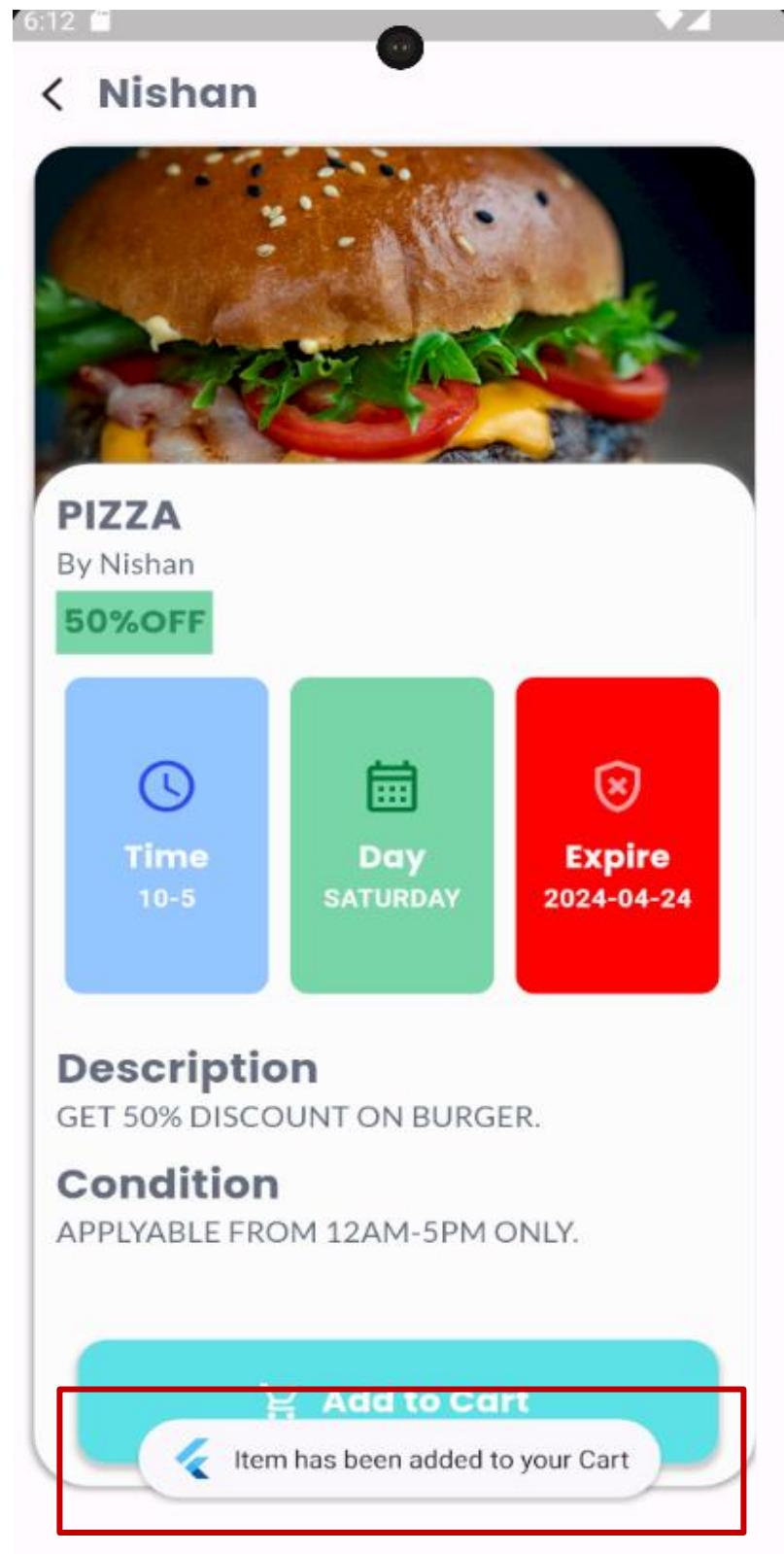


Figure 228: Add to Cart Testing -4.

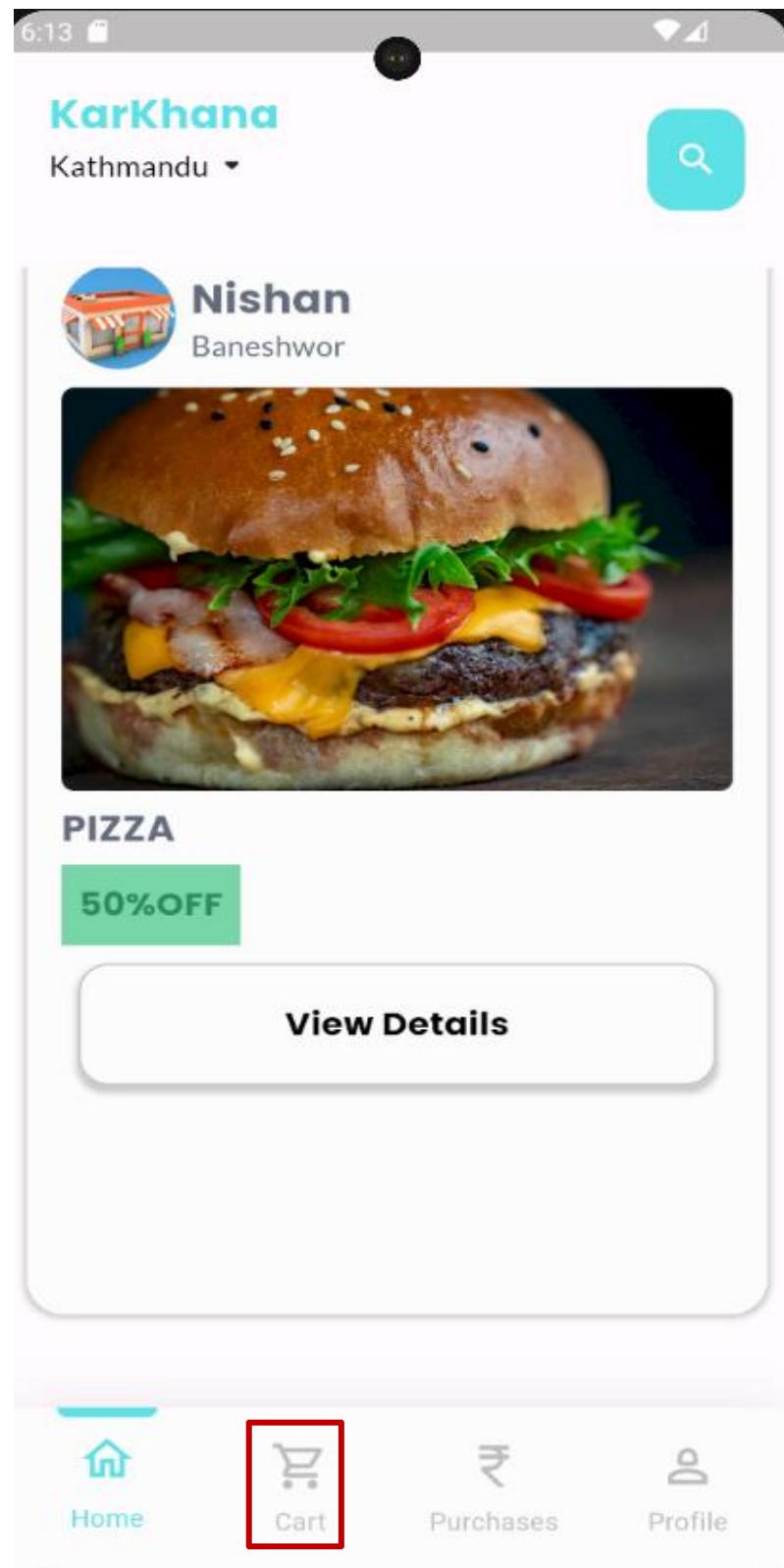


Figure 229: Add to Cart Testing -5.

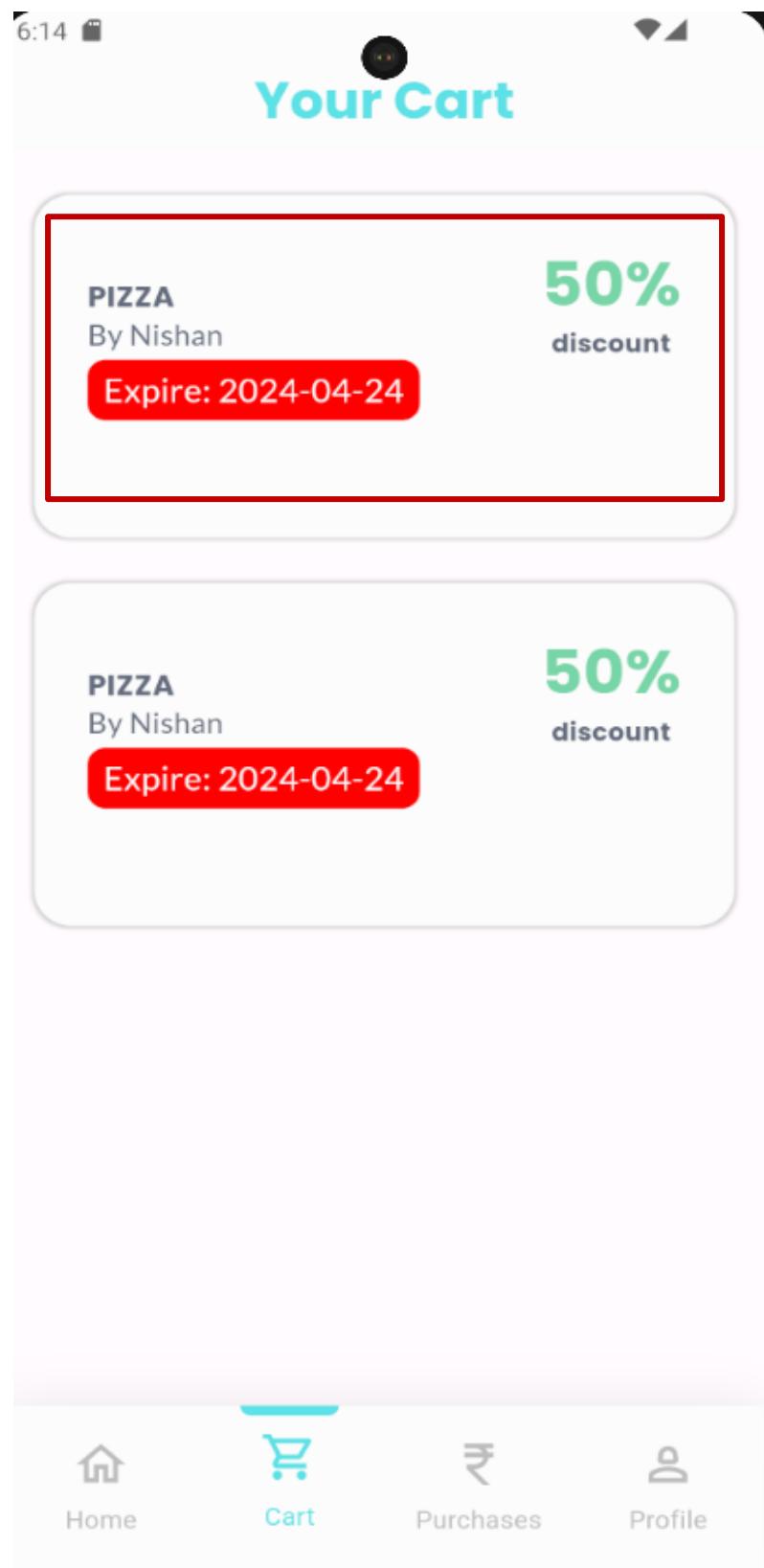


Figure 230: Add to Cart Testing -5.

#### 4.2.12 Search success

objective	To browse deals using search bar.
<b>Actions</b>	Enter a deal title related word in search bar and search.
<b>Input</b>	Search: "Pizza"
<b>Expected results</b>	Any deal's title from the database similar to the searched deal title should be displayed.
<b>Obtained Results</b>	Similar deals were displayed.
<b>Conclusion</b>	Test Successful.

Table 31: Search success testing table.

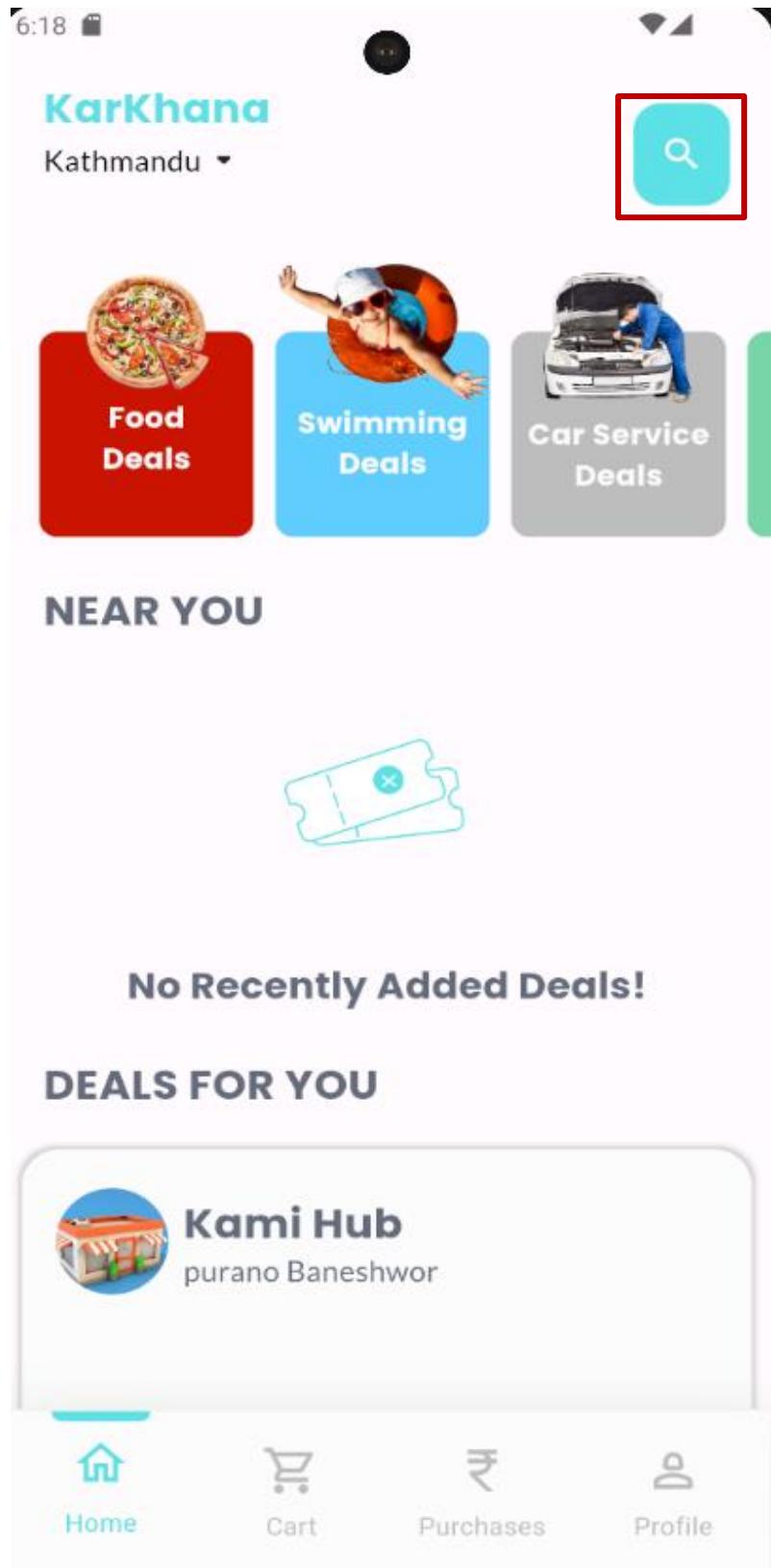


Figure 231: Search testing -1.

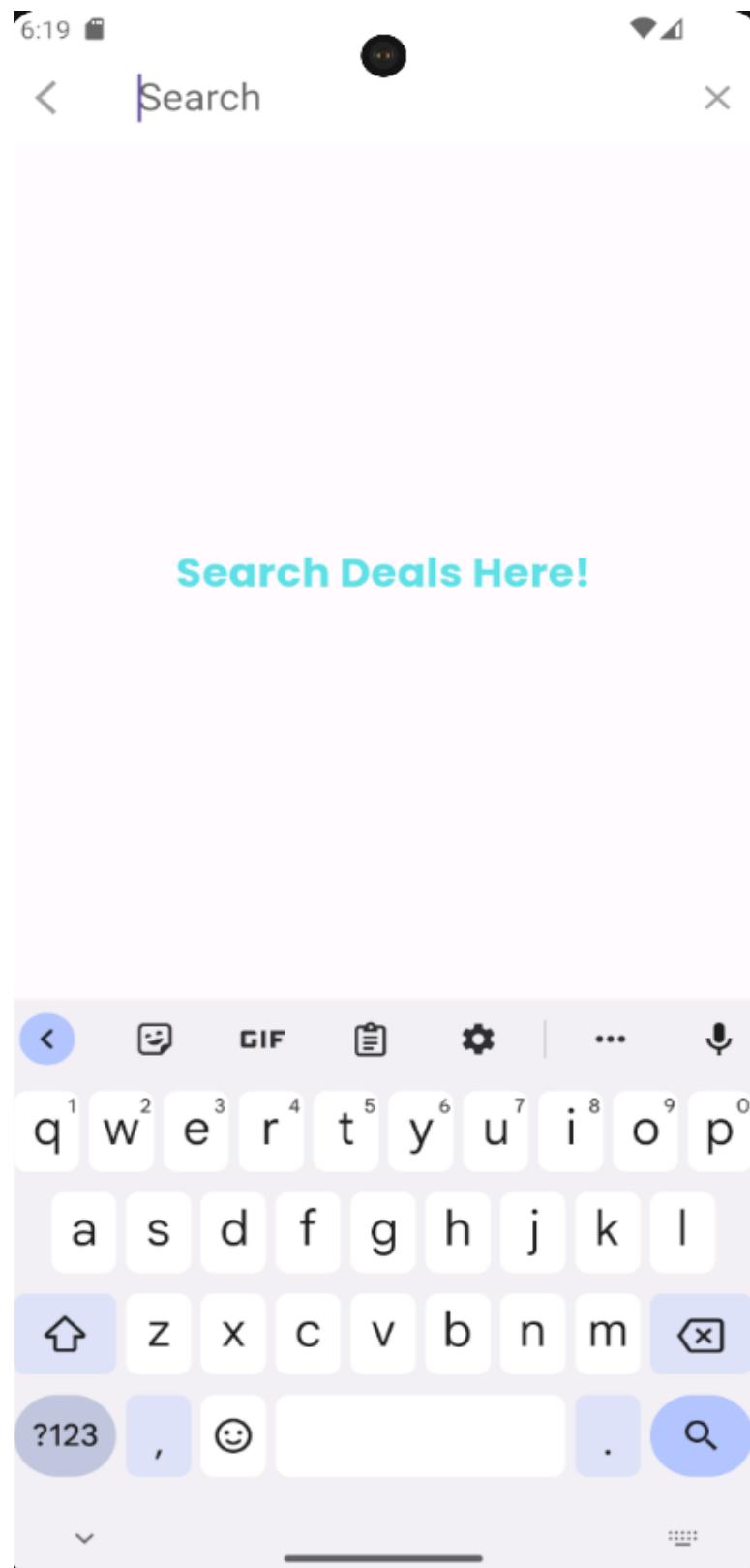


Figure 232: Search Testing -2.

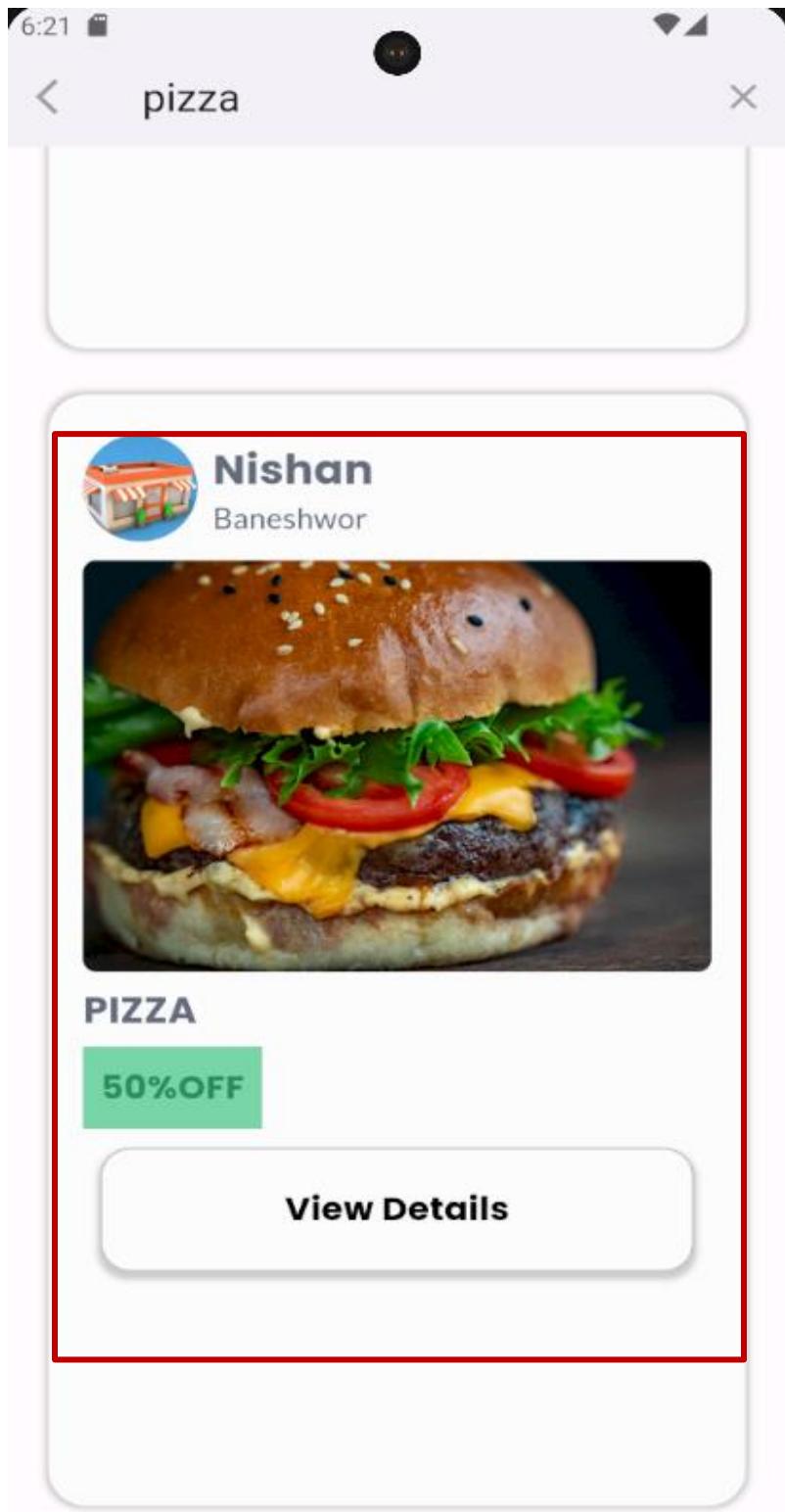


Figure 233: Search Testing -3.

**4.2.13 Search with invalid deal title**

<b>objective</b>		To browse invalid deals using search bar.
<b>Actions</b>	Enter a invalid deal title in search bar and search.	
<b>Input</b>	Search: "xxxxx"	
<b>Expected results</b>	No deal should be displayed.	
<b>Obtained Results</b>	No deal was displayed.	
<b>Conclusion</b>	Test Successful.	

*Table 32: Invalid search testing table.*

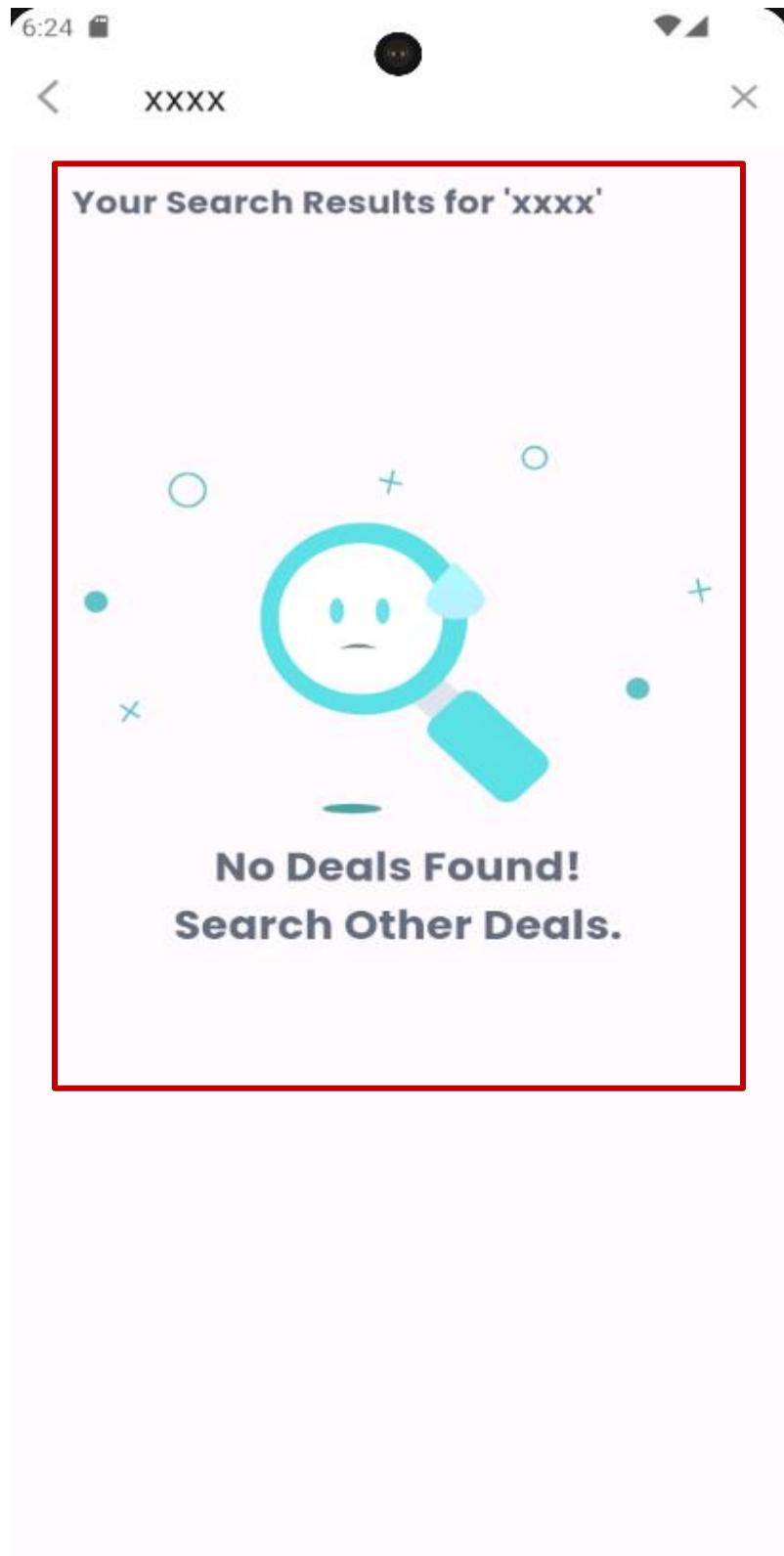


Figure 234: Invalid Deal Search Testing.

#### 4.2.14 Calculated coupon price

objective	To display calculated price for each coupon.
<b>Actions</b>	Navigate to cart and select a deal then press on buy coupon.
<b>Expected results</b>	Details of coupon with generated coupon price should be displayed. Also, different deals should have different coupon rate according to the discount they are providing.
<b>Obtained Results</b>	Details of coupon with generated coupon price was displayed.
<b>Conclusion</b>	Test Successful.

Table 33: Calculated coupon price testing table.

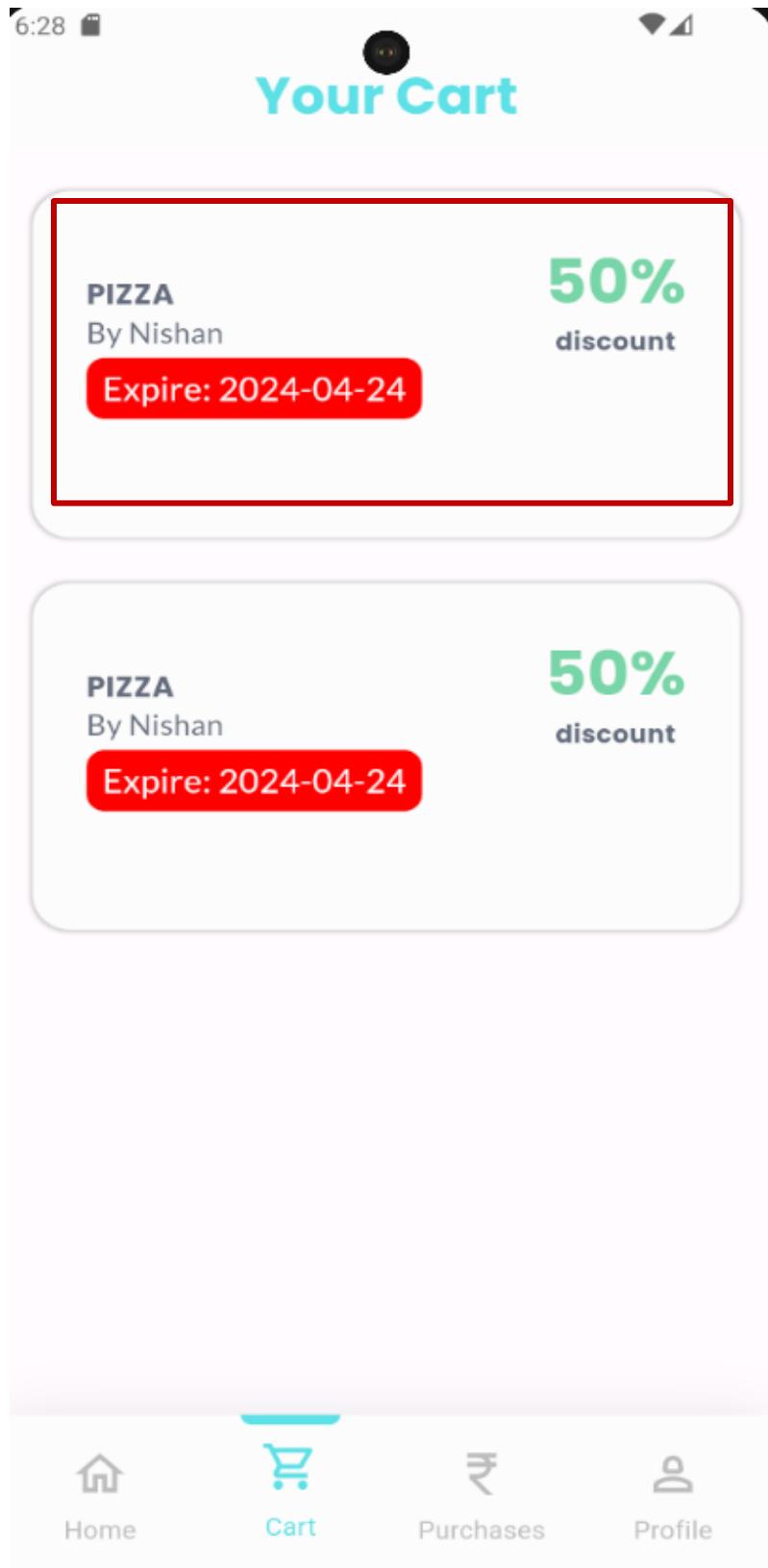


Figure 235: Calculated Coupon price Testing -1.

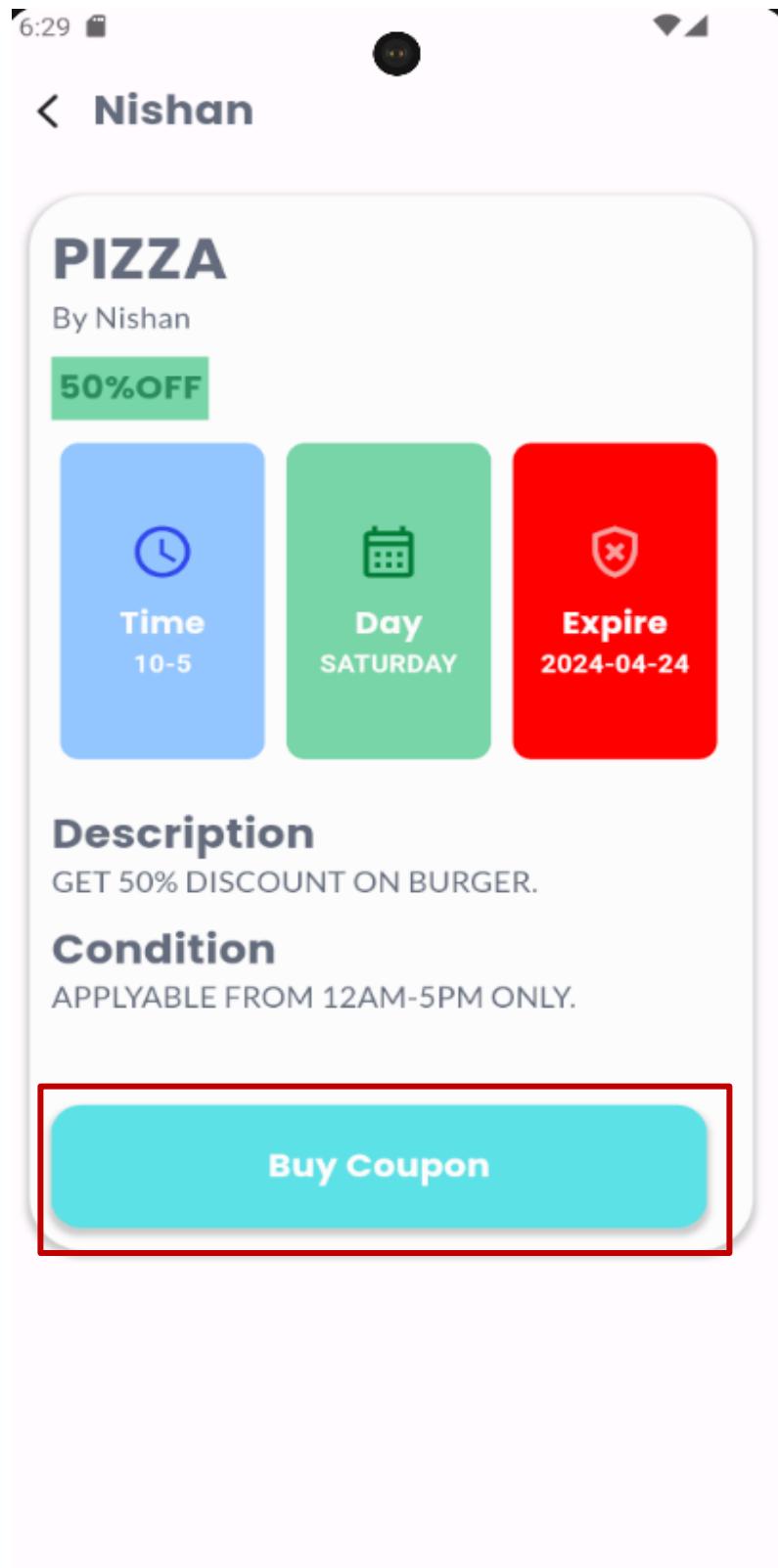


Figure 236: Calculated Coupon Price Testing -2.

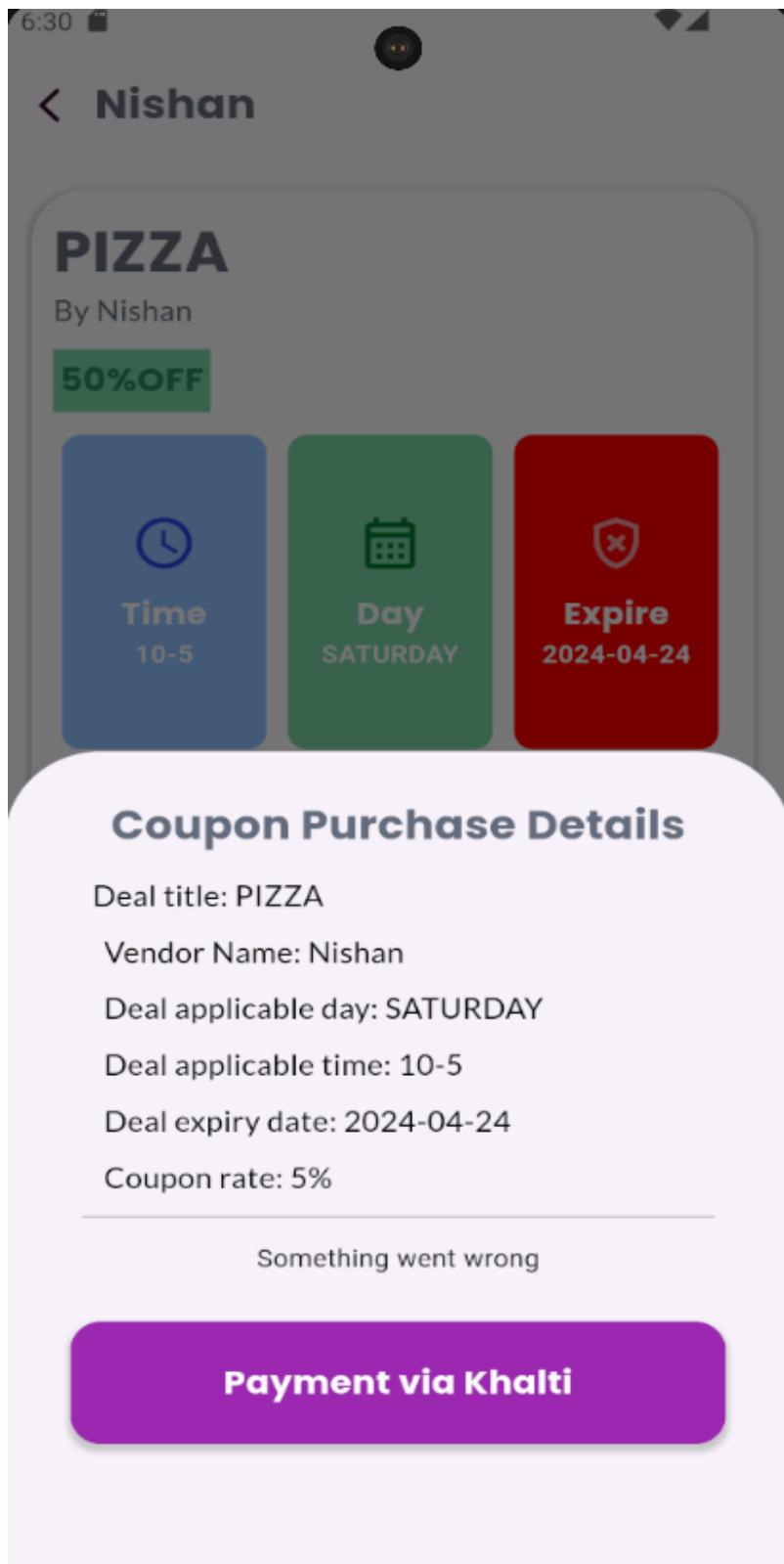


Figure 237: Calculated Coupon Price Testing -3.

#### 4.2.15 Send feedback

objective	To send feedback to KarKhana (Admin)
<b>Actions</b>	Navigate to feedback page and fill out the message. Then enter the send button
<b>Expected results</b>	Success message should be displayed, and feedback message should be stored in database.
<b>Obtained Results</b>	Success message was displayed, and feedback message was stored in database.
<b>Conclusion</b>	Test Successful.

Table 34: Send feedback testing table.

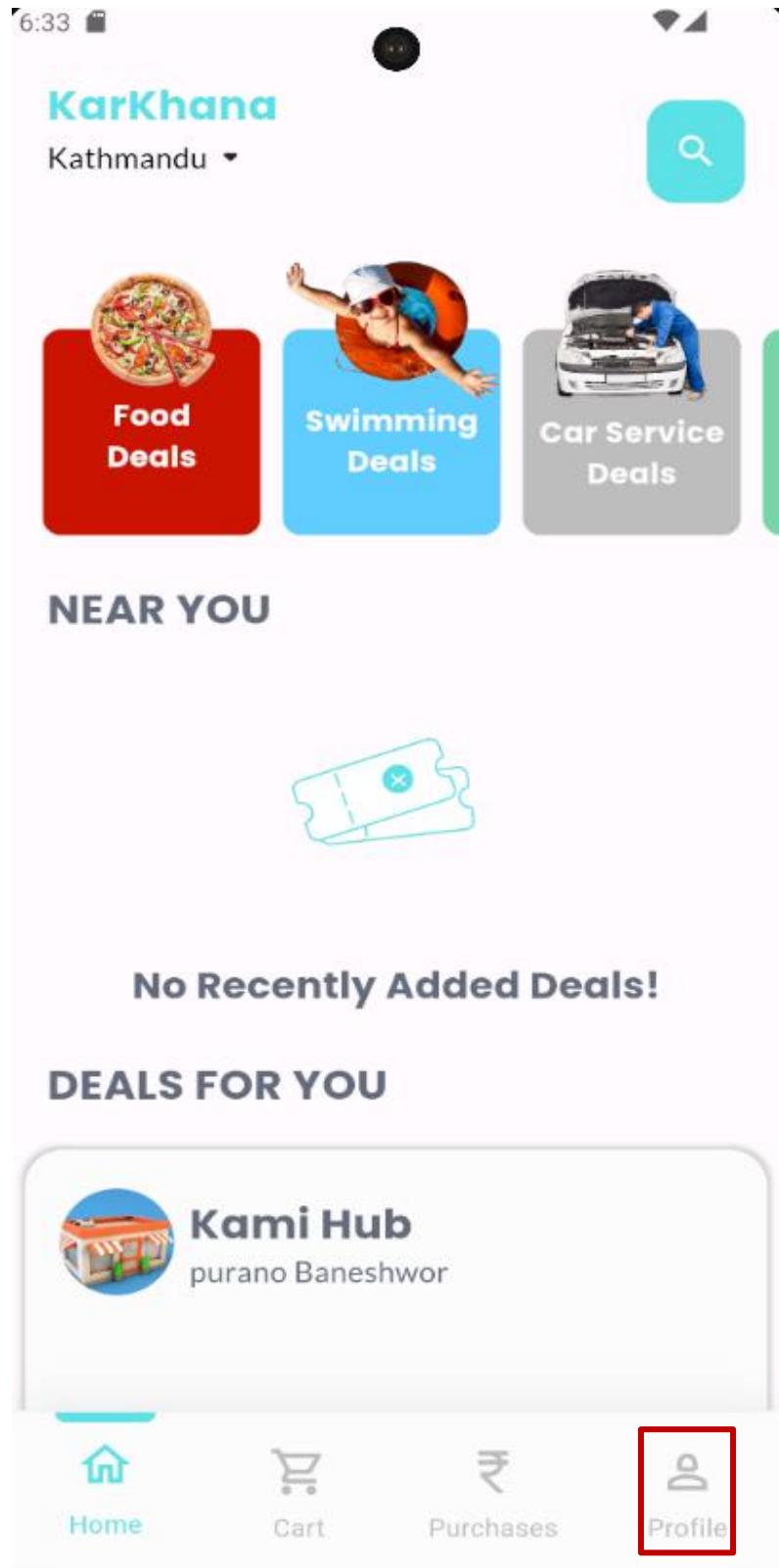


Figure 238: Send Feedback Testing -1.

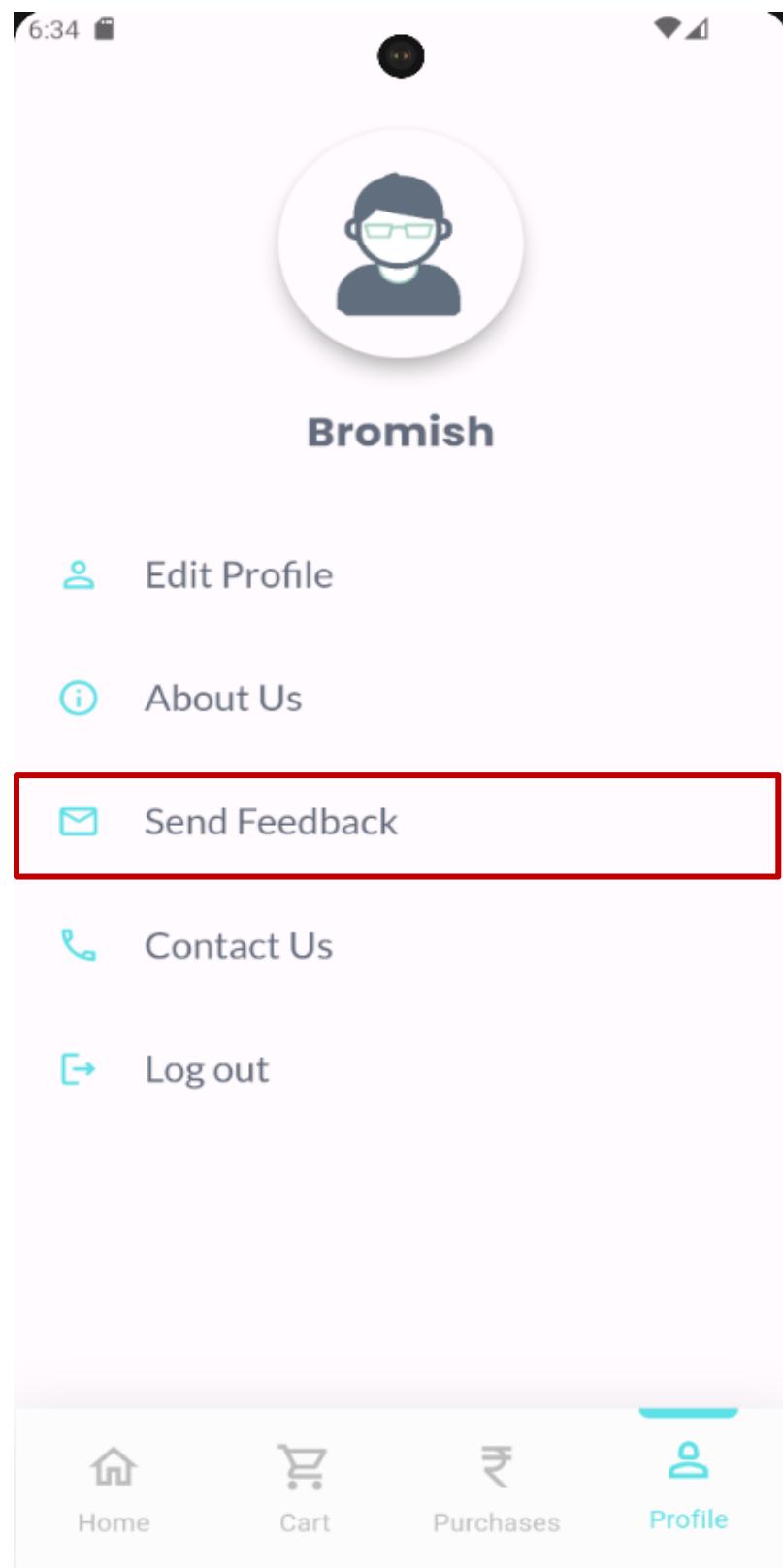


Figure 239: Send Feedback Testing -2.

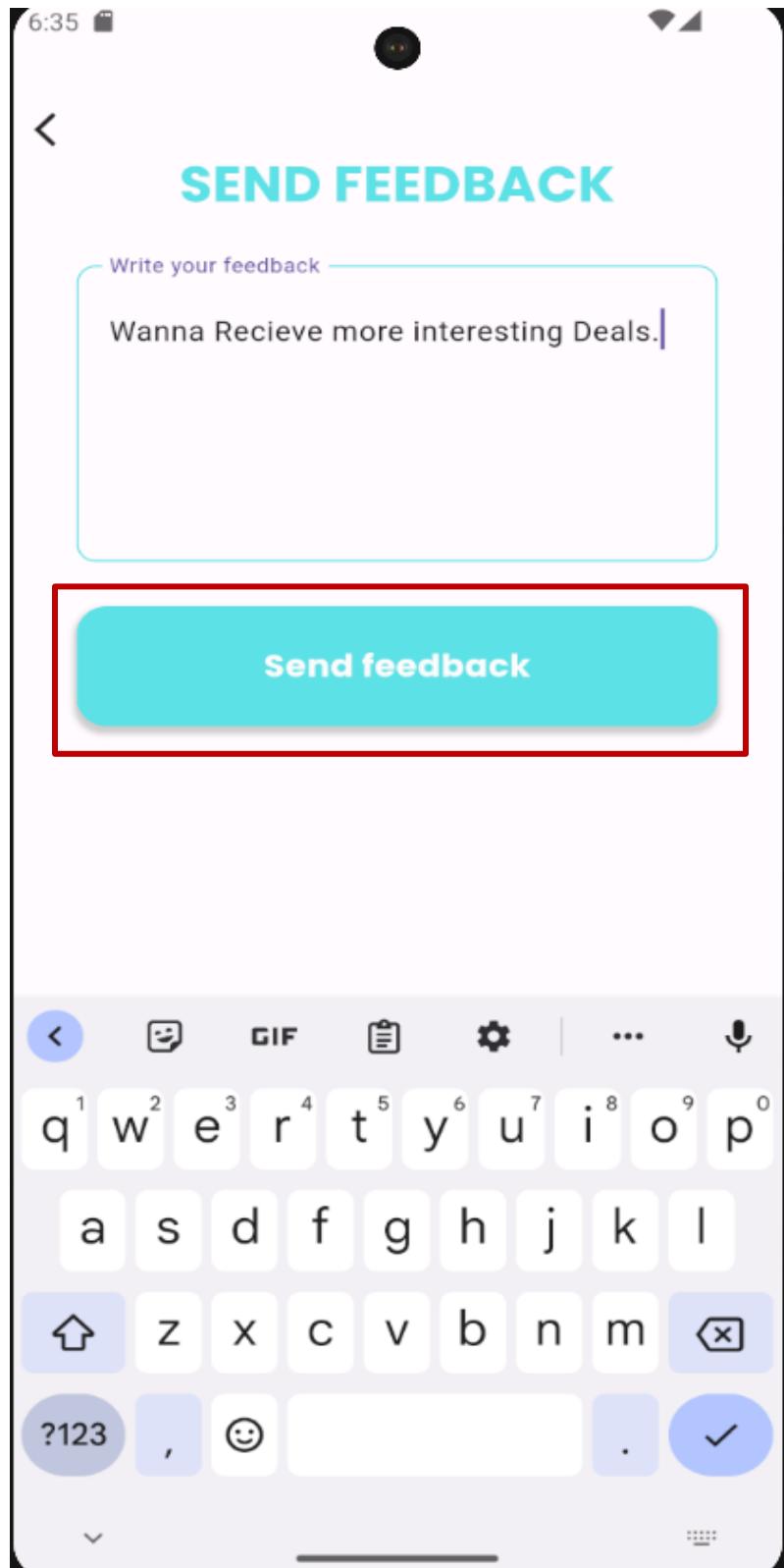


Figure 240: Send Feedback Testing -3.

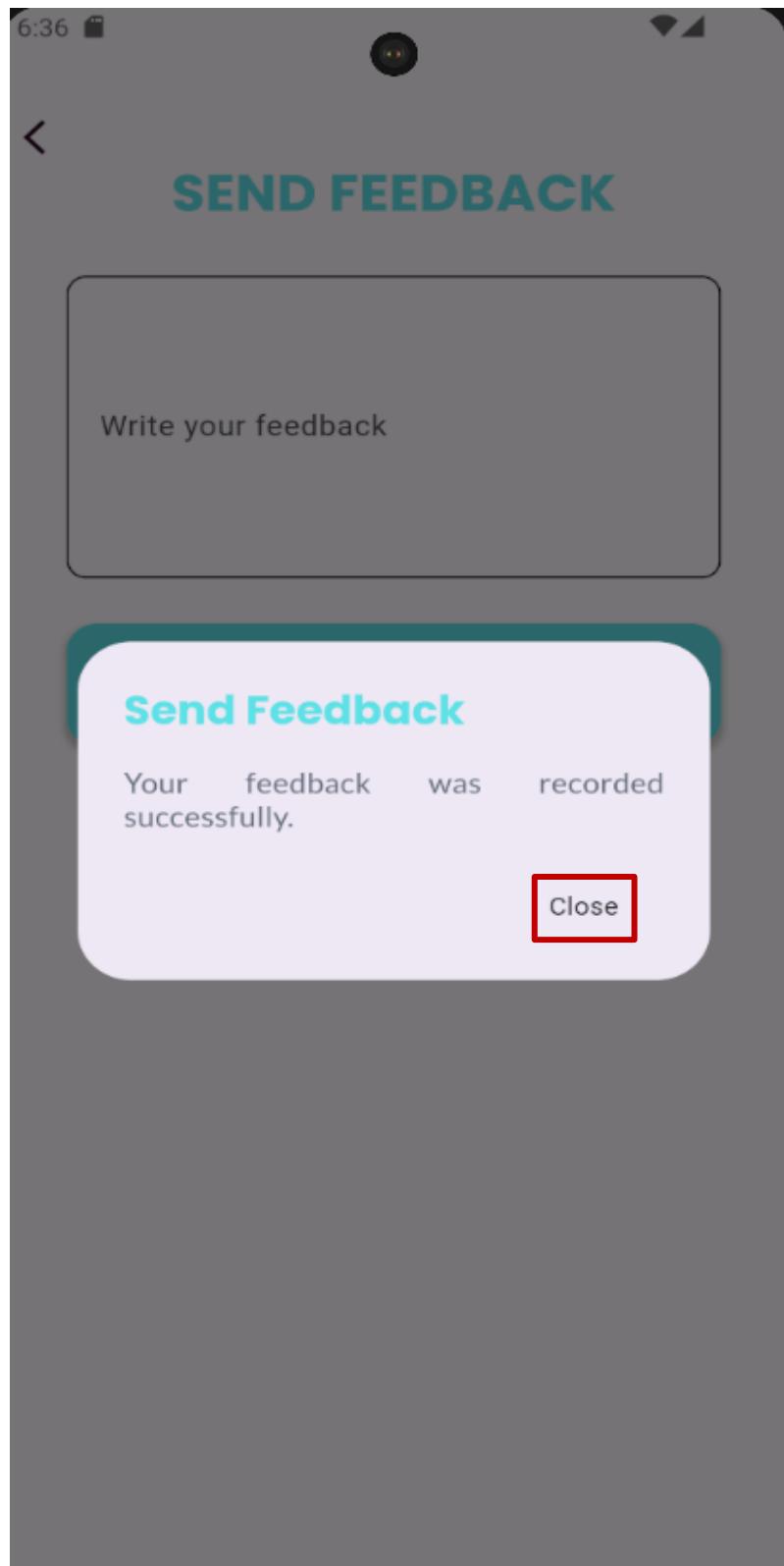


Figure 241: Send Feedback Testing -4.

The screenshot shows a 'Change feedback' interface for a customer named 'Bromish'. The form includes fields for Customer user (Bromish), Customer id (19), Customer Name (Bromish), Customer Email (np01cp4a210202@islingtoncollege.edu.np), and a Feedbacks field containing the message 'Wanna Recieve more interesting Deals.' Below the form are three icons: a pencil, a plus sign, and an eye.

Figure 242: Feedback Successfully Updated in Database.

#### 4.2.16 Send feedback field validation.

objective	To get field validation error in feedback page.
<b>Actions</b>	Press send button with empty message field.
<b>Input</b>	Message: “ ”
<b>Expected results</b>	Field validation error should be displayed.
<b>Obtained Results</b>	Field validation error was displayed.
<b>Conclusion</b>	Test Successful.

Table 35: Send feedback field validation testing table.

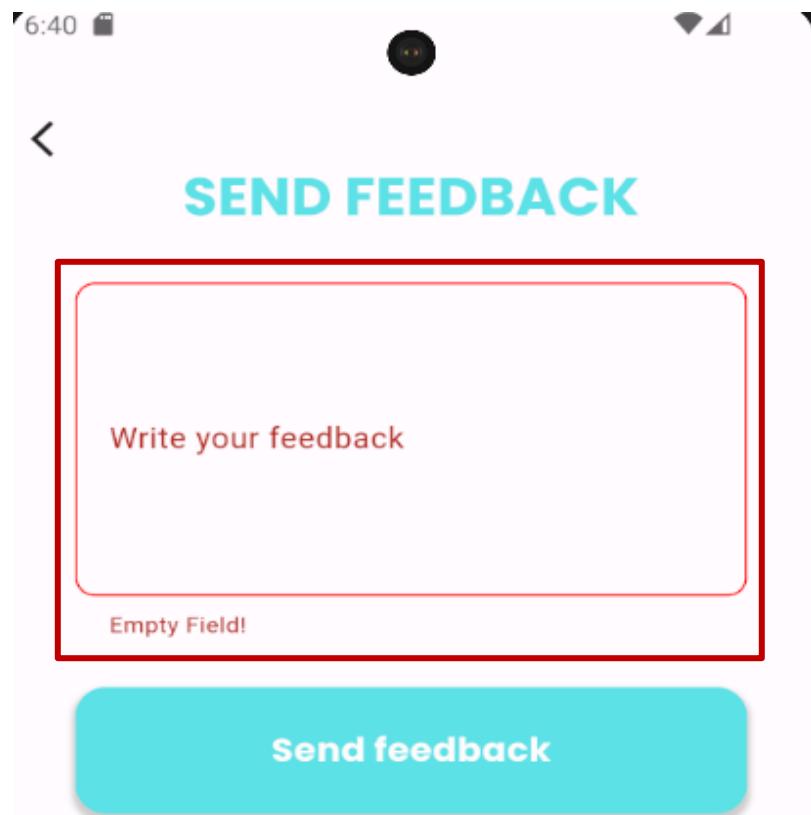


Figure 243: Send Feedback Validation Testing.

**4.2.17 Edit profile**

<b>objective</b>		To edit name of the user through edit profile page.
<b>Actions</b>		Enter new name and press update profile button.
<b>Input</b>		First Name: "Flaming", Last Name: "Eagle"
<b>Expected results</b>		Success message should be displayed.
<b>Obtained Results</b>		Success message was displayed.
<b>Conclusion</b>		Test Successful.

*Table 36: Edit profile testing table.*

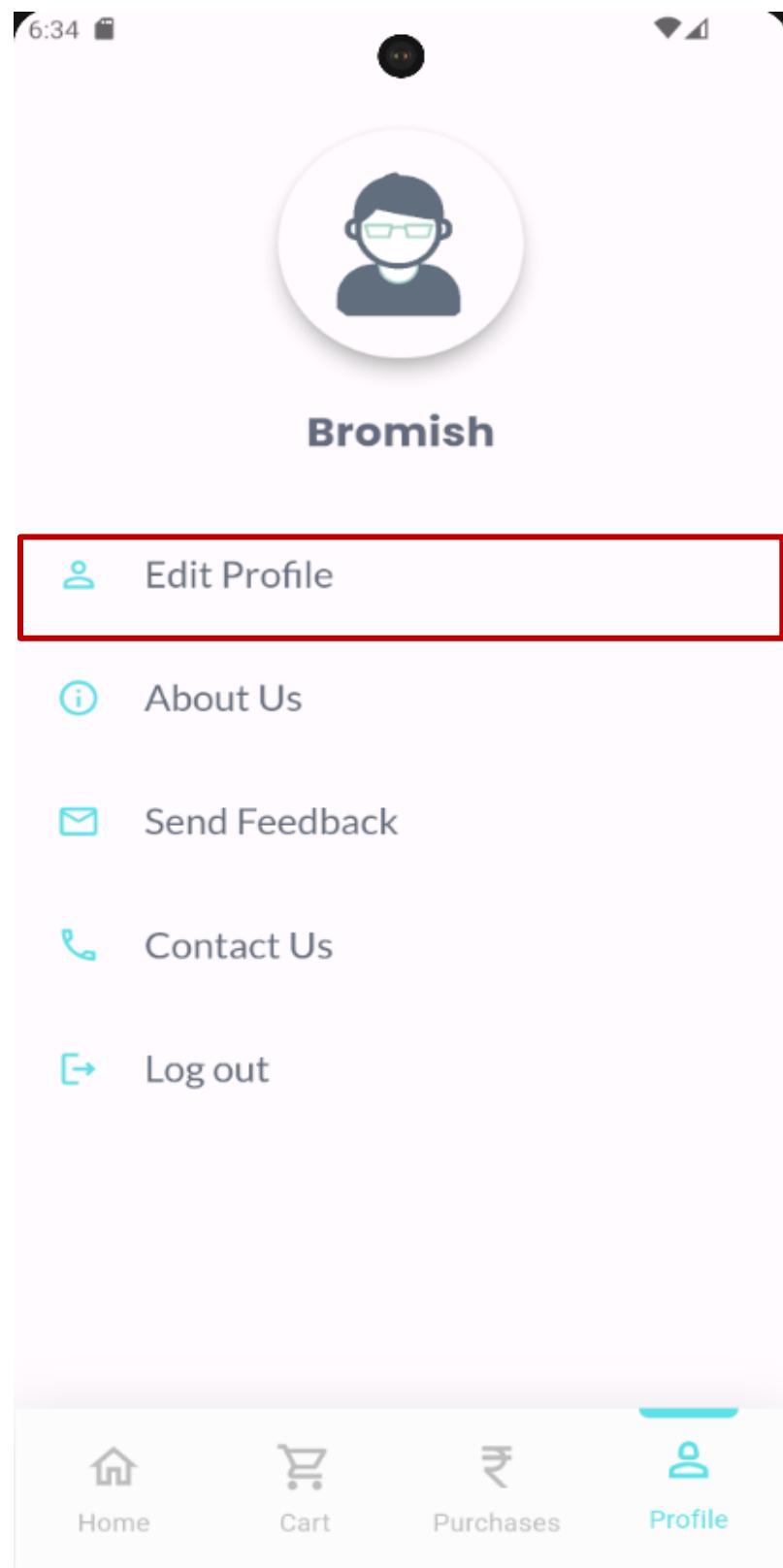


Figure 244: Testing Profile Testing -1.



Figure 245: Edit Profile Testing -2.

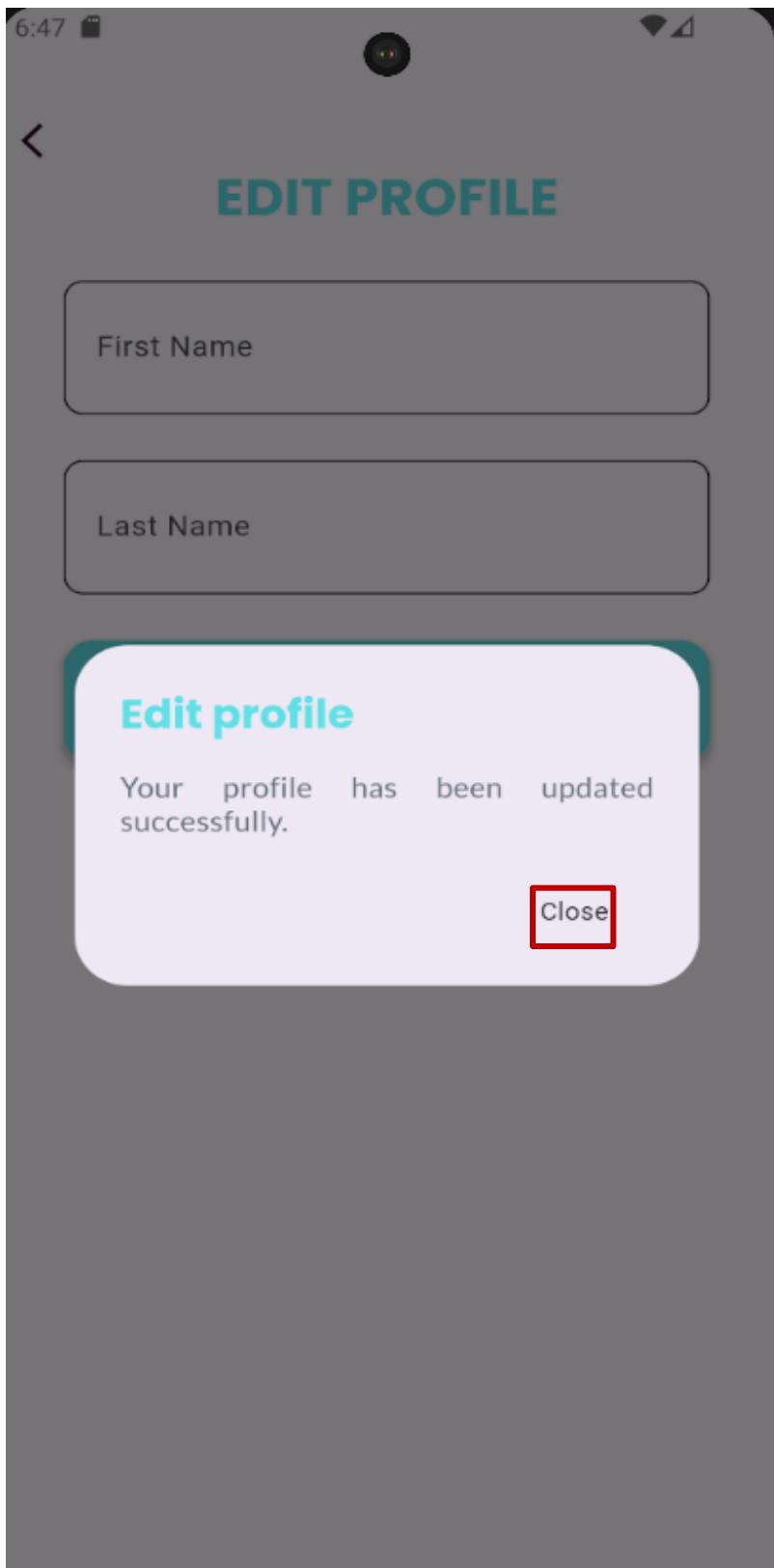


Figure 246: Edit Profile Testing -3.

The screenshot shows a user profile edit form with the following fields:

- First name:** Tushar
- Last name:** Adhikari
- Email address:** np01cp4a210202@islingtoncollege.edu.np
- Staff status:** Unselected (checkbox)
- Designates whether the user can log into this admin site.**
- Active:** Selected (checkbox)
- Designates whether this user should be treated as active. Unselect this instead of deleting accounts.**

A red box highlights the first two fields (First name and Last name).

Figure 247: Profile Successfully Updated in Database.

#### 4.2.18 Edit profile field validation

objective	To get field validation error in edit profile page.
<b>Actions</b>	Press update button with empty first and last field.
<b>Input</b>	First Name: " ", Last Name: " "
<b>Expected results</b>	Field validation error should be displayed.
<b>Obtained Results</b>	Field validation error was displayed.
<b>Conclusion</b>	Test Successful.

Table 37: Edit profile field validation testing table.

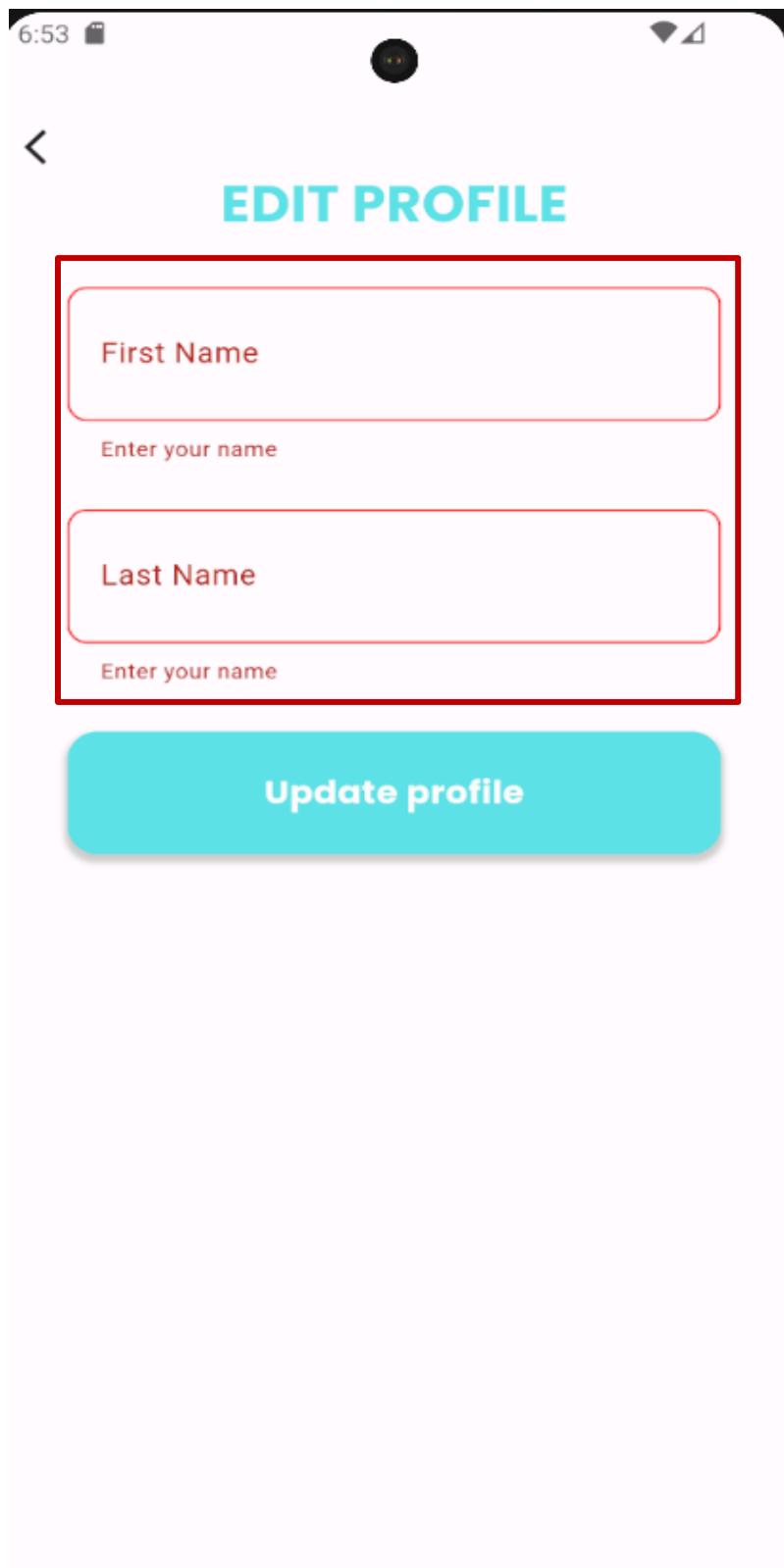


Figure 248: Edit Profile Validation Testing.

#### 4.2.19 Purchase coupon

<b>objective</b>	To purchase a coupon successfully and get a success message. Also display the purchased coupon in purchases section.
<b>Actions</b>	Navigate to cart and select a deal to buy. Press Buy button. Again further, press pay via Khalti button.
<b>Expected results</b>	Khalti interface should be displayed. Khalti success and payment success should be display.
<b>Obtained Results</b>	Khalti interface was displayed. Khalti success and payment success was display.
<b>Conclusion</b>	Test Successful.

Table 38: Purchase coupon testing table.

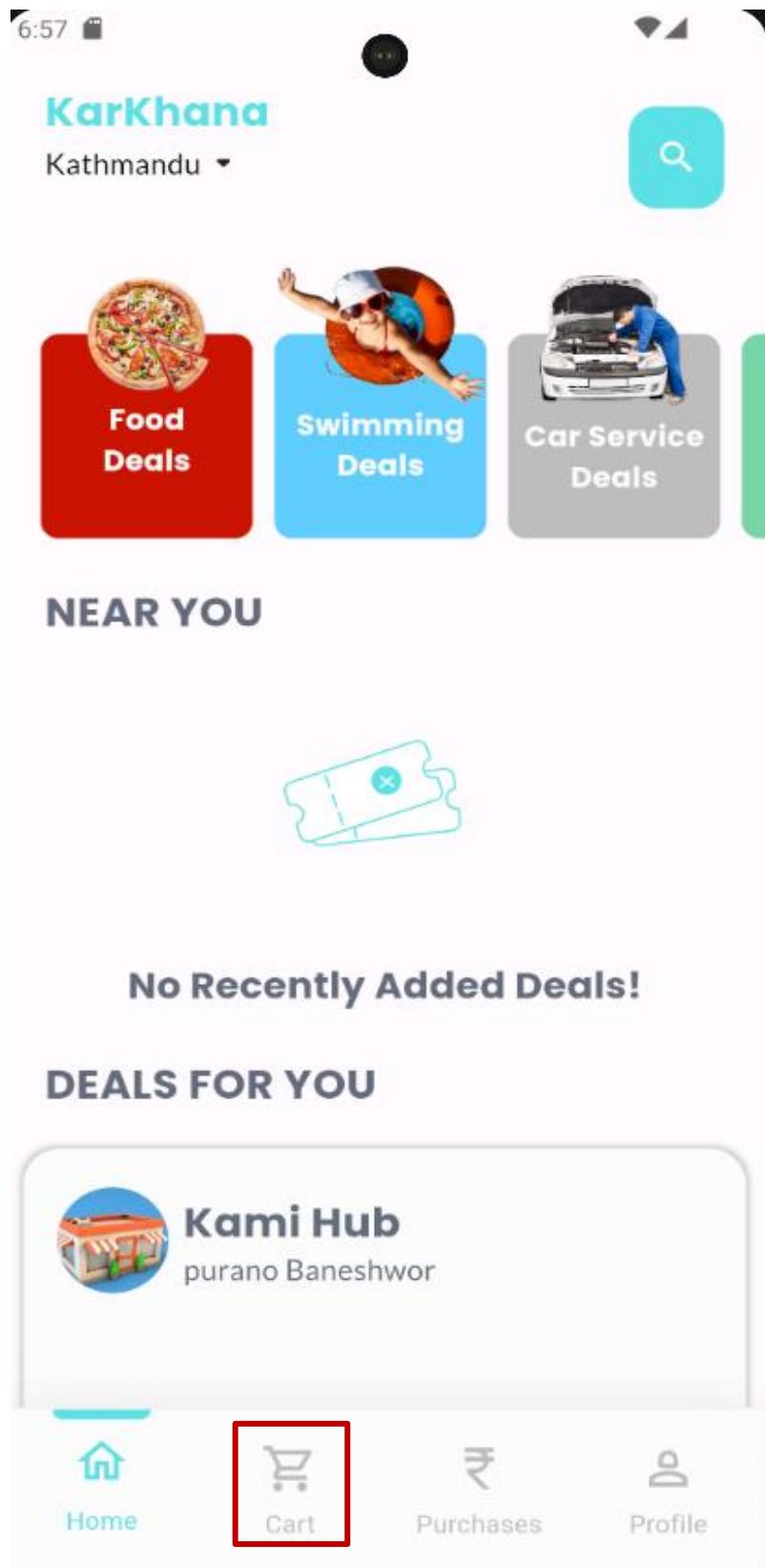


Figure 249: Coupon Purchase Testing -1.

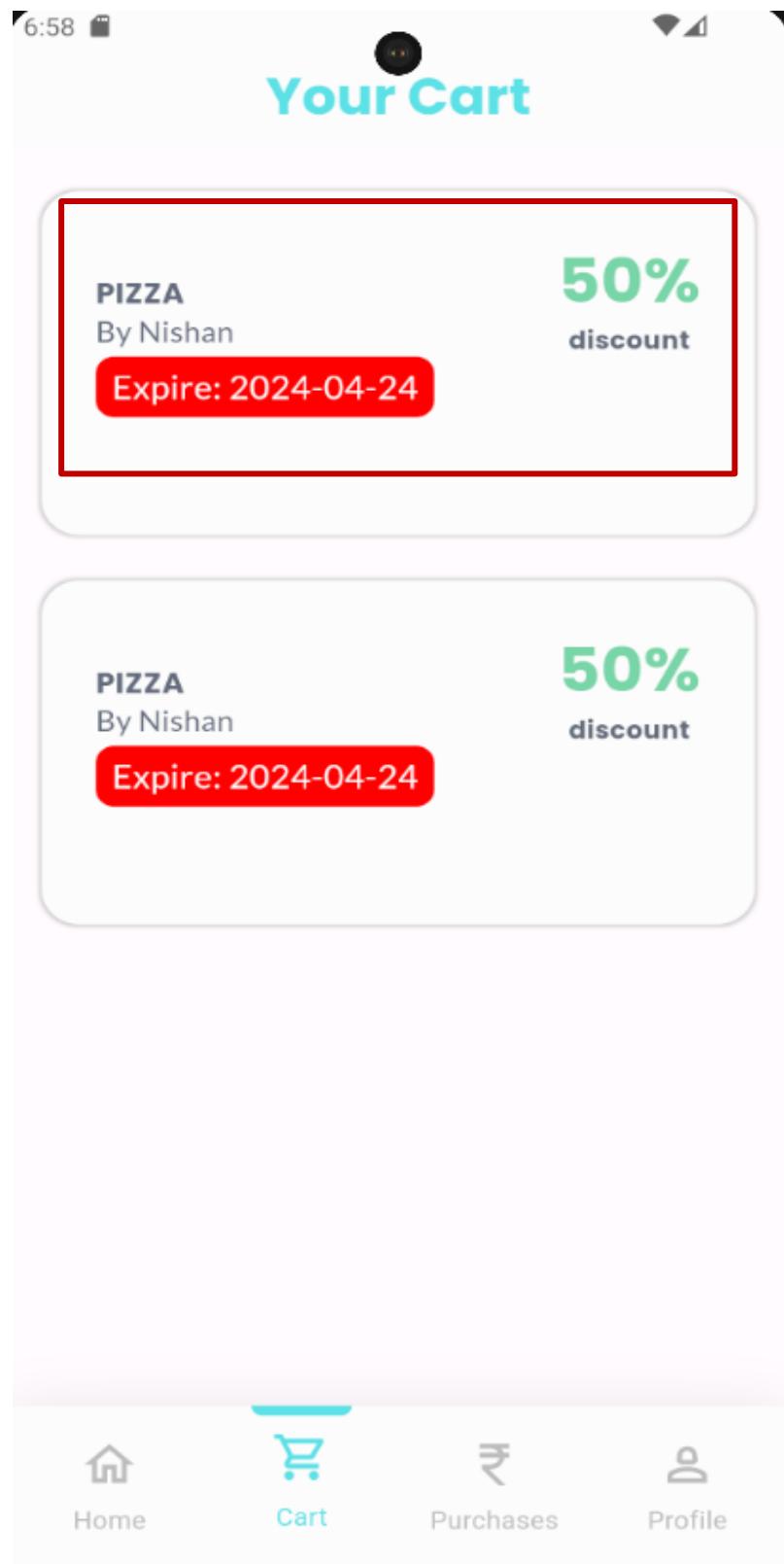


Figure 250: Coupon Purchase Testing -2.

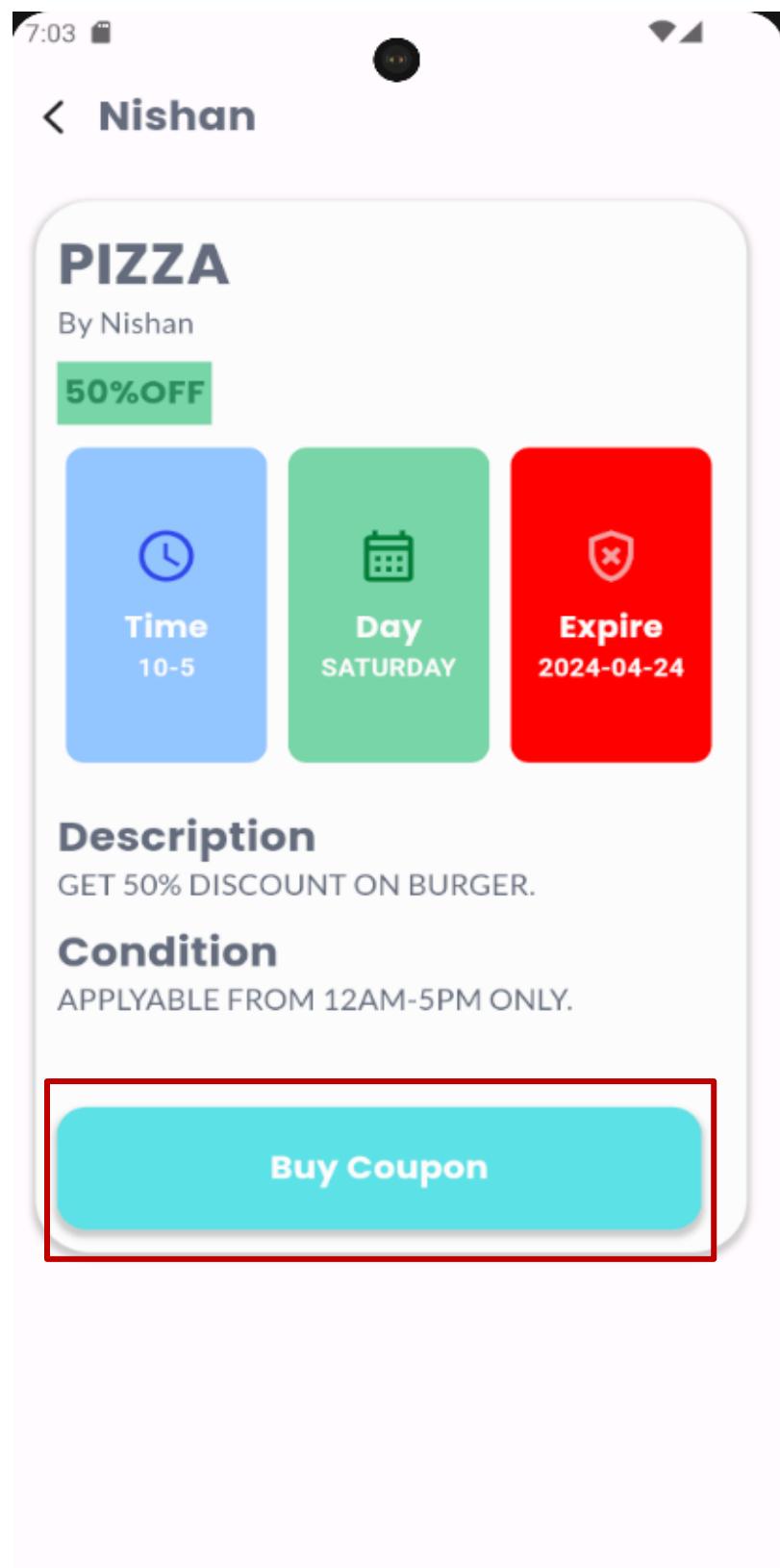


Figure 251: Coupon Purchase Testing -3.

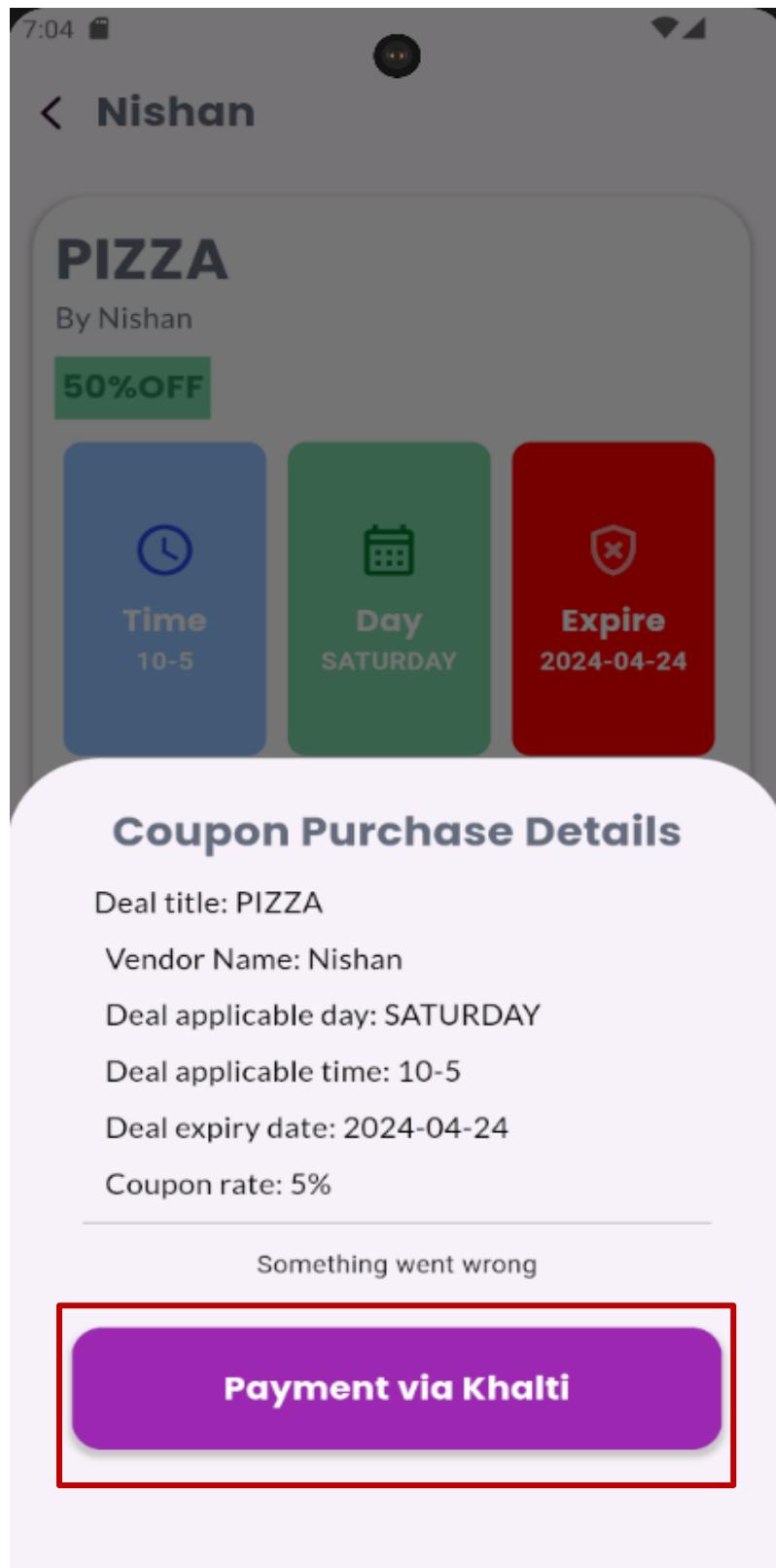


Figure 252: Coupon Purchase Testing -4.



Figure 253: Coupon Purchase Testing -5.

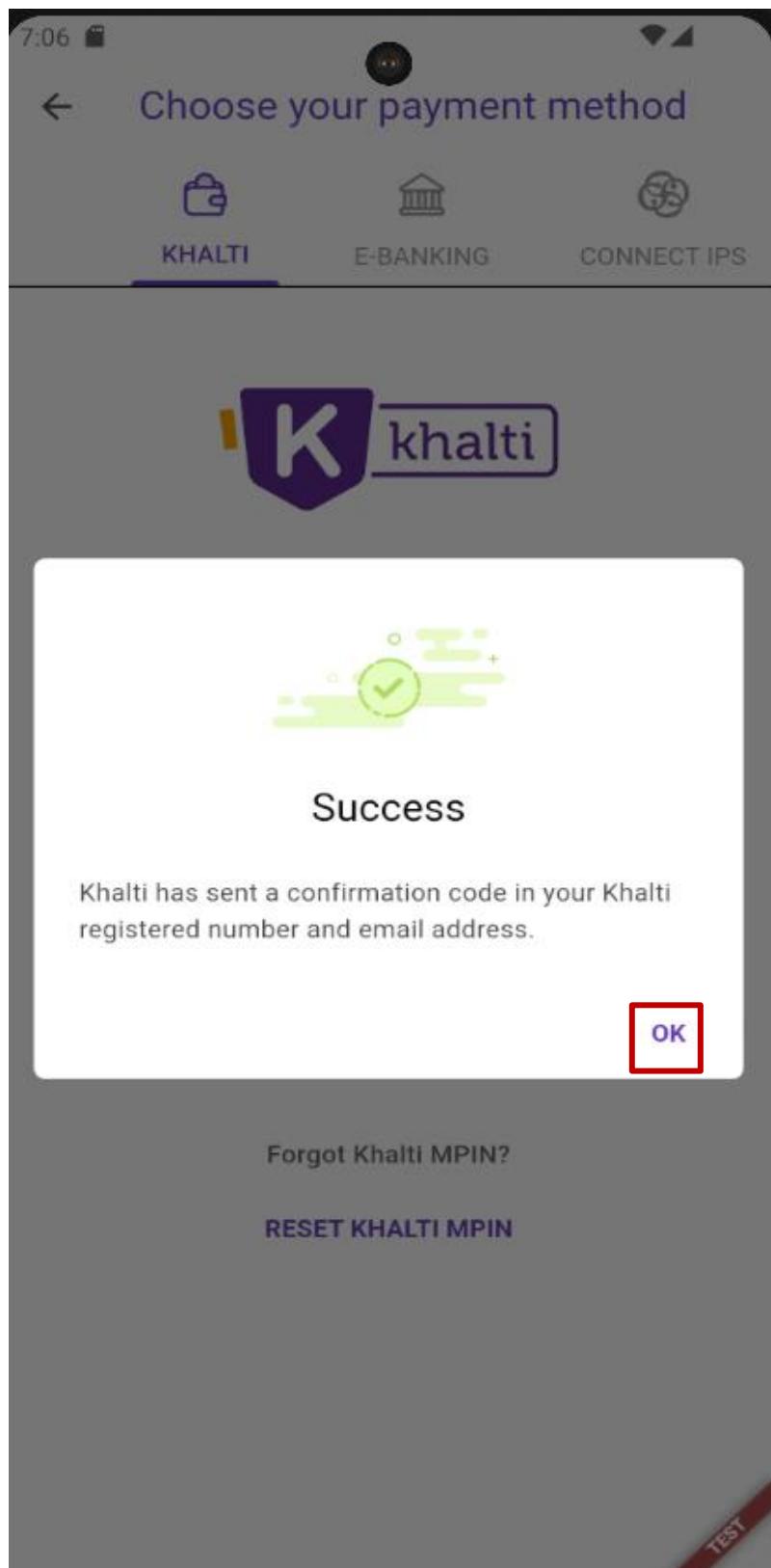


Figure 254: Coupon Purchase Testing -6.

