



**islington college**  
(इस्लिंग्टन कॉलेज)

**Module Code & Module Title**  
**CS6P05NI Final Year Project**

**Contactless Transportation Payment (Web, Mobile and Linux)**

**Assessment Weightage & Type**  
**40% Final Project Report**  
**Semester**  
**2020 Spring**

**Assignment Due Date:**

**Assignment Submission Date:**

**Word Count: 12300**

*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.*

## Acknowledgement

continuous guidance and encouragement throughout the inception of the project idea, development, and its completion. I am of the firm belief that without the critical appraisal from the supervisors, this contactless payment system would not have seen the light of the day.

The insight and the reflection that I have been able to receive during this time period has helped me follow a proper methodology for project development and implement the solutions in the project through an iterative process. I am also thankful to Islington College for providing the resources and the support which were requisite for the completion of this project.

## Abstract

Cash based transaction, especially in transportation industry is slower than contactless transaction. With an already slow method of transaction in transportation; COVID-19 pandemic, with lockdown and limitation of human contact, brought a shift in our way of life. While it brought changes in the daily functioning of our lives, it also propelled the need to reflect on the way we functioned. Sudden suspension of public vehicles, which serves as a primary source for daily traveling, with the premise of preventing COVID transmission by cutting off contact with any living being or objects including cash was unprecedented.

This sudden interlude led to the reasoning of creating a contactless payment transportation suitable for Nepal. This project report provides a brief introduction to the reasoning of the project with the aim of using technology as easier and safer method of traveling and facilitating the transition of traditional cash based payment system to contactless payment system. The report also delves on the aims, objectives, and structure of the project with a focus on the project as solution and action oriented. To highlight the project's solution oriented aspect, the report provides a brief outline of the project background with a comparison of features and critical evaluation of the solution. It also provides the rundown of the development phases using RUP methodology. With the inclusion of pre and post survey results in the report; it aims to provide the summation of the design and implementation reflecting the aim of project being action oriented.

Critical assessment of any project is required to assure its viability. This report through the inclusion of the test plan and a critical analysis on the testing, advantages, limitation, and the future work aims to critically assess the outcome and feasibility of the work.

## Table of Contents

Chapter 1. Introduction .....	1
1.1 Project Description .....	2
1.2 Current Scenario .....	3
1.3 Problem Domain .....	4
1.4 Project as A Solution .....	5
1.5 Aim and Objectives .....	6
1.6 Structure of The Report.....	7
1.6.1 Background .....	7
1.6.2 Development .....	7
1.6.3 Testing and Analysis .....	7
1.6.4 Conclusion.....	7
Chapter 2. Background .....	8
2.1 About the End Users .....	8
2.1.1 Customer.....	8
2.1.2 End User.....	8
2.2 Understanding the Solution .....	9
2.2.1 Solution for End Users.....	9
2.2.2 Solution for Customers .....	9
2.3 Similar Projects .....	10
2.3.1 NCMC India .....	10
2.3.2 EZ-Link Singapore .....	11
2.3.3 Oyster Card London .....	12
2.4 Comparisons .....	13
Chapter 3. Development .....	14

3.1	Considered Methodologies.....	14
3.1.1	Waterfall .....	14
3.1.1.1	Advantages of Waterfall .....	14
3.1.1.2	Disadvantages of Waterfall .....	15
3.1.1.3	Justification for Not Choosing Waterfall .....	15
3.1.2	Scrum .....	16
3.1.2.1	Advantages of Scrum.....	16
3.1.2.2	Disadvantages of Scrum .....	16
3.1.2.3	Justification for Not Choosing Scrum .....	17
3.1.3	Rapid Application Development (RAD).....	18
3.1.3.1	Advantages of RAD.....	18
3.1.3.2	Disadvantages of RAD.....	18
3.1.3.3	Justification for Not Choosing RAD .....	19
3.2	Selected Methodology.....	20
3.2.1	Rational Unified Process .....	20
3.2.1.1	Justification for Using RUP .....	20
3.3	Phases of Methodology.....	22
3.3.1	Inception .....	22
3.3.2	Elaboration .....	22
3.3.3	Construction .....	22
3.3.4	Transition.....	23
3.4	Survey Results.....	24
3.4.1	Pre-Survey Results.....	25
3.4.2	Post-Survey Results .....	25
3.5	Requirement Analysis .....	26

3.5.1	Feature Requirements .....	26
3.5.2	Software Requirements .....	27
3.5.2.1	Planning and Development Tracking .....	27
3.5.2.2	IDE .....	27
3.5.2.3	Web Application .....	27
3.5.2.4	Mobile Application .....	27
3.5.2.5	Desktop Application .....	27
3.5.2.6	Web Plugins / SDK.....	28
3.5.2.7	Mobile Plugins.....	28
3.5.3	Hardware Requirements.....	29
3.6	Design.....	30
3.6.1	Application Logo .....	30
3.6.2	System Design .....	31
3.6.2.1	Use Case Diagram.....	31
3.6.2.2	ER Diagram .....	32
3.6.2.3	Activity Diagram.....	33
3.6.3	Feature Design .....	36
3.6.3.1	User Signup .....	36
3.6.3.2	Login User.....	38
3.6.3.3	Manage Profile.....	40
3.6.3.4	Write Card.....	42
3.6.3.5	Deploy Card .....	44
3.6.3.6	Top-up.....	46
3.6.3.7	Read Card.....	48
3.6.3.8	View Travel Stats .....	52

3.7	Implementation.....	54
3.7.1	Database Design .....	54
3.7.2	Web Application.....	59
3.7.2.1	App Separation .....	59
3.7.2.2	Web Application Endpoints .....	60
3.7.2.3	User Signup .....	62
3.7.2.4	User Signup with OTP.....	64
3.7.2.5	Login (Web and Device API).....	66
3.7.2.6	Logout.....	68
3.7.2.7	Admin Dashboard .....	68
3.7.2.8	User Dashboard.....	70
3.7.2.9	User Profile .....	73
3.7.2.10	Top-up .....	76
3.7.2.11	Invalid Trips .....	79
3.7.3	Desktop Application.....	81
3.7.3.1	Write Card.....	81
3.7.3.2	Test Card .....	84
3.7.3.3	Deploy Card .....	87
3.7.3.4	Read Card.....	89
3.7.4	Mobile Application .....	94
Chapter 4.	Testing and Analysis .....	98
4.1	Test Plan .....	98
4.1.1	Unit Testing, Test Plan .....	98
4.1.2	System Testing, Test Plan.....	98
4.2	Unit Testing.....	99

4.2.1	User Signup.....	99
4.2.1.1	Form Validation.....	99
4.2.1.2	Database Unique Constraints (duplicate phone and email) .....	101
4.2.1.3	Valid Signup.....	103
4.2.2	User Login .....	105
4.2.2.1	Web Application .....	105
4.2.2.2	Mobile Login.....	108
4.2.2.3	Desktop Login .....	110
4.2.3	User Logout.....	113
4.2.3.1	Web Logout.....	113
4.2.3.2	Mobile Logout .....	114
4.2.3.3	Desktop Logout.....	115
4.2.4	View Dashboard / Stats .....	117
4.2.4.1	Web Application .....	117
4.2.4.2	Mobile Application .....	122
4.2.5	Manage Profile .....	123
4.2.5.1	Invalid Data Upload.....	123
4.2.5.2	Unique Key Violation.....	124
4.2.5.3	Valid Data Change .....	126
4.2.5.4	Update Credentials .....	128
4.2.5.5	Password Change.....	130
4.2.5.6	Password Reset.....	134
4.2.6	Top-up (Load Card) .....	139
4.2.6.1	Invalid Amount .....	139
4.2.6.2	Exceeding Payment Limit.....	140