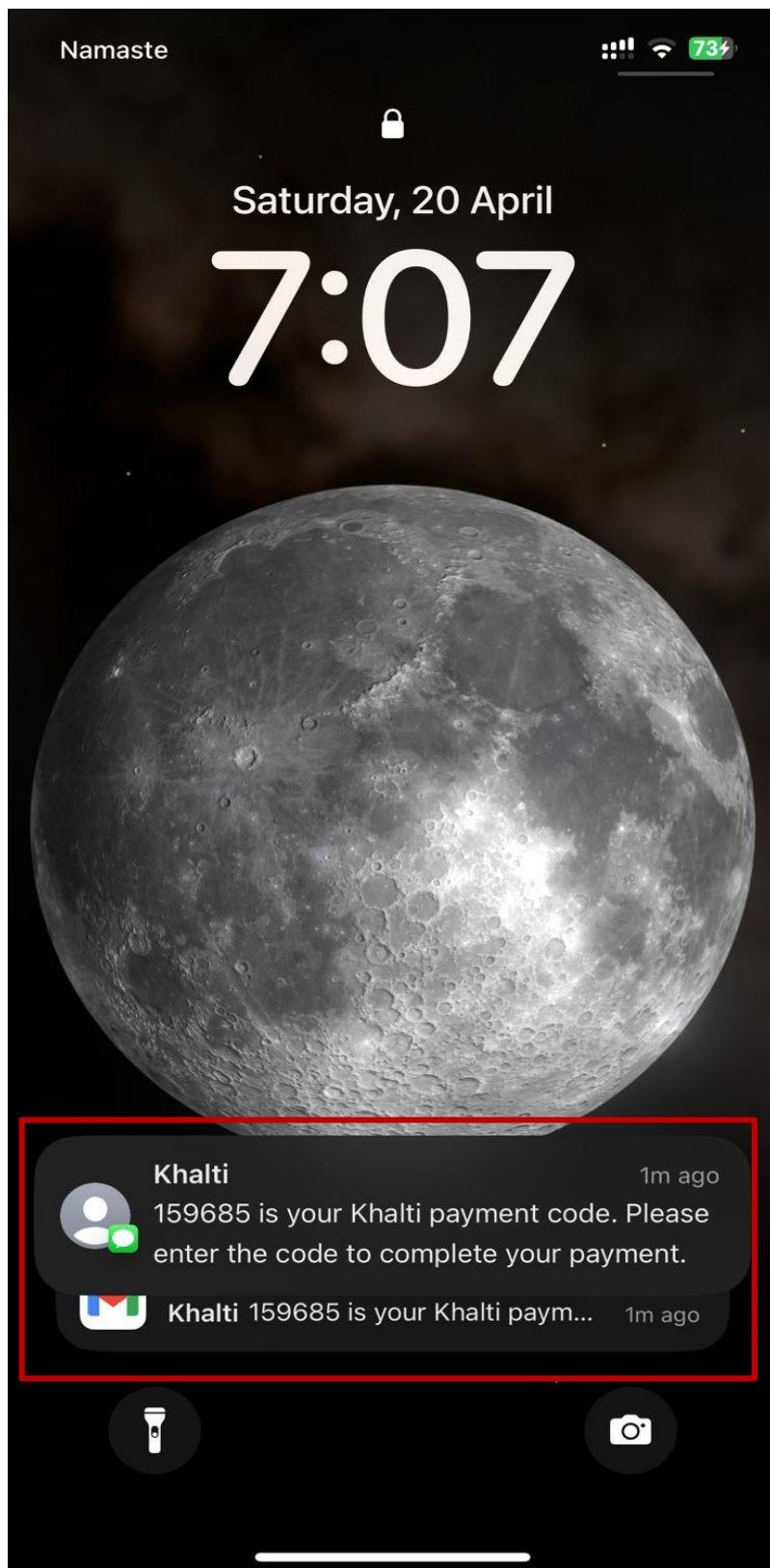


Figure 255: Code Received in SMS and Gmail.

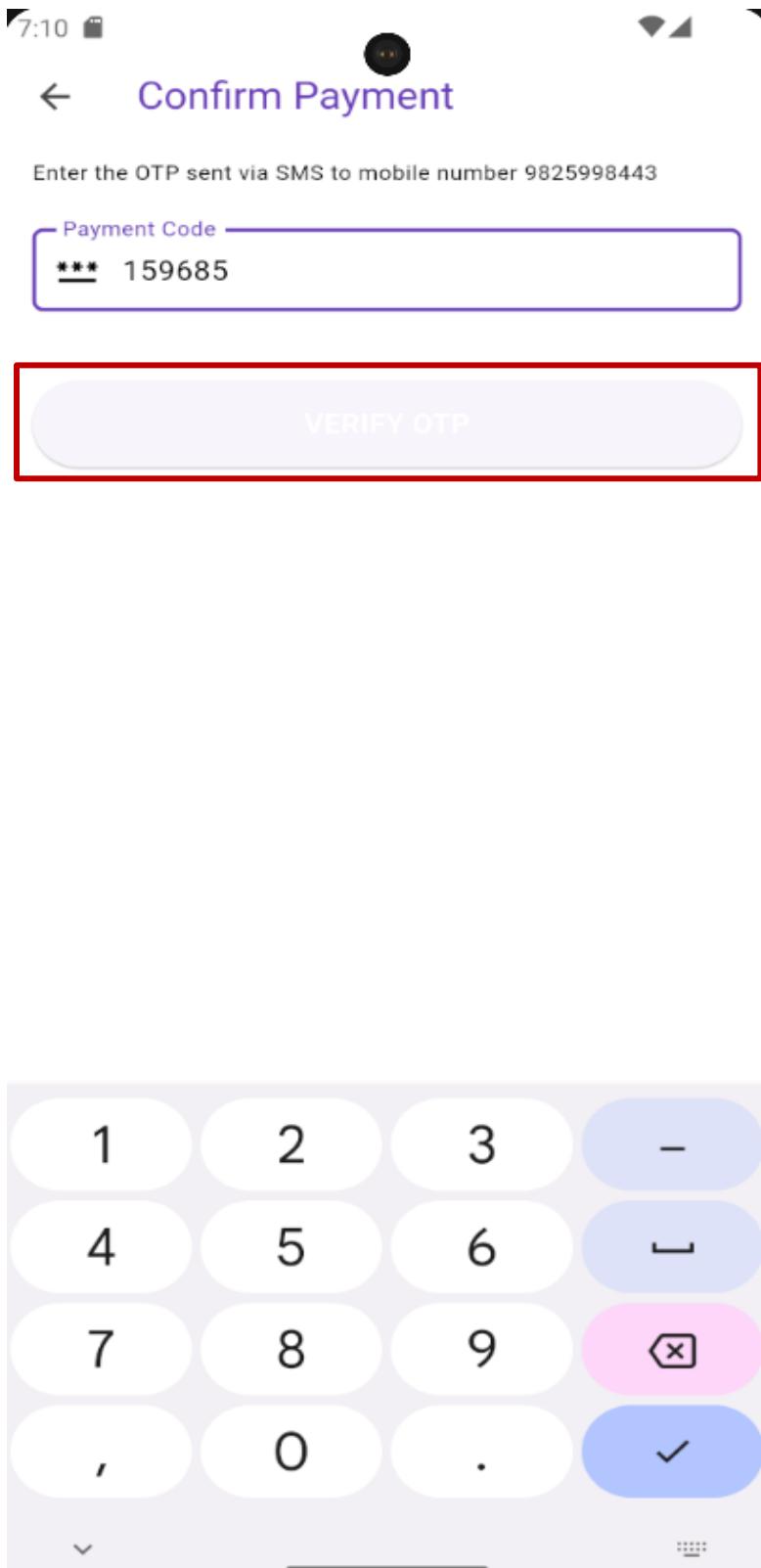


Figure 256: Coupon Purchase Testing -7.

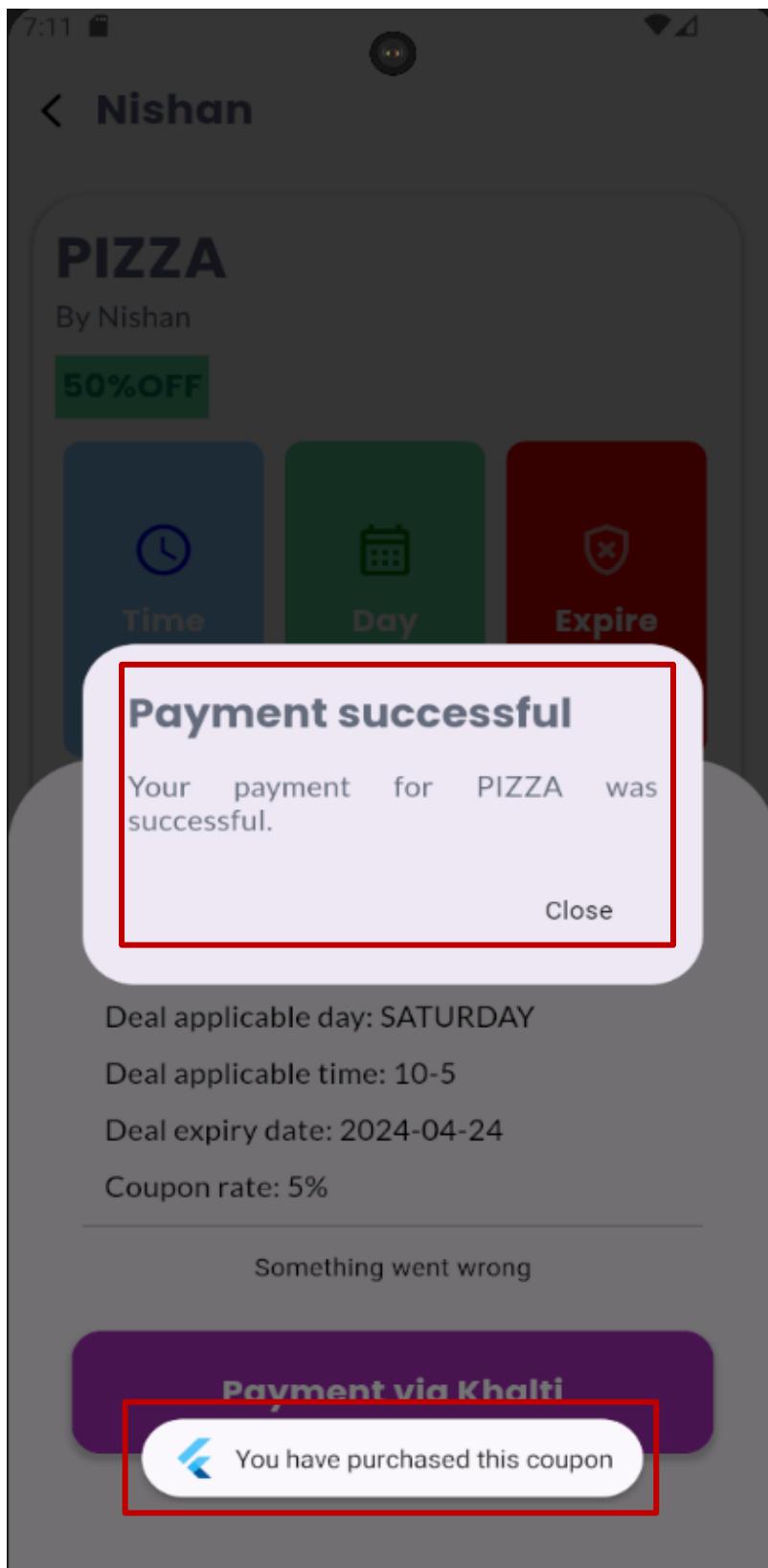


Figure 257: Coupon Purchased Successfully.

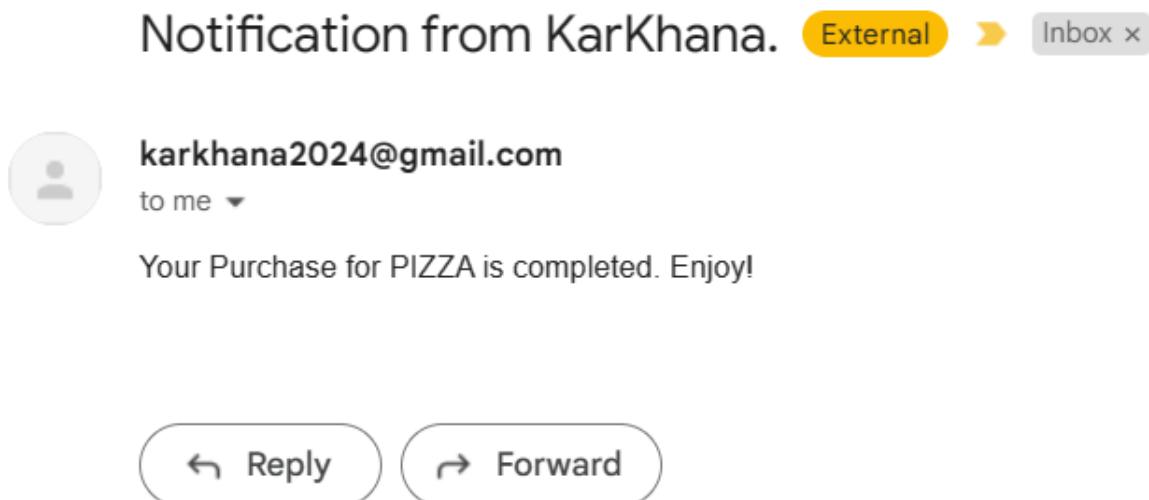


Figure 258: Successful Coupon Purchase Notification in Gmail.

#### 4.2.20 Purchase coupon failed.

objective	To display error while purchase of coupon fails.
<b>Actions</b>	Select deal and press buy coupon button then again press pay via Khalti button. Then without moving any further press back button.
<b>Expected results</b>	Payment error message should be displayed.
<b>Obtained Results</b>	Payment error message was displayed.
<b>Conclusion</b>	Test Successful.

Table 39: Purchase coupon failed testing table.

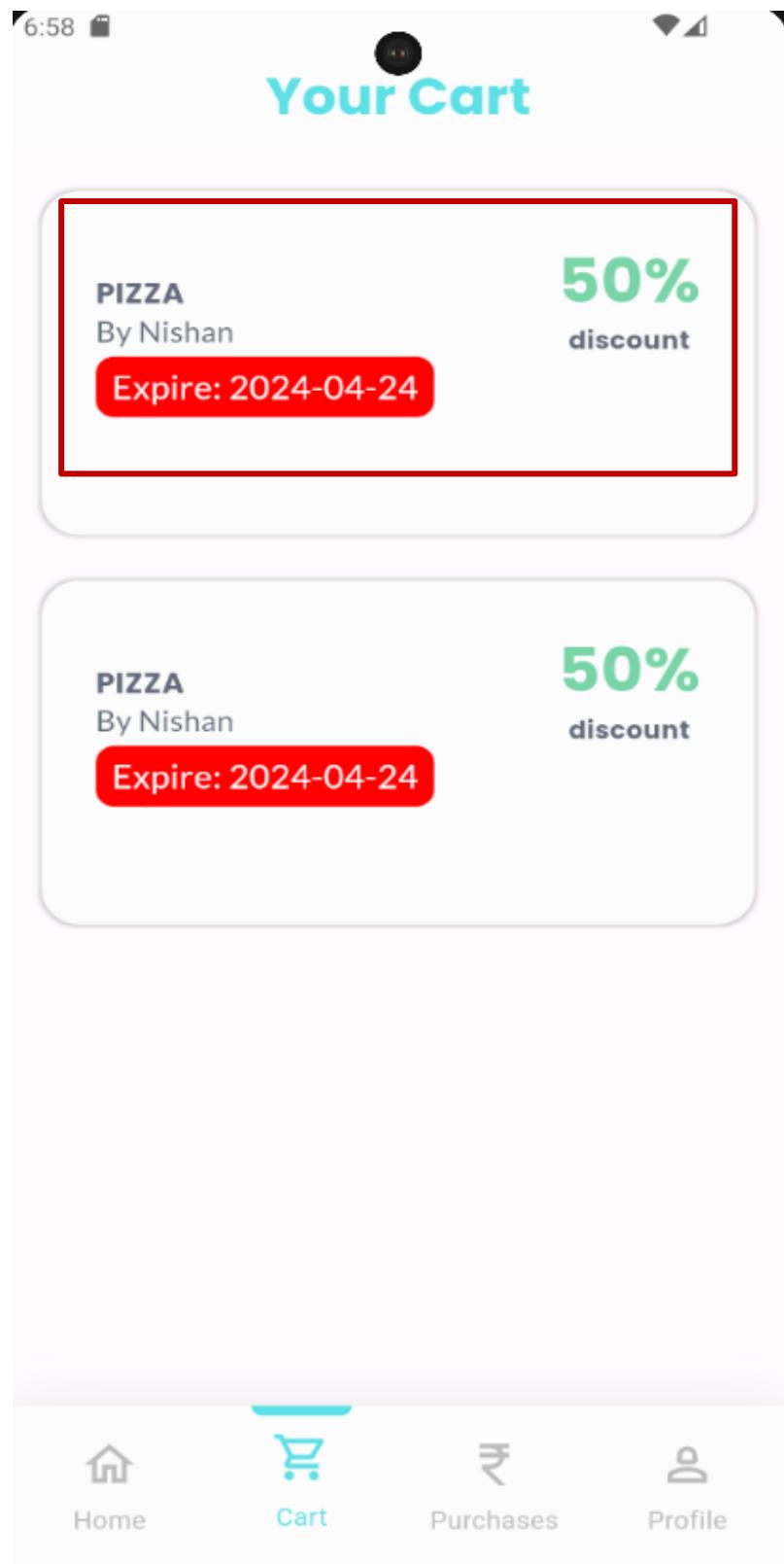


Figure 259: Coupon Purchase Failed Testing -1.

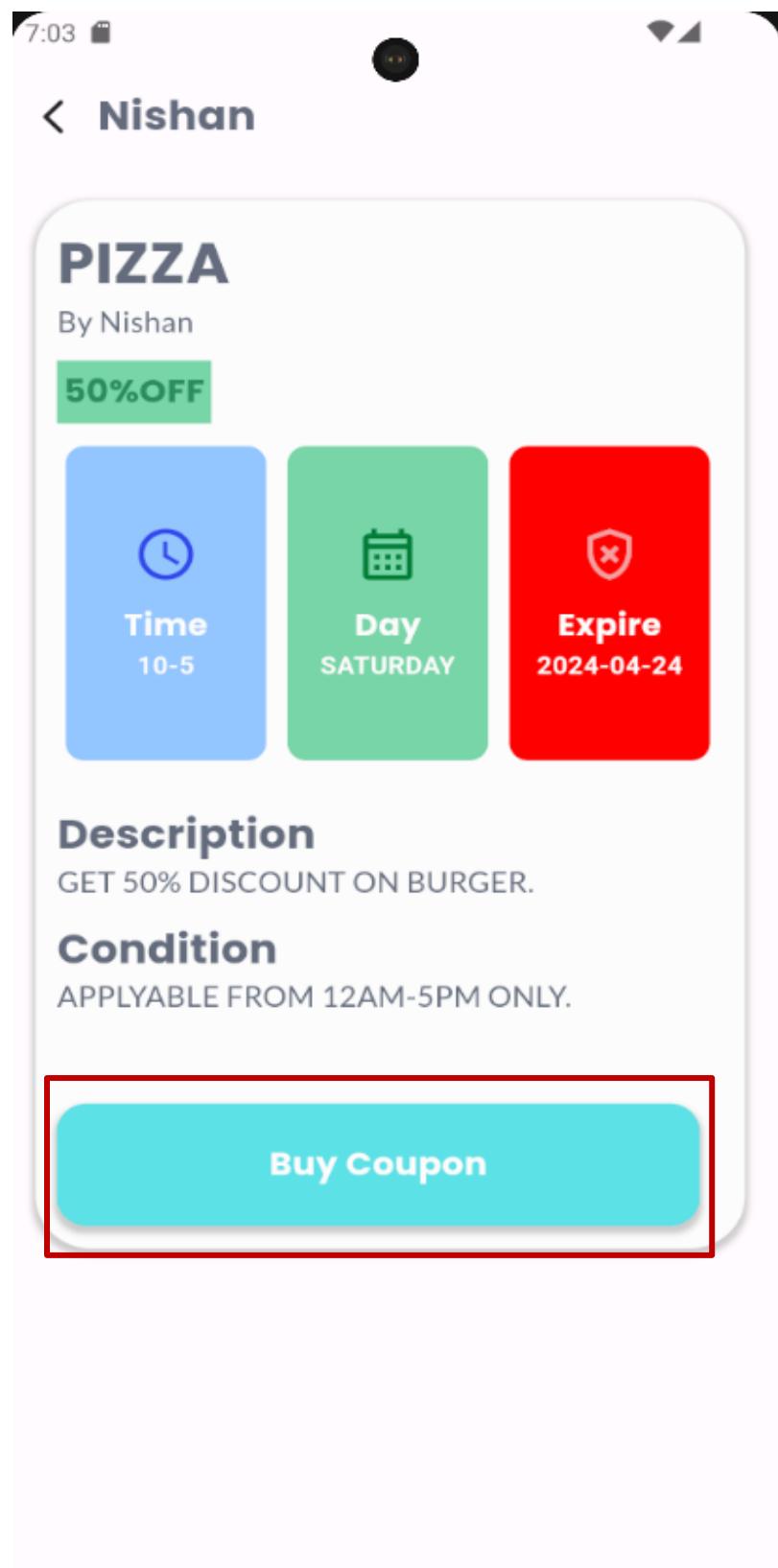


Figure 260: Coupon Purchase Failed Testing -2.

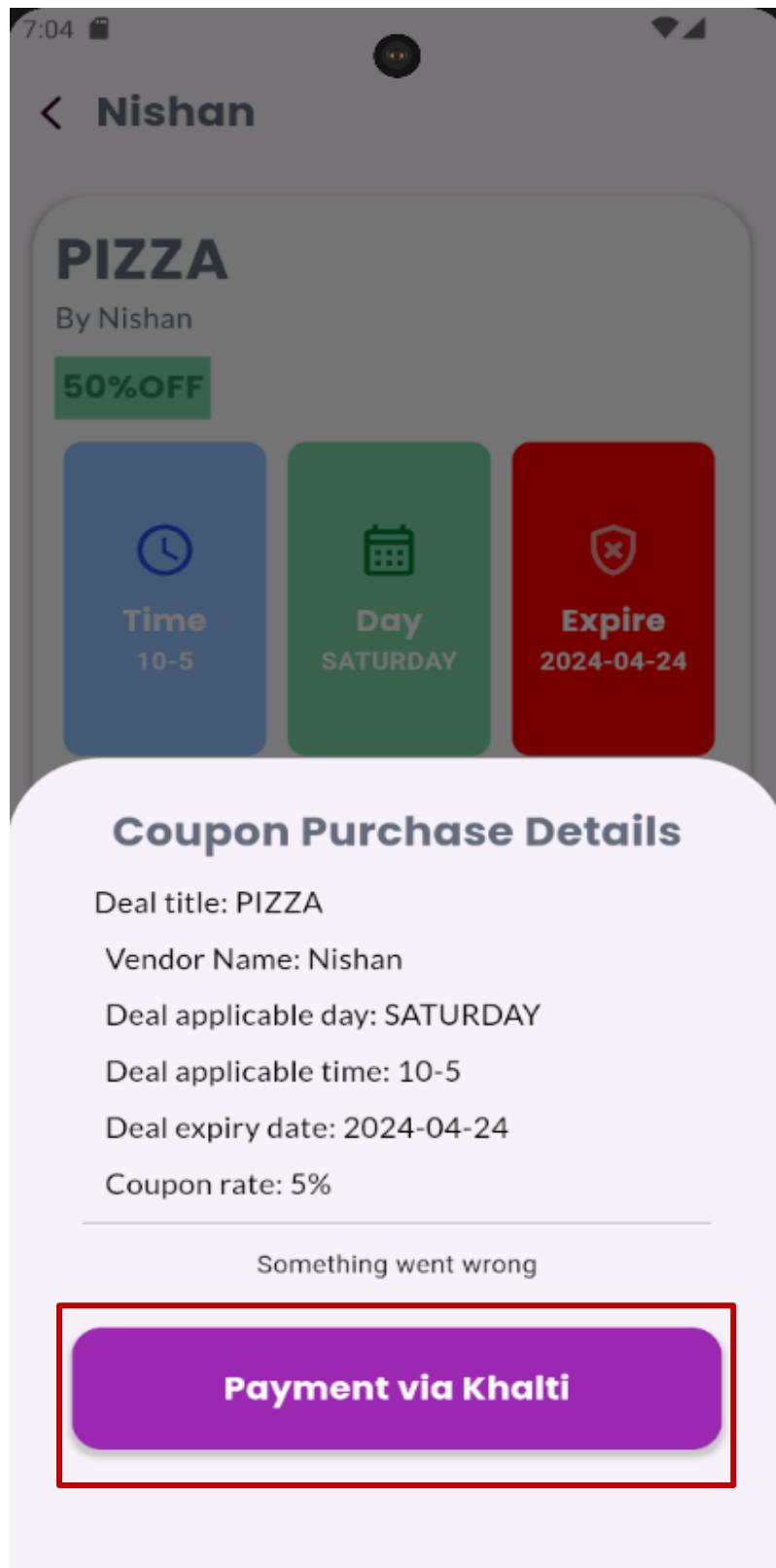


Figure 261: Coupon Purchase Failed Testing -3.



Figure 262: Coupon Purchased Failed Testing -4.

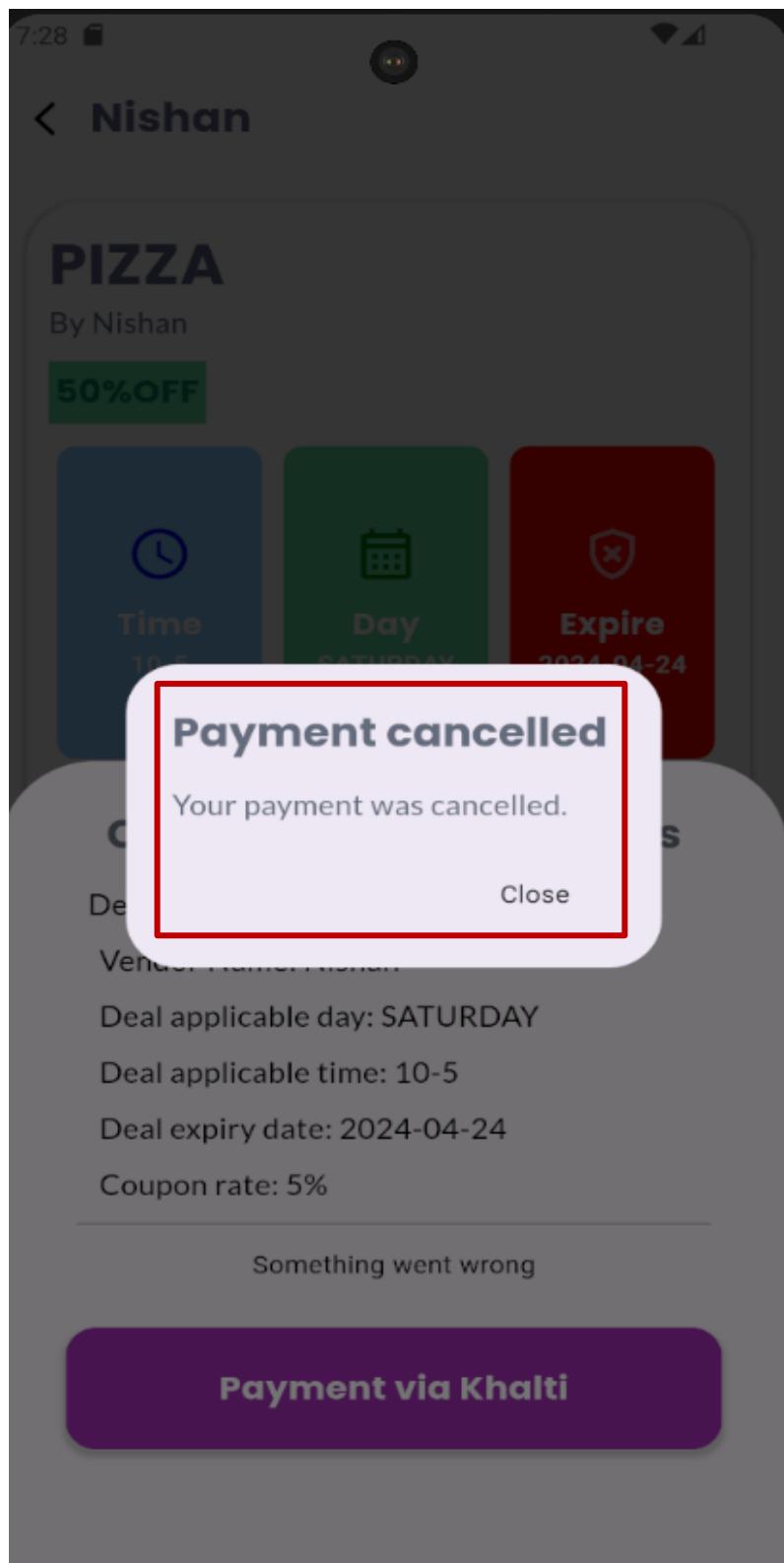


Figure 263: Coupon Purchased Failed Testing -5.

**4.2.21 Active coupon share**

objective	To share active coupon.
<b>Actions</b>	Navigate to purchases section and select an active coupon then select share option.
<b>Expected results</b>	Sharing option should be displayed.
<b>Obtained Results</b>	Sharing option was displayed.
<b>Conclusion</b>	Test Successful.

*Table 40: Share coupon testing table.*

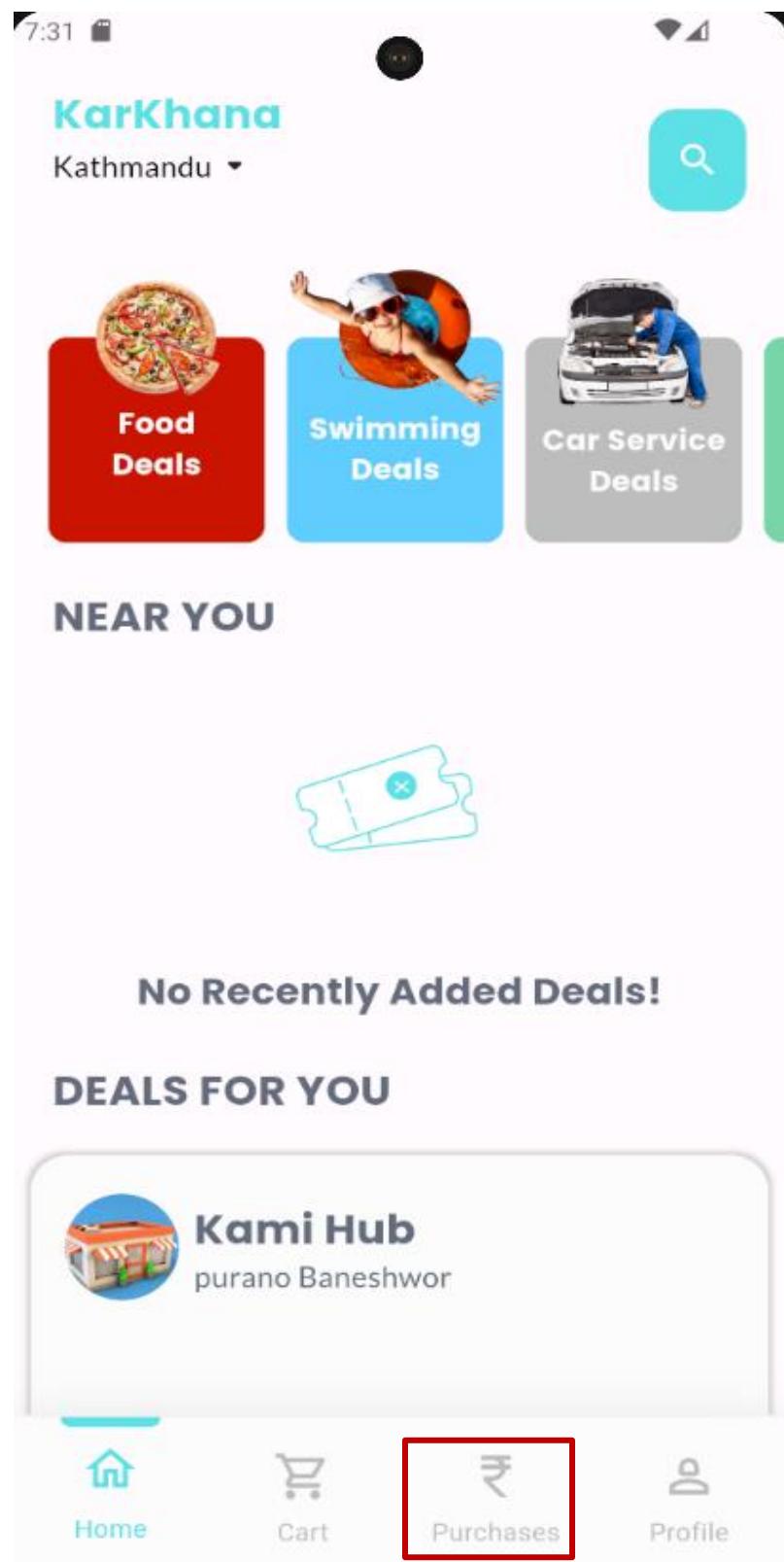


Figure 264: Active Coupon Share Testing -1.

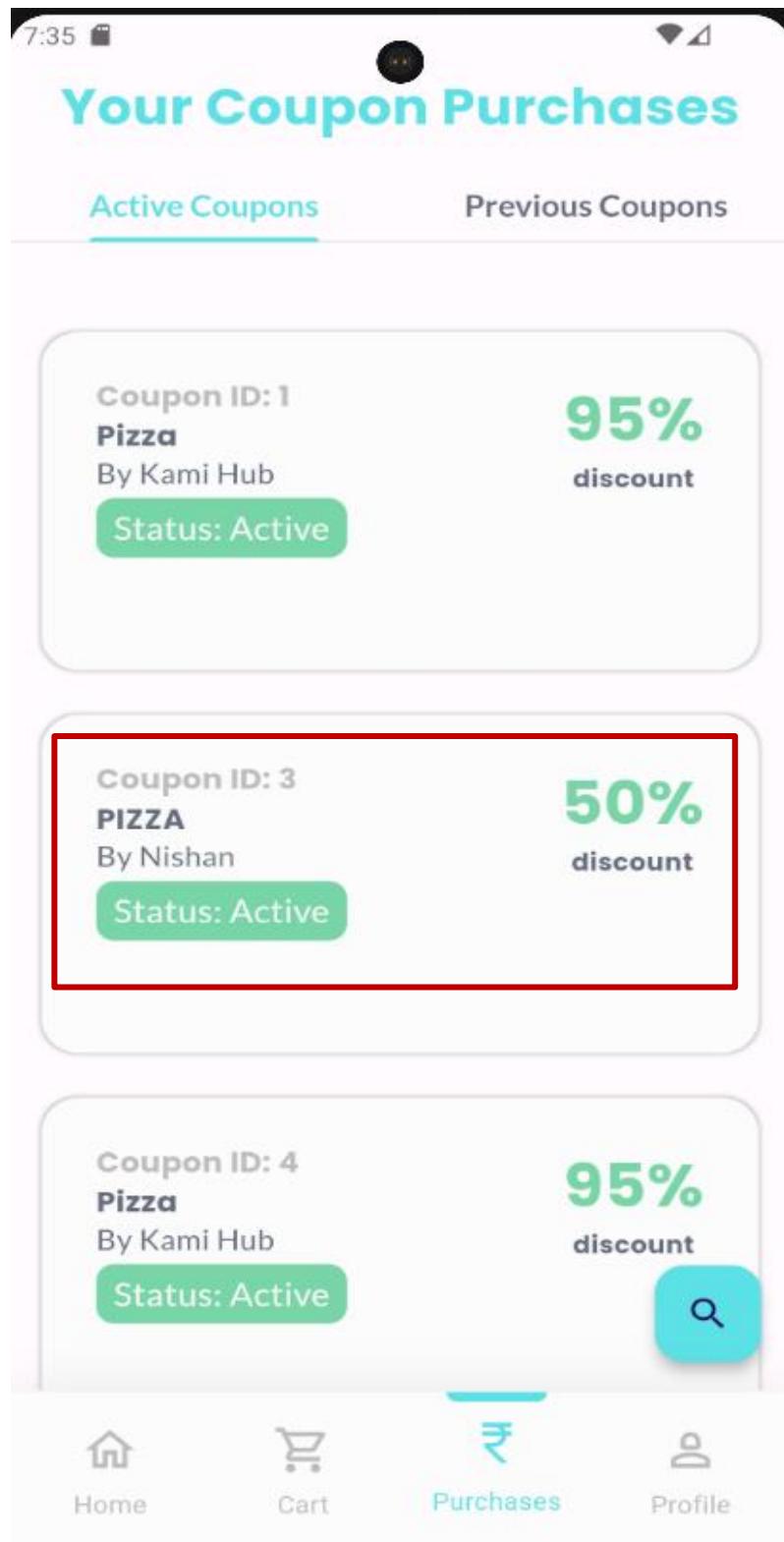


Figure 265: Active Coupon Share Testing -2.

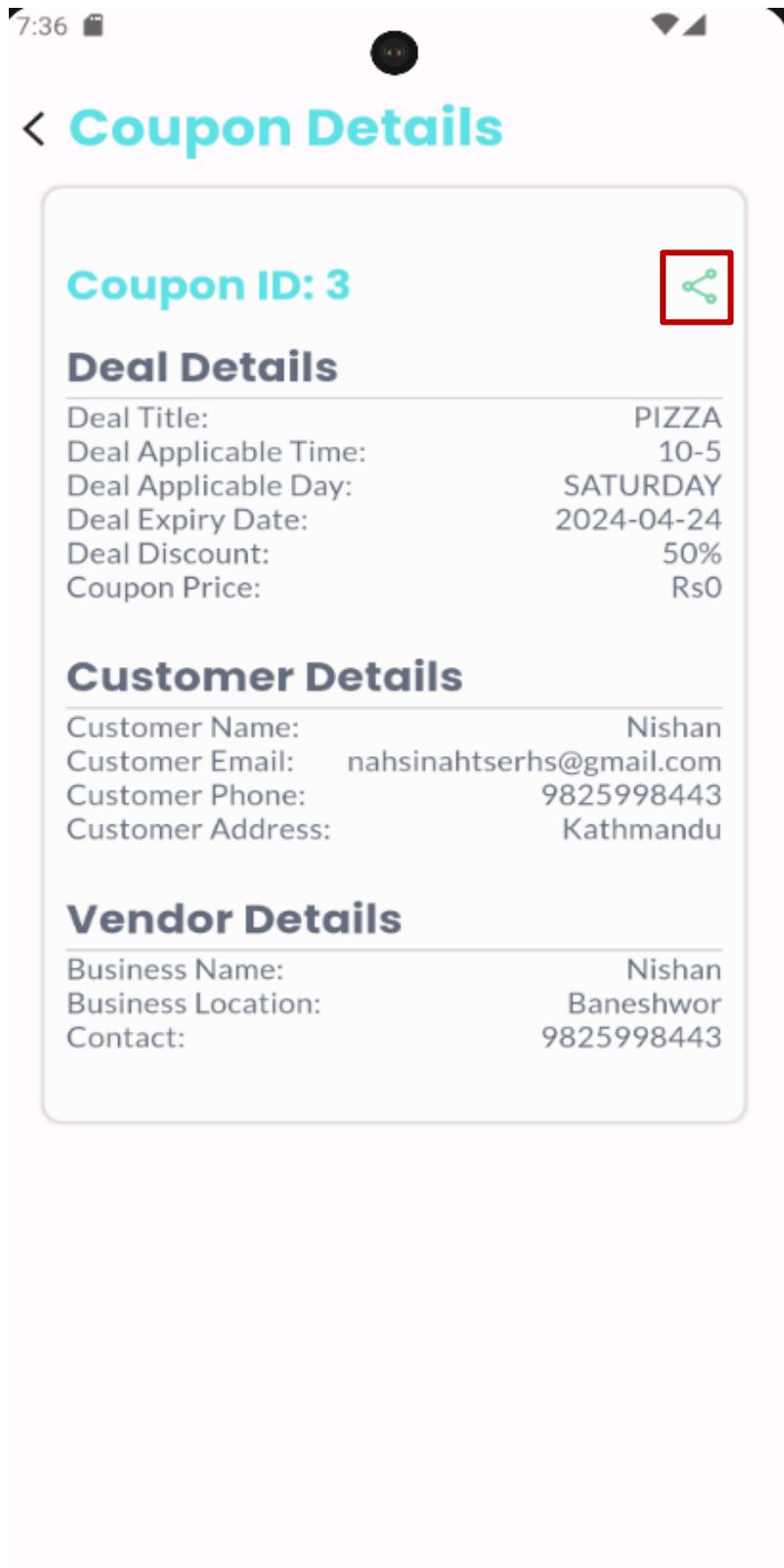


Figure 266: Active Coupon Share Testing -3.

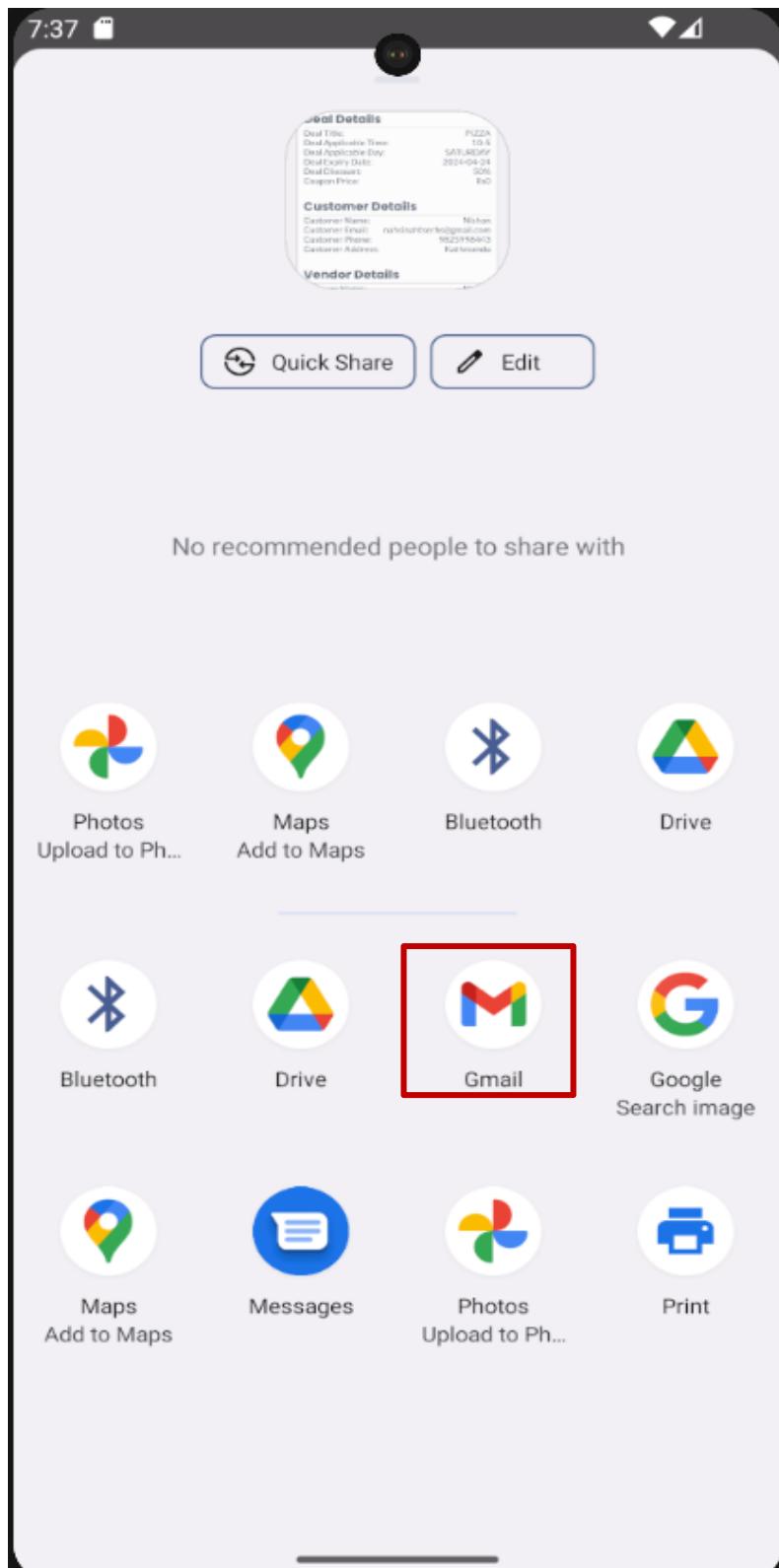


Figure 267: Active Coupon Share Testing -4.

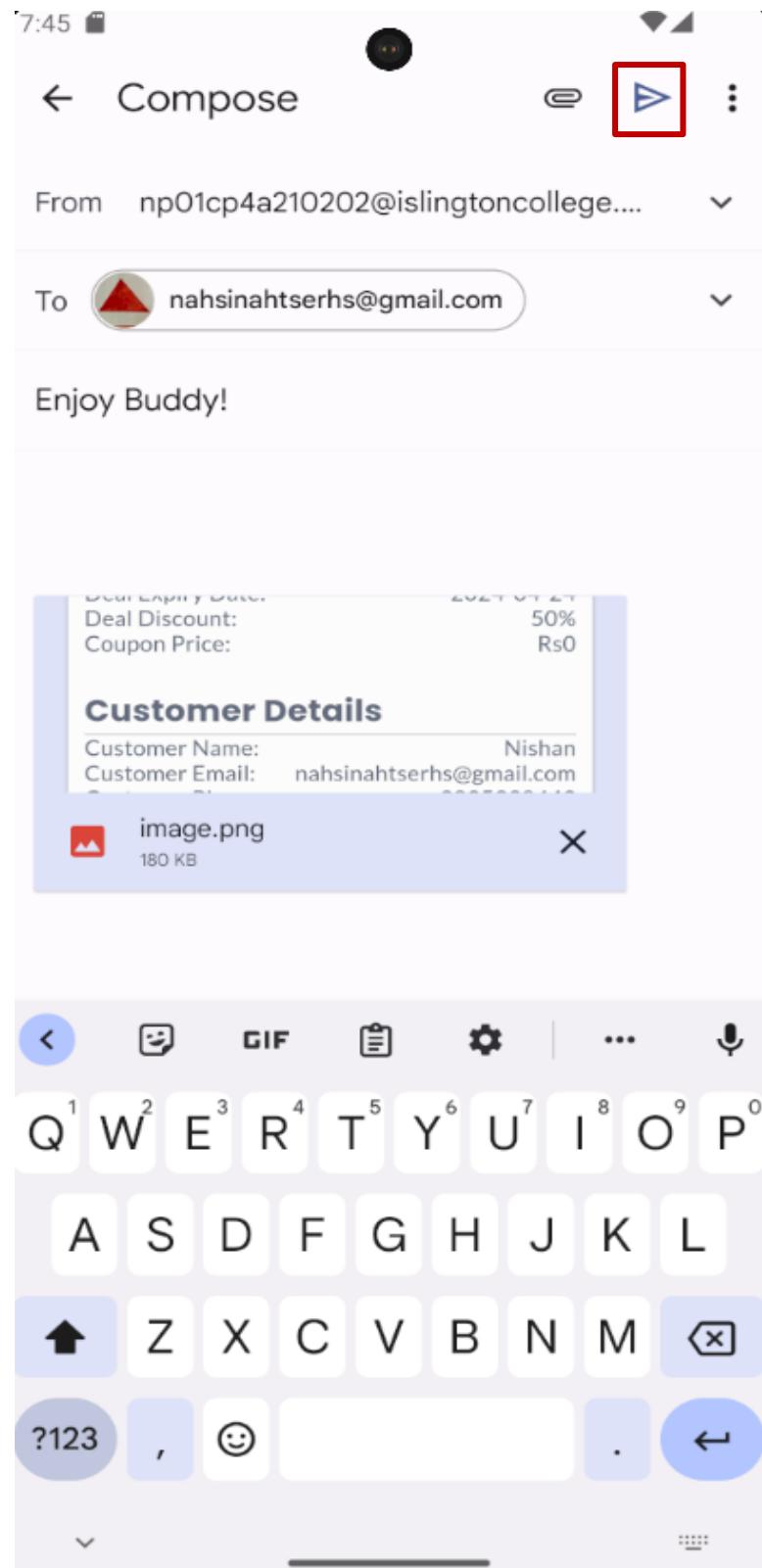


Figure 268: Active Coupon Share Testing -5.

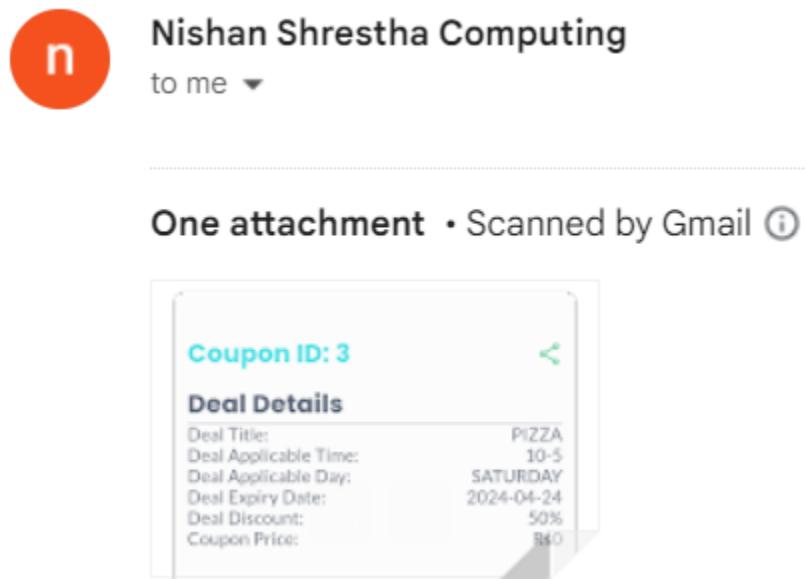


Figure 269: Coupon Successfully Received.

#### 4.2.22 Display active coupon

objective	To display active coupon
<b>Actions</b>	Navigate to purchases section
<b>Expected results</b>	Coupons with active status should be displayed.
<b>Obtained Results</b>	Coupons with active status were displayed.
<b>Conclusion</b>	Test Successful.

Table 41: Display active coupon testing table.

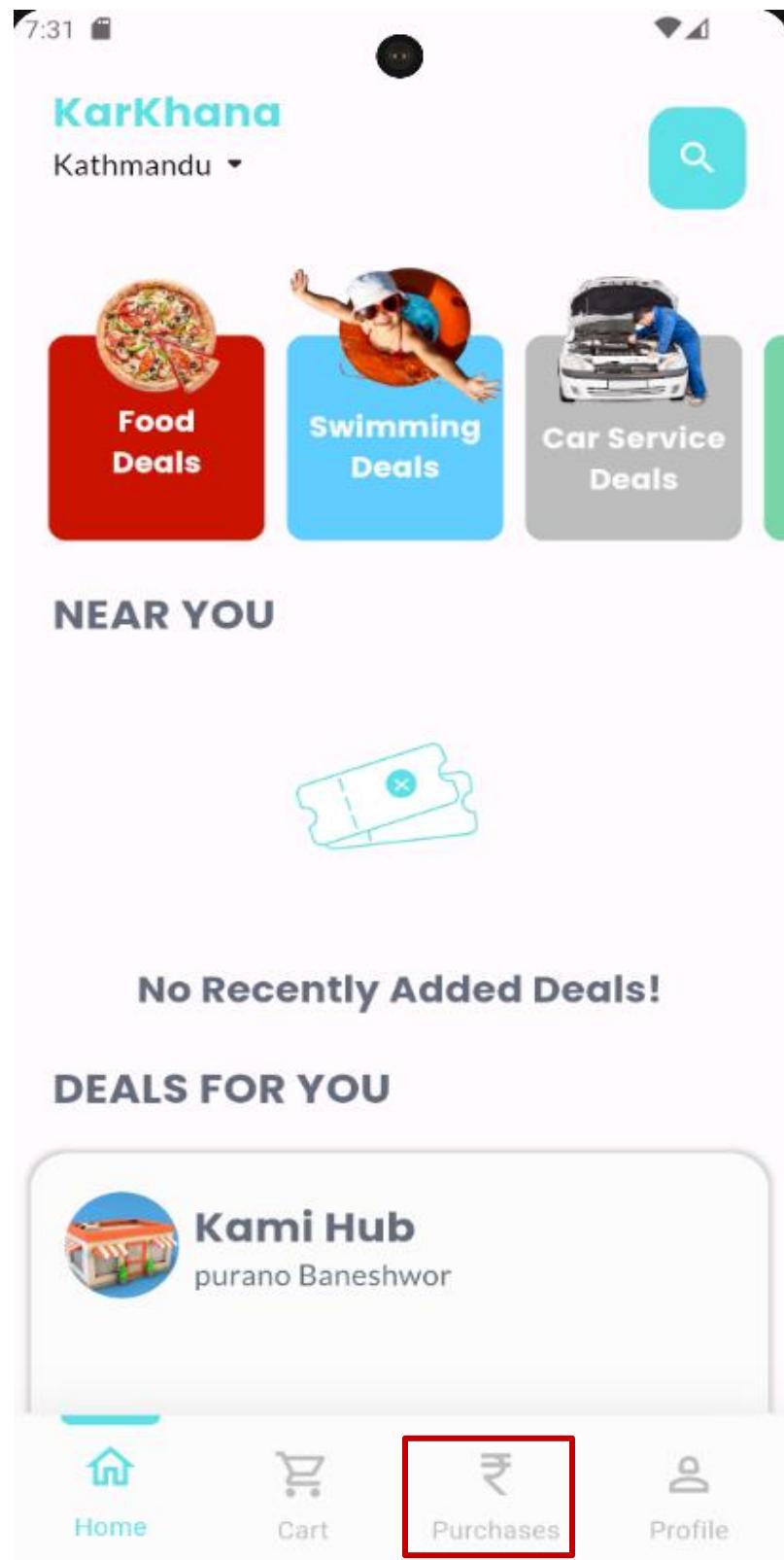


Figure 270: Active Coupon Testing -1.

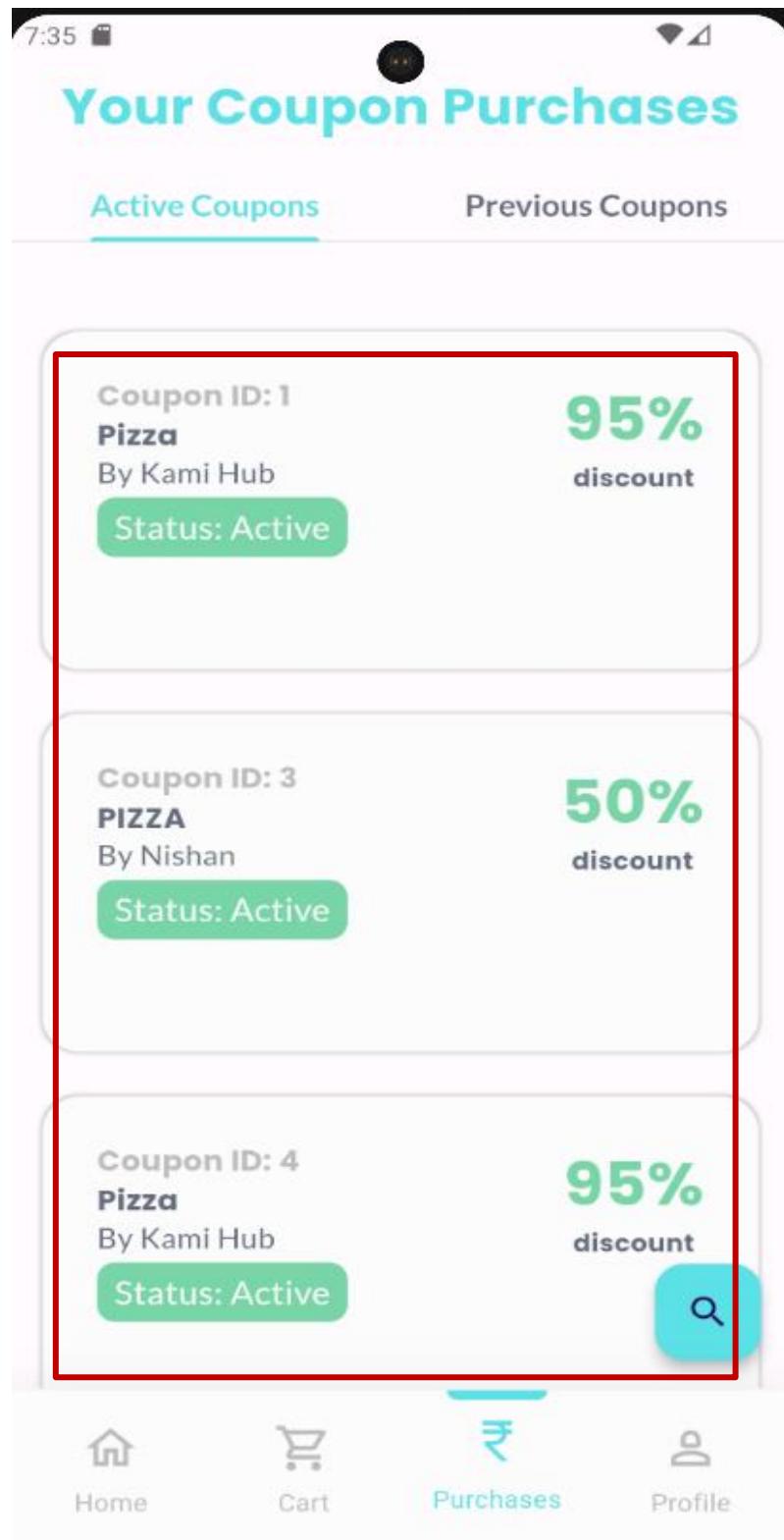


Figure 271: Active Coupon Testing -2.

**4.2.23 Display previous coupon**

objective	To display previous coupon
<b>Actions</b>	Navigate to purchases section and scroll to previous coupon section
<b>Expected results</b>	Coupons with used status should be displayed.
<b>Obtained Results</b>	Coupons with used status were displayed.
<b>Conclusion</b>	Test Successful.

*Table 42: Display previous coupon testing table.*

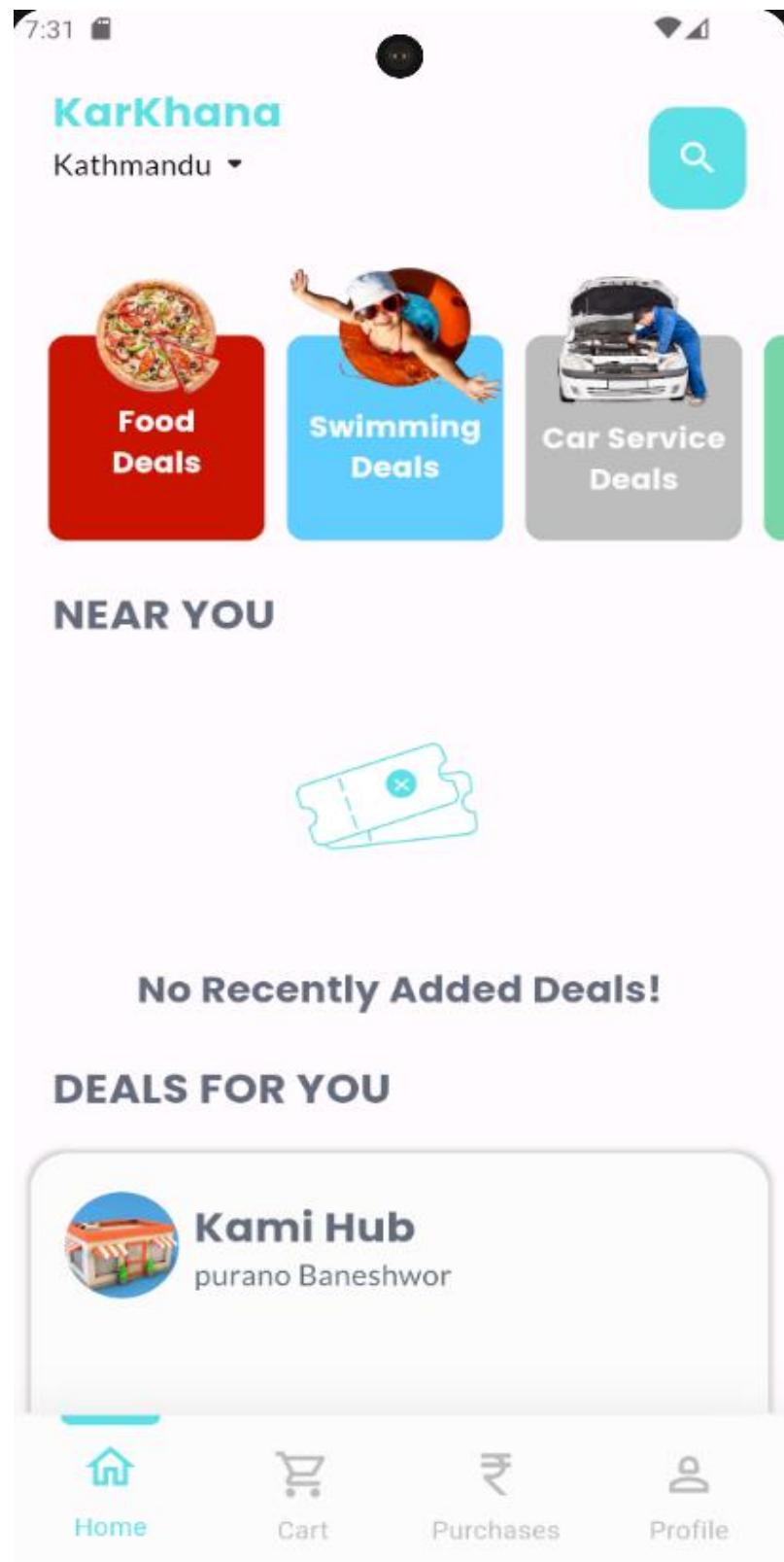


Figure 272: Previous Coupon Testing -1.



Figure 273: Previous Coupon Testing -2.

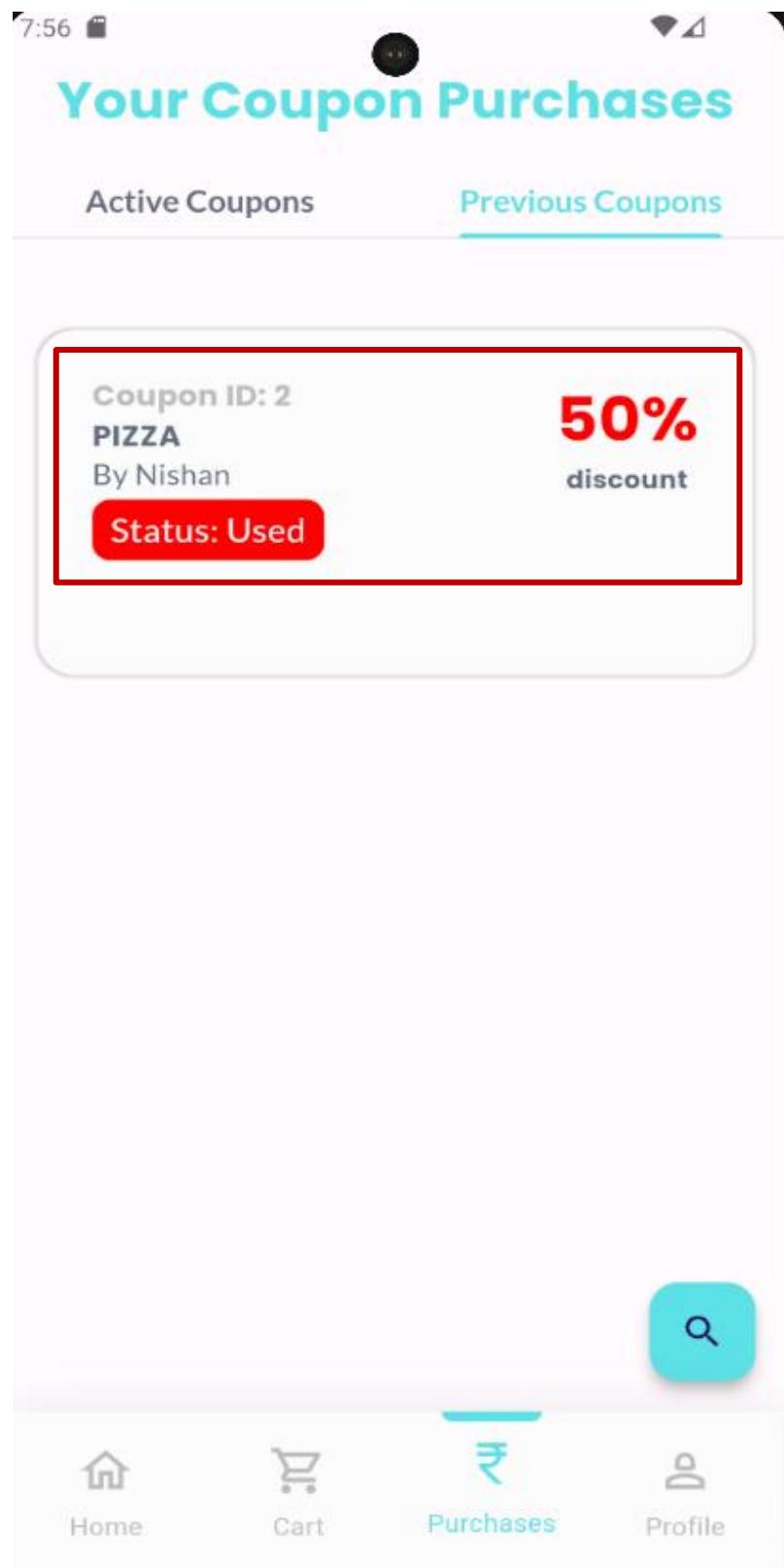


Figure 274: Previous Coupon Testing -3.

**4.2.24 Current location recommended deals**

objective	To display deals available in current location.
<b>Actions</b>	Set current location of user.
<b>Expected results</b>	Deals from same current location should be displayed.
<b>Obtained Results</b>	Deals from same current location was displayed.
<b>Conclusion</b>	Test Successful.

*Table 43: Current location recommendation testing table.*

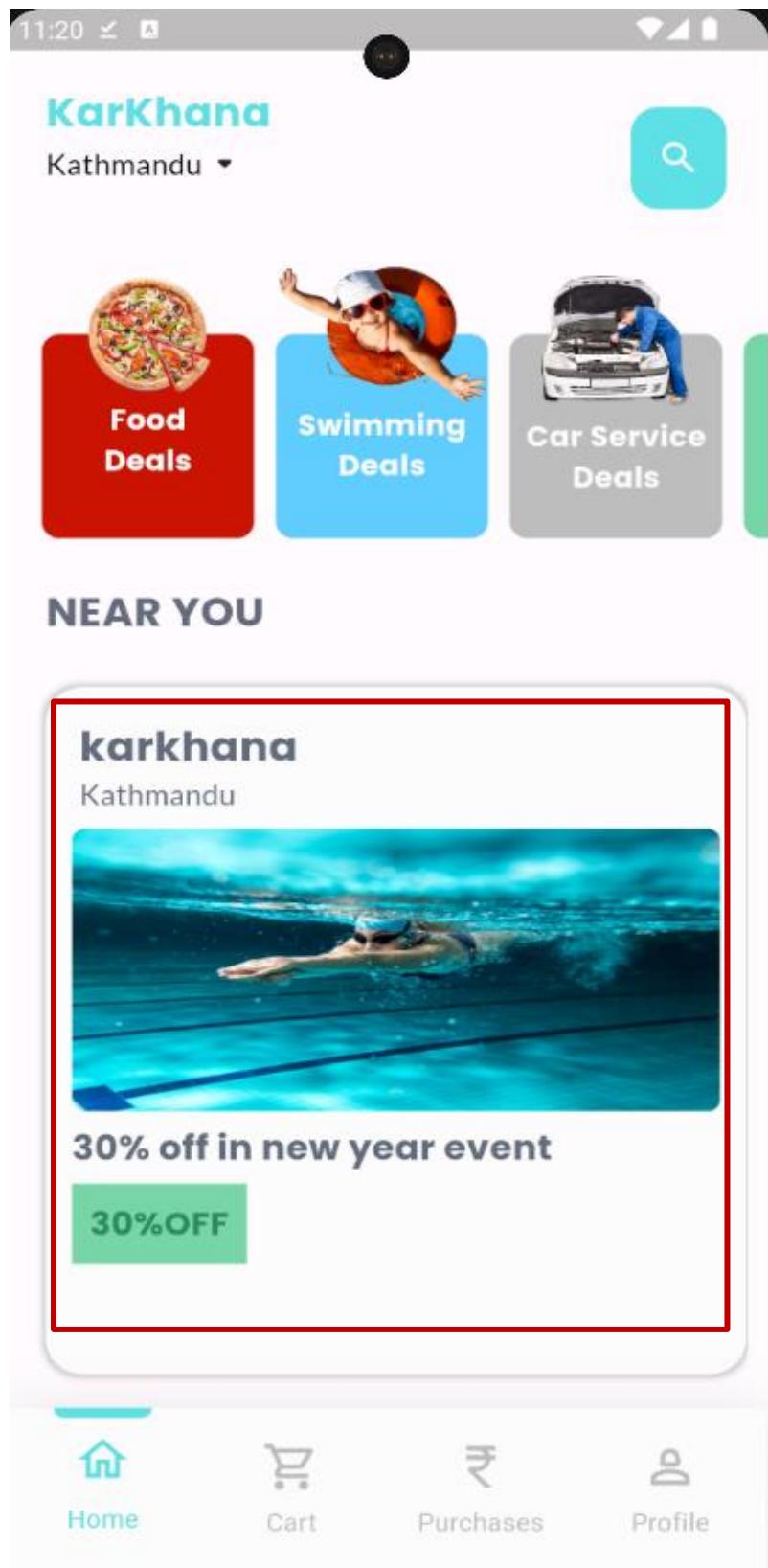


Figure 275: Current Location Recommendation Testing.

**4.2.25 Generated coupon displayed.**

objective	To view generate coupon.
<b>Actions</b>	Navigate to active coupons of purchases section and select one active coupon to view the generated coupon
<b>Expected results</b>	Coupon with details should be displayed.
<b>Obtained Results</b>	Coupon with details was displayed.
<b>Conclusion</b>	Test Successful.

*Table 44: Generated coupon testing table.*

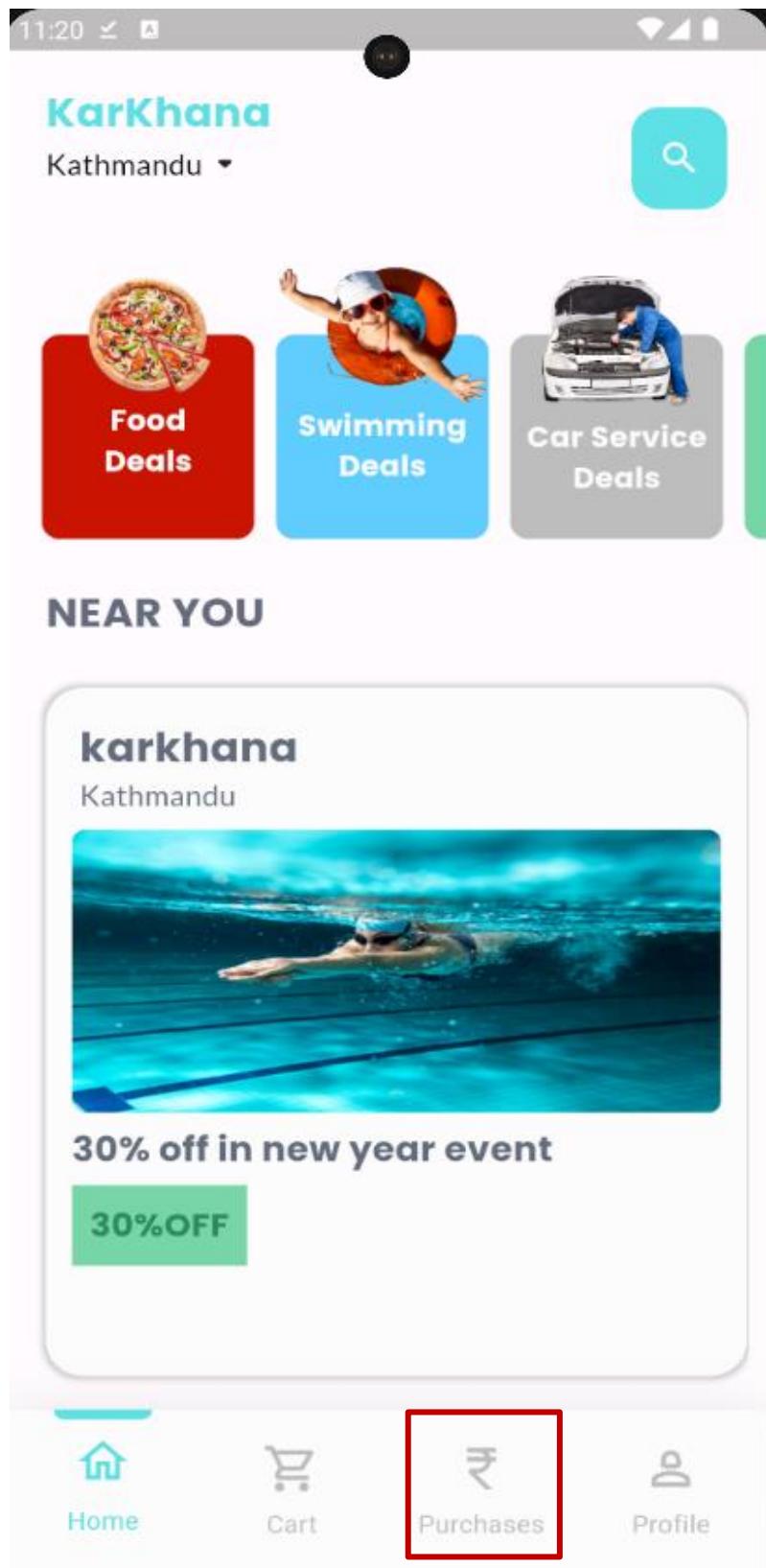


Figure 276: Generate Coupon Display Testing -1.

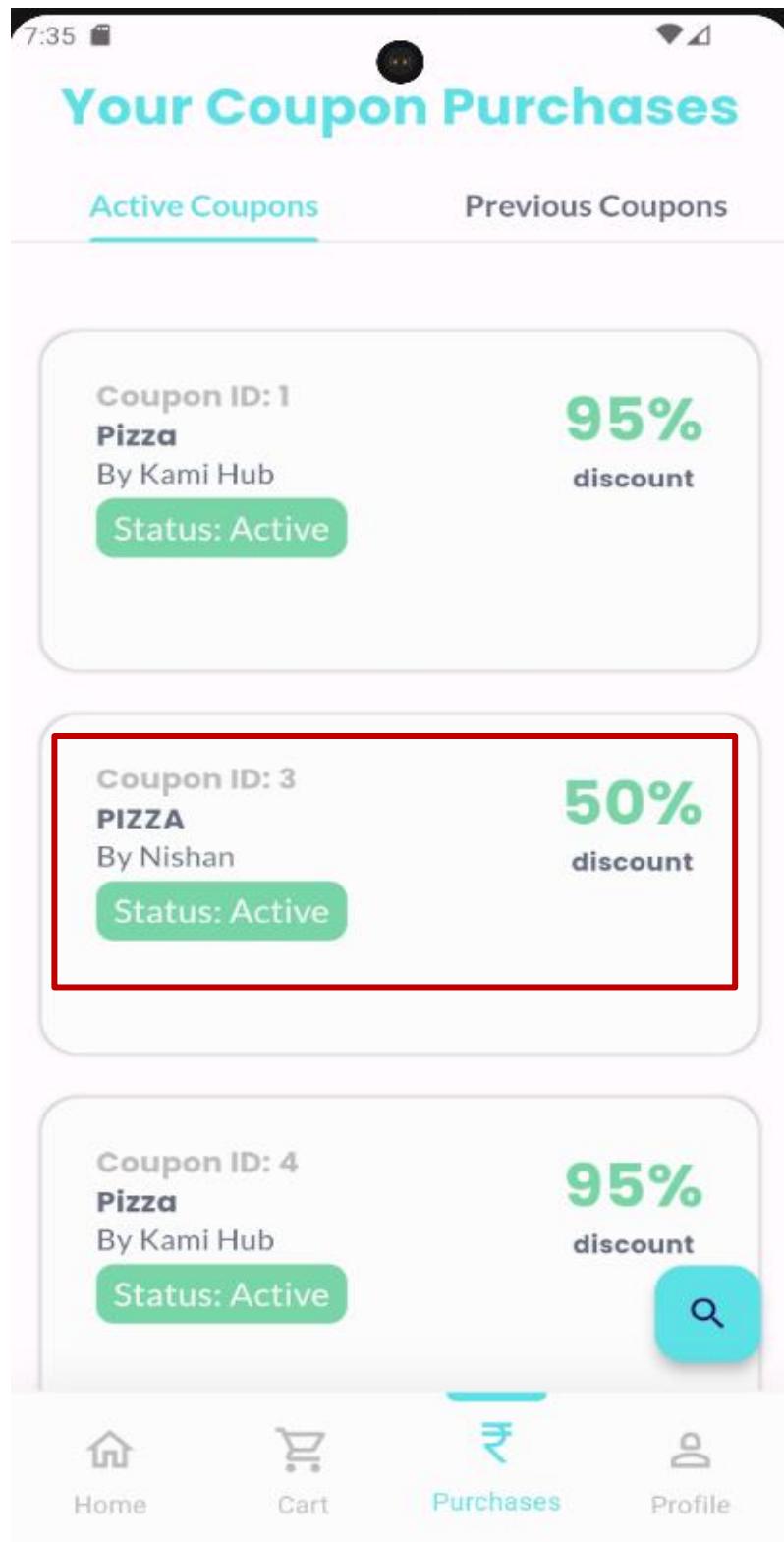


Figure 277: Generated Coupon Testing -2.

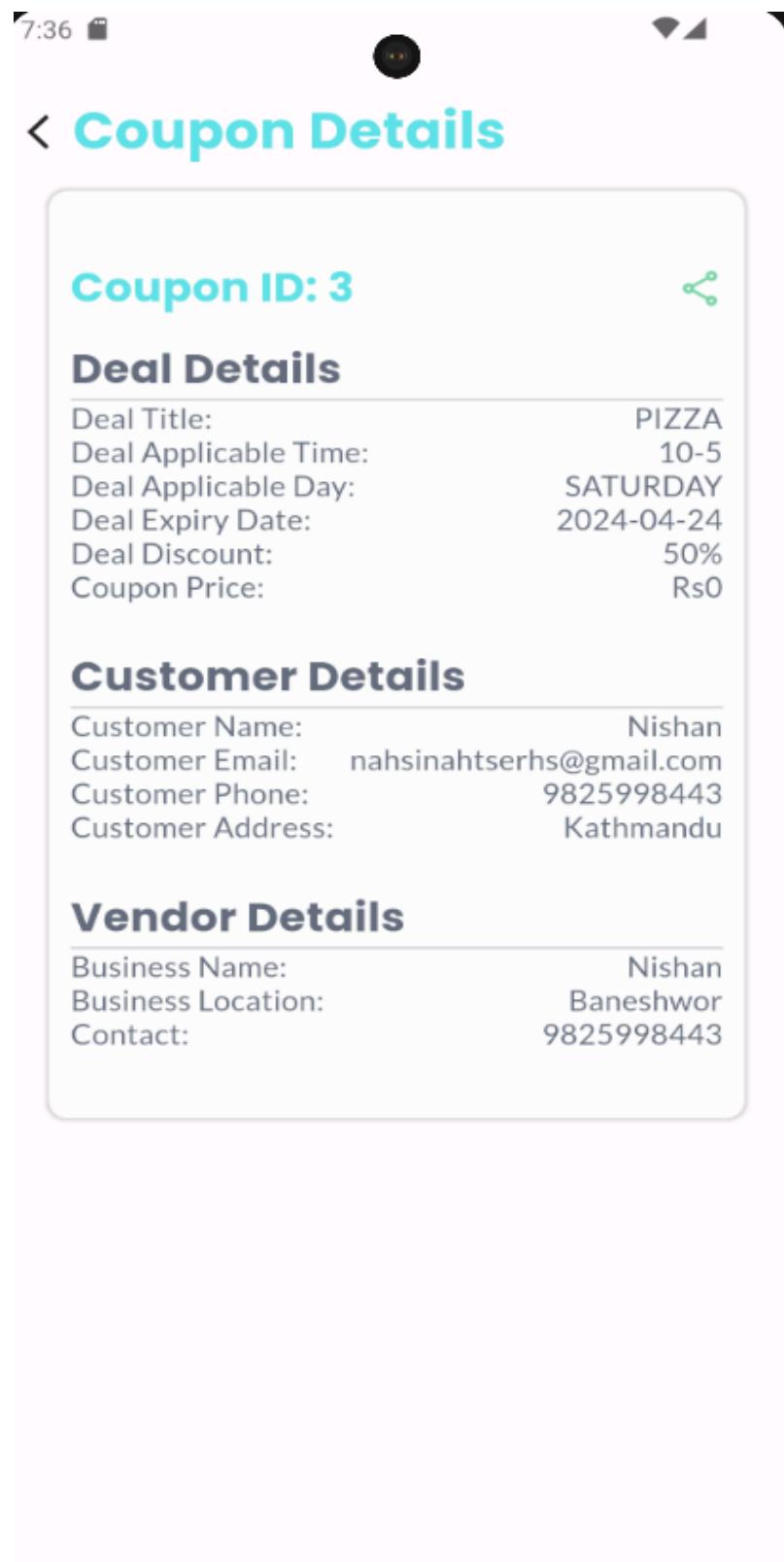


Figure 278: Generated Coupon Testing -3.

#### 4.2.26 Logout

objective	To logout of the system.
Actions	Press on logout button and confirm logout.
Expected results	Logout success message should be displayed, and user should be navigated to welcome screen.
Obtained Results	Logout success message was displayed, and user was navigated to welcome screen.
Conclusion	Test Successful.

Table 45: Logout testing table.

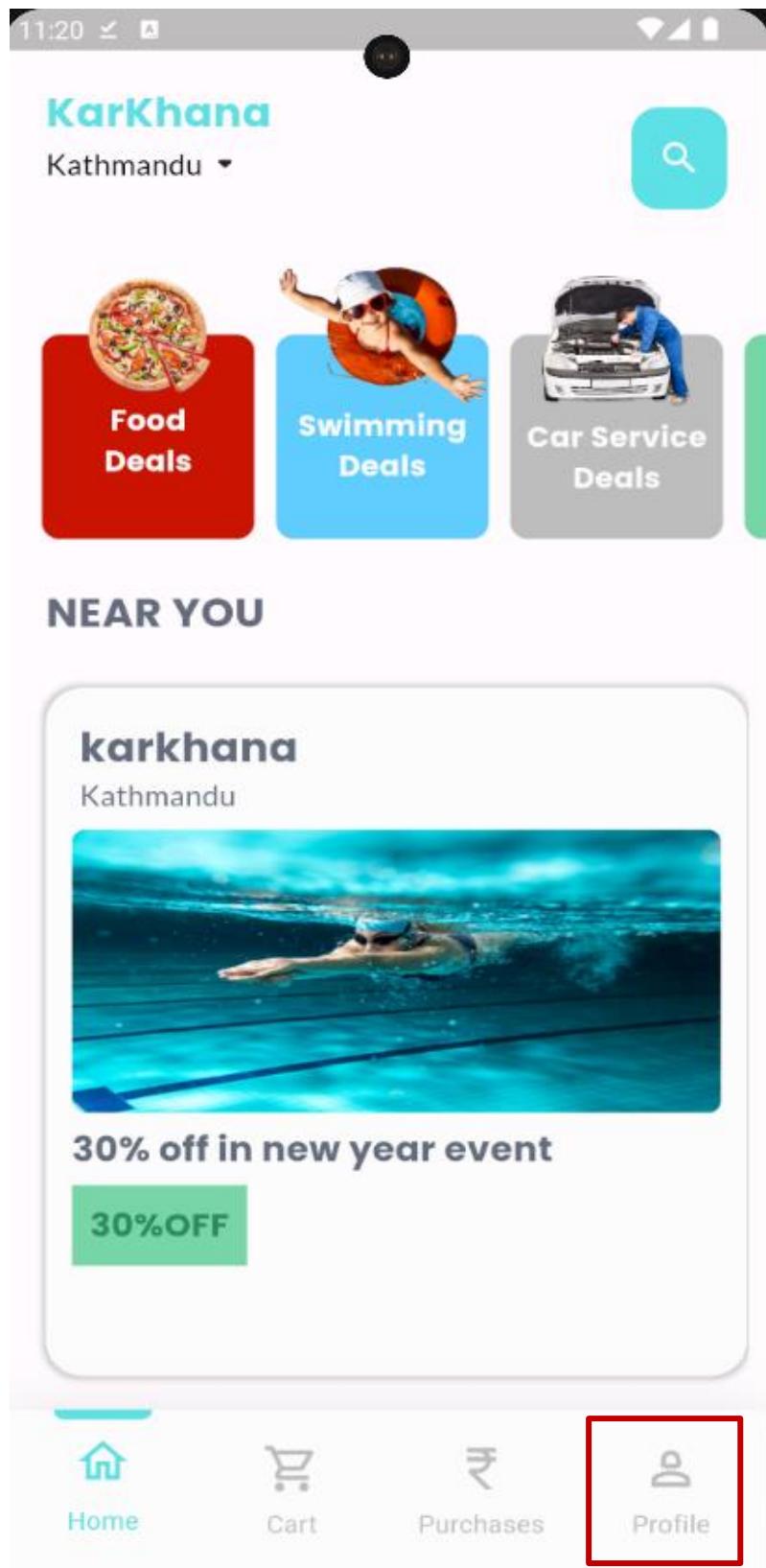


Figure 279: Logout Testing -1.

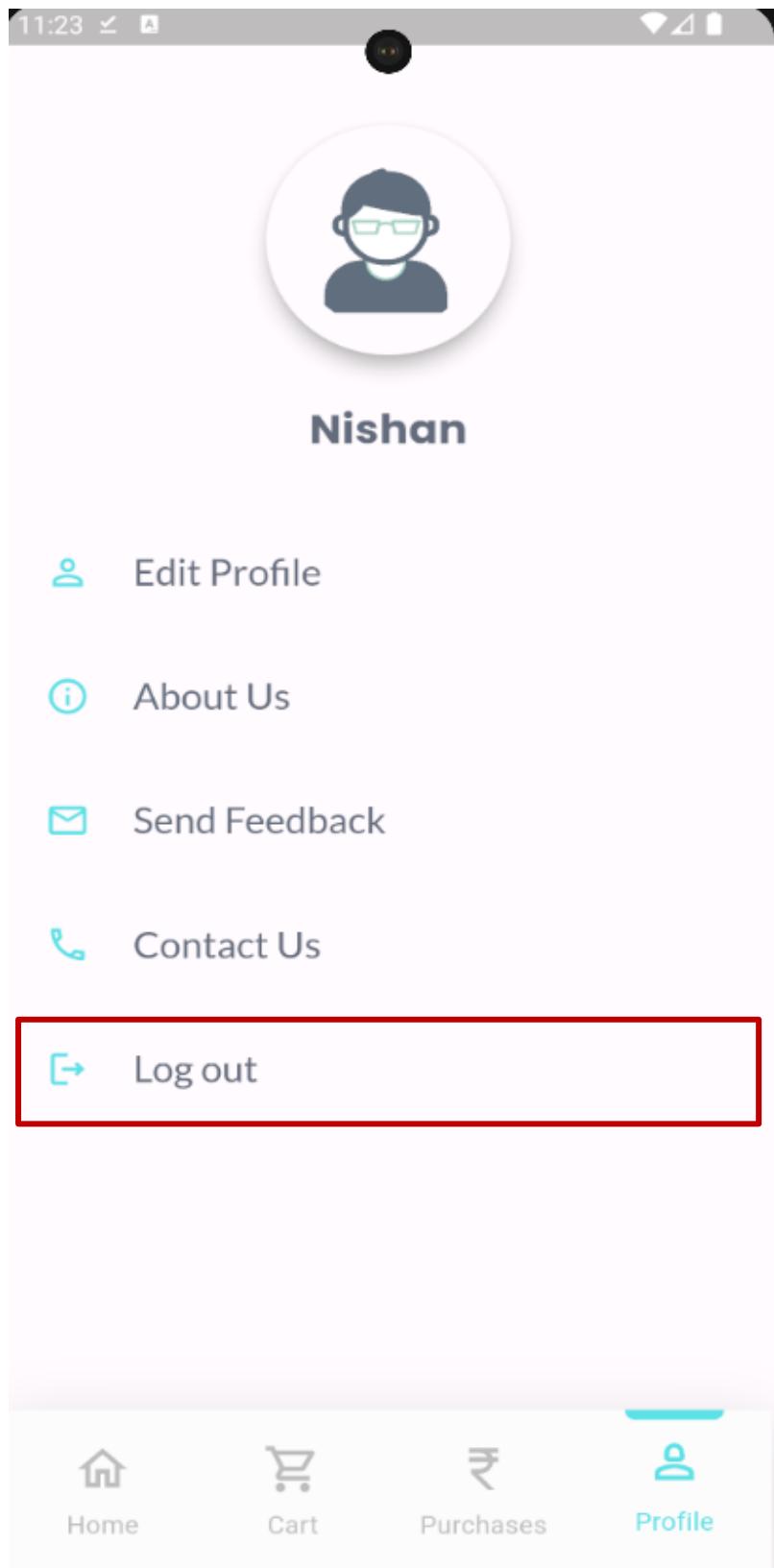


Figure 280: Logout Testing -2.

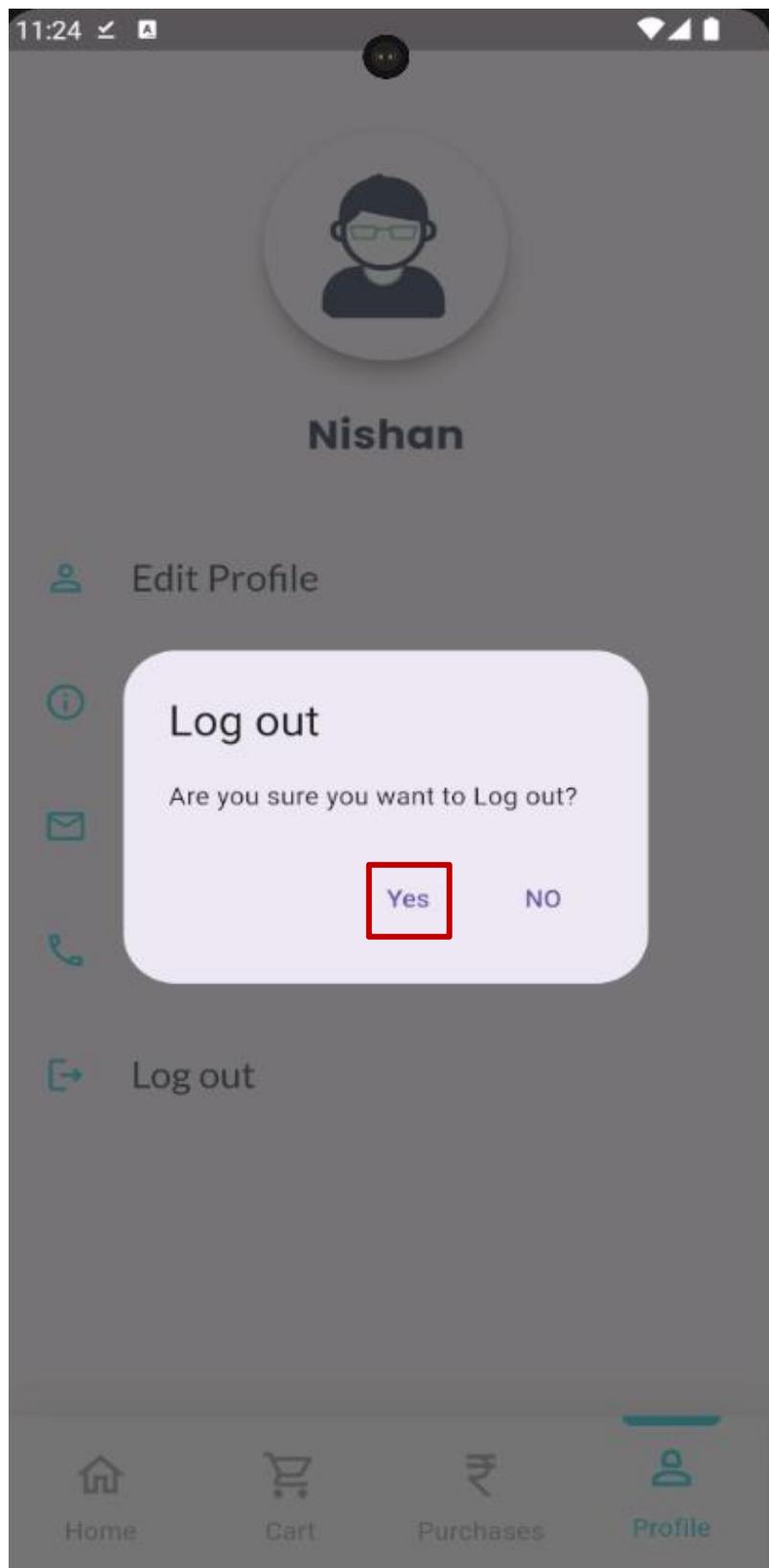


Figure 281: Logout Testing -3.

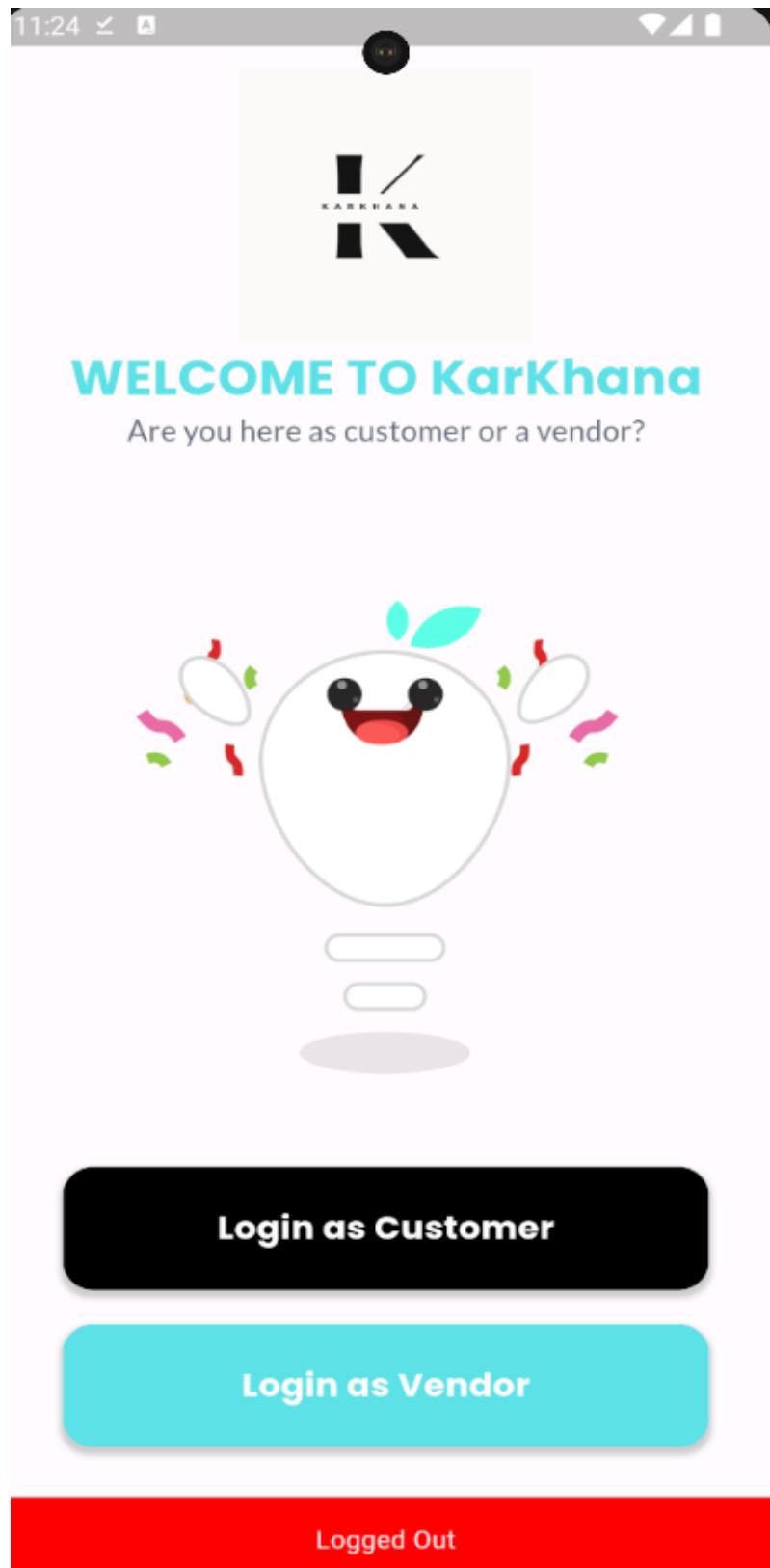


Figure 282: Logout Testing -4.

**For Vendor****4.2.28 Post registration details**

<b>objective</b>	To post registration details to admin or store it in database.
<b>Actions</b>	Fill out the registration form and press signup button.
<b>Input</b>	Name: "Karkhana", Email:" karkhana2024@gmail.com", Phone no: "9814024441", Location:" Kathmandu", About Business: "Since 1990", Opening days = "MON – WED", Opening time: "10AM – 3PM"
<b>Expected results</b>	Success message should be displayed, and registration details should be posted to admin panel.
<b>Obtained Results</b>	Success message was displayed, and registration details was posted to admin panel.
<b>Conclusion</b>	Test Successful.

Table 46: Post registration details testing table.

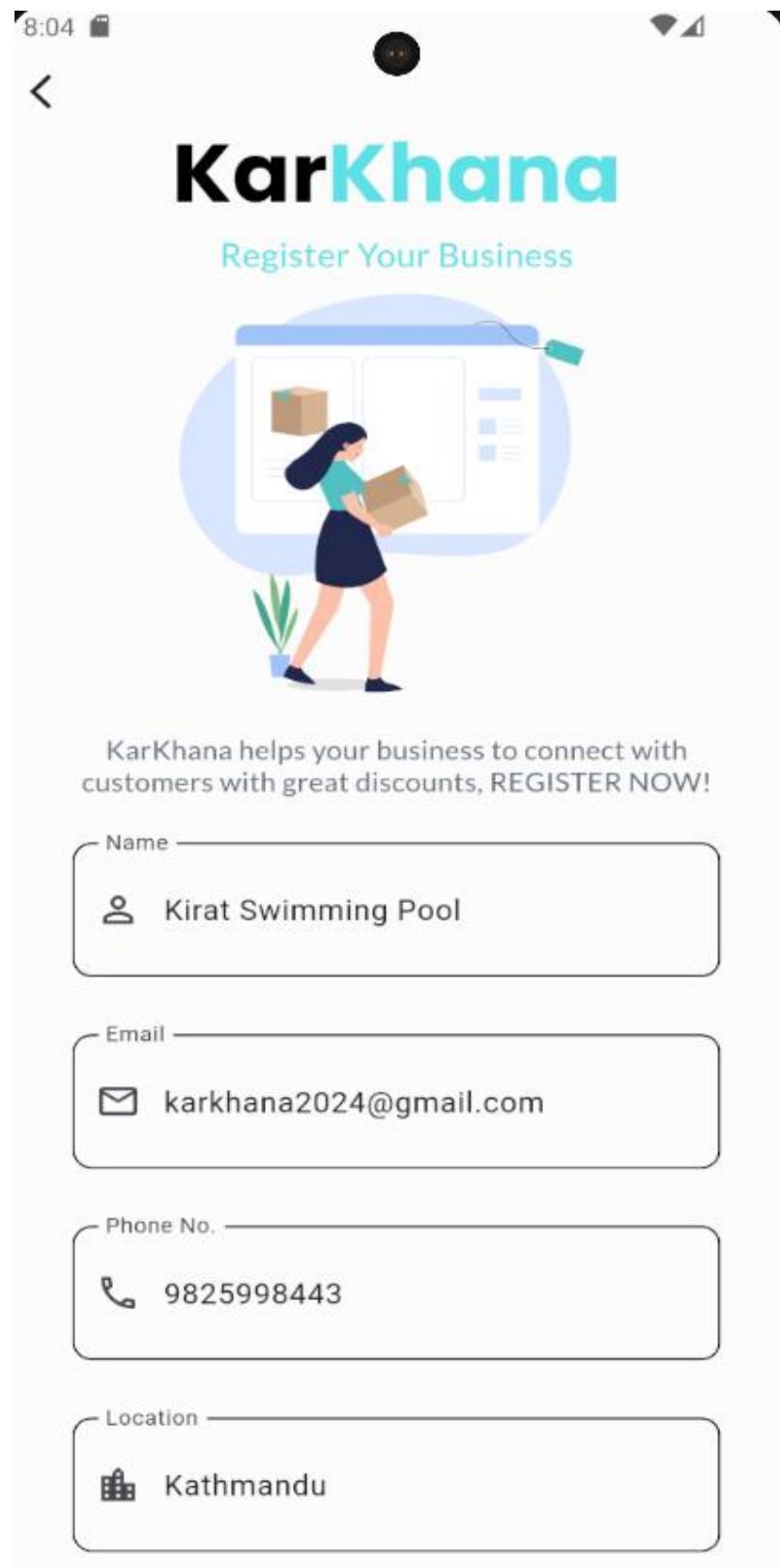


Figure 283: Registration testing -1.

8:04

Location —  
Kathmandu

About Business —  
Since 1990

Opening days —  
MON - WED

Opening Time —  
10AM -10PM

Password —  
.....

Confirm Password —  
.....

**signup**

Already have an Account? [Login](#)

Figure 284: Registration testing -2.

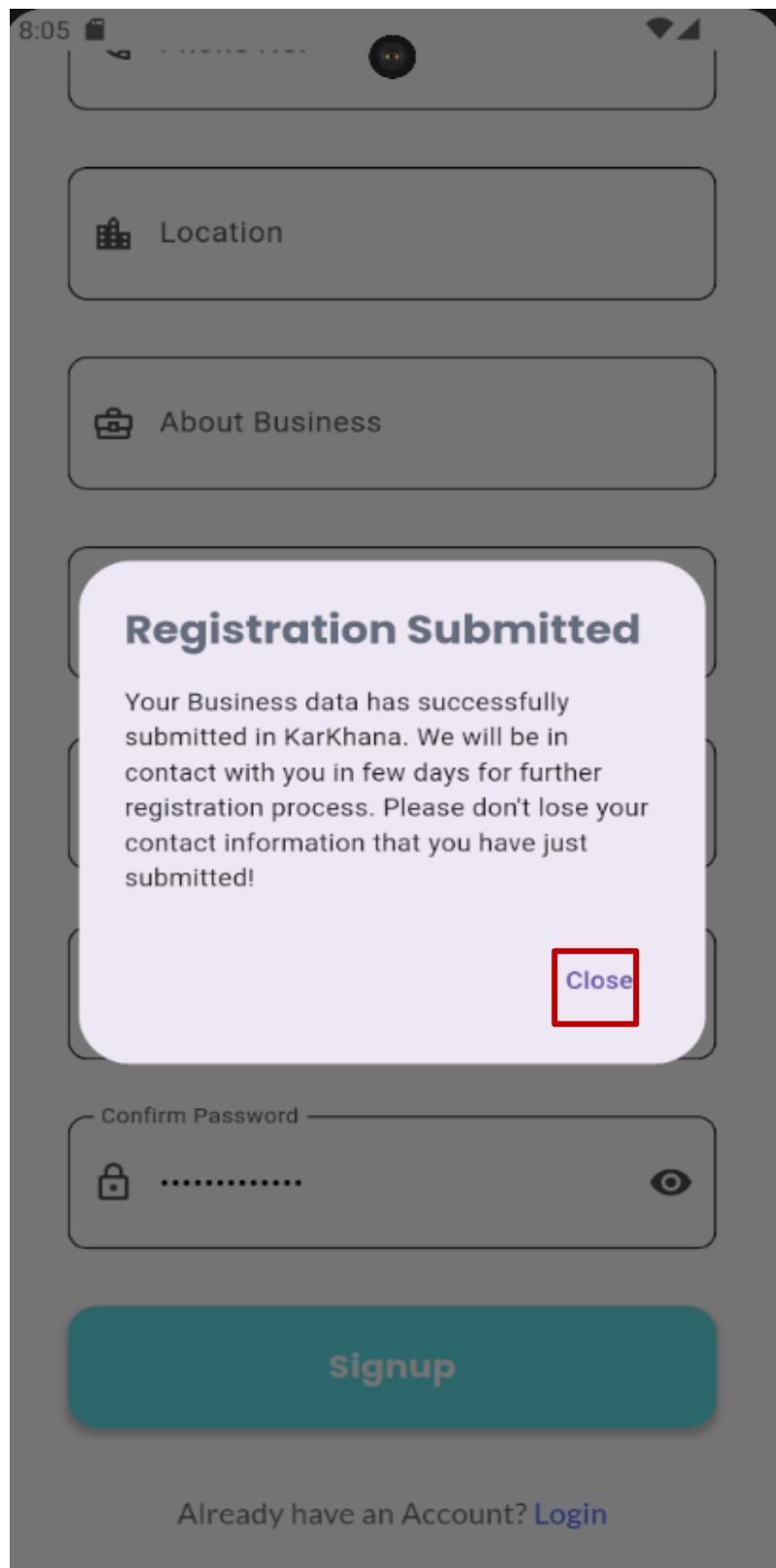


Figure 285: Registration testing -3.

#### 4.2.29 Registration field validation

objective		To check the field validation in registration page.
Actions	Enter empty field and press signup button.	
Input	Name: "", Email: "", Phone no: "", Location: "", About: "", Opening days: "", Opening time: ""	
Expected results	Field validation error should be displayed.	
Obtained Results	Field validation error was displayed.	
Conclusion	Test Successful.	

Table 47: Registration field validation testing table.

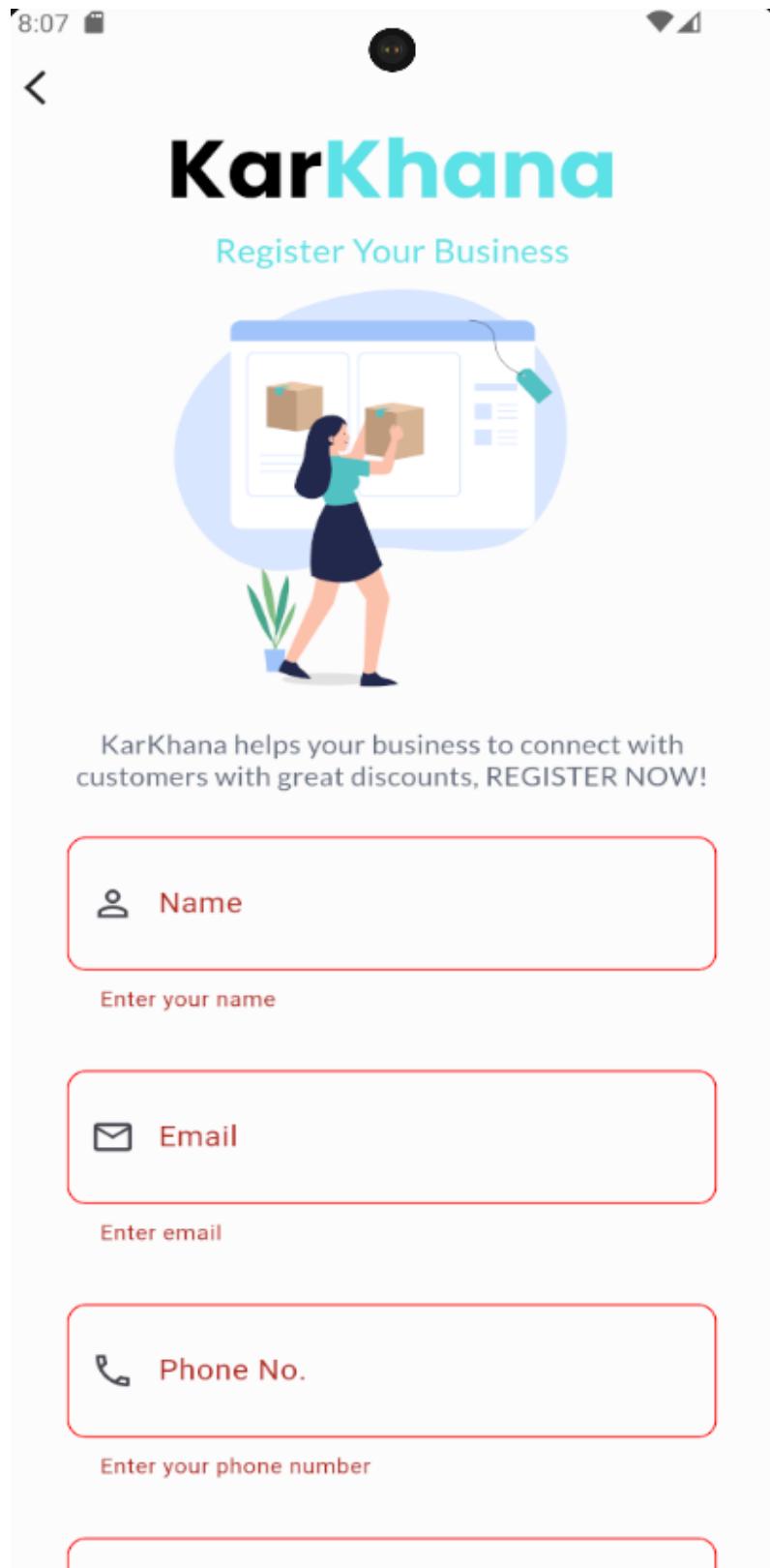


Figure 286: Registration field validation testing -1.

The image shows a registration form with several input fields and validation messages. The fields are outlined in red, indicating they are empty or invalid.

- About Business:** An icon of a briefcase. The message "Empty Field!" is displayed below the field.
- Opening days:** An icon of a calendar. The message "Empty Field!" is displayed below the field.
- Opening Time:** An icon of a clock. The message "Empty Field!" is displayed below the field.
- Password:** An icon of a lock. The message "Enter password" is displayed below the field. To the right is an eye icon for password visibility.
- Confirm Password:** An icon of a lock. The message "Enter password" is displayed below the field. To the right is an eye icon for password visibility.

**signup**

Already have an Account? [Login](#)

Figure 287: Registration field validation testing -2.

#### 4.2.30 Login error

objective		To check invalid login credentials in login page.
<b>Actions</b>	Enter wrong email while logging in.	
<b>Input</b>	Email: "gamer.trill@gmail.com", Password:"12345@psn"	
<b>Expected results</b>	Error message should be displayed.	
<b>Obtained Results</b>	Error message was displayed.	
<b>Conclusion</b>	Test Successful.	

Table 48: Login error testing table.

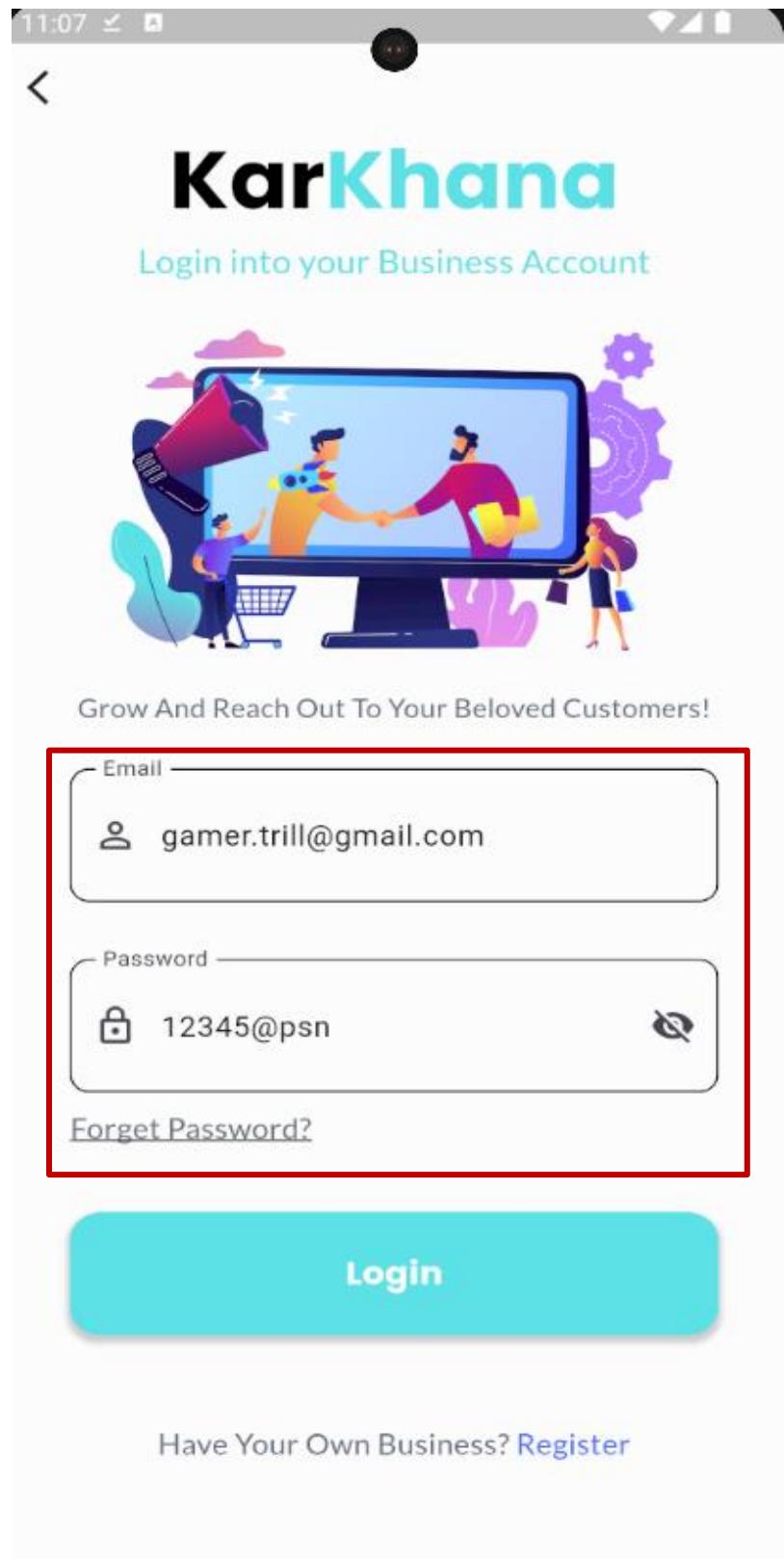


Figure 288: Login Error.

#### 4.2.31 Login success

<b>objective</b>	To login successfully into the system.
<b>Actions</b>	Enter login credentials and press login button.
<b>Input</b>	Email: "karkhana2024@gmail.com", Password: "nishan@5270"
<b>Expected results</b>	Login success message should be displayed and page should be navigated to homepage.
<b>Obtained Results</b>	Login success message was displayed, and page was navigated to homepage.
<b>Conclusion</b>	Test Successful.

Table 49: Login success testing table.

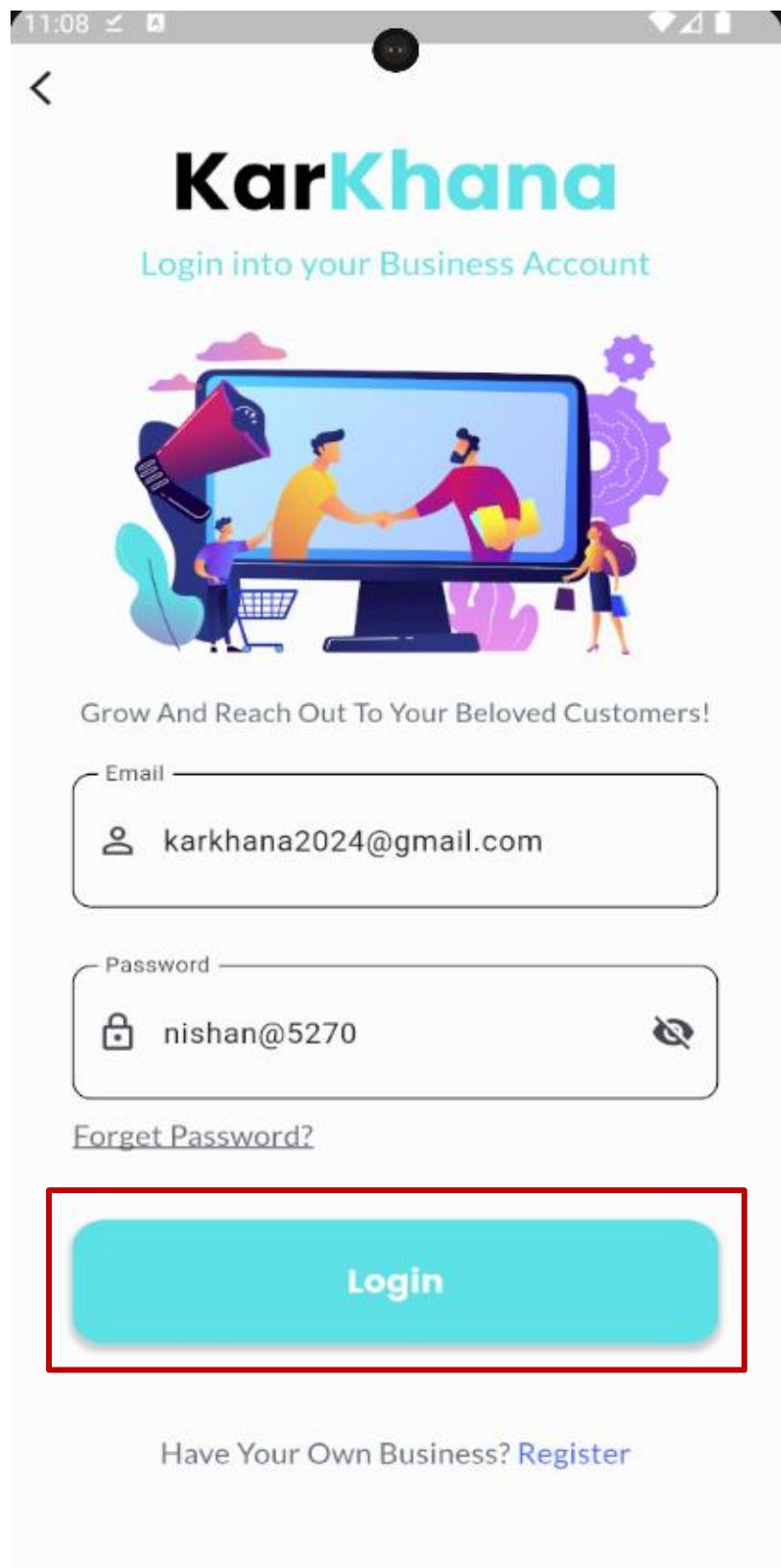


Figure 289: Login Testing -1.

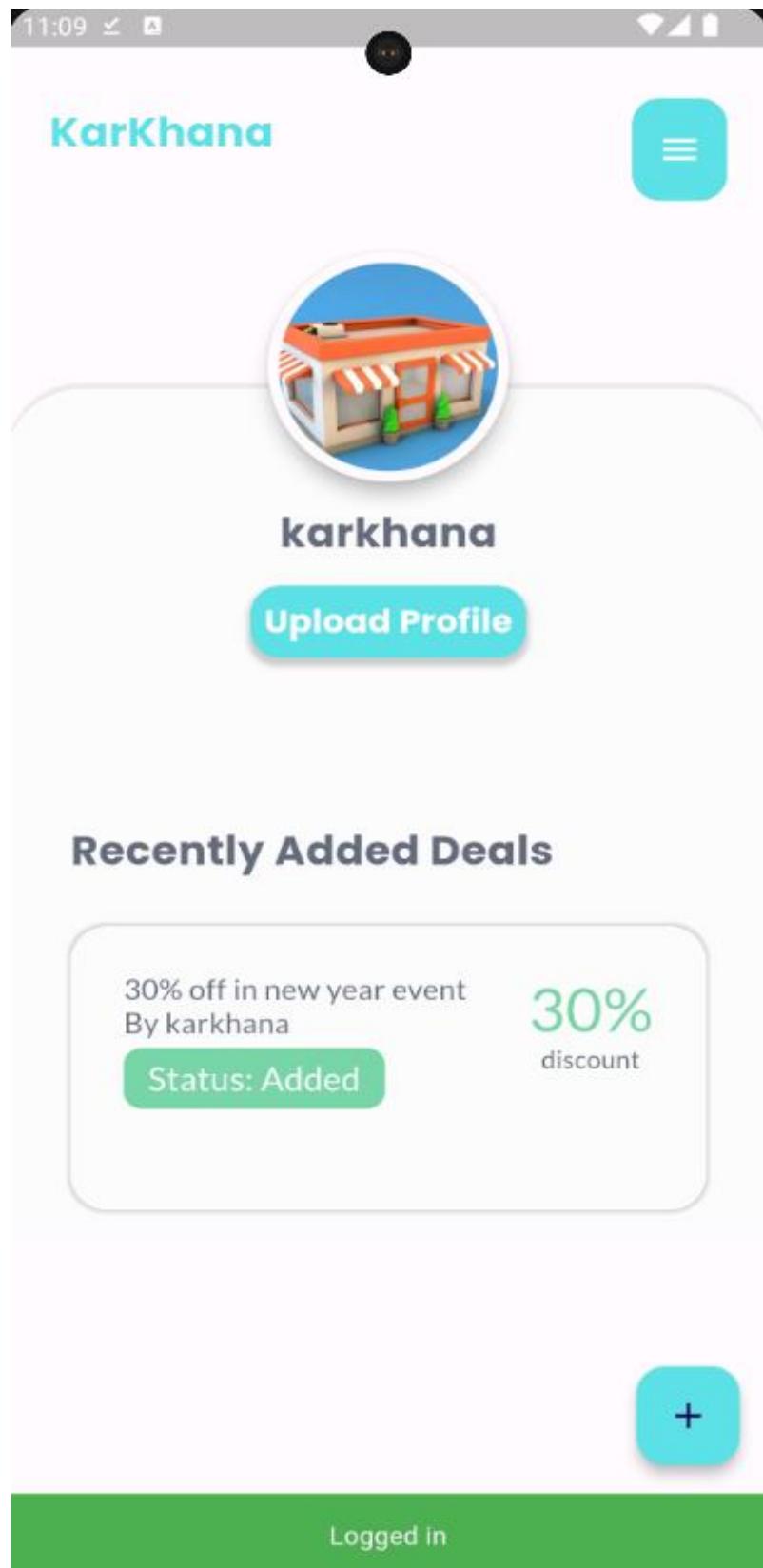


Figure 290: Logged in Successfully.

**4.2.32 Login field error**

<b>objective</b>	To check the field validation in login page.
<b>Actions</b>	Enter empty field and press login button.
<b>Input</b>	Email: "", Password: ""
<b>Expected results</b>	Field validation error should be displayed.
<b>Obtained Results</b>	Field validation error was displayed.
<b>Conclusion</b>	Test Successful.

*Table 50: Login field validation testing table.*

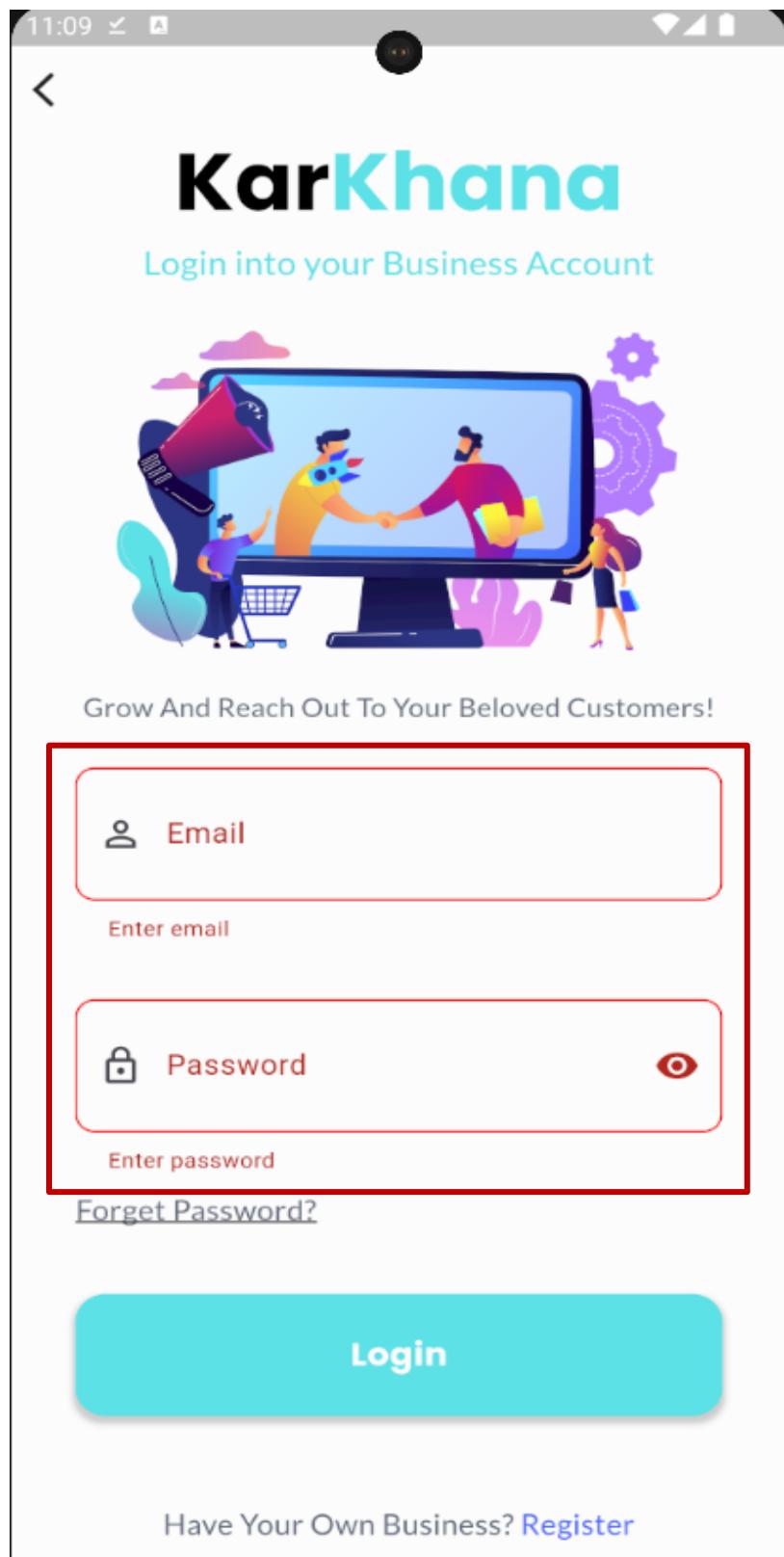


Figure 291: Login Validation testing.

**4.2.33 Add deal before uploading business profile**

objective	To Add Deal before uploading business profile in the system.
<b>Actions</b>	Press Add deal button and provide required data.
<b>Expected results</b>	Error message should be displayed.
<b>Obtained Results</b>	Error message was displayed.
<b>Conclusion</b>	Test Successful.

*Table 51: Adding deal before uploading business profile testing table.15*

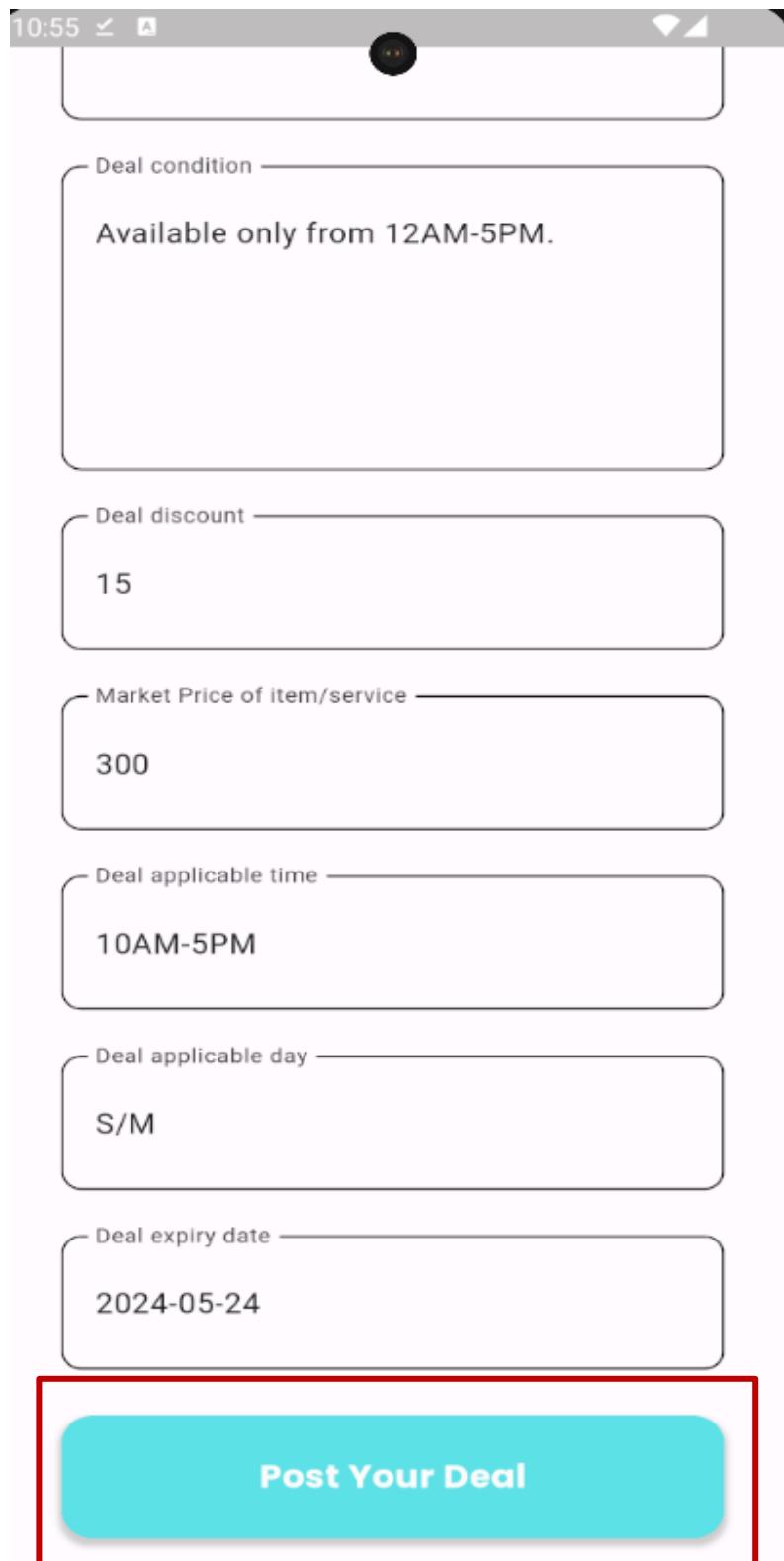


Figure 292: Add deal before business profile -1.

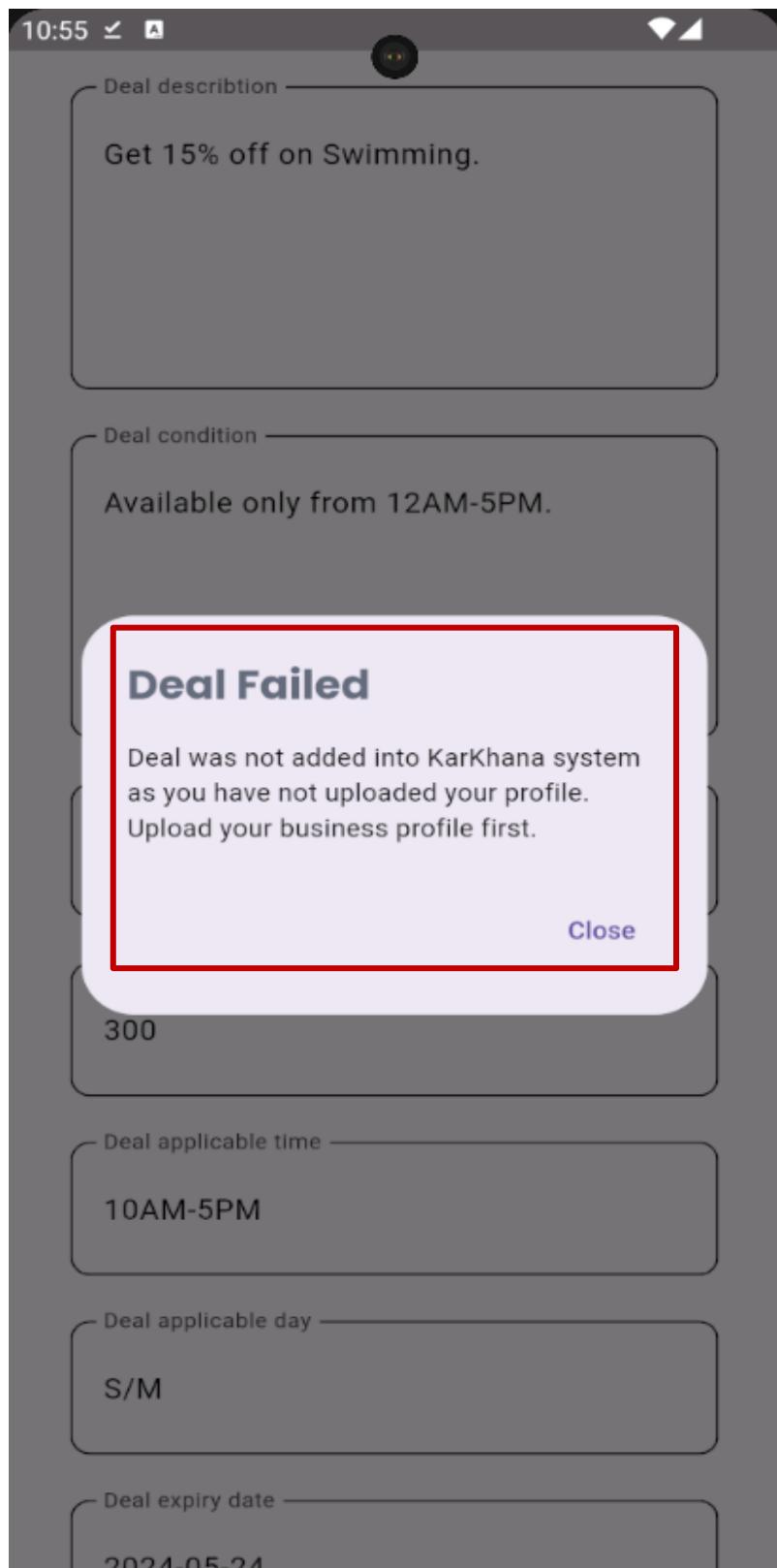


Figure 293: Add deal before business profile -2.

**4.2.34 Upload business profile**

<b>objective</b>	To upload business profile in the system.
<b>Actions</b>	Press upload profile button in homepage.
<b>Expected results</b>	Success message should be displayed.
<b>Obtained Results</b>	Success message was displayed.
<b>Conclusion</b>	Test Successful.

*Table 52: Upload business profile testing table.*

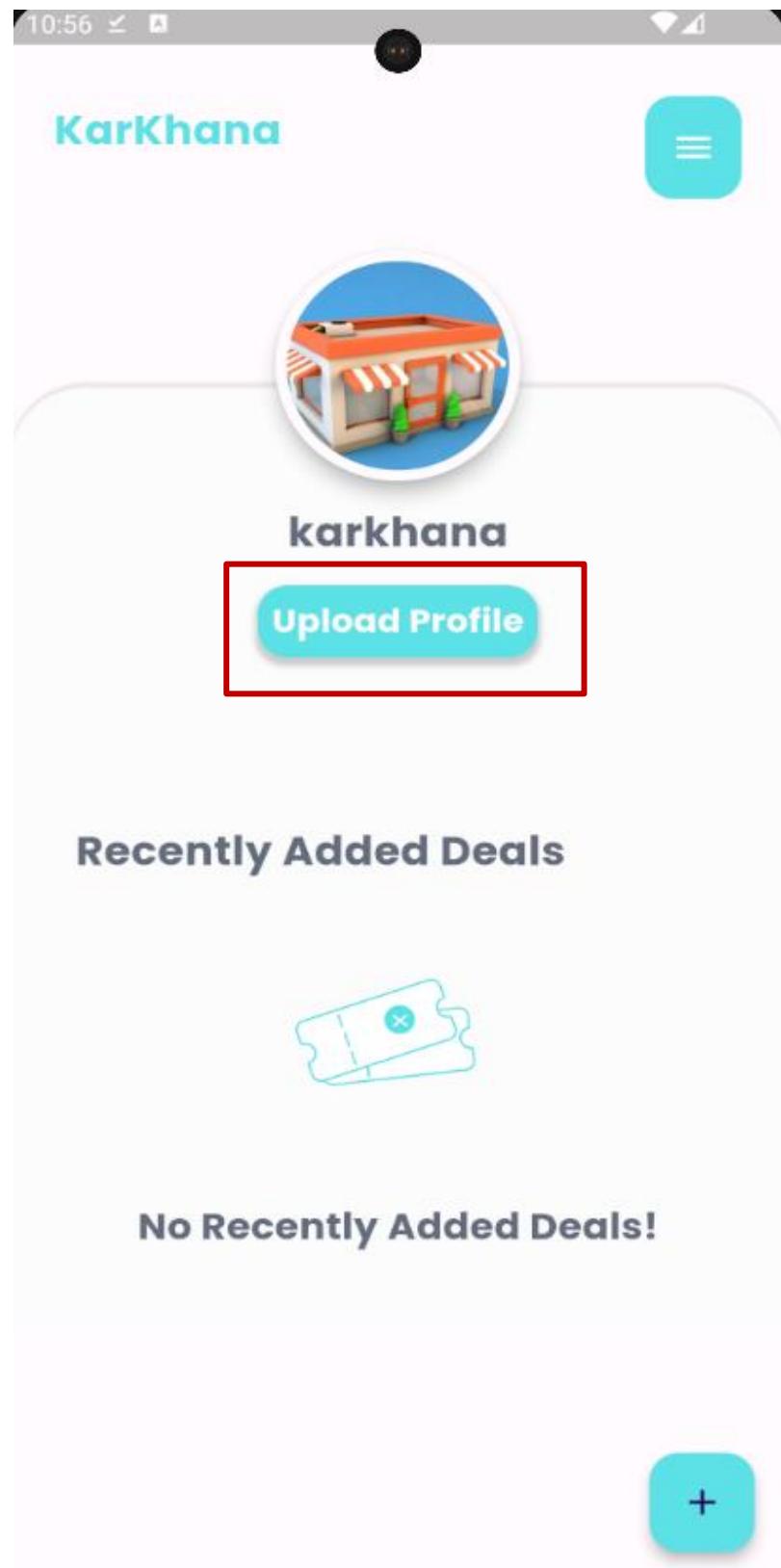


Figure 294: Upload profile testing -1.

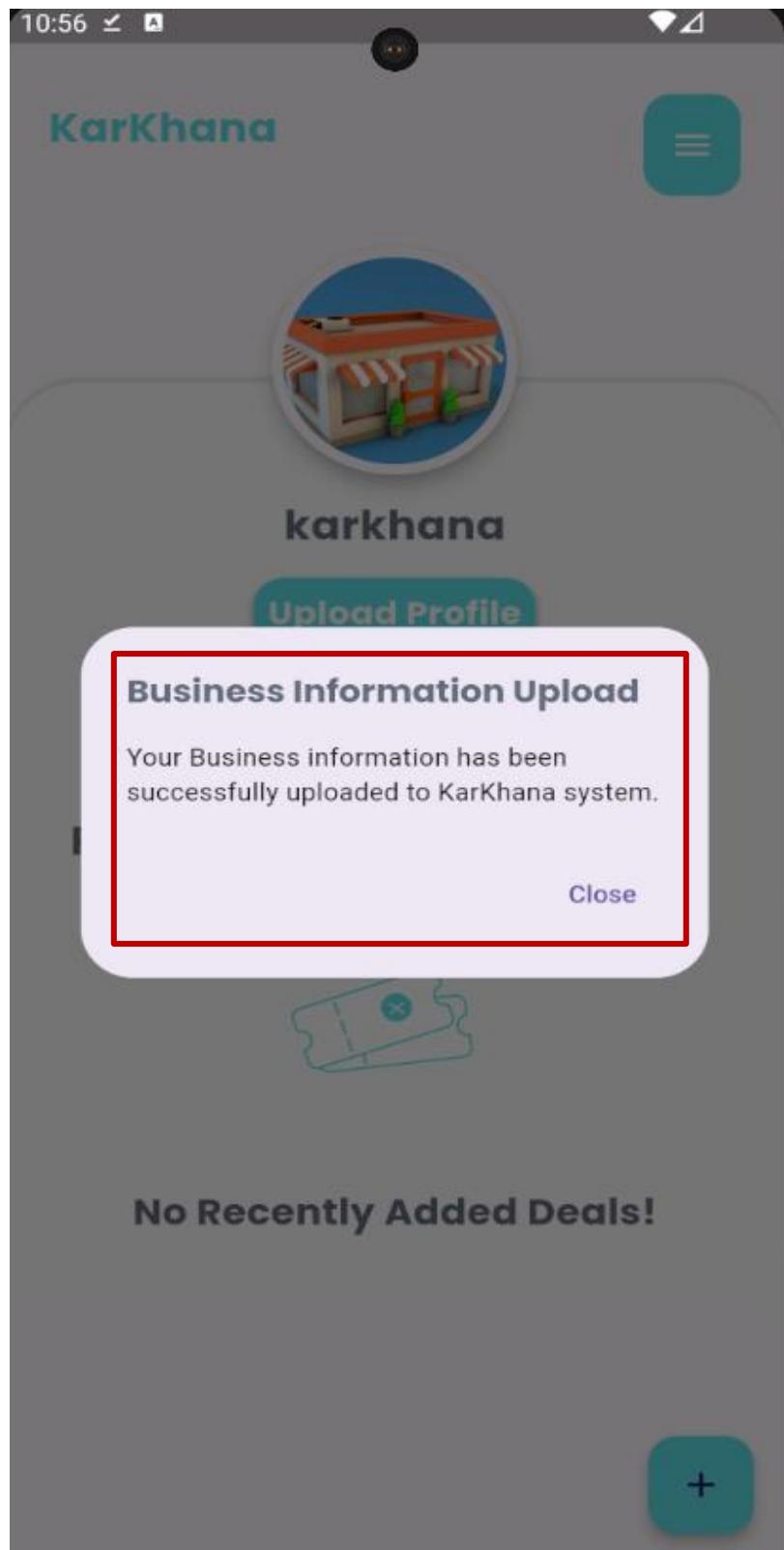


Figure 295: Upload profile testing -2.

#### 4.2.35 Add deal field validation

<b>objective</b>	To add deal with empty field.
<b>Actions</b>	Enter empty fields and press post your deal button.
<b>Input</b>	Deal title: "", Category: "", photo: "", description: "", condition: "", discount: "", price: "", applicable time: "", applicable day: "", expiry date: ""
<b>Expected results</b>	Field validation error should be displayed.
<b>Obtained Results</b>	Field validation error was displayed.
<b>Conclusion</b>	Test Successful.

Table 53: Add deal field validation testing table.

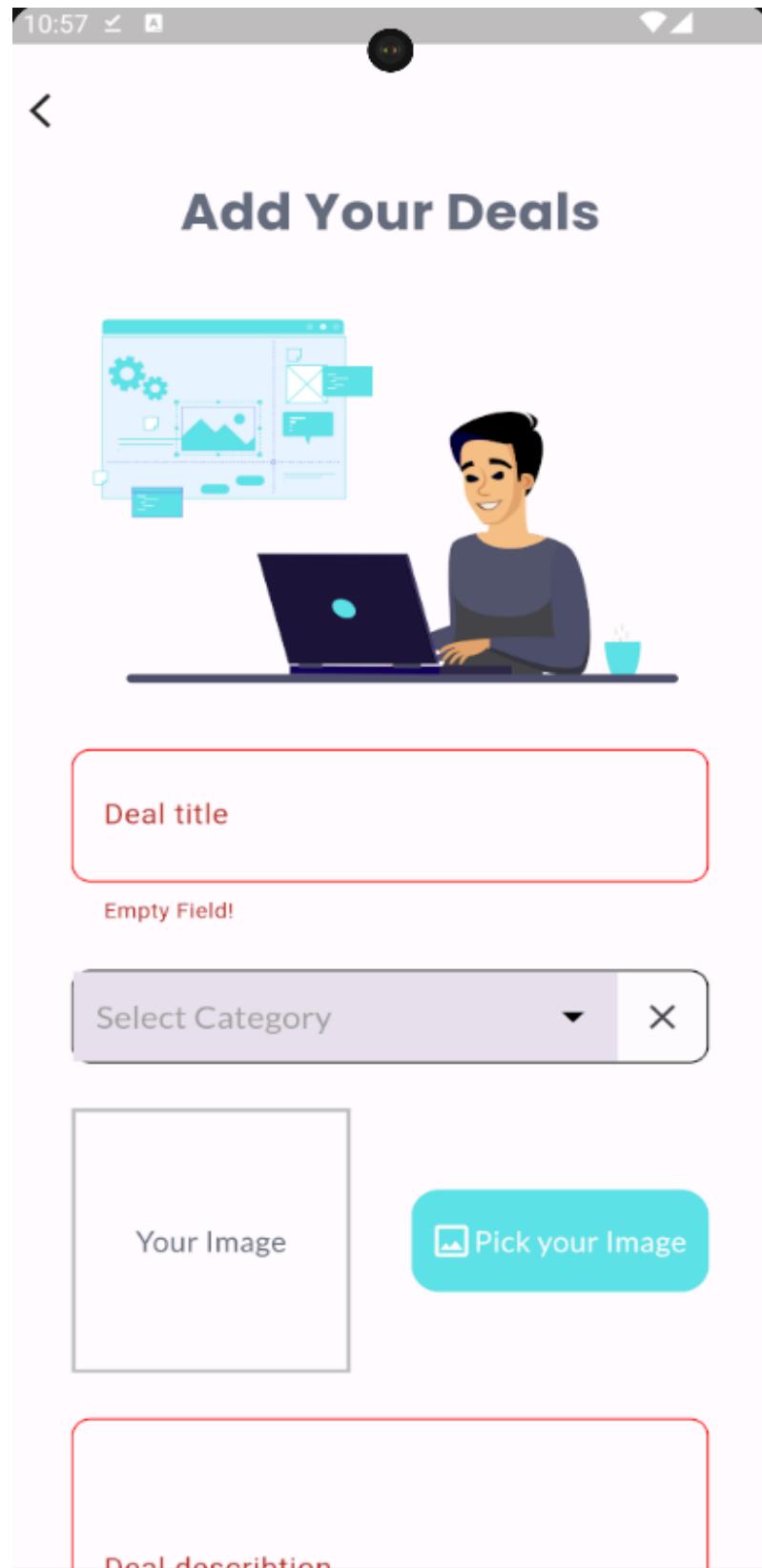


Figure 296: Add Deals Validation Testing -1.

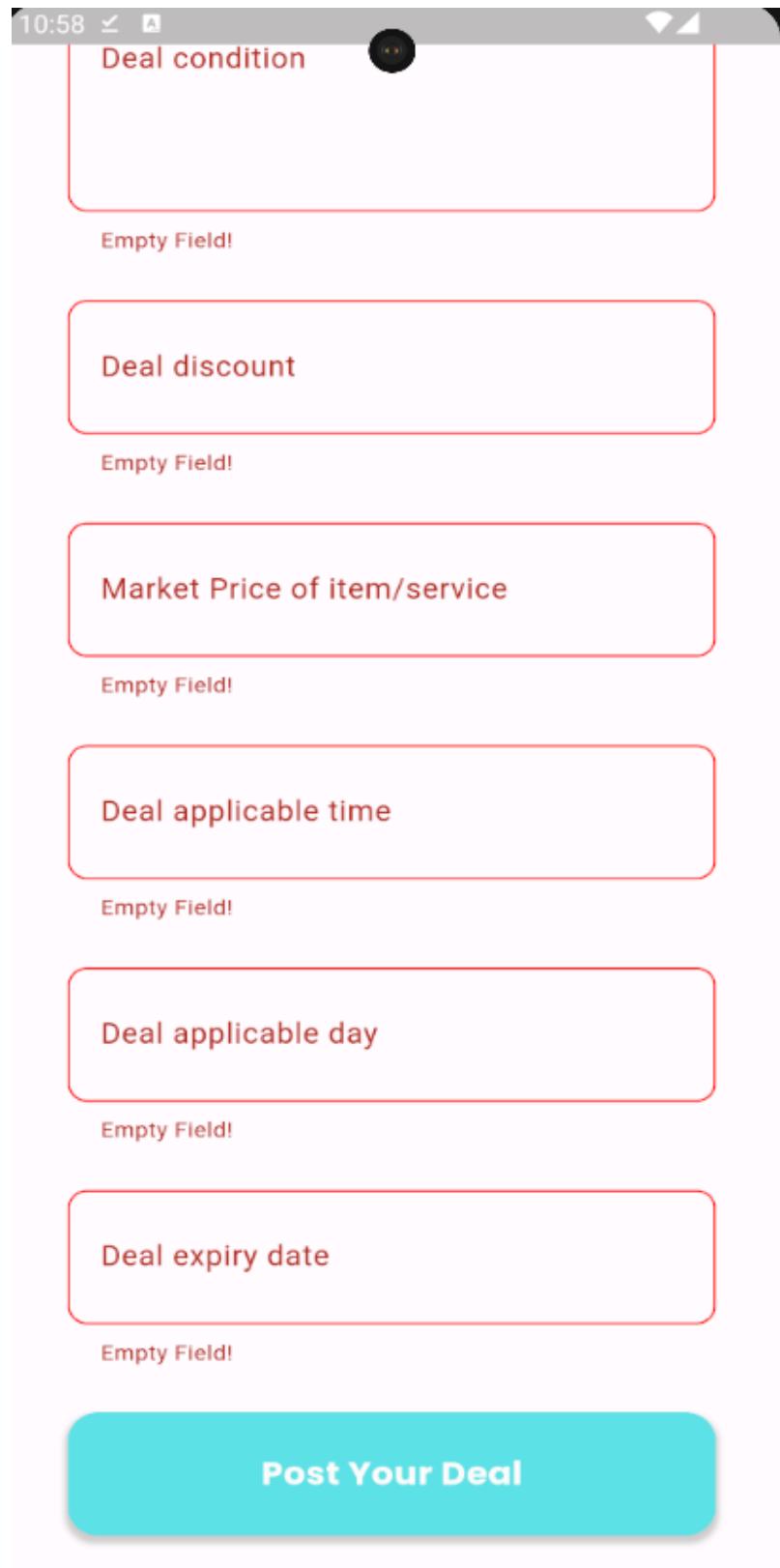


Figure 297: Add Deals Validation Testing -2.

#### 4.2.36 Add deal success

<b>objective</b>	To add deal to KarKhana system.
<b>Actions</b>	Enter deal details and press post deal button.
<b>Input</b>	Deal title: "30% off in new year event", Category: "Event Deals", description: "New Year's Eve, also known as Old Year's Day or Saint Sylvester's Day in many countries.", Condition: "Use before expire.", discount: "30", Price: "300", applicable time: "12PM-5PM", applicable day: "WED", expiry date: "2024-05-31",
<b>Expected results</b>	Success message should be displayed, and deal should be displayed in customer side of the application.
<b>Obtained Results</b>	Success message was displayed, and deal was displayed in customer side of the application.
<b>Conclusion</b>	Test Successful.

Table 54: Add deal testing table.

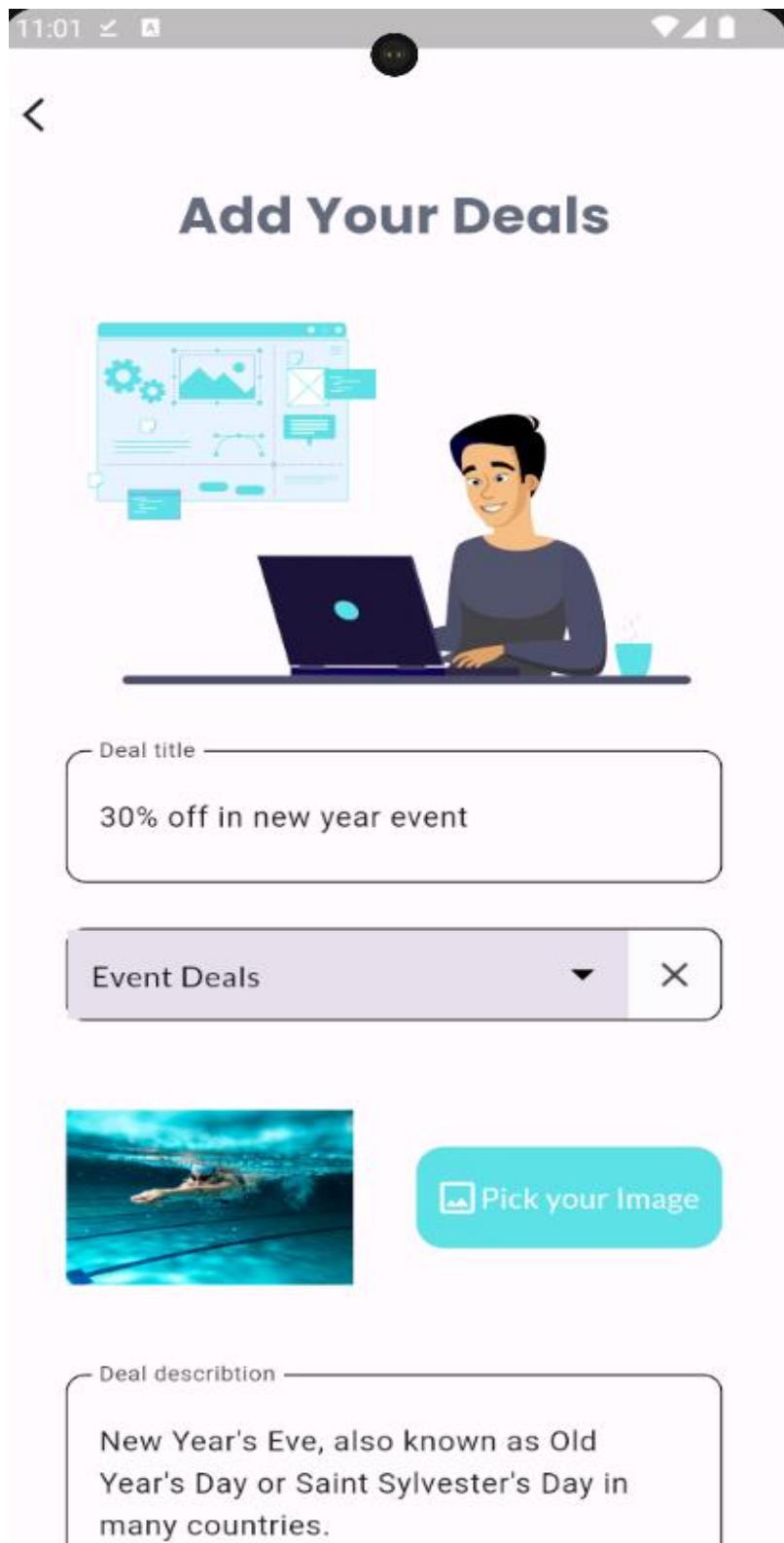


Figure 298: Add Deals Testing -1.

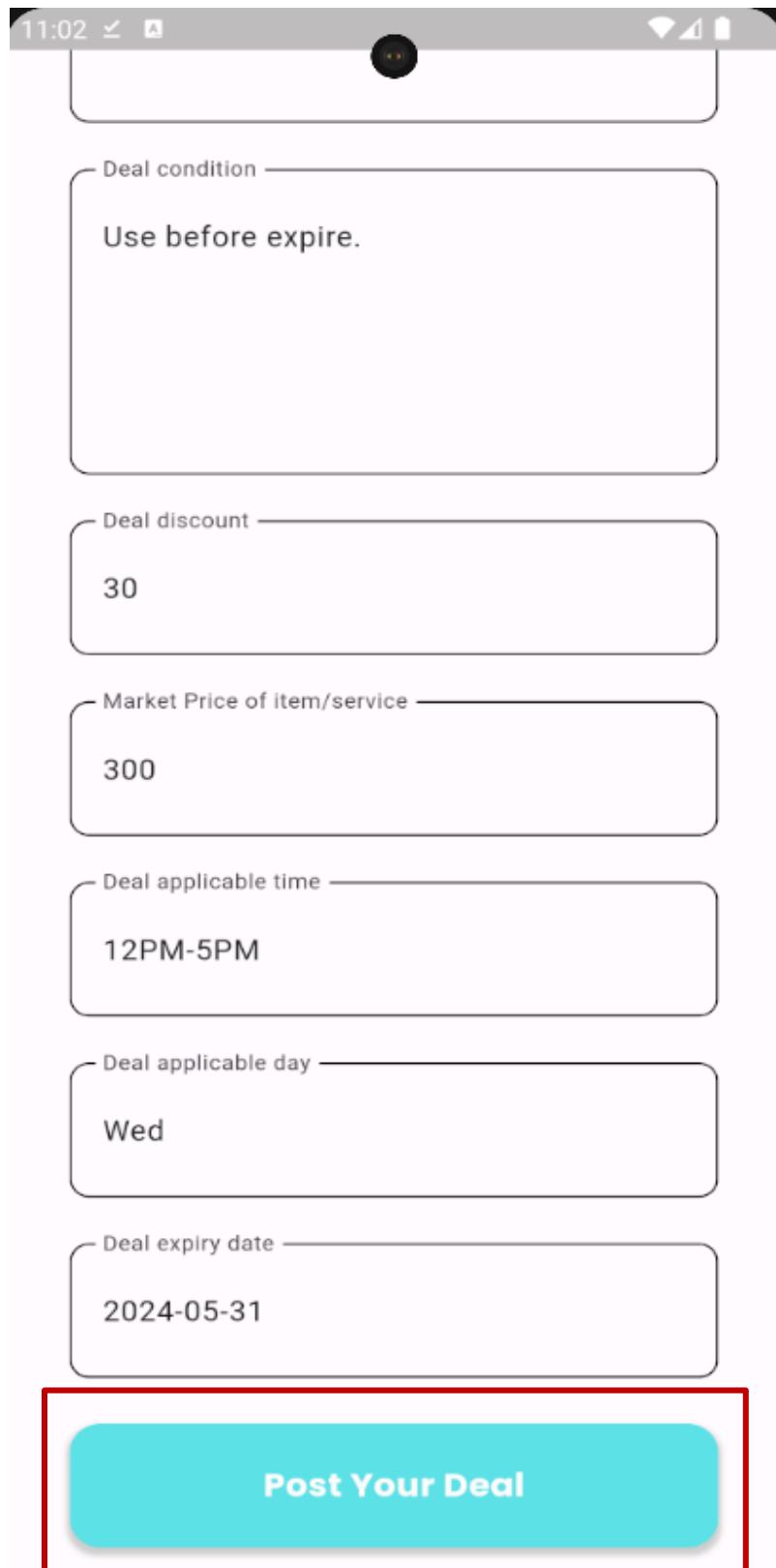


Figure 299: Add Deals Testing -2.

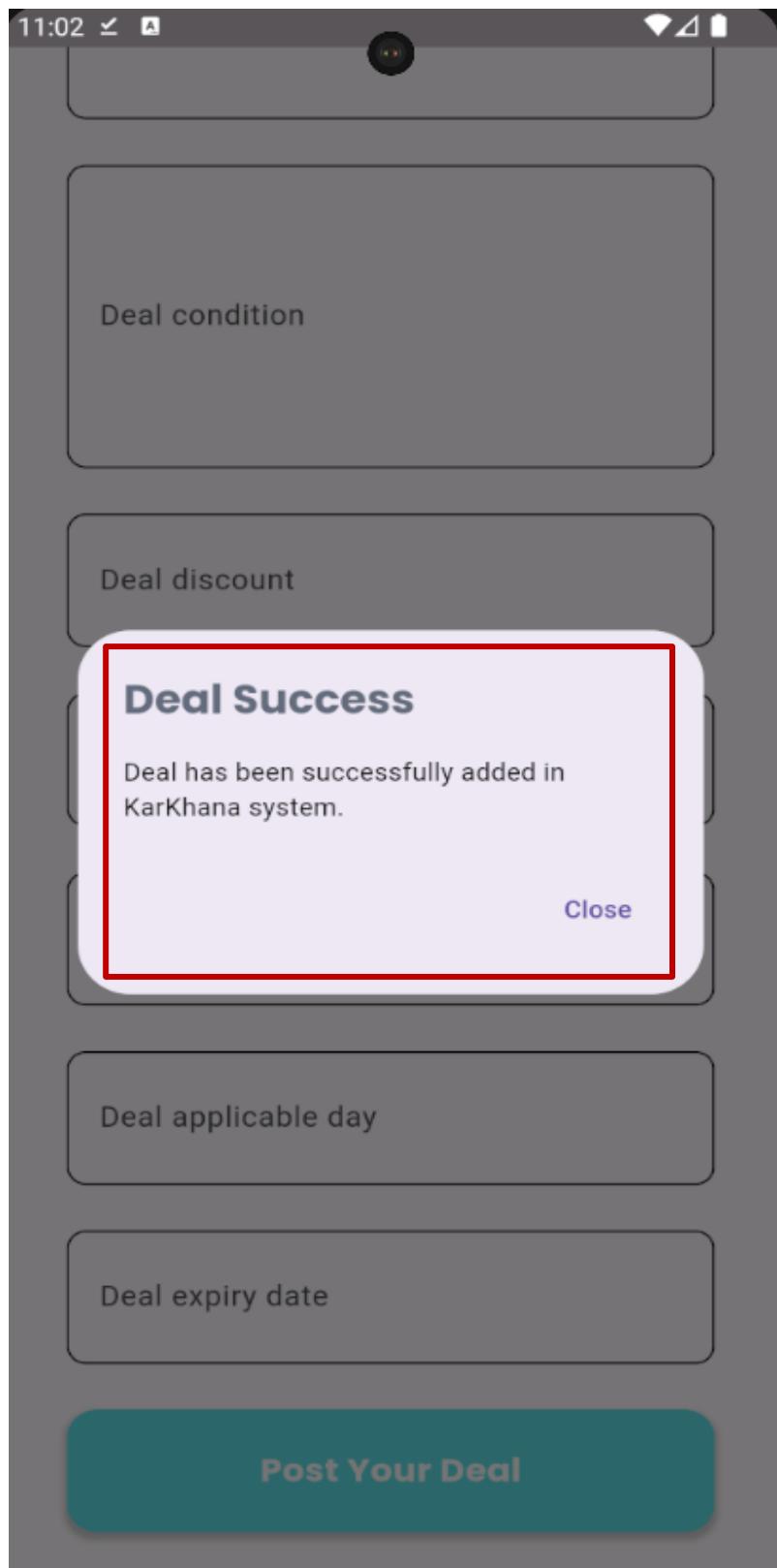


Figure 300: Add Deals Testing -3.

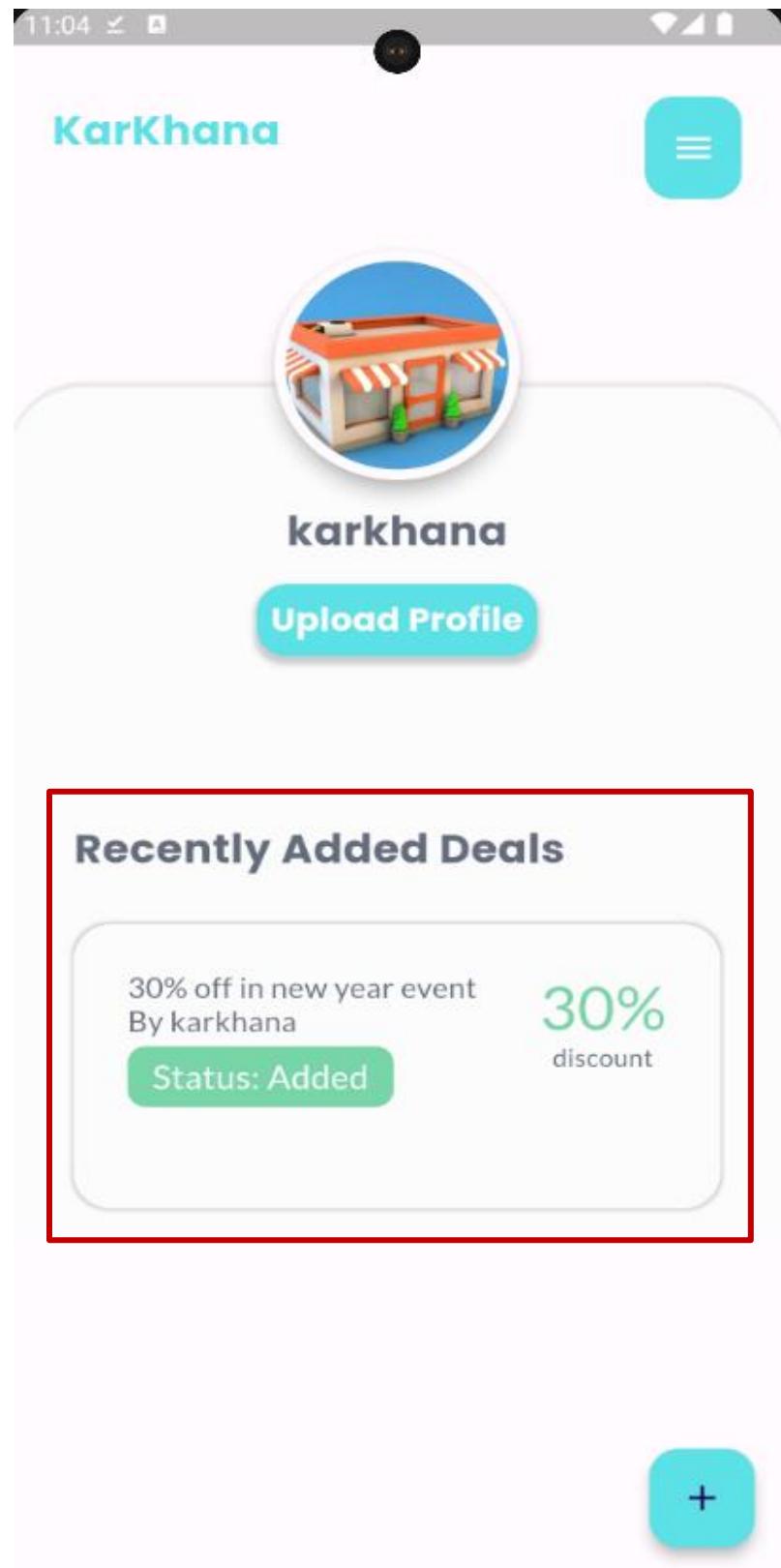


Figure 301: Add Deals Testing -4.

#### 4.2.37 Confirm coupon success.

objective	To confirm active coupon.
<b>Actions</b>	Select coupon to redeem then press confirm button. After confirming the coupon, check the same coupon's status. (Also remember the coupon id throughout the process to know which coupon is being used.)
<b>Expected results</b>	Success message should be displayed and that coupon should now change its status to "used" and should be displayed in previous coupon section.
<b>Obtained Results</b>	Success message was displayed and that coupon has changed its status to "used" and was also displayed in previous coupon section.
<b>Conclusion</b>	Test Successful.

Table 55: Confirm coupon testing table.

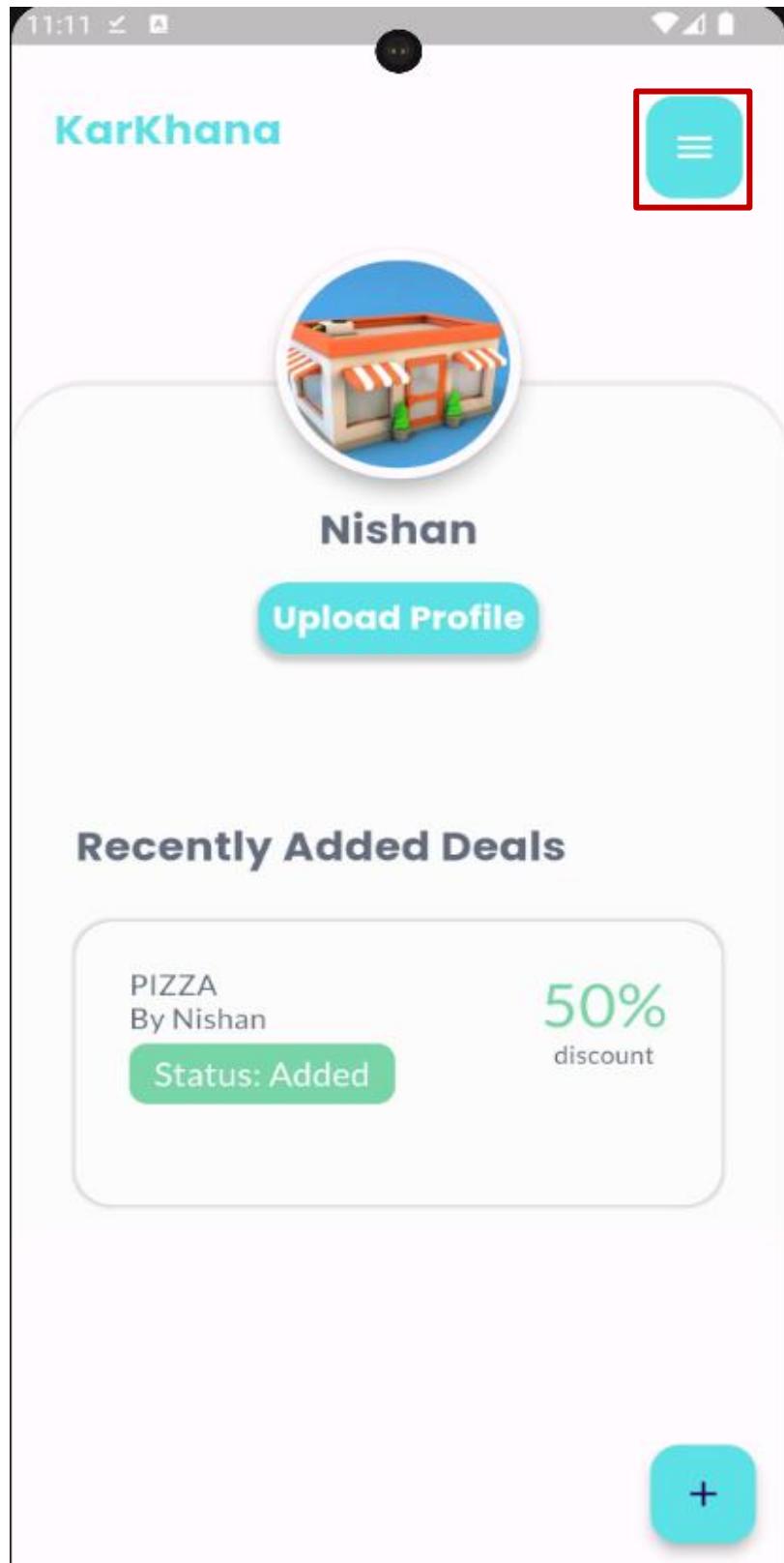


Figure 302: Confirm Coupon Testing -1.

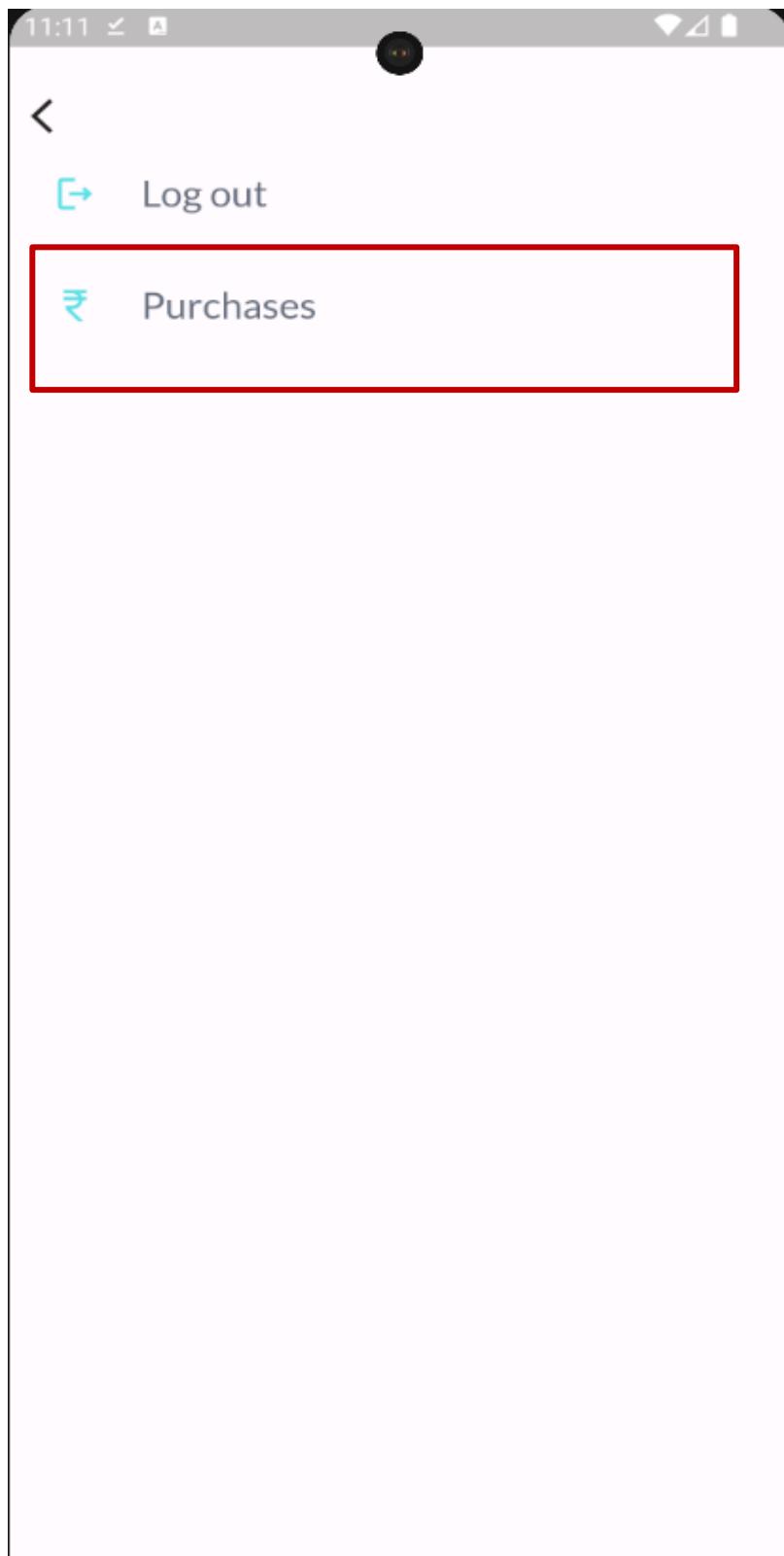


Figure 303: Confirm Coupon Testing -2.

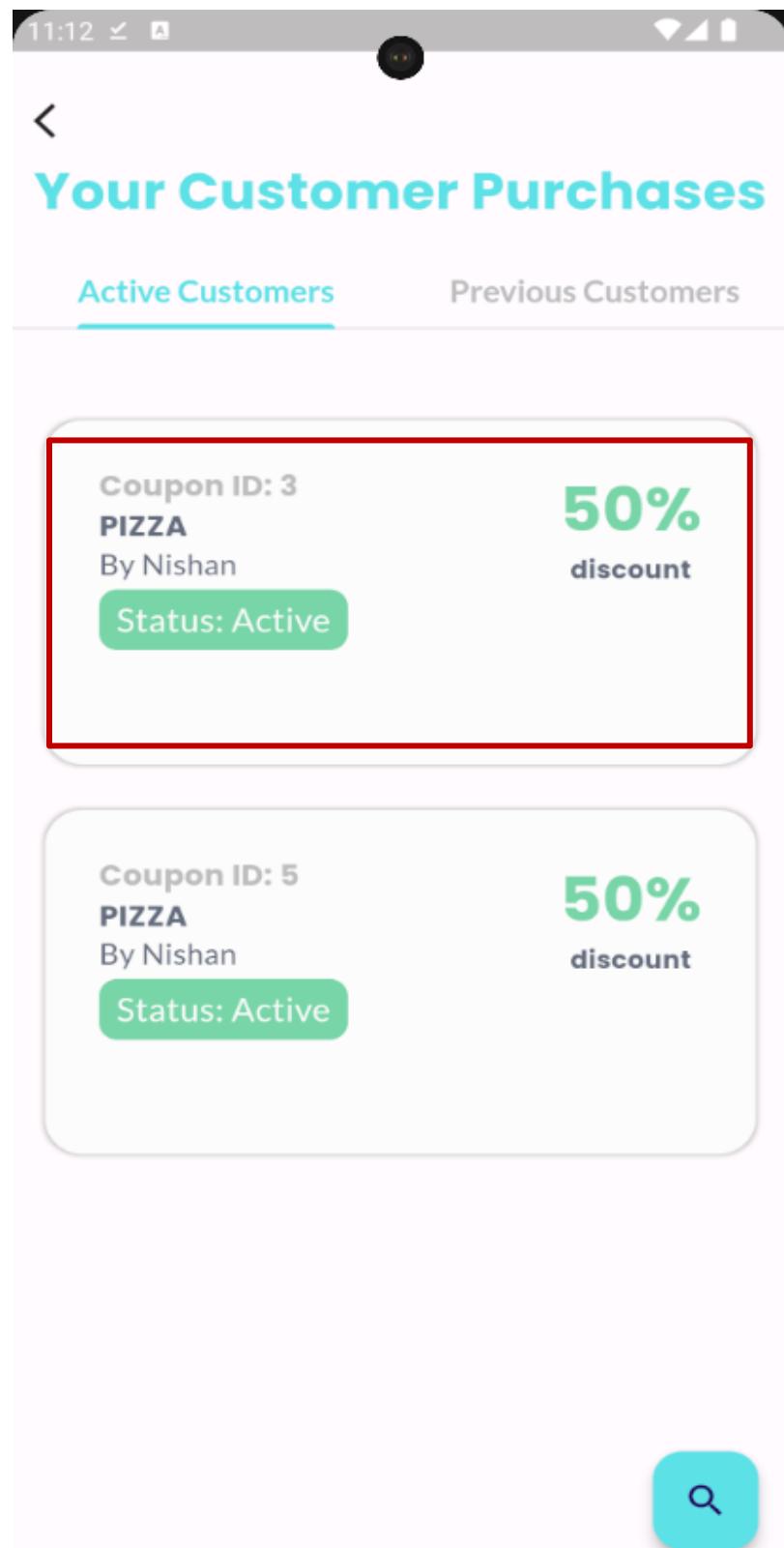


Figure 304: Confirm Coupon Testing -3.

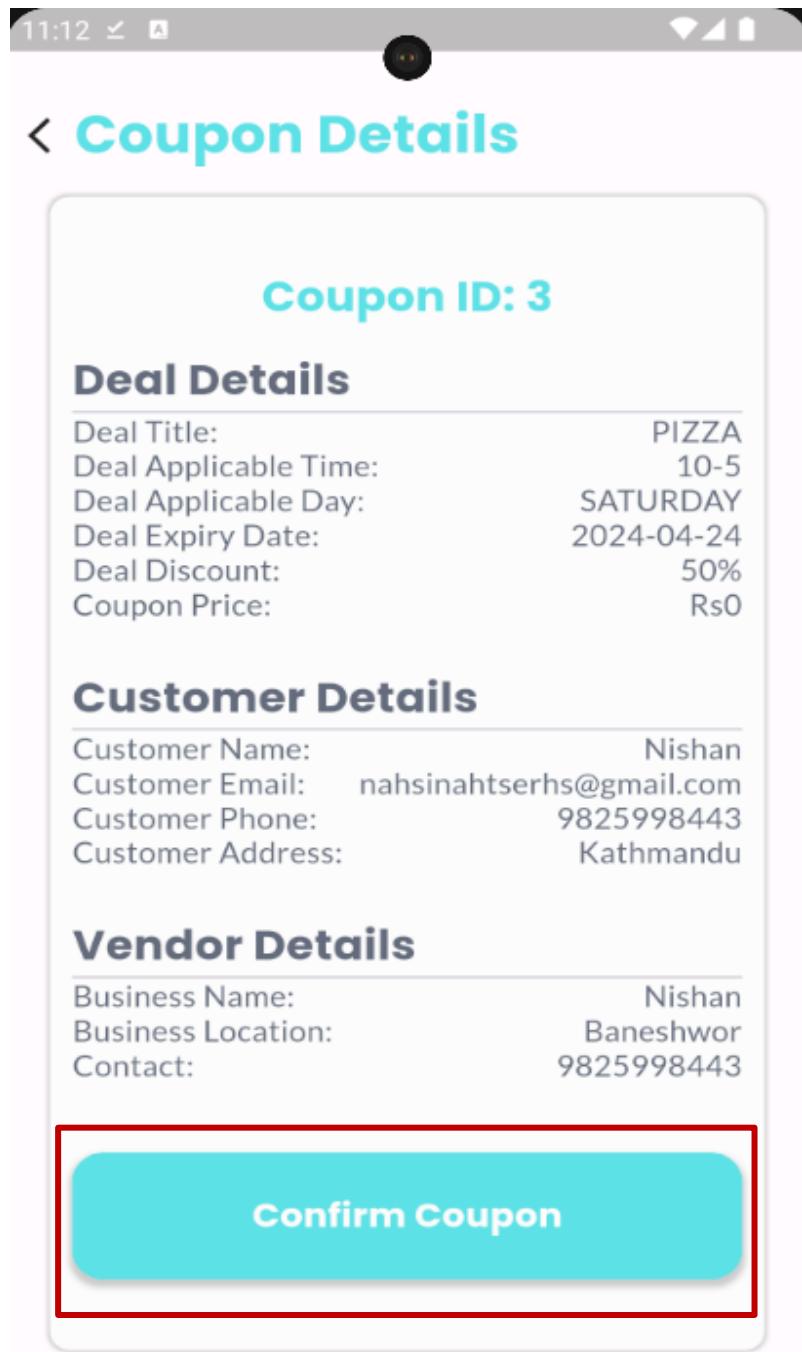


Figure 305: Confirm Coupon Testing -4.