4.2.6.3	Right Amount .....	141
4.2.7	Manage Users .....	144
4.2.7.1	View User.....	144
4.2.7.2	Search User.....	146
4.2.7.3	Change User Status / Give Discount.....	148
4.2.7.4	View Invalid Trips.....	152
4.2.8	Write Card .....	154
4.2.8.1	Initial Write .....	154
4.2.8.2	Confirm Card Write .....	157
4.2.9	Deploy Card.....	159
4.2.9.1	Correct Deploy .....	159
4.2.9.2	Un-deploy Card.....	161
4.2.10	Read Card.....	163
4.2.10.1	Tap With 0 Balance .....	163
4.2.10.2	Tap Twice on Entry.....	164
4.2.10.3	Tap Exit Before Entry .....	166
4.2.10.4	Tap on Entry and Exit .....	167
4.3	System Testing .....	172
4.3.1	Application Build Tests .....	172
4.3.1.1	Web Build.....	172
4.3.1.2	Phone Build.....	173
4.3.1.3	Desktop Build.....	174
4.3.2	API Data Tests .....	176
4.3.2.1	Card-Not-Written User Data.....	176
4.3.2.2	Travel Log Data .....	177

4.3.3 API Permissions Tests.....	178
4.3.3.1 Travel Log API for Admin .....	178
4.3.3.2 Card Write Data API for User.....	179
4.4 Critical Analysis.....	180
4.4.1 Working Under Development Methodology .....	180
4.4.2 Analyzing Failed Test Cases .....	181
4.4.2.1 User Dashboard Test.....	181
4.4.2.2 Travel Entry/Exit Test.....	183
4.4.3 Test Summary .....	184
4.4.3.1 Strong Aspects of the System.....	184
4.4.3.2 Weak Aspects of the System .....	184
Chapter 5. Conclusion .....	185
5.1 Legal, Social and Ethical Issues.....	186
5.1.1 Legal Issues .....	186
5.1.1.1 User Privacy.....	186
5.1.1.2 Card Data Breach .....	186
5.1.1.3 Limited Tracking.....	186
5.1.2 Social Issues .....	187
5.1.2.1 Travel Syndicate .....	187
5.1.2.2 Replacing Bus Staff.....	187
5.1.3 Ethical Issues .....	188
5.1.3.1 Card Scams .....	188
5.1.3.2 System Exploit (Distance Calculation) .....	188
5.2 Advantages .....	190
5.2.1 Faster Payment .....	190

5.2.2	In Built Discount.....	190
5.2.3	Expenditure Tracking.....	190
5.2.4	Customer Relation Establishment .....	190
5.2.5	Eco Friendliness Promotion.....	191
5.2.6	Eradication of Slower Systems .....	191
5.3	Limitations.....	192
5.3.1	Phone Number Change .....	192
5.3.2	Cash Payment Tracking .....	192
5.3.3	Locating Exit for Invalid Trips.....	192
5.3.4	Transaction Time (Sever Location).....	193
5.3.5	Mobile Internet for Buses.....	193
5.4	Future Work .....	194
5.4.1	Addressing Limitations.....	194
5.4.1.1	Phone Number Change Feature .....	194
5.4.1.2	Track Cash Payment.....	194
5.4.1.3	Implementing AI .....	194
5.4.1.4	Stable Internet Connection for Bus .....	194
5.4.2	Asynchronous Programming in Desktop Application .....	195
5.4.3	Cancel Card Operations .....	195
5.4.4	Tap to Pay Using Apple or Google Pay .....	195
5.4.5	Using Better Card Validation .....	195
5.4.6	Individual Buckets for Users .....	196
5.4.7	Web Features in Mobile Application with Notifications .....	196
5.4.8	Disable Card if Stolen.....	196
5.4.9	Discount Plans.....	196

Chapter 6. References .....	197
Chapter 7. Appendix .....	201
7.1 Appendix A: Pre-Survey .....	201
7.1.1 Pre-Survey Form .....	201
7.1.2 Sample of Filled Pre-Survey Forms .....	204
7.1.3 Pre-Survey Result .....	207
7.2 Appendix C: Sample Codes .....	213
7.2.1 Sample Code of the UI .....	213
7.2.1.1 Desktop Tree View.....	213
7.2.1.2 Mobile Data Tiles .....	214
7.2.1.3 Web Forms .....	216
7.2.1.4 Web Tables.....	217
7.2.2 Sample Code of the Applications.....	219
7.2.2.1 Read / Write Cards.....	219
7.2.2.2 Server Requests using Python.....	220
7.3 Appendix D: Designs.....	221
7.3.1 Gantt Chart .....	221
7.3.1.1 Final Gantt Chart.....	221
7.3.1.2 Gantt Chart Iterations.....	223
7.3.2 Work Breakdown Structure.....	224
7.3.3 Hardware Architecture.....	225
7.3.4 ERD Iterations .....	226
7.3.5 Data Flow Diagrams .....	229
7.3.5.1 Context Level DFD.....	229
7.3.5.2 Level 1 DFD .....	230

7.3.5.3	Level 2 DFD Iterations .....	231
7.3.6	Use Case Diagrams .....	233
7.3.6.1	Final Use Case Diagram.....	233
7.3.6.2	Use Case Iterations.....	234
7.3.6.3	High Level Use Case Description.....	236
7.3.7	Wireframes .....	239
7.3.8	Mock-ups .....	241
7.3.8.1	Web Mockups .....	241
7.3.8.2	Mobile Mockups .....	246
7.4	Appendix E: Screenshots of the System .....	247
7.4.1	Web Application.....	247
7.4.1.1	Customer .....	247
7.4.1.2	Admin.....	254
7.4.2	Desktop Application.....	256
7.4.3	Mobile Application .....	258
7.5	Appendix F: User Feedback.....	259
7.5.1	User Feedback Form .....	259
7.5.2	Sample of Filled User Feedback Forms.....	262
7.5.3	User Feedback Result .....	265
7.6	Appendix G: Future Work.....	270
7.6.1	Readings for Future Work.....	270
7.6.1.1	Firebase Cloud Messaging .....	270
7.6.1.2	EMV Card .....	273
7.6.1.3	Asynchronous Programming (Tkinter).....	275
7.7	Similar Projects Continued.....	278

7.7.1	Go Card Queensland.....	278
7.7.2	Bicing Barcelona.....	279

## Table of Figures

Figure 1 Components of Payment System .....	2
Figure 2 NCMC Operational Card .....	10
Figure 3 EZ-Link Operational Cards .....	11
Figure 4 Oyster Card in Operation (EveningStandard, 2014) .....	12
Figure 5 Steps Involved in Waterfall Model (Lewis, 2019) .....	14
Figure 6 Basic View of Scrum Methodology (Laurance, 2020) .....	16
Figure 7 Four Phases of Rapid Application Development (Lucidchart, 2018) .....	18
Figure 8 Phases of RUP (RationalSoftwareCorporation, 2002) .....	20
Figure 9 Age Demographic of Responders .....	24
Figure 10 Occupation of Responders .....	24
Figure 11 Required Plugins in Web Application .....	28
Figure 12 Application Logo .....	30
Figure 13 System Use Case .....	31
Figure 14 System ER Diagram .....	32
Figure 15 User Web Application Activity Diagram .....	33
Figure 16 Admin Web Application Activity Diagram .....	34
Figure 17 Admin Desktop Application Activity Diagram .....	35
Figure 18 User Signup Level 2 DFD .....	36
Figure 19 Sign Up User Communication Diagram .....	37
Figure 20 Sign Up User Sequence Diagram .....	37
Figure 21 Login User Level 2 DFD .....	38
Figure 22 Login User Communication Diagram .....	38
Figure 23 Login User Sequence Diagram .....	39
Figure 24 Manage Profile Level 2 DFD .....	40
Figure 25 Manage Profile Communication Diagram .....	40
Figure 26 Manage Profile Sequence Diagram .....	41
Figure 27 Write Card Level 2 DFD .....	42
Figure 28 Write Card Communication Diagram .....	42
Figure 29 Write Card Sequence Diagram .....	43
Figure 30 Test Card Sequence Diagram .....	43

Figure 31 Deploy Card Level 2 DFD .....	44
Figure 32 Deploy Card Communication Diagram .....	44
Figure 33 Deploy Card Sequence Diagram.....	45
Figure 34 Top-up Level 2 DFD .....	46
Figure 35 Top-up Communication Diagram.....	46
Figure 36 Top-up Sequence Diagram .....	47
Figure 37 Entry Read DFD .....	48
Figure 38 Exit Read DFD .....	49
Figure 39 Read Card Communication Diagram.....	50
Figure 40 Read Card Sequence Diagram .....	51
Figure 41 View Travel Stats Level 2 DFD.....	52
Figure 42 View Travel Stats Communication Diagram .....	52
Figure 43 View Travel Stats Sequence Diagram.....	53
Figure 44 Model for User Status.....	54
Figure 45 Custom User Model.....	54
Figure 46 User Model Manager.....	55
Figure 47 Model for Account Creation.....	55
Figure 48 Model for Cards.....	56
Figure 49 Model for Payments .....	56
Figure 50 Model for Bus Data.....	56
Figure 51 Model for Bus Travel Log .....	56
Figure 52 Model for User Entries.....	57
Figure 53 Model for User Exits .....	58
Figure 54 Project App Separation .....	59
Figure 55 URLs for Accounts App .....	60
Figure 56 URLs for Raspberry Pi .....	60
Figure 57 URLs for Login and Travel History on Mobile Devices .....	61
Figure 58 URLs for Making Travel Log.....	61
Figure 59 Basic Signup Form Code .....	62
Figure 60 User Creation without OTP Validation.....	63
Figure 61 Registration UI.....	64

Figure 62 Verify Phone UI .....	64
Figure 63 User Registration Model Form.....	64
Figure 64 Registration Backend Buffer.....	65
Figure 65 Registration Backend .....	65
Figure 66 Login UI.....	66
Figure 67 Login Front End Code .....	66
Figure 68 Login Backend.....	67
Figure 69 Mobile/Desktop Login Backend.....	67
Figure 70 Logout Backend .....	68
Figure 71 Admin Dashboard Table.....	68
Figure 72 Admin Dashboard Search .....	69
Figure 73 Admin Dashboard Backend.....	69
Figure 74 User Dashboard .....	70
Figure 75 User Stat board .....	71
Figure 76 User Dashboard and Stat board Backend.....	72
Figure 77 User Profile from User Page.....	73
Figure 78 User Credential from User Page .....	73
Figure 79 User Profile Backend.....	74
Figure 80 User Profile from Admin View.....	75
Figure 81 User Profile from Admin Backend .....	75
Figure 82 Top-up UI .....	76
Figure 83 Khalti Integration in Top-up .....	76
Figure 84 Payment form Backend .....	77
Figure 85 Khalti Verification Backend.....	78
Figure 86 Invalid Trips Table UI .....	79
Figure 87 Invalid Trips UI .....	79
Figure 88 Invalid Trips Backend .....	80
Figure 89 Invalid Trips Front End .....	80
Figure 90 Write Card GUI.....	81
Figure 91 Write Card Application Backend.....	82
Figure 92 Write Card Server Backend.....	83

Figure 93 Test Card GUI .....	84
Figure 94 Test Card Application Backend .....	85
Figure 95 Test Card Server Backend .....	86
Figure 96 Deploy Card GUI .....	87
Figure 97 Deploy Card Application Backend .....	87
Figure 98 Deploy Card Server Backend .....	88
Figure 99 Read Card GUI.....	89
Figure 100 Entry Tap Application Backend .....	90
Figure 101 Exit Tap Application Backend.....	91
Figure 102 Entry Read Server Backend.....	91
Figure 103 Google Maps Distance Backend .....	92
Figure 104 Exit Read Backend (1) .....	92
Figure 105 Exit Read Backend – Fare Calculation (2) .....	93
Figure 106 Mobile Login UI .....	94
Figure 107 Mobile Dashboard UI.....	94
Figure 108 Mobile Travel Log Backend .....	94
Figure 109 Travel Log Serializer .....	95
Figure 110 User Data Serializer .....	95
Figure 111 Travel Log Front End (1) .....	96
Figure 112 Travel Log Front End (2) .....	97
Figure 113 Adding Invalid Data to Fields.....	99
Figure 114 JavaScript Validation on Signup.....	100
Figure 115 Server Side Signup Validation.....	100
Figure 116 Existing Data in Database .....	101
Figure 117 Error Messages for Unique Constraint Violation.....	102
Figure 118 Valid User Data in Signup Form .....	103
Figure 119 Entering Phone Verification Code .....	104
Figure 120 SMS Code on Entered Mobile Phone.....	104
Figure 121 Redirect To Login Page after Account Creation .....	104
Figure 122 New Signup in Database Dashboard .....	104
Figure 123 Invalid Data in Web Login .....	105

Figure 124 Login Failure Message .....	105
Figure 125 Valid Data Entry in Web Login.....	106
Figure 126 User Redirected to Dashboard After Login.....	107
Figure 127 Invalid Data in Mobile Application.....	108
Figure 128 Invalid Login Message in Mobile App .....	108
Figure 129 Valid Data Entry in Mobile Login .....	109
Figure 130 Redirect to Dashboard After Login .....	109
Figure 131 Empty Field Validation on Desktop Login.....	110
Figure 132 Wrong Credential Validation on Desktop Login.....	110
Figure 133 Valid Credentials in Desktop Login.....	111
Figure 134 Desktop Login Successful .....	112
Figure 135 Pressing Web Logout Button.....	113
Figure 136 Login Page Redirect.....	113
Figure 137 Mobile Logout Action.....	114
Figure 138 Mobile Redirect after Logout .....	114
Figure 139 Clicking Desktop Logout.....	115
Figure 140 Successful Logout .....	116
Figure 141 Web Dashboard with Travel History .....	117
Figure 142 Travel Stats related to Travel Distances.....	118
Figure 143 Travel Stats for Walking Distance and Carbon Emission .....	119
Figure 144 Clicking on Change Button.....	120
Figure 145 Changed Travel History After Button Click .....	121
Figure 146 Mobile Dashboard for Logged In User.....	122
Figure 147 Mobile Travel Log .....	122
Figure 148 JavaScript Field Validation .....	123
Figure 149 Adding Existing Email in Profile.....	124
Figure 150 Error Message for Database Constraint Violation in Update Profile .....	125
Figure 151 Valid Data Update in User Profile.....	126
Figure 152 Profile Data Update Success .....	127
Figure 153 Default Credentials.....	128
Figure 154 Uploaded Credentials Preview .....	129

Figure 155 Updated Credentials.....	129
Figure 156 Entering Incorrect Old Password.....	130
Figure 157 Error Messages for Incorrect Passwords .....	131
Figure 158 Entering Different Passwords .....	131
Figure 159 Error Message for Password Mismatch.....	132
Figure 160 Entering Valid Data .....	132
Figure 161 Password Change Message .....	133
Figure 162 Logging In With New Password.....	133
Figure 163 Login Successful .....	133
Figure 164 Entering Valid Email for Password Reset.....	134
Figure 165 Password Reset Link Sent Message.....	134
Figure 166 Password Reset Link via Email .....	135
Figure 167 Password Reset Form.....	135
Figure 168 Invalid Password Entry in Reset.....	136
Figure 169 Error Message for Invalid Password in Reset .....	136
Figure 170 Resetting Password with Proper Data.....	137
Figure 171 Password Change Message .....	137
Figure 172 Logging In With New Password.....	138
Figure 173 User Logged In.....	138
Figure 174 String Validation for Payment.....	139
Figure 175 Negative Amount Validation for Payment.....	139
Figure 176 Exceeding Payment Limit.....	140
Figure 177 Exceeding Top-up Amount Limit .....	140
Figure 178 Entering Valid Amount.....	141
Figure 179 Payment Portal with Entered Amount.....	142
Figure 180 Confirmation Code for Verification.....	142
Figure 181 Payment Confirmation Code .....	143
Figure 182 Top-up Successful Message .....	143
Figure 183 Updated Card Amount.....	143
Figure 184 Unverified Users.....	144
Figure 185 Resubmission Required Users.....	144