**4.2.38 Display active coupon**

<b>objective</b>		<b>To display active coupons</b>
<b>Actions</b>		To navigate to active coupon section of customer purchases page
<b>Expected results</b>		Coupons with active status should be displayed.
<b>Obtained Results</b>		Coupons with active status was displayed.
<b>Conclusion</b>		Test Successful.

*Table 56: Display active coupons testing table.*

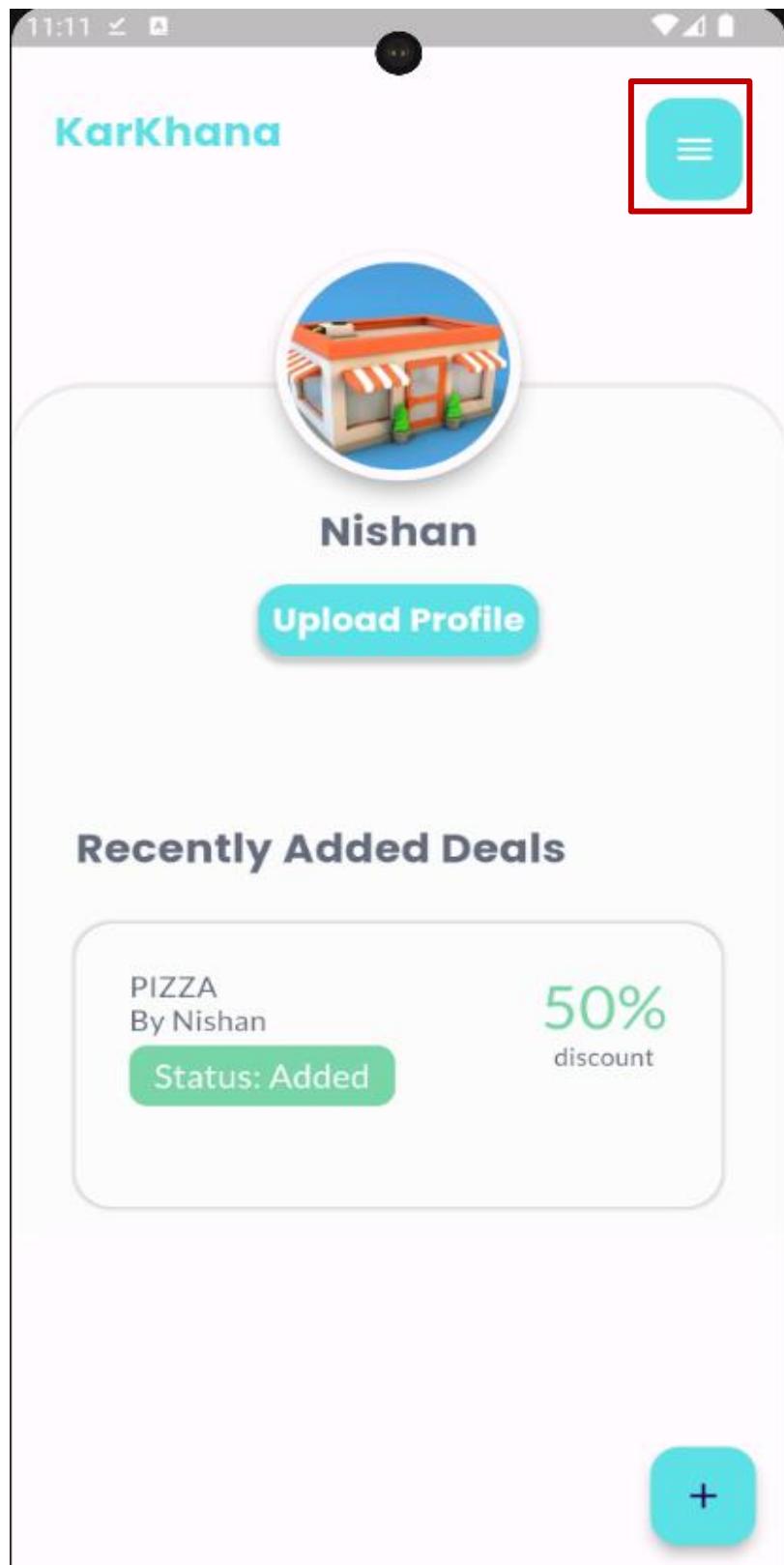


Figure 306: Display active coupons testing -1.

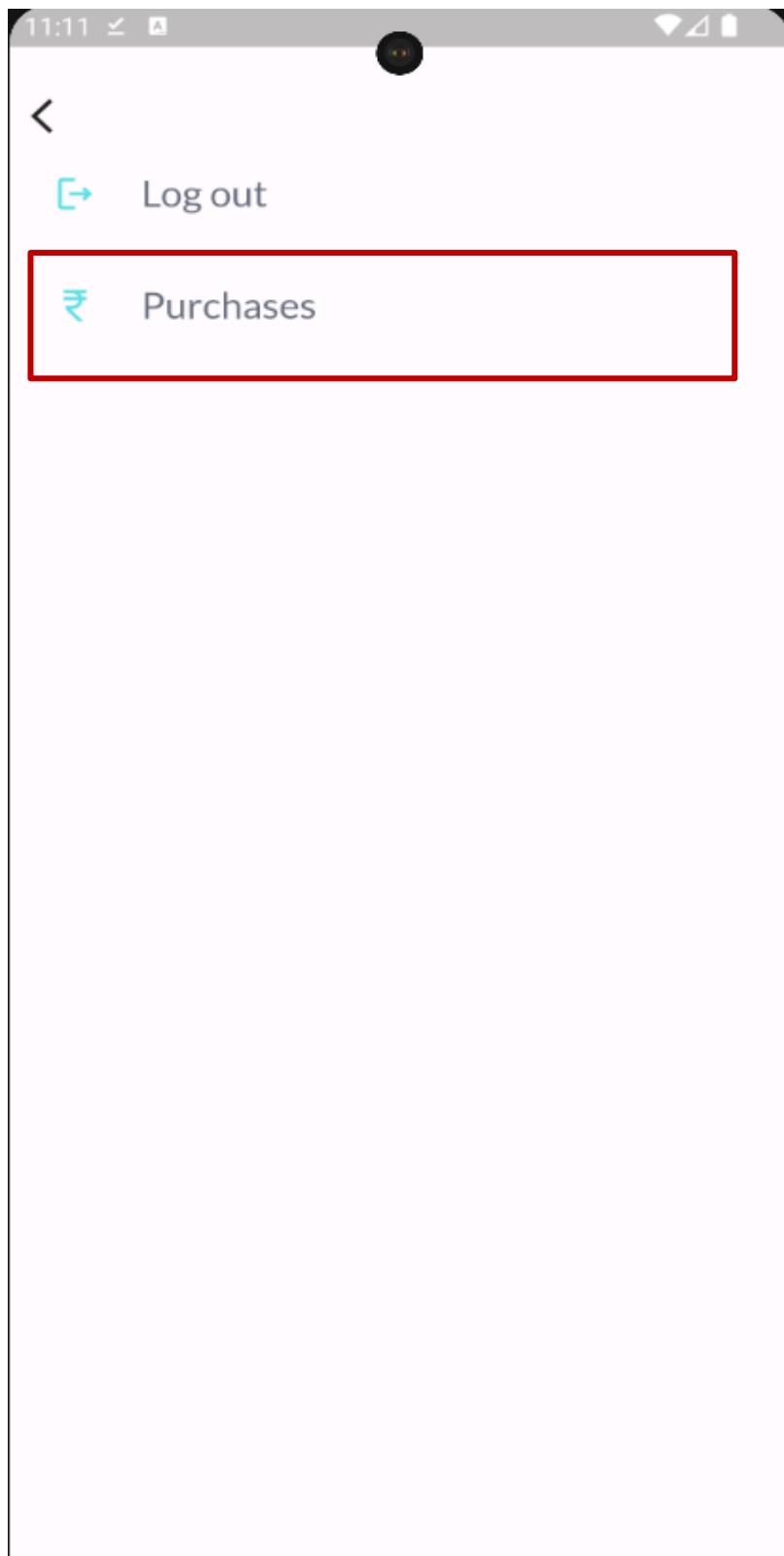


Figure 307: Display active coupons testing -2.

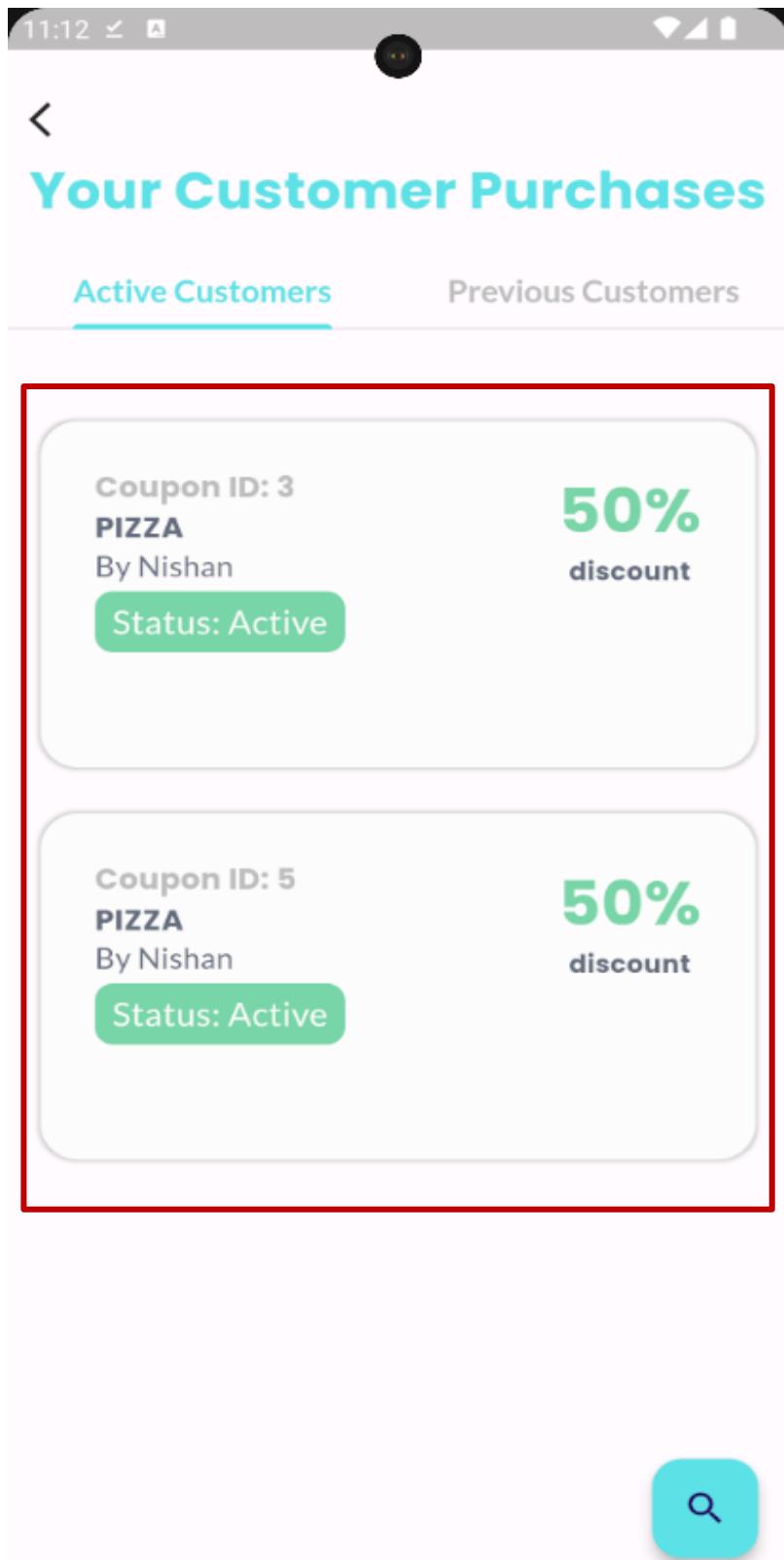


Figure 308: Display active coupons testing -4.

#### 4.2.39 Display previous coupon

objective	To display previous coupons
<b>Actions</b>	To navigate to previous coupon section of customer purchases page
<b>Expected results</b>	Coupons with previous status should be displayed.
<b>Obtained Results</b>	Coupons with previous status was displayed.
<b>Conclusion</b>	Test Successful.

Table 57: Display previous coupons testing table.

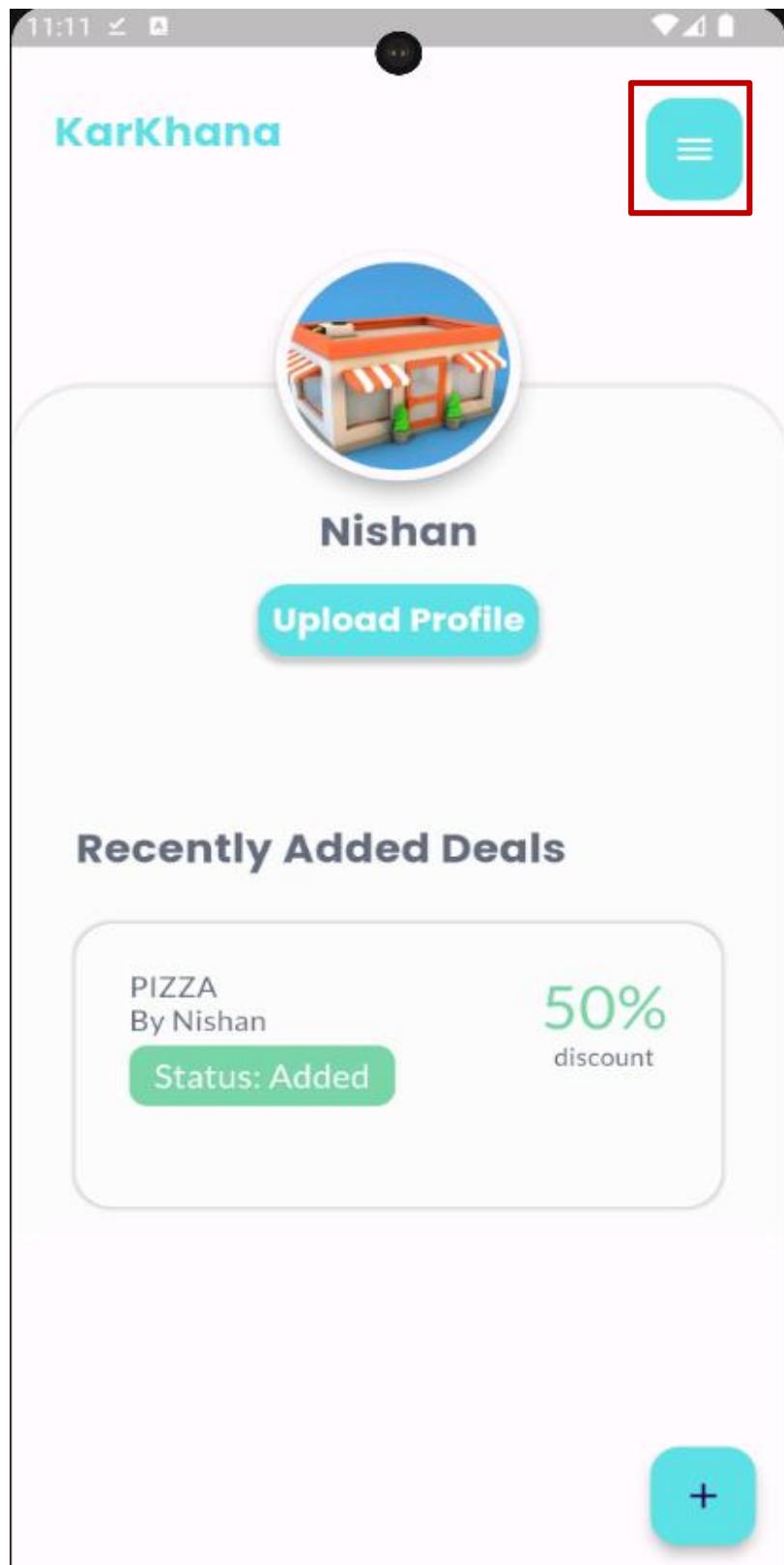


Figure 309: Display previous coupons testing -1.

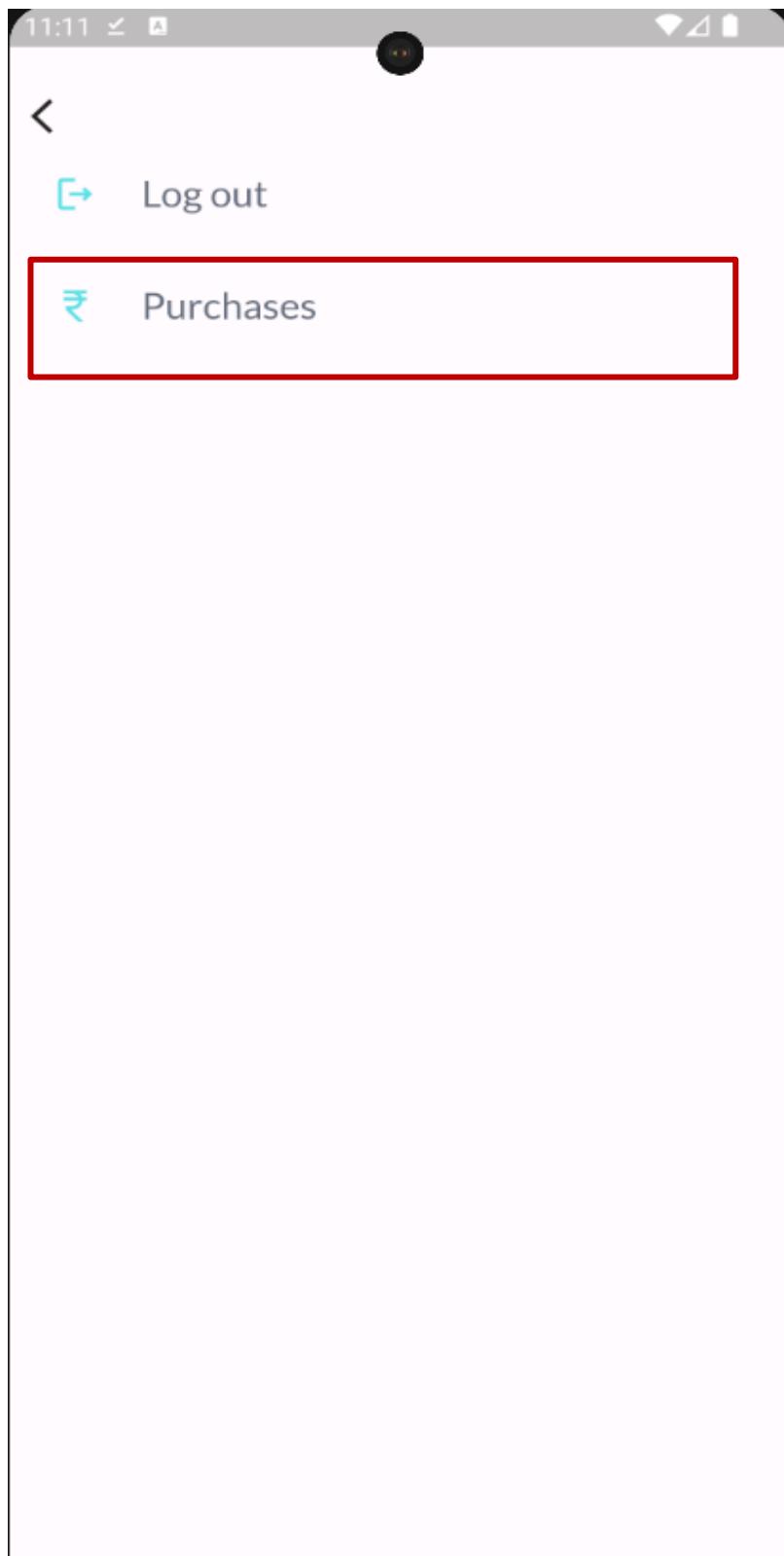


Figure 310: Display previous coupons testing -2.

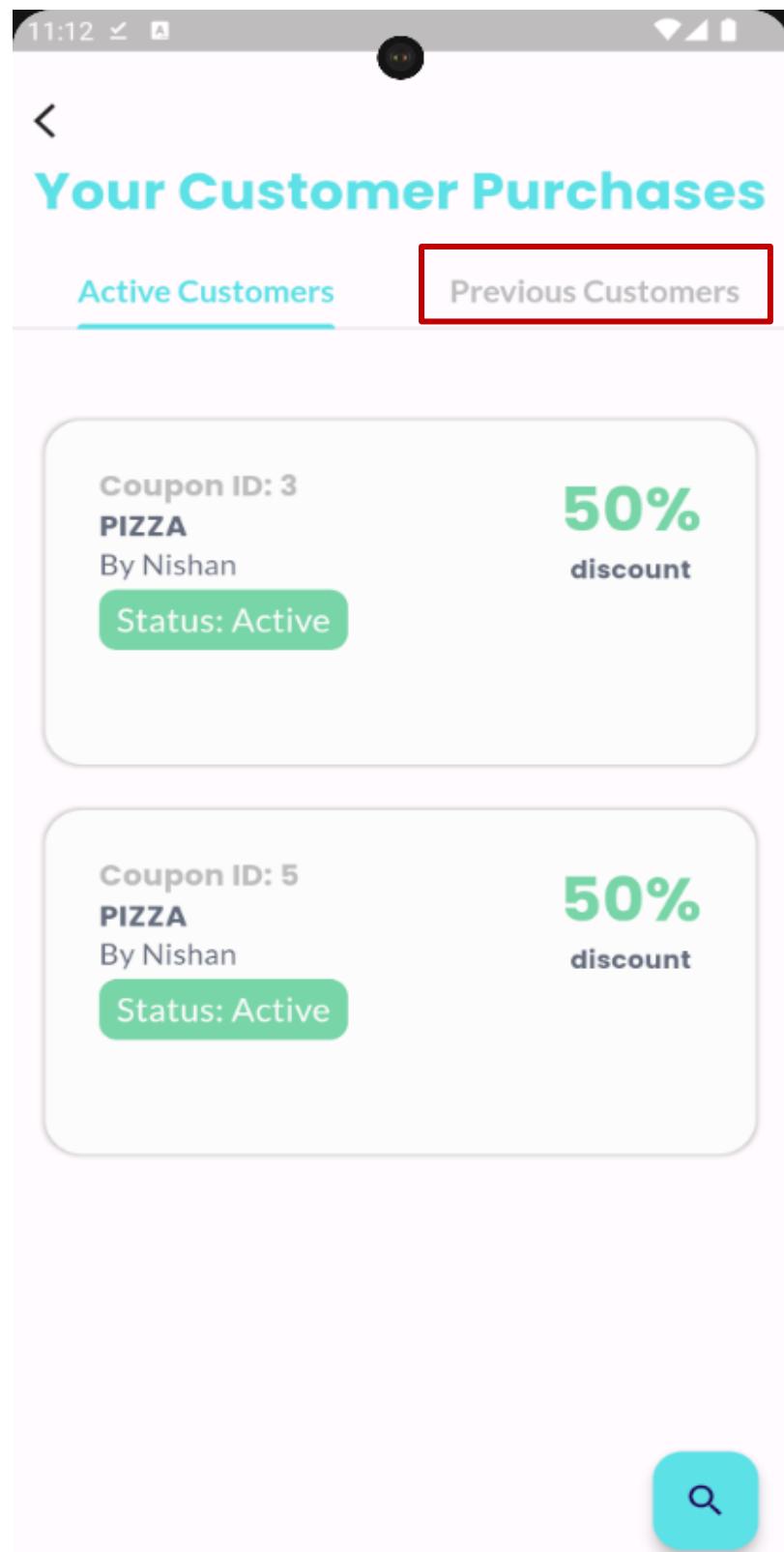


Figure 311: Display previous coupons testing -3.

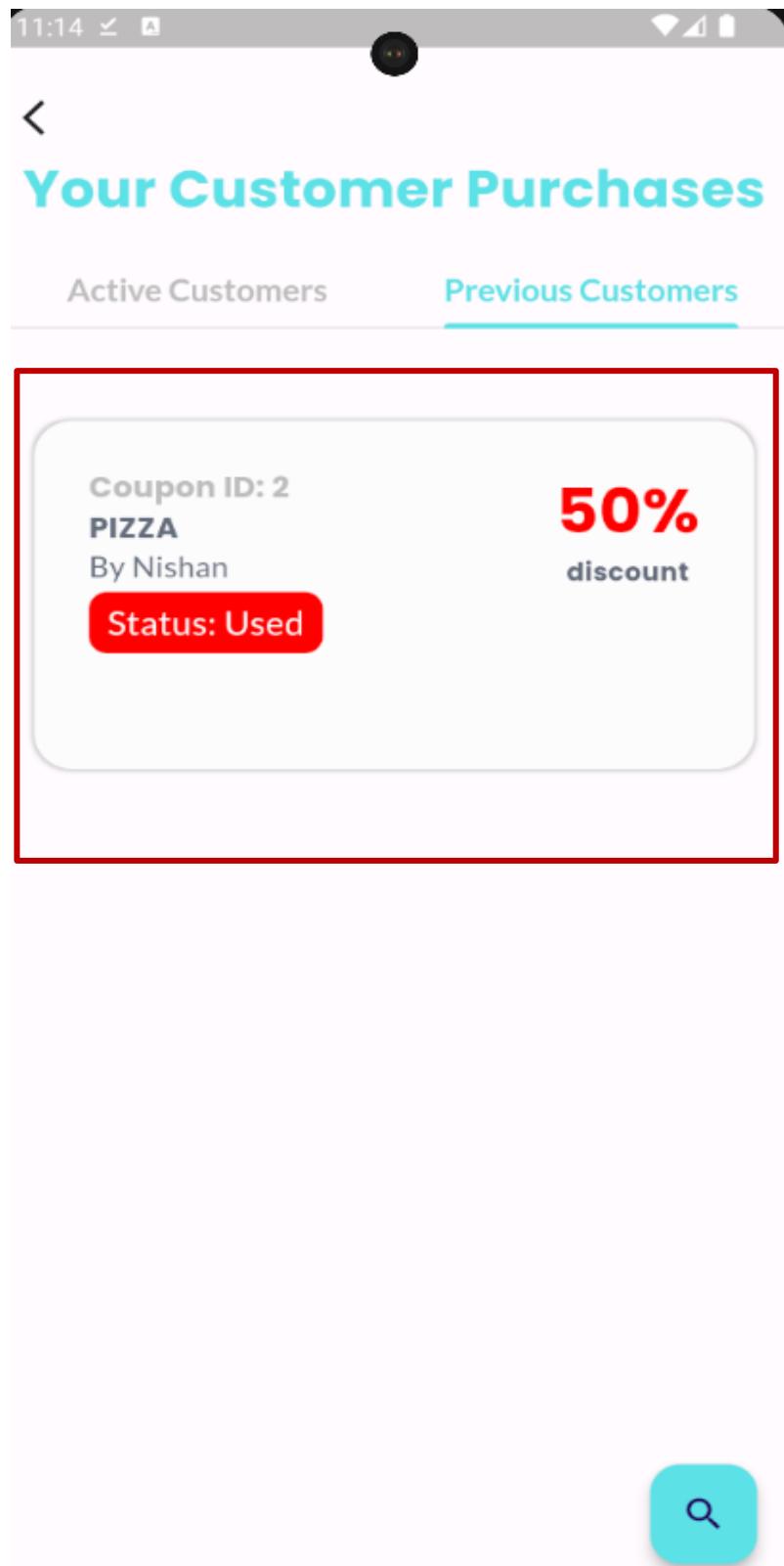


Figure 312: Display previous coupons testing -4.

#### 4.2.40 Recently added deals

objective	To display recently added deal
<b>Actions</b>	Add deal and navigate to homepage.
<b>Expected results</b>	Recently added deal should be displayed.
<b>Obtained Results</b>	Recently added deal was displayed.
<b>Conclusion</b>	Test Successful.

Table 58: Recently added deal testing table.

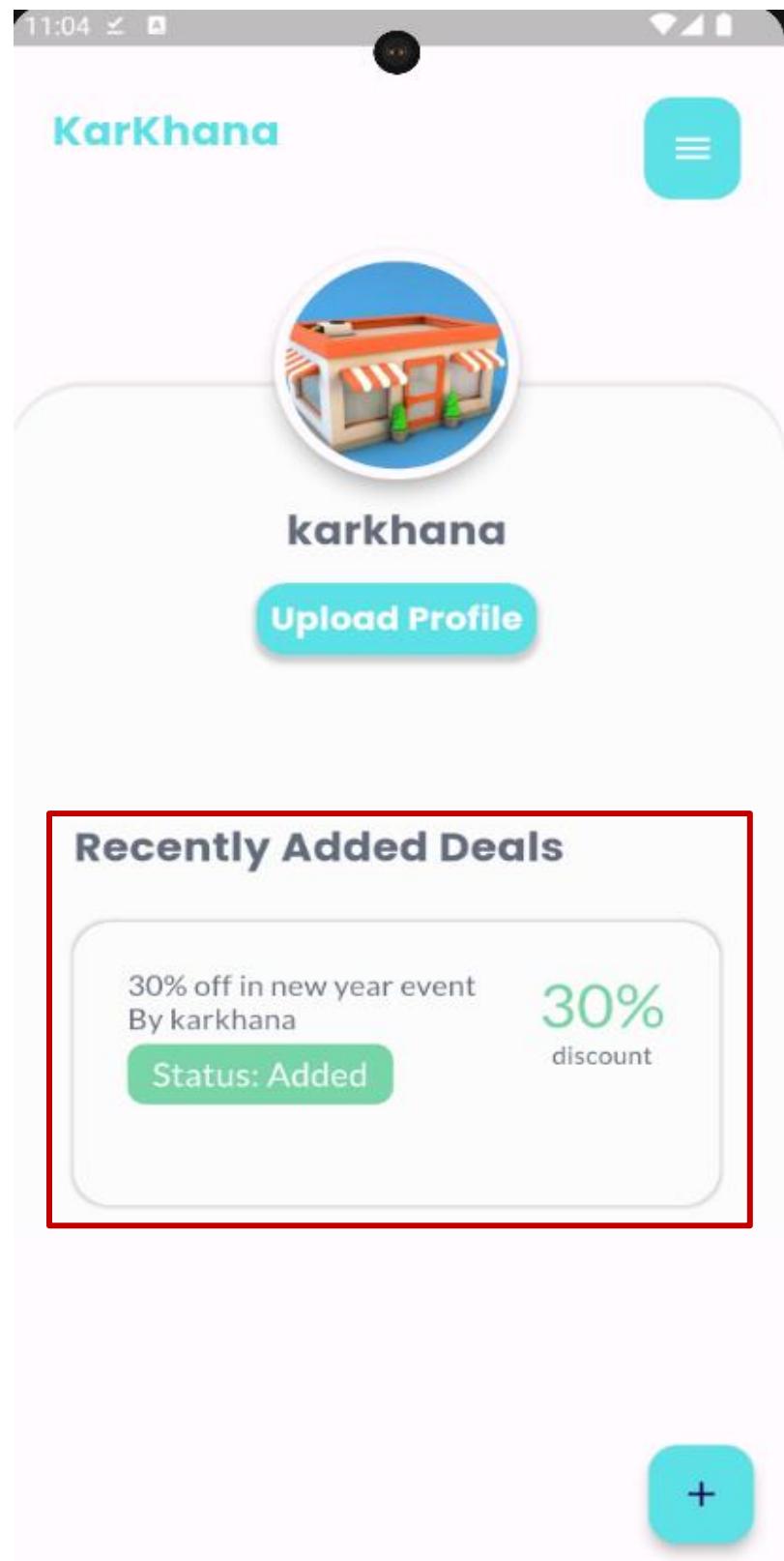


Figure 313 Recently added deal testing.

#### 4.2.41 Logout

<b>objective</b>	To logout of the system.
<b>Actions</b>	Press on logout button and confirm logout.
<b>Expected results</b>	Logout success message should be displayed, and user should be navigated to welcome screen.
<b>Obtained Results</b>	Logout success message was displayed, and user was navigated to welcome screen.
<b>Conclusion</b>	Test Successful.

Table 59: Logout testing table.

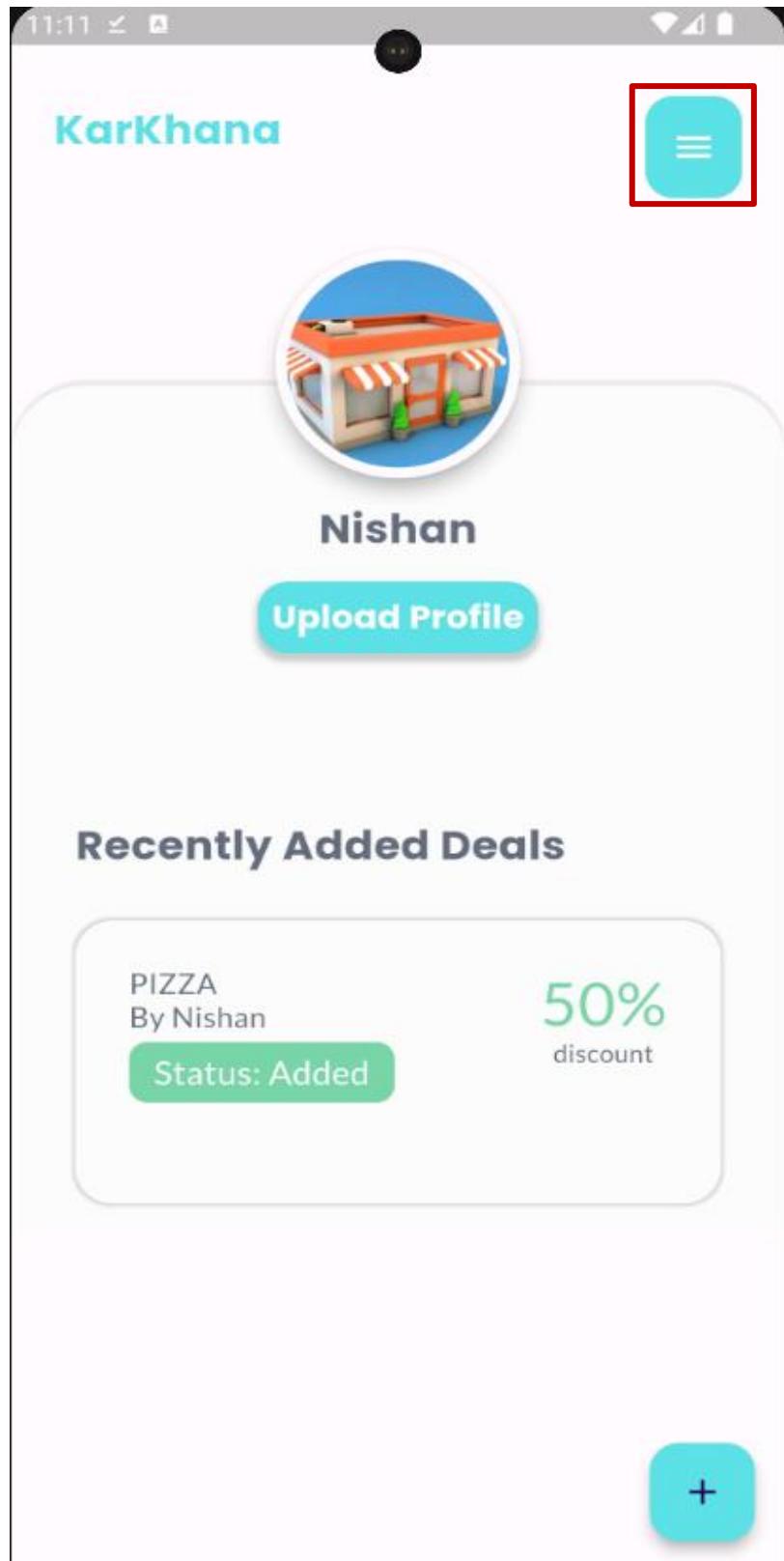


Figure 314: Logout testing -1.

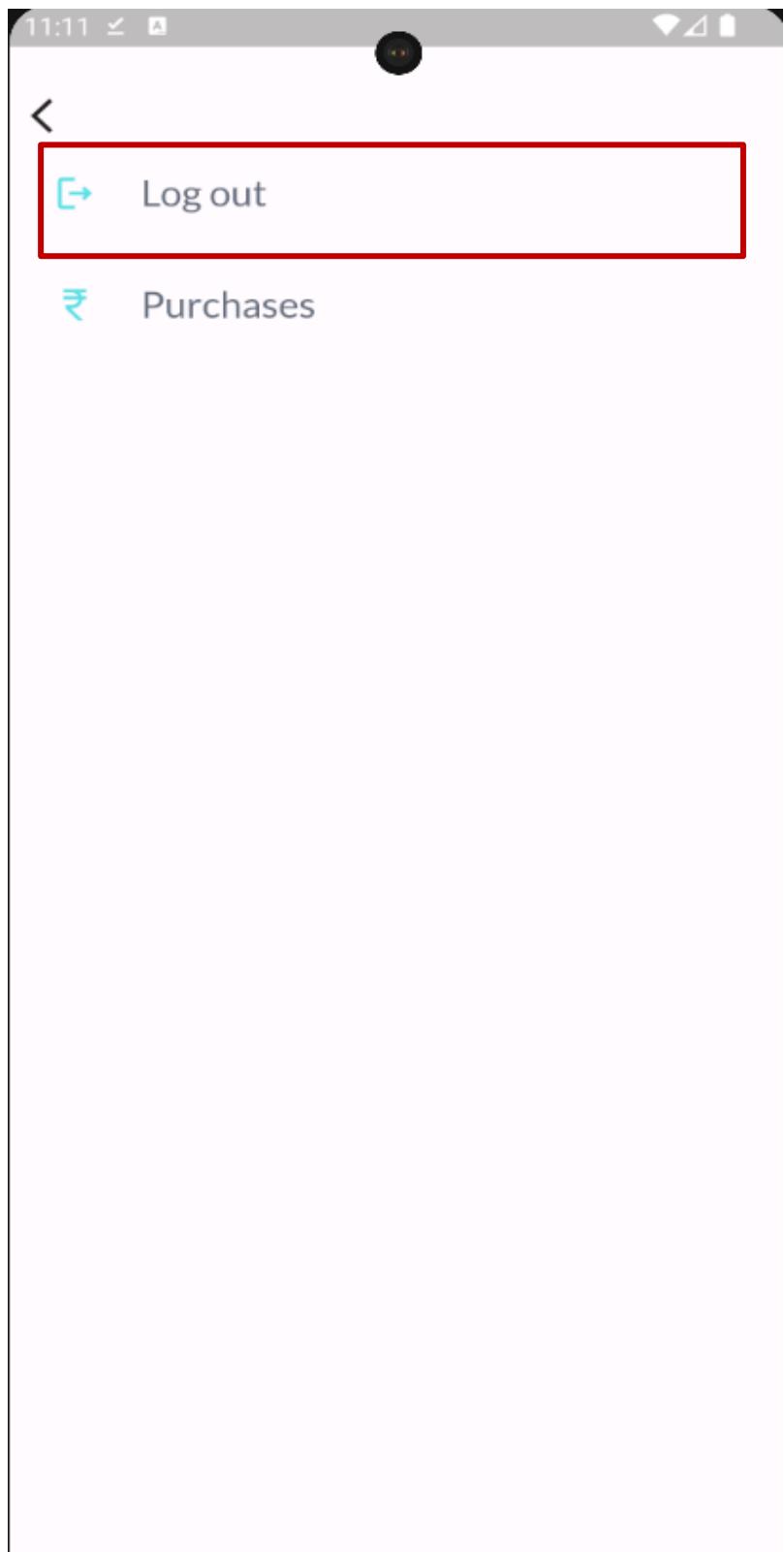


Figure 315: Logout testing -2.

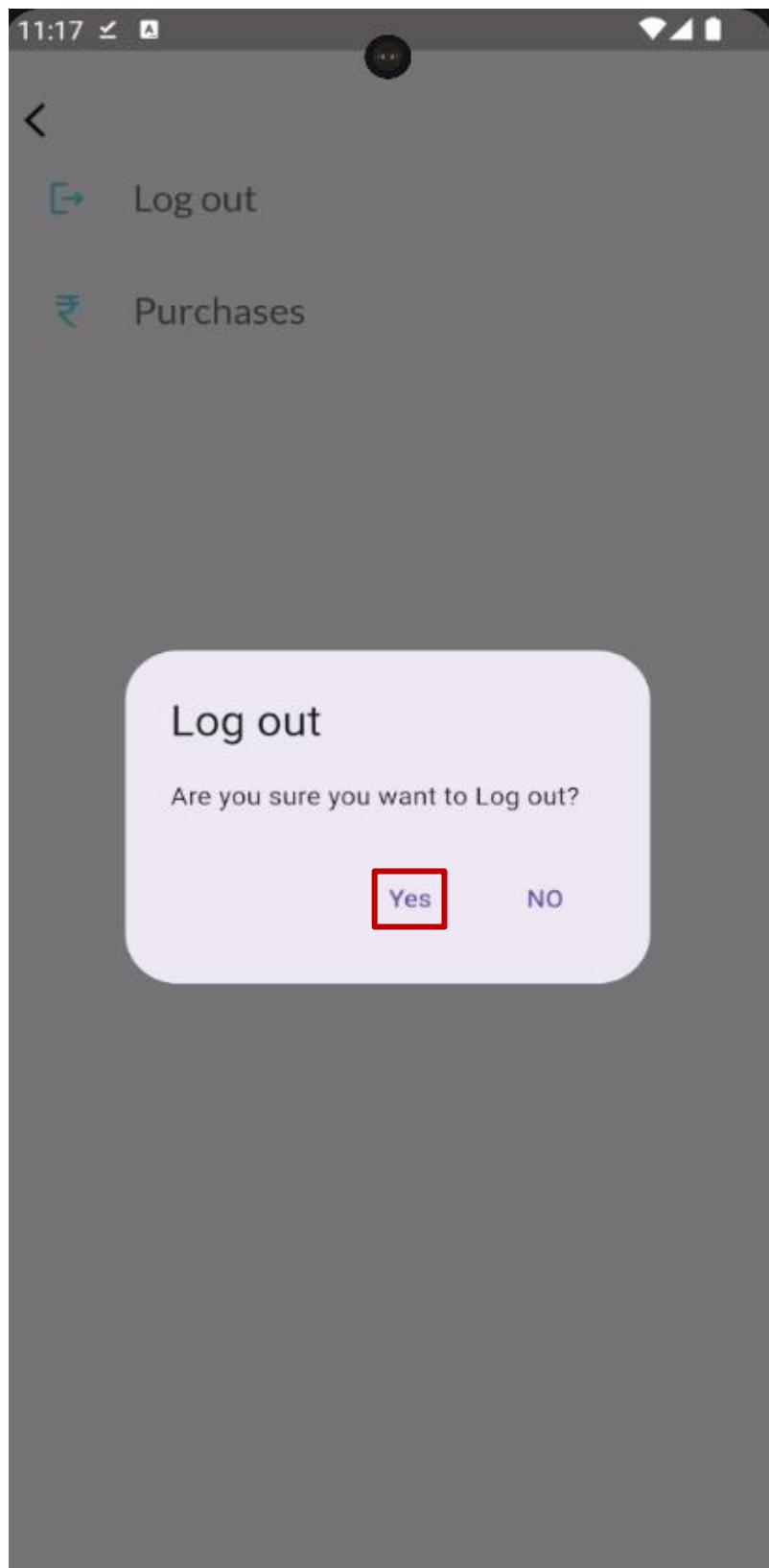


Figure 316: Logout testing -3.

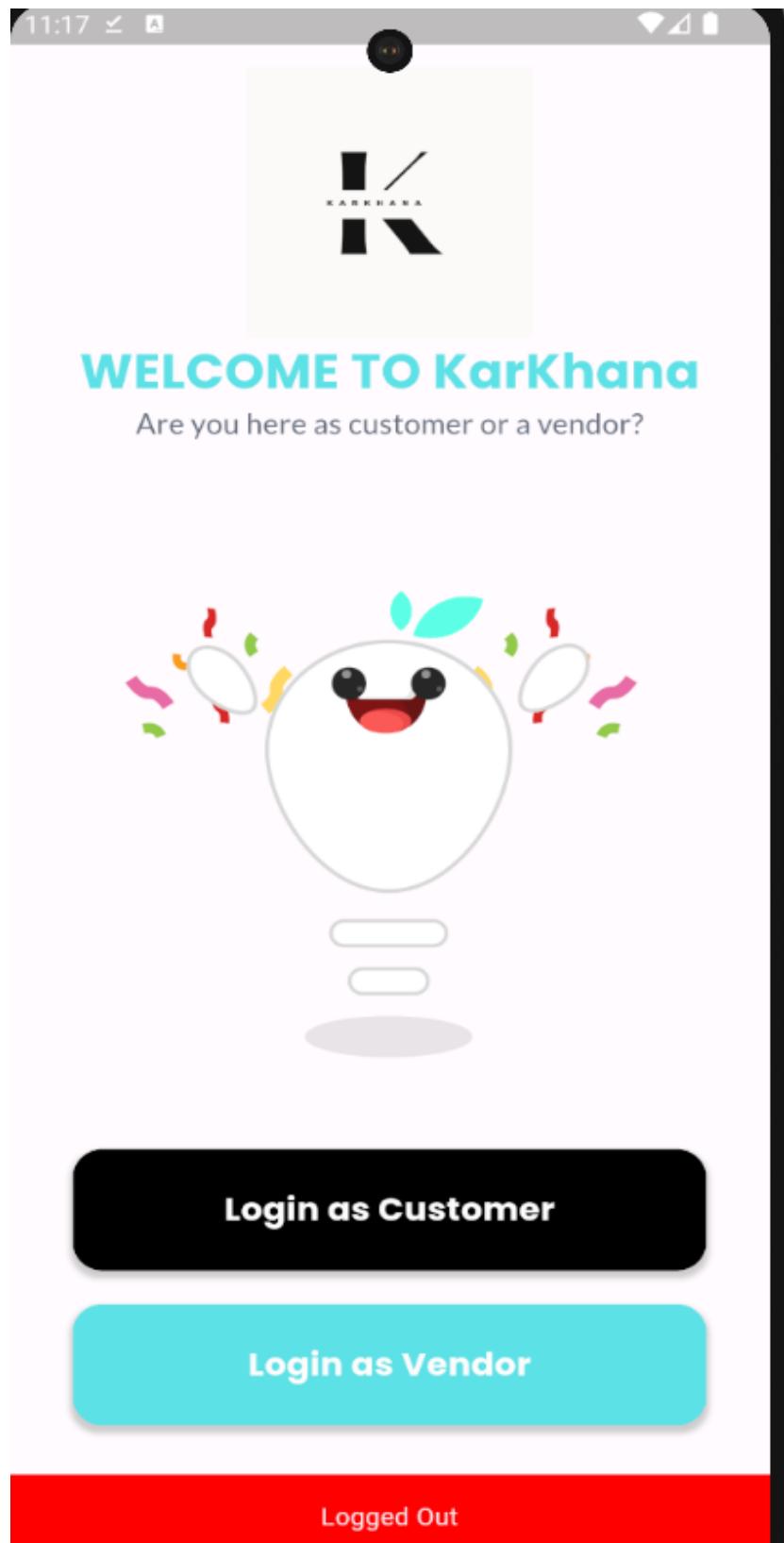


Figure 317: Logout testing -4.

---

## 4.3 SYSTEM TESTING

---

### 4.3.1 Connectivity Testing

---

<b>objective</b>	To check if the application works without internet.
<b>Actions</b>	Turn off the WIFI of the mobile system.
<b>Expected results</b>	An error message should be displayed, and the application should stop working.
<b>Obtained Results</b>	An error message was displayed, and the application stopped working.
<b>Conclusion</b>	Test Successful.

*Table 60: Connectivity testing.*

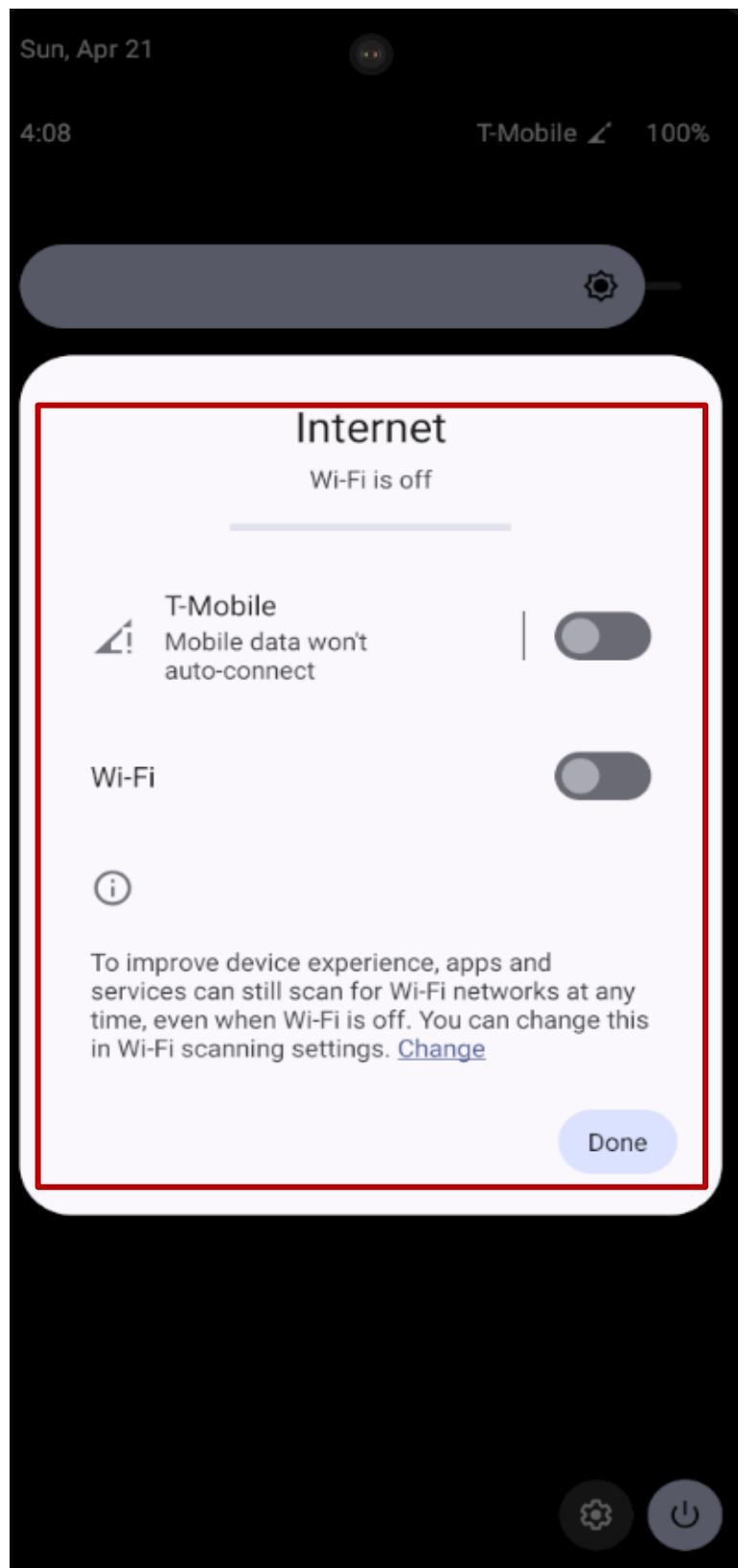


Figure 318: Connectivity testing -1.

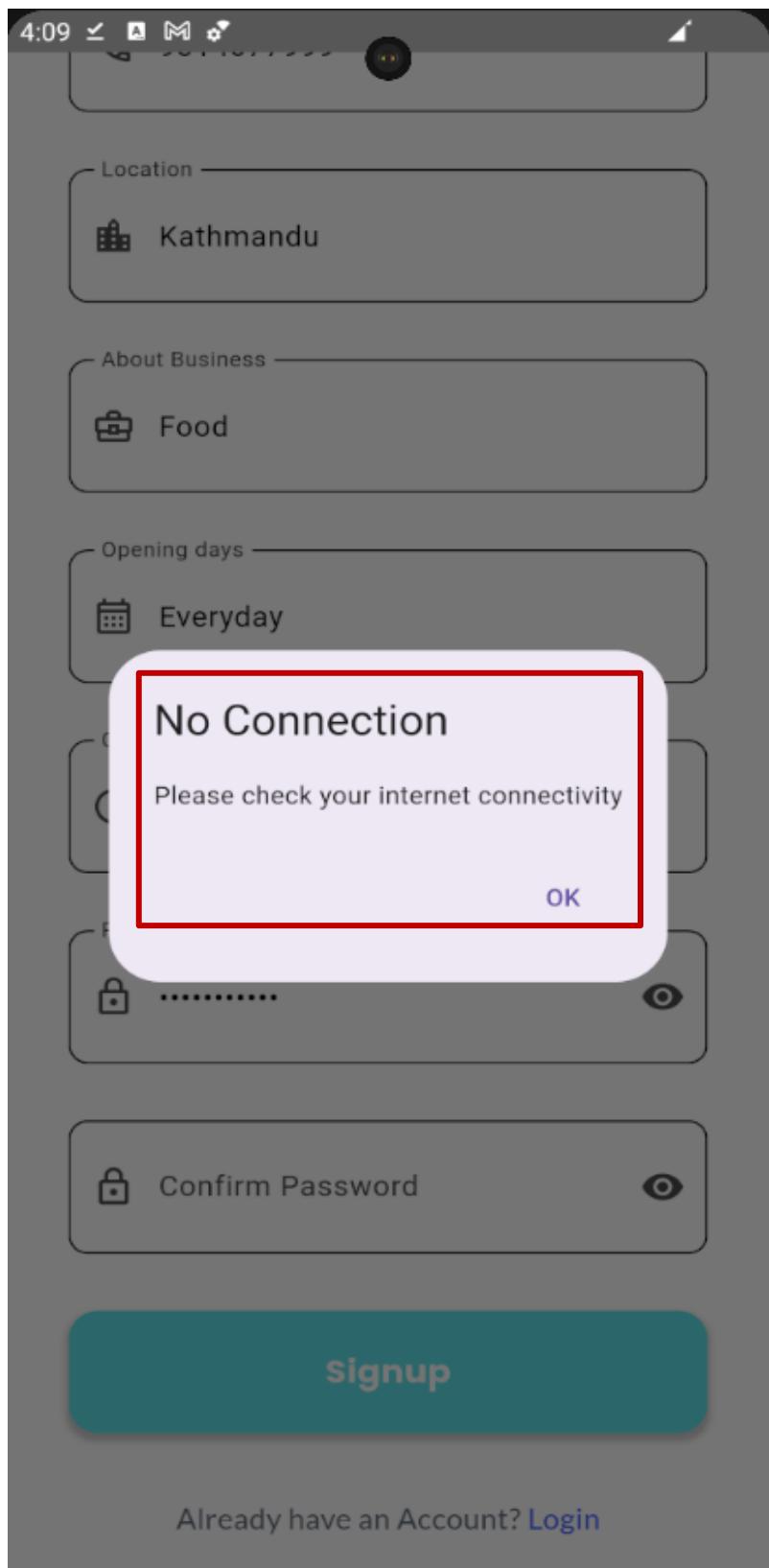


Figure 319: Connectivity Testing -2.

#### 4.3.2 Compatibility testing

<b>objective</b>	To check the compatibility of the application in different devices.
<b>Actions</b>	Run the application in android mobile.
<b>Expected results</b>	No error should be displayed, and application should be compatible to both devices.
<b>Obtained Results</b>	No error was displayed, and application was compatible to both devices.
<b>Conclusion</b>	Test Successful.

Table 61: Compatibility Testing.

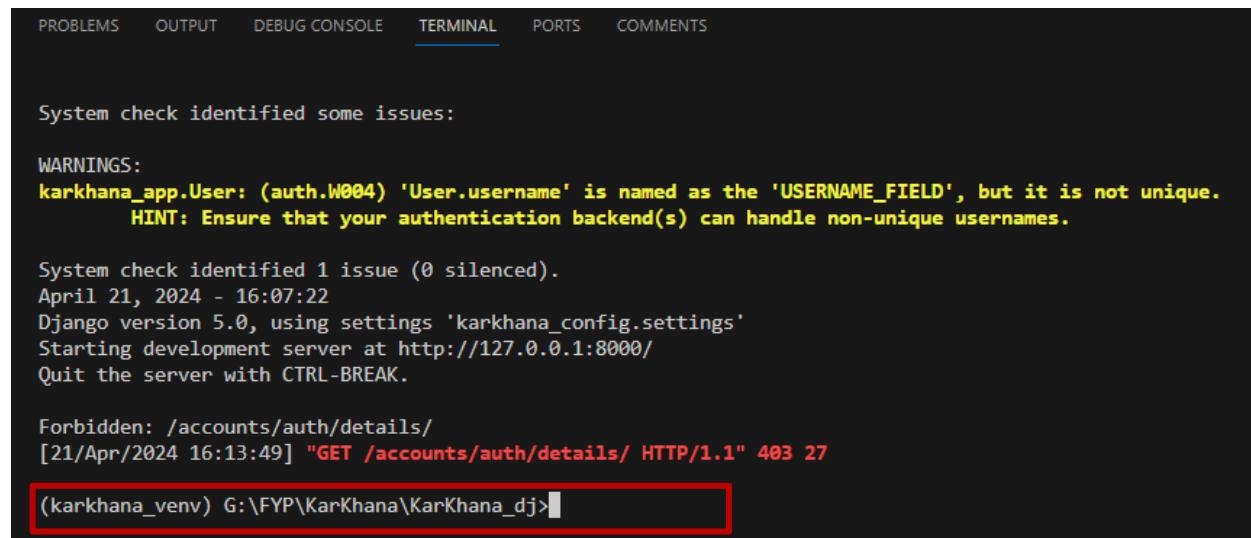


Figure 320: Compatibility test in Android phone.

### 4.3.3 Application without online server

<b>objective</b>	To run flutter project without running Django project.
<b>Actions</b>	Run the flutter project without Django project.
<b>Expected results</b>	Error should occur and system should crash.
<b>Obtained Results</b>	Error was occurred and system crashed.
<b>Conclusion</b>	Test Successful.

Table 62: Application without online server testing.



```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMENTS

System check identified some issues:

WARNINGS:
karkhana_app.User: (auth.W004) 'User.username' is named as the 'USERNAME_FIELD', but it is not unique.
    HINT: Ensure that your authentication backend(s) can handle non-unique usernames.

System check identified 1 issue (0 silenced).
April 21, 2024 - 16:07:22
Django version 5.0, using settings 'karkhana_config.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

Forbidden: /accounts/auth/details/
[21/Apr/2024 16:13:49] "GET /accounts/auth/details/ HTTP/1.1" 403 27

(karkhana_venv) G:\FYP\KarKhana\KarKhana_dj>█

```

Figure 321: Application without online server testing -1 (Django Project not running).

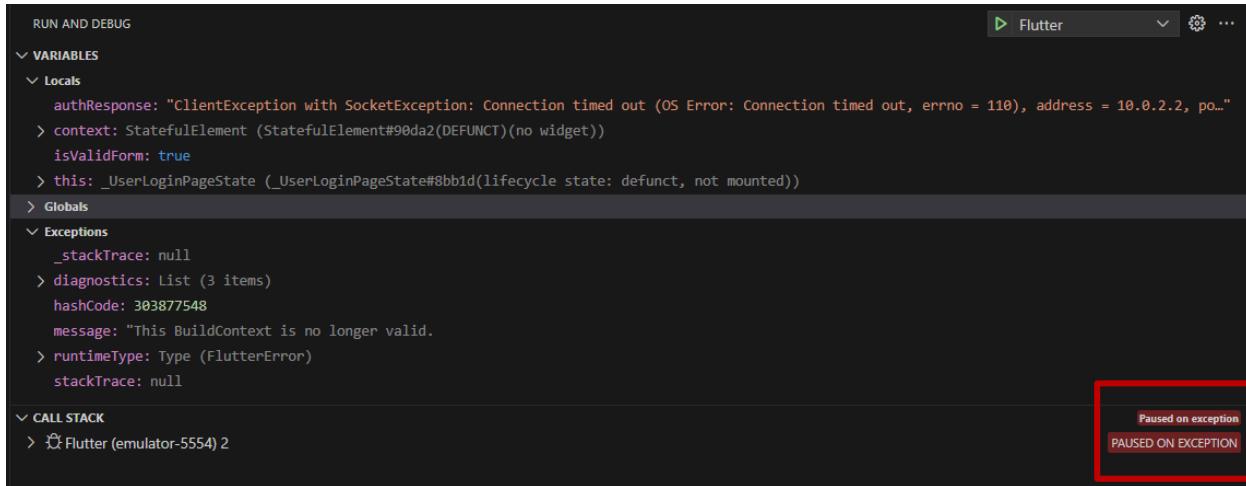


Figure 322: Application without online server testing -2 (Error Occurred in flutter project).

#### 4.3.4 System walkthrough testing for customer

objective	To test if the application is working smoothly for customer.
<b>Actions</b>	Navigate to login page from welcome page, Login into the system, get current location, select a deal, then add that deal to cart, buy the coupon for that deal then logout.
<b>Expected results</b>	No error should occur in this process.
<b>Obtained Results</b>	No error occurred in this process.
<b>Conclusion</b>	Test Successful.

Table 63: System walkthrough testing table.

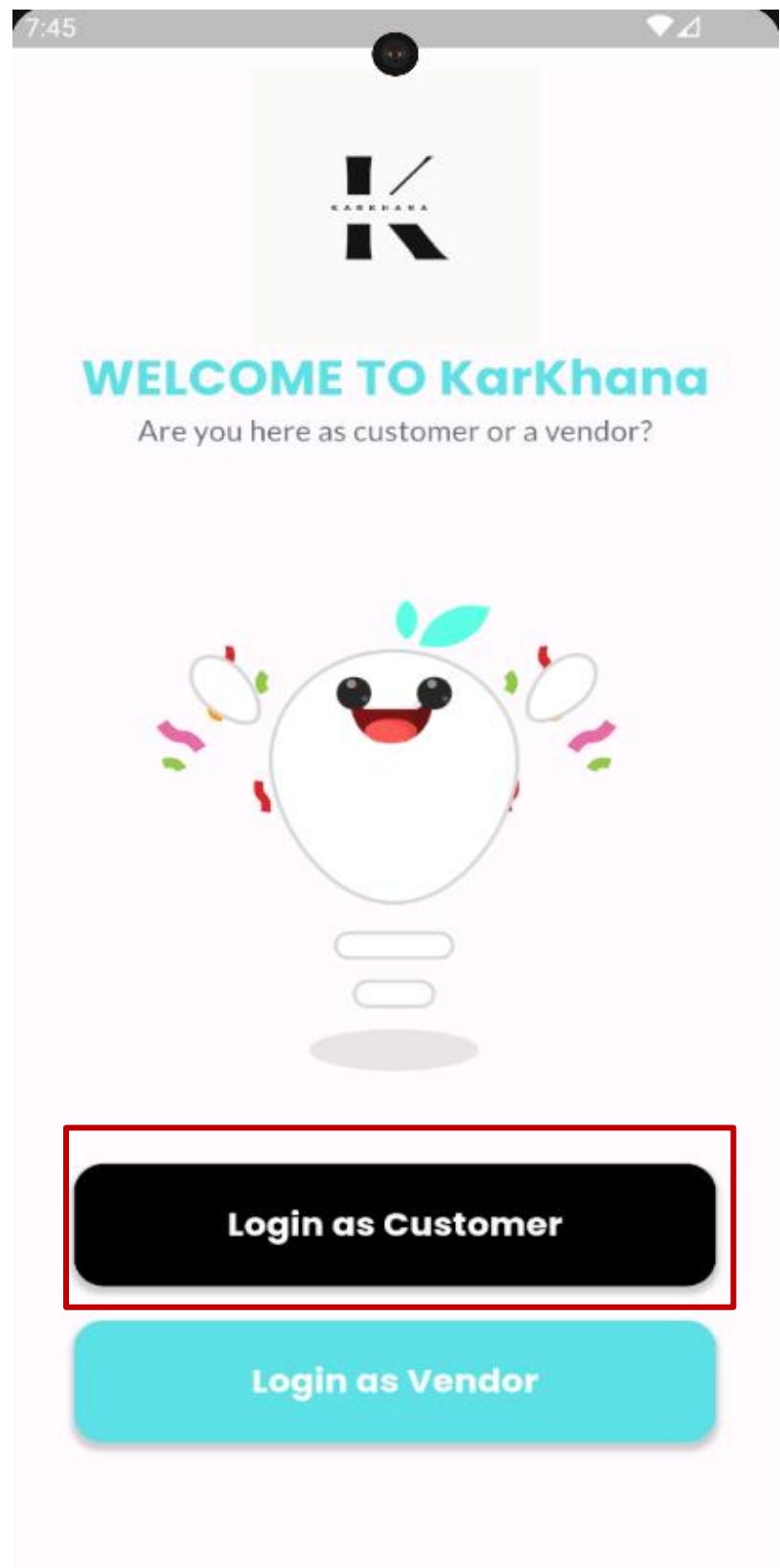


Figure 323: System walkthrough testing -1.

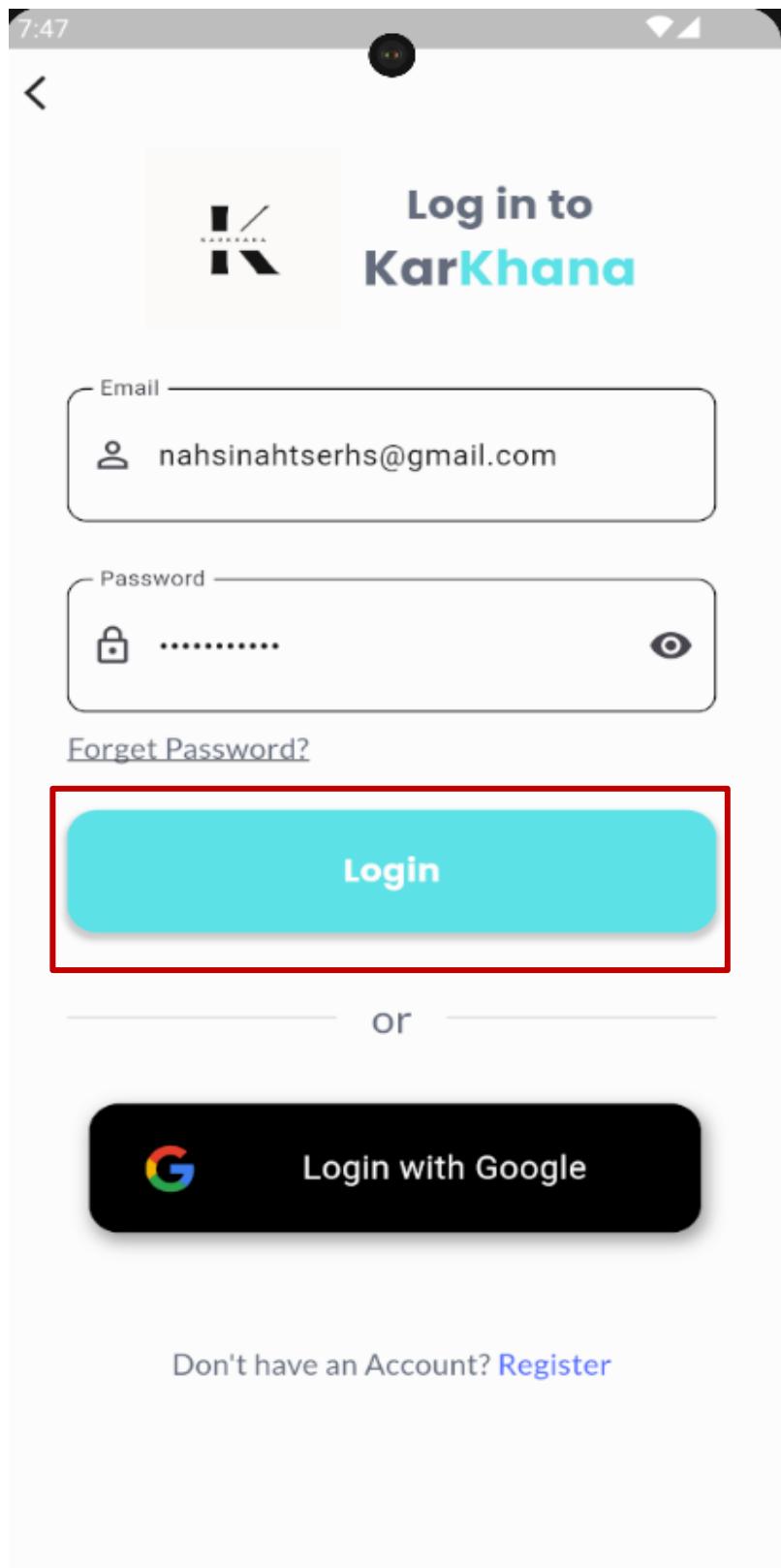


Figure 324: System walkthrough testing -2.

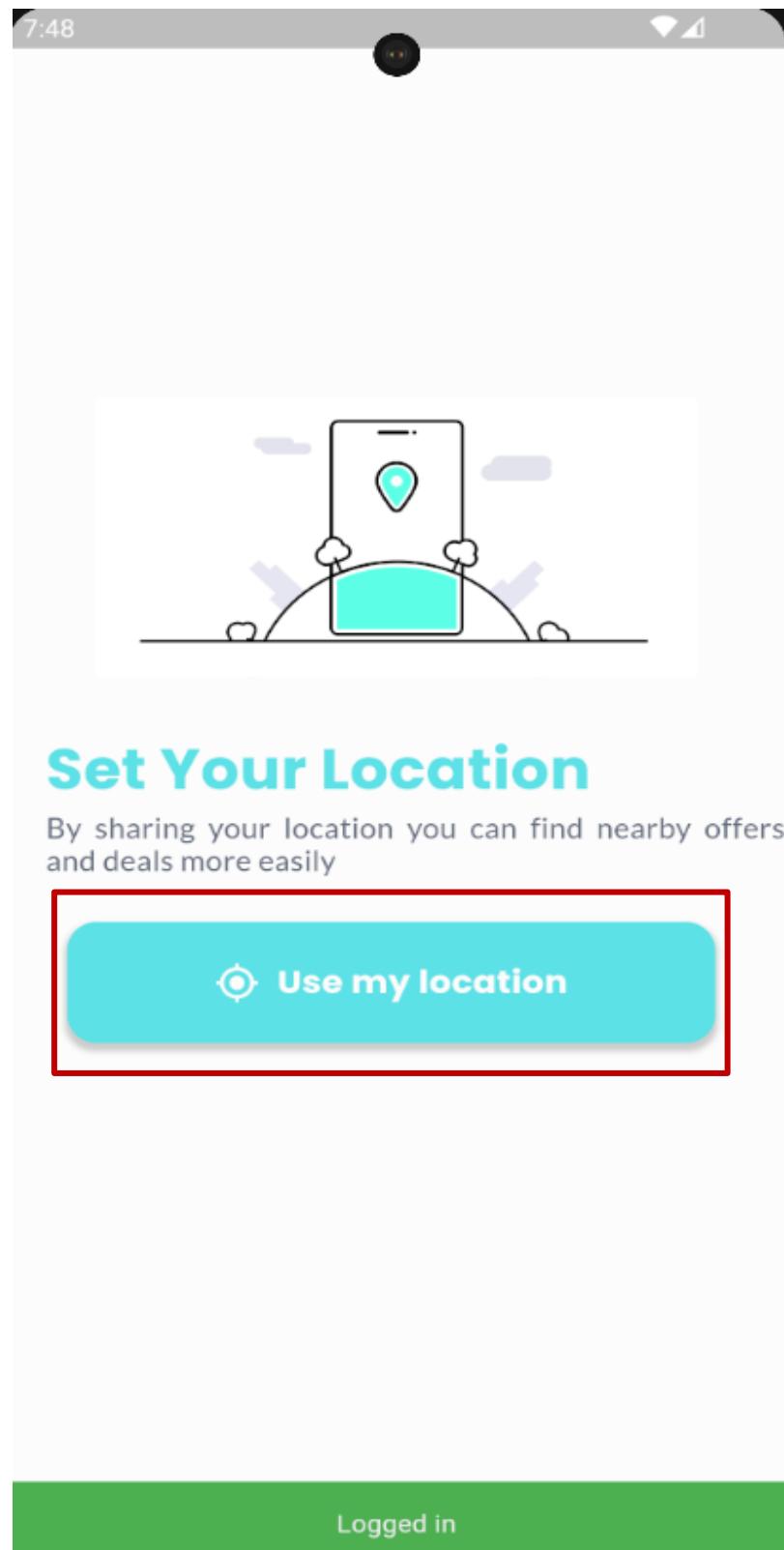


Figure 325: System walkthrough testing -3.

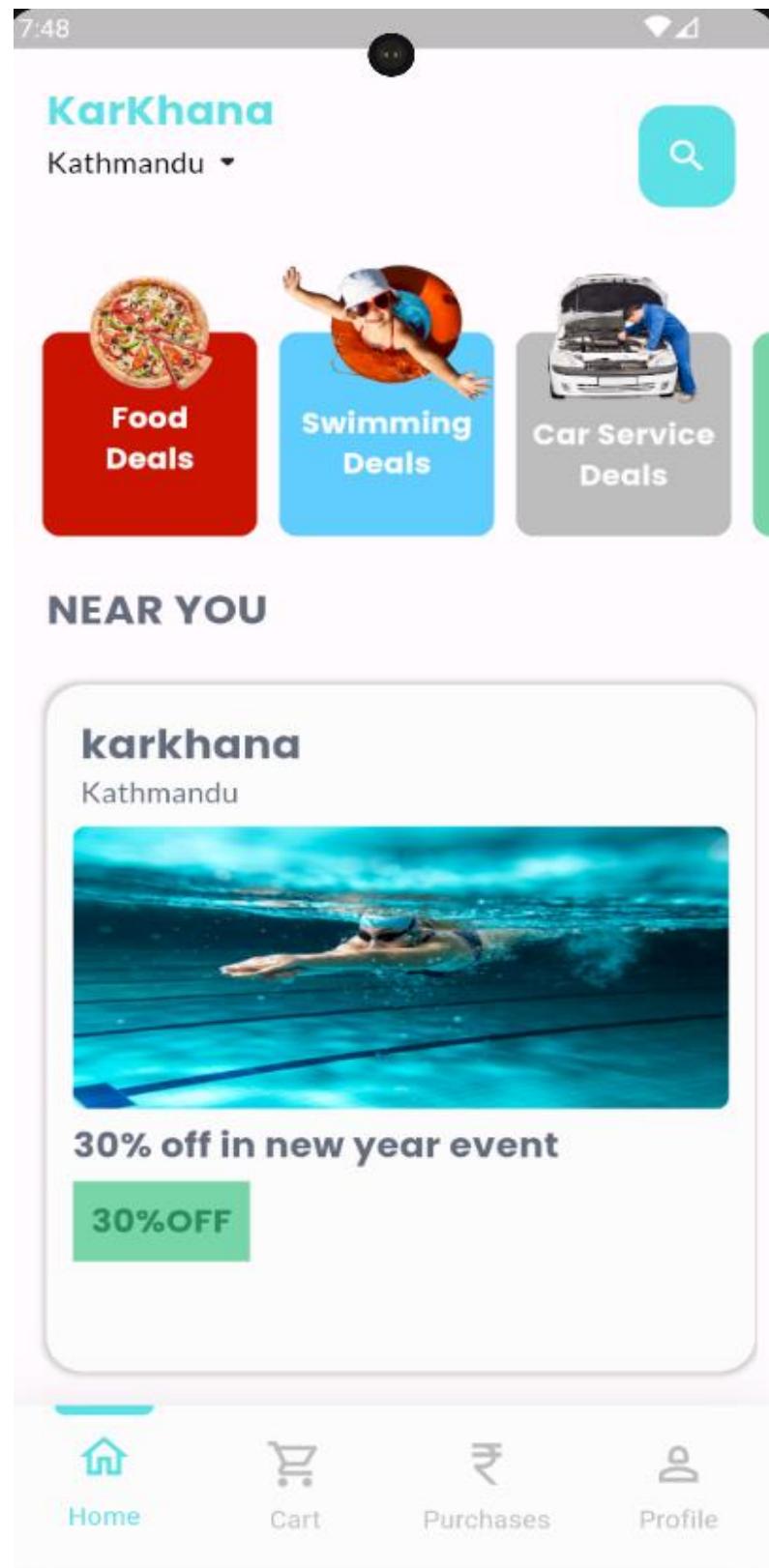


Figure 326: System walkthrough testing -4.

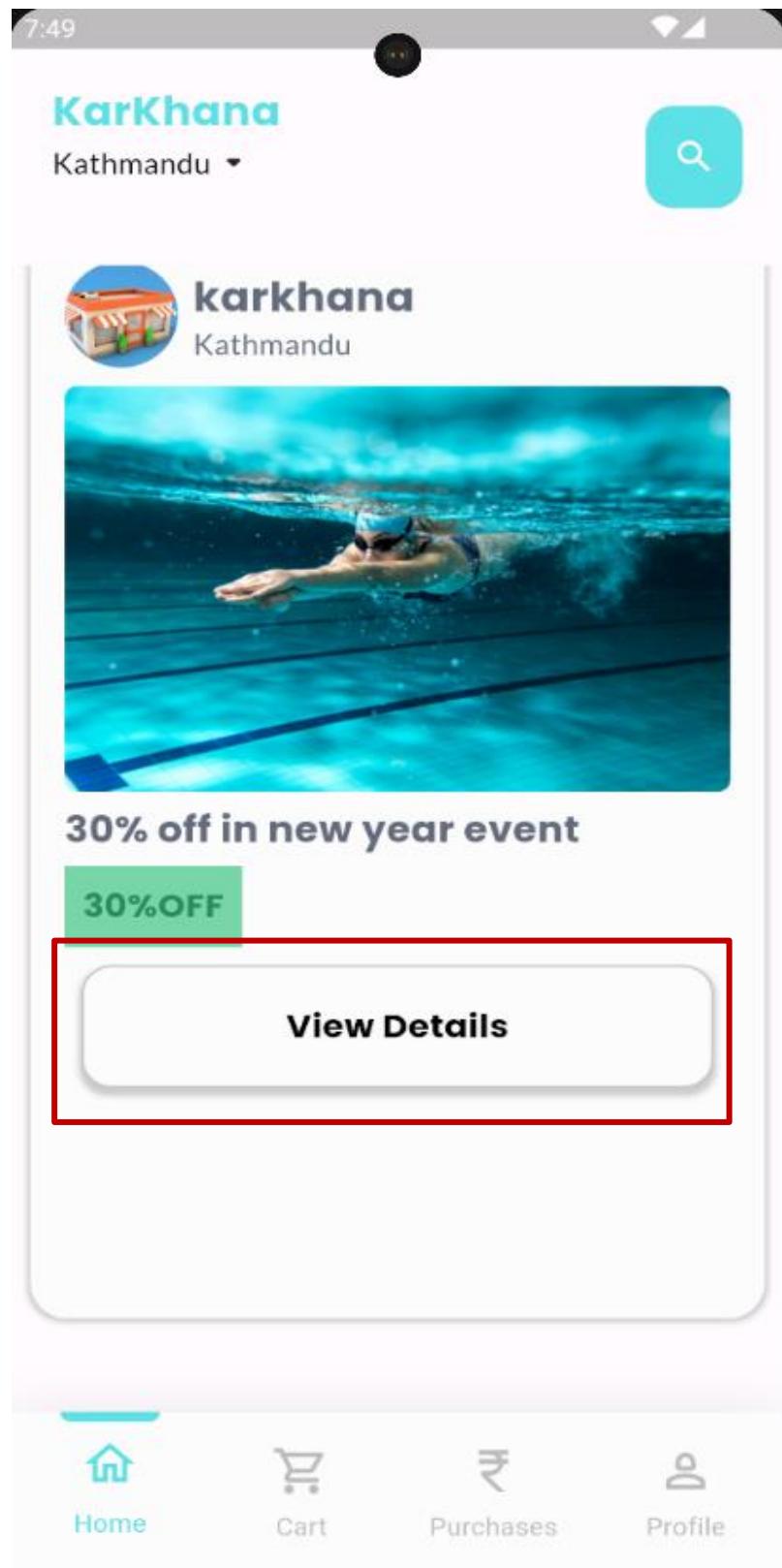


Figure 327: System walkthrough testing -5.

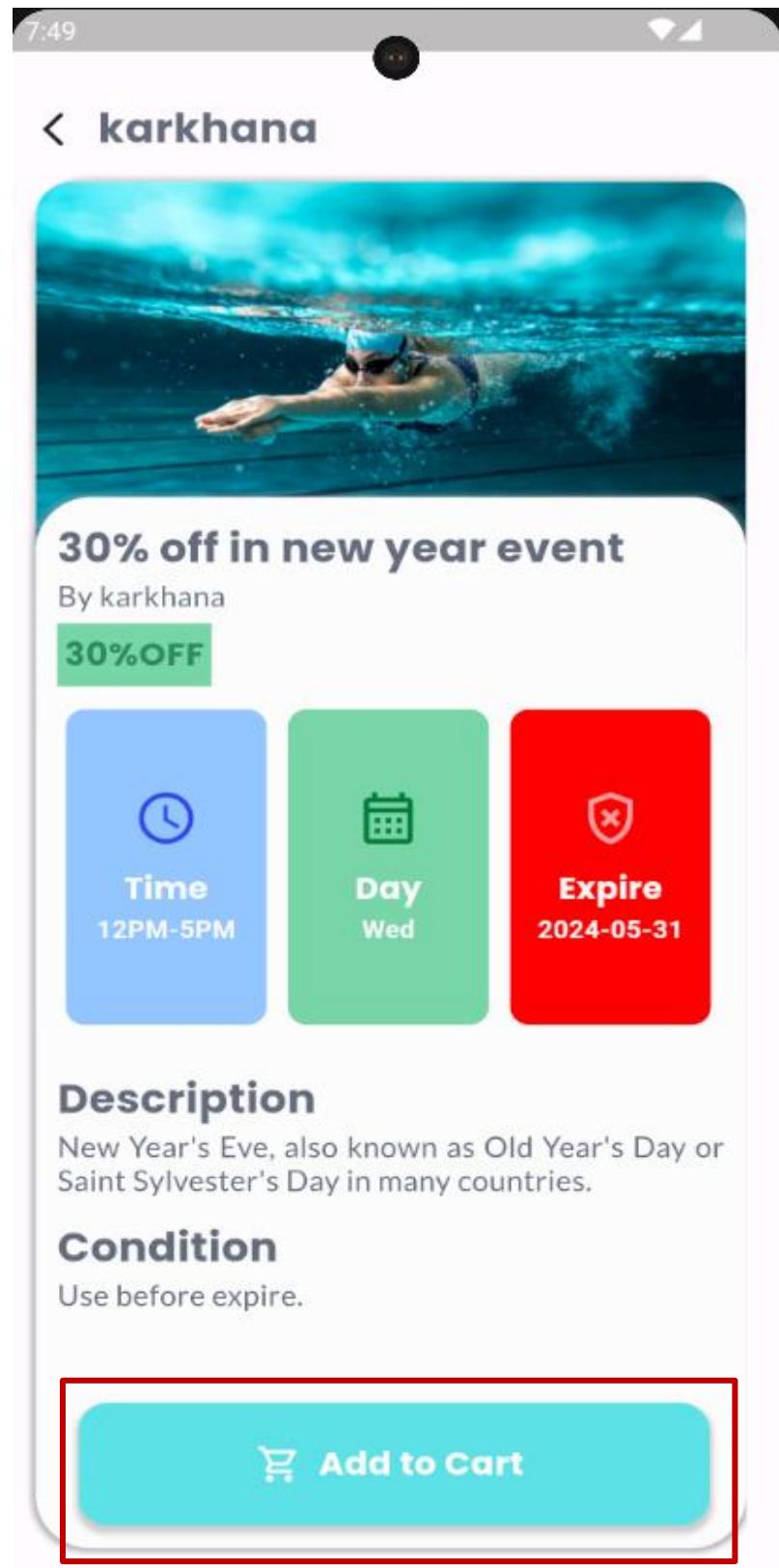


Figure 328: System walkthrough testing -6.

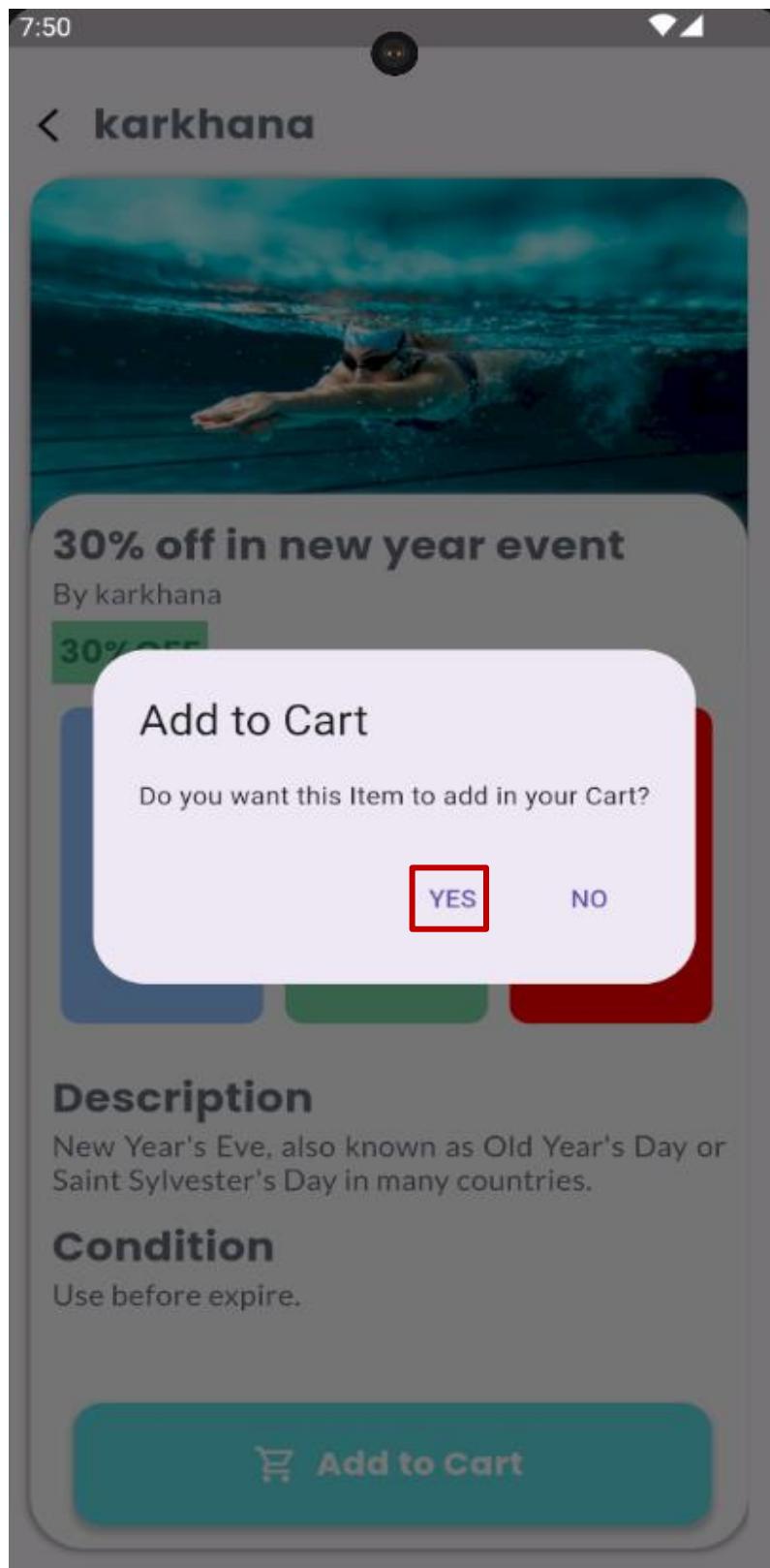


Figure 329: System walkthrough testing -7.