Figure 186 Verified Users.....	145
Figure 187 Users with Pending Review.....	145
Figure 188 Customer Search Field.....	146
Figure 189 Unmatching Search Field .....	147
Figure 190 Matching Search Field.....	147
Figure 191 Viewing Pending Review User .....	148
Figure 192 Account Review Form .....	148
Figure 193 Current User Verification Status .....	149
Figure 194 Current Card Discount Status.....	149
Figure 195 Changing User Status .....	150
Figure 196 Update Confirmation .....	150
Figure 197 Account Status Changed.....	151
Figure 198 Card Discounted.....	151
Figure 199 Invalid Trips Dashboard .....	152
Figure 200 Entry Points for Invalid Trips.....	152
Figure 201 Write Without Selecting User .....	154
Figure 202 User Selection to Write Card.....	155
Figure 203 Current Card Status of Selected User.....	155
Figure 204 Card Written for Selected User.....	156
Figure 205 Running Test Card Operation.....	157
Figure 206 Card Write Confirmation Dialog Box.....	158
Figure 207 Card Written Status Change Confirmation .....	158
Figure 208 User Card Status After Confirmation .....	158
Figure 209 Card Deploy Action .....	159
Figure 210 Current Delivery Status .....	159
Figure 211 Card Deliver Confirmation .....	160
Figure 212 Updated Delivery Status.....	160
Figure 213 Card Un-deploy Action .....	161
Figure 214 Current Card Delivery Status.....	161
Figure 215 Deploying Card for 'card-delivered' User .....	162
Figure 216 Updated Card Delivery Status.....	162

Figure 217 Error Message for Invalid Entry Tap .....	163
Figure 218 First Valid Entry .....	164
Figure 219 Second Tap on Entry.....	165
Figure 220 Error Message for Exit Tap Before Entry Tap .....	166
Figure 221 Successful Card Entry for Valid Trip.....	167
Figure 222 Unsuccessful Exit Tap .....	168
Figure 223 Successful Card Entry for Valid Trip (2) .....	169
Figure 224 Card Exit Successful for Valid Entry.....	170
Figure 225 Trip Entry Data .....	170
Figure 226 Trip Exit Data.....	171
Figure 227 Latest Build Log on Heroku .....	172
Figure 228 Flutter Build Successful.....	173
Figure 229 Application Installed on Android Device .....	173
Figure 230 Read Card Application .....	174
Figure 231 Write Card Application.....	175
Figure 232 Card-Not-Written User Details for Admin .....	176
Figure 233 Successful API Response for Travel Log .....	177
Figure 234 Denied Permission for Admin Travel Log .....	178
Figure 235 Permission Denied for User on Admin API.....	179
Figure 236 Variable Name in Server .....	181
Figure 237 Variable Name in Template .....	181
Figure 238 Fixing Variable Spelling.....	182
Figure 239 Fixed Stat Board for Walking Trips.....	182
Figure 240 Wrong Coordinates on Entry .....	183
Figure 241 Bus Route Simulation.....	188
Figure 242 Bus Shorter Distance Exploit.....	189
Figure 243 Way Points in Bus Route.....	189
Figure 244 Pre Survey Form (1).....	201
Figure 245 Pre Survey Form (2).....	202
Figure 246 Pre Survey Form (3).....	203
Figure 247 Pre Survey Form Sample (1).....	204

Figure 248 Pre Survey Form Sample (2).....	205
Figure 249 Pre Survey Form Sample (3).....	206
Figure 250 First Question Response .....	207
Figure 251 Second Question Response.....	207
Figure 252 Third Question Response.....	208
Figure 253 Fourth Question Response .....	208
Figure 254 Fifth Question Response .....	209
Figure 255 Sixth Question Response.....	209
Figure 256 Seventh Question Response.....	210
Figure 257 Eighth Question Response.....	210
Figure 258 Nineth Question Response .....	211
Figure 259 Tenth Question Response.....	211
Figure 260 Eleventh Question Response .....	212
Figure 261 Tree View for Desktop.....	213
Figure 262 Load Tree View in GUI.....	213
Figure 263 Log Tile (1) .....	214
Figure 264 Log Tile (2) .....	215
Figure 265 Log Tile Container .....	215
Figure 266 General Form Template in Web Application.....	216
Figure 267 Table Tabs in Table.....	217
Figure 268 Table Tabs in Admin Dashboard .....	218
Figure 269 Write Card .....	219
Figure 270 Write Card .....	219
Figure 271 Python Server Requests .....	220
Figure 272 Final Gantt Chart for Inception and Elaboration .....	221
Figure 273 Final Gantt Chart for First 5 Iterations .....	221
Figure 274 Final Gantt Chart for Last 5 Iterations .....	222
Figure 275 Final Gantt Chart for Transition .....	222
Figure 276 First Gantt Chart Revision .....	223
Figure 277 Second Gantt Chart Revision .....	223
Figure 278 Work Breakdown Structure Chart.....	224

Figure 279 Circuit Diagram for GPS Module (Hackster, 2019).....	225
Figure 280 Circuit Diagram for RFID Module (Gus, 2017) .....	225
Figure 281 ER Diagram First Iteration.....	226
Figure 282 ER Diagram Second Iteration.....	226
Figure 283 ER Diagram Third Iteration.....	227
Figure 284 ER Diagram Final Iteration .....	228
Figure 285 Context Level DFD .....	229
Figure 286 Level 1 DFD .....	230
Figure 287 Sign Up User Level 2 DFD First Iteration .....	231
Figure 288 Sign Up User Level 2 DFD Second Iteration .....	232
Figure 289 Final Use Case Diagram .....	233
Figure 290 Use Case Diagram First Iteration .....	234
Figure 291 Use Case Diagram Second Iteration .....	235
Figure 292 Wire Frames for Base Web Pages .....	239
Figure 293 Wireframes for Mobile Application.....	240
Figure 294 Web Signup Mockup .....	241
Figure 295 Web Login Mockup.....	241
Figure 296 Web User Dashboard Mockup .....	242
Figure 297 Web User Travel Stat board Mockup .....	242
Figure 298 Web User Travel Log Mockup .....	243
Figure 299 Web User Profile Mockup.....	243
Figure 300 Web Admin Dashboard Mockup.....	244
Figure 301 Web Admin View User Mockup .....	244
Figure 302 Web Admin View User Page Mockup.....	245
Figure 303 Mobile Login Screen Mockup .....	246
Figure 304 Mobile Dashboard Screen Mockup .....	246
Figure 305 Registration UI.....	247
Figure 306 Verify Phone UI .....	247
Figure 307 Login UI .....	248
Figure 308 User Dashboard .....	248
Figure 309 User Stat Board (1) .....	249

Figure 310 User Stat Board (2) .....	249
Figure 311 User Profile .....	250
Figure 312 User Credentials.....	250
Figure 313 User Top-up .....	251
Figure 314 Reset Password Screen.....	251
Figure 315 Reset Link Sent Screen.....	252
Figure 316 Password Change Screen.....	252
Figure 317 Password Changed Screen.....	253
Figure 318 Admin Dashboard.....	254
Figure 319 Admin Search User UI.....	254
Figure 320 Admin View User.....	255
Figure 321 Admin View Invalid Trips Users.....	255
Figure 322 Admin View Invalid Trip .....	255
Figure 323 Desktop Login Screen .....	256
Figure 324 Desktop Dashboard.....	256
Figure 325 Desktop Deploy Card .....	257
Figure 326 Desktop Read Card.....	257
Figure 327 Mobile Login .....	258
Figure 328 Mobile Dashboard .....	258
Figure 329 User Feedback Form (1) .....	259
Figure 330 User Feedback Form (2) .....	260
Figure 331 User Feedback Form (3) .....	261
Figure 332 User Feedback Sample (1) .....	262
Figure 333 User Feedback Sample (2) .....	263
Figure 334 User Feedback Sample (3) .....	264
Figure 335 Average Age Group for Sample Space .....	265
Figure 336 Percentage of Commuters .....	265
Figure 337 System User Experience Feedback .....	266
Figure 338 User Signup Feedback.....	266
Figure 339 Read Card Feedback .....	267
Figure 340 Implementation Expectation Feedback .....	267

Figure 341 Mobile Feature Replication Feedback .....	268
Figure 342 Quality of Transportation Feedback .....	268
Figure 343 Buss Staff Replacement Feedback .....	269
Figure 344 Extra Feature Request Feedback.....	269
Figure 345 Firebase for Web Setup (Google, 2021).....	270
Figure 346 Receiving Firebase Messages on Web App (Google, 2021).....	271
Figure 347 FCM Package for Django (Rojko, 2021).....	272
Figure 348 Introduction to EMV (Square Up, 2016) .....	273
Figure 349 EMV Complementary with NFC (EMV Connection, 2015) .....	274
Figure 350 Async Programming Introduction for Tkinter (Hallen, 2018).....	275
Figure 351 Threading in Python (Hallen, 2018).....	276
Figure 352 Ending Functions in Async (Hallen, 2018).....	277
Figure 353 Tiers of Go Card.....	278
Figure 354 Bicing Card in Operation .....	279

## Table of Tables

Table 1 Feature Comparison Between Similar Projects .....	13
Table 2 Justification 1 for Waterfall .....	15
Table 3 Justification 2 for Waterfall .....	15
Table 4 Justification 3 for Waterfall .....	15
Table 5 Justification 1 for Scrum .....	17
Table 6 Justification 2 for Scrum .....	17
Table 7 Justification 3 for Scrum .....	17
Table 8 Justification 1 for RAD .....	19
Table 9 Justification 2 for RAD .....	19
Table 10 Justification 3 for RAD .....	19
Table 11 Justification 1 for RUP .....	20
Table 12 Justification 2 for RUP .....	21
Table 13 Justification 3 for RUP .....	21
Table 14 Justification 4 for RUP .....	21
Table 15 Justification 5 for RUP .....	21
Table 16 User Signup From Validation Description.....	99
Table 17 Database Unique Constraint Test Description.....	101
Table 18 Signup User with Valid Data Test Description .....	103
Table 19 Invalid Login Test Description for Web Application.....	105
Table 20 Valid Login Test Description for Web Application.....	106
Table 21 Invalid Login Test Description for Mobile Application .....	108
Table 22 Valid Login Test Description for Mobile Application.....	109
Table 23 Invalid Login Test Description for Desktop .....	110
Table 24 Valid Login Test Description for Desktop.....	111
Table 25 Web Logout Test Description .....	113
Table 26 Mobile Logout Action Description .....	114
Table 27 Desktop Logout Test Description.....	115
Table 28 Web Dashboard Stats Test Description.....	117
Table 29 Web Travel Log Buttons Test .....	120
Table 30 Mobile Dashboard Test Description.....	122

Table 31 Invalid Profile Data Upload Test Description .....	123
Table 32 Unique Key Constraint Violation in Profile Test Description .....	124
Table 33 Valid Data Update Test Description.....	126
Table 34 Update Credentials Test Description .....	128
Table 35 Password Change Test Description .....	130
Table 36 Password Reset Test Description.....	134
Table 37 Load Invalid Amount on Card Test Description .....	139
Table 38 Higher Payment Amount Test Description.....	140
Table 39 Right Payment Amount Test Description.....	141
Table 40 View User Test Description .....	144
Table 41 Search User Test Description.....	146
Table 42 User Status Change Test Description .....	148
Table 43 View Invalid Test Description .....	152
Table 44 Initial Write Test Description.....	154
Table 45 Card Test Description.....	157
Table 46 Deploy Card Test Description.....	159
Table 47 Un-Deploy Card Test Description.....	161
Table 48 Tap Twice on Entry Test Description.....	164
Table 49 Tap on Exit before Entry Description .....	166
Table 50 Valid Taps Test Description.....	167
Table 51 Web Build Test Description .....	172
Table 52 Phone Build Test Description .....	173
Table 53 Desktop Build Test Description.....	174
Table 54 User Data API Test Description.....	176
Table 55 Travel Log Data API Test Description .....	177
Table 56 Travel Log API Permission Test Description .....	178
Table 57 Card Write API Permission Test Description .....	179

## Chapter 1. Introduction

Completing a monetary transaction between two parties without the use of cash or direct contact between two parties is known as contactless payment system. Contactless payment is achieved by using mediums that operate on radio frequency to transfer data between the user and the receiver. Some popular forms of these mediums are RIFD cards, key fobs or tags that transmit radio frequency which is then read by a scanner.

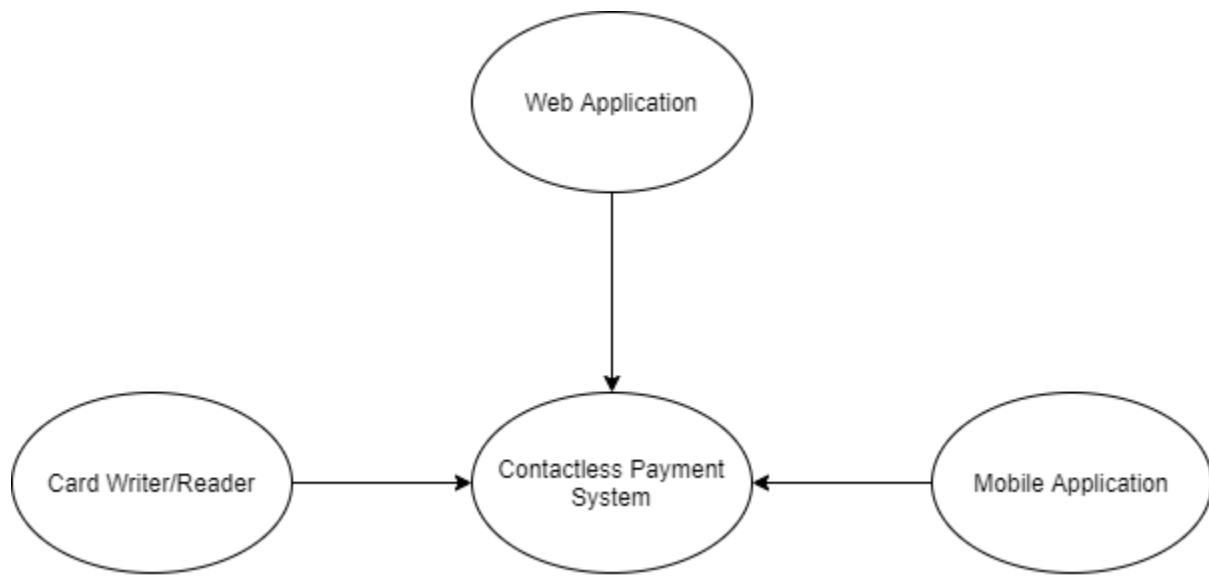
In a country where 95% of transactions are still cash based (Neupane, 2018), implementing contactless transactions into suitable sectors brings a lot of advantage to the economy in general (Durkin, 2018). Even with or without the concerns of coronavirus, cashless transactions help businesses a lot (Xiao, 2020). In the current scenario of the world, this technology sides to the advantages provided by cashless transactions over its disadvantages. (Pettinger, 2020).