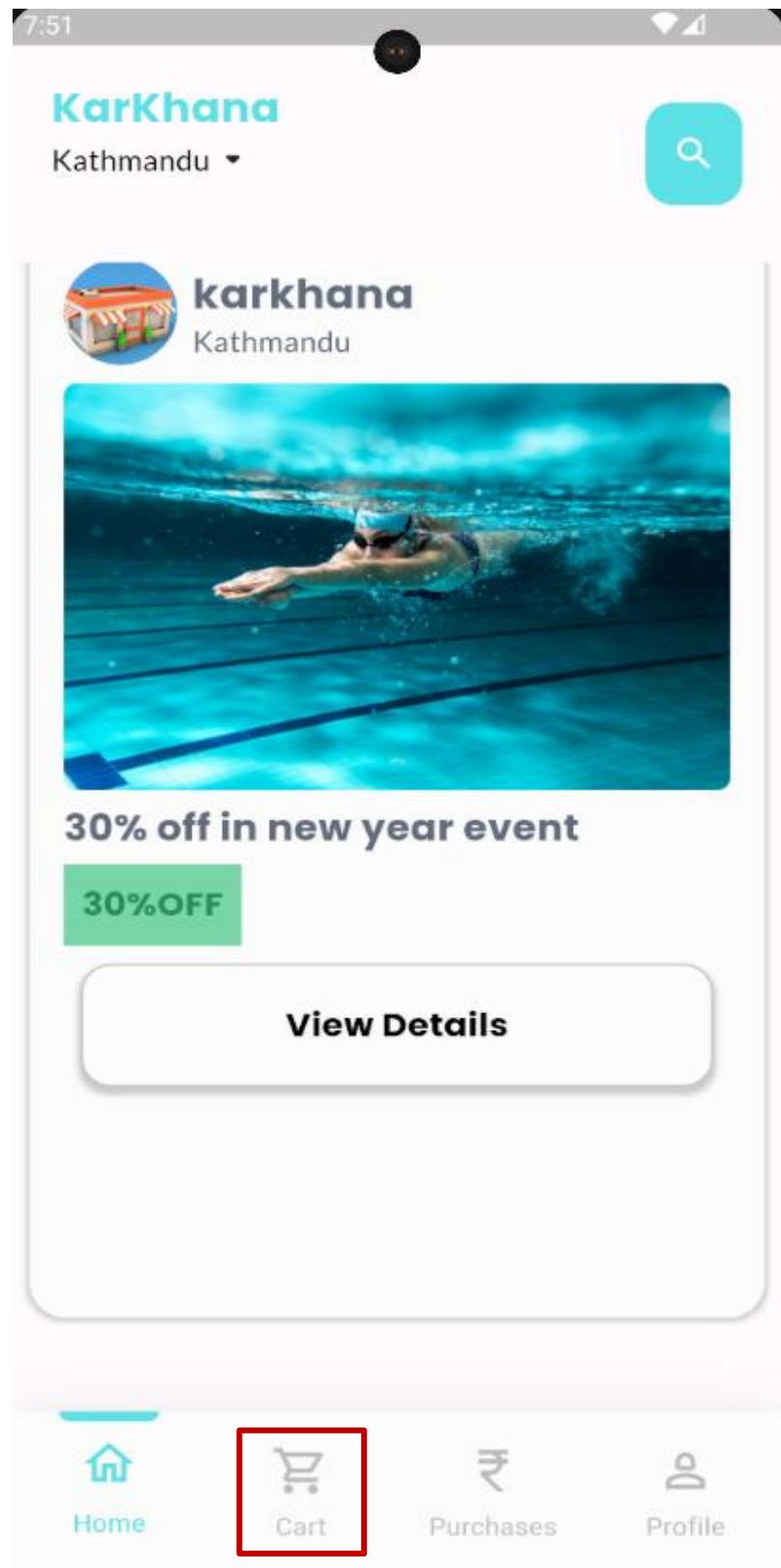


Figure 330: System walkthrough testing -8.

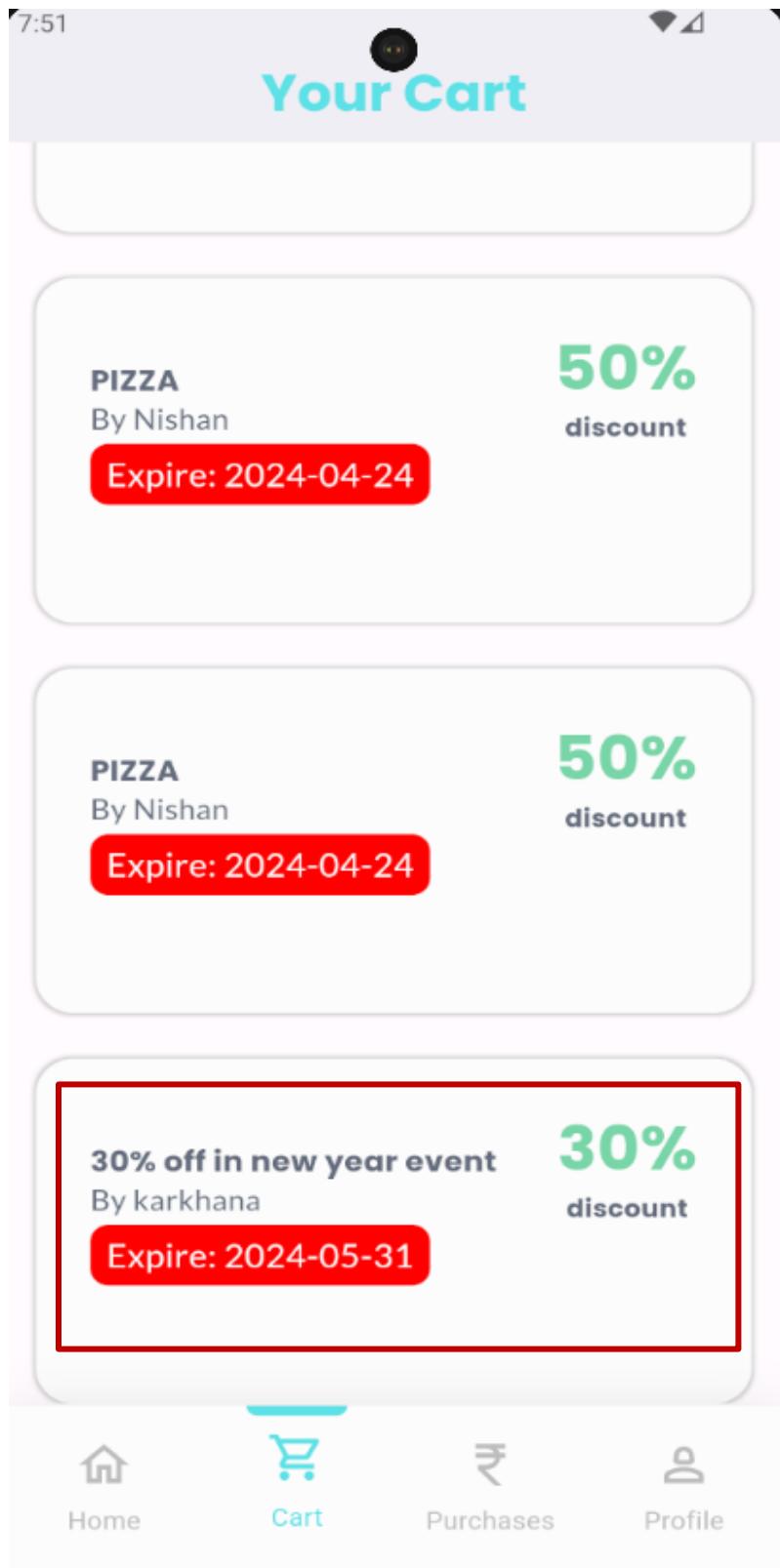


Figure 331: System walkthrough testing -9.

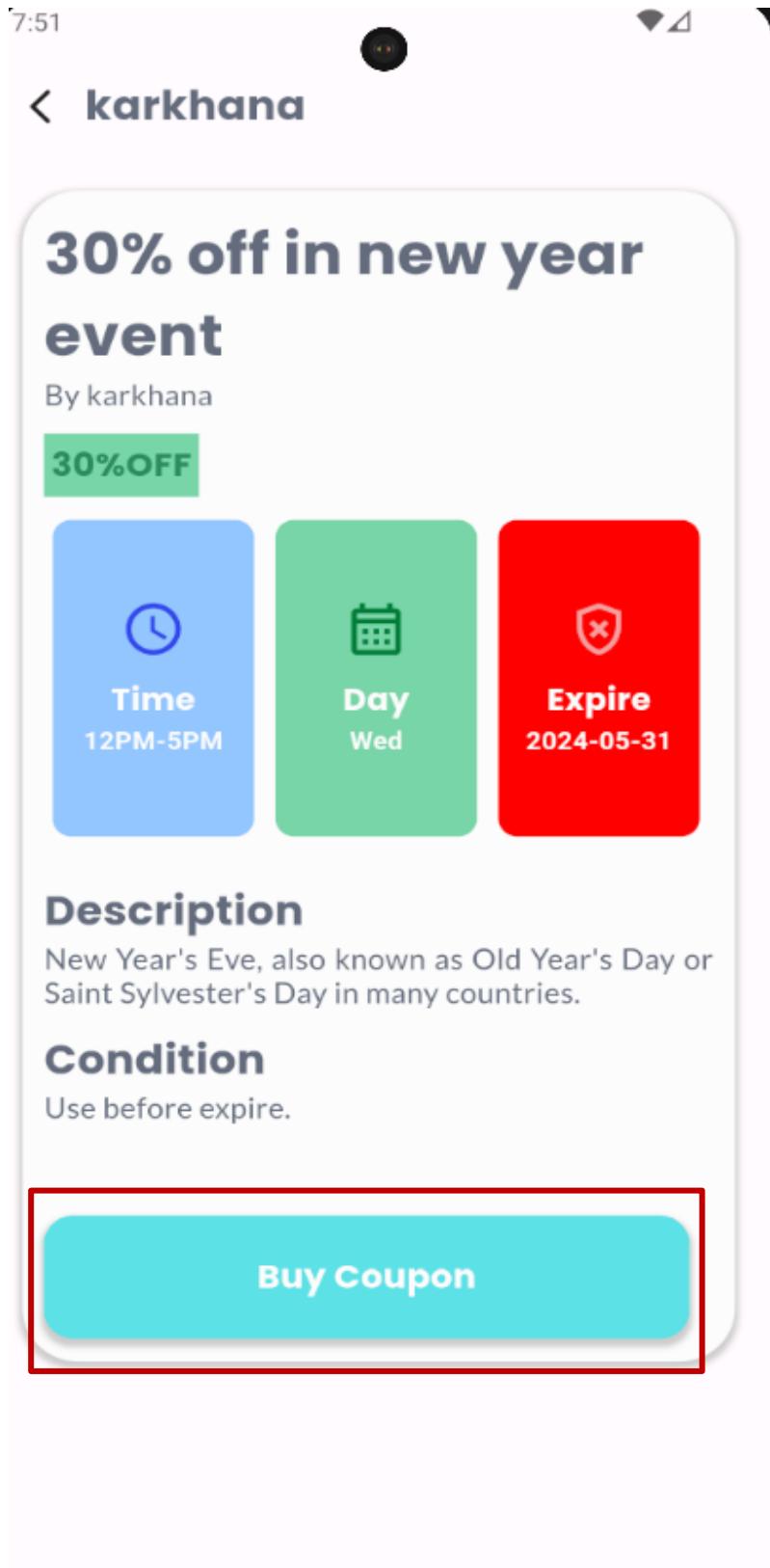


Figure 332: System walkthrough testing -10.

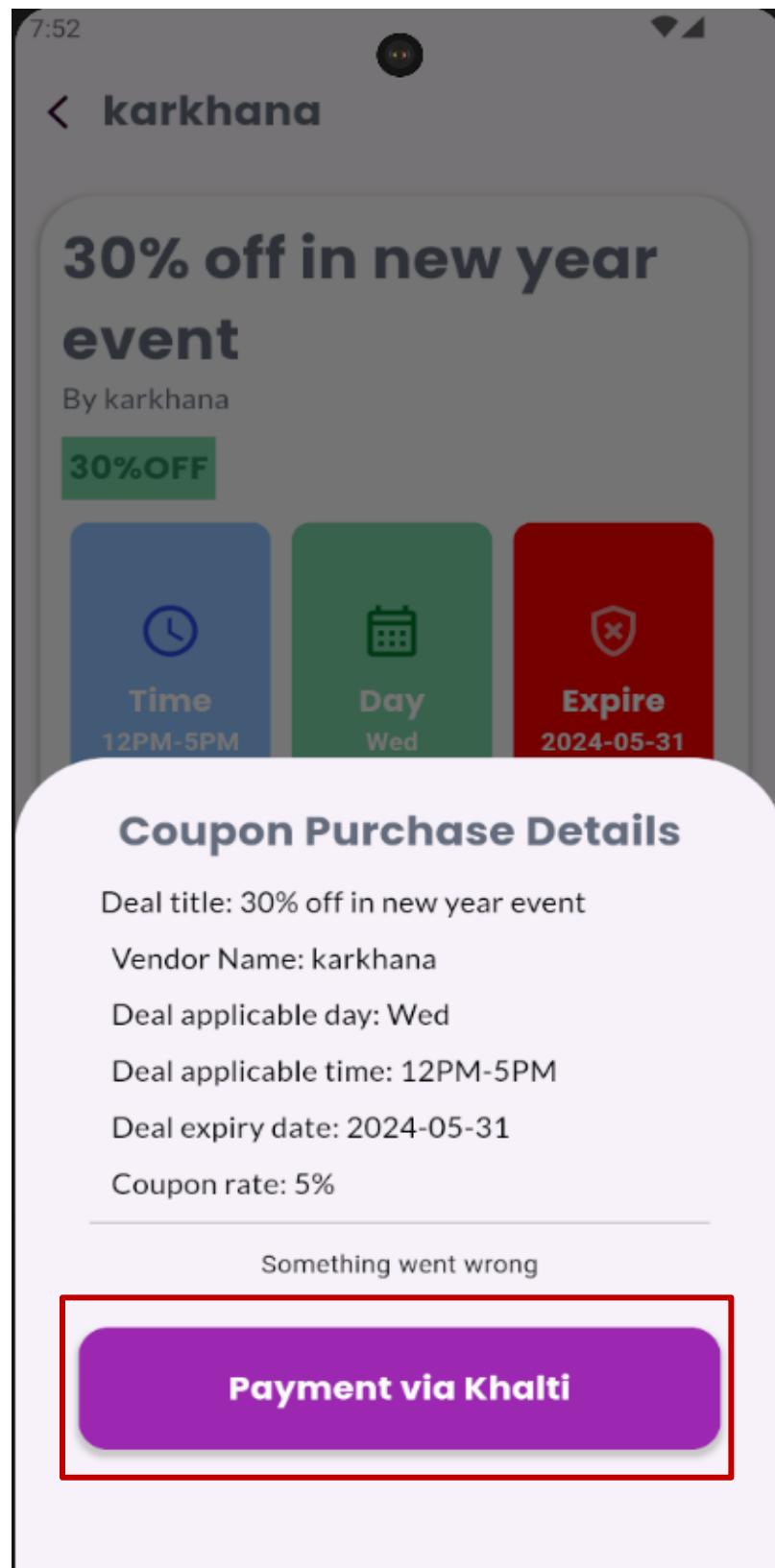


Figure 333: System walkthrough testing -11.



Figure 334: System walkthrough testing -12.



Figure 335: System walkthrough testing -13.

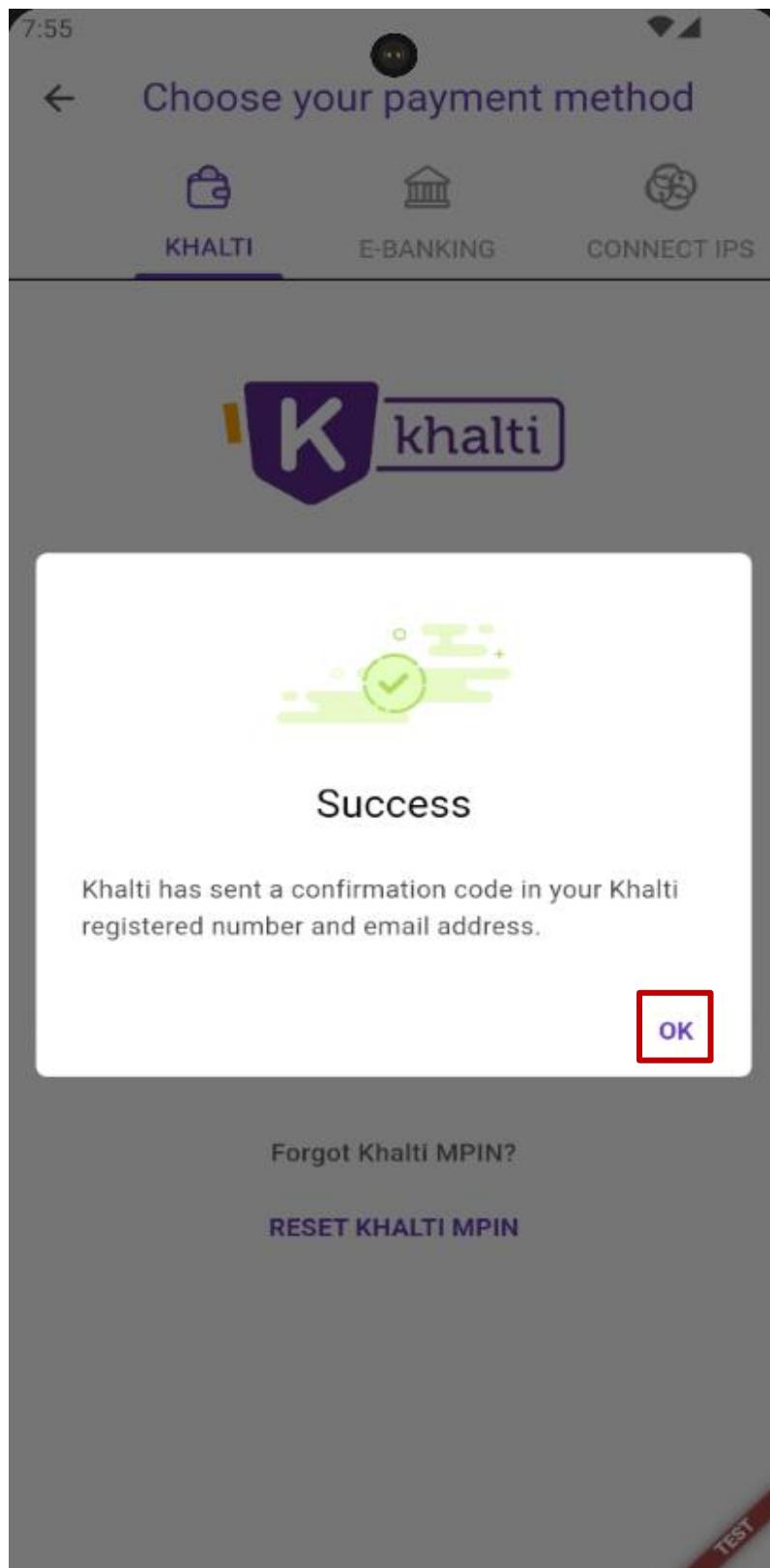


Figure 336: System walkthrough testing -14.

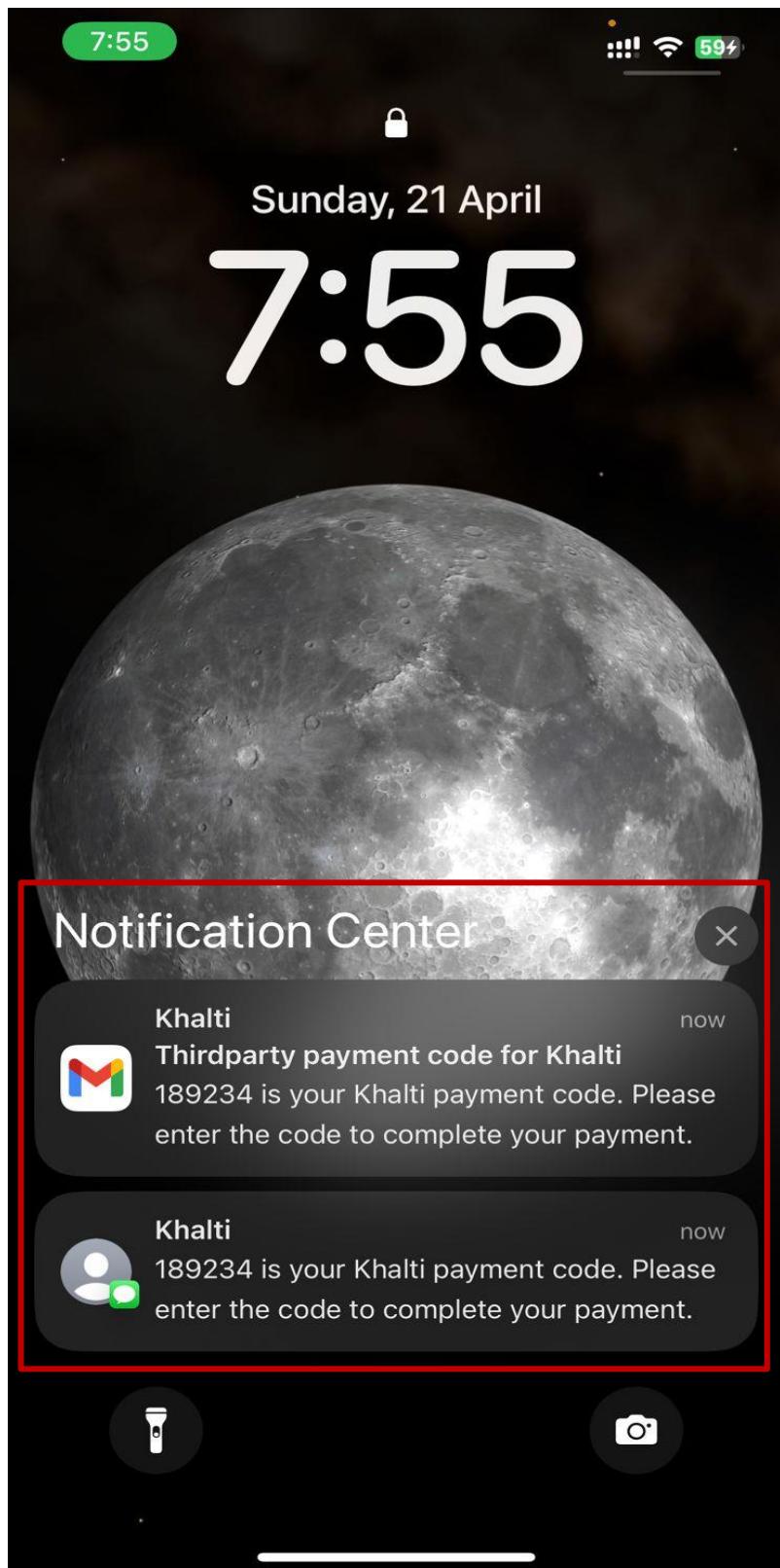


Figure 337: System walkthrough testing -15.

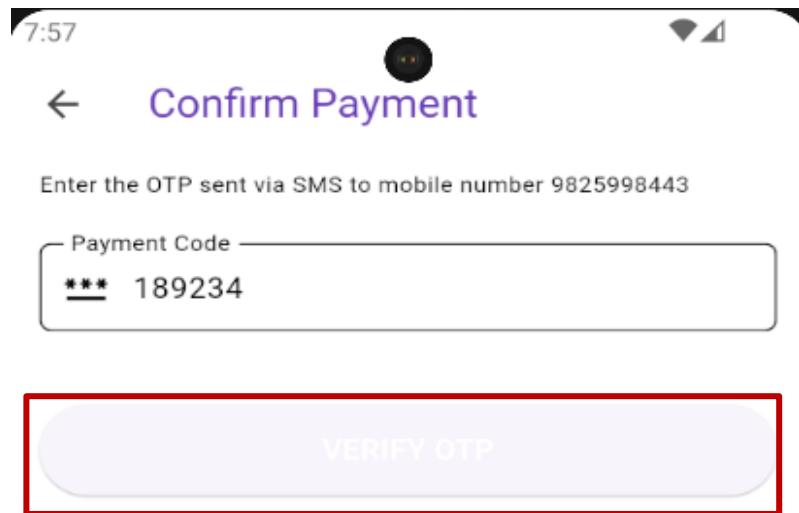


Figure 338: System walkthrough testing -16.

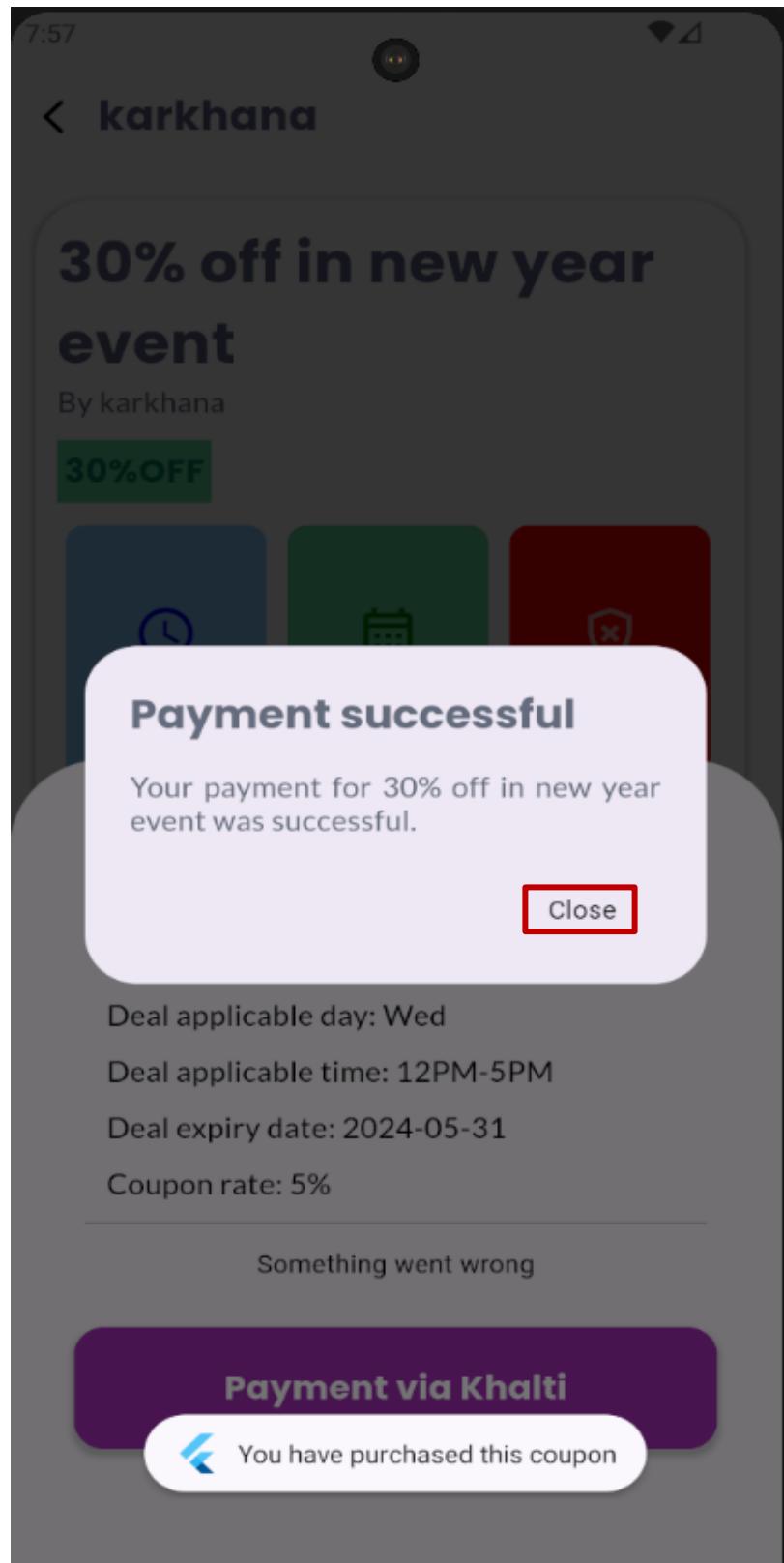


Figure 339: System walkthrough testing -17.

## Notification from KarKhana.

Inbox ×



karkhana2024@gmail.com

to me ▾

Your Purchase for 30% off in new year event is completed. Enjoy!

Reply

Forward



Figure 340: System walkthrough testing -18.

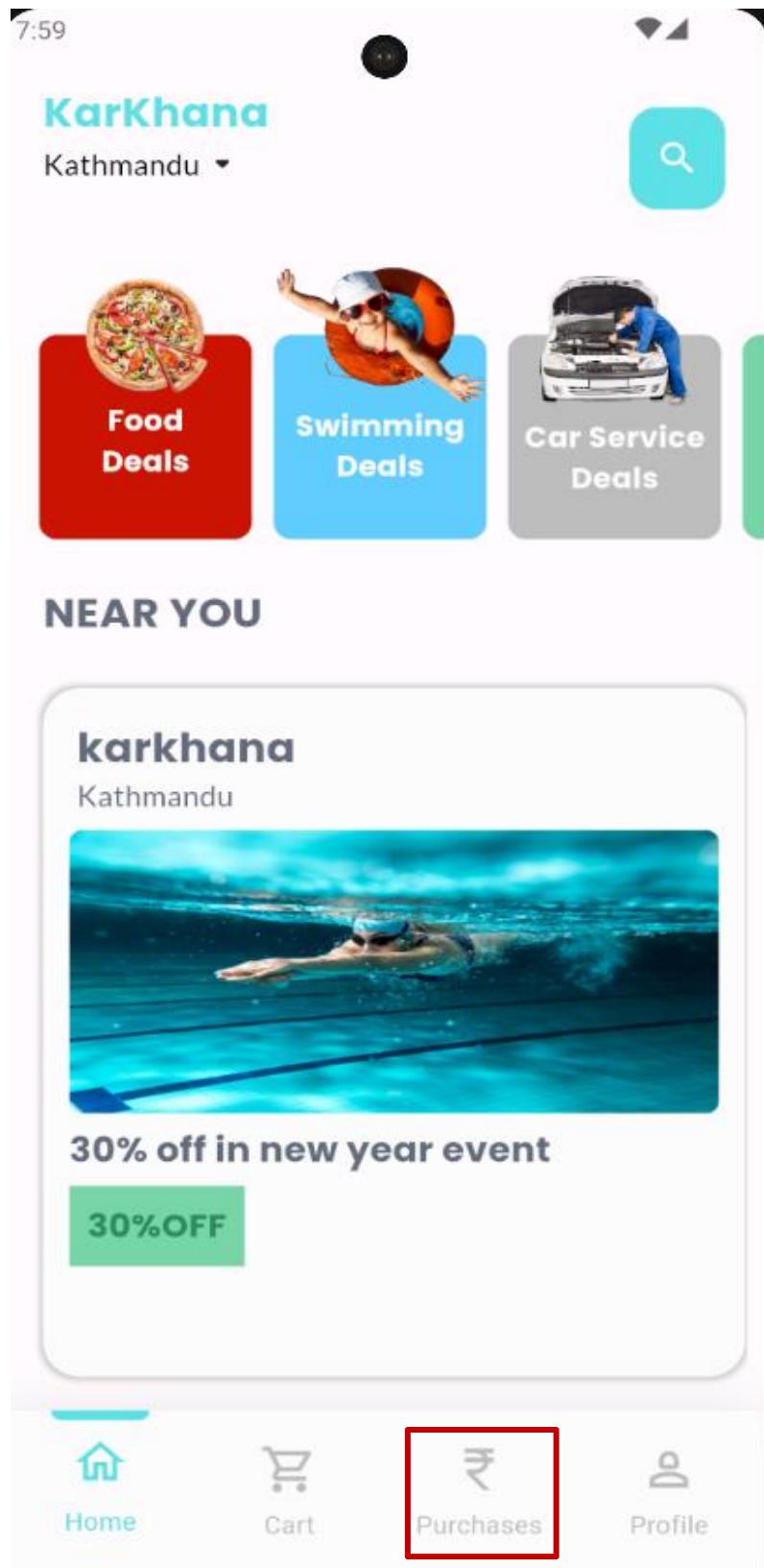


Figure 341: System walkthrough testing -19.

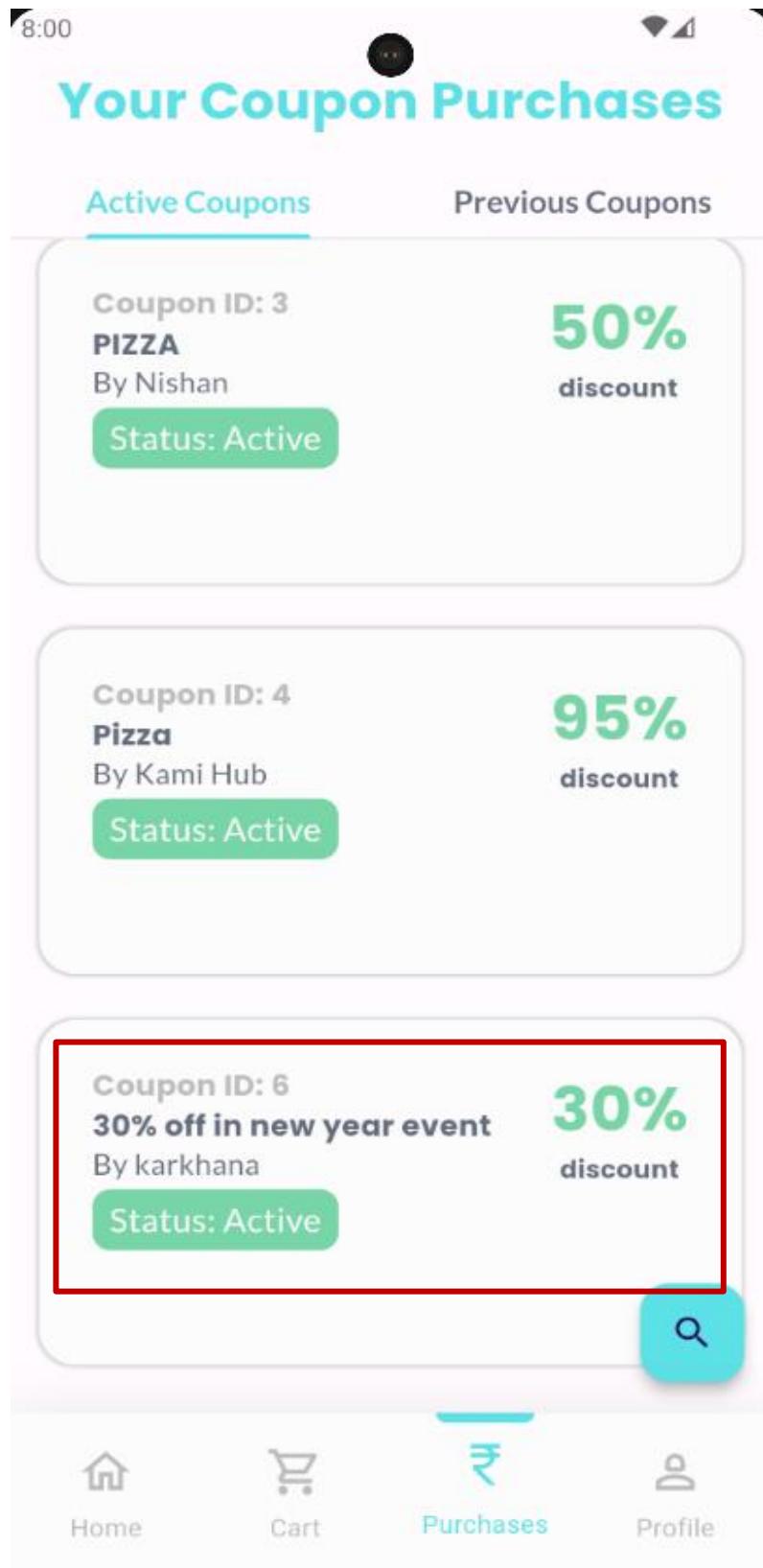


Figure 342: System walkthrough testing -20.

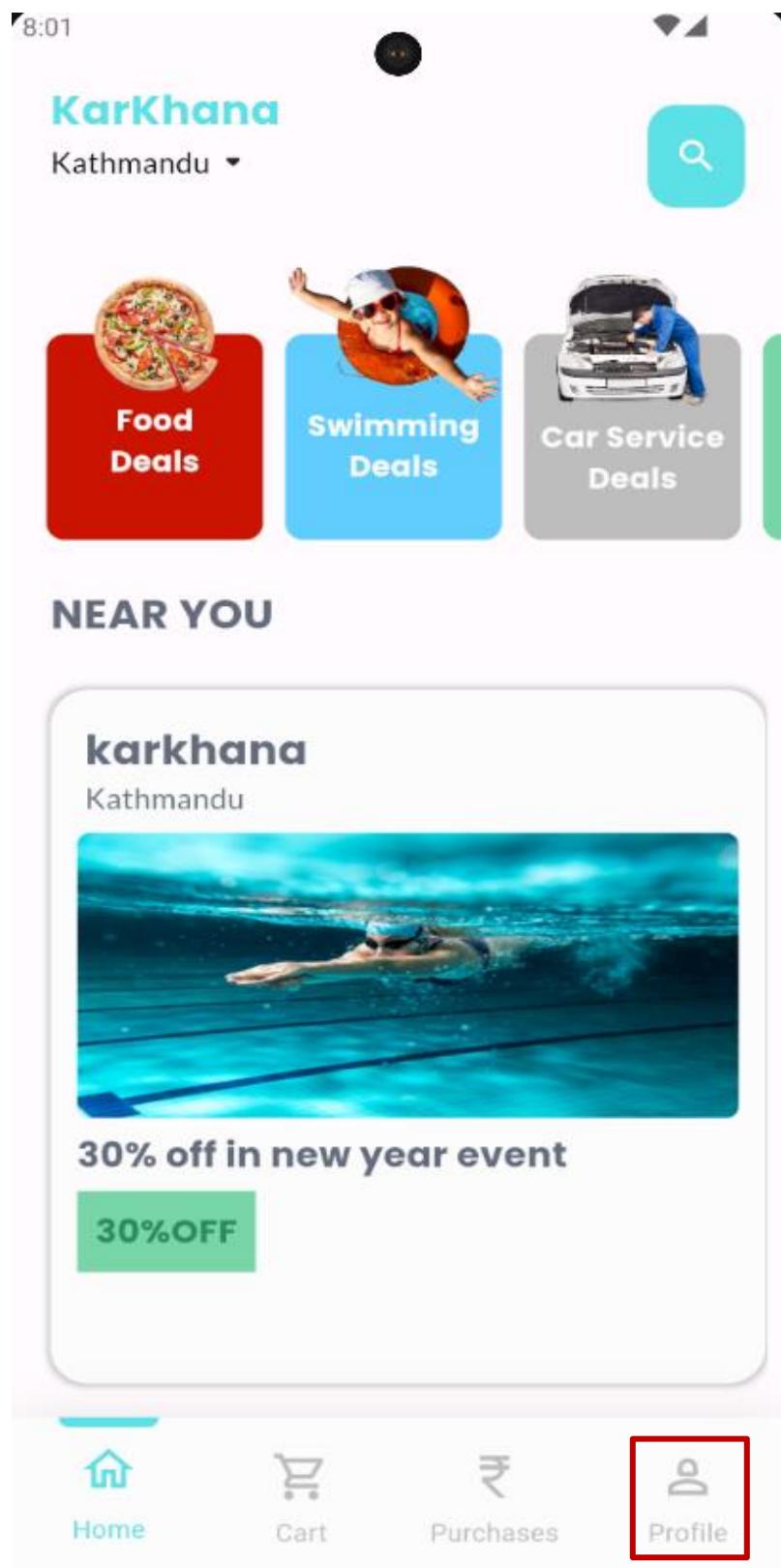


Figure 343: System walkthrough testing -21.

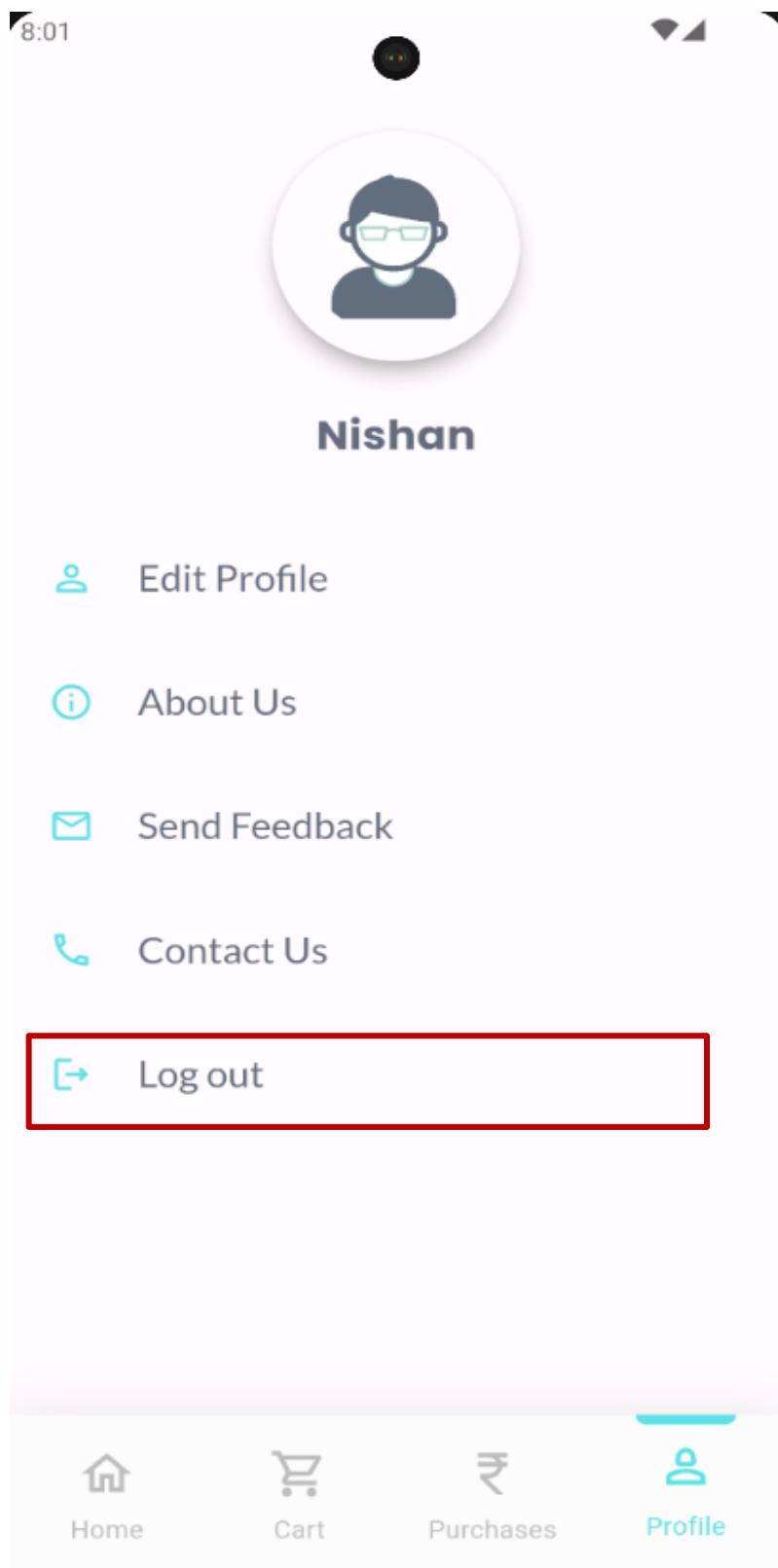


Figure 344: System walkthrough testing -22.

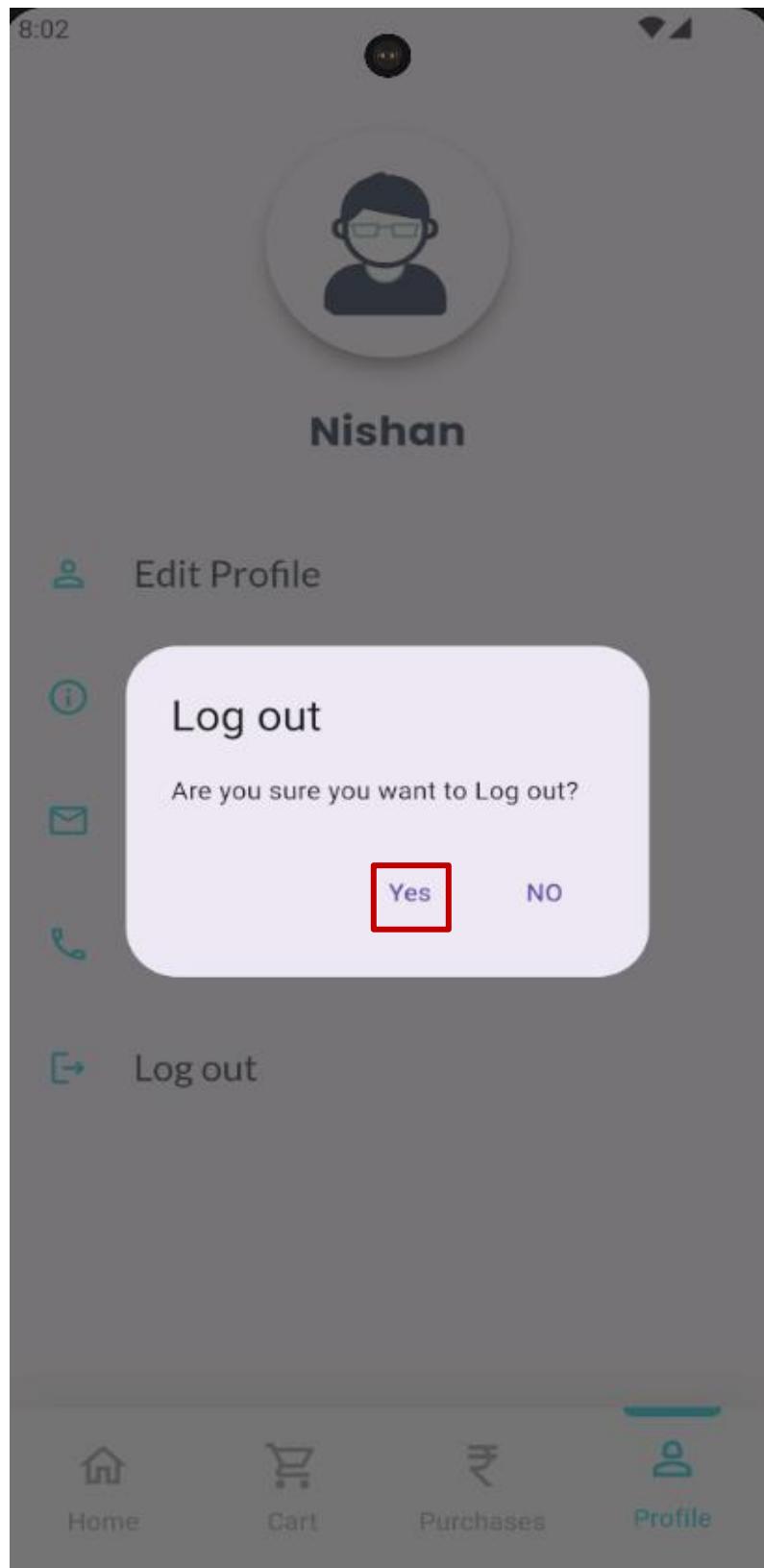


Figure 345: System walkthrough testing -23.

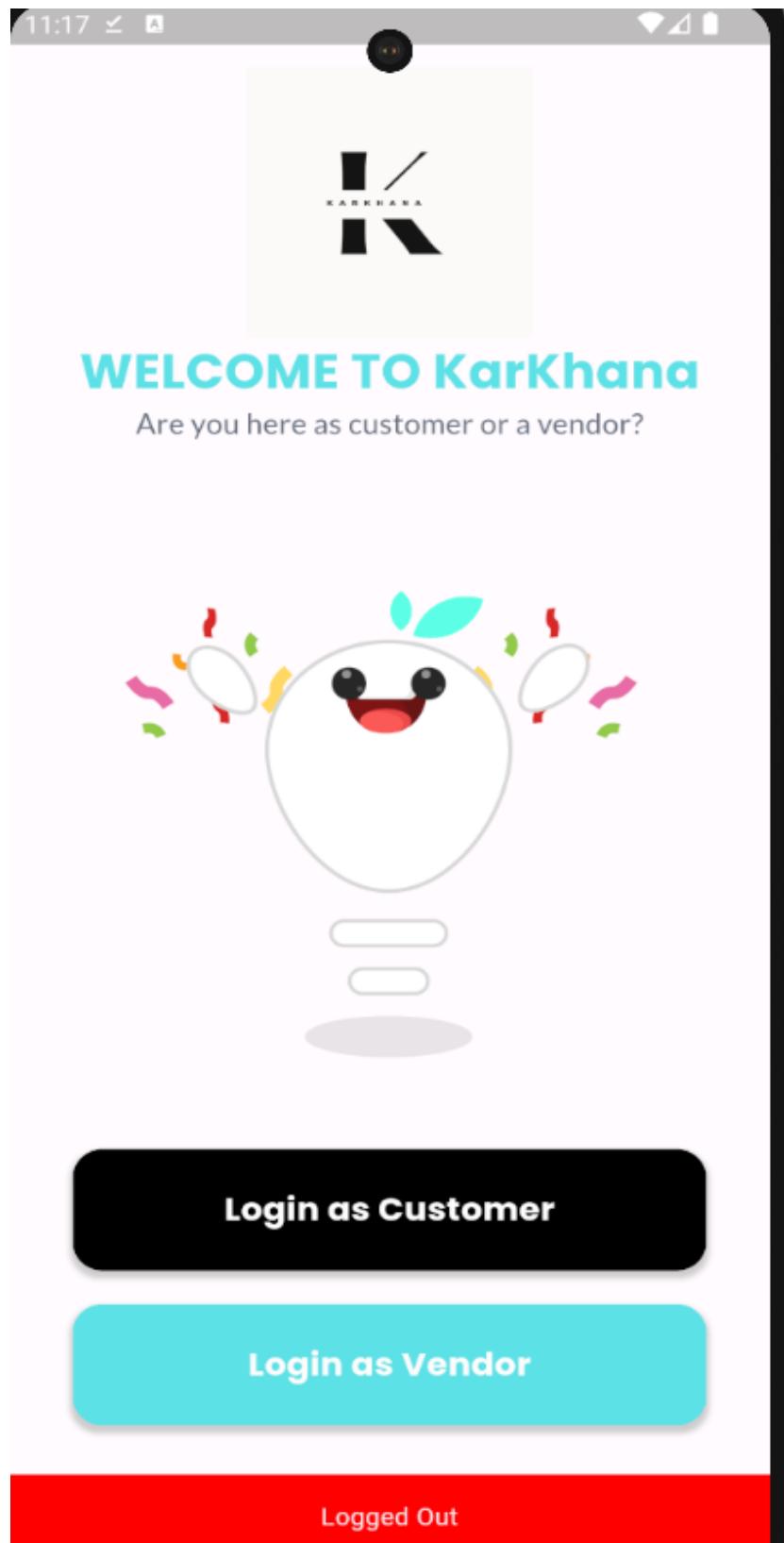


Figure 346: System walkthrough testing -24.

**4.3.4 System walkthrough testing for vendor.**

<b>objective</b>	To test if the application is working smoothly for vendor.
<b>Actions</b>	Navigate to login page from welcome page, Login into the system, add a deal, then view the deal added, confirm the coupon for that deal then logout.
<b>Expected results</b>	No error should occur in this process.
<b>Obtained Results</b>	No error occurred in this process.
<b>Conclusion</b>	Test Successful.

*Table 64: System walkthrough testing table for vendor.*

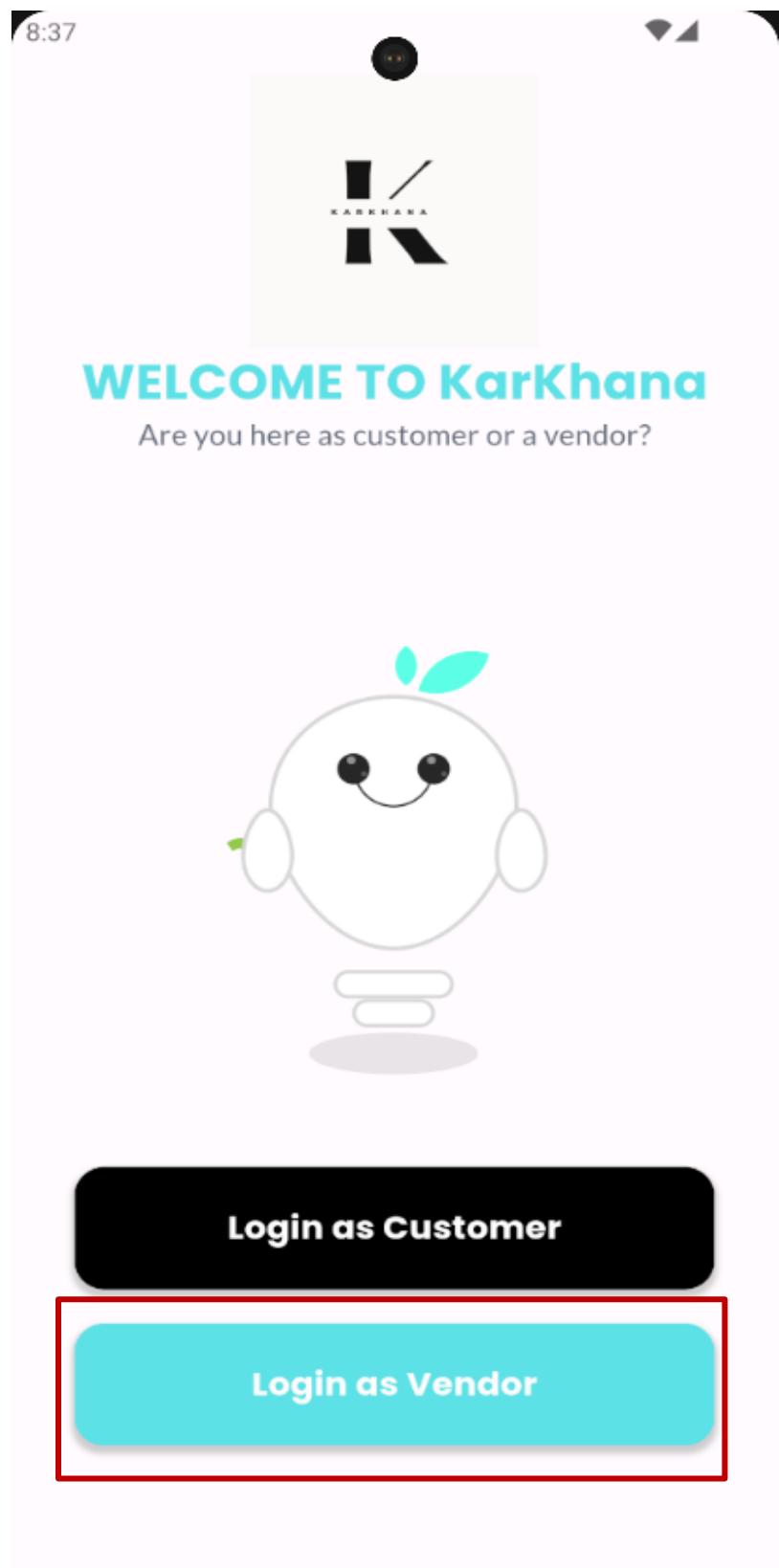


Figure 347: System walkthrough testing table for vendor -1.

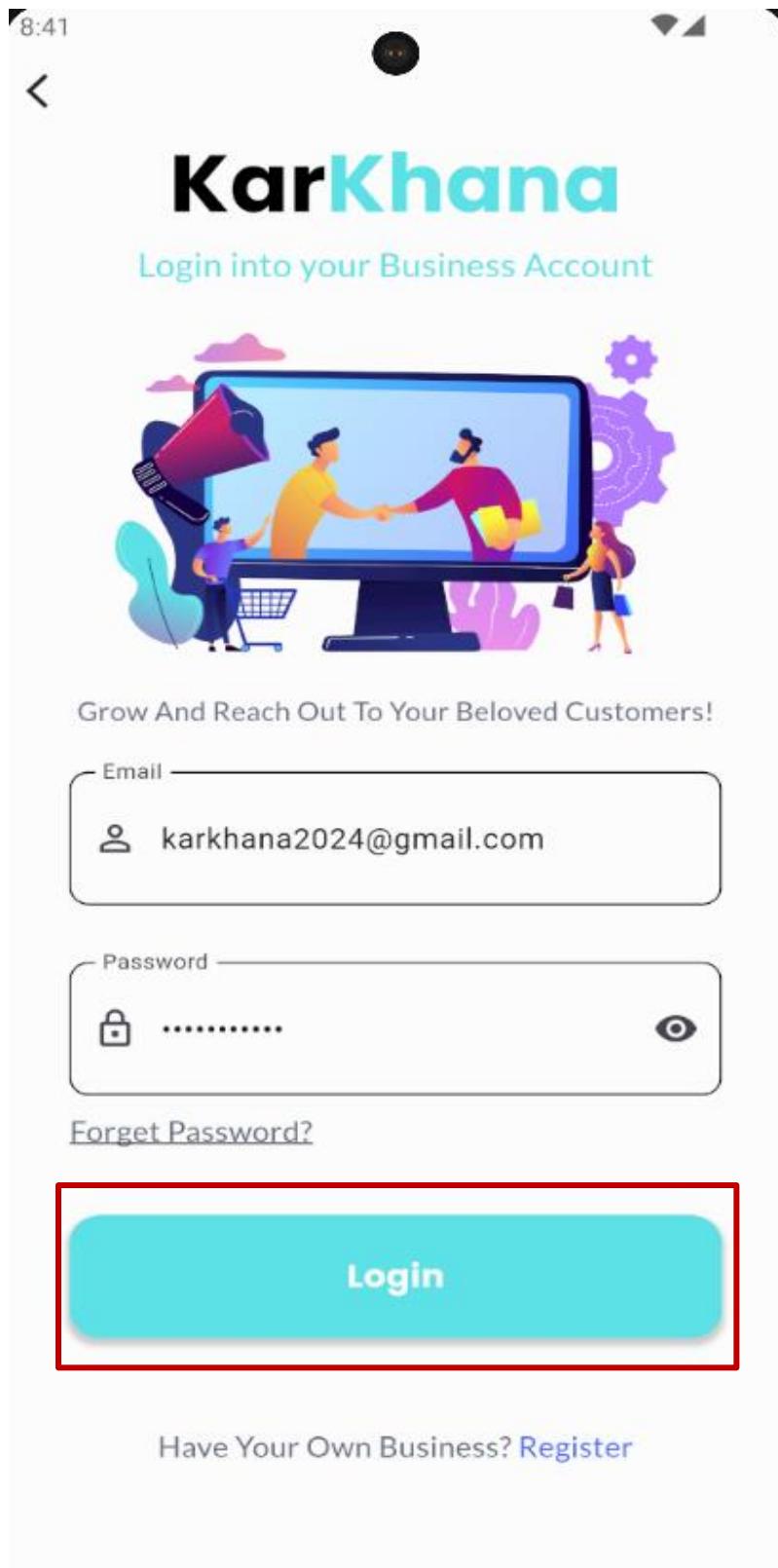


Figure 348: System walkthrough testing table for vendor -2.

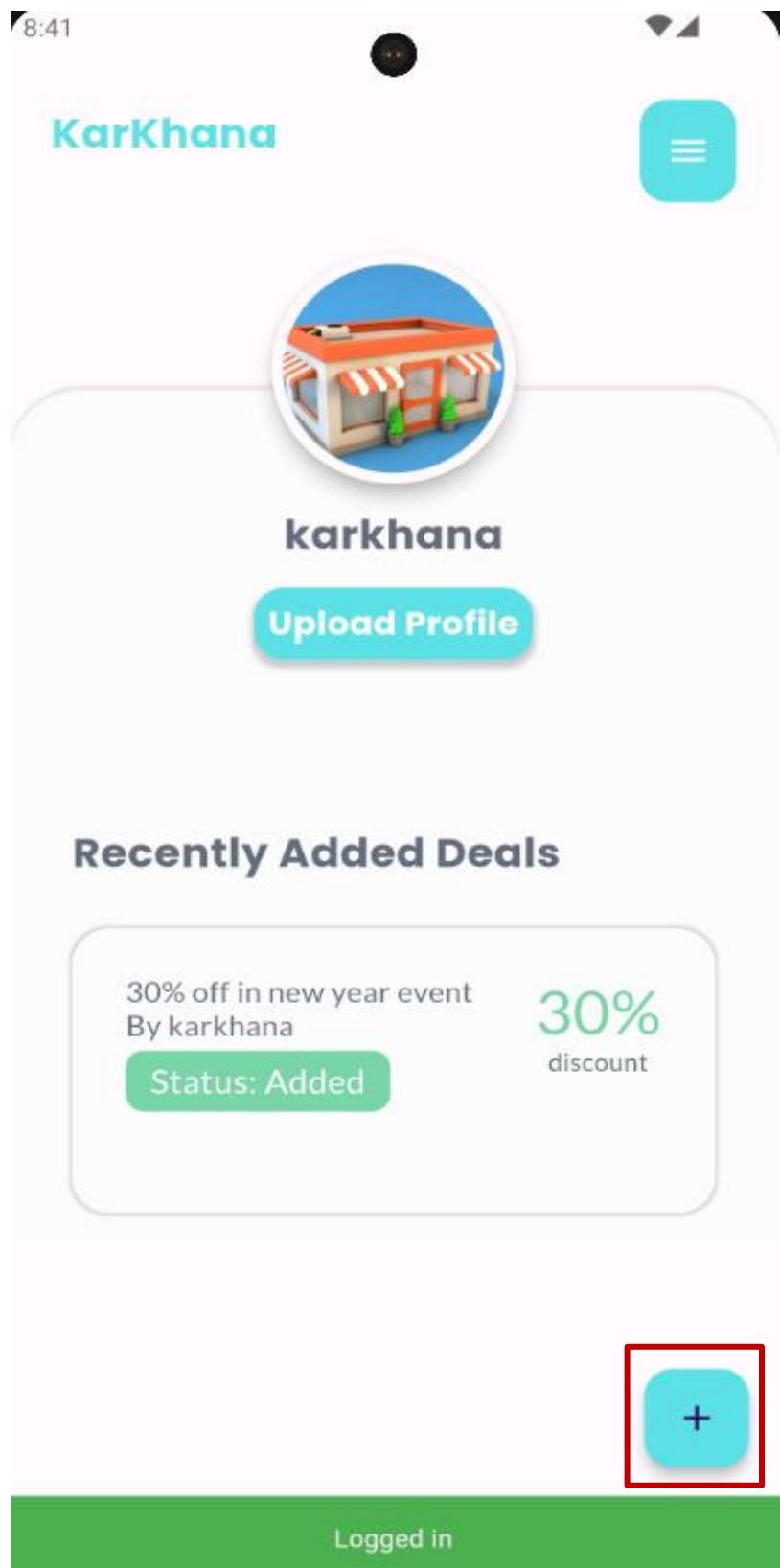


Figure 349: System walkthrough testing table for vendor -3.

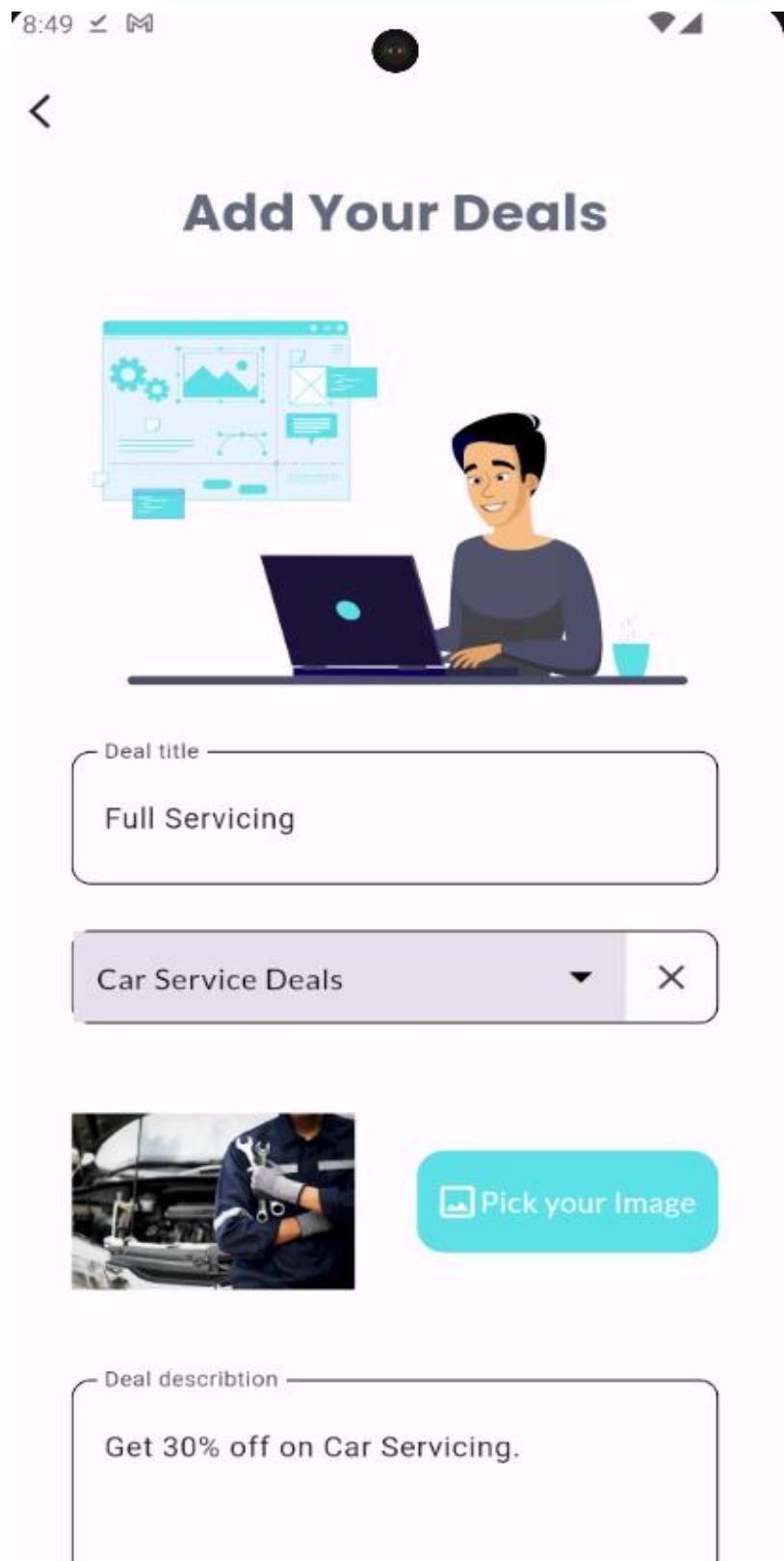


Figure 350: System walkthrough testing table for vendor -4.

A screenshot of a smartphone displaying a mobile application for posting deals. The screen shows the following fields:

- Deal condition: Use before it expires.
- Deal discount: 30
- Market Price of item/service: 1500
- Deal applicable time: 10AM-2PM
- Deal applicable day: Everyday
- Deal expiry date: 2024-04-30

At the bottom is a large teal button with the text "Post Your Deal". This button is highlighted with a red rectangular border.

Figure 351: System walkthrough testing table for vendor -5.

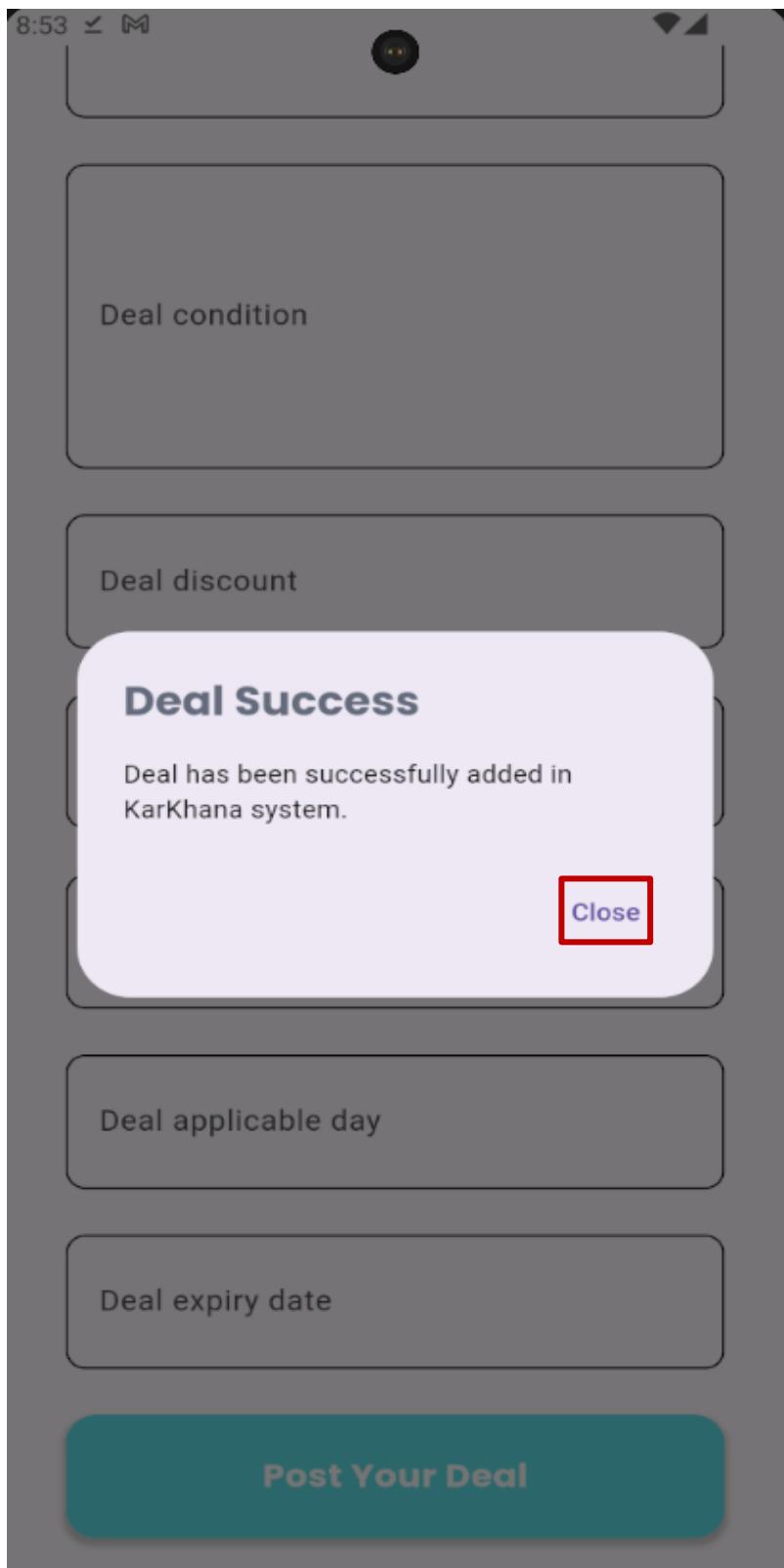


Figure 352: System walkthrough testing table for vendor -6.

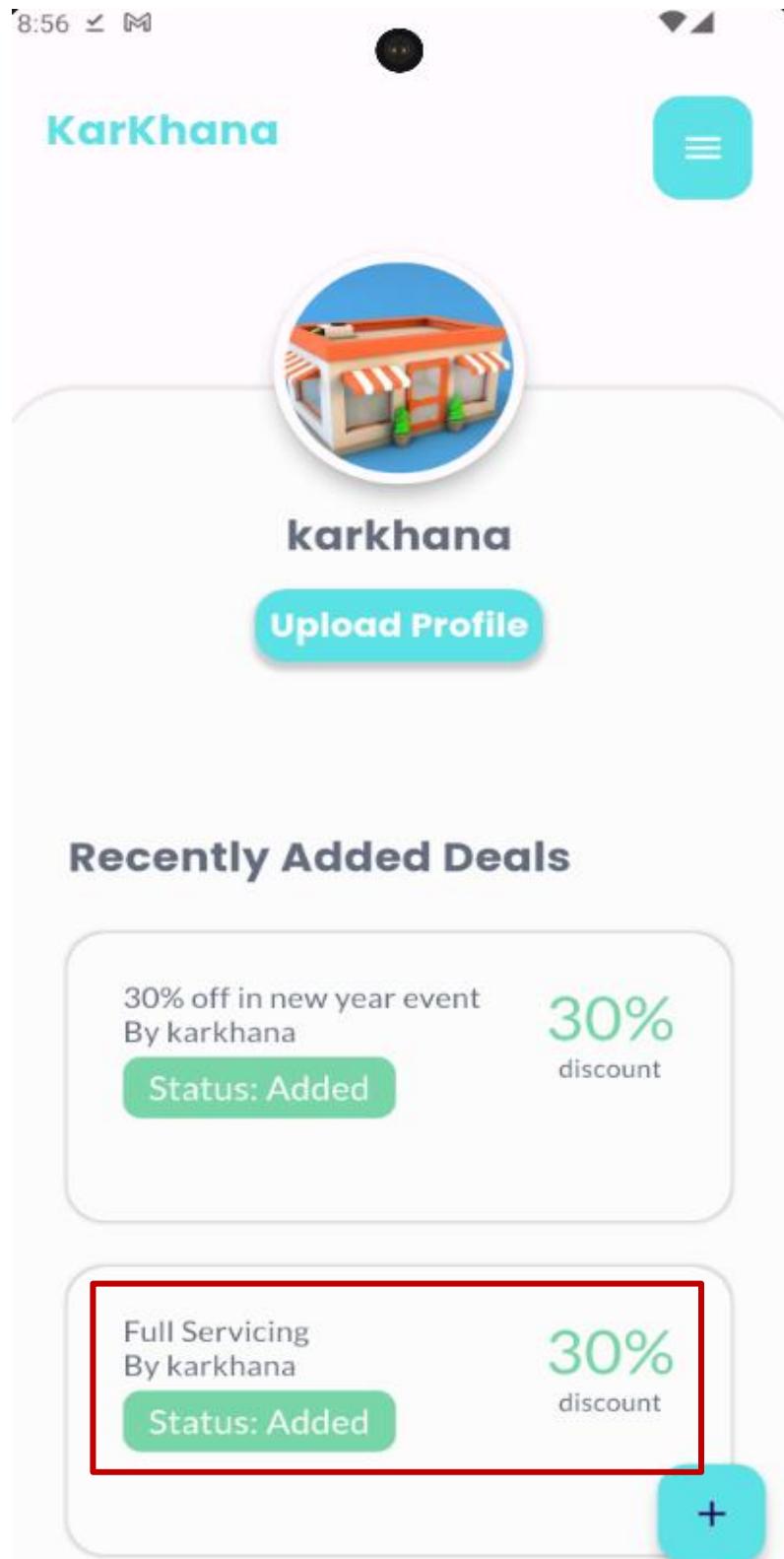


Figure 353: System walkthrough testing table for vendor -7.



Figure 354: System walkthrough testing table for vendor -8.

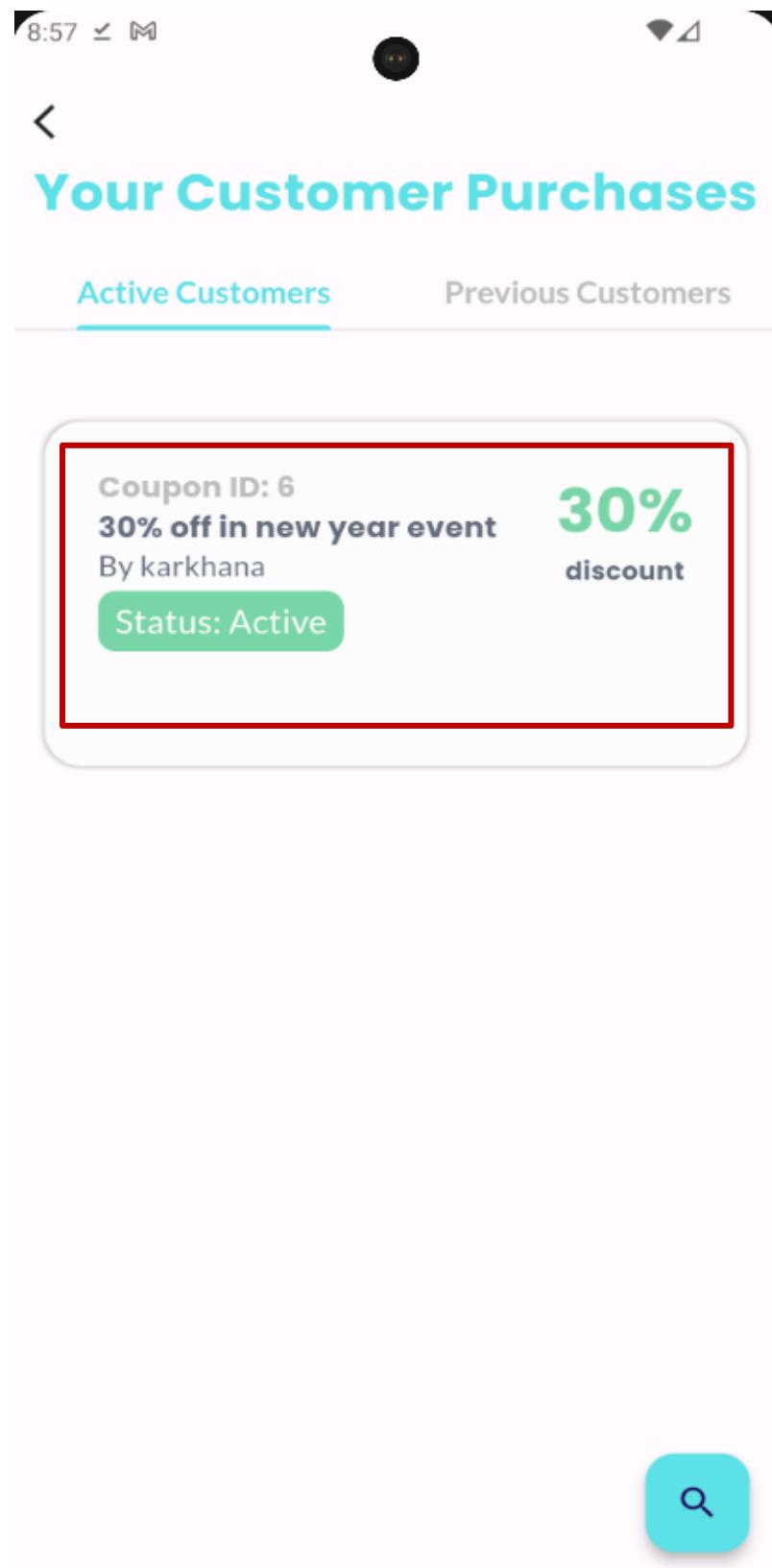


Figure 355: System walkthrough testing table for vendor -9.



Figure 356: System walkthrough testing table for vendor -10.

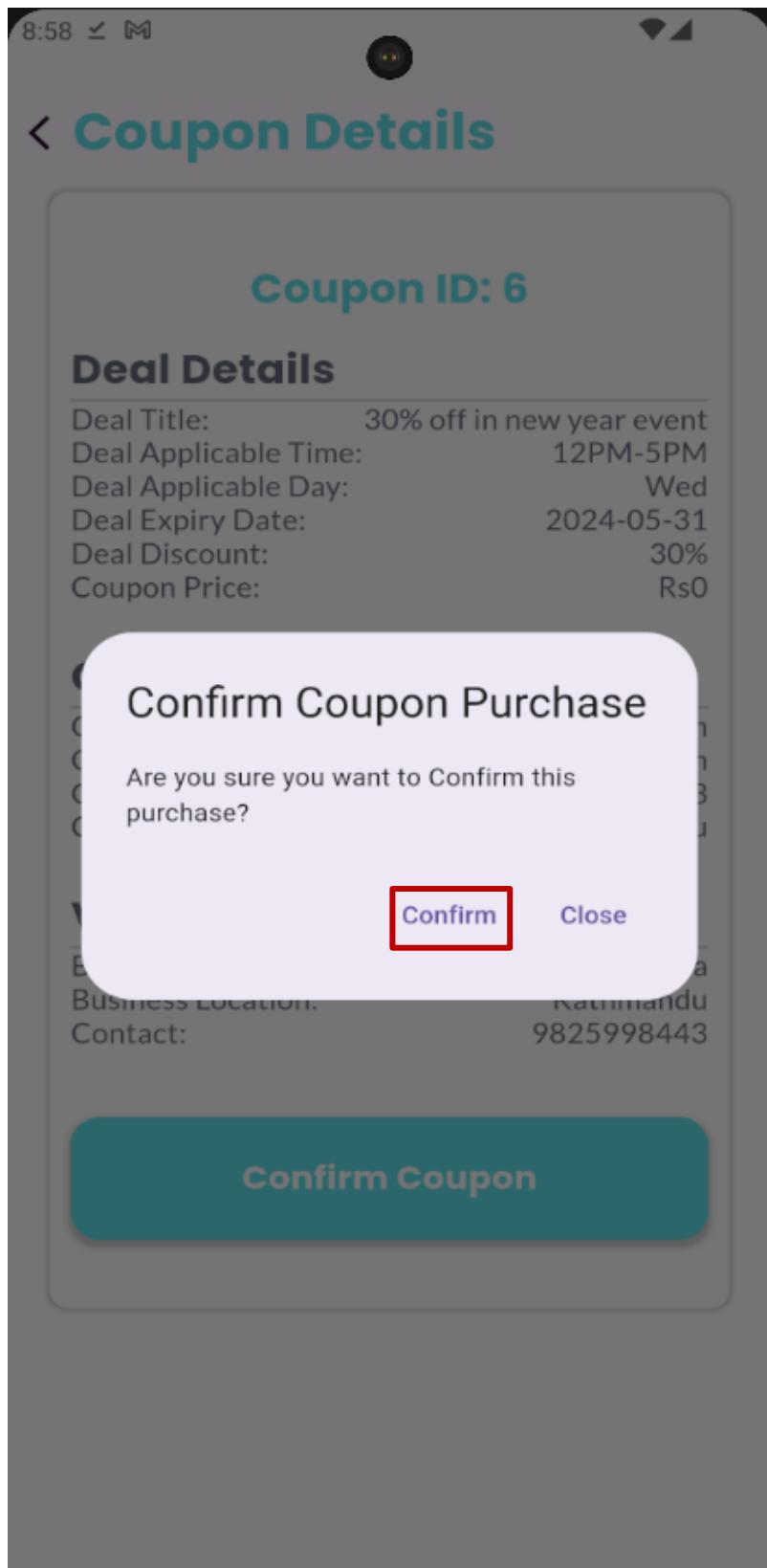


Figure 357: System walkthrough testing table for vendor -11.



karkhana2024@gmail.com

to me ▾

Your Coupon for deal 30% off in new year event has been used.

Reply

Forward



Figure 358: System walkthrough testing table for vendor -12.

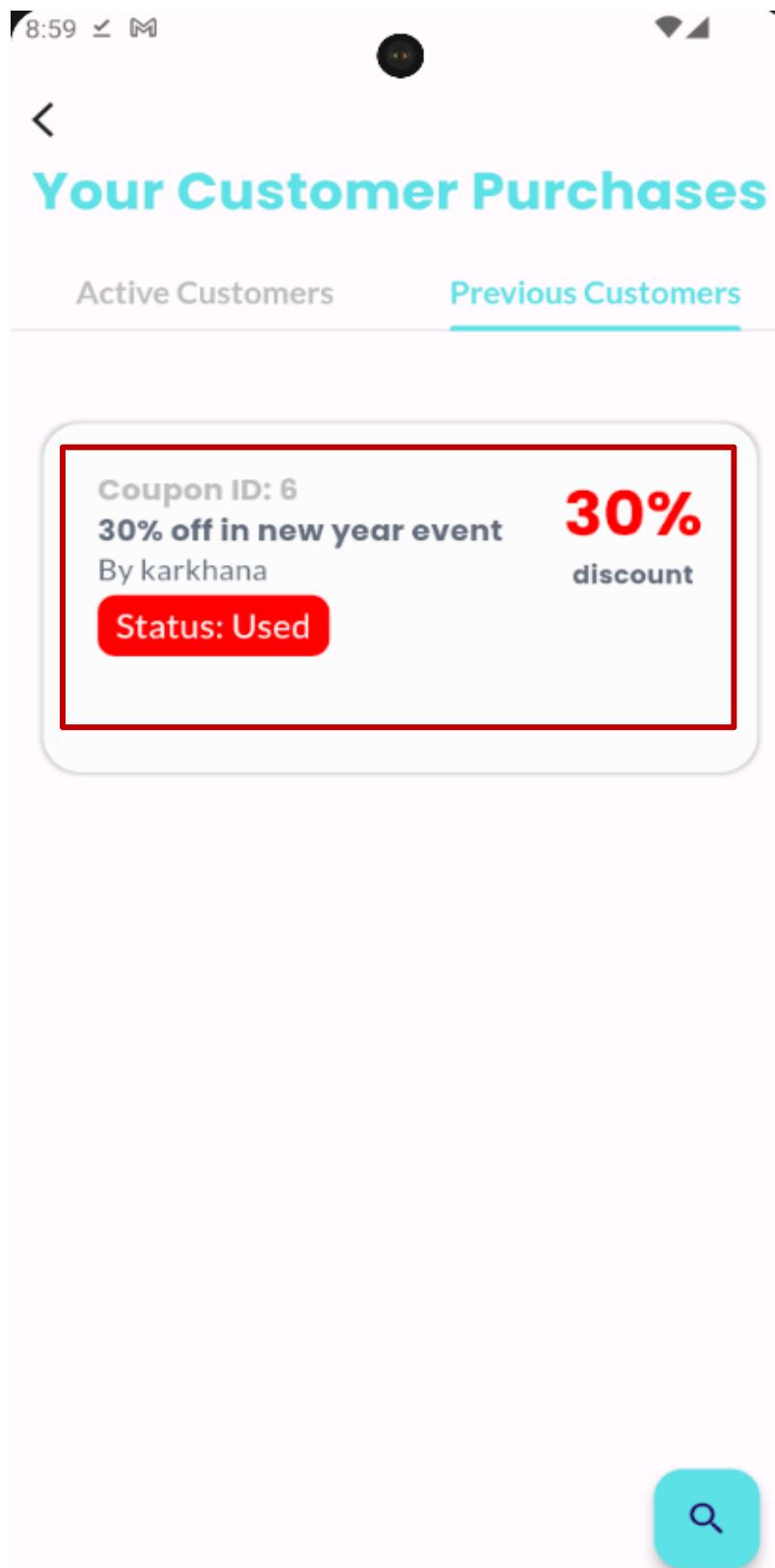


Figure 359: System walkthrough testing table for vendor -13.

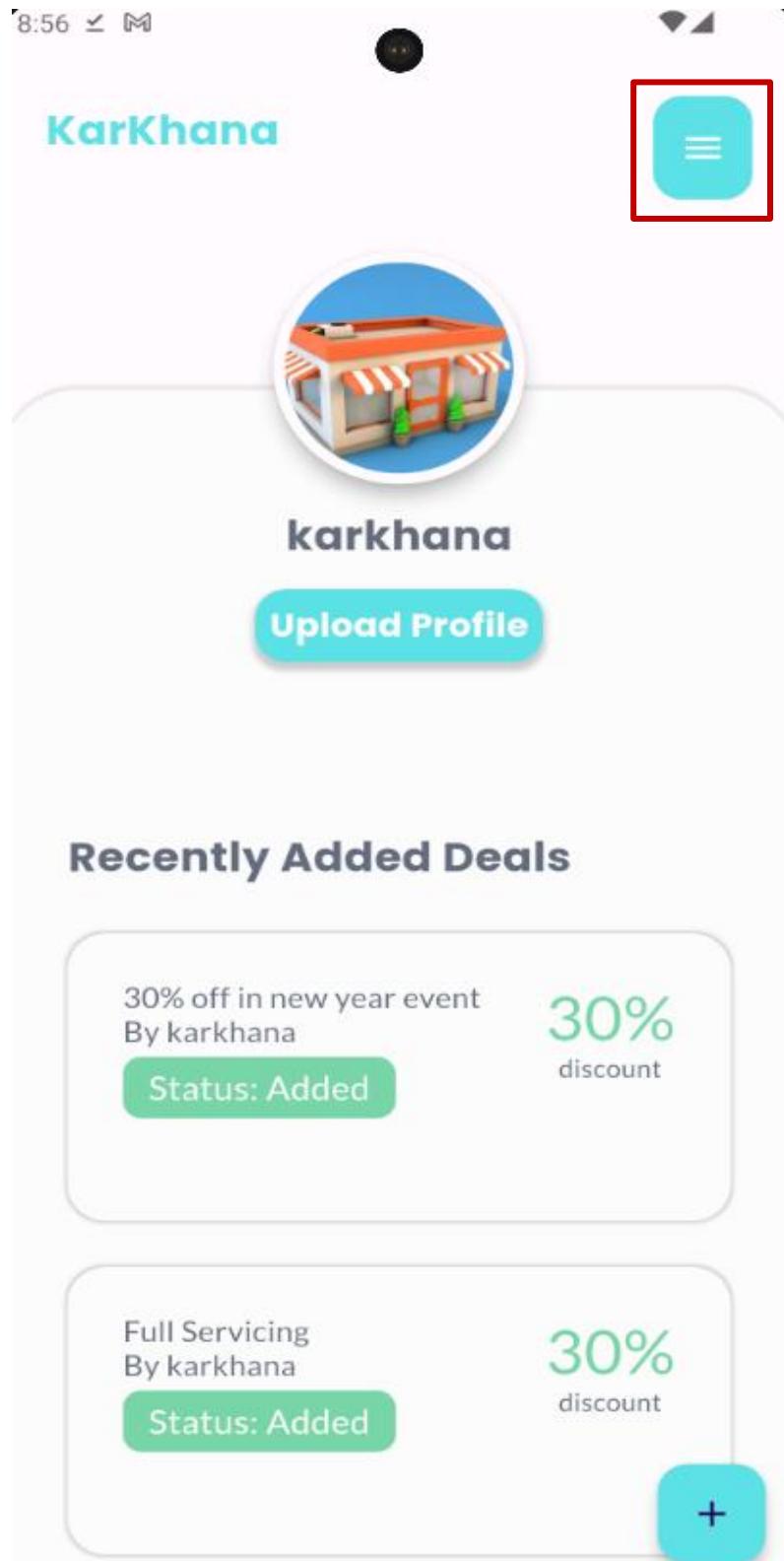


Figure 360: System walkthrough testing table for vendor -14.



Figure 361: System walkthrough testing table for vendor -15.