With all these things in mind, I have decided to make a contactless payment system for public transportation which is suitable for Nepal. Using this technology in the most basic state of transportation would encourage people to use modern solutions for primitive problems which would not only be limited to this scenario. A user will be able to use a card for payment on public transports and the amount in card can be loaded using a web application, and a mobile application, which will have a subset of features from web app, will allow a user to see their travel data and statistics.

### 1.1 Project Description

This contactless payment system is made under the development methodology; Rational Unified Process (RUP). The features of these projects were analyzed concurrently while being able to build the application with the guidelines provided by the development methodology.

The gist of the system is to provide cards for registered users which they can use to pay their fares for public transportation. This monetary transaction is then recorded, which is useful for bus-administration to keep track of their commuters and for the commuters to view their travel history along with expenses. All the functionality of writing and reading cards for monetary transactions are built on raspberry pi along with required components whereas viewing and managing data can be done through web and mobile application.



*Figure 1 Components of Payment System*

## 1.2 Current Scenario

The only attempt on mass RFID card payment in Nepal was launched by Nuspay in 2015 later adopted by Sajha Yatayat, the only active public transportation chain in Nepal in 2017 (Red Apple PR Agency , 2015) (Chhetri, 2017). I have mentioned this payment system as an attempt since Sajha Yatayat has not been successful in implementing this payment solution within their busses, with their website not even listing the way to get their smart card. This situation for a large transportation chain shows that there is still a way to implement card payment in Nepal.

Failure of implementing smart card in Sajha Bus could be attributed to many problems residing in Nepal's transportation industry but one of the prominent reasons is their use case. As quoted from an article during its launch, "The cardholders can contact the Sajha's office to recharge their balance or they can handover the recharge amount to the onboard bus conductor. The Yatayat informed that they are preparing for more easy means to recharge the card including recharge cards." (Chhetri, 2017), it shows that the inefficiency of the solution that was aimed to be implemented. Furthermore, an easier solution for payment was never introduced and the idea of having a smart payment method slowly faded away with time.

### 1.3 Problem Domain

- 1) Cash based transaction, especially in transportation industry is slower than contactless transaction (Gundaniya, 2019).
- 2) Students and other people who can claim discounts in public transportation need to show their cards along with money every time they board a bus which makes it tedious if the person is travelling multiple times a day.
- 3) Middlemen like drivers and conductors of the vehicle have the ability to manipulate the total amount of cash that the bus owner makes or spends in a day.
- 4) Cost calculation methods for passengers is outdated in public transportation since there is no proper implementation of bus stops in Nepal.
- 5) Foreigners visiting Nepal might not have the proper understanding of currency which makes it difficult to travel to spontaneous trips without planning.
- 6) Although cash is not the primary source of transmission of corona virus, it can be unsanitary in public transportation where one note is handled by a lot of people in a single day.
- 7) In Nepal, if a public transportation staff is tested positive for corona virus, there is no efficient way of contact tracing their passengers (Khabarhub, 2020).

#### 1.4 Project as A Solution

This project acts as a direct solution to all of the problem scenarios stated above. People will not need to use cash in public transportation which will make the experience faster. Students won't need to carry both their money and ID cards to claim a discount since the discount data will be stored in their RFID card itself. Since all the transactions are done digitally, there is no need for drivers to submit their earnings to the main counter and it eliminates the time and effort to gather and deposit all the cash that needs to be distributed throughout the chain of vehicles.

The cost for the consumer will be calculated per kilometer basis which means that the customer will be charged per kilometer that they have travelled. This in turn makes it easier for tourists to understand how the travel fares work in Nepal and encourages them to spend more to travel on local transportation.

Since all records of passengers who have used a vehicle is stored in a database, it makes it easier to contact trace them. With this in mind, it can even make crime rates drop as investigators can use the travel history data of a person to determine their location in a particular point of time if they have used any public transportation.

### 1.5 Aim and Objectives

The aim of this project is to create a contactless payment system to make travelling within Nepal seamless, technology friendly and easier than currently existing system. A web application will be built where users can sign up and verify their credentials to use the cards, and login later to view their travel data and pay for the card. A mobile application will be built for registered users who can login and view their travel log and data.

The objectives which are the basis of the completion of the project are:

- To learn about web applications, mobile applications, and their workings.
- To use IOT devices to build a prototype of contactless payment system for the end user.
- To understand and implement usage of database along with web application and the card reader.
- To understand about API programming (street distance API and writing API to store and retrieve data).
- To learn about designing user interface.
- To learn about cost calculation in the travel market.

## 1.6 Structure of The Report

### 1.6.1 Background

This section of the report explains about the target market for the developed system. With, the target market in mind, existing products for the market are analyzed to make the development of the current system, on par or better than the existing systems. This includes comparison between features of existing products with aimed features of the application before development.

### 1.6.2 Development

This section documents the process of selecting a development methodology suitable for this project. Selection of methodology is based on the ease of implementation and viability of the methodology for the given project.

The upcoming phase documents the survey analysis for the project before and after development which is done using the selected methodology. Requirement analysis, design of the project and construction of the project is documented in respect to the guidelines provided by the methodology.

### 1.6.3 Testing and Analysis

This section documents the test plans and the test outcomes of the complete project. If deemed necessary, this section shows changes made in the project, if required by the test results.

Furthermore, with the result of particular test cases, analysis on how the tests affect the system in the future is analyzed and documented in this section of the report.

### 1.6.4 Conclusion

Limitation of the developed system, work left to be done or improvements that could be made in the future to this project is documented in this section of the report. Appropriate reasoning for all the limitations and future work for the project have been documented as well.

## Chapter 2. Background

### 2.1 About the End Users

#### 2.1.1 Customer

The customers of this project will be public transportation chains. Although the specific customer has not been selected for this project, the abstract requirement for the customer to use this project can be identified which is, to be able to deploy contactless payment system within their chain of public transportations. The general assumption as a developer for this project is that the target customers are willing to fix all the problems associated with cash based payment system (as mentioned in the problem statement) and make transportation facilities seamless in the department of payment.

#### 2.1.2 End User

With the background of the customer understood, the end users of this project are commuters who use public transportation on a regular basis. From the preliminary survey done amongst regular commuters, the results show that most of the people are willing to sign up to the service if it is easier to pay through the card without having to interact with the transportation conductor. This shows that there could be underlying dissatisfaction among users who cannot necessarily deem it as a problem. To avoid these dissatisfactions, thorough interaction with users and implementation of the ‘solution’ can be done to increase the interaction of end users with the system, which is a vital part for the project to grow.

## 2.2 Understanding the Solution

Contactless payment system works as a solution for primitive cash based transactions only if the system has been applied correctly to eliminate all the problems associated with the latter. This brings us to the point on how the system should be understood on the basis of problem that it is trying to eliminate, and how the end users will benefit from it.

### 2.2.1 Solution for End Users

First of all, the main goal for this project is to make it easier for commuters to travel on a regular basis with a component of letting them be interactive with their travel routine or history. As a project, this goal needs to be addressed by having a reliable system that eliminates the problems that were found out during research phase and direct interaction with the users themselves.

For the development of these features, some assumptions need to be understood. There should be no involvement of cash in any way possible, commuters need not worry about having physical money for travel. This problem is addressed by the logic of using a card to pay inside the vehicle, and a web application with mobile application (in future) where users can easily load their cards using their preferred method of electronic transaction.

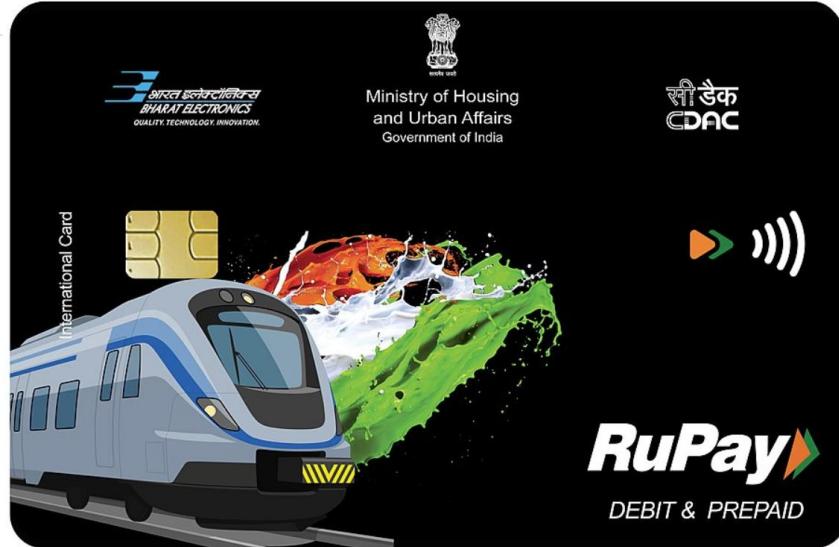
### 2.2.2 Solution for Customers

Another assumption for the usage of this project is for the transportation owners to be able to engage with their commuters. This was never possible before in our market, but with the implementation of a digitalized system for payment, owners can now know their people and adjust their business models accordingly. In development, this feature is completed by allowing the administration to have an account where they can monitor user interaction with their system, have data for each commuter that uses their service and know more about patterns present in travel among the users. These features are applied with an assumption that knowing the client data helps in business growth which has never been done before in the Nepali market.

## 2.3 Similar Projects

3<sup>rd</sup> and 4<sup>th</sup> projects for comparison have been stated in the [appendix section](#).

### 2.3.1 NCMC India



*Figure 2 NCMC Operational Card*

NCMC (National Common Mobility Card) is a card approved by the Ministry of Housing and Urban Affairs under the Government of India which is meant to be used as a payment gateway for all the travel methods in India. Although this project's full operation has been put in a halt by the ongoing pandemic, it was officially launched in 4 March 2019 (TheHindu, 2019) (GadgetsNowBureau, 2019). This card uses RuPay which is a card payment service in India launched by the National Payments Corporation of India (TheEconomicTimes, 2012).

**Platform:** RuPay Payment Gateway

**Technologies:** EMV (card), C,C++,Java

#### Features:

- Tap to pay.
- 'Store balance' on card (VikasPedia, 2019).

### 2.3.2 EZ-Link Singapore



Figure 3 EZ-Link Operational Cards

EZ-Link card is a contactless payment card for public transportation in Singapore. EZ-Link has been in operation since 2002 and has issued more than 40 million cards till date (EZ-Link, 2020). It is a trailblazer in this industry and has been one of the most successful contactless payment methods around the world. It not only allows users to use card but also has a line of wearables with contactless payment mediums built in.

**Platform:** Android and iOS (mobile application)

**Technologies:** NFC/QR payments(cards), Kotlin, Java, vue.js, React.js and C#.

#### Features:

- Mobile app to load card and track transactions
- Payment medium other than card
- Tap to pay
- Locally issued bank card required to make top-ups

### 2.3.3 Oyster Card London



Figure 4 Oyster Card in Operation (Evening Standard, 2014)

Oyster card is a contactless payment card for public transportation in London. It works in London underground, busses, and the railway as well. It has been in operation since 2003 and has served more than 86million cards till date. Users need to tap in the card before entering and leaving the transport medium to pay using the card.

**Platform:** Encrypted NFC(MiFare Classic Card System), Web Application (Abel, 2014)

**Technologies:** C# (ASAP .net framework)

**Features:**

- Web Application to setup payment
- Tap to Pay
- Top-ups for card

## 2.4 Comparisons

*Table 1 Feature Comparison Between Similar Projects*

Features	My Project	NCMC	EZ-Link	Oyster	Go Card	Bicing
Design	Good	Poor	Good	Fair	Fair	Good
Sign Up	✓	✓	✓	✓	✓	✓
Sign In	✓	✗	✓	✓	✓	✓
Dashboard	✓	✗	✓	✓	✗	✓
Expenditure Tracking	✓	✗	✓	✓	✓	✓
Travel History	✓	✗	✓	✓	✓	✓
Travel Statistics	✓	✗	✗	✗	✗	✓
Top Up	✓	✗	✓	✓	✓	✓

## Chapter 3. Development

### 3.1 Considered Methodologies

#### 3.1.1 Waterfall

Waterfall model is the earliest model to be introduced for software development. The waterfall module works on progression of steps which is based on linear and sequential development of a product (Lewis, 2019). This methodology follows seven steps in a linear manner which results in the final product at the end step.

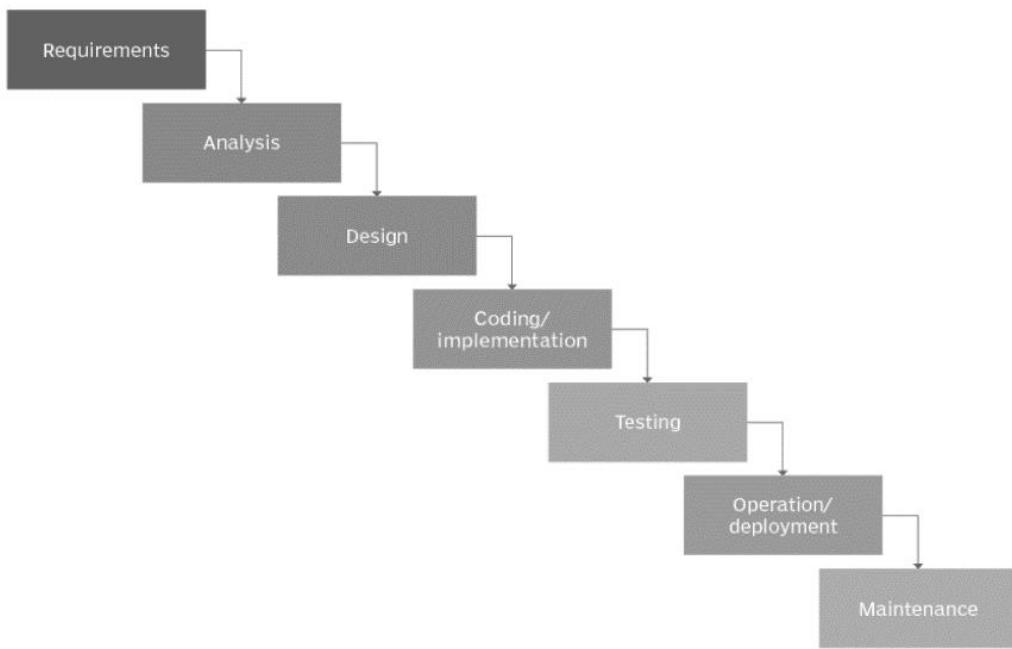


Figure 5 Steps Involved in Waterfall Model (Lewis, 2019)

##### 3.1.1.1 Advantages of Waterfall

- Waterfall module is very basic and simple to understand which fits for all kind of developers.
- Being one of the earliest models in software development, it guarantees the development of a complete product if followed correctly.
- Stages of this model are specifically separated that helps define milestones and deadlines.

### 3.1.1.2 Disadvantages of Waterfall

- Waterfall does not support the idea of revisions which in turn makes the development to restart if a flaw is found during any stage.
- This model does not lean with the idea of requirements changing during the development which is a very common scenario in real life development.
- Testing is not done until the end of the lifecycle.
- A working product cannot be obtained until the end of the lifecycle.

### 3.1.1.3 Justification for Not Choosing Waterfall

*Table 2 Justification 1 for Waterfall*

Case Study Scenario	Complete development of my system comprises of three major aspects, i.e. web, mobile and hardware which need to communicate with each other.
Features	Embedded Testing
Justification	Starting to test the communication of these three aspects of the projects only at the end of the lifecycle could bring out lot of errors that cannot be addressed.