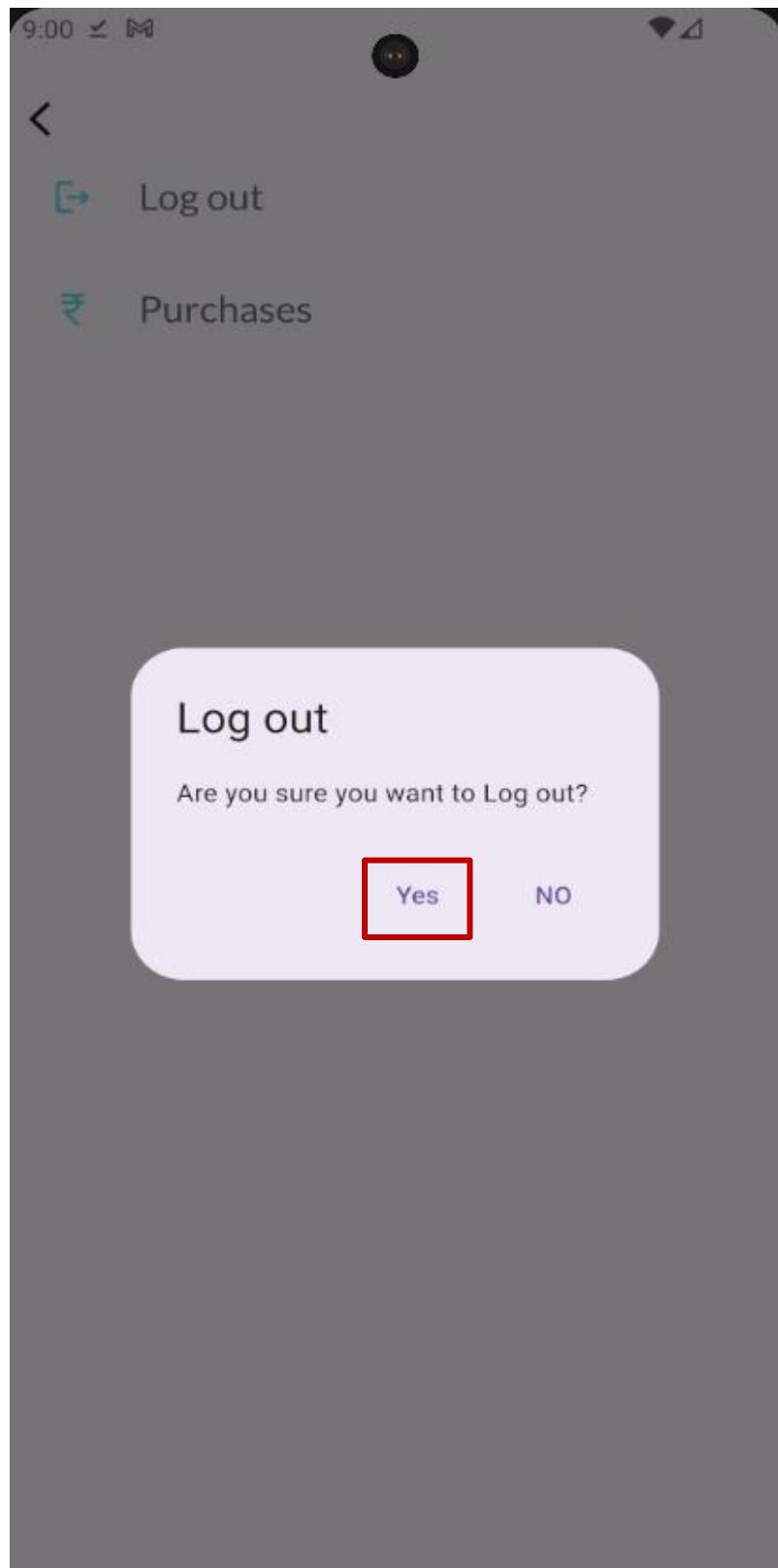


Figure 362: System walkthrough testing table for vendor -16.

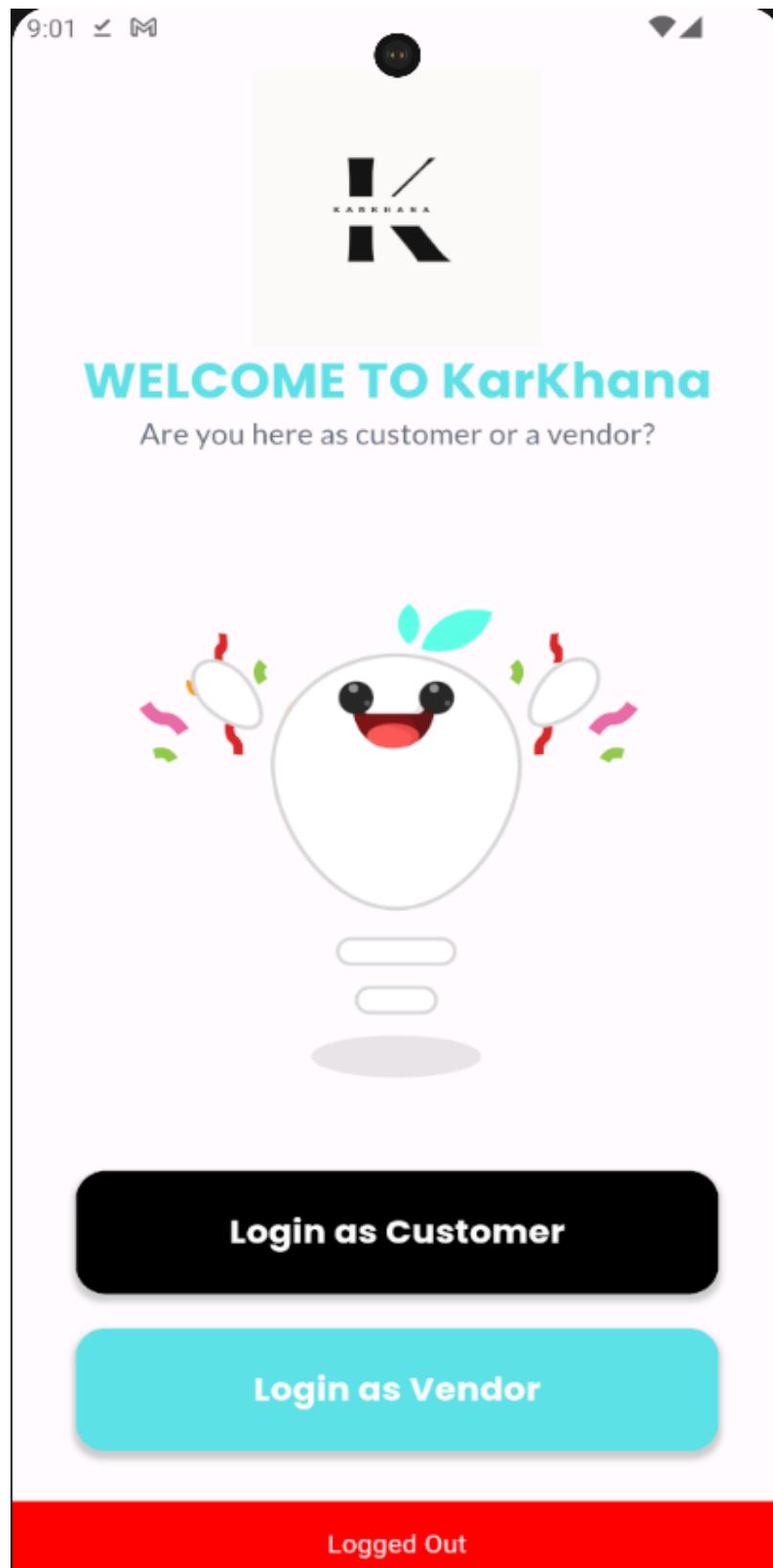


Figure 363: System walkthrough testing table for vendor -17.

## 4.4 Critical Analysis

---

### 4.4.1 Development with Methodology

---

My recent software development experience has taught me the importance of using development strategies to ensure that a project is completed successfully. I've learned that software engineering is more than just creating code; it's a systematic, well-thought-out process that begins with meticulous planning and system architecture. In fact, if one wishes to complete a work on time, good time management is an essential skill that should not be overlooked. Planning and feature development are prioritized, resulting in a more efficient coding process and a shorter overall development time. Iterations are critical throughout development because they improve the system's overall usefulness and stability. Surveys done before and after a project is completed are critical for identifying needs, updating current requirements, and ensuring the project's success.

### 4.4.2 Testing Analysis

---

Testing is an essential part of the KarKhana mobile application development process, since it ensures dependability, security, and functioning. The application's key functionality, including customer and vendor registration, login, showing current and past coupons, adding products to a cart, deal near me suggestions, and deal searching, have been designed and rigorously tested. Furthermore, the user experience and interface have been thoroughly tested to guarantee that consumers can utilize them easily. Special care has been taken to provide secure and adaptable payment solutions, with an emphasis on smooth payment integration.

During development, just a few unresolvable difficulties were discovered. Given additional time, these difficulties can be solved in future modifications of the application. The registration and login processes have been thoroughly tested and proven to be functional, with exception handling consistent with industry norms. A consistent and responsive user experience across all devices has been delivered, resulting in increased productivity. Permission management mechanisms are working as expected. It is vital to note that the program requires an active internet connection, which has been carefully tested and proven to work properly. These elements work together to ensure that users have easy access to all of the functionality available through the KarKhana mobile application.

## CHAPTER 5: OVERALL CRITICAL ANALYSIS

The KarKhana project is a functioning mobile application that offers sellers and customers an easy platform for participating in discounts and promotions. The documentation comprehensively details the system's design, implementation, and testing procedures. One of KarKhana's primary features is its ability to effectively integrate online payment processing, search capabilities, and location-based services, resulting in a smooth and practical user experience. This link allows suppliers to reach a larger audience and give clients a variety of incentives.

The project documentation demonstrates that the development team tackled the project in a methodical and structured manner. The use of a certain technique displays a dedication to best practices and completing the project on time and within budget. The testing procedure looks to have been extensive, with detailed test cases and real-world unit and system testing.

Measures have been taken to ensure the system's security, including protections to secure user data and prevent unwanted access. The documentation also includes details on the system architecture and code organization, which will be useful for future maintenance and updates.

The KarKhana system was developed utilizing the Django framework for the backend and the Dart programming language for the frontend, as described in the released documentation. The documentation illustrates the benefits of each technology and how they were used to achieve the project's goals, demonstrating the development team's technical competency and talents.

There may be possibilities to modify or change the system's technical stack to increase scalability and performance as the user base expands. Furthermore, the documentation may benefit from further information on any specific issues or solutions identified while employing these technologies, as well as how they were addressed. This knowledge would be useful for future development teams that want to employ similar technology to build comparable solutions.

## CHAPTER 6: CONCLUSION

KarKhana is a smartphone application that provides value to Nepalese customers and businesses. This platform allows companies to promote discounts and offers to clients, and users may join up and search for deals, offers, discounts, and sales at various KarKhana-registered businesses. The software seeks to foster a mutually beneficial relationship by allowing companies to attract new consumers while giving users with a quick and user-friendly approach to find savings on a wide range of services.

One of the primary advantages of KarKhana for users is the ability to find large savings at local companies. Users may purchase offers with digital payment methods, making the procedure quick and convenient. Customers can also redeem their bought bargains by visiting companies directly through the app, which improves the overall user experience. Therefore, KarKhana is a wonderful resource for people wishing to save money while supporting local companies.

KarKhana allows businesses to acquire new clients with appealing discounts and promotions. Businesses who join the KarKhana platform may also post tempting bargains, such as happy hour specials, to generate sales during off-peak hours and boost their exposure. Businesses may create dynamic discount programs with various time constraints to better serve their customer base.

KarKhana has the potential to provide enormous value to both customers and businesses in Nepal. The program offers a simple and user-friendly platform for identifying and providing discounts, making it an excellent tool for promoting local businesses and boosting the Nepalese economy.

KarKhana is a promising platform with the ability to benefit both customers and businesses in Nepal. With its user-friendly layout and simple payment options, the app has the potential to become a go-to platform for identifying and providing savings on a wide range of services. KarKhana, when correctly implemented, can alter the Nepalese market by boosting economic growth and helping local enterprises.

---

## 6.1 LEGAL, SOCIAL AND ETHICAL ISSUES

---

### 6.1.1 Legal Issues

KarKhana, being a deals application in Nepal, must be aware of a variety of regulatory limitations. First and foremost, they must verify compliance with the Data Protection Act, which governs the acquisition and use of personal data. This includes gaining consent for data collection, guaranteeing data security, and granting individuals access to and management over their personal information. To assure secure and legitimate payment processing, KarKhana must follow relevant e-payment norms and regulations, such as the Payment and Settlement Systems Act. They would have to follow the Consumer Protection Act, which requires providing accurate information, upholding agreements, and issuing refunds or compensation as needed.

Fourth, KarKhana would need to protect their intellectual property while not infringing on the intellectual property rights of others. They must have clear and enforceable contracts with their vendors and customers. KarKhana must comply with tax legislation and register for any applicable taxes on their revenues, as well as ensure that suppliers meet their tax requirements. By complying to these legal issues, KarKhana may operate ethically, gain confidence from users and business partners, and contribute positively to the local economy.

### 6.1.2 Social Issues

KarKhana, being a deals application based in Nepal, might face several social issues that must be handled. One major barrier is accessibility, as a substantial percentage of the population may not have access to the internet or cellphones, restricting their capacity to use the platform. Furthermore, the digital gap between urban and rural communities has the potential to worsen accessibility concerns. KarKhana must guarantee that their platform is accessible to Nepali speakers who may not be fluent in English since language hurdles might be an issue. Concerns may also be raised about the sorts of deals and businesses listed on the site, as well as the possibility of supporting unethical behavior.

KarKhana must consider the impact it may have on local companies. There is a possibility that the platform may lead to a pricing war or disadvantage smaller companies in contrast to larger enterprises. KarKhana must guarantee that their platform is inclusive and doesn't discriminate against any one group. This might include making the site accessible to persons with disabilities, ensuring that discounts are provided to people from all socioeconomic levels, and refraining from discriminatory marketing methods. By tackling these societal challenges and stressing inclusion, KarKhana can work to build a platform that benefits all stakeholders while also benefiting the local community and economy.

#### **6.1.3 Ethical Issues**

---

KarKhana, as a deals application operating in Nepal, must be aware of several potential ethical issues. It must verify that customer data is collected and used responsibly, in accordance with clear data protection requirements. KarKhana must offer its customers fair and transparent pricing, and it must not push products or services that are harmful to society or the environment. While picking suppliers or promoting transactions, the application must be honest and prevent unethical or discriminating behavior. Fourth, KarKhana must maintain fair and ethical partnerships with merchants and offer clear information about how their products or services will be marketed.

The application must provide exceptional customer service and rapidly resolve any complaints or concerns. KarKhana should encourage corporate social responsibility by supporting locally owned enterprises, ecologically friendly goods or services, and community engagement through charitable donations or volunteer labor. By addressing these ethical concerns, KarKhana can operate in an ethical and socially responsible manner, helping to create a more sustainable and fair economy in Nepal for both consumers and sellers.

---

## 6.2 ADVANTAGES

---

Some of the advantages of the KarKhana system are described below:

- i. **Access to exclusive deals:** Customers can access special offers and discounts on items and services they might not have otherwise been able to afford.
- ii. **Increased sales and revenue for vendors:** KarKhana provides a platform for vendors to market their businesses and reach a larger audience, thereby boosting sales and income.
- iii. **Convenient and easy to use:** The KarKhana application is user-friendly and easy to navigate, making it simple for users to browse and purchase discounts.
- iv. **Secure payment system:** KarKhana offers an e-payment mechanism for coupon purchases, ensuring that consumer transactions are safe and secure.
- v. **Location-based deals:** The application's current location tracker feature allows users to discover deals and promotions near their current location, facilitating the finding of local deals.
- vi. **Notifications:** KarKhana's notification feature sends consumers information on their purchases and when their coupons have been used, keeping them informed and up to date.
- vii. **Time-limited deals:** KarKhana offers time-limited discounts, creating a sense of urgency for buyers to avail the deal before it expires, thereby potentially increasing sales and revenue for vendors.
- viii. **Customer loyalty:** By providing special offers and incentives to repeat customers, KarKhana can help establish customer loyalty and encourage them to return for future purchases.
- ix. **Environmental benefits:** KarKhana can contribute to reducing the environmental impact of consumption by promoting environmentally friendly products or services, thereby fostering a more sustainable economy.

### 6.3 LIMITATIONS

---

Some of the limitations of the KarKhana system are:

- i. **Limited Availability:** KarKhana may not be available in every region, which can restrict the number of potential customers and sellers who can participate.
- ii. **Limited deal options:** Customers may have limited options if the application does not offer discounts on all items or services.
- iii. **Limited vendor selection:** The application may have a restricted number of suppliers, limiting the variety of items or services available for purchase by customers.
- iv. **Dependence on technology:** KarKhana relies on technology, which may lead to downtime or technical malfunctions, disrupting the user experience.
- v. **Inaccurate deal information:** Customers may become dissatisfied and lose trust if the application contains inaccurate or deceptive offer information.
- vi. **Privacy concerns:** KarKhana's collection and use of client data may raise privacy concerns, particularly if data is not adequately protected or handled ethically.
- vii. **Lack of customer support:** The application may not provide adequate customer assistance, leading to dissatisfaction and potential loss of customers.
- viii. **Dependence on customer loyalty:** KarKhana's success may rely on customer loyalty, which can be challenging to maintain in a highly competitive industry.

## 6.4 FUTURE WORK

---

KarKhana has various potential areas for expansion in the future. One such avenue is leveraging machine learning and artificial intelligence to offer increasingly personalized suggestions and deals. By utilizing AI to tailor offers and promotions based on user activity, interests, and past purchases, KarKhana could enhance user engagement and loyalty.

Integrating social components such as shared lists and connections between users with similar interests could further enhance community participation and user loyalty. Implementing real-time assistance and chatbots for improved customer support would also be beneficial. Additionally, incorporating more user comments and reviews would provide valuable insights for providers regarding consumer preferences.

KarKhana could also explore new partnerships and collaborations with other companies or platforms. For example, integrating with other e-commerce sites or social media networks could allow users to access deals and promotions directly from those platforms. Moreover, collaborating with local businesses or service providers to offer users unique deals and promotions on a variety of products and services could be advantageous.

To summarize, some potential new features for KarKhana to implement in the future include:

- i. Use of machine learning and AI for personalized suggestions and promotions.
- ii. Vendor profile review and ratings.
- iii. QR code generator for coupons.
- iv. Implementation of chat feature to connect customers and vendors directly for real-time support.

Some existing features that could be enhanced in the future include:

- i. Improving the location-based recommendation system with more AI-based capabilities.
- ii. Enabling sharing of deals, not just coupons.

- iii. Adding more payment options.
- iv. Expanding the range of deal categories and locations within the system.

By staying ahead of the curve and innovating, KarKhana can continue to be a valuable tool for both consumers and vendors, driving sales and growth while providing users with savings and value.

How to implement Future work is described more in below appendix.

## CHAPTER 7: BIBLIOGRAPHY

Bryant, N., 2023. *Havard*. [Online]

Available at: <https://www.pon.harvard.edu/daily/negotiation-skills-daily/for-a-mutually-beneficial-agreement-collaboration-is-key/>

[Accessed 13 12 2023].

dart.dev, 2023. *Dart*. [Online]

Available at:

<https://dart.dev/overview#:~:text=Dart%20is%20a%20client%2Doptimized,runtime%20platform>

[Accessed 13 12 2023].

Django Community, 2023. *Django*. [Online]

Available at: <https://www.djangoproject.com/community/logos/>

[Accessed 13 December 2023].

eneko, 2021. *Github*. [Online]

Available at: <https://github.com/eneko/RegEx>

[Accessed 13 December 2023].

Google, 2013. *Wikimedia*. [Online]

Available at: <https://commons.wikimedia.org/wiki/File:Sqlite-square-icon.svg>

[Accessed 13 December 2023].

Google, 2019. *wikimedia*. [Online]

Available at: [https://commons.wikimedia.org/wiki/File:Dart\\_logo.png](https://commons.wikimedia.org/wiki/File:Dart_logo.png)

[Accessed 13 12 2023].

Khalti Digital Wallet, 2018. *Brands of the World*. [Online]

Available at: <https://www.brandsoftheworld.com/logo/khalti>

[Accessed 13 December 2023].

Oracle, 2023. *Oracle*. [Online]

Available at: <https://www.oracle.com/database/what-is-database/>

[Accessed 28 12 2023].

simc, 2023. *Github*. [Online]

Available at: <https://github.com/isar/hive>

[Accessed 13 December 2023].

Wikipedia, 2008. *Wikipedia*. [Online]

Available at: <https://en.wikipedia.org/wiki/File:Python-logo-notext.svg>

[Accessed 13 12 2023].

Wikipedia, 2024. *Wikipedia*. [Online]

Available at:

<https://en.wikipedia.org/wik...text=A%20prototype%20is%20an%20early,by%20system%20analysts%20and%20users.>

[Accessed 17 March 2024].

## CHAPTER 8: APPENDIX

### 8.1 APPENDIX 1: PRE-SURVEY

#### 8.1.1 PRE-SURVEY FORM

##### KarKhana

B I U ↲ ↳

KarKhana is a mobile application for people and businesses in Nepal which enables businesses to showcase their discounts and offers to customers/users. Users can easily sign up to buy deals from places like restaurants and salons. The app is super easy to use - just enter your phone number to get started. KarKhana helps users find awesome deals and lets businesses promote special offers. It's all about making a great marketplace for deals that benefits both customers and businesses in Nepal.

Email \*

Valid email address

This form is collecting email addresses. [Change settings](#)

Do you like discounts and sales? \*

Yes

No

Maybe

Figure 364: Pre-Survey form question screenshot-1.

Are you familiar using this type of deals related application? \*

Yes

No

On what scale do you rate the difficulty of getting discounts on your favourite items or services? \*

1

2

3

4

5

Easy



Difficult

How often do you search for discounted item and services? \*

Daily

Weekly

Monthly

Yearly

Figure 365: Pre-Survey form question screenshot-2.

What kind of experience did you have while searching for deals? \*

- Very Good
- Good
- Bad
- Worst
- Awful

...

How important is to give discounts to customers for any type of businesses? \*

- Very Important
- Somewhat important
- Less important
- Not at all

...

What kind of deals are you more interested in? \*

- Items like food
- Services like dental, futsal, spas, carwash etc.
- Other...

Figure 366: Pre-Survey form question screenshot-3.

Do you agree this application will make the new deals/offers either near or far location \*  
more reachable to you?

- Strongly Agree
- Maybe
- Disagree

How often do you get notify about any new deals offered near you or at other location? \*

- Very fast
- Rarely
- Only after the deal date expires
- Not at all

Would you like to buy and share all types of deals for your family and friends to enjoy too? \*

- Yes
- No
- Maybe

Figure 367: Pre-Survey form question screenshot-4.

Do you think this application will help in growth of businesses by the offers/deals they provide even to their unreachable customers through this application? \*

- Yes
- No
- Maybe

...

Feedback\*

Long-answer text

Figure 368: Pre-Survey form question screenshot-5.

### 8.1.2 Sample of Filled Pre-Survey Forms

## KarKhana

KarKhana is a mobile application for people and businesses in Nepal which enables businesses to showcase their discounts and offers to customers/users. Users can easily sign up to buy deals from places like restaurants and salons. The app is super easy to use - just enter your phone number to get started. KarKhana helps users find awesome deals and lets businesses promote special offers. It's all about making a great marketplace for deals that benefits both customers and businesses in Nepal.

nahsinahtserhs@gmail.com [Switch accounts](#)



\* Indicates required question

Email \*

np01cp4a210202@islingtoncollege.edu.np

Do you like discounts and sales? \*

- Yes
- No
- Maybe

Figure 369: Pre-Survey filled form screenshot-1.

Are you familiar using this type of deals related application?

Yes

No

[Clear selection](#)

On what scale do you rate the difficulty of getting discounts on your favorite items or services?

1

2

3

4

5

Easy

Difficult

[Clear selection](#)

How often do you search for discounted item and services?

Daily

Weekly

Monthly

Yearly

[Clear selection](#)

Figure 370: Pre-Survey filled form screenshot-2.

What kind of experience did you have while searching for deals?

- Very Good
- Good
- Bad
- Worst
- Awful

[Clear selection](#)

How important is it to give discounts to customers for any type of businesses?

- Very important
- Somewhat important
- Less important
- Not at all important

[Clear selection](#)

Figure 371: Pre-Survey filled form screenshot-3.

What kind of deals are you more interested in?

Items like food  
 Services like dental, futsal, spas, carwash etc.  
 Others

[Clear selection](#)

Do you agree this application will make the new deals/offers either near or far location more reachable to you?

Strongly agree  
 Maybe  
 Might agree  
 Disagree

[Clear selection](#)

Figure 372: Pre-Survey filled form screenshot-4.

How often do you get notify about any new deals offered near you or at other location?

- Very fast
- Rarely
- Only after the deals date expire
- Not at all

[Clear selection](#)

Would you like to buy and share all type of deals for your family and friends to enjoy too?

- Yes
- No
- Maybe

[Clear selection](#)

Figure 373: Pre-Survey filled form screenshot-5.

Do you think this application will help in growth of businesses by the offers/deals they provide even to their unreachable customers through this application?

Yes  
 No  
 Maybe

[Clear selection](#)

Feedback

good concept for a deal application. Best of luck!

[Submit](#)  Page 1 of 1 [Clear form](#)

Figure 374: Pre-Survey filled form screenshot-6.

### 8.1.3 Pre-Survey Result

We are glad to inform that the pre-survey was a success, with useful input and insights that assisted us in refining our system design and functioning. Their feedback will assist us in developing a system that actually satisfies the requirements and expectations of our clients.

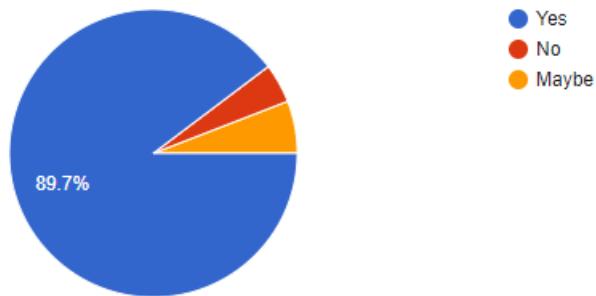
**Questions and response summary from the survey are as follows:**

i. Response 1

Do you like discounts and sales?

 Copy

68 responses



*Figure 375: Response 1.*

As the whole system is related to discounts and deals so, first question for the survey was to get idea if the users would like discounts or not. In response, 89.7% of people liked discount, 5.9% disliked and 4.4% of people were not sure.

## ii. Response 2

Are you familiar using this type of deals related application?

 Copy

68 responses

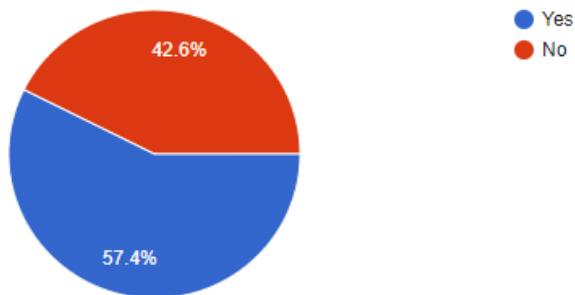


Figure 376: Response 2.

Respondents were also asked if they were familiar with any deals related application like the KarKhana system to know how differently, our system can be built from others. Here, 57.4% seems familiar and 42.6% of them seems unfamiliar with this kind of system.

## iii. Response 3

On what scale do you rate the difficulty of getting discounts on your favourite items or services?

 Copy

68 responses

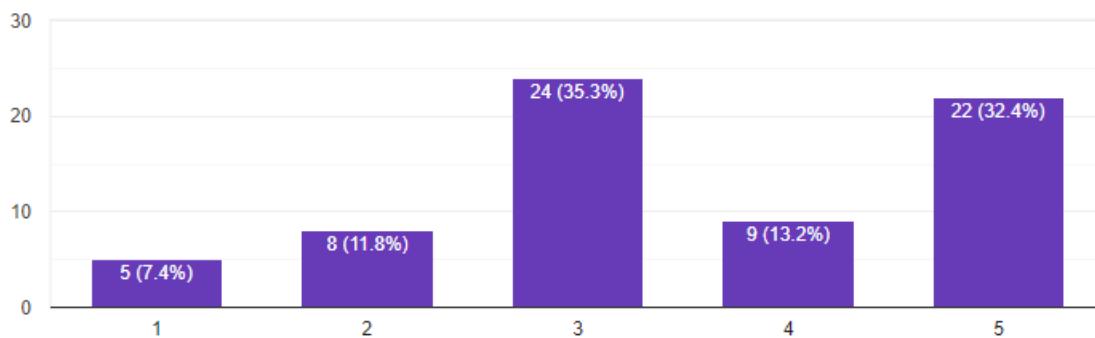


Figure 377: Response 3.

Respondents were asked to rate the difficulty of getting discount in the market. 22 people i.e., 32.4% of them rated 5 which determines it is very difficult to bargain discounts, 9 people i.e., 13.2% of them rated 4, some find it in neutral state as 24 people i.e., 35.3% of them rated 3. At last rating 1 and 2 are 5 people i.e., 7.4% of them and 8 people i.e., 11.8% of them finds to easy to find discounts.

#### iv. Response 4

How often do you search for discounted item and services?

 Copy

68 responses

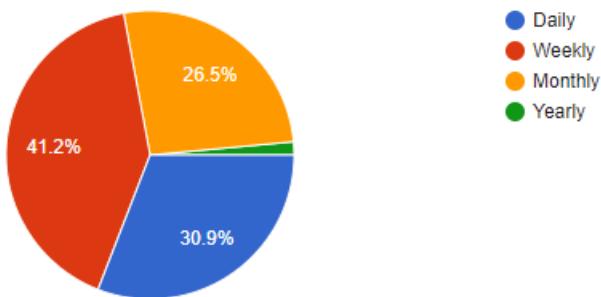


Figure 378: Response 4.

Respondents were asked how much they look out for discounts in market. And from this. 41.2% of them search weekly, 26.5% of them search monthly, 30.9% of them search daily and 1.5% of them search yearly.

#### v. Response 5

What kind of experience did you have while searching for deals?

 Copy

68 responses

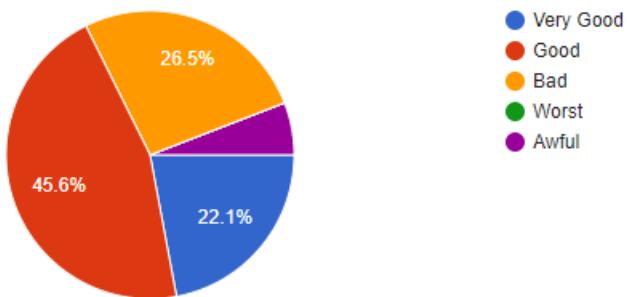


Figure 379: Response 5.

Respondents were asked about their experience while they search for any discounts in their favorites items or service. 45.6% of them responded they had good experience, 26.5% of them had bad experience, 22.1% of them had very good experience and 5.9% had awful experience.

#### vi. Response 6

How important is to give discounts to customers for any type of businesses?

 Copy

68 responses

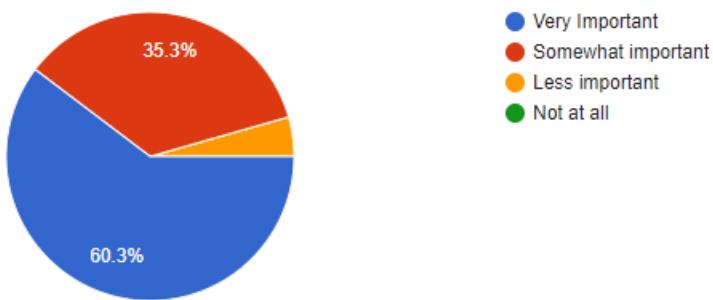


Figure 380: Response 6.

Respondents were asked about the importance of giving discount to customer by any type of businesses. Here, they responded as, 60.3% of them says its very important to give discount, 35.3% of them thinks its somewhat important, 4.4% of them thinks its less important.

#### vii. Response 7

What kind of deals are you more interested in?

 Copy

68 responses



Figure 381: Response 7

As KarKhana will include all type of deals, so respondents were asked about type of discounts they were interested in. 60.3% of them are interested in food deals, 10.3% of them are interested in others and 29.4% of them are interested in service deals.

### viii. Response 8

Do you agree this application will make the new deals/offers either near or far location [!\[\]\(3f5627a3603c9b9d96a8c53aa21b737b\_img.jpg\) Copy](#)

68 responses

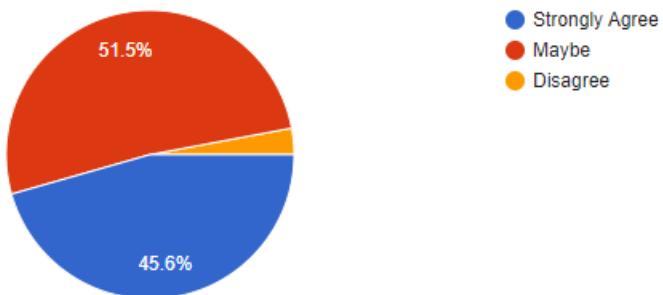


Figure 382: Response 8.

KarKhana will attempt to bring the best deals from 3-4 places, with a priority on launch period, and more locations will be added in the future. As a result, respondents were questioned if they agreed with this application making offers more accessible to customers. In this, 51.5% of them were not sure, 45.6% of them were agreeing strongly.

### ix. Response 9

How often do you get notify about any new deals offered near you or at other location? [!\[\]\(60922df75582df7e643e7995ae6d52e9\_img.jpg\) Copy](#)

68 responses

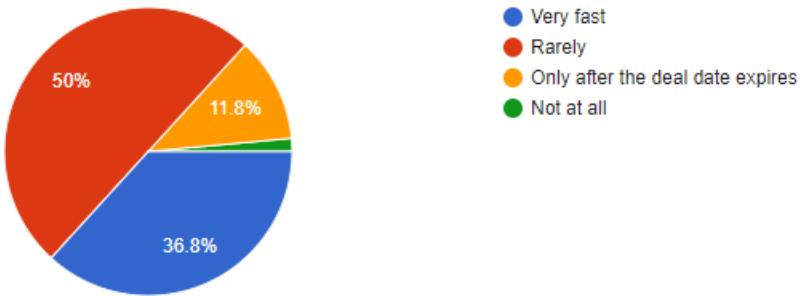


Figure 383: Response 9.

Respondents were asked about how often they get notify about deals and offers. From this, 50% of them were notified rarely, 36.8% of them were notified very fast, 11.8% of them were notified only after the deals date expired.

#### x. Response 10

Would you like to buy and share all types of deals for your family and friends to enjoy too?

 Copy

68 responses

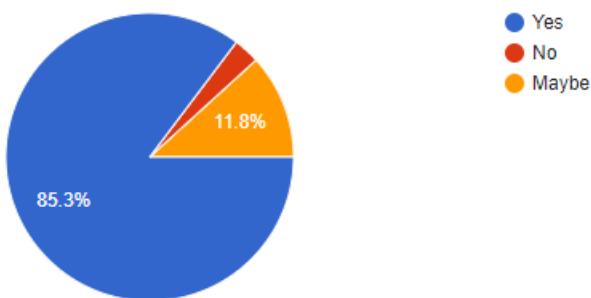


Figure 384: Response 10.

Customers can share coupon with their loved ones through brisk deals so respondents were asked if they would like that kind of features. From this, 85.3% of the liked this kind of feature, 11.8% of them were not sure about the feature.

#### xi. Response 11

Do you think this application will help in growth of businesses by the offers/deals they provide even to their unreachable customers through this application?

 Copy

68 responses

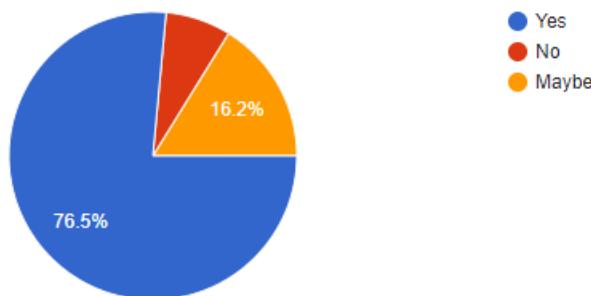


Figure 385: Response 11.

Businesses will be able to post their deals and they can reach many customers in one platform. As deals will reach out to many customers, their business will be promoting itself in the meantime. Therefore, Businesses can grow with the help of KarKhana. So, the respondents were asked if this platform will help business to grow and in response, 76.5% of them agreed, 16.2% of them were not sure.

---

## 8.2 APPENDIX 2: POST-SURVEY

---

### 8.2.1 Post-Survey Form

---

We are glad to report that the post-survey was a success, with excellent input from our users assisting us in identifying areas for development and refinement. We were really thrilled to discover that our users found the system simple to use and liked how quickly and efficiently they could identify and get bargains.

## KarKhana

---

B I U ↲ ✖

KarKhana is a mobile application for people and businesses in Nepal which enables businesses to showcase their discounts and offers to customers/users. Users can easily sign up to buy deals from places like restaurants and salons. The app is super easy to use - just enter your phone number to get started. KarKhana helps users find awesome deals and lets businesses promote special offers. It's all about making a great marketplace for deals that benefits both customers and businesses in Nepal.

---

Name \*

Short-answer text

How do you rate the UI of KarKhana System? \*

1	2	3	4	5	
Very Bad	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very Good

Figure 386: post-Survey form screenshot-1.

How satisfied were you with the Khalti payment option for buying deal coupon? \*

- Very Satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very Disatisfied

Did you find Coupon Sharing feature helpful to share deals with your friends and families? \*

- Yes
- No
- Maybe

Figure 387: post-Survey form screenshot-2.

Did you find Add to cart feature helpful? \*

Very Helpful

Helpful

Not Helpful

...

How easy was it to search deals on KarKhana? \*

1

2

3

4

5

Very Difficult

Very Easy

Figure 388: post-Survey form screenshot-3.

How easy is to post deals on KarKhana? \*

- Very Easy
- Easy
- Difficult
- Very Difficult

How satisfied are you with the KarKhana Mobile application? \*

- Very Satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very Dissatisfied

Figure 389: post-Survey form screenshot-4.

\*\*\*

Would you Recommend our Deal purchasing platform to others? \*

- Yes
- No
- Maybe

Figure 390: post-Survey form screenshot-5.

### 8.2.2 SAMPLE OF FILLED POST-SURVEY FORMS

## KarKhana

KarKhana is a mobile application for people and businesses in Nepal which enables businesses to showcase their discounts and offers to customers/users. Users can easily sign up to buy deals from places like restaurants and salons. The app is super easy to use - just enter your phone number to get started. KarKhana helps users find awesome deals and lets businesses promote special offers. It's all about making a great marketplace for deals that benefits both customers and businesses in Nepal.

\* Indicates required question

Name \*

Jod

How do you rate the UI of KarKhana System? \*

1

2

3

4

5

Very Bad

Very Good

Figure 391: post-Survey filled form screenshot-1.

How satisfied were you with the Khalti payment option for buying deal coupon? \*

- Very Satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very Disatisfied

Did you find Coupon Sharing feature helpful to share deals with your friends and families? \*

- Yes
- No
- Maybe

Figure 392: post-Survey filled form screenshot-2.

Did you find Add to cart feature helpful? \*

Very Helpful

Helpful

Not Helpful

How easy was it to search deals on Karkhana? \*

1

2

3

4

5

Very Difficult

Very Easy

Figure 393: post-Survey filled form screenshot-3.

How easy is to post deals on KarKhana? \*

- Very Easy
- Easy
- Difficult
- Very Difficult

How satisfied are you with the KarKhana Mobile application? \*

- Very Satisfied
- Satisfied
- Neutral
- Dissatisfied
- Very Dissatisfied

Figure 394: post-Survey filled form screenshot-4.

Would you Recommend our Deal purchasing platform to others? \*

- Yes
- No
- Maybe

*Figure 395: post-Survey filled form screenshot-5.*

### 8.2.3 Post-Survey Result

#### I. Response 1

How do you rate the UI of KarKhana System?

 Copy

30 responses

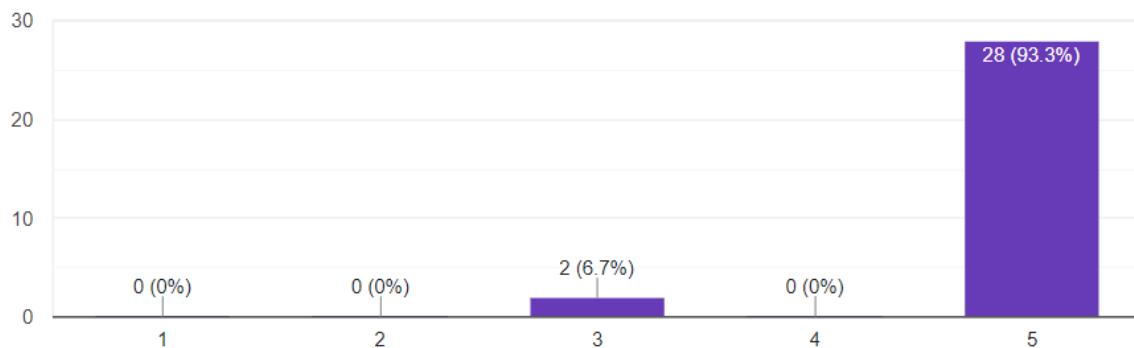


Figure 396: post-Survey response 1.

Respondents were asked to rate the UI design of the KarKhana system. Most of them i.e., 93.3% loved the design by rating 5, 2 of them i.e., 6.7% rated it decent 3 which is neither good nor bad.

#### II. Response 2

How satisfied were you with the Khalti payment option for buying deal coupon?

 Copy

30 responses

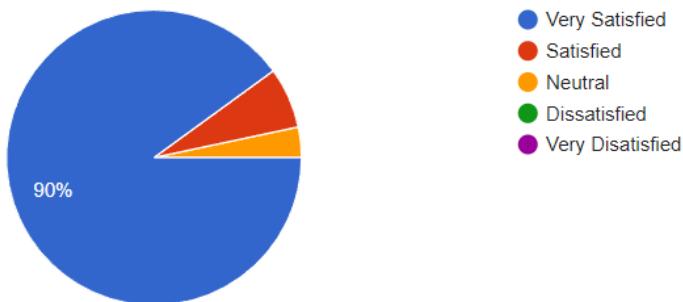


Figure 397: post-Survey response 2.

As Karkhana also provides online payment feature, so the respondents were asked if they were satisfied with the feature. 90% from them were very satisfied about the feature.

### III. Response 3

Did you find Coupon Sharing feature helpful to share deals with your friends and families?

[Copy](#)

30 responses

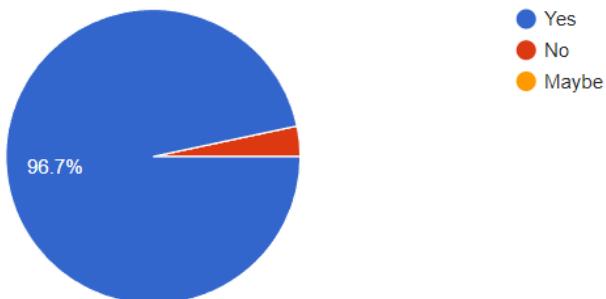


Figure 398: post-Survey response 3.

Customers can buy coupon for their love ones in this application as it provides the feature of coupon sharing too. So respondents were asked if this feature was helpful or not, so from that, 96.7% of them found it.

### IV. Response 4

Did you find Add to cart feature helpful?

[Copy](#)

30 responses

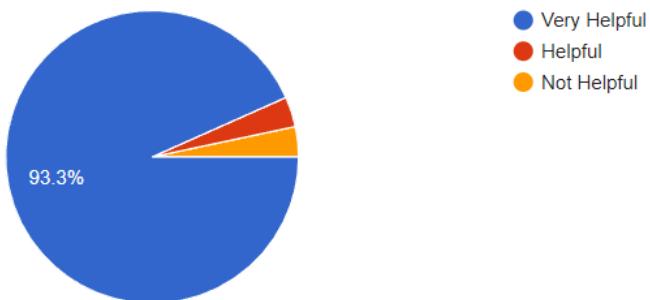


Figure 399: post-Survey response 4.

Customers can add their favourite deals in their cart. Add to cart is very useful feature and it is also implemented in KarKhana application. Here, 93.3% of them found this feature helpful.

## V. Response 5

How easy was it to search deals on KarKhana?

 Copy

30 responses

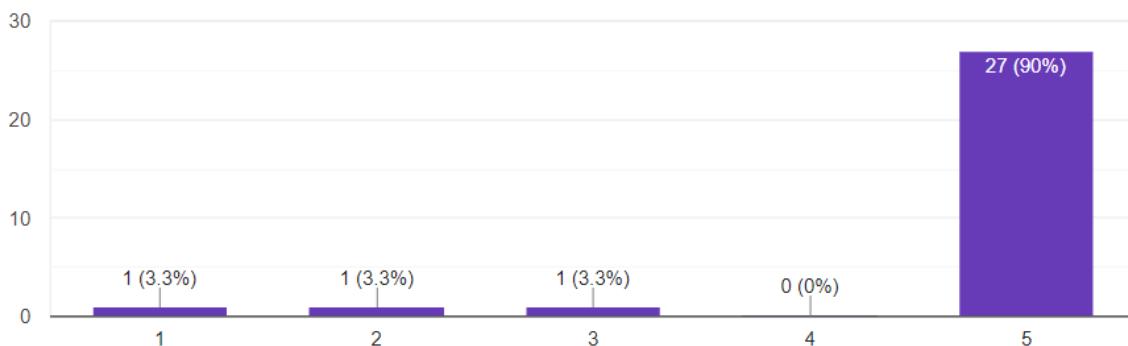


Figure 400: post-Survey response 5.

Search bar is also implemented in this application to make it easy for the customers to find their desired deals quickly. Here, Most of them i.e., 90% of them found easy to search deals, 3.3% of them rated 3 so they had neutral feeling about this and 3.3% of them rated 1 and 2 respectively which means 3.3% of them each found it hard to search for deals in the system.

## VI. Response 6

How easy is to post deals on KarKhana?

 Copy

30 responses

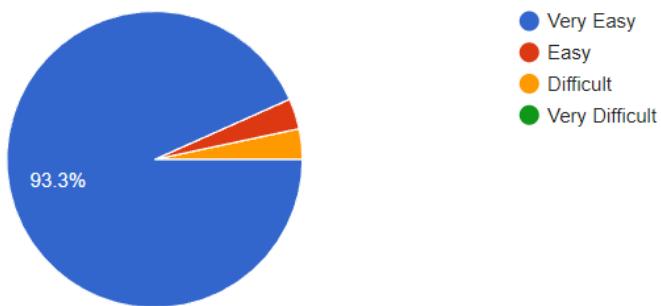


Figure 401: post-Survey response 6.

In this application, vendors can post their deals from their business. So, respondents were asked how hard was it to post deals in the KarKhana system. Here, 93.3% of them found it very easy.

#### VII. Response 7

How satisfied are you with the KarKhana Mobile application?

 Copy

30 responses

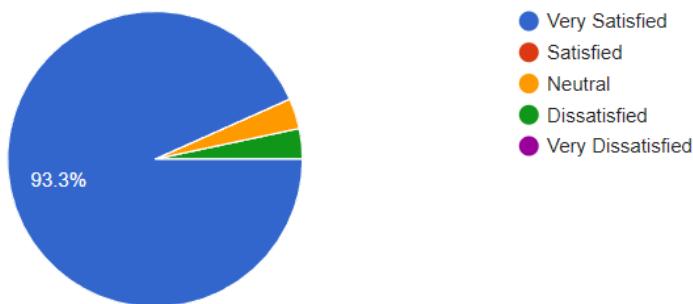


Figure 402: post-Survey response 7.

Respondents were about how they feel about the application. Most of them which is 93.3% of them were very satisfied with the application.

#### VIII. Response 8

Would you Recommend our Deal purchasing platform to others?

 Copy

30 responses

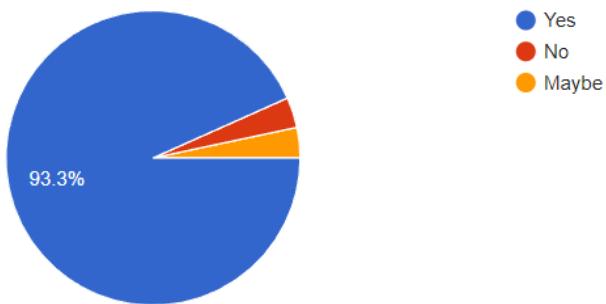


Figure 403: post-Survey response 8.

---

## 8.3 APPENDIX 3: SAMPLE CODE

---

### 8.3.1 Sample Code of the UI

---

#### For Customers

##### User Login UI

```
// ignore_for_file: use_build_context_synchronously,  
prefer_const_constructors_in_immutables, unnecessary_new,  
non_constant_identifier_names, prefer_final_fields, avoid_print,  
prefer_const_constructors, invalid_use_of_visible_for_testing_member,  
invalid_use_of_protected_member
```

```
import 'package:karkhana/API/auth_api.dart';  
  
import 'package:karkhana/Packages/Packages.dart';
```

```
class UserLoginPage extends StatefulWidget {  
  
    UserLoginPage({  
        super.key,  
    });  
  
    @override  
    State<UserLoginPage> createState() => _UserLoginPageState();  
}
```

```
class _UserLoginPageState extends State<UserLoginPage> {  
  
    TextEditingController _UserEmail = new TextEditingController();
```

```
TextEditingController _UserPw = new TextEditingController();

 GlobalKey<FormState> _formKey = new GlobalKey<FormState>();

Future signIn() async {

    try {

        final user = await GoogleSignInApi.login();

        if (user == null) {

            ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
                duration: Duration(milliseconds: 1000),
                content: Text('Sign in Failed')));
        } else {

            Navigator.of(context)
                .push(MaterialPageRoute(builder: (context) => askLocation()));

        }
    } on PlatformException catch (e) {

        print("Platform Exception");

    } catch (e) {

        ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
            duration: Duration(milliseconds: 1000),
            content: Text('Something went wrong')));

        print("Error is \$e");
    }
}
```

```
}
```

```
@override  
void initState() {  
    _UserEmail;  
    _UserPw;  
    super.initState();  
}
```

```
@override  
Widget build(BuildContext context) {  
    return GestureDetector(  
        onTap: () {  
            FocusScopeNode currentFocus = FocusScope.of(context);  
  
            if (!currentFocus.hasPrimaryFocus) {  
                currentFocus.unfocus();  
            }  
        },  
        child: Scaffold(  
            resizeToAvoidBottomInset: false,  
            backgroundColor: Colours.backgroundColor,  
            body: SafeArea(
```

```
child: SingleChildScrollView(  
  child: Form(  
    key: _formKey,  
    child: Column(  
      children: [  
        Padding(  
          padding: EdgeInsets.only(top: 10.h),  
          child: Row(  
            mainAxisAlignment: MainAxisAlignment.start,  
            children: [  
              SizedBox(  
                width: 10.w,  
              ),  
              GestureDetector(  
                child: Icon(Icons.arrow_back_ios),  
                onTap: () {  
                  Navigator.pop(context);  
                },  
              ),  
            ],  
          ),  
        Padding(  
          padding: EdgeInsets.only(left: 10.w),  
          child: FormField(  
            controller: TextEditingController(  
              text: '1234567890'),  
            builder: (FormFieldState state) {  
              return TextFormField(  
                controller: state.controller,  
                decoration: InputDecoration(  
                  prefixIcon: Icon(Icons.lock),  
                  hintText: 'Enter your password',  
                ),  
                obscureText: true,  
              );  
            },  
          ),  
        ),  
      ],  
    ),  
  ),  
);
```

```
padding:  
    EdgeInsets.symmetric(horizontal: 30.w, vertical: 10.h),  
  
child: Column(  
  
    mainAxisAlignment: MainAxisAlignment.start,  
  
    children: [  
  
        Row(  
  
            mainAxisAlignment: MainAxisAlignment.center,  
  
            children: [  
  
                Padding(  
  
                    padding: EdgeInsets.only(right: 10.w),  
  
                    child: Image.asset(  
  
                        "assets/images/logomain.png",  
  
                        height: 90.h,  
  
                        width: 90.w,  
  
  
            ),  
  
            Column(  
  
                children: [  
  
                    LargeText(  
  
                        text: "Log in to",  
  
                    ),  
  
                    Center(  
  
                        child: Row(  

```

```
children: [  
    LargeText(  
        text: "Kar",  
        size: 24,  
    ),  
    LargeText(  
        text: "Khana",  
        color: Colours.secondaryColor,  
        size: 24,  
    )  
    )),  
    ],  
),  
],  
),  
SizedBox(height: 20.h),  
NtextField(  
    controller: _UserEmail,  
    name: 'Email',  
    prefix: const Icon(Icons.person_outline),  
    validator: ValidationOfFields.valEmail,  
),
```

```
SizedBox(  
    height: 25.h,  
,  
    NtextField(  
        controller: _UserPw,  
        name: 'Password',  
        obscure: true,  
        isPassword: true,  
        prefix: const Icon(Icons.lock_outline),  
        validator: ValidationOfFields.valPassword,  
,  
    ),  
    SizedBox(  
        height: 8.h,  
,  
    ),  
    GestureDetector(  
        child: smallText(  
            text: "Forget Password?",  
            decoration: TextDecoration.underline,  
,  
        ),  
        onTap: () {  
            showModalBottomSheet(  
                shape: const RoundedRectangleBorder(  
                    borderRadius: BorderRadius.vertical(  

```

```
        top: Radius.circular(50.0),  
        ),  
        ),  
        context: context,  
        isScrollControlled: true,  
        builder: (context) {  
          return SizedBox(  
            height: 400.h,  
            child: ForgetPassword(),  
          );  
        },  
      );  
    },  
  ),  
  SizedBox(height: 20.h),  
  ButtonContainer(  
    text: 'Login',  
    onClick: () async {  
      final isValidForm =  
        _formKey.currentState!.validate();  
      if (isValidForm) {  
        var authResponse =  
          await userAuth(_UserEmail.text, _UserPw.text);  
      }  
    },  
  ),  
);
```

```
if (authResponse.runtimeType == String) {  
  
    showDialog(  
  
        context: context,  
  
        builder: (BuildContext context) =>  
  
            AlertDialog(  
  
                title: const Text('Invalid Input'),  
  
                content: Text(  
  
                    authResponse,  
  
                ),  
  
                actions: [  
  
                    TextButton(  
  
                        onPressed: () {  
  
                            Navigator.pop(context);  
  
                        },  
  
                        child: const Text('Close'))  
  
                ],  
  
            ));  
  
} else if (authResponse.runtimeType == User) {  
  
    print("Hatti");  
  
    User user = authResponse;  
  
    context.read<UserCubit>().emit(user);  
  
    if (user.is_user == 1) {  
  
        ScaffoldMessenger.of(context)
```

```
.showSnackBar(const SnackBar(  
    backgroundColor: Colors.green,  
    content: Text(  
        "Logged in",  
        textAlign: TextAlign.center,  
    )),  
  
    Navigator.of(context).pushReplacement(  
        MaterialPageRoute(  
            builder: (context) => askLocation())));  
  
} else {  
  
    await logOutUser(user.token!);  
  
    showDialog(  
        context: context,  
        builder: (BuildContext context) =>  
            AlertDialog(  
                title: const Text('Invalid Input'),  
                content: const Text(  
                    'Unable to log in with provided credentials.'),  
                actions: [  
                    TextButton(  
                        onPressed: () async {  
                            Navigator.pop(context);  
                        },  
                ],  
            ),  
    );  
}  
}
```

```
        child: const Text('Close')),  
    ],  
);  
}  
}  
},  
butColor: Colours.secondaryColor,  
butborderColor: Colours.secondaryColor,  
,  
SizedBox(  
height: 60.h,  
child: Padding(  
padding: EdgeInsets.only(top: 15.h),  
child: Row(  
children: [  
const Flexible(  
flex: 1,  
child: Divider(  
color: Colours.smallColor,  
thickness: 0.5,  
height: 0,  
),
```

```
        ),  
        Center(  
            child: Container(  
                padding:  
                    EdgeInsets.symmetric(horizontal: 16.w),  
                child: smallText(  
                    text: 'or',  
                    size: 17.sp,  
                ),  
            ),  
        ),  
        const Flexible(  
            flex: 1,  
            child: Divider(  
                color: Colours.smallColor,  
                thickness: 0.5,  
                height: 0,  
            ),  
        ),  
    ],  
,  
,
```

```
SizedBox(  
    height: 15.h,  
)  
  
googleButton(  
    image: "assets/images/google.svg",  
    text: 'Login with Google',  
    butColor: Colours.textColor,  
    onClick: signIn,  
)  
  
SizedBox(  
    height: 50.h,  
)  
  
Row(  
    mainAxisAlignment: MainAxisAlignment.center,  
    children: [  
        smallText(  
            text: "Don't have an Account? ",  
)  
  
        GestureDetector(  
            onTap: () {  
                Navigator.of(context).push(PageTransition(  
                    type: PageTransitionType.bottomToTop,  
                    child: UserSignUp(),  
                ))  
            },  
        )  
    ]  
)
```

```
    )),
  },
  child: smallText(
    text: "Register",
    color: Colours.borderColor,
  )));
],
),
SizedBox(
  height: 10.h,
),
],
),
],
),
],
),
),
),
),
),
),
),
),
),
),
),
),
),
),
);
}
```

```
@override  
void dispose() {  
    _UserEmail.dispose();  
    _UserPw.dispose();  
  
    super.dispose();  
}  
}
```

**User Registration UI**

```
// ignore_for_file: use_build_context_synchronously, prefer_final_fields,
non_constant_identifier_names, unused_import, prefer_const_constructors

import 'package:karkhana/API/auth_api.dart';

import 'package:karkhana/Packages/Packages.dart';

import 'dart:io';

import 'package:karkhana/usersPages/customerPages/HomePages/splashscreen.dart';

class UserSignUp extends StatefulWidget {

  const UserSignUp({super.key});

  @override
  State<UserSignUp> createState() => _UserSignUpState();
}

class _UserSignUpState extends State<UserSignUp> {

  GlobalKey<FormState> _formKey = GlobalKey<FormState>();

  TextEditingController _UserName = TextEditingController();

  TextEditingController _UserEmail = TextEditingController();

  TextEditingController _UserPw = TextEditingController();

  TextEditingController _UserPwConfirm = TextEditingController();
```

```
@override  
void initState() {  
    // TODO: implement initState  
  
    _UserEmail;  
  
    _UserName;  
  
    _UserPw;  
  
    _UserPwConfirm;  
  
    super.initState();  
}  
  
}
```

```
@override  
Widget build(BuildContext context) {  
    return GestureDetector(  
        onTap: () {  
            FocusScopeNode currentFocus = FocusScope.of(context);  
  
            if (!currentFocus.hasPrimaryFocus) {  
                currentFocus.unfocus();  
            }  
        },  
        child: Scaffold(  
            resizeToAvoidBottomInset: true,
```

```
backgroundColor: Colours.backgroundColor,  
body: SafeArea(  
    child: SingleChildScrollView(  
        child: Form(  
            key: _formKey,  
            child: Column(  
                children: [  
                    Padding(  
                        padding: EdgeInsets.only(top: 10.h),  
                        child: Row(  
                            mainAxisAlignment: MainAxisAlignment.start,  
                            children: [  
                                SizedBox(  
                                    width: 10.w,  
                                ),  
                                GestureDetector(  
                                    child: Icon(Icons.arrow_back_ios),  
                                    onTap: () {  
                                        Navigator.pop(context);  
                                    },  
                                ),  
                            ],  
                        ),  
                    ),  
                ],  
            ),  
        ),  
    ),  
),
```

```
 ),  
 Padding(  
 padding: EdgeInsets.symmetric(horizontal: 30.w),  
 child: Column(  
 children: [  
 Center(  
 child: Column(  
 children: [  
 RichText(  
 text: TextSpan(  
 style: const TextStyle(  
 color:  
 Colors.blue), //apply style to all  
 children: [  
 TextSpan(  
 text: 'Kar',  
 style: GoogleFonts.poppins(  
 color: Colours.textColor,  
 fontSize: 40.sp,  
 fontWeight: FontWeight.w700,  
 decoration: TextDecoration.none,  
 ),  
 ),
```

```
    TextSpan(  
        text: 'Khana',  
        style: GoogleFonts.poppins(  
            color: Colours.secondaryColor,  
            fontSize: 40.sp,  
            fontWeight: FontWeight.w700,  
            decoration: TextDecoration.none,  
        )),  
    ]),  
    smallText(  
        text: "Registration",  
        size: 16,  
        color: Colours.secondaryColor,  
    )  
],  
,  
,  
Center(  
    child: Column(  
        children: [  
            Lottie.asset('assets/lottie/getDiscount.json',  
                height: 180.h, width: 350.w),  
            smallText(  
                text: 'Get  
    
```

```
        text: "Be a member and get your best deals here!",  
        size: 14,  
    ),  
],  
)),  
SizedBox(height: 25.h),  
NtextField(  
    controller: _UserName,  
    name: 'Name',  
    prefix: const Icon(Icons.person_outline),  
    validator: ValidationOfFields.valName),  
SizedBox(  
    height: 25.h,  
,  
NtextField(  
    controller: _UserEmail,  
    name: 'Email',  
    prefix: const Icon(Icons.mail_outline),  
    validator: ValidationOfFields.valEmail,  
,  
SizedBox(  
    height: 25.h,  
,
```

```
NtextField(  
    controller: _UserPw,  
    name: 'Password',  
    obscure: true,  
    prefix: const Icon(Icons.lock_outline),  
    isPassword: true,  
    validator: ValidationOfFields.valPassword,  
,  
SizedBox(  
    height: 25.h,  
,  
NtextField(  
    controller: _UserPwConfirm,  
    name: 'Confirm Password',  
    obscure: true,  
    prefix: const Icon(Icons.lock_outline),  
    isPassword: true,  
    validator: ValidationOfFields.valPassword,  
,  
SizedBox(  
    height: 25.h,  
,  
ButtonContainer(
```

```
butborderColor: Colours.secondaryColor,  
text: 'Signup',  
butColor: Colours.secondaryColor,  
onClick: () async {  
    final isValidForm =  
        _formKey.currentState!.validate();  
    if (isValidForm) {  
        var authResponse = await registerUser(  
            _UserName.text,  
            _UserEmail.text,  
            _UserPw.text,  
            _UserPwConfirm.text,  
        );  
  
        if (authResponse.runtimeType == String) {  
            showDialog(  
                context: context,  
                builder: (BuildContext context) =>  
                    AlertDialog(  
                        title: const Text('Invalid Input'),  
                        content: Text(  
                            authResponse,  
                        ),  
                    ),  
            );  
        } else {  
            Navigator.pushNamed(context, '/home');  
        }  
    }  
},
```

```
actions: [  
    TextButton(  
        onPressed: () {  
            Navigator.pop(context);  
        },  
        child: const Text('Close'))  
    ],  
);  
}  
} else {  
    Navigator.of(context).push(  
        MaterialPageRoute(builder: (context) {  
            return LoadingSplashScreen();  
        }),  
    );  
}  
}  
},  
),  
SizedBox(  
    height: 30.h,  
),  
Row(  
    mainAxisAlignment: MainAxisAlignment.center,
```

```
children: [  
    Text(  
        "Already have an Account? ",  
        style: TextStyle(  
            color: Colours.greyColor,  
            fontSize: 12.sp,  
            fontWeight: FontWeight.w400,  
            fontFamily: 'ProximaNova',  
            decoration: TextDecoration.none),  
    ),  
    GestureDetector(  
        onTap: () {  
            Navigator.of(context).push(  
                MaterialPageRoute(builder: (context) {  
                    return UserLoginPage();  
                }),  
            );  
        },  
        child: Text(  
            "Login",  
            style: TextStyle(  
                color: Colours.borderColor,  
                fontSize: 12.sp,
```

```
        fontWeight: FontWeight.w400,  
        fontFamily: 'ProximaNova',  
        decoration: TextDecoration.none),  
,  
,  
],  
,  
SizedBox(  
    height: 20.h,  
,  
],  
,  
),  
],  
,  
),  
],  
,  
),  
),  
),  
),  
),  
),  
),  
);  
}  
  
@override
```

```
void dispose() {  
    _UserName.dispose();  
    _UserEmail.dispose();  
    _UserPw.dispose();  
    _UserPwConfirm.dispose();  
    super.dispose();  
}  
}
```

**Forgot Password UI**

```
// ignore_for_file: unnecessary_new, non_constant_identifier_names, prefer_final_fields,  
annotate_overrides
```

```
import 'package:karkhana/Packages/Packages.dart';
```

```
class ForgetPassword extends StatefulWidget {
```

```
    const ForgetPassword({super.key});
```

```
    @override
```

```
    State<ForgetPassword> createState() => _ForgetPasswordState();
```

```
}
```

```
class _ForgetPasswordState extends State<ForgetPassword> {
```

```
    TextEditingController _UserEmail = new TextEditingController();
```

```
    GlobalKey<FormState> _formKey = new GlobalKey<FormState>();
```

```
    void dispose() {
```

```
        _UserEmail.dispose();
```

```
        super.dispose();
```

```
}
```

```
@override

Widget build(BuildContext context) {

    return GestureDetector(
        onTap: () {
            FocusScopeNode currentFocus = FocusScope.of(context);

            if (!currentFocus.hasPrimaryFocus) {
                currentFocus.unfocus();
            }
        },
        child: Scaffold(
            resizeToAvoidBottomInset: true,
            backgroundColor: Colours.backgroundColor,
            body: Padding(
                padding: const EdgeInsets.all(19.0),
                child: Form(
                    key: _formKey,
                    child: SingleChildScrollView(
                        child: Column(
                            mainAxisAlignment: MainAxisAlignment.start,
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                                SizedBox(
                                    height: 10,
                                ),
                                Text(
                                    'Enter your details',
                                    style: TextStyle(
                                        color: Colors.black,
                                        fontSize: 16,
                                        fontWeight: FontWeight.w500,
                                    ),
                                ),
                                SizedBox(
                                    height: 10,
                                ),
                                TextFormField(
                                    controller: nameController,
                                    decoration: InputDecoration(
                                        border: OutlineInputBorder(),
                                        labelText: 'Name',
                                    ),
                                ),
                                SizedBox(
                                    height: 10,
                                ),
                                TextFormField(
                                    controller: emailController,
                                    decoration: InputDecoration(
                                        border: OutlineInputBorder(),
                                        labelText: 'Email',
                                    ),
                                ),
                                SizedBox(
                                    height: 10,
                                ),
                                TextFormField(
                                    controller: passwordController,
                                    obscureText: true,
                                    decoration: InputDecoration(
                                        border: OutlineInputBorder(),
                                        labelText: 'Password',
                                    ),
                                ),
                                SizedBox(
                                    height: 10,
                                ),
                                TextFormField(
                                    controller: confirmPasswordController,
                                    obscureText: true,
                                    decoration: InputDecoration(
                                        border: OutlineInputBorder(),
                                        labelText: 'Confirm Password',
                                    ),
                                ),
                                SizedBox(
                                    height: 10,
                                ),
                                Container(
                                    width: double.infinity,
                                    padding: EdgeInsets.all(10),
                                    margin: EdgeInsets.all(10),
                                    decoration: BoxDecoration(
                                        color: Colors.purple,
                                        borderRadius: BorderRadius.circular(10),
                                    ),
                                    child: Text(
                                        'Sign Up',
                                        style: TextStyle(
                                            color: Colors.white,
                                            fontSize: 16,
                                            fontWeight: FontWeight.w500,
                                        ),
                                    ),
                                ),
                                SizedBox(
                                    height: 10,
                                ),
                                Row(
                                    mainAxisAlignment: MainAxisAlignment.end,
                                    children: [
                                        Text(
                                            'Already have an account? ',
                                            style: TextStyle(
                                                color: Colors.black,
                                                fontSize: 14,
                                            ),
                                        ),
                                        Text(
                                            'Log In',
                                            style: TextStyle(
                                                color: Colors.purple,
                                                fontSize: 14,
                                                fontWeight: FontWeight.w500,
                                            ),
                                        ),
                                    ],
                                ),
                            ],
                        ),
                    ),
                ),
            ),
        ),
    );
}
```

```
height: 10.h,  
,  
LargeText(  
text: "Forget Password",  
size: 25,  
,  
SizedBox(  
height: 10.h,  
,  
Center(  
child: Lottie.asset('assets/lottie/forgetpwMail.json',  
height: 100.h, width: 180.w),  
,  
NtextField(  
controller: _UserEmail,  
name: 'Email',  
prefix: const Icon(Icons.person_outline),  
autofocus: true,  
validator: ValidationOfFields.valEmail,  
,  
ListTile(  
title: Align(  
alignment: Alignment.center,
```

```
        child: smallText(  
            text: "A reset link will be sent to the above email!",  
        ),  
        ),  
    ),  
  
    Container(  
        padding: const EdgeInsets.all(8),  
        child: Padding(  
            padding: EdgeInsets.only(  
                bottom: MediaQuery.of(context).viewInsets.bottom),  
            child: ButtonContainer(  
                text: 'Send link',  
                onClick: () {  
                    final isValidForm = _formKey.currentState!.validate();  
                    if (isValidForm) {  
                        Navigator.pop(context);  
                    }  
                },  
                butColor: Colours.secondaryColor,  
                butborderColor: Colours.secondaryColor,  
            ),  
        ),  
    ),
```

```
],  
),  
),  
),  
),  
);  
}  
}
```

**Set User Location UI**

```
// ignore_for_file: camel_case_types, use_build_context_synchronously
```

```
import 'package:karkhana/API/auth_api.dart';  
import 'package:karkhana/Packages/Packages.dart';
```

```
class askLocation extends StatefulWidget {  
  const askLocation({super.key});  
  
  @override  
  State<askLocation> createState() => _askLocationState();  
}
```

```
class _askLocationState extends State<askLocation> {  
  late String lat;  
  late String lon;  
  String currentLocation = "";
```

```
Future<Position> getCurrentLocationUser() async {  
  bool serviceEnabled;  
  LocationPermission permission;  
  serviceEnabled = await Geolocator.isLocationServiceEnabled();  
  if (!serviceEnabled) {
```

```
await Geolocator.openLocationSettings();

return Future.error('Location services are disabled');

}

permission = await Geolocator.checkPermission();

if (permission == LocationPermission.denied) {

    permission = await Geolocator.requestPermission();

    if (permission == LocationPermission.denied) {

        return Future.error('Location permissions are denied');

    }

}

if (permission == LocationPermission.deniedForever) {

    return Future.error(
        'Location permission are permanently denied, we cannot request');

}

return await Geolocator.getCurrentPosition();

}

Future<void> getUserAddress(Position position) async {

List<Placemark> placemarks =

    await placemarkFromCoordinates(position.latitude, position.longitude);
```

```
// ignore: avoid_print
print(placemarks);

// ignore: unused_local_variable
Placemark place = placemarks[0];

currentLocation = '${place.locality}';
${place.subAdministrativeArea},${place.administrativeArea},${place.country},${place.iso
CountryCode}";

setState(() {});

var box = await Hive.openBox(tokenBox);

var keybox = box.get("token");

await addLocation(keybox, currentLocation);

Navigator.of(context).pushReplacement(PageTransition(
    type: PageTransitionType.bottomToTop,
    child: btmNavigationBar(),
));

// Navigator.of(context).pushReplacement(
//     MaterialPageRoute(builder: (context) => LoadingSplashScreen2()));
}

@Override
```

```
Widget build(BuildContext context) {  
  return Scaffold(  
    backgroundColor: Colours.backgroundColor,  
    body: SingleChildScrollView(  
      child: Padding(  
        padding: const EdgeInsets.symmetric(vertical: 200),  
        child: Column(  
          children: [  
            Center(  
              child: Lottie.asset('assets/lottie/location.json',  
                height: 160.h, width: 320.w),  
            ),  
            SizedBox(  
              height: 5.h,  
            ),  
            Padding(  
              padding: const EdgeInsets.only(left: 20),  
              child: Column(  
                crossAxisAlignment: CrossAxisAlignment.start,  
                children: [  
                  LargeText(  
                    text: "Set Your Location",  
                    size: 28,
```

```
        color: Colours.secondaryColor,  
    ),  
  
    smallText(  
  
        text:  
  
            "By sharing your location you can find nearby offers and deals more  
easily",  
  
        overFlow: TextOverflow.visible,  
  
        maxline: 2,  
    ),  
  
    SizedBox(  
  
        height: 20.h,  
    ),  
  
    ButtonContainerIcon(  
  
        butborderColor: Colours.secondaryColor,  
  
        text: "Use my location",  
  
        butColor: Colours.secondaryColor,  
  
        onClick: () async {  
  
            Position position = await getCurrentLocationUser();  
  
            // locationMessage =  
  
            //           'Latitude   of   Place:${position.latitude},   Longitude   of  
Place:${position.longitude}';  
  
            getUserAddress(position);  
  
            setState(() {});  
        },  
    ),  
),
```

```
    },  
    icon: Icons.my_location_outlined,  
),  
],  
(  
),  
],  
),  
));  
}  
}
```

**User Homepage UI**

```
// ignore_for_file: prefer_const_constructors_in_immutables,
non_constant_identifier_names, prefer_final_fields, prefer_const_constructors

import 'dart:async';

import 'package:karkhana/API/auth_api.dart';
import 'package:karkhana/Packages/Packages.dart';

class HomePage extends StatefulWidget {

    HomePage({
        super.key,
    });

    @override
    State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
    final StreamController<String> _locationController =
        StreamController<String>();
    late final String _locationData;
}
```

```
@override  
void initState() {  
    super.initState();  
}
```

```
@override  
void didChangeDependencies() {  
    super.didChangeDependencies();  
  
    final user = context.read<UserCubit>().state;  
  
    _locationData = "${user.location}";  
  
    // simulate fetching location data after 2 seconds  
  
    Future.delayed(const Duration(seconds: 2), () {  
        _locationController.add(_locationData);  
    });  
}
```

```
@override  
void dispose() {  
    _locationController.close();  
    super.dispose();  
}
```

FetchLocationDeals \_DealsInLocation = FetchLocationDeals();

```
@override

Widget build(BuildContext context) {

    User user = context.read<UserCubit>().state;

    int id = user.id!;

    return RefreshIndicator(
        onRefresh: () async {
            GetCart(id);
            setState(() {});
        },
        child: Scaffold(
            appBar: PreferredSize(
                preferredSize: Size.fromHeight(95.h),
                child: SafeArea(
                    child: Container(
                        margin: EdgeInsets.only(top: 15.h, bottom: 15.h),
                        padding: EdgeInsets.only(left: 20.w, right: 20.w),
                        child: Row(
                            mainAxisAlignment: MainAxisAlignment.spaceBetween,
                            children: [
                                Column(
                                    crossAxisAlignment: CrossAxisAlignment.start,
                                    children: [
                                        Text("Product Name"),
                                        Text("Price"),
                                        Text("Quantity"),
                                        Text("Total Price")
                                    ],
                                ),
                                Column(
                                    crossAxisAlignment: CrossAxisAlignment.end,
                                    children: [
                                        Text("Remove"),
                                        Text("Edit")
                                    ],
                                )
                            ],
                        ),
                    ),
                ),
            ),
        ),
    );
}
```

```
LargeText(  
    text: "KarKhana",  
    color: Colours.secondaryColor,  
)  
  
Row(  
    children: [  
        StreamBuilder<String>(  
            stream: _locationController.stream,  
            builder: (context, snapshot) {  
                if (snapshot.connectionState ==  
                    ConnectionState.waiting) {  
                    // show a loading indicator if the data is still being fetched  
                    return SizedBox(  
                        height: 15.h,  
                        width: 15.w,  
                        child: const CircularProgressIndicator(  
                            strokeWidth: 2,  
                            color: Colours.secondaryColor,  
)  
                );  
                } else if (snapshot.hasError) {  
                    // show an error message if there was an error fetching the data  
                    return smallText(  
                        text: "An error occurred while fetching data.",  
                        color: Colours.errorColor,  
                    )  
                } else {  
                    // show the fetched data  
                    return Text(  
                        text: snapshot.data!,  
                        style: TextStyle(  
                            color: Colours.primaryColor,  
                            fontSize: 16.sp,  
                            fontWeight: FontWeight.bold,  
                        ),  
                    )  
                }  
            }  
        )  
    ]  
)
```

```
        text: 'Error loading location',  
        color: Colors.red,  
    );  
  
} else {  
  
    // show the location data if it was fetched successfully  
  
    return smallText(  
  
        text: snapshot.data!,  
        color: Colours.textColor,  
    );  
  
}  
  
},  
,  
const IconData(Icons.arrow_drop_down_rounded),  
],  
,  
],  
,  
),  
Center(  
child: InkWell(  
onTap: () {  
    showSearch(context: context, delegate: SearchUser());  
},  
child: Container(  

```

```
width: 45.w,  
height: 45.h,  
decoration: BoxDecoration(  
    borderRadius: BorderRadius.circular(15),  
    color: Colours.secondaryColor,  
,  
child: const Icon(  
    Icons.search_outlined,  
    color: Colours.backgroundColor,  
,  
,  
,  
)  
[  
,  
,  
,  
,  

```

```
child: Padding(  
    padding: EdgeInsets.symmetric(horizontal: 18.w),  
  
    child: Row(  
        mainAxisAlignment: MainAxisAlignment.start,  
  
        children: [  
  
            GestureDetector(  
                onTap: () {  
  
                    Navigator.of(context).push(PageTransition(  
                        type: PageTransitionType.rightToLeftWithFade,  
  
                        child: const FoodDeals()));  
  
                },  
  
                child: CategoryContainer(  
                    backgroundColor: const Color(0xFFCB1104),  
  
                    categoryName: "Food",  
  
                    categoryImage:  
                        Image.asset('assets/images/pizzas.png'),  
  
                )),  
  
            SizedBox(  
                width: 10.w,  
  
            ),  
  
            GestureDetector(  
                onTap: () {  
  
                    Navigator.of(context).push(PageTransition(  
                        type: PageTransitionType.rightToLeftWithFade,  
                        child: const FoodDeals()));  
  
                },  
  
                child: CategoryContainer(  
                    backgroundColor: const Color(0xFFCB1104),  
  
                    categoryName: "Food",  
  
                    categoryImage:  
                        Image.asset('assets/images/pizzas.png'),  
  
                )),  
  
        ],  
    ),  
);
```

```
        type: PageTransitionType.rightToLeftWithFade,  
        child: CategorySwimming())));  
  
,  
  
child: CategoryContainer(  
  
    backgroundColor: Color(0xFF60CDFF),  
  
    categoryName: "Swimming",  
  
    categoryImage:  
  
        Image.asset('assets/images/swimming.png'),  
  
    )),  
  
SizedBox(  
  
    width: 10.w,  
  
,  
  
GestureDetector(  
  
    onTap: () {  
  
        Navigator.of(context).push(PageTransition(  
  
            type: PageTransitionType.rightToLeftWithFade,  
  
            child: const CategoryCarService())));  
  
    },  
  
child: CategoryContainer(  
  
    backgroundColor: Colours.greyColor,  
  
    categoryName: "Car Service",  
  
    categoryImage:  
  
        Image.asset('assets/images/servicing.png')
```

```
        )),  
  
        SizedBox(  
            width: 10.w,  
        ),  
  
        GestureDetector(  
            onTap: () {  
                Navigator.of(context).push(PageTransition(  
                    type: PageTransitionType.rightToLeftWithFade,  
                    child: const CategoryHealth()));  
            },  
  
            child: CategoryContainer(  
                backgroundColor: Colours.greenColor,  
                categoryName: "Heath",  
                categoryImage: Image.asset(  
                    'assets/images/health.png',  
                    height: 70.h,  
                ),  
            ),  
        )),  
  
        SizedBox(  
            width: 10.w,  
        ),  
    ],  
),
```

```
 ),  
 ),  
 SizedBox(  
   height: 20.h,  
 ),  
 Padding(  
   padding: EdgeInsets.only(left: 10.w, right: 10.w),  
   child: Center(  
     child: Column(  
       children: [  
         Row(  
           children: [  
             SizedBox(  
               width: 10.w,  
             ),  
             Align(  
               alignment: Alignment.centerLeft,  
               child: LargeText(text: "NEAR YOU")),  
           ],  
         ),  
         FutureBuilder(  
           future: _DealsInLocation.getLocationDeals(  
             query: user.location),
```

builder:

```
(BuildContext context, AsyncSnapshot snapshot) {  
  
  if (snapshot.connectionState ==  
      ConnectionState.waiting) {  
  
    return const Center(  
      child: CircularProgressIndicator());  
  
  } else if (snapshot.hasError) {  
  
    return const Center(  
      child: Text("Something went wrong"),  
    );  
  
  } else {  
  
    if (snapshot.hasData) {  
  
      if (snapshot.hasData &&  
          snapshot.data!.isEmpty) {  
  
        return Padding(  
          padding: EdgeInsets.only(bottom: 20.h),  
          child: Column(  
            children: [  
              Lottie.asset(  
                "assets/lottie/noDealsFound.json",  
                height: 180,  
                width: 250,  
              ),  
            ],  
          ),  
        );  
  
    } else {  
      return const Center(  
        child: Text("No deals found"),  
      );  
    }  
  }  
}
```

```
LargeText(  
    text: "No Recently Added Deals!",  
    size: 18,  
,  
],  
,  
);  
}  
else {  
    return Column(  
        children: [  
            SizedBox(  
                height: 15.h,  
,  
            Center(  
                child: SingleChildScrollView(  
                    scrollDirection: Axis.horizontal,  
                    child: Row(  
                        children: snapshot.data  
                            .map<Widget>((e) {  
                                String VendorBusiness_Name =  
                                    "${e.Business_name}";  
                                String VendorBusiness_Location =  
                                    "${e.Business_location}";  
                            }  
                        )  
                    )  
                )  
            )  
        ]  
    );  
}  
}
```

```
// String VendorBusiness_Image =  
//    "${e.Business_photo}";  
  
String Deals_Title =  
    "${e.Deals_title}";  
  
String Deals_Photo =  
    "${e.Deals_photo}";  
  
int Deals_Price = e.Deals_price;  
  
String Deals_Desc =  
    "${e.Deals_desc}";  
  
String Deals_category =  
    "${e.Deals_category}";  
  
String Deals_condition =  
    "${e.Deals_condition}";  
  
int Deals_Discount =  
    e.Deals_discount;  
  
String Deals_applicable_time =  
    "${e.Deals_applicable_time}";  
  
String Deals_applicable_day =  
    "${e.Deals_applicable_day}";  
  
String Deals_expiry_date =  
    "${e.Deals_expiry_date}";  
  
int Deal_user = e.Deals_user;  
  
String Vendor_Phone =
```

```
"${e.Business_Phone}";

return GestureDetector(
    onTap: () {
        Navigator.of(context).push(
            PageTransition(
                type: PageTransitionType.rightToLeftWithFade,
                child: scrollDealDetail(
                    VendorBusinessName:
                        VendorBusiness_Name,
                    VendorBusinessLocation:
                        VendorBusiness_Location,
                    Business_Phone:
                        Vendor_Phone,
                    // VendorBusinessPhoto:
                    //     VendorBusiness_Image,
                    Deals_Title:
                        Deals_Title,
                    Deals_Price:
                        Deals_Price,
                    Deals_Photo:
                        Deals_Photo,
                    Deals_Desc:
                        Deals_Desc,
```

```
Deals_Desc,  
Deals_category:  
Deals_category,  
Deals_condition:  
Deals_condition,  
Deals_Discount:  
Deals_Discount,  
Deals_applicable_time:  
Deals_applicable_time,  
Deals_applicable_day:  
Deals_applicable_day,  
Deals_expiry_date:  
Deals_expiry_date,  
Deal_user:  
Deal_user));  
},  
child: Padding(  
padding: EdgeInsets.symmetric(  
horizontal: 10.w,  
vertical: 5.h),  
child: NearYouDeals(  
VendorBusiness_Name:  
VendorBusiness_Name,
```

```
VendorBusiness_Location:  
    VendorBusiness_Location,  
    Business_Phone: Vendor_Phone,  
    // VendorBusiness_Photo:  
    // VendorBusiness_Image,  
    Deals_Title: Deals_Title,  
    Deals_Price: Deals_Price,  
    Deals_Photo: Deals_Photo,  
    Deals_Desc: Deals_Desc,  
    Deals_category:  
        Deals_category,  
    Deals_condition:  
        Deals_condition,  
    Deals_Discount:  
        Deals_Discount,  
    Deals_applicable_time:  
        Deals_applicable_time,  
    Deals_applicable_day:  
        Deals_applicable_day,  
    Deals_expiry_date:  
        Deals_expiry_date,  
    Deal_user: Deal_user,  
,
```

```
        ),  
        );  
    }).toList(),  
    ),  
    ),  
    SizedBox(  
        width: 50.h,  
    )  
],  
);  
}  
}  
} else {  
    return Padding(  
        padding: EdgeInsets.symmetric(vertical: 10),  
        child: Column(  
            children: [  
                Lottie.asset(  
                    "assets/lottie/search_empty.json",  
                ),  
                LargeText(  
                    text: "No Deals Found!",  
                ),  
                LargeText(text: "Search Other Deals."),  
            ],  
        ),  
    );  
}  
}
```

```
        ],
      ),
    );
  }

},
),
Row(
  children: [
    SizedBox(
      width: 10.w,
    ),
    Align(
      alignment: Alignment.centerLeft,
      child: LargeText(text: "DEALS FOR YOU"),
    ],
  ),
FutureBuilder(
  future: DealsData(),
  builder:
    (BuildContext context, AsyncSnapshot snapshot) {
      if (snapshot.connectionState ==
          ConnectionState.waiting) {
        return const Center(

```

```
        child: CircularProgressIndicator()));

    } else if (snapshot.hasError) {

        return const Center(
            child: Text("Something went wrong"),
        );

    } else {

        if (snapshot.hasData) {

            if (snapshot.hasData &&

                snapshot.data!.isEmpty) {

                return Padding(
                    padding: EdgeInsets.only(bottom: 20.h),
                    child: Column(
                        children: [
                            Lottie.asset(
                                "assets/lottie/noDealsFound.json",
                                height: 180,
                                width: 250,
                            ),
                            LargeText(
                                text: "No Recently Added Deals!",
                                size: 18,
                            ),
                        ],
                    ),
                );
            }
        }
    }
}
```

```
        ),  
    );  
}  
else {  
    return Column(  
        children: [  
            SizedBox(  
                height: 15.h,  
            ),  
            Center(  
                child: Column(  
                    children:  
                        snapshot.data.map<Widget>((e) {  
                            String VendorBusiness_Name =  
                                "${e.Business_name}";  
                            String VendorBusiness_Location =  
                                "${e.Business_location}";  
                            // String VendorBusiness_Image =  
                            //    "${e.Business_photo}";  
                            String Deals_Title =  
                                "${e.Deals_title}";  
                            String Deals_Photo =  
                                "${e.Deals_photo}";  
                            int Deals_Price = e.Deals_price;  
                        },  
                    ),  
                ),  
            ),  
        ],  
    );  
}
```

```
String Deals_Desc = "${e.Deals_desc}";

String Deals_category =
    "${e.Deals_category}";

String Deals_condition =
    "${e.Deals_condition}";

int Deals_Discount = e.Deals_discount;

String Deals_applicable_time =
    "${e.Deals_applicable_time}";

String Deals_applicable_day =
    "${e.Deals_applicable_day}";

String Deals_expiry_date =
    "${e.Deals_expiry_date}";

int Deal_user = e.Deals_user;

String Vendor_Phone =
    "${e.Business_Phone}";

return GestureDetector(
    onTap: () {
        Navigator.of(context).push(
            PageTransition(
                type: PageTransitionType
                    .rightToLeftWithFade,
                child: scrollDealDetail(
                    VendorBusinessName:
```

VendorBusiness\_Name,  
VendorBusinessLocation:  
VendorBusiness\_Location,  
Business\_Phone:  
Vendor\_Phone,  
// VendorBusinessPhoto:  
// VendorBusiness\_Image,  
Deals\_Title:  
Deals\_Title,  
Deals\_Price:  
Deals\_Price,  
Deals\_Photo:  
Deals\_Photo,  
Deals\_Desc:  
Deals\_Desc,  
Deals\_category:  
Deals\_category,  
Deals\_condition:  
Deals\_condition,  
Deals\_Discount:  
Deals\_Discount,  
Deals\_applicable\_time:  
Deals\_applicable\_time,

```
Deals_applicable_day:  
    Deals_applicable_day,  
Deals_expiry_date:  
    Deals_expiry_date,  
Deal_user:  
    Deal_user));  
},  
child: HomeDealContainer(  
    VendorBusiness_Name:  
        VendorBusiness_Name,  
    VendorBusiness_Location:  
        VendorBusiness_Location,  
    Business_Phone: Vendor_Phone,  
    // VendorBusiness_Photo:  
    //    VendorBusiness_Image,  
    Deals_Title: Deals_Title,  
    Deals_Price: Deals_Price,  
    Deals_Photo: Deals_Photo,  
    Deals_Desc: Deals_Desc,  
    Deals_category: Deals_category,  
    Deals_condition: Deals_condition,  
    Deals_Discount: Deals_Discount,  
    Deals_applicable_time:
```

```
        Deals_applicable_time,  
        Deals_applicable_day:  
            Deals_applicable_day,  
        Deals_expiry_date:  
            Deals_expiry_date,  
        Deal_user: Deal_user,  
,  
);  
});  
,  
SizedBox(  
    height: 20.h,  
)  
,  
);  
}  
}  
} else {  
    return Padding(  
        padding: EdgeInsets.symmetric(vertical: 10),  
        child: Column(  
            children: [  
                Lottie.asset(  
                    "assets/lottie/search_empty.json",  

```

```
        ),  
        LargeText(  
            text: "No Deals Found!",  
        ),  
        LargeText(text: "Search Other Deals."),  
    ],  
),  
);  
}  
}  
});  
],  
)),  
)  
],  
),  
),  
));  
}  
}
```

**User Cart UI**

```
// ignore_for_file: non_constant_identifier_names, unused_local_variable
```

```
import 'package:karkhana/API/auth_api.dart';  
import 'package:karkhana/Packages/Packages.dart';
```

```
class AddToCart extends StatefulWidget {
```

```
    const AddToCart({super.key});
```

```
    @override
```

```
    State<AddToCart> createState() => _AddToCartState();
```

```
}
```

```
class _AddToCartState extends State<AddToCart> {
```

```
    @override
```

```
    @override
```

```
    void initState() {
```

```
        super.initState();
```

```
}
```

```
    @override
```

```
    Widget build(BuildContext context) {
```

```
        User user = context.read<UserCubit>().state;
```

```
int id = user.id!;

return RefreshIndicator(
    onRefresh: () async {
        GetCart(id);
        setState(() {});
    },
    child: Scaffold(
        appBar: AppBar(
            centerTitle: true,
            backgroundColor: Colours.backgroundColor,
            elevation: 0,
            automaticallyImplyLeading: false,
            title: LargeText(
                text: "Your Cart",
                size: 25,
                color: Colours.secondaryColor,
            ),
        ),
        body: Padding(
            padding: EdgeInsets.symmetric(horizontal: 10.w),
            child: SingleChildScrollView(
                physics: const AlwaysScrollableScrollPhysics(),

```

```
scrollDirection: Axis.vertical,  
child: FutureBuilder(  
  future: GetCart(id),  
  builder: (BuildContext context, AsyncSnapshot snapshot) {  
    if (snapshot.connectionState == ConnectionState.waiting) {  
      return const Center(child: CircularProgressIndicator());  
    } else if (snapshot.hasError) {  
      return const Center(  
        child: Text("Something went wrong"),  
      );  
    } else {  
      if (snapshot.hasData) {  
        if (snapshot.hasData && snapshot.data.isEmpty) {  
          return Column(  
            mainAxisSize: MainAxisSize.center,  
            children: [  
              SizedBox(  
                height: 50.h,  
              ),  
              Center(  
                child: Lottie.asset(  
                  'assets/lottie/noDealsFound.json',  
                  height: 200.h,  
                ),  
              ),  
            ],  
          );  
        } else {  
          return Column(  
            mainAxisSize: MainAxisSize.center,  
            children: [  
              Center(  
                child: Lottie.asset(  
                  'assets/lottie/noDealsFound.json',  
                  height: 200.h,  
                ),  
              ),  
            ],  
          );  
        }  
      } else {  
        return const Center(child: CircularProgressIndicator());  
      }  
    }  
  },  
);
```

```
width: 200.w,  
)  
)  
LargeText(  
text: "No Deals In Cart",  
size: 18,  
)  
],  
);  
} else {  
return Column(  
children: [  
Center(  
child: Column(  
children: snapshot.data.map<Widget>((e) {  
int Deals_Id = e.Deals_Id;  
String Deals_title = "${e.Deals_title}";  
int Deals_price = e.Deals_price;  
String Deals_photo = "${e.Deals_photo}";  
String Deals_category = "${e.Deals_category}";  
String Deals_desc = "${e.Deals_desc}";  
String Deals_condition = "${e.Deals_condition}";  
String Business Phone = "${e.Vendor Phone}";
```

```
int Deals_discount =
    int.parse("${e.Deals_discount}");

String Deals_applicable_time =
    "${e.Deals_applicable_time}";

String Deals_applicable_day =
    "${e.Deals_applicable_day}";

String Deals_expiry_date =
    "${e.Deals_expiry_date}";

String Business_name = "${e.Business_name}";

String Business_location =
    "${e.Business_location}";

// String Business_photo = "${e.Business_photo}";

int Customer_User = e.Customer_User;

int Deals_user = e.Vendor_User;

return GestureDetector(
    onTap: () {
        Navigator.of(context).push(PageTransition(
            type: PageTransitionType.bottomToTop,
            child: CartDealsDetails(
                VendorBusinessName: Business_name,
                // Business_photo: Business_photo,
                Business_location: Business_location,

```

```
Business_Phone: Business_Phone,  
Deals_Title: Deals_title,  
Deals_Price: Deals_price,  
Deals_Photo: Deals_photo,  
Deals_Desc: Deals_desc,  
Deals_category: Deals_category,  
Deals_condition: Deals_condition,  
Deals_Discount: Deals_discount,  
Deals_applicable_time:  
    Deals_applicable_time,  
Deals_applicable_day:  
    Deals_applicable_day,  
Deals_expiry_date: Deals_expiry_date,  
Customer_User: Customer_User,  
Vendor_User: Deals_user,  
));  
},  
child: smallContainer(  
    dealTitle: Deals_title,  
    businessName: Business_name,  
    discount: Deals_discount,  
    frontLabel: "Expire",  
    newLabel: Deals_expiry_date,
```

```
        smallboxWidght: 0.38,  
        iscart: true,  
    )));
}),
],
);
}  
} else {  
    return Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
            SizedBox(  
                height: 50.h,  
            ),  
            Center(  
                child: Lottie.asset(  
                    'assets/lottie/noDealsFound.json',  
                    height: 200.h,  
                    width: 200.w,  
                ),  
            ),  
            LargeText(  
        ],
    );
}
```

```
    text: "No Deals In Cart",
    size: 18,
),
],
);
}

},
),
),
),
),
),
),
);
}

@Override
void dispose() {
super.dispose();
}
}
```

**User Purchase UI**

```
// ignore_for_file: camel_case_types

import 'package:karkhana/Packages/Packages.dart';

class purchases extends StatefulWidget {

    const purchases({super.key});

    @override
    State<purchases> createState() => _purchasesState();
}

class _purchasesState extends State<purchases> {

    @override
    Widget build(BuildContext context) {
        User user = context.read<UserCubit>().state;
        int id = user.id!;
        return Scaffold(
            floatingActionButton: FloatingActionButton(
                onPressed: () {
                    showSearch(context: context, delegate: SearchCoupon());
                },
                backgroundColor: Colours.secondaryColor,
```

```
        child: const Icon(Icons.search_outlined)),  
  
    body: DefaultTabController(  
  
        length: 2, // length of tabs  
  
        initialIndex: 0,  
  
        child: Padding(  
  
            padding: EdgeInsets.only(top: 20.h),  
  
            child: Column(children: <Widget>[  
  
                Padding(  
  
                    padding: const EdgeInsets.all(8.0),  
  
                    child: Center(  
  
                        child: LargeText(  
  
                            text: "Your Coupon Purchases",  
  
                            color: Colours.secondaryColor,  
  
                            size: 25,  
  
                        ),  
  
                    ),  
  
                ),  
  
            ),  
  
            TabBar(  
  
                indicatorColor: Colours.secondaryColor,  
  
                indicatorWeight: 3,  
  
                labelStyle: GoogleFonts.lato(  
  
                    color: Colours.textColor,  
  
                    fontSize: 15.sp,
```

```
height: 2,  
fontWeight: FontWeight.w600,  
,  
labelColor: Colours.secondaryColor,  
unselectedLabelColor: Colours.textColor2,  
tabs: const [  
    Tab(text: 'Active Coupons'),  
    Tab(text: 'Previous Coupons'),  
,  
,  
Expanded(  
    child: TabBarView(children: <Widget>[  
        const activeCoupon(),  
        PreviousCoupon(  
            id: id,  
        )  
    ]),  
,  
]),  
)),  
);  
}  
}
```

**Profile UI**

```
// ignore_for_file: file_names, prefer_const_constructors_in_immutables,  
use_build_context_synchronously, prefer_const_constructors
```

```
import 'package:karkhana/API/auth_api.dart';  
  
import 'package:karkhana/Packages/Packages.dart';
```

```
class UserSettings extends StatefulWidget {
```

```
    UserSettings({
```

```
        super.key,
```

```
    });
```

```
    @override
```

```
    State<UserSettings> createState() => UserSettingsState();
```

```
}
```

```
class UserSettingsState extends State<UserSettings> {
```

```
    //GoogleSignInAccount user = GoogleSignInAccount;
```

```
    @override
```

```
    void initState() {
```

```
        super.initState();
```

```
    }
```

```
@override  
void dispose() {  
    super.dispose();  
}  
  
@override  
Widget build(BuildContext context) {  
    User user = context.read<UserCubit>().state;  
    return WillPopScope(  
        onWillPop: () async {  
            return false;  
        },  
        child: Scaffold(  
            body: SafeArea(  
                child: SingleChildScrollView(  
                    child: Column(  
                        mainAxisAlignment: MainAxisAlignment.spaceEvenly,  
                        children: [  
                            SizedBox(  
                                height: 35.h,  
                            ),  
                            Row(  
                                mainAxisAlignment: MainAxisAlignment.spaceEvenly,
```

```
mainAxisAlignment: MainAxisAlignment.center,  
children: [  
    Center(  
        child: Material(  
            elevation: 8,  
            shape: const CircleBorder(),  
            child: Container(  
                width: 130,  
                height: 130,  
                decoration: const BoxDecoration(  
                    shape: BoxShape.circle,  
                ),  
                child: Padding(  
                    padding: const EdgeInsets.all(15.0),  
                    child: ClipOval(  
                        child: Image.asset(  
                            'assets/images/user.png',  
                            fit: BoxFit.cover,  
                        ),  
                    ),  
                ),  
            ),  
        ),  
    ),
```

```
        ),  
        ],  
        ),  
        SizedBox(  
            height: 20.h,  
        ),  
        Text(  
            "${user.name}",  
            style: GoogleFonts.poppins(  
                color: Colours.textColor2,  
                fontSize: 20.sp,  
                fontWeight: FontWeight.w700,  
            ),  
            ),  
        SizedBox(  
            height: 25.h,  
        ),  
        TextButtons(  
            label: "Edit Profile",  
            icon: const Icon(  
                Icons.person_outline,  
                color: Colours.secondaryColor,  
            ),
```

```
onClick: () {  
    Navigator.of(context).push(PageTransition(  
        type: PageTransitionType.rightToLeft,  
        child: EditProfile(),  
    )),  
},  
  
SizedBox(  
    height: 5.h,  
,  
TextButtons(  
    label: "About Us",  
    icon: const Icon(  
        Icons.info_outline,  
    color: Colours.secondaryColor,  
,  
    onClick: () {  
        Navigator.of(context).push(PageTransition(  
            type: PageTransitionType.rightToLeft,  
            child: AboutUs(),  
        )),  
    },  
    SizedBox(  
        height: 5.h,
```

```
        ),  
  
        TextButtons(  
  
            label: "Send Feedback",  
  
            icon: const Icon(  
  
                Icons.mail_outline_outlined,  
  
                color: Colours.secondaryColor,  
  
            ),  
  
            onClick: () {  
  
                Navigator.of(context).push(PageTransition(  
  
                    type: PageTransitionType.rightToLeft,  
  
                    child: SendFeedback(),  
  
                ));  
  
            }),  
  
        SizedBox(  
  
            height: 5.h,  
  
        ),  
  
        TextButtons(  
  
            label: "Contact Us",  
  
            icon: const Icon(  
  
                Icons.phone_outlined,  
  
                color: Colours.secondaryColor,  
  
            ),  
  
            onClick: () {
```

```
Navigator.of(context).push(PageTransition(  
    type: PageTransitionType.rightToLeft,  
    child: ContactUs(),  
( ));  
)  
SizedBox(  
    height: 5.h,  
)  
TextButtons(  
    label: "Log out",  
    icon: const Icon(  
        Icons.logout_outlined,  
        color: Colours.secondaryColor,  
)  
    onClick: () {  
        showDialog(  
            context: context,  
            builder: (BuildContext context) => AlertDialog(  
                title: const Text('Log out'),  
                content: const Text(  
                    'Are you sure you want to Log out?'),  
                actions: [  
                    TextButton(  
                       
```

```
onPressed: () async {

    User user =
        context.read<UserCubit>().state;
    await logOutUser(user.token!);

    ScaffoldMessenger.of(context)
        .showSnackBar(const SnackBar(
            duration:
                Duration(milliseconds: 1000),
            backgroundColor:
                Colours.errorColor,
            content: Text(
                "Logged Out",
                textAlign: TextAlign.center,
            )));
}

Navigator.of(context).pushAndRemoveUntil(
    MaterialPageRoute(
        builder: (context) =>
            WelcomeScreen(),
    ),
    (route) => false);
},

child: const Text('Yes')),
```

```
TextButton(  
    onPressed: () {  
        Navigator.pop(context);  
    },  
    child: const Text('NO'))  
],  
());  
},  
),  
)),  
);  
}  
}
```

**Deal UI**

```
// ignore_for_file: file_names, must_be_immutable, non_constant_identifier_names,  
use_key_in_widget_constructors, avoid_types_as_parameter_names,  
unnecessary_string_interpolations, prefer_const_constructors,  
prefer_adjacent_string_concatenation
```

```
import 'package:karkhana/Packages/Packages.dart';
```

```
class HomeDealContainer extends StatefulWidget {
```

```
    final String? VendorBusiness_Name;
```

```
    final String? VendorBusiness_Location;
```

```
    // final String? VendorBusiness_Photo;
```

```
    String? Business_Phone;
```

```
    String Deals_Title;
```

```
    int Deals_Price;
```

```
    String Deals_Photo;
```

```
    String Deals_Desc;
```

```
    String Deals_category;
```

```
    String Deals_condition;
```

```
    int Deals_Discount;
```

```
    String Deals_applicable_time;
```

```
    String Deals_applicable_day;
```

```
    String Deals_expiry_date;
```

```
    int Deal_user;
```

```
HomeDealContainer({  
    Key,  
    key,  
    this.VendorBusiness_Name,  
    this.VendorBusiness_Location,  
    // this.VendorBusiness_Photo,  
    this.Business_Phone,  
    required this.Deals_Price,  
    required this.Deals_Title,  
    required this.Deals_Photo,  
    required this.Deals_Desc,  
    required this.Deals_category,  
    required this.Deals_condition,  
    required this.Deals_Discount,  
    required this.Deals_applicable_time,  
    required this.Deals_applicable_day,  
    required this.Deals_expiry_date,  
    required this.Deal_user,  
});  
  
@override  
State<HomeDealContainer> createState() => _HomeDealContainerState();
```

```
}
```

```
class _HomeDealContainerState extends State<HomeDealContainer> {
```

```
    @override
```

```
    Widget build(BuildContext context) {
```

```
        String photo = widget.Deals_Photo;
```

```
        return Column(
```

```
            children: [
```

```
                Container(
```

```
                    height: (MediaQuery.of(context).size.height * 0.54).h,
```

```
                    width: (MediaQuery.of(context).size.width * 0.9).w,
```

```
                    decoration: BoxDecoration(
```

```
                        color: Colours.backgroundColor,
```

```
                        borderRadius: BorderRadius.circular(20),
```

```
                        boxShadow: [
```

```
                            BoxShadow(
```

```
                                color: Colors.grey.withOpacity(0.5),
```

```
                                spreadRadius: 1.8,
```

```
                                blurRadius: 2,
```

```
                                // changes position of shadow
```

```
                            ),
```

```
                ],
```

```
            ),
```

```
child: Padding(  
    padding: const EdgeInsets.all(18.0),  
    child:  
        Column(crossAxisAlignment: CrossAxisAlignment.start, children: [  
            Row(  
                children: [  
                    const CircleAvatar(  
                        radius: 30,  
                        backgroundImage:  
                            AssetImage("assets/images/businessprofile.jpg"),  
                    ),  
                    Padding(  
                        padding: const EdgeInsets.all(8.0),  
                        child: Column(  
                            crossAxisAlignment: CrossAxisAlignment.start,  
                            children: [  
                                GestureDetector(  
                                    onTap: () {},  
                                    child: LargeText(  
                                        text: "${widget.VendorBusiness_Name}"),  
                                        smallText(text: "${widget.VendorBusiness_Location}")  
                            ],  
                        ),  
                    ),  
                ],  
            ),  
        ],  
    ),  
);
```

```
)  
],  
,  
SizedBox(  
height: 5.h,  
,  
Container(  
height: (MediaQuery.of(context).size.height * 0.2).h,  
width: (MediaQuery.of(context).size.width * 0.9).w,  
decoration: BoxDecoration(  
image: DecorationImage(  
fit: BoxFit.fill,  
image: NetworkImage(photo),  
,  
borderRadius: BorderRadius.circular(7),  
,  
,  
Padding(  
padding: EdgeInsets.symmetric(vertical: 5.h),  
child: LargeText(  
text: "${widget.Deals_Title}",  
maxline: 2,  
size: 18,
```

```
),
),
Container(
height: (MediaQuery.of(context).size.height * 0.04).h,
width: (MediaQuery.of(context).size.width * 0.2).w,
color: Colours.greenColor,
child: Center(
child: LargeText(
text: "${widget.Deals_Discount}" + "%OFF",
color: Color.fromARGB(255, 43, 140, 93),
size: 16,
),
),
),
Padding(
padding: EdgeInsets.only(
top: 8.h,
left: 8.w,
right: 8.w,
),
),
child: ButtonContainer(
butborderColor: Colours.greyColor,
text: "View Details".
```

```
textColor: Colours.textColor,  
butColor: Colours.backgroundColor,  
onClick: () {  
    Navigator.push(  
        context,  
        MaterialPageRoute(  
            builder: (context) => scrollDealDetail(  
                VendorBusinessName:  
                    "${widget.VendorBusiness_Name}",  
                VendorBusinessLocation:  
                    "${widget.VendorBusiness_Location}",  
                Business_Phone: "${widget.Business_Phone}",  
                // VendorBusinessPhoto:  
                //    "${widget.VendorBusiness_Photo}",  
                Deals_Title: widget.Deals_Title,  
                Deals_Price: widget.Deals_Price,  
                Deals_Photo: widget.Deals_Photo,  
                Deals_Desc: widget.Deals_Desc,  
                Deals_category: widget.Deals_category,  
                Deals_condition: widget.Deals_condition,  
                Deals_Discount: widget.Deals_Discount,  
                Deals_applicable_time:  
                    widget.Deals_applicable_time,
```

```
        Deals_applicable_day:  
            widget.Deals_applicable_day,  
            Deals_expiry_date: widget.Deals_expiry_date,  
            Deal_user: widget.Deal_user));  
        },  
    ),  
    ],  
),  
SizedBox(  
    height: 20.h,  
),  
],  
);  
}  
}
```

**Active Coupon UI**

```
// ignore_for_file: file_names, camel_case_types, non_constant_identifier_names,  
unused_local_variable
```

```
import 'package:karkhana/API/auth_api.dart';  
  
import 'package:karkhana/Packages/Packages.dart';
```

```
class activeCoupon extends StatefulWidget {  
  
  const activeCoupon({super.key});  
  
  @override  
  State<activeCoupon> createState() => _activeCouponState();  
}
```

```
class _activeCouponState extends State<activeCoupon> {  
  
  @override  
  Widget build(BuildContext context) {  
  
    User user = context.read<UserCubit>().state;  
  
    int id = user.id!;  
  
    return SingleChildScrollView(  
      child: Padding(  
        padding: EdgeInsets.symmetric(vertical: 20.h),  
        child: FutureBuilder(<
```

```
future: GetCouponDetails(id),  
  
builder: (BuildContext context, AsyncSnapshot snapshot) {  
  
  if (snapshot.connectionState == ConnectionState.waiting) {  
  
    return const Center(child: CircularProgressIndicator());  
  
  } else if (snapshot.hasError) {  
  
    return const Center(  
  
      child: Text("Something went wrong"),  
  
    );  
  
  } else {  
  
    if (snapshot.hasData) {  
  
      if (snapshot.hasData && snapshot.data.isEmpty) {  
  
        return Column(  
  
          mainAxisAlignment: MainAxisAlignment.center,  
  
          children: [  
  
            SizedBox(  
  
              height: 50.h,  
  
            ),  
  
            Center(  
  
              child: Lottie.asset(  
  
                'assets/lottie/noDealsFound.json',  
  
                height: 200.h,  
  
                width: 200.w,  
  
              ),  

```

```
),
LargeText(
    text: "No Deals In Cart",
    size: 18,
    color: Colours.textColor2,
),
],
);
}

} else {
return Column(
    children: [
        Center(
            child: Column(
                children: snapshot.data.map<Widget>((e) {
                    int couponId = int.parse("${e.couponID}");
                    String Deals_title = "${e.deal_title}";
                    int Deals_discount = int.parse("${e.deal_discount}");
                    String Deals_applicable_time =
                        "${e.deal_applicable_time}";
                    String Deals_applicable_day =
                        "${e.deal_applicable_day}";
                    String Deals_expiry_date = "${e.deal_expiry_date}";
                    int Customer_User = e.Customer_user;
                })
            )
        )
    ]
);
```

```
int Deals_user = e.Vendor_user;

String Business_name = "${e.VendorBusiness_Name}";

String Business_location =
    "${e.VendorBusiness_Location}";

String Business_Phone = "${e.Vendor_Phone}";

int Total_coupon_price =
    int.parse("${e.Total_Coupon_Price}");

String CustomerName = "${e.Customer_Name}";

String CustomerEmail = "${e.Customer_Email}";

String CustomerLocation = "${e.Customer_Address}";

String CustomerPhone = "${e.Customer_Phonenum}";

String coupon_Status = "${e.coupon_Status}";

String Payment_Complete = "${e.Payment_Complete}";

// String Deals_photo = "${e.Deals_photo}";

if (coupon_Status == "Active") {

    return GestureDetector(
        onTap: () {
            Navigator.of(context).push(PageTransition(
                type: PageTransitionType.bottomToTop,
                child: activeCouponDeal(
                    couponID: couponId,
                    DealTitle: Deals_title,

```

```
    DealTime: Deals_applicable_time,  
    DealDay: Deals_applicable_day,  
    DealExpire: Deals_expiry_date,  
    DealDiscount: Deals_discount,  
    CouponPrice: Total_coupon_price,  
    CustomerName: CustomerName,  
    CustomerEmail: CustomerEmail,  
    CustomerPhone: CustomerPhone,  
    CustomerAddress: CustomerLocation,  
    BusinessName: Business_name,  
    BusinessLocation: Business_location,  
    BusinessContact: Business_Phone,  
    isActiveisCustomer: true,  
));  
,  
child: smallContainer(  
    dealTitle: Deals_title,  
    businessName: Business_name,  
    discount: Deals_discount,  
    frontLabel: "Status",  
    newLabel: coupon_Status,  
    smallboxColor: Colours.greenColor,  
    smallboxWidght: 0.28,
```

```
        couponID: couponId,  
    )),  
}  
    } else {  
        return Container();  
    }  
}).toList()),  
,  
],  
);  
}  
}  
} else {  
    return Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
            SizedBox(  
                height: 50.h,  
            ),  
            Center(  
                child: Lottie.asset(  
                    'assets/lottie/noDealsFound.json',  
                    height: 200.h,  
                    width: 200.w,  
                ),  
        ],  
    );  
}
```

```
        ),  
        LargeText(  
            text: "No Deals In Cart",  
            size: 18,  
            color: Colours.textColor2,  
        ),  
        ],  
    );  
}  
}  
},  
),  
);  
}  
}
```

**Previous Coupon UI**

```
// ignore_for_file: file_names, must_be_immutable, non_constant_identifier_names,  
unused_local_variable
```

```
import 'package:karkhana/API/auth_api.dart';  
  
import 'package:karkhana/Packages/Packages.dart';
```

```
class PreviousCoupon extends StatefulWidget {
```

```
    int id;  
  
    bool isVendor;  
  
    PreviousCoupon({  
        super.key,  
        required this.id,  
        this.isVendor = false,  
    });
```

```
    @override  
    State<PreviousCoupon> createState() => _PreviousCouponState();  
}
```

```
class _PreviousCouponState extends State<PreviousCoupon> {
```

```
    @override  
    Widget build(BuildContext context) {
```

```
return SingleChildScrollView(  
    child: Padding(  
        padding: EdgeInsets.symmetric(vertical: 20.h),  
        child: FutureBuilder(  
            future: widget.isVendor  
                ? GetVendorCouponDetails(widget.id)  
                : GetCouponDetails(widget.id),  
            builder: (BuildContext context, AsyncSnapshot snapshot) {  
                if (snapshot.connectionState == ConnectionState.waiting) {  
                    return const Center(child: CircularProgressIndicator());  
                } else if (snapshot.hasError) {  
                    return const Center(  
                        child: Text("Something went wrong"),  
                    );  
                } else {  
                    if (snapshot.hasData) {  
                        if (snapshot.hasData && snapshot.data.isEmpty) {  
                            return Column(  
                                mainAxisAlignment: MainAxisAlignment.center,  
                                children: [  
                                    SizedBox(  
                                        height: 50.h,  
                                    ),  
                                ],  
                            );  
                        } else {  
                            return Column(  
                                mainAxisAlignment: MainAxisAlignment.start,  
                                children: [  
                                    Text("Coupon Details"),  
                                    Text("Coupon ID: ${snapshot.data['id']}"),  
                                    Text("Coupon Name: ${snapshot.data['name']}"),  
                                    Text("Coupon Description: ${snapshot.data['description']}"),  
                                    Text("Coupon Type: ${snapshot.data['type']}"),  
                                    Text("Coupon Status: ${snapshot.data['status']}"),  
                                    Text("Coupon Expiry Date: ${snapshot.data['expiry']}"),  
                                    Text("Coupon Usage Count: ${snapshot.data['usage']} / ${snapshot.data['max_usage']}"),  
                                    Text("Coupon Points: ${snapshot.data['points']}"),  
                                    Text("Coupon Terms and Conditions: ${snapshot.data['terms_and_conditions']}"),  
                                ],  
                            );  
                        }  
                    }  
                }  
            },  
        ),  
    ),  
);
```

```
        Center(  
            child: Lottie.asset(  
                'assets/lottie/noDealsFound.json',  
                height: 200.h,  
                width: 200.w,  
            ),  
            ),  
        LargeText(  
            text: "No Deals In Cart",  
            size: 18,  
        ),  
    ],  
);  
}  
} else {  
    return Column(  
        children: [  
            Center(  
                child: Column(  
                    children: snapshot.data.map<Widget>((e) {  
                        int couponId = int.parse("${e.couponID}");  
                        String Deals_title = "${e.deal_title}";  
                        int Deals_discount = int.parse("${e.deal_discount}");  
                        String Deals_applicable_time =  
                    },  
                ),  
            ),  
        ],  
    );  
}  
}
```

```
"${e.deal_applicable_time}";

String Deals_applicable_day =
"${e.deal_applicable_day}";

String Deals_expiry_date = "${e.deal_expiry_date}";

int Customer_User = e.Customer_user;

int Deals_user = e.Vendor_user;

String Business_name = "${e.VendorBusiness_Name}";

String Business_location =
"${e.VendorBusiness_Location}";

String Business_Phone = "${e.Vendor_Phone}";

int Total_coupon_price =
int.parse("${e.Total_Coupon_Price}");

String CustomerName = "${e.Customer_Name}";

String CustomerEmail = "${e.Customer_Email}";

String CustomerLocation = "${e.Customer_Address}";

String CustomerPhone = "${e.Customer_Phonenumbe}r";

String coupon_Status = "${e.coupon_Status}";

String Payment_Complete = "${e.Payment_Complete}";

// String Deals_photo = "${e.Deals_photo}";

if (coupon_Status == "Used") {

    return GestureDetector(
```

```
onTap: () {
    Navigator.of(context).push(PageTransition(
        type: PageTransitionType.bottomToTop,
        child: activeCouponDeal(
            couponID: couponId,
            DealTitle: Deals_title,
            DealTime: Deals_applicable_time,
            DealDay: Deals_applicable_day,
            DealExpire: Deals_expiry_date,
            DealDiscount: Deals_discount,
            CouponPrice: Total_coupon_price,
            CustomerName: CustomerName,
            CustomerEmail: CustomerEmail,
            CustomerPhone: CustomerPhone,
            CustomerAddress: CustomerLocation,
            BusinessName: Business_name,
            BusinessLocation: Business_location,
            BusinessContact: Business_Phone)));
},
child: smallContainer(
    dealTitle: Deals_title,
    businessName: Business_name,
    discount: Deals_discount,
```

```
        frontLabel: "Status",
        newLabel: coupon_Status,
        discountColor: Colours.errorColor,
        smallboxWidght: 0.26,
        couponID: couponId,
    ));
} else {
    return Container();
}
}).toList(),
),
],
);
}
} else {
return Column(
mainAxisAlignment: MainAxisAlignment.center,
children: [
SizedBox(
height: 50.h,
),
Center(
child: Lottie.asset(
```

```
'assets/lottie/noDealsFound.json',  
height: 200.h,  
width: 200.w,  
,  
,  
LargeText(  
text: "No Deals In Cart",  
size: 18,  
,  
],  
);  
}  
}  
},  
,  
),  
);  
}  
}
```

**For Vendor****Vendor Registration UI**

```
// ignore_for_file: file_names, prefer_final_fields, non_constant_identifier_names,  
prefer_const_constructors, unused_local_variable
```

```
import 'package:karkhana/API/auth_api.dart';  
  
import 'package:karkhana/Packages/Packages.dart';
```

```
class VendorSignUp extends StatefulWidget {  
  
  const VendorSignUp({super.key});  
  
  @override  
  State<VendorSignUp> createState() => _VendorSignUpState();  
}
```

```
class _VendorSignUpState extends State<VendorSignUp> {  
  
  GlobalKey<FormState> _formKey = GlobalKey<FormState>();  
  
  TextEditingController _VendorName = TextEditingController();  
  
  TextEditingController _VendorEmail = TextEditingController();  
  
  TextEditingController _VendorPhone = TextEditingController();  
  
  TextEditingController _VendorLocation = TextEditingController();  
  
  TextEditingController _VendorAbout = TextEditingController();  
  
  TextEditingController _VendorOpeningDays = TextEditingController();
```

```
TextEditingController _VendorOpeningTime = TextEditingController();
```

```
TextEditingController _UserPw = TextEditingController();
```

```
TextEditingController _UserPwConfirm = TextEditingController();
```

```
@override
```

```
void initState() {
```

```
    // TODO: implement initState
```

```
    _VendorName;
```

```
    _VendorEmail;
```

```
    _VendorPhone;
```

```
    _VendorLocation;
```

```
    _VendorAbout;
```

```
    _VendorOpeningDays;
```

```
    _VendorOpeningTime;
```

```
    _UserPw;
```

```
    _UserPwConfirm;
```

```
    super.initState();
```

```
}
```

```
@override
```

```
Widget build(BuildContext context) {
```

```
    return GestureDetector(
```

```
        onTap: () {
```

```
FocusScopeNode currentFocus = FocusScope.of(context);

if (!currentFocus.hasPrimaryFocus) {
    currentFocus.unfocus();
}

},
child: Scaffold(
    resizeToAvoidBottomInset: true,
    backgroundColor: Colours.backgroundColor,
    body: SafeArea(
        child: SingleChildScrollView(
            child: Form(
                key: _formKey,
                child: Column(
                    children: [
                        Padding(
                            padding: EdgeInsets.only(top: 10.h),
                            child: Row(
                                mainAxisAlignment: MainAxisAlignment.start,
                                children: [
                                    SizedBox(
                                        width: 10.w,
                                    ),
                                ],
                            ),
                        ),
                    ],
                ),
            ),
        ),
    ),
);
```

```
GestureDetector(  
    child: Icon(Icons.arrow_back_ios),  
    onTap: () {  
        Navigator.pop(context);  
    },  
,  
],  
,  
),  
  
Padding(  
    padding: EdgeInsets.symmetric(horizontal: 30.w),  
    child: Column(  
        children: [  
            Center(  
                child: Column(  
                    children: [  
                        RichText(  
                            text: TextSpan(  
                                style: const TextStyle(  
                                    color:  
                                        Colors.blue), //apply style to all  
                                children: [  
                                    TextSpan(  

```

```
        text: 'Kar',  
        style: GoogleFonts.poppins(  
            color: Colours.textColor,  
            fontSize: 40.sp,  
            fontWeight: FontWeight.w700,  
            decoration: TextDecoration.none,  
        ),  
        ),  
    TextSpan(  
        text: 'Khana',  
        style: GoogleFonts.poppins(  
            color: Colours.secondaryColor,  
            fontSize: 40.sp,  
            fontWeight: FontWeight.w700,  
            decoration: TextDecoration.none,  
        )),  
    ])),  
    smallText(  
        text: "Register Your Business",  
        size: 16,  
        color: Colours.secondaryColor,  
    )  
],
```

```
        ),  
        ),  
        Center(  
            child: Column(  
                children: [  
                    Lottie.asset('assets/lottie/growBusiness2.json',  
                        height: 190.h, width: 350.w),  
                    SizedBox(height: 10.h),  
                    smallText(  
                        text:  
                            "KarKhana helps your business to connect with",  
                        size: 13,  
                    ),  
                    smallText(  
                        text:  
                            "customers with great discounts, REGISTER NOW!",  
                        size: 13,  
                    )  
                ],  
            )),  
            SizedBox(height: 20.h),  
            NtextField(  
                controller: _VendorName,
```

```
        name: 'Name',  
  
        prefix: const Icon(Icons.person_outline),  
  
        validator: ValidationOfFields.valName),  
  
    SizedBox(  
  
        height: 25.h,  
  
(  
  
    TextFormField(  
  
        controller: _VendorEmail,  
  
        name: 'Email',  
  
        prefix: const Icon(Icons.mail_outline),  
  
        validator: ValidationOfFields.valEmail,  
  
(  
  
    SizedBox(  
  
        height: 25.h,  
  
(  
  
    TextFormField(  
  
        controller: _VendorPhone,  
  
        name: 'Phone No.',  
  
        prefix: const Icon(Icons.phone_outlined),  
  
        validator: ValidationOfFields.valPhone,  
  
(  
  
    SizedBox(  
  
        height: 25.h,
```

```
        ),  
  
        TextFormField(  
            controller: _VendorLocation,  
            name: 'Location',  
            prefix: const Icon(Icons.location_city),  
            validator: ValidationOfFields.valField,  
        ),  
  
        SizedBox(  
            height: 25.h,  
        ),  
  
        TextFormField(  
            controller: _VendorAbout,  
            name: 'About Business',  
            prefix: const Icon(Icons.business_center_outlined),  
            validator: ValidationOfFields.valField,  
        ),  
  
        SizedBox(  
            height: 25.h,  
        ),  
  
        TextFormField(  
            controller: _VendorOpeningDays,  
            name: 'Opening days',  
            prefix: const Icon(Icons.calendar_month_outlined),  
        ),
```

```
validator: ValidationOfFields.valField,  
(  
),  
SizedBox(  
height: 25.h,  
(  
),  
NtextField(  
controller: _VendorOpeningTime,  
name: 'Opening Time',  
prefix: const Icon(Icons.watch_later_outlined),  
validator: ValidationOfFields.valField,  
(  
),  
SizedBox(  
height: 25.h,  
(  
),  
NtextField(  
controller: _UserPw,  
name: 'Password',  
obsecure: true,  
prefix: const Icon(Icons.lock_outline),  
isPassword: true,  
validator: ValidationOfFields.valPassword,  
(  
),  
SizedBox(  
)
```

```
height: 25.h,  
,  
NtextField(  
controller: _UserPwConfirm,  
name: 'Confirm Password',  
obsecure: true,  
prefix: const Icon(Icons.lock_outline),  
isPassword: true,  
validator: ValidationOfFields.valPassword,  
,  
SizedBox(  
height: 25.h,  
,  
ButtonContainer(  
butborderColor: Colours.secondaryColor,  
text: 'Signup',  
butColor: Colours.secondaryColor,  
onClick: () async {  
final isValidForm =  
    _formKey.currentState!.validate();  
if (isValidForm) {  
var authResponse = await VendorRequestRegister(  
    _VendorName.text,
```

```
        _VendorEmail.text,  
        _VendorPhone.text,  
        _VendorLocation.text,  
        _VendorAbout.text,  
        _VendorOpeningDays.text,  
        _VendorOpeningTime.text,  
    );  
  
    // ignore: use_build_context_synchronously  
    showDialog(  
        context: context,  
        builder: (BuildContext context) =>  
            AlertDialog(  
                title: LargeText(  
                    text: 'Registration Submitted'),  
                content: const Text(  
                    "Your Business data has successfully submitted in KarKhana.  
                    We will be in contact with you in few days for further registration process. Please don't  
                    lose your contact information that you have just submitted!",  
                ),  
                actions: [  
                    TextButton(  
                        onPressed: () {  
                            Navigator.pop(context);  
                        }  
                    )  
                ]  
            )  
    );  
}
```

```
        },  
        child: const Text('Close'))  
    ],  
);  
  
setState(() {  
    _VendorName.clear();  
    _VendorEmail.clear();  
    _VendorPhone.clear();  
    _VendorLocation.clear();  
    _VendorAbout.clear();  
    _VendorOpeningDays.clear();  
    _VendorOpeningTime.clear();  
});  
}  
},  
),  
SizedBox(  
    height: 30.h,  
),  
Row(  
    mainAxisAlignment: MainAxisAlignment.center,  
    children: [  
        smallText(text: "Already have an Account? "),  
    ]  
);
```

```
GestureDetector(  
    onTap: () {  
        Navigator.of(context).pushReplacement(  
            MaterialPageRoute(  
                builder: (context) =>  
                    VendorLoginPage()));  
    },  
    child: smallText(  
        text: "Login",  
        color: Colours.borderColor,  
    ),  
),  
],  
(  
    SizedBox(  
        height: 20.h,  
    ),  
),  
],  
(  
    ),  
],  
(  
),  
),
```

```
    ),  
    ),  
);  
}  
  
@override  
void dispose() {  
    _VendorEmail.dispose();  
    _VendorName.dispose();  
    _VendorPhone.dispose();  
    _VendorLocation.dispose();  
    super.dispose();  
}  
}
```

**Vendor Login UI**

```
// ignore_for_file: file_names, prefer_const_constructors_in_immutables,
unnecessary_new, non_constant_identifier_names, prefer_final_fields,
prefer_const_constructors, invalid_use_of_visible_for_testing_member,
invalid_use_of_protected_member, use_build_context_synchronously

import 'package:karkhana/API/auth_api.dart';

import 'package:karkhana/Packages/Packages.dart';

class VendorLoginPage extends StatefulWidget {

  VendorLoginPage({
    super.key,
  });

  @override
  State<VendorLoginPage> createState() => _VendorLoginPageState();
}

class _VendorLoginPageState extends State<VendorLoginPage> {
  TextEditingController _UserEmail = new TextEditingController();
  TextEditingController _UserPw = new TextEditingController();

  GlobalKey<FormState> _formKey = new GlobalKey<FormState>();
```

```
@override  
  
void initState() {  
  
    // TODO: implement initState  
  
    _UserEmail;  
  
    _UserPw;  
  
    super.initState();  
  
}  
  
  
@override  
  
Widget build(BuildContext context) {  
  
    return GestureDetector(  
  
        onTap: () {  
  
            FocusScopeNode currentFocus = FocusScope.of(context);  
  
  
            if (!currentFocus.hasPrimaryFocus) {  
  
                currentFocus.unfocus();  
  
            }  
  
        },  
  
        child: Scaffold(  
  
            resizeToAvoidBottomInset: false,  
  
            backgroundColor: Colours.backgroundColor,  
  
            body: SafeArea(  
  
                child: SingleChildScrollView(  
                   
```

```
child: Form(  
  key: _formKey,  
  child: Column(  
    children: [  
      Padding(  
        padding: EdgeInsets.only(top: 10.h),  
        child: Row(  
          mainAxisAlignment: MainAxisAlignment.start,  
          children: [  
            SizedBox(  
              width: 10.w,  
            ),  
            GestureDetector(  
              child: Icon(Icons.arrow_back_ios),  
              onTap: () {  
                Navigator.pop(context);  
              },  
            ),  
          ],  
        ),  
      ),  
      Padding(  
        padding:
```

```
EdgeInsets.symmetric(horizontal: 30.w, vertical: 5.h),  
child: Column(  
    crossAxisAlignment: CrossAxisAlignment.start,  
    children: [  
        Center(  
            child: Column(  
                children: [  
                    RichText(  
                        text: TextSpan(  
                            style: const TextStyle(  
                                color:  
                                    Colors.blue), //apply style to all  
                            children: [  
                                TextSpan(  
                                    text: 'Kar',  
                                    style: GoogleFonts.poppins(  
                                        color: Colours.textColor,  
                                        fontSize: 40.sp,  
                                        fontWeight: FontWeight.w700,  
                                        decoration: TextDecoration.none,  
                                    ),  
                                ),  
                                TextSpan(  
                                    text: ' ',  
                                    style: GoogleFonts.poppins(  
                                        color: Colours.textColor,  
                                        fontSize: 40.sp,  
                                        fontWeight: FontWeight.w700,  
                                        decoration: TextDecoration.none,  
                                    ),  
                                ),  
                            ],  
                        ),  
                    ),  
                ],  
            ),  
        ),  
    ],  
),
```

```
text: 'Khana',  
style: GoogleFonts.poppins(  
color: Colours.secondaryColor,  
fontSize: 40.sp,  
fontWeight: FontWeight.w700,  
decoration: TextDecoration.none,  
)),  
]),  
smallText(  
text: "Login into your Business Account",  
size: 16,  
color: Colours.secondaryColor,  
)  
],  
),  
),  
Column(  
children: [  
Lottie.asset('assets/lottie/growBusiness.json',  

```

```
        size: 13,  
    )  
],  
,  
SizedBox(height: 20.h),  
NtextField(  
    controller: _UserEmail,  
    name: 'Email',  
    prefix: const Icon(Icons.person_outline),  
    validator: ValidationOfFields.valEmail,  
,  
SizedBox(  
    height: 25.h,  
,  
NtextField(  
    controller: _UserPw,  
    name: 'Password',  
    obscure: true,  
    isPassword: true,  
    prefix: const Icon(Icons.lock_outline),  
    validator: ValidationOfFields.valPassword,  
,  
SizedBox(
```

```
height: 8.h,  
,  
GestureDetector(  
    child: smallText(  
        text: "Forget Password?",  
        decoration: TextDecoration.underline,  
,  
    onTap: () {  
        showModalBottomSheet(  
            shape: const RoundedRectangleBorder(  
                borderRadius: BorderRadius.vertical(  
                    top: Radius.circular(50.0),  
                ),  
            ),  
            context: context,  
            isScrollControlled: true,  
            builder: (context) {  
                return SizedBox(  
                    height: 400.h,  
                    child: VendorForgetPw(),  
                );  
            },  
        );  
    },  
);
```

```
    },  
    ),  
    SizedBox(  
      height: 30.h,  
    ),  
    ButtonContainer(  
      text: 'Login',  
      onClick: () async {  
        final isValidForm =  
          _formKey.currentState!.validate();  
        if (isValidForm) {  
          var authResponse =  
            await userAuth(_UserEmail.text, _UserPw.text);  
          if (authResponse.runtimeType == String) {  
            showDialog(  
              context: context,  
              builder: (BuildContext context) =>  
                AlertDialog(  
                  title: const Text('Invalid Input'),  
                  content: Text(  
                    authResponse,  
                  ),  
                  actions: [  

```

```
TextButton(  
    onPressed: () {  
        Navigator.pop(context);  
    },  
    child: const Text('Close'))  
,  
));  
  
} else if (authResponse.runtimeType == User) {  
  
    User user = authResponse;  
  
    context.read<UserCubit>().emit(user);  
  
    if (user.is_user == 2) {  
  
        ScaffoldMessenger.of(context).showSnackBar(  
            const SnackBar(  
                duration:  
                    Duration(milliseconds: 1000),  
                backgroundColor: Colors.green,  
                content: Text(  
                    "Logged in",  
                    textAlign: TextAlign.center,  
                )));  
  
        Navigator.of(context).pushReplacement(  
            MaterialPageRoute(  
                builder: (context) =>
```

```
        VendorHomePage());  
    } else {  
        await logOutUser(user.token!);  
        showDialog(  
            context: context,  
            builder: (BuildContext context) =>  
                AlertDialog(  
                    title: const Text('Invalid Input'),  
                    content: const Text(  
                        'Unable to log in with provided credentials.'),  
                    actions: [  
                        TextButton(  
                            onPressed: () async {  
                                Navigator.pop(context);  
                            },  
                            child: const Text('Close')),  
                    ],  
                ));  
    }  
},  
butColor: Colours.secondaryColor,
```

```
butborderColor: Colours.secondaryColor,  
(  
),  
SizedBox(  
height: 40.h,  
(  
),  
Row(  
mainAxisAlignment: MainAxisAlignment.center,  
children: [  
smallText(  
text: "Have Your Own Business? ",  
(  
),  
GestureDetector(  
onTap: () {  
Navigator.push(  
context,  
MaterialPageRoute(  
builder: (context) =>  
VendorSignUp()));  
},  
child: smallText(  
text: "Register",  
color: Colours.borderColor,  
))
```

```
    ],
),
],
),
),
],
),
),
),
),
),
),
),
),
),
),
),
),
);
}

@Override
void dispose() {
    _UserEmail.dispose();
    _UserPw.dispose();

    super.dispose();
}

}
```

**Vendor Homepage UI**

```
// ignore_for_file: prefer_const_constructors,  
prefer_const_literals_to_create_immutables, file_names, use_super_parameters,  
prefer_const_constructors_in_immutables, unnecessary_string_escapes,  
use_key_in_widget_constructors, use_build_context_synchronously,  
non_constant_identifier_names, sized_box_for_whitespace
```

```
import 'package:karkhana/API/auth_api.dart';  
  
import 'package:karkhana/Packages/Packages.dart';
```

```
class VendorHomePage extends StatefulWidget {  
  
  VendorHomePage({super.key});
```

```
  @override  
  
  State<VendorHomePage> createState() => _VendorHomePageState();  
  
}
```

```
class _VendorHomePageState extends State<VendorHomePage> {
```

```
  @override  
  
  Widget build(BuildContext context) {
```

```
    User user = context.read<UserCubit>().state;
```

```
    double width = MediaQuery.of(context).size.height * 1;
```

```
int id = user.id!;

return Scaffold(
    appBar: PreferredSize(
        preferredSize: Size.fromHeight(90.h),
        child: SafeArea(
            child: Container(
                margin: EdgeInsets.only(top: 15.h, bottom: 15.h),
                padding: EdgeInsets.only(left: 20.w, right: 20.w),
                child: Row(
                    mainAxisAlignment: MainAxisAlignment.spaceBetween,
                    children: [
                        Column(
                            crossAxisAlignment: CrossAxisAlignment.start,
                            children: [
                                Padding(
                                    padding: EdgeInsets.only(top: 10.h),
                                    child: LargeText(
                                        text: "Karkhana",
                                        color: Colours.secondaryColor,
                                    ),
                                ),
                            ],
                        ),
                    ],
                ),
            ),
        ),
    ),
)
```

```
Center(  
    child: InkWell(  
        onTap: () {  
            Navigator.of(context).push(PageTransition(  
                type: PageTransitionType.rightToLeft,  
                child: VendorSetting()));  
        },  
        child: Container(  
            width: 45.w,  
            height: 45.h,  
            decoration: BoxDecoration(  
                borderRadius: BorderRadius.circular(15),  
                color: Colours.secondaryColor,  
            ),  
            child: const Icon(  
                Icons.menu,  
                color: Colours.backgroundColor,  
            ),  
        ),  
    ),  
),  
],  
,
```

```
        ),  
        ),  
        ),  
  
floatingActionButton: FloatingActionButton(  
  
    onPressed: () {  
  
        Navigator.of(context).push(PageTransition(  
  
            type: PageTransitionType.rightToLeft,  
  
            child: VendorAddDeals(),  
  
        ));  
  
    },  
  
    backgroundColor: Colours.secondaryColor,  
  
    child: const Icon(Icons.add)),  
  
body: SingleChildScrollView(  
  
    child: SafeArea(  
  
        child: Column(children: [  
  
            Container(  
  
                width: width,  
  
                color: Colors.transparent,  
  
                child: Stack(  
  
                    children: [  
  
                        Padding(  
  
                            padding: EdgeInsets.only(top: 60.h),  
  
                            child: Container(  

```

```
decoration: BoxDecoration(  
    color: Colours.backgroundColor,  
    borderRadius: const BorderRadius.only(  
        topLeft: Radius.circular(40),  
        topRight: Radius.circular(40),  
    ),  
    boxShadow: [  
        BoxShadow(  
            color: Colors.grey.withOpacity(0.2),  
            spreadRadius: 0.09,  
            blurRadius: 0.2,  
            offset: const Offset(  
                0.2, -3) // changes position of shadow  
        ),  
    ],  
,  
    child: Padding(  
        padding: EdgeInsets.symmetric(  
            horizontal: 30.w, vertical: 10.h),  
        child: Column(  
            crossAxisAlignment: CrossAxisAlignment.start,  
            children: [  
                SizedBox(  

```

```
height: 180.h,  
,  
LargeText(text: "Recently Added Deals"),  
FutureBuilder<List<Deals?>>(  
future: SpecificDealsData(id),  
builder: (context, snapshot) {  
// Widget widget = const Text("");  
if (snapshot.connectionState ==  
ConnectionState.waiting) {  
return const Center(  
child: CircularProgressIndicator());  
} else if (snapshot.hasError) {  
return const Center(  
child: Text("Something went wrong"),  
);  
} else {  
if (snapshot.hasData) {  
if (snapshot.hasData &&  
snapshot.data!.isEmpty) {  
return Padding(  
padding:  
EdgeInsets.only(bottom: 20.h),  
child: Container(  
;
```

```
width: width,  
child: Column(  
    children: [  
        Lottie.asset(  
            "assets/lottie/noDealsFound.json",  
            height: 180,  
            width: 250,  
        ),  
        LargeText(  
            text:  
                "No Recently Added Deals!",  
            size: 18,  
        ),  
    ],  
),  
(,),  
);  
}  
else {  
    return Column(  
        children: [  
            Center(  
                child: Column(  
                    children: snapshot.data!,  
                ),  
            ),  
        ],  
    );  
}  
}
```

```
.map<Widget>((e) {  
    return RecentContainer(  
        dealTitle:  
            "${e!.Deals_title}",  
        businessName: "${user.name}",  
        discount: e.Deals_discount,  
        frontLabel: "Status",  
        newLabel: "Added",  
        smallboxColor:  
            Colours.greenColor,  
        smallboxWidght: 0.3,  
    );  
});  
],  
);  
}  
} else {  
    return Padding(  
        padding:  
            EdgeInsets.symmetric(vertical: 10),  
        child: Column(  
            children: [  
               
```



```
child: Material(  
    elevation: 8,  
    shape: const CircleBorder(),  
    child: Container(  
        width: 130,  
        height: 130,  
        decoration: const BoxDecoration(  
            shape: BoxShape.circle,  
        ),  
        child: Padding(  
            padding: const EdgeInsets.all(6.0),  
            child: ClipOval(  
                child: Image.asset(  
                    "assets/images/businessprofile.jpg",  
                    fit: BoxFit.cover,  
                ),  
            ),  
        ),  
    ),  
    child: SizedBox(  
        height: 10.h,
```

```
        ),  
        LargeText(  
            text: user.name!,  
        ),  
        SizedBox(  
            height: 10.h,  
        ),  
        ButtonContainer(  
            butborderColor: Colours.secondaryColor,  
            text: "Upload Profile",  
            butColor: Colours.secondaryColor,  
            onClick: () async {  
                int pk = user.id!;  
  
                String BusinessName = user.name!;  
  
                String BusinessEmail = user.email!;  
  
                String BusinessPhone = user.phonenumber!;  
  
                String BusinessToken = user.token!;  
  
                String BusinessLocation = user.location!;  
  
                String BusinessAbout = user.VendorBusiness_about!;  
  
                String BusinessOpenDays =  
                    user.VendorBusiness_opening_days!;  
  
                String BusinessOpenTime =  
                    user.VendorBusiness_opening_time!;
```

```
var authResponse = await postVendorBusinessDetails(  
    pk,  
    BusinessName,  
    BusinessEmail,  
    BusinessPhone,  
    BusinessToken,  
    BusinessLocation,  
    BusinessAbout,  
    BusinessOpenDays,  
    BusinessOpenTime,  
);  
  
if (authResponse.runtimeType == String) {  
    showDialog(  
        context: context,  
        builder: (BuildContext context) =>  
            AlertDialog(  
                title: LargeText(  
                    text: 'Business Information Upload',  
                    size: 16,  
                ),  
                content: const Text(  
                    "Your Business information has been successfully uploaded to  
                    KarKhana system.",  
                ),  
            ),  
    );  
}
```

```
        ),  
  
        actions: [  
  
            TextButton(  
  
                onPressed: () {  
  
                    Navigator.pop(context);  
  
                },  
  
                child: const Text('Close'))  
  
            ],  
  
        )),  
  
    } else {  
  
        showDialog(  
  
            context: context,  
  
            builder: (BuildContext context) =>  
  
                AlertDialog(  
  
                    title: LargeText(  
  
                        text: 'Business Information Upload',  
  
                        size: 16,  
  
                    ),  
  
                    content: const Text(  
  
                        "Your Business information has already been uploaded to  
KarKhana system.",  
  
                    ),  
  
                    actions: [  
  
                ),  
  
            ),  
  
        );  
  
    }  
}
```

```
TextButton(  
    onPressed: () {  
        Navigator.pop(context);  
    },  
    child: const Text('Close'))  
,  
),  
},  
height: 5,  
width: 130,  
,  
],  
,  
],  
,  
],  
,  
),  
],  
,  
),  
));  
}  
}
```

**Vendor Add Deals UI**

```
// ignore_for_file: file_names, unused_import, prefer_final_fields, unused_field,
unnecessary_new,           avoid_print,           non_constant_identifier_names,
prefer_const_declarations, prefer_const_constructors

import 'dart:convert';

import 'dart:ffi';

import 'dart:io';

import 'package:karkhana/Packages/Packages.dart';

import 'package:dropdownfield2/dropdownfield2.dart';

import 'package:image_picker/image_picker.dart';

import 'package:intl/intl.dart';

import 'package:http/http.dart' as http;

class VendorAddDeals extends StatefulWidget {

  const VendorAddDeals({super.key});

  @override
  State<VendorAddDeals> createState() => _VendorAddDealsState();
}

class _VendorAddDealsState extends State<VendorAddDeals> {
```

```
TextEditingController _dealtile = TextEditingController();  
TextEditingController _dealphoto = TextEditingController();  
TextEditingController _dealcategory = TextEditingController();  
TextEditingController _dealdesc = TextEditingController();  
TextEditingController _dealcondition = TextEditingController();  
TextEditingController _dealdiscount = TextEditingController();  
TextEditingController _dealprice = TextEditingController();  
TextEditingController _dealapplicabletime = TextEditingController();  
TextEditingController _dealapplicableday = TextEditingController();  
TextEditingController _dealexpirydate = TextEditingController();
```

```
 GlobalKey<FormState> _formKey = new GlobalKey<FormState>();  
File? image;
```

```
 Future pickImage() async {  
   try {  
     final image = await ImagePicker().pickImage(source: ImageSource.gallery);  
     if (image == null) return;  
  
     final imageTemporary = File(image.path);  
     setState(() {  
       this.image = imageTemporary;  
     });  
   };
```

```
    } on PlatformException catch (e) {  
  
    print('Failed to pick image: $e');  
    }  
}  
  
@override  
Widget build(BuildContext context) {  
  
    User user = context.read<UserCubit>().state;  
  
    String BusinessName = user.name!;  
  
    String BusinessLocation = user.location!;  
  
    String BusinessPhone = user.phonenumber!;  
  
    int pk = user.id!;  
  
  
    Future<void> postDeals(  
        String businessName,  
        String businessLocation,  
        String dealTitle,  
        String BusinessPhone,  
        int price,  
        String dealCategory,  
        String dealDescription,  
        String dealCondition,  
        int discount,
```

```
String dealApplicableTime,  
String dealApplicableDay,  
String dealExpiryDate,  
int pk,  
File? image,  
) async {  
  
final url = '$baseUrl/accounts/deals/';  
  
String PK = pk.toString();  
  
final request = http.MultipartRequest('POST', Uri.parse(url));  
  
  
// Add image file to the request  
  
if (image != null) {  
  
final imageField = await http.MultipartFile.fromPath(  
    'deal_photo',  
    image.path,  
);  
  
request.files.add(imageField);  
}  
  
  
// Add other fields to the request  
  
request.fields['BusinessName'] = businessName;  
request.fields['BusinessLocation'] = businessLocation;  
request.fields['Business_Phone'] = BusinessPhone;
```

```
request.fields['deal_title'] = dealTitle;  
request.fields['deal_price'] = price.toString();  
request.fields['deal_category'] = dealCategory;  
request.fields['deal_desc'] = dealDescription;  
request.fields['deal_condition'] = dealCondition;  
request.fields['deal_discount'] = discount.toString();  
request.fields['deal_applicable_time'] = dealApplicableTime;  
request.fields['deal_applicable_day'] = dealApplicableDay;  
request.fields['deal_expiry_date'] = dealExpiryDate;  
request.fields['user'] = PK;  
  
final response = await request.send();  
  
if (response.statusCode == 200 || response.statusCode == 201) {  
    setState(() {  
        _dealttitle.clear();  
        _dealprice.clear();  
        _dealcategory.clear();  
        _dealdesc.clear();  
        _dealcondition.clear();  
        _dealdiscount.clear();  
        _dealapplicabletime.clear();  
        _dealapplicableday.clear();  
    });  
}
```

```
_dealexpirydate.clear();

});

// ignore: use_build_context_synchronously

return showDialog(
  context: context,
  builder: (BuildContext context) => AlertDialog(
    title: LargeText(text: 'Deal Success'),
    content: const Text(
      "Deal has been successfully added in KarKhana system.",
    ),
    actions: [
      TextButton(
        onPressed: () {
          Navigator.pop(context);
        },
        child: const Text('Close'))
    ],
  )));
}

} else {
  // ignore: use_build_context_synchronously
  return showDialog(
    context: context,
    builder: (BuildContext context) => AlertDialog(
```

```
title: LargeText(text: 'Deal Failed'),  
  
content: const Text(  
  
    "Deal was not added into KarKhana system as you have not uploaded your  
profile. Upload your business profile first.",  
  
,  
  
actions: [  
  
    TextButton(  
  
        onPressed: () {  
  
            Navigator.pop(context);  
  
        },  
  
        child: const Text('Close'))  
  
    ],  
  
));  
  
}  
  
}  
  
  
// Future<dynamic> postDeals(  
  
// String BusinessName,  
  
// String BusinessLocation,  
  
// String deal_Title,  
  
// int deal_price,  
  
// String deal_category,  
  
// String deal_desc,
```

```
// String deal_condition,  
  
// int deal_discount,  
  
// String deal_applicable_time,  
  
// String deal_applicable_day,  
  
// String deal_expiry_date,  
  
// int user,  
  
// File? image,  
  
// ) async {  
  
//   var url = Uri.parse("$baseUrl/accounts/deals/");  
  
//   var res = await http.post(url,  
  
//     headers: <String, String>{  
  
//       'Content-Type': 'application/json; charset=UTF-8',  
  
//     },  
  
//     body: jsonEncode(<String, dynamic>{  
  
//       "BusinessName": BusinessName,  
  
//       "BusinessLocation": BusinessLocation,  
  
//       "Business_Phone": BusinessPhone,  
  
//       "deal_title": deal_Title,  
  
//       "deal_photo": image,  
  
//       "deal_price": deal_price,  
  
//       "deal_category": deal_category,  
  
//       "deal_desc": deal_desc,  
  
//       "deal_condition": deal_condition,
```

```
//      "deal_discount": deal_discount,  
  
//      "deal_applicable_time": deal_applicable_time,  
  
//      "deal_applicable_day": deal_applicable_day,  
  
//      "deal_expiry_date": deal_expiry_date,  
  
//      "user": user,  
  
//    }));  
  
//  if (res.statusCode == 200 || res.statusCode == 201) {  
  
//    setState(() {  
  
//      _dealttitle.clear();  
  
//      _dealprice.clear();  
  
//      _dealcategory.clear();  
  
//      _dealdesc.clear();  
  
//      _dealcondition.clear();  
  
//      _dealdiscount.clear();  
  
//      _dealapplicabletime.clear();  
  
//      _dealapplicableday.clear();  
  
//      _dealexpirydate.clear();  
  
//    });  
  
//    // ignore: use_build_context_synchronously  
  
//    return showDialog(  
  
//      context: context,  
  
//      builder: (BuildContext context) => AlertDialog(  
  
//        title: LargeText(text: 'Deal Success'),
```

```
//           content: const Text(  
  
//           "Deal has been successfully added in BriskDeals system.",  
  
//           ),  
  
//           actions: [  
  
//           TextButton(  
  
//           onPressed: () {  
  
//               Navigator.pop(context);  
  
//           },  
  
//           child: const Text('Close')  
  
//           ],  
  
//       ));  
  
//   } else {  
  
//     // ignore: use_build_context_synchronously  
  
//   return showDialog(  
  
//       context: context,  
  
//       builder: (BuildContext context) => AlertDialog(  
  
//           title: LargeText(text: 'Deal Failed'),  
  
//           content: const Text(  
  
//           "Deal was not added into KarKhana system as you have not uploaded your  
profile. Upload your business profile first.",  
  
//           ),  
  
//           actions: [  
  
//           TextButton(  

```

```
//           onPressed: () {  
//             Navigator.pop(context);  
//           },  
//           child: const Text('Close')  
//         ],  
//       );  
//     }  
//   }  
  
// Category list  
  
List<String> categories = [  
  "Swimming Deals",  
  "Food Deals",  
  "Car Service Deals",  
  "Health Deals",  
  "Cosmetic Deals",  
  "Event Deals",  
];  
  
return GestureDetector(  
  onTap: () {  
    FocusScopeNode currentFocus = FocusScope.of(context);  
  },  
);
```

```
if (!currentFocus.hasPrimaryFocus) {  
    currentFocus.unfocus();  
}  
,  
child: Scaffold(  
body: SafeArea(  
    child: SingleChildScrollView(  
        child: Column(  
            children: [  
                Padding(  
                    padding: EdgeInsets.only(  
                        top: 20.h,  
                    ),  
                ),  
                child: Row(  
                    mainAxisAlignment: MainAxisAlignment.start,  
                    children: [  
                        SizedBox(  
                            width: 10.w,  
                        ),  
                        GestureDetector(  
                            child: Icon(Icons.arrow_back_ios),  
                            onTap: () {  
                                Navigator.pop(context);  
                            },  
                        ),  
                    ],  
                ),  
            ],  
        ),  
    ),  
);
```

```
        },  
        ),  
        ],  
        ),  
        ),  
        ),  
        Form(  
            key: _formKey,  
            child: Padding(  
                padding:  
                    EdgeInsets.symmetric(vertical: 20.h, horizontal: 30.w),  
                child: Column(  
                    children: [  
                        Padding(  
                            padding: EdgeInsets.only(bottom: 20.h),  
                            child: LargeText(  
                                text: "Add Your Deals",  
                                size: 25,  
                            ),  
                        ),  
                        LottieBuilder.asset(  
                            "assets/lottie/adddeals.json",  
                            height: 180.h,  
                            width: 300.w,
```

```
),  
SizedBox(  
    height: 20.h,  
,  
NtextField(  
    controller: _dealtile,  
    name: "Deal title",  
    validator: ValidationOfFields.valField),  
SizedBox(  
    height: 20.h,  
,  
Container(  
    width: 350,  
    decoration: BoxDecoration(  
        border: Border.all(  
            color: Colours.textColor,  
            width: 1,  
,  
        borderRadius: BorderRadius.circular(10),  
        boxShadow: [  
            BoxShadow(  
                color: Colours.backgroundColor.withOpacity(0.5),  
                spreadRadius: 1.1,
```

```
        blurRadius: 1.1,  
        // changes position of shadow  
    ),  
],  
(  
    ),  
    child: DropDownField(  
        controller: _dealcategroy,  
        hintText: "Select Category",  
        items: categories,  
        textStyle: GoogleFonts.lato(  
            color: Colours.textColor,  
            fontSize: 14.sp,  
            fontWeight: FontWeight.w400,  
            decoration: TextDecoration.none,  
        ),  
    ),  
    ),  
    ),  
    SizedBox(  
        height: 20.h,  
    ),  
    Row(  
        mainAxisAlignment: MainAxisAlignment.spaceBetween,  
        children: [  

```

```
image != null

    ? Image.file(
        image!,
        width: 150,
        height: 150,
    )

: Container(
    height: 150,
    width: 150,
    decoration: BoxDecoration(
        // Replace Colors.blue with the desired background color
        border: Border.all(
            color: Colours
                .greyColor, // Replace Colors.black with the desired border
            color
        ),
        width:
            2.0, // Replace 2.0 with the desired border width
    ),
),
),
child: Center(
    child: smallText(text: "Your Image")),
Center(
    child: InkWell(

```

```
onTap: () {  
    pickImage();  
},  
  
child: Container(  
  
    width: 140.w,  
  
    height: 45.h,  
  
    decoration: BoxDecoration(  
  
        borderRadius: BorderRadius.circular(15),  
  
        color: Colours.secondaryColor,  
    ),  
  
    child: Row(  
  
        mainAxisAlignment: MainAxisAlignment.center,  
  
        children: [  
  
            const Icon(  
  
                Icons.image_outlined,  
  
                color: Colours.backgroundColor,  
            ),  
  
            smallText(  
  
                text: "Pick your Image",  
  
                color: Colours.backgroundColor,  
            ),  
  
            SizedBox(  
  
                width: 2.w,
```

```
        ),  
        ],  
        ),  
        ),  
        ),  
    )  
],  
(  
    ),  
    SizedBox(  
        height: 20.h,  
    ),  
    TextFormField(  
        controller: _dealdesc,  
        name: "Deal description",  
        validator: ValidationOfFields.valField,  
        needArea: true,  
    ),  
    SizedBox(  
        height: 20.h,  
    ),  
    TextFormField(  
        controller: _dealcondition,  
        name: "Deal condition",  
    )
```

```
validator: ValidationOfFields.valField,  
needArea: true,  
,  
SizedBox(  
height: 20.h,  
,  
NtextField(  
controller: _dealdiscount,  
name: "Deal discount",  
validator: ValidationOfFields.valField),  
SizedBox(  
height: 20.h,  
,  
NtextField(  
controller: _dealprice,  
name: "Market Price of item/service",  
validator: ValidationOfFields.valField),  
SizedBox(  
height: 20.h,  
,  
NtextField(  
controller: _dealapplicabletime,  
name: "Deal applicable time",
```

```
        validator: ValidationOfFields.valField),  
  
        SizedBox(  
  
            height: 20.h,  
  
  
        TextFormField(  
  
            controller: _dealapplicableday,  
  
            name: "Deal applicable day",  
  
            validator: ValidationOfFields.valField),  
  
        SizedBox(  
  
            height: 20.h,  
  
        ),  
  
        TextFormField(  
  
            controller: _dealexpirydate,  
  
            name: "Deal expiry date",  
  
            validator: ValidationOfFields.valField,  
  
            ontap: () async {  
  
                DateTime? showDate = await showDatePicker(  
  
                    context: context,  
  
                    initialDate: DateTime.now(),  
  
                    firstDate: DateTime(1990),  
  
                    lastDate: DateTime(2700));  
  
  
                if (showDate != null) {
```

```
        setState(() {  
  
            _dealexpirydate.text =  
                DateFormat('yyyy-MM-dd').format(showDate);  
        });  
    }  
},  
,  
SizedBox(  
    height: 20.h,  
,  
ButtonContainer(  
    butborderColor: Colours.secondaryColor,  
    text: "Post Your Deal",  
    butColor: Colours.secondaryColor,  
    onClick: () {  
        final isValidForm =  
            _formKey.currentState!.validate();  
  
        if (isValidForm) {  
            int price = int.parse(_dealprice.text);  
            int discount = int.parse(_dealdiscount.text);  
            postDeals(  
                BusinessName,
```

```
BusinessLocation,  
_dealtitle.text,  
BusinessPhone,  
price,  
_dealcategory.text,  
_dealdesc.text,  
_dealcondition.text,  
discount,  
_dealapplicabletime.text,  
_dealapplicableday.text,  
_dealexpirydate.text,  
pk,  
image,  
);  
}  
})  
],  
,  
,  
,  
],  
,
```

```
 ),  
 ),  
 );  
 }  
 }
```

**Vendor Purchase UI**

```
// ignore_for_file: file_names, prefer_const_constructors

import 'package:karkhana/Packages/Packages.dart';

class VendorDealPurchased extends StatefulWidget {

  const VendorDealPurchased({super.key});

  @override
  State<VendorDealPurchased> createState() => _VendorDealPurchasedState();
}

class _VendorDealPurchasedState extends State<VendorDealPurchased> {

  @override
  Widget build(BuildContext context) {
    User user = context.read<UserCubit>().state;
    int id = user.id!;
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          showSearch(context: context, delegate: SearchVendorCoupon());
        },
        backgroundColor: Colours.secondaryColor,
```

```
        child: const Icon(Icons.search_outlined)),  
  
    body: DefaultTabController(  
  
        length: 2, // length of tabs  
  
        initialIndex: 0,  
  
        child: Column(children: <Widget>[  
  
            Padding(  
  
                padding: EdgeInsets.only(top: 40.h),  
  
                child: Row(  
  
                    mainAxisAlignment: MainAxisAlignment.start,  
  
                    children: [  
  
                        SizedBox(  
  
                            width: 10.w,  
  
                        ),  
  
                        GestureDetector(  
  
                            child: Icon(Icons.arrow_back_ios),  
  
                            onTap: () {  
  
                                Navigator.pop(context);  
  
                            },  
  
                        ),  
  
                    ],  
  
                ),  
  
            Padding(  
        
```

```
padding: const EdgeInsets.all(8.0),  
child: Center(  
    child: LargeText(  
        text: "Your Customer Purchases",  
        color: Colours.secondaryColor,  
        size: 25,  
    ),  
,  
,  
),  
TabBar(  
    indicatorColor: Colours.secondaryColor,  
    indicatorWeight: 3,  
    labelStyle: GoogleFonts.lato(  
        color: Colours.textColor,  
        fontSize: 15.sp,  
        height: 2,  
        fontWeight: FontWeight.w600,  
    ),  
    labelColor: Colours.secondaryColor,  
    unselectedLabelColor: Colours.greyColor,  
    tabs: const [  
        Tab(text: 'Active Customers'),  
        Tab(text: 'Previous Customers'),
```

```
        ],  
        ),  
        Expanded(  
            child: TabBarView(children: <Widget>[  
                activeVPurchase(),  
                PreviousCoupon(  
                    id: id,  
                    isVendor: true,  
                )  
            ]),  
        )  
    ])),  
);  
}  
}
```

## 8.4 APPENDIX 4: DESIGNS

### 8.4.1 GANTT CHART

#### 8.4.1.1 Final Gantt Chart

Revised and updated Gantt chart of the whole project.

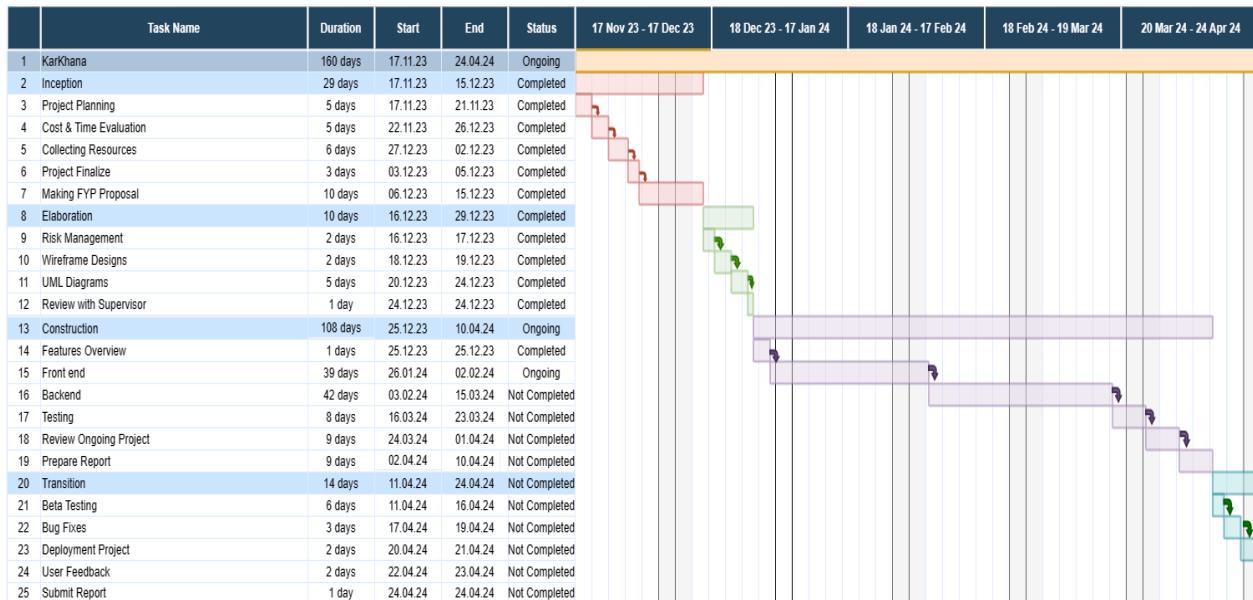


Figure 404:Final Gantt Chart.

#### 8.4.1.2 Gantt Chart Iteration

Initial Gantt Chart concept for the whole project.

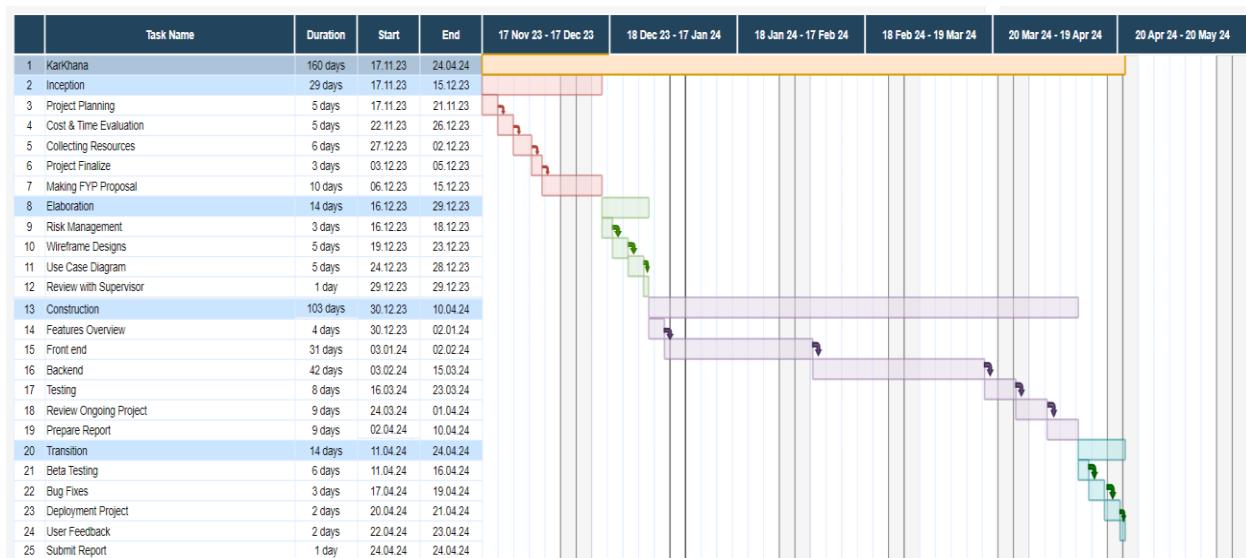


Figure 405: First Iteration of Gantt Chart.

## 8.4.2 Use Case Diagram

### 8.4.2.1 Final Use Case Diagram

Initial Diagram was revised and updated according to the present system and final use case was created.

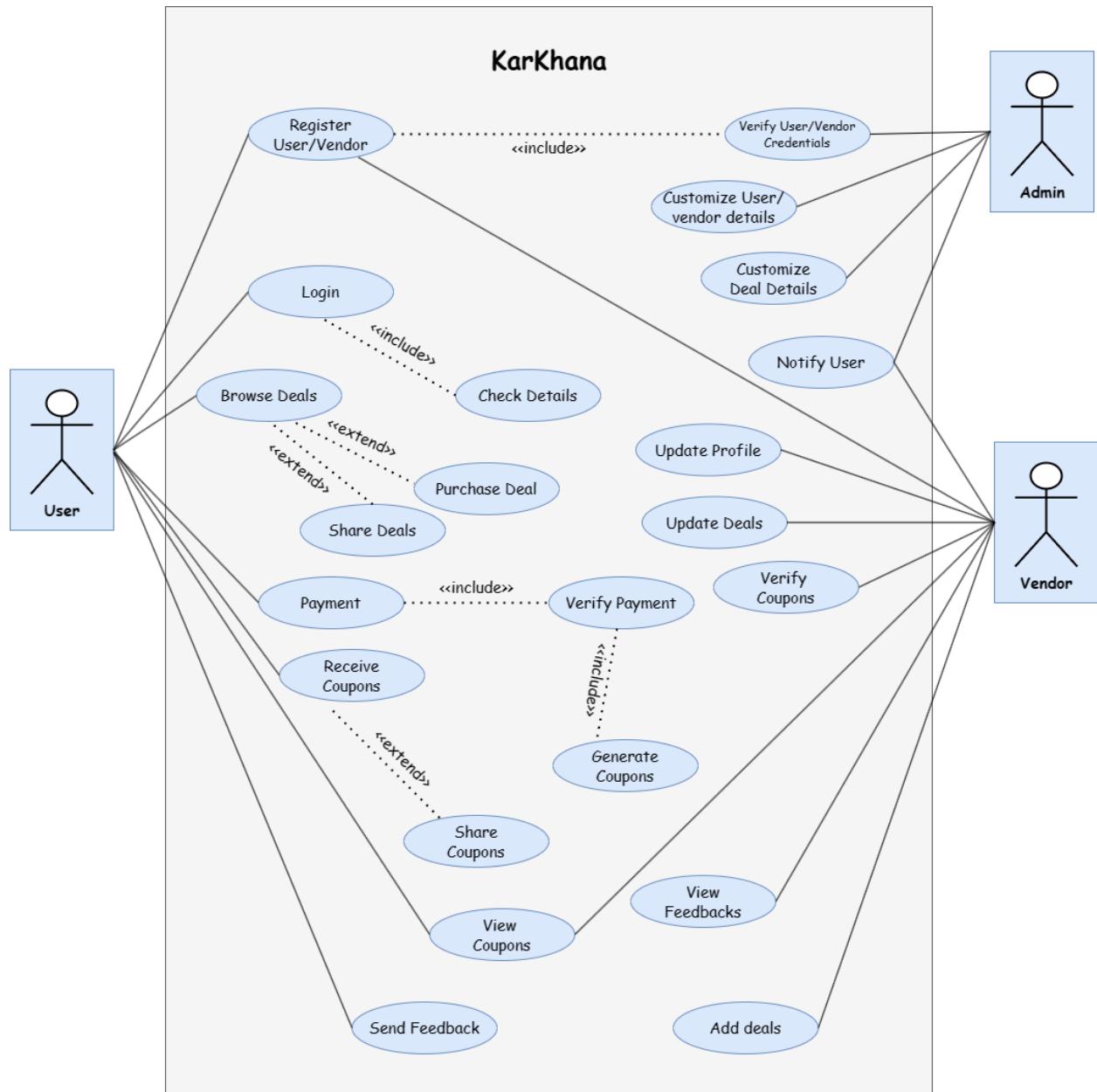


Figure 406: Final Use Case Diagram.

### 8.4.2.2 Initial Use Case Diagram

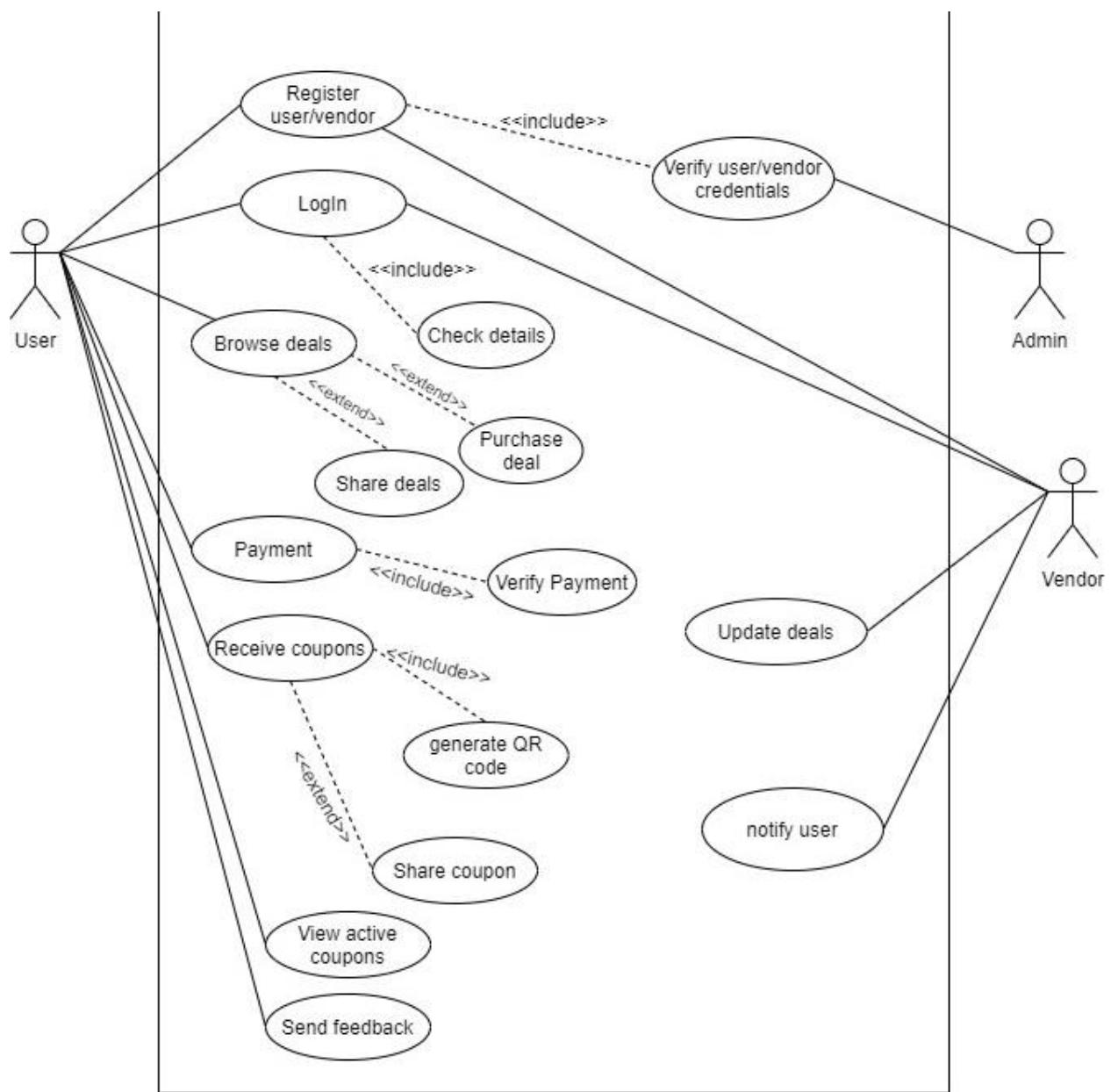


Figure 407: Initial Use Case Diagram.

### 8.4.2.3 High Level Use Case Description

#### User Registration

Use Case	User Registration
Actor	User
Description	User provides their details and start the registration process. They need to verify their email, so an email is sent to them by KarKhana where a token is provided which is further used in the registration process. After verifying the email using that token, a customer will be successfully registered in the system.

Table 65: High Level Use Case of User Registration.

#### Vendor Registration

Use Case	Vendor Registration
Actor	Vendor
Description	First the vendors will submit their business details and send it to Admin. Then, Admin contacts vendor through phone call for further registration process and if everything goes well, admin registers the vendor's business to KarKhana system and notify them through email.

Table 66: High Level Use Case of Vendor Registration.

#### User/Vendor Login

Use Case	Login
Actor	User, vendor
Description	Both user and vendor will have their own respective Login UI from where they will fill up login form and can log into the application.

Table 67: High Level Use Case of User/Vendor Login

### Browse Deals

Use Case	<b>Browse deals</b>
Actor	User
Description	User can browse deals throughout the application to purchase the deals of their likings. They can even use search bar to search for their desired deals faster.

Table 68: High Level Use Case of Browse Deals.

### Payment

Use Case	<b>Payment</b>
Actor	User
Description	Users buy coupons through online payment. For online payment, Khalti payment service will be used in this application.

Table 69: High Level Use Case of Payment.

### View Purchased Coupons

Use Case	<b>View purchased coupon</b>
Actor	User, Vendor
Description	After purchasing the coupon user will be displayed their active coupon in purchase section of the application and at the same time, vendor side will also have purchase section that will display the current coupon of their bought by customers. And there will be a previously purchased section for both vendors and customers which displays the used coupon respectively.

Table 70: High Level Use Case of View Purchased Coupons.

### Share Active Coupons

Use Case	<b>Share active coupon</b>
Actor	User, Vendor
Description	After purchasing and coupon is displayed in the active section of the customer purchases. They can share the active coupons with their friends and family.

Table 71: High Level Use Case of Share Active Coupons.

### Customize User/Vendor Details

Use Case	Customize user/vendor details
Actor	Admin
Description	After the registration of customer and vendors, all the details are stored in the database which are all in control of Admin of the KarKhana system. So, the Admin can customize the details as per needed.

Table 72: High Level Use Case of Customize User/Vendor Details.

### Customize Deal Details

Use Case	Customize deals details
Actor	Admin
Description	After the vendors add deals to the KarKhana system, all the details are stored in the database which are all in control of Admin of the KarKhana system. So, the Admin can also customize the details as per needed.

Table 73: High Level Use Case of Customize Deal Details.

### Notify User

Use Case	Notify user
Actor	Admin, vendor
Description	Admin can send email notification to both vendor and customers. After the purchase and after the usage of a coupon, customers are always about the status of the coupon by vendors.

Table 74: High Level Use Case of Notify User.

### Upload Profile

Use Case	Upload profile
Actor	Vendor
Description	A vendor cannot add deals without uploading their business details in the KarKhana system. So, this is a must for any registered vendor to make their deals happen.

Table 75: High Level Use Case of Upload Profile.

### Add Deals

Use Case	Add deals
Actor	Vendor
Description	After uploading their business details, vendor can add deals in KarKhana which will be displayed to the customers to buy and use.

Table 76: High Level Use Case of Add Deals.

### View Coupons Details

Use Case	View Coupon details
Actor	User, Vendor
Description	After buying a coupon and is generated, both customer and vendor can view the details of the coupon containing all the important details like deal, vendor, and customer details.

Table 77: High Level Use Case of View Coupon Details.

### Verify Active Coupons

Use Case	Verify active coupon
Actor	Vendor
Description	To use the coupon purchased, vendor must verify and confirm it first. When the vendor confirms the coupon then the status of coupon changes from active to used.

Table 78: High Level Use Case of Verify Active Coupons.

### 8.4.3 ERD Iterations

#### 8.4.3.1 Final ERD of the system

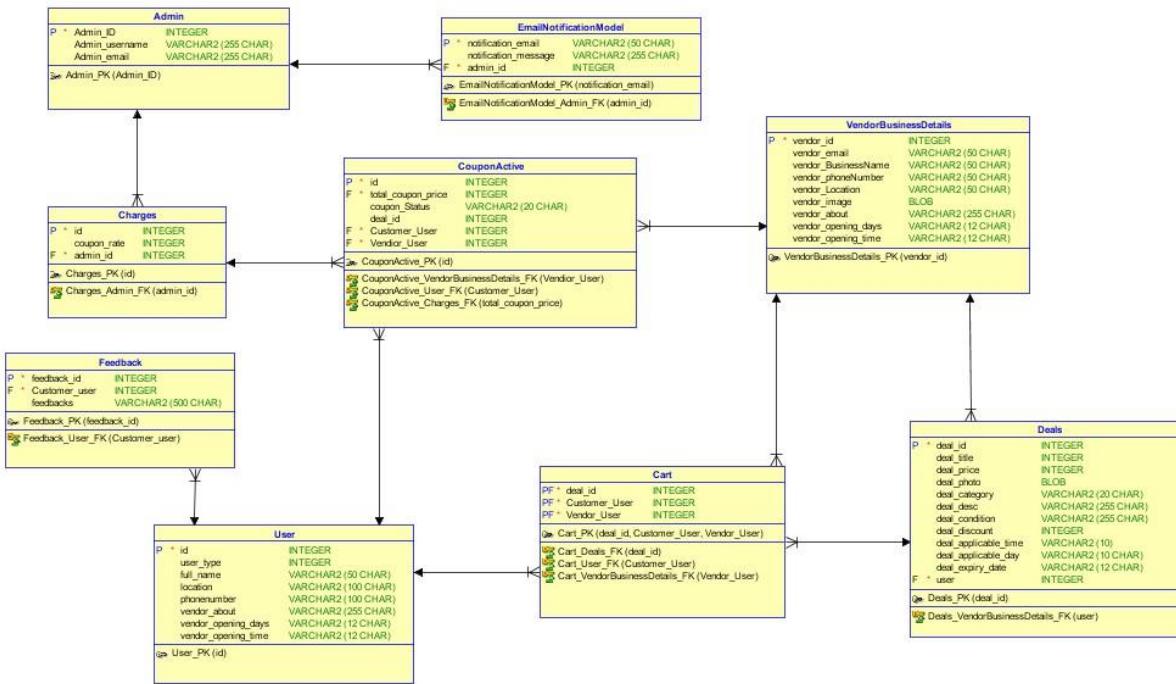


Table 79: Final ERD of the system.

### 8.4.3.2 ERD Iteration

Initially an ERD was created according to how the project's overview entities and attributes would look like.

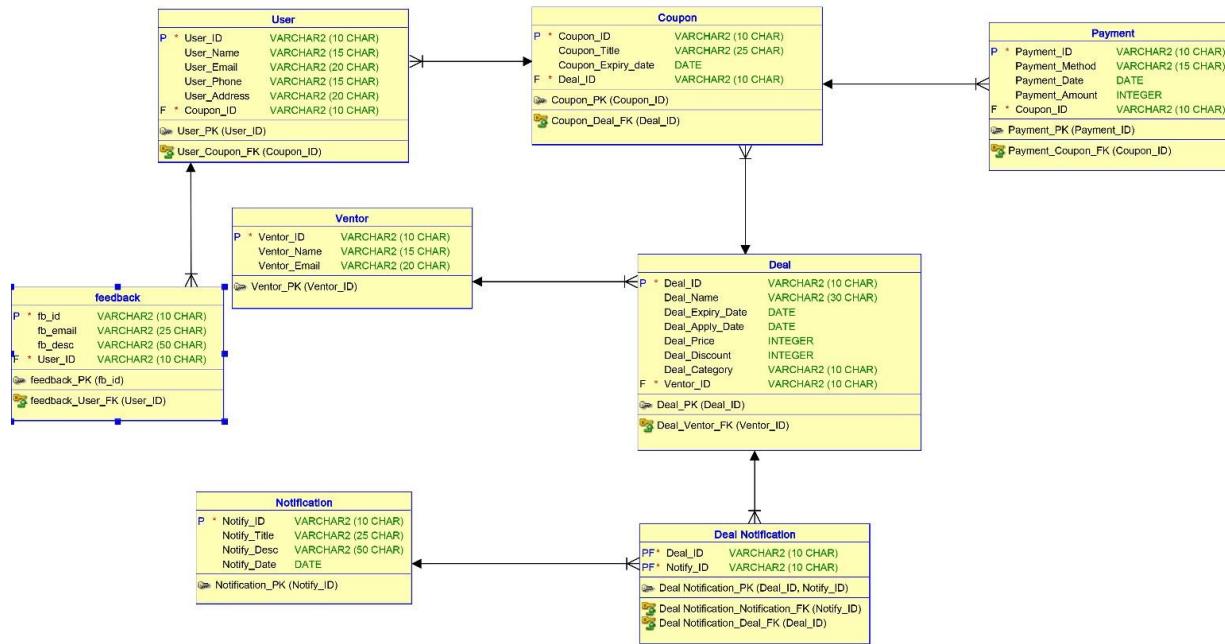


Table 80: First Iteration of ERD of the system.

---

#### 8.4.4 WIREFRAMES

---

##### Set Location Page



Figure 408: Wireframe of Set Location Page

## Search Page

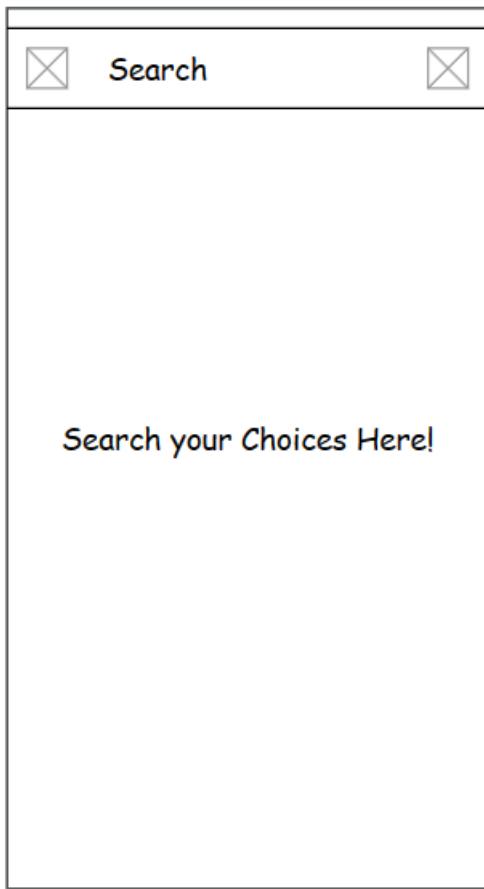


Figure 409: Wireframe of Search Page.

## Deal Details Page

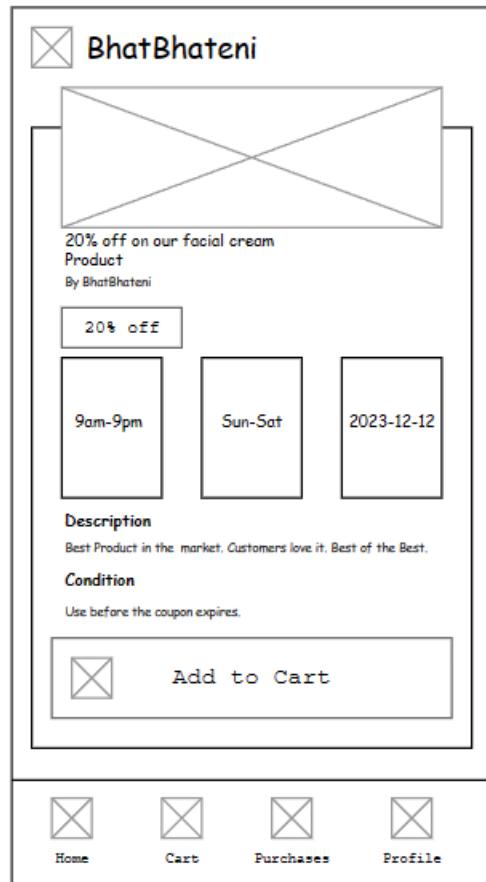


Figure 410: Wireframe of Deal Details Page.

## Cart Page

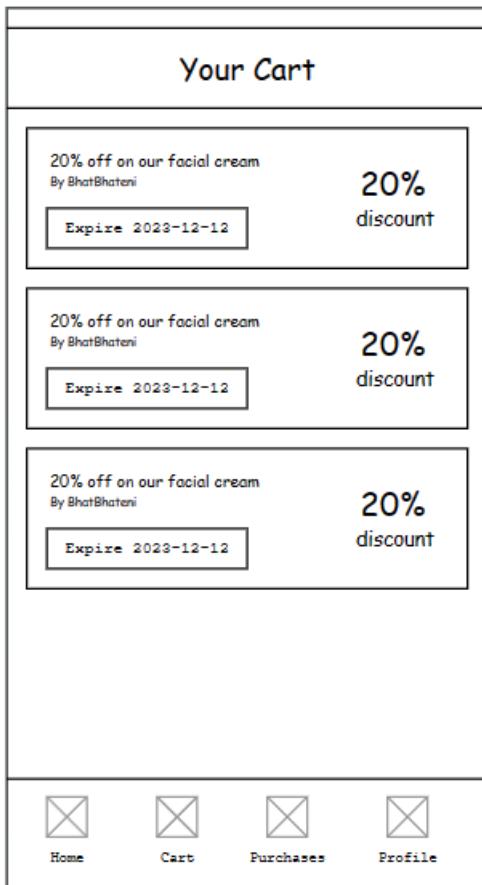


Figure 411: Wireframe of Cart Page.

## Payment Page



Figure 412: Wireframe of Payment Page.

## Coupon Details Page



Figure 413: Wireframe of Coupon Details page.

## Profile Page

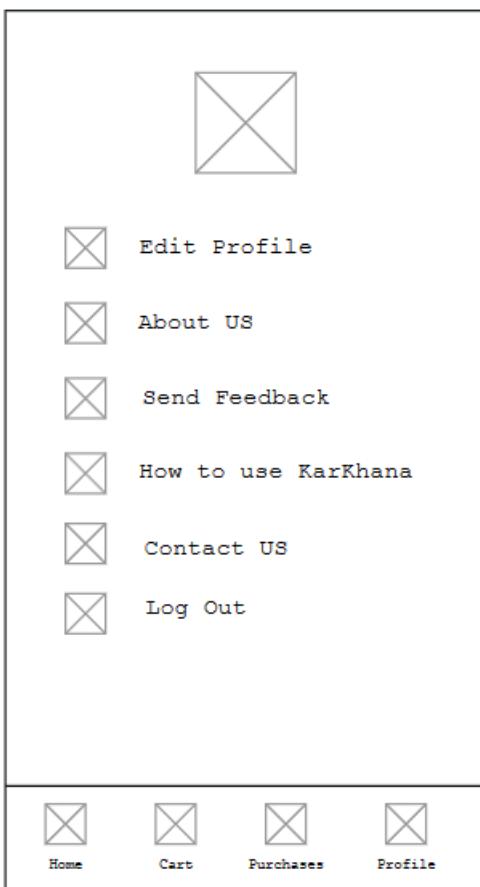
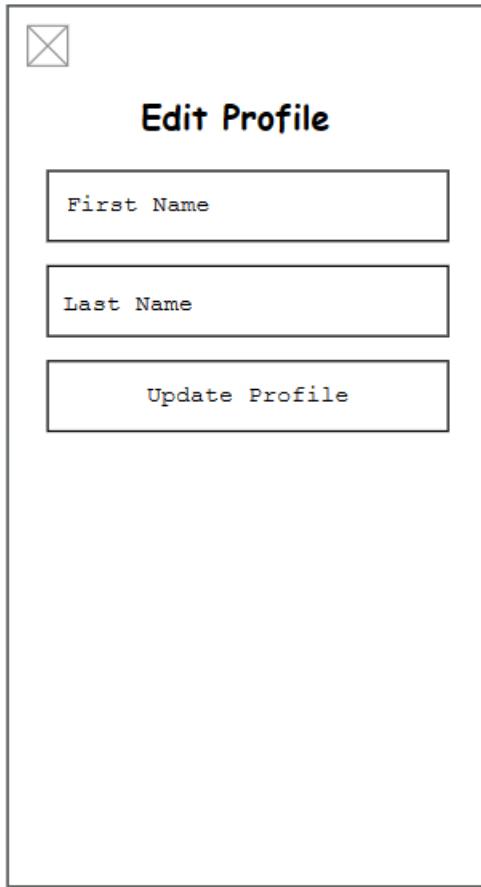


Figure 414: Wireframe of Profile Page.

## Edit Profile Page



A wireframe diagram of an 'Edit Profile' page. The page has a title 'Edit Profile' at the top. Below the title are two input fields: 'First Name' and 'Last Name'. At the bottom of the page is a single button labeled 'Update Profile'.

Figure 415: Wireframe of Edit Profile Page.

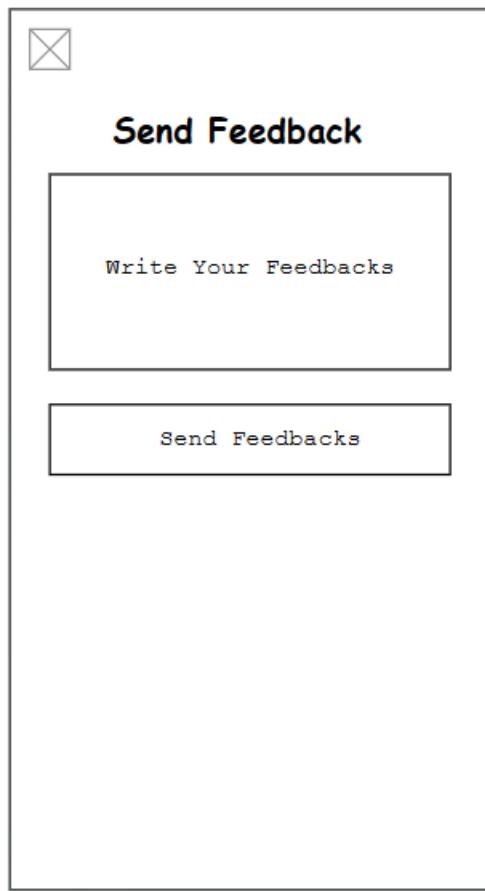
**Send Feedback Page**

Figure 416: Wireframe of Send Feedback Page.

## Share Page

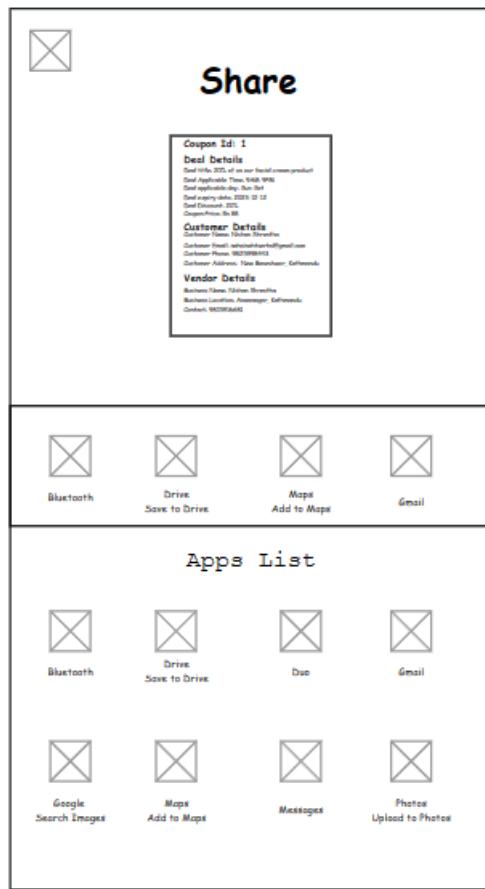
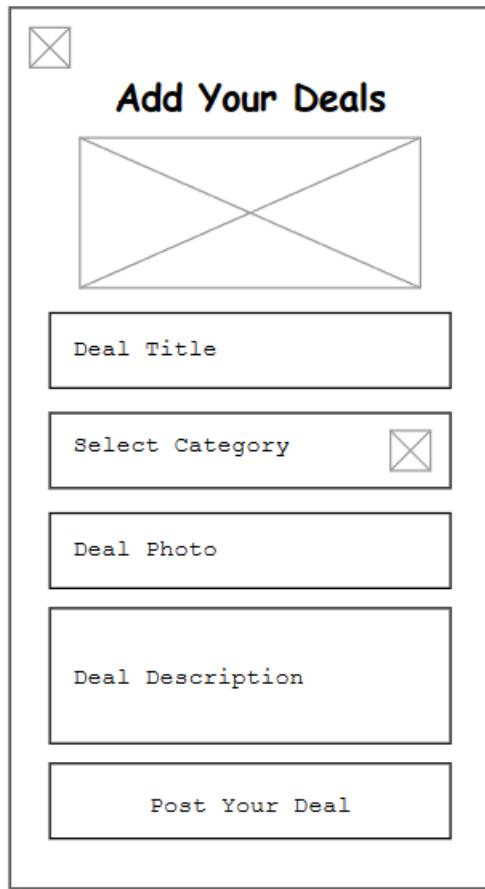


Figure 417: Wireframe of share page.

**Vendor Add Deals Page**

The wireframe shows a vertical form for adding deals. At the top is a header with a close button and the title "Add Your Deals". Below the header is a large rectangular input field with a diagonal cross, likely for deal photos. This is followed by five horizontal input fields: "Deal Title", "Select Category" (with a small close button to its right), "Deal Photo", "Deal Description", and finally a "Post Your Deal" button at the bottom.

Figure 418: Wireframe of Vendor Add deals page.

## Vendor Active Coupons Page



Figure 419: Wireframe of Vendor Active Coupons Page.

## Vendor Active Coupon Details Page

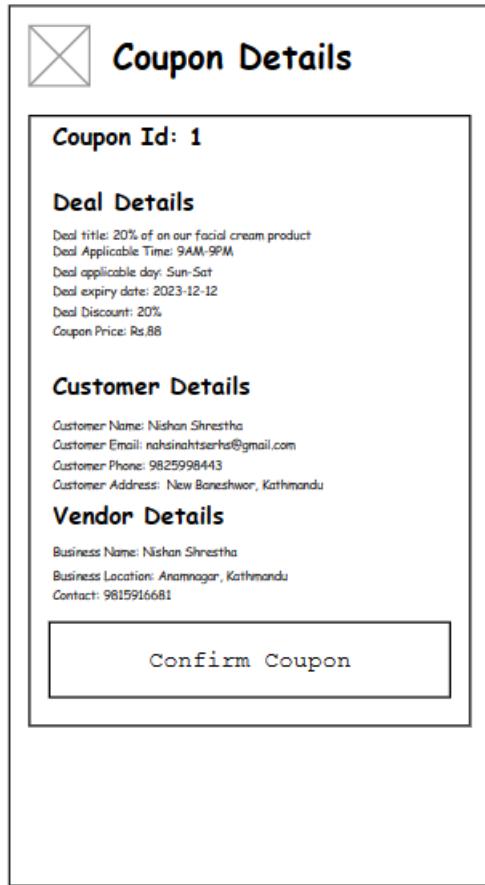


Figure 420: Wireframe of Vendor Active Coupon Details Page.

## Vendor Used Coupons Page

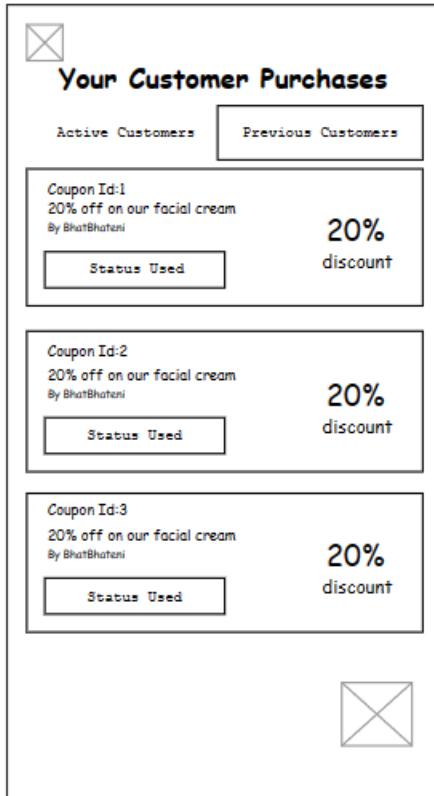


Figure 421: Wireframe of Vendor Used Coupons Page.

---

## **8.5 APPENDIX: 5 SRS (SOFTWARE REQUIREMENTS SPECIFICATION)**

---

### **8.5.1 PROJECT TITLE**

KarKhana ("Empowering Choices, Connecting Vendors")

### **8.5.2 CATEGORY**

Android Application using Flutter and Django.

### **8.5.3 INTRODUCTION**

#### **8.5.3.1 Purpose**

The Software Requirements Specification (SRS) document is made to provide thorough details about what the system can do and what it can't do. This document focuses on what's needed for Android. It also gives a lot of information about what the application must have.

#### **8.5.3.2 Targeted Audiences and Reading Suggestions**

This document is for the people building a system, helping them create an Android app. The app is meant for customers who want to find the best deals and offers nearby and far away. Anyone, including the public and sellers putting up discounts, can use this app. The document is like a guide for the developers, specifically those at KarKhana. It shows them what features are needed in an Android app, and users can refer to this document to understand those features.

#### **8.5.3.3 Project Scope**

The android application is being built for specific deals and offers on services and goods in order to tackle the problem of finding discounts that consumers confront in real life. Some of the features that are required for android application are listed below:

- i. Set your Current Location.
- ii. Near you Recommendation.
- iii. Login/Registration from both users and vendor side.
- iv. Payment from e-wallet.
- v. Generates Coupon.
- vi. Active/Previous coupon data of customers.

- vii. Send Feedback.
- viii. Purchased/ Redeem coupon notifier.
- ix. Business Credentials Verification.
- x. Shareable coupon.

#### **8.5.3.4 Existing System**

In the Present Context, the system is all manual. If people want deals or notifications about them, they have to be physically present in a specific location. It's a hassle because you have to go from one place to another. This manual system has some problems:

- i. You have to be there to know about deals or at least leave your home to check if there are any deals in a particular business.
- ii. Deals you like will only be available after a lot of negotiations or bargaining.
- iii. Some deals might end, and you won't even know about it.
- iv. Sending a service or product with a great discounted deal is not easy. You have to buy it first and then physically send it to the person.
- v. Businesses that aren't well-known won't be able to share their deals and offers, affecting their promotions.

#### **8.5.3.5 Purposed System**

The main goal of this proposed system is to replace the current way of getting deals by creating a user-friendly Android app that anyone can use. Here are some advantages for both customers and businesses with this proposed system:

##### **Benefits for Users:**

- i. Discover amazing deals from local businesses.
- ii. Buy the deals you want using a digital payment method.
- iii. Use the KarKhana app to directly visit businesses and redeem your discounts.

##### **Benefits for Businesses:**

- i. Attract new customers by offering appealing discounts.
- ii. Advertise attractive happy hour discounts to boost business during off-peak hours.
- iii. Increase visibility by registering on the KarKhana platform.
- iv. Create dynamic discount programs with time constraints, like daily, weekly, monthly, or yearly offers.

#### 8.5.4 Overall Description

##### i. System Perspective



Figure 422: System Perspective of Android Application.

#### **8.5.4.1 System Perspective**

**SP1:** Login of Users (Customers) and Vendors

**SP2:** Registration of both type of users

**SP3:** Buy coupon of deals

**SP4:** Online payment

**SP5:** Search for deals and coupons

**SP6:** Set current location

**SP7:** Send Feedback

**SP8:** Redeem coupon

**SP9:** View Coupon Status

**SP10:** Share Coupon

#### **8.5.4.2 System Features**

##### **SF1: Login**

Customer and vendor both type of users can login into this application for their own respective goal of using this application.

##### **SF2: Register**

Both customer and vendor can register to KarKhana using mobile application only as registering of both type of user is made possible in this application.

##### **SF3: Set current location**

can set their location through one click of a button. Their current location is tracked and saved by the system.

**SF4: Location Based Recommendation System**

Customer will be displayed all the deals in their area along with all the other deals posted in the system by the vendors.

**SF5: Add to cart**

Customers can save their favorite deals in their cart, and from where they can buy the coupon for those respective deals.

**SF6: Payment**

Online payment is also available to buy coupon. Khalti's Online payment service is used in this application

**SF7: Coupon Generate**

After buying the deal, a coupon is generated with all the information of deal, customer and vendor along with calculated coupon price. This generated coupon is used by the customer to utilize the deal they bought from the KarKhana application.

**SF8: Confirm Coupon**

The vendor will be required to validate the usage of the coupon when the customer wants to use it.

**SF9: Share coupon**

Customers are able to share their active coupons to their friends and family.

**SF10: Add deals**

Deals can be added by vendors from within the application.

### **8.5.5 USER CLASS AND CHARACTERISTICS**

The END users and their roles while using the KarKhana application are given below:

#### **UC1: Customer user**

**UC1.1:** A user friendly interface is built for users to do any operations.

**UC1.2:** To get access to all the features, register to the system.

**UC1.3:** Buy coupon of the desired deals.

**UC1.4:** Set current location to get new deals recommended.

**UC1.5:** Get information about detail information about deal like applicable time, date, day, condition of deal, discount and business source.

**UC1.6:** Pay online with Khalti.

**UC1.7:** Provide feedback to the system.

**UC1.8:** Search for desired deals from search bar.

**UC1.9:** Search for coupons in purchases section.

**UC1.10:** Edit profile if wrong details were registered.

#### **UC2: Vendor user**

**UC2.1:** Register business details to reach to customers.

**UC2.2:** Upload profile to the system after registration.

**UC2.3:** Add new deals and offers in the application.

**UC2.4:** Redeem coupons of their respective business.

**UC2.5:** View their respective business coupon's status.

**UC3: Admin**

**UC3.1:** Provides update to the system.

**UC3.2:** Handles details of deals and coupons.

**UC3.3:** Register vendors after authenticating their business details.

**UC3.4:** Send notification to both type of users i.e. Customer and Vendor through email.

**UC3.5:** Handles all the users.

**UC3.6:** Sets coupon rate.

**8.5.6 ASSUMPTION AND DEPENDENCIES**

**AS1:** The public i.e., customers/users who seek for discounted deals and services are the main target audience of this program

**AS2:** The mobile application ‘KarKhana’ will be developed and handed to users as online application which requires internet to run.

## 8.5.7 FUNCTIONAL REQUIREMENTS

### 8.5.7.1 REGISTRATION

Req. ID	Requirement Description	
FR.01	Sys. Req.	System Requirement
	ID	
	SR.01	Users can view their respective register form.
	SR.02	The system will determine whether or not the required section was entered.
	SR.03	Authentication of the entered details of the user will be verified by the system
	SR.04	For customer, a token is sent by the system to the mail provided by the customer while registering to verify the email. For vendor, entered details are sent to admin for further process
	SR.05	the system will display a message depending on whether the user registration was successful or not,

Table 81: Functional Requirements of Register.

### 8.5.7.2 Login

Req. ID	Requirement Description	
	Sys. Req.	System Requirement
	ID	
FR.02	Both types of users can login into this KarKhana application	
	<b>SR.01</b>	Users can view their respective login form.
	<b>SR.02</b>	The system will validate whether or not the required section was entered.
	<b>SR.03</b>	System must check if the provided details are valid or not.
	<b>SR.04</b>	the system will display a message depending on whether the user registration was successful or not.
	<b>SR.05</b>	After login success customer should be navigated to set location page and vendor should navigate straight to homepage.

Table 82: Functional Requirements of Login.

### 8.5.7.3 Add Deal to Cart

Req. ID	Requirement Description	
FR.03	Sys.	System Requirement
	Req. ID	
	<b>SR.01</b>	Customer can browse and select desired deal.
	<b>SR.02</b>	After customer confirms to add the deal to cart, System should display a success message
	<b>SR.03</b>	System must display the added deal into the respective customer's cart.

Table 83: Functional Requirements of Add deals to cart

#### 8.5.7.4 Buy Coupons

Req. ID	Requirement Description	
	Sys. Req. ID	System Requirement
FR.04	Customer can buy the coupon of the deal added to their cart.	
	SR.01	Customer can view the deals added to cart in their cart page.
	SR.02	When the customer selects the deal from cart and proceeds to buy then system generates and displays the coupon details.
	SR.03	System calculates the coupon rate and also displays that in the coupon details
	SR.04	If customer clicks on buy option, then Khalti's interface is displayed by the system.
	SR.05	After successful Khalti procedure, system displays the success message that the coupon purchase is successful.
	SR.06	Along with success message, system also sends email notification about the successful purchase of the coupon to the customer.

Table 84: Functional Requirements of Buy Coupons.

### 8.5.7.5 User Feedback

Req. ID	Requirement Description	
FR.05	Customer can send feedback to the system	
Sys. Req. ID	System Requirement	
<b>SR.01</b>	Customer can view the feedback page.	
<b>SR.02</b>	System checks the field authentication if customer is trying to send empty message or not.	
<b>SR.03</b>	When customer clicks the send button, system gets the email, customer id and customer name of the logged in customer.	
<b>SR.04</b>	System stores the feedback along with customer details in the database for admin to review.	

Table 85: Functional Requirements of User Feedback.

### 8.5.7.6 Payment

Req. ID	Requirement Description	
FR.06	Customer can pay through e-payment i.e., Khalti. <b>Sys. Req. ID</b>	
<b>SR.01</b>	Customer can pay via Khalti.	
<b>SR.02</b>	System displays the Khalti interface to customer and proceed for further payment procedure.	
<b>SR.03</b>	After Khalti procedure if it's a success, then success message must be displayed	
<b>SR.04</b>	In case of not successful payment, system displays the failed message.	

Table 86: Functional Requirements of Payment.

### 8.5.7.7 Search

Req. ID	Requirement Description	
	Sys. Req.	System Requirement
ID		
<b>FR.07</b>		Customer can search for their desired deals and their purchased coupon. Vendor can also search for their coupon purchased by the customers.
	<b>SR.01</b> <b>SR.02</b> <b>SR.03</b> <b>SR.04</b> <b>SR.05</b>	Customer can search for their desired deals by the deal title and also can search for their purchased coupon by the coupon id in purchases page. Vendor can also search for the coupon of their business bought by the customers. The system will check if the provided deal title or coupon Id is available or not. In case of searched data not available, "no data found" message should be displayed by the system. If the deal with the provided title or coupon with the provided id is available then the system should display all the matching deals or coupons to the customer

Table 87: Functional Requirements of Search.

### 8.5.7.8 Coupon Status

Req. ID	Requirement Description	
FR.08	Sys. Req.	System Requirement
	ID	
	<b>SR.01</b>	After purchasing a coupon, system must display it as Active coupon in active section of the purchases page.
	<b>SR.02</b>	Along with that, system also updates respective vendor side and displays the purchased coupon of their respective deals.
	<b>SR.03</b>	When the customer wants to redeem the coupon, vendor confirms the coupon then again system updates the status of the coupon in both side of the user.
	<b>SR.04</b>	After redemption success, system sends email notification to user about the redemption.
	<b>SR.05</b>	Now the system displays the redeemed coupon in previous coupon section of purchases page of both user side.

Table 88: Functional Requirements of Coupon Status.

### 8.5.7.9 Add Deals

Req. ID	Requirement Description	
FR.09	Sys. Req. ID	System Requirement
	<b>SR.01</b>	Vendor should enter deals detail in add deal page.
	<b>SR.02</b>	System validates the fields whether valid details are entered or not.
	<b>SR.03</b>	System checks if the vendor has uploaded their business profile or not.
	<b>SR.04</b>	If profile is uploaded then system should post the deal to the application which is then displayed to customers.
	<b>SR.05</b>	If profile is not uploaded then the system should ask the vendor to upload the profile first.

Table 89: Functional Requirements of Add Deals.

### 8.5.8 Non-Functional Requirements

#### 8.5.8.1 Performance Requirements

Req. ID	Requirement Description	Priority
PR.01	It should take at most of 5 to 10 seconds to launch the application.	Could
PR.02	Before everything, if user is log in then homepage should be displayed else welcome screen should be display which ask if user want to login as customer or vendor.	Should
PR.03	This application should display welcome page, if the user is not logged in.	Should
PR.04	This application should display the location page for customer and homepage for vendor at fist after successfully log in.	Should
PR.05	All the features will be functional on the android application for all the users	Should

Table 90: Performance Requirements.

### 8.5.8.2 Safety Requirements

Req. ID	Requirement Description	Priority
<b>SR.01</b>	Permission is asked by the application when it needs to track the user's current location.	Should
<b>SR.02</b>	There are no advertisements of anything or anyone displayed in this application	Should
<b>SR.03</b>	Non-requirement software should not be installed without the access of information about the user.	Should
<b>SR.04</b>	Application shall be made free from any attacks	Could
<b>SR.05</b>	Once used coupon cannot be re-used, application should recognize and update the coupon status after every redemption.	Should

Table 91: Safety Requirements.

## **8.6 APPENDIX 6: USER MANUAL**

---

A user manual is a book that outlines how to use, maintain, and operate a certain product or service. It acts as a user's reference manual and is frequently provided together with the item or service. Step-by-step instructions, illustrations, and explanations are typically included in the handbook to assist consumers utilize the product or service properly and efficiently. A well-written user manual may improve consumers' entire experience with the product or service by assisting them in avoiding misunderstanding, blunders, and aggravation.

Below are given manual for both type of users i.e., Customer and Vendor.

### 8.6.1 Customer

Step 1: Register into KarKhana.

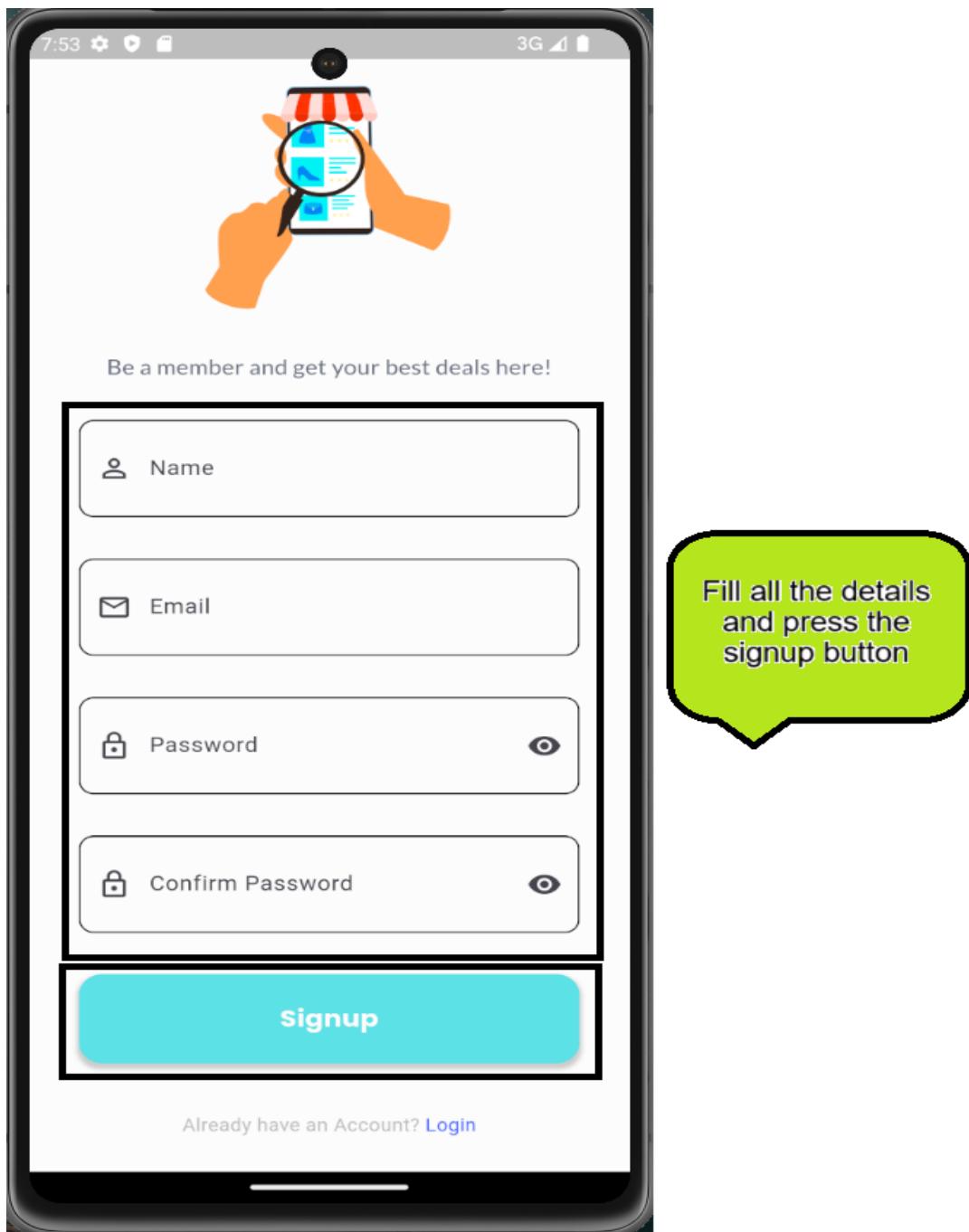


Figure 423: Customer manual Step-1.

Step 2: Login to the system.

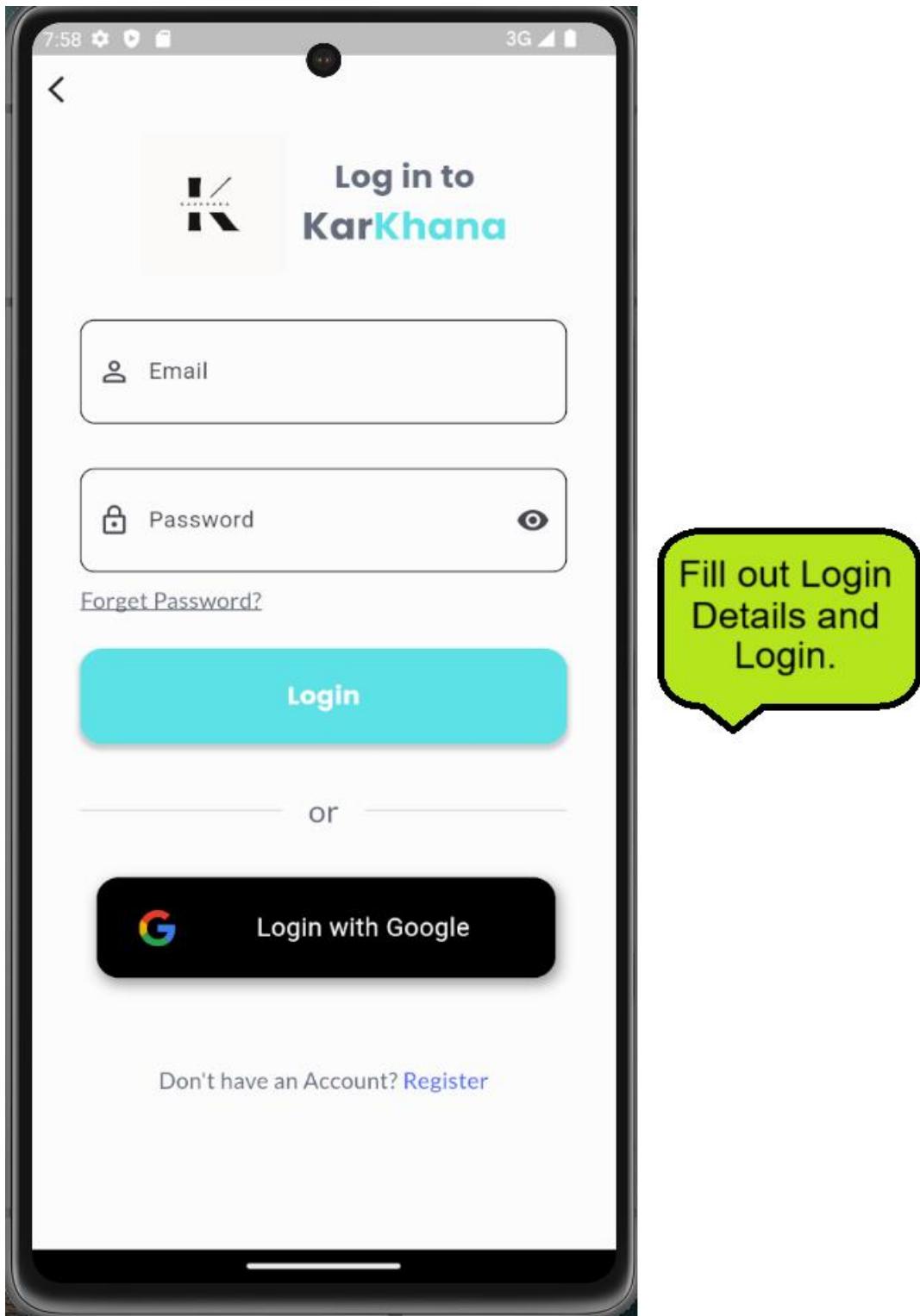


Figure 424: Customer manual Step-2.

## Step 3: Set your Current Location

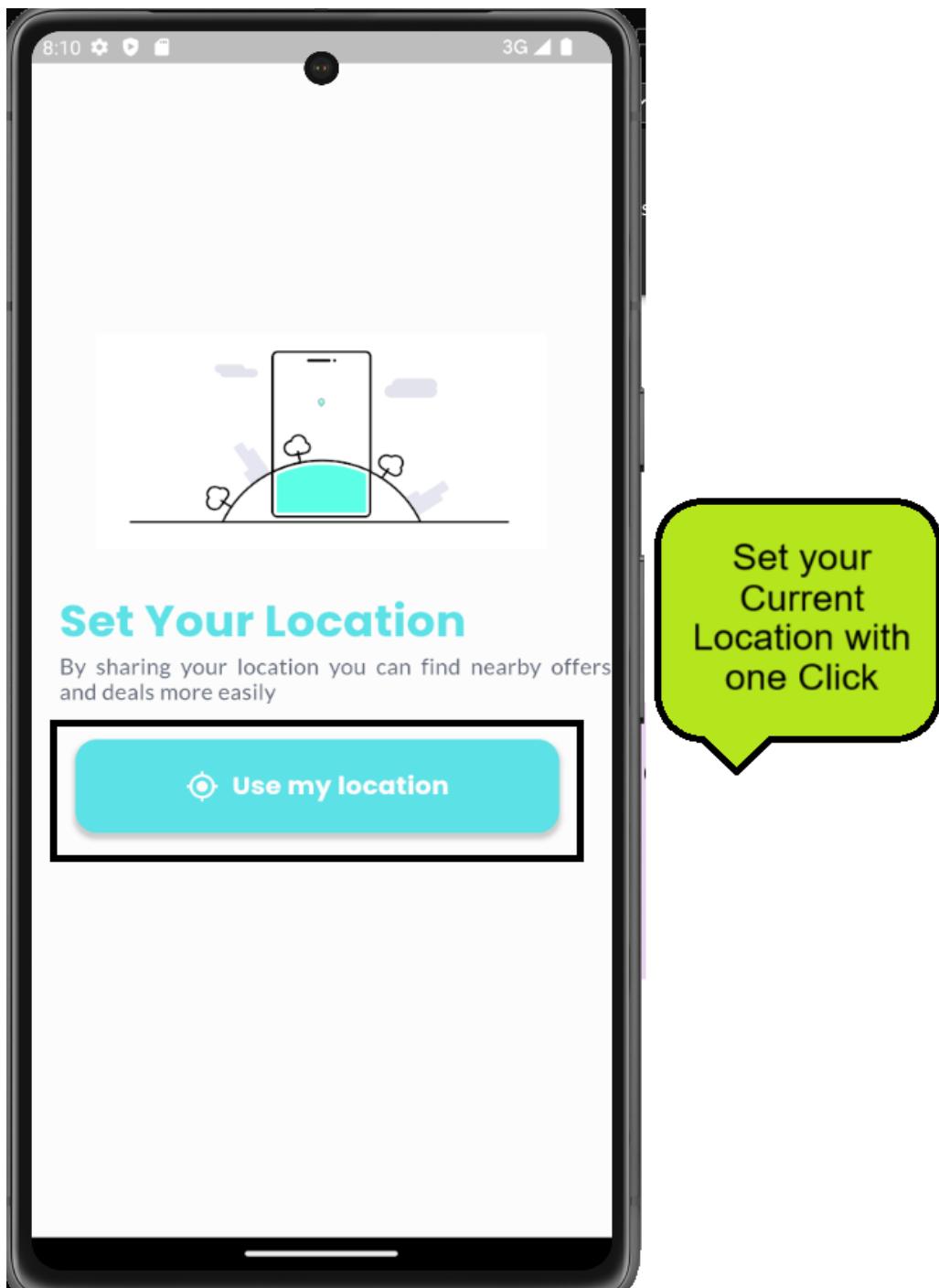


Figure 425: Customer manual Step-3.

## Step 4: Browse or Search for Deals

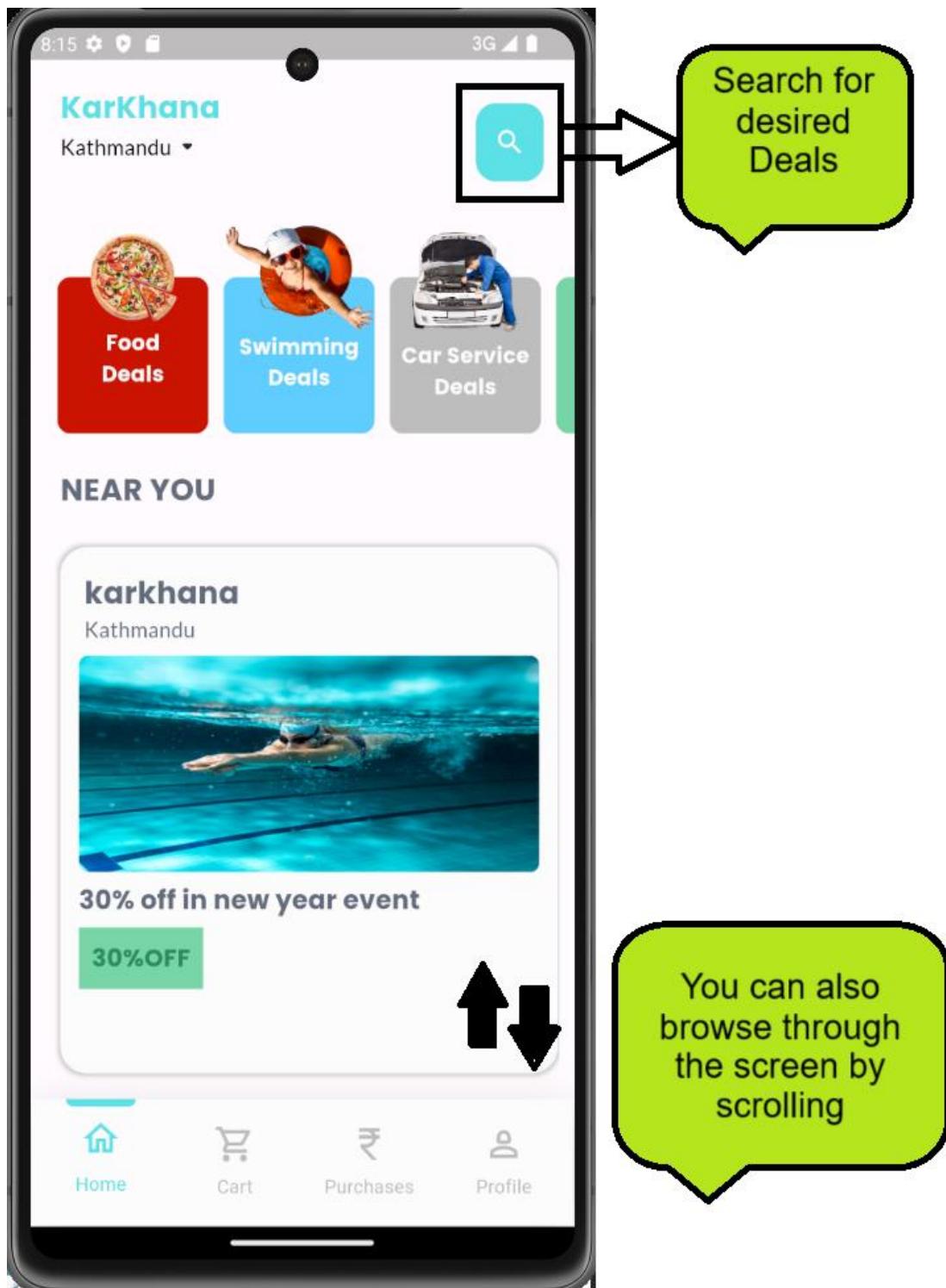


Figure 426: Customer manual Step-4.

## Step 5: Add Deal to Cart

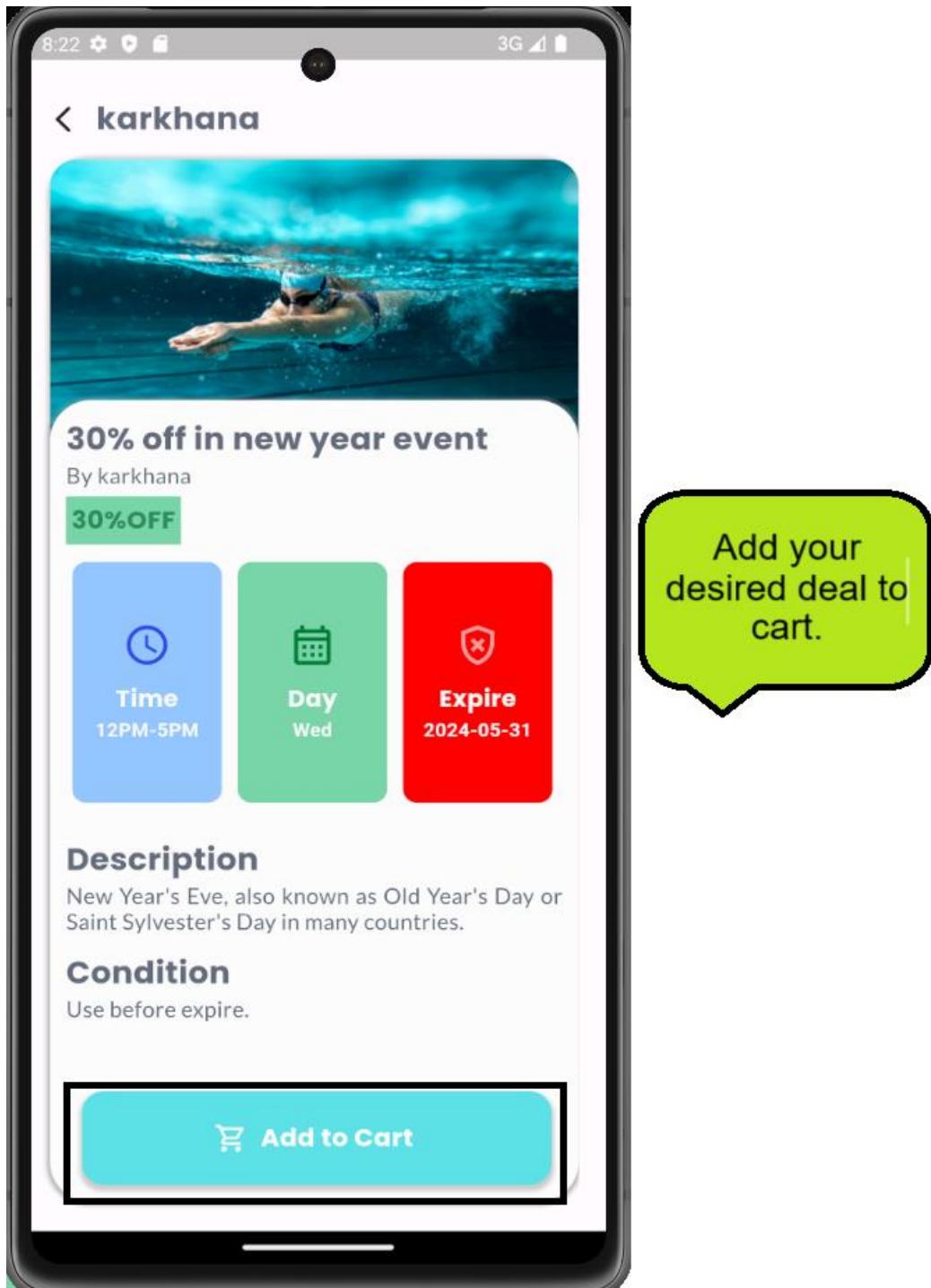


Figure 427: Customer manual Step-5.

Step 6: View your Cart.

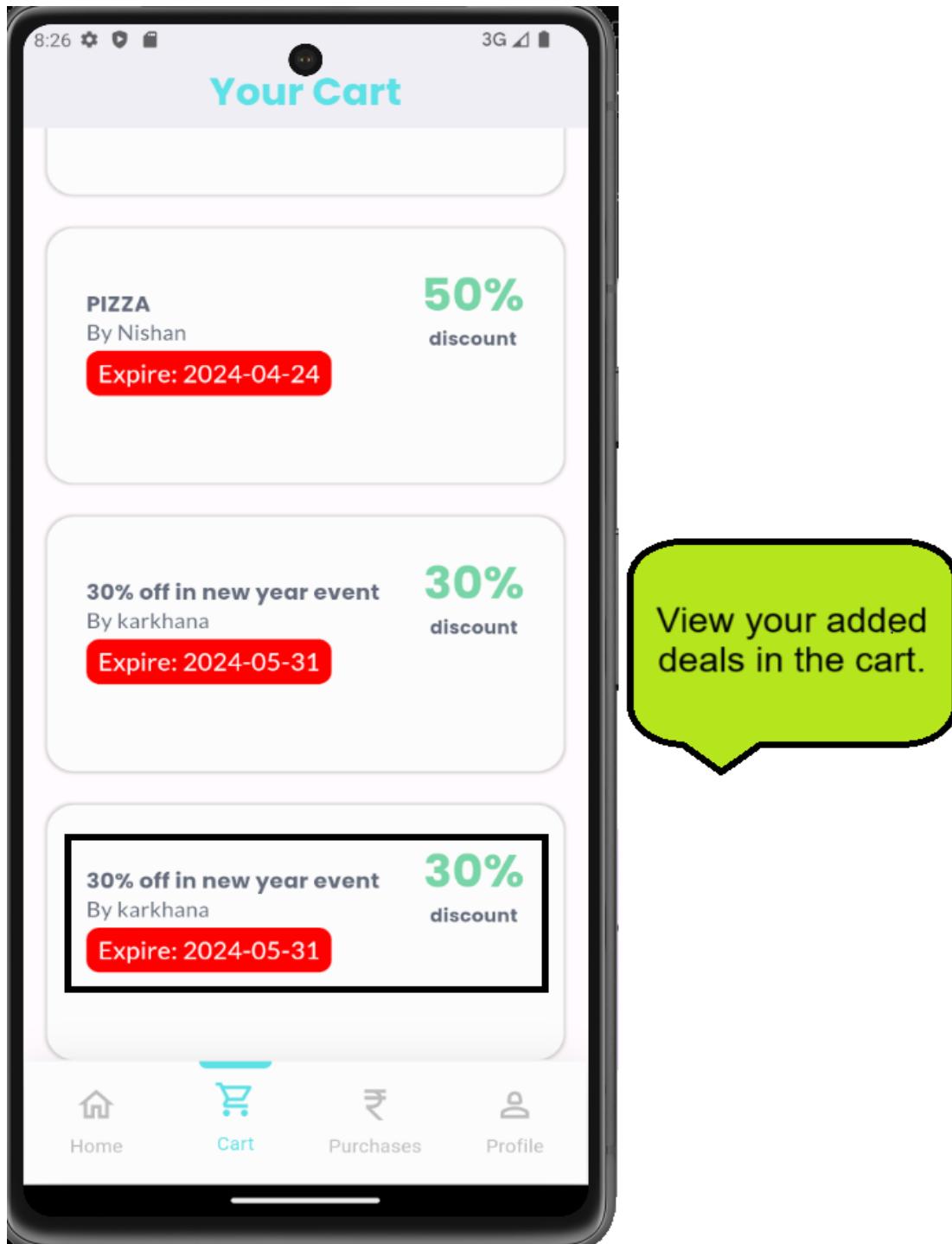


Figure 428: Customer manual Step-6.

Step 7: Buy Coupon from cart.

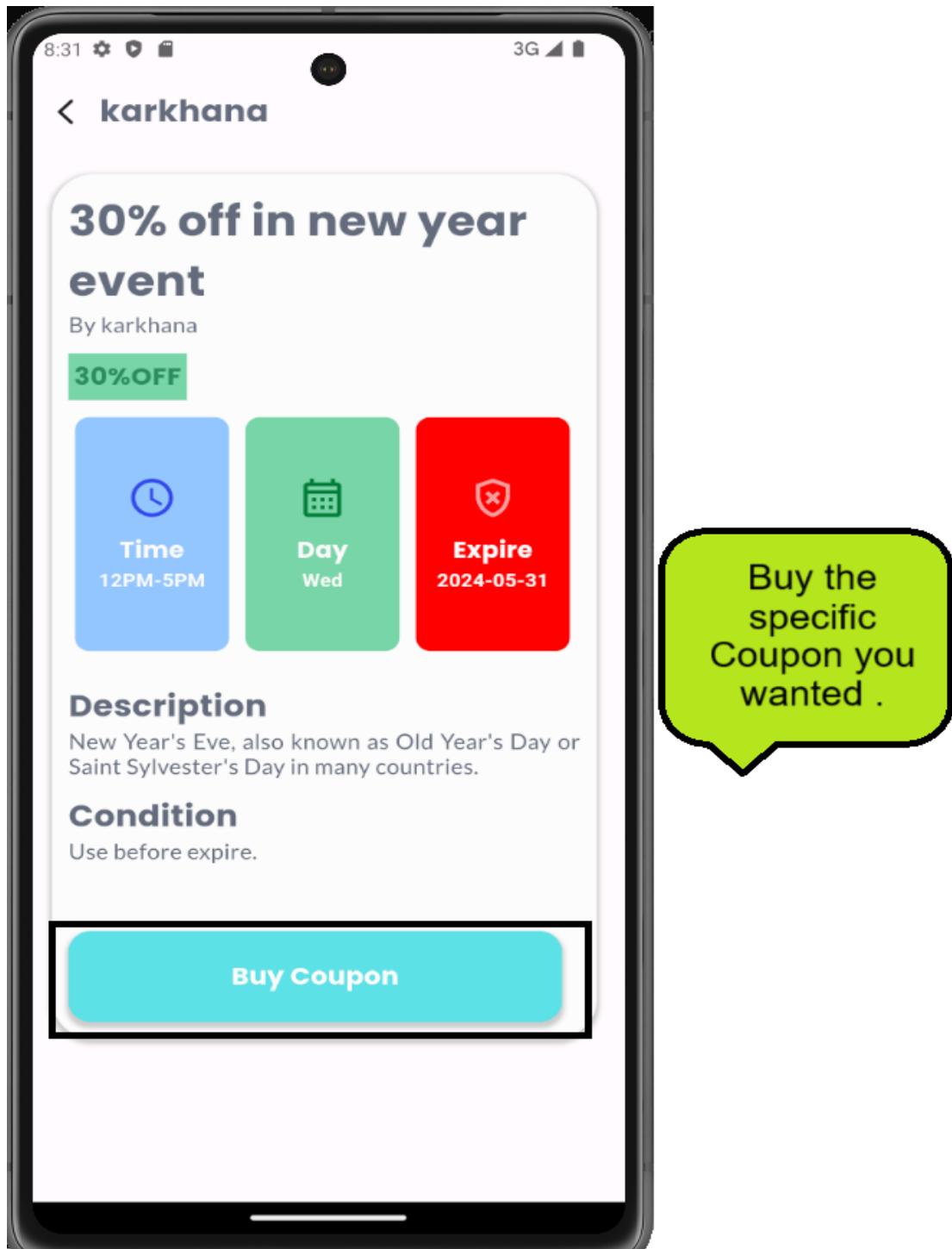


Figure 429: Customer manual Step-7.

## Step 8: View your Active Coupons

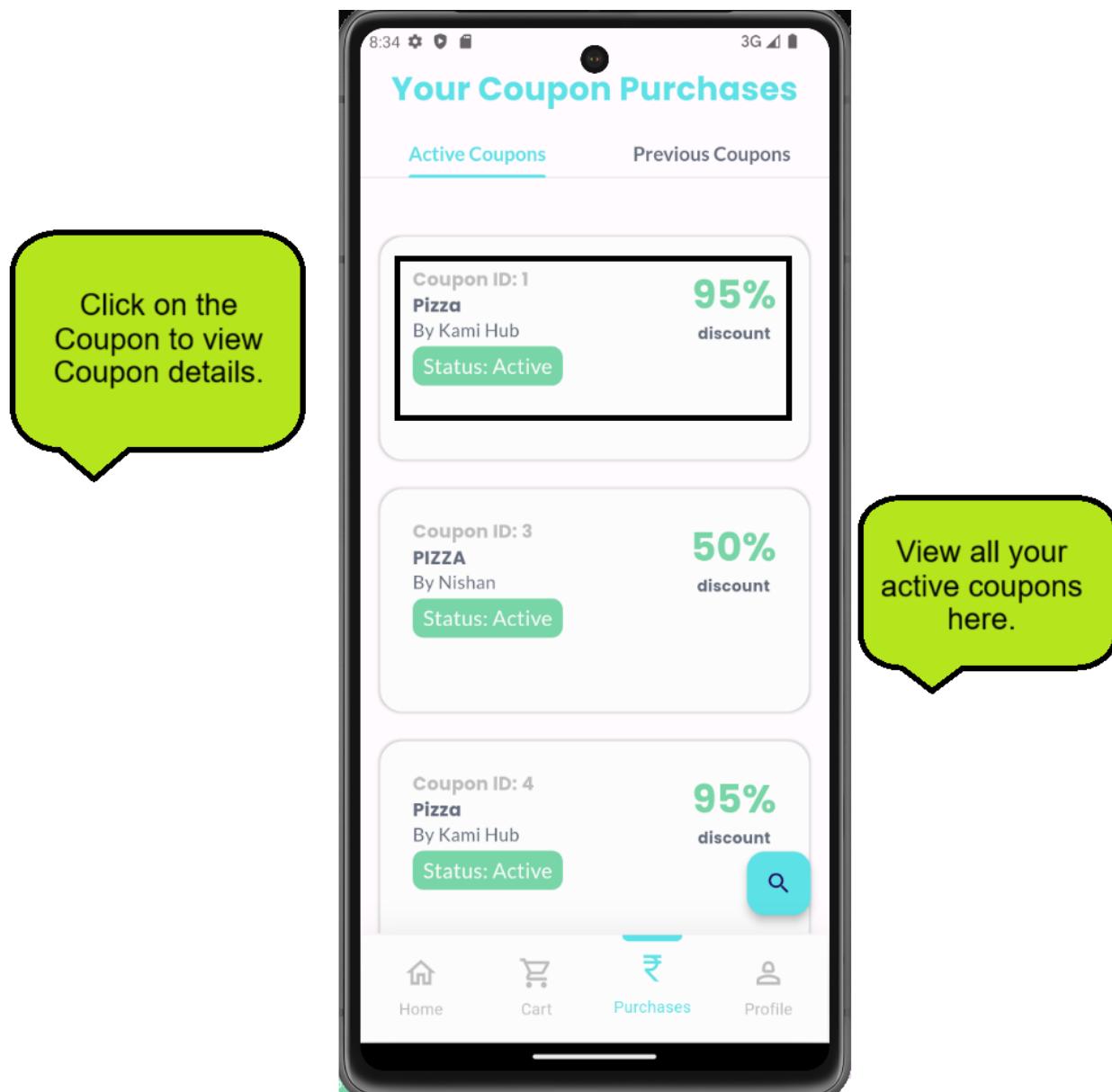


Figure 430: Customer manual Step-8.

Step 9: View your Used Coupon.

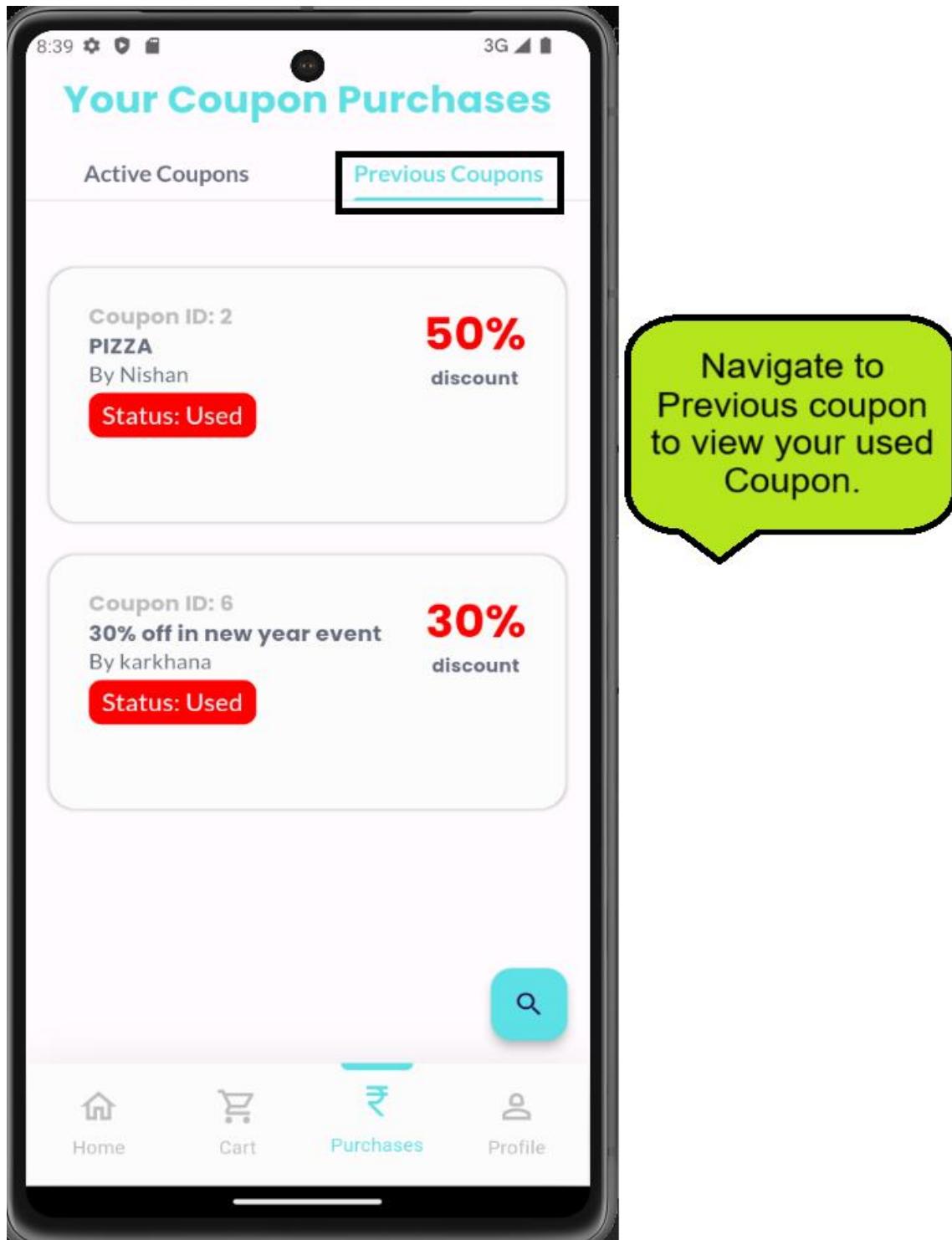
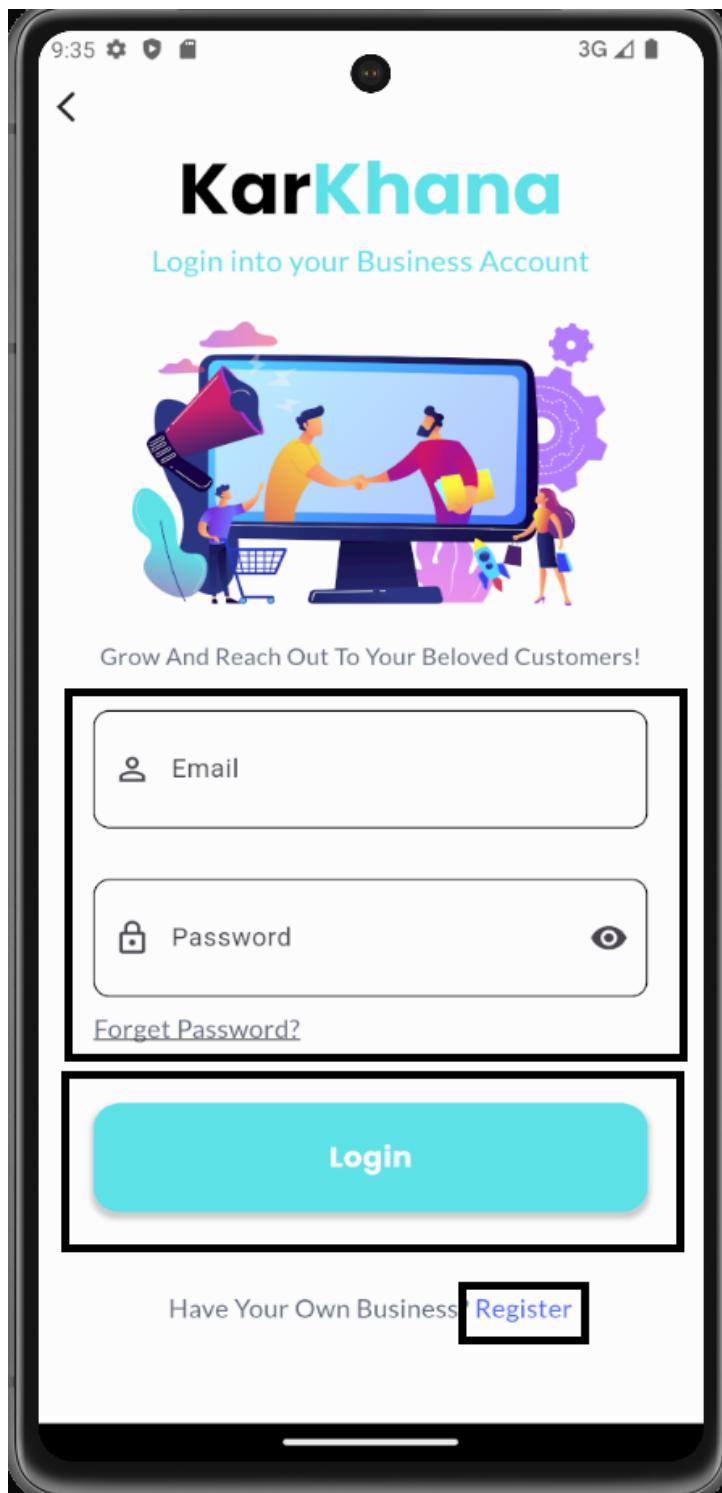


Figure 431: Customer manual Step-9.

### 8.6.2 For Vendor

Step 1: Login as Vendor.



Fill up the  
Details and  
login.

If not  
registered, go  
to register.

Figure 432: Vendor manual Step-1.

Step 2: Upload business profile and add your deals.

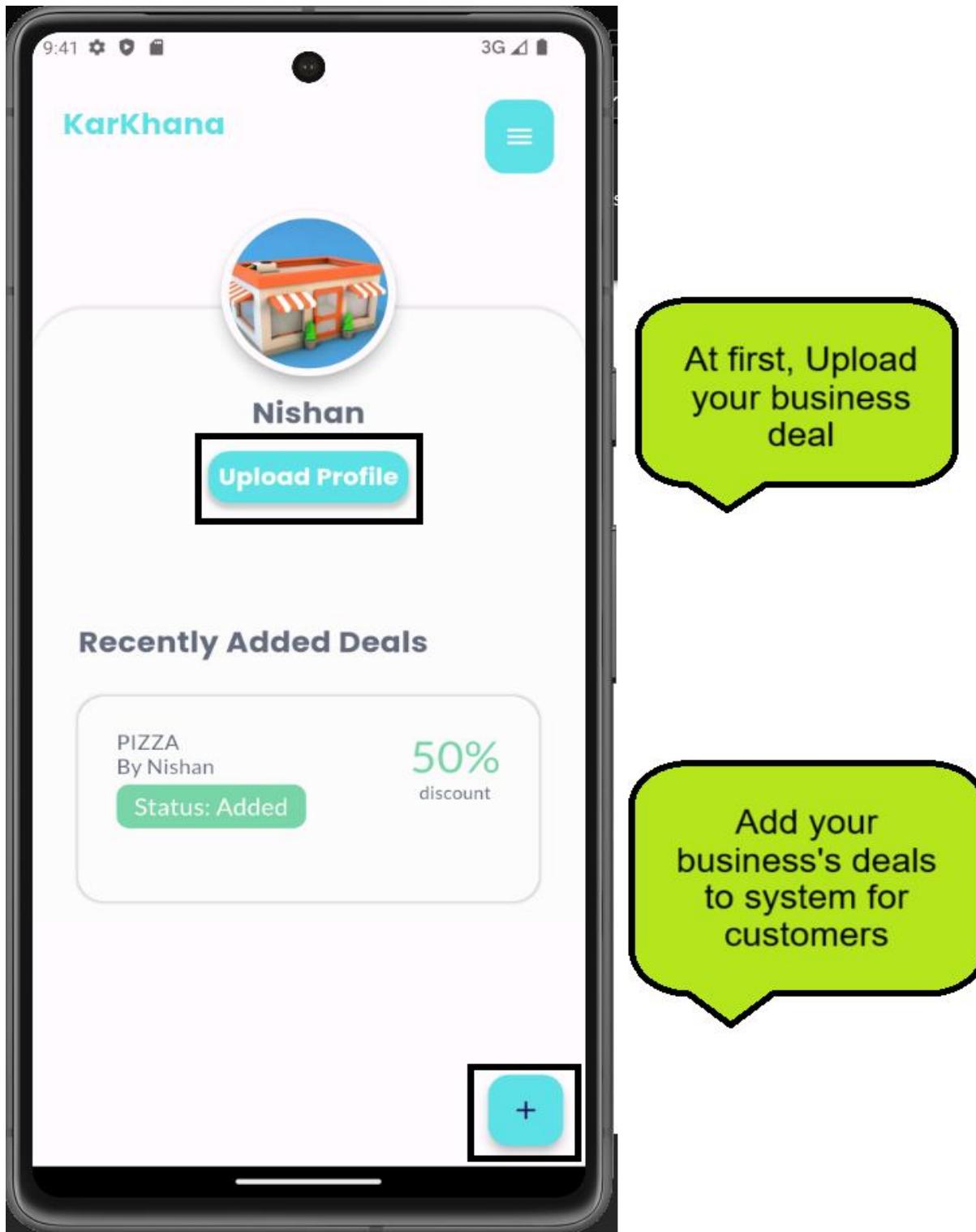


Figure 433: Vendor manual Step-2.

Step 3: Confirm Coupon by the Vendor.

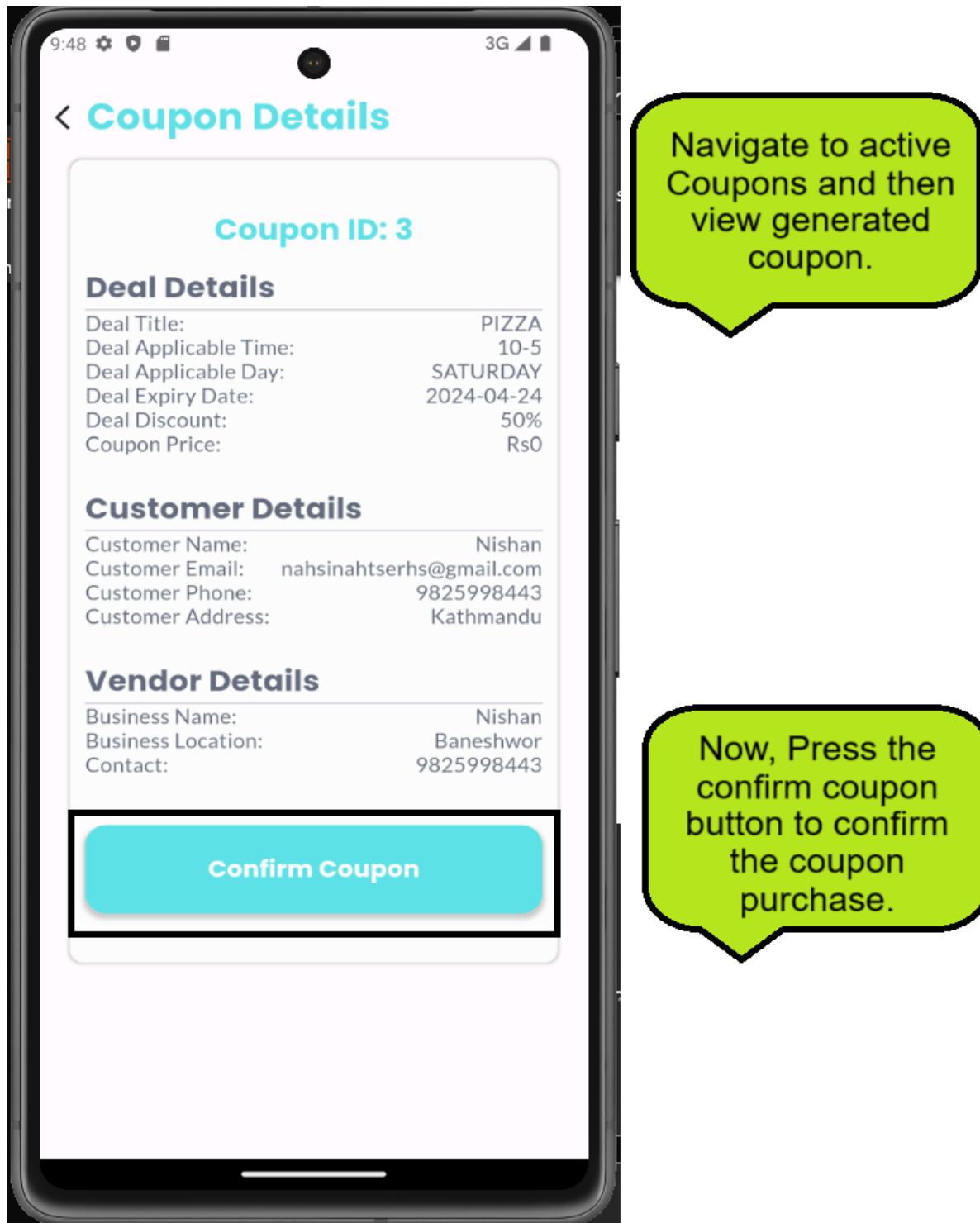


Figure 434: Vendor manual Step-3.

## 8.7 APPENDIX 7: FUTURE WORK

### 8.6.1 READING FOR FUTURE WORK

#### Rating

#### How to implement a Rating Bar in Flutter

In this article I will show you how to implement a rating bar in your Flutter application. A rating bar is generally composed of 5 stars and allow your user to rate a product, an article or anything else. You can also use a rating bar to show informations to your application user like fat, calories, prices of a receipt. What ever is your application subject you always need a rating bar somewhere in your Flutter application.

#### Install the dependency

First of all, create a small basic project to test rating bar dependency. I will show you step by step how to do this from a scratch project

Create your project test project in Android Studio. I named my project 'rating\_bar'. You can choose any platform you need because rating\_bar library support Android, iOS, Linux, macOS, web and Windows platform

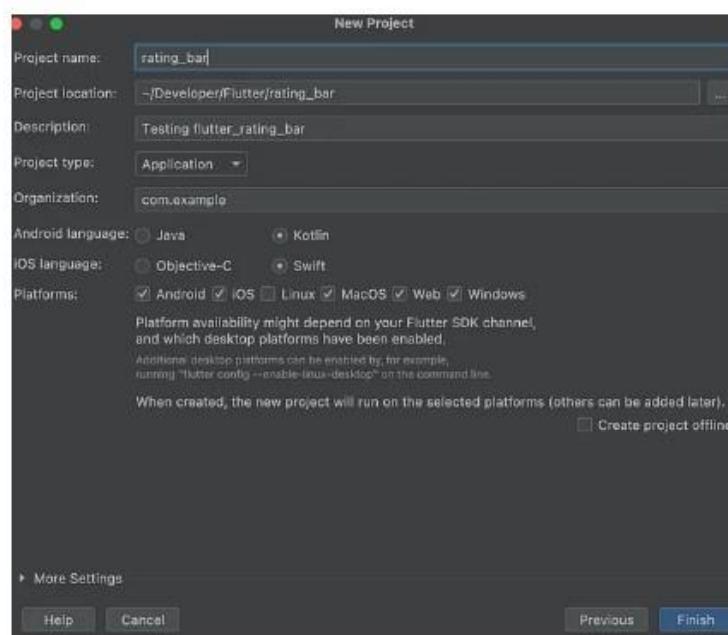


Figure 435: Rating bar research.

# flutter\_rating\_bar 4.0.1

Published 10 months ago • [sarbagyastha.com.np](#) Null safety

SDK FLUTTER PLATFORM ANDROID IOS LINUX MACOS WEB WINDOWS

1.7K

[Readme](#) [Changelog](#) [Example](#) [Installing](#) [Versions](#) [Scores](#)

## Use this package as a library

Depend on it

Run this command:

With Flutter:

```
$ flutter pub add flutter_rating_bar
```

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

```
dependencies:  
  flutter_rating_bar: ^4.0.1
```

Alternatively, your editor might support flutter pub get. Check the docs for your editor to learn more.

Figure 436: Rating Bar Research (II).

## QR generator

**Step 1: First add the following dependency in your pubspec.yaml file**

```
Dart
dependencies:
  path_provider: ^1.6.24
  qr_flutter: ^3.2.0
  barcode_scan_fix: ^1.0.2
```

Now click on **pub.get** to configure.

AD

**Step 2: Now navigate to main.dart() file and return MaterialApp()**  
 First, we have declared **MyApp()** in *runApp* in the main function. Then we have created **StatelessWidget** for **MyApp** in which we have returned **MaterialApp()**. In this **MaterialApp()** we have given the title of our App then declared the theme of our App as primarySwatch as indigo. Then we have given our first screen of our slider app in the home: **HomePage()**

Figure 437: QR generator (I)

```
Dart
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      //Given Title
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      //Given Theme Color
      theme: ThemeData(
        primarySwatch: Colors.indigo,
      ),
      //Declared first page of our app
      home: HomePage(),
    );
  }
}
```

Figure 438: QR generator (II)

**Step 3: Declare StatefulWidget() for HomePage**

In **StatefulWidget()** return **Scaffold()** widget. In **Scaffold()** Widget in the body section, we have declared **Container()** having width and height. In that **Container()** we have given **Column** inside which we have given **mainAxisAlignment** as center and **CrossAxisAlignment** as a stretch. After that we have given an **Image** as **Network Image** below that **Image** we have given rounded **FlatButton()** having color, border-radius, and border side. On that **FlatButton** we have given **onPressed** method which is used to navigate to the next screen **ScanQR()**.

Below that we have given another rounded **FlatButton()** having color, border-radius, and border side. On that **FlatButton** we have given **onPressed** method which is used to navigate to the next screen **GenerateQR()**.

```

Dart

class HomePage extends StatefulWidget {
  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(

```

Figure 439: QR generator (III)

**Step 4: Navigating to ScanQR() page.**

In this file first, we have imported package **barcode\_scan\_fix**. Then we have created **StatefulWidget** for **ScanQR** class. In that state we have declared String as **qrCodeResult = 'Not yet Scanned'**. After that in **Scaffold()** we have declared **Appbar** with the title. In the body section, we have declared **Column()** wrapped with **Container()** and given **mainAxisAlignment** as center and **CrossAxisAlignment** as a stretch. After that, we have declared two text widgets first text widget has declared for the given title. And the second text widget is used to declare the String which we have given. After that, we have given a **FlatButton()** and on its **onPressed** method, we have given **codeScanner** which is used to scan QR code which we have imported from the **bar\_code\_scan\_fix** library.

```

Dart

import 'package:barcode_scan_fix/barcode_scan.dart';
import 'package:flutter/material.dart';

class ScanQR extends StatefulWidget {
  @override
  _ScanQRState createState() => _ScanQRState();
}

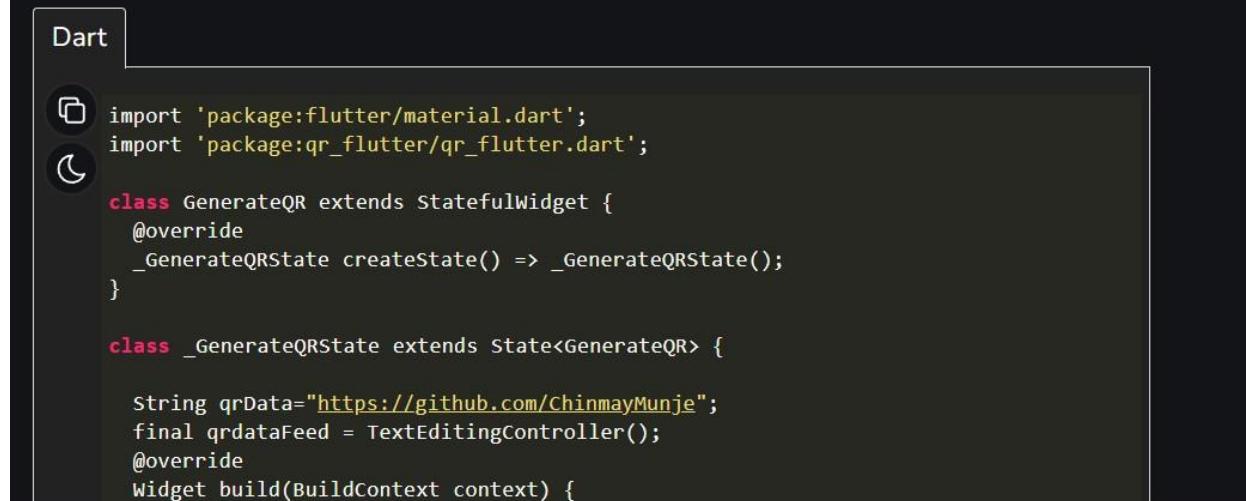
class _ScanQRState extends State<ScanQR> {
  String qrCodeResult = "Not Yet Scanned";
}

```

Figure 440: QR generator (IV)

**Step 5: Navigating to GenerateQr() page.**

First we have given **StatefulWidget()** for **GenerateQr()** class. In that, we have given the final variable as **TextEditingController()**. In the **Scaffold()** section we have given *appbar* along with the title. In the **body** section, we have given **Container()** having **Column()** with *mainAxisAlignment* as center and *CrossAxisAlignment* as a stretch. Wrapped with **SingleChildScrollView()**. After that, we have given **TextField()** for giving input link to convert into a barcode. After that, we have given **FlatButton()** which will convert our link into a QR code. In the *onPressed* of this **FlatButton()** we have declared its logic in which the imported library *qr\_link* is used to convert the link into QR code.



The screenshot shows a Dart code editor window. The code is as follows:

```
import 'package:flutter/material.dart';
import 'package:qr_flutter/qr_flutter.dart';

class GenerateQR extends StatefulWidget {
  @override
  _GenerateQRState createState() => _GenerateQRState();
}

class _GenerateQRState extends State<GenerateQR> {

  String qrData = "https://github.com/ChinmayMunje";
  final qrdataFeed = TextEditingController();
  @override
  Widget build(BuildContext context) {
```

Figure 441: QR generator (V)

## Chatbot

Follow these steps to create a Flutter Chatbot:

### Step 1: Setup an account in Kommunicate

Login to your Kommunicate dashboard and navigate to the Bot Integration section.

If you do not have an account, you can create one here for free.

Locate the Kompose section and click on Integrate Bot.

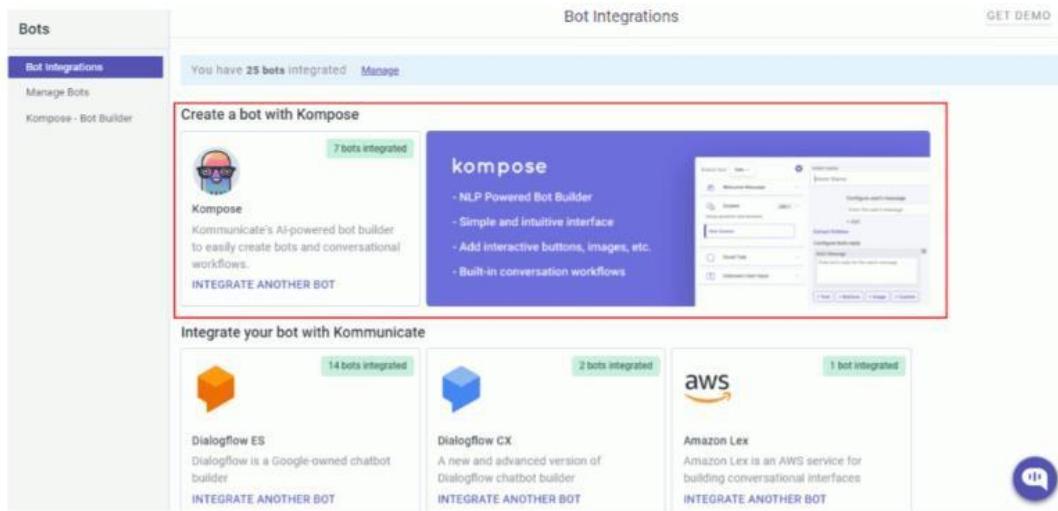


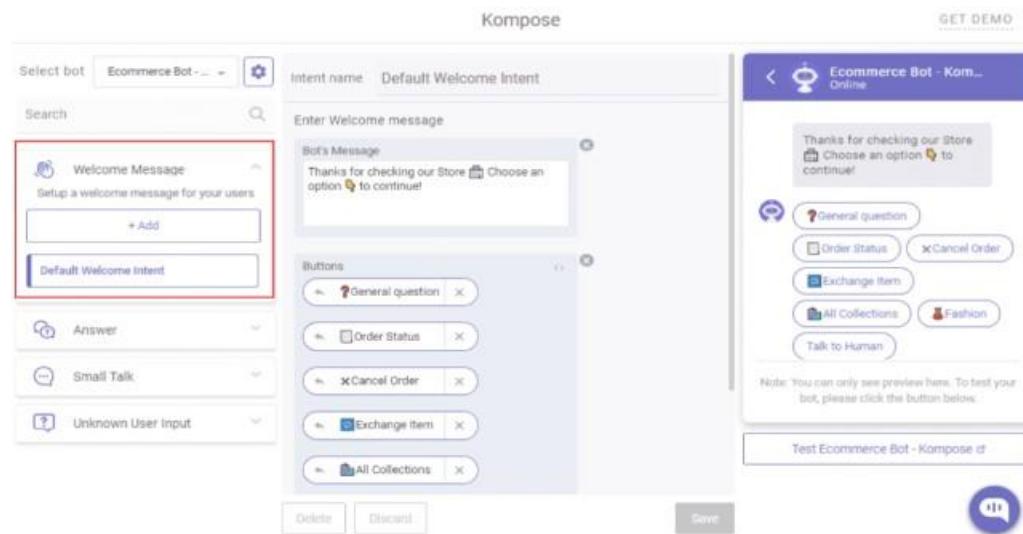
Figure 442: Implementing Chatbot in flutter application research (I)

### Step 2: Create welcome messages & answers for your flutter chatbot

Go to the 'Kompose – Bot Builder' section and select the bot you created.

First set the welcome message for your chatbot. The welcome message is the first message that the chatbot sends to the user who initiates a chat.

- Click the “Welcome Message” section, in the “Enter Welcome message – Bot’s Message” box provide the message your chatbot should be shown to the users when they open the chat and then save the welcome intent.



Once the welcome message creation is completed, the next step is to create answers for your chatbot.

Figure 443: Implementing Chatbot in Flutter Application Research (II)

## Another E-payment option implementation (E-Sewa)

**Prerequisite**

You must have a **merchant** account of **eSewa** from where you will have **clientID** and **secretKey**.

👉👉👉 For making merchant account, visit [here](#).

After getting **clientID** and **secretKey**, you can proceed to integration part easily.

If you only want to test, you can use the following credentials:

```
clientID: "JB8BBQ4aD0UqIThFJwAKBgAXEUkEGQUBBAwdOgABHD4DChwUAB0R"  
secretKey: "BhwIwQQADhIYSxIExMcAgFXFhcOBwAKBgAXE0=="
```

For testing phase, **eSewa ID** and **password** are:

```
eSewa ID: 9806800001/9806800002/9806800003/9806800004/9806800005  
Password: Nepal@123
```

Let's begin:

1. Install [esewa\\_pnp package](#):

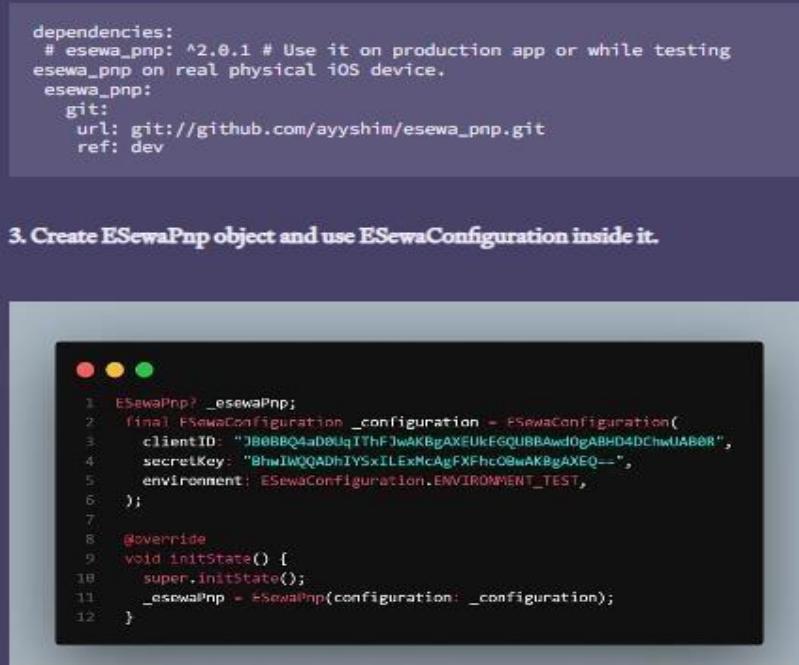
```
esewa_pnp: <latest version>
```

2. Configuration (supported for Android and iOS only)

*For Android:* Add the following line inside `AndroidManifest.xml`

```
<application  
    ...  
    android:theme="@style/Theme.AppCompat.Light.NoActionBar"  
    ...>
```

Figure 444: E-Sewa payment integration research (I)



```
dependencies:
  # esewa_pnp: ^2.0.1 # Use it on production app or while testing
  esewa_pnp on real physical iOS device.
  esewa_pnp:
    git:
      url: git://github.com/ayyshim/esewa_pnp.git
      ref: dev

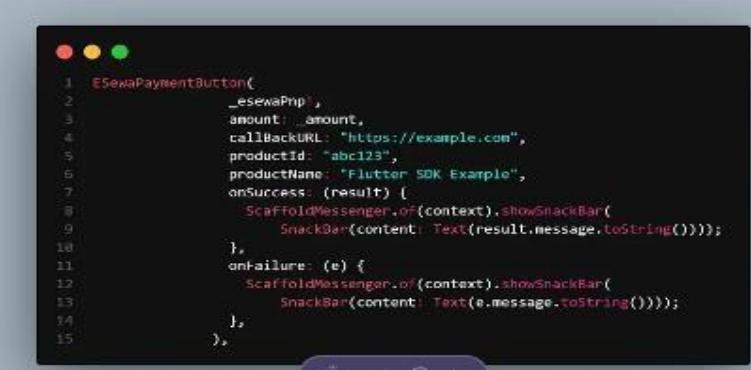
3. Create ESewaPnp object and use ESewaConfiguration inside it.
```

```
1: EsewaPnp? _esewaPnp;
2: final EsewaConfiguration _configuration = EsewaConfiguration(
3:   clientID: "JB0BBQ4aD8UqITHFJwAKBgAXEUkFGQUBBawd0gABHD4DChwJABER",
4:   secretKey: "BhuIWQADHIYSxILExhMcAgFXFhcOBwAKBgAXEQ--",
5:   environment: EsewaConfiguration.ENVIRONMENT_TEST,
6: );
7:
8: @override
9: void initState() {
10:   super.initState();
11:   _esewaPnp = EsewaPnp(configuration: _configuration);
12: }
```

4. By default, you have been provided 3 types of eSewa Button to initiate payment. (For this, you have to add assets file. You can download it from here).

Note: amount should be of type double.

- **ESewaPaymentButton with dark background**



```
1: EsewaPnp(
2:   _esewaPnp,
3:   amount: _amount,
4:   callBackURL: "https://example.com",
5:   productId: "abc123",
6:   productName: "Flutter SDK Example",
7:   onSuccess: (result) {
8:     ScaffoldMessenger.of(context).showSnackBar(
9:       SnackBar(content: Text(result.message.toString())));
10:   },
11:   onFailure: (e) {
12:     ScaffoldMessenger.of(context).showSnackBar(
13:       SnackBar(content: Text(e.message.toString())));
14:   },
15: ),
```

Figure 445: E-Sewa payment integration research (II)