*Table 3 Justification 2 for Waterfall*

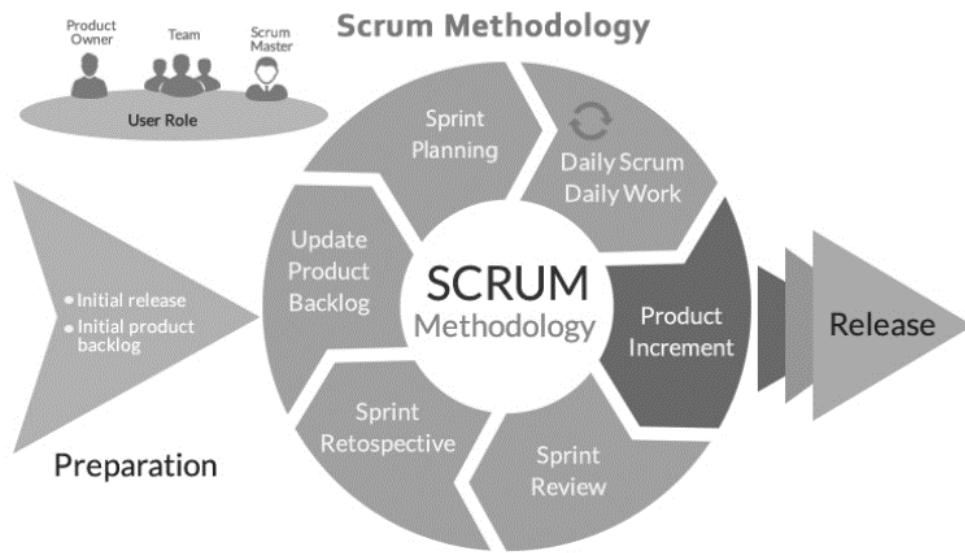
Case Study Scenario	Requirements for a project that involves payment may change during development.
Features	Requirement Reassessment
Justification	During the development of this project, chances of requirements changing are very high due to the nature of the project which is not assessed by this development model.

*Table 4 Justification 3 for Waterfall*

Case Study Scenario	Communication between all the aspects of the project is complex and vital.
Features	Incremental Iterative Development
Justification	In a project consisting of components communicating with each other, incremental updates are required to prevent fundamental breakdown of the project, which is not possible without iterations which this model lacks.

### 3.1.2 Scrum

Scrum is a model for software development based on agile methodology. While the idea of continuous improvement in the system which is the core of agile methodology, scrum model focuses on establishing an idea, implementing the idea, and adjusting the work based on the reflection of the current work done. Development based on scrum works based on sprint cycles where each sprint can be one week to four weeks where development is done based on a goal presented in sprint planning (Agile Alliance, 2020).



*Figure 6 Basic View of Scrum Methodology (Laurance, 2020)*

#### 3.1.2.1 Advantages of Scrum

- Scrum is a very cost effective model for developing a product when used correctly.
- Each goal for a sprint cycle is clearly defined which makes it easy for the team to work with a clear goal.

#### 3.1.2.2 Disadvantages of Scrum

- Although scrum can be modified for a solo project, it is not designed to be implemented by a solo developer.
- This model does not reinforce a deadline which makes the certainty of completion of the project within a time frame to be vague.
- Unless there is a good team to work with, sprint cycles can have a risk of not adding much useful to the final product.

### 3.1.2.3 Justification for Not Choosing Scrum

*Table 5 Justification 1 for Scrum*

Case Study Scenario	This project is being developed by a single person and in its entirety of development, the developer is not allowed to take external help.
Features	Team Size Evaluation
Justification	Scrum is more suited for a team development. Time required for separation of team roles for an individual takes a lot of time which could be used elsewhere.

*Table 6 Justification 2 for Scrum*

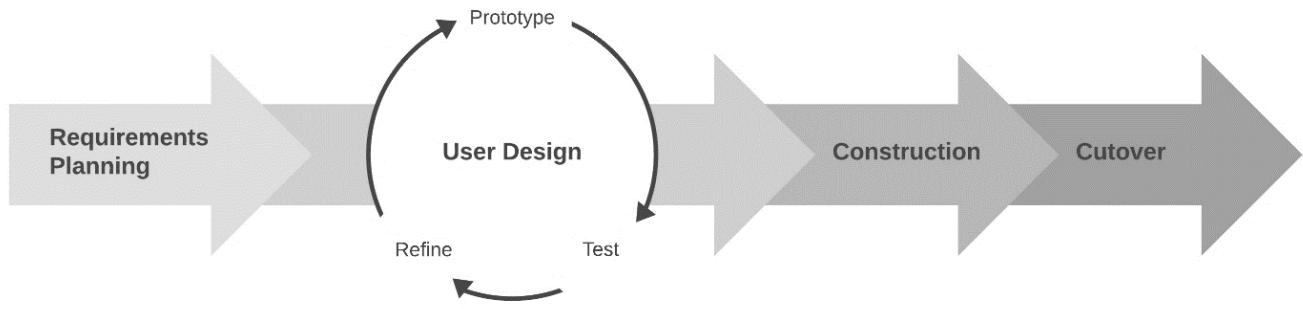
Case Study Scenario	My project has many small features that need a very small iteration period although two small iterations need to be worked on concurrently.
Features	Extensive Sprint Cycles
Justification	Sprints for this methodology takes a lot of time even for a simple feature (one to four weeks), which is meant to be worked on by a team. Fitting number of sprint cycles has a high chance of exceeding the project deadline.

*Table 7 Justification 3 for Scrum*

Case Study Scenario	This project is not meant to start a business or make a profit for stakeholders which are non-existent in this project.
Features	Business Modelling / Stake Holder Communication
Justification	This methodology has a lot of emphasis on stakeholders and client. Due to the lack of clients in my project, there is no compelling reason to simulate myself as a stake holder, scrum master and the scrum team.

### 3.1.3 Rapid Application Development (RAD)

Developed by James Martin in 1991, Rapid Application Development (RAD) falls under agile development methodology, which means that it supports the change of requirements with incremental approach, which in this model, is completed using prototypes (Singh, 2019) (O'Carroll, 2020). Development of prototypes is done by creating an initial prototype and adding features to the prototype according to testing and user feedbacks until the final project is achieved.



*Figure 7 Four Phases of Rapid Application Development (Lucidchart, 2018)*

Testing is an integral part of this model which helps in eliminating all errors before the final system is presented.

#### 3.1.3.1 Advantages of RAD

- Change of requirements can be assessed in the next version of the prototype.
- The end product is very likely to be welcomed by users as it is developed under the feedbacks of users themselves.
- When used with a small team, development time is shorter compared to other models.

#### 3.1.3.2 Disadvantages of RAD

- It is not suitable for products that need a lot of time to develop.
- The system that needs to be developed must be very modular since each prototype needs to have an incremental update.
- It constantly needs user requirements throughout its development period.

### 3.1.3.3 Justification for Not Choosing RAD

*Table 8 Justification 1 for RAD*

Case Study Scenario	Usage of hardware of this project does not require prototyping as requirement for hardware has already been met and unlikely to change.
Features	Prototyping
Justification	Although time between iterations/prototypes are very short and quick, the development of multiple prototypes might cause the project cost to go out my budget.

*Table 9 Justification 2 for RAD*

Case Study Scenario	There are no users to constantly give feedbacks during the development of the project.
Features	Constant User Communication
Justification	Lack of users for feedback during the development makes this model unusable as its core development relies on user feedback.

*Table 10 Justification 3 for RAD*

Case Study Scenario	This is a complex project meant to be completed within five to six months period after the start of development
Features	Project Time Evaluation
Justification	RAD is suitable for projects that intend to have a small development time which can put out a lot of prototypes. This project's time for this model is way over its efficiency for development.

### 3.2 Selected Methodology

#### 3.2.1 Rational Unified Process

This is a software development process from Rational which is a division of tech giant IBM (TechTerms, 2005).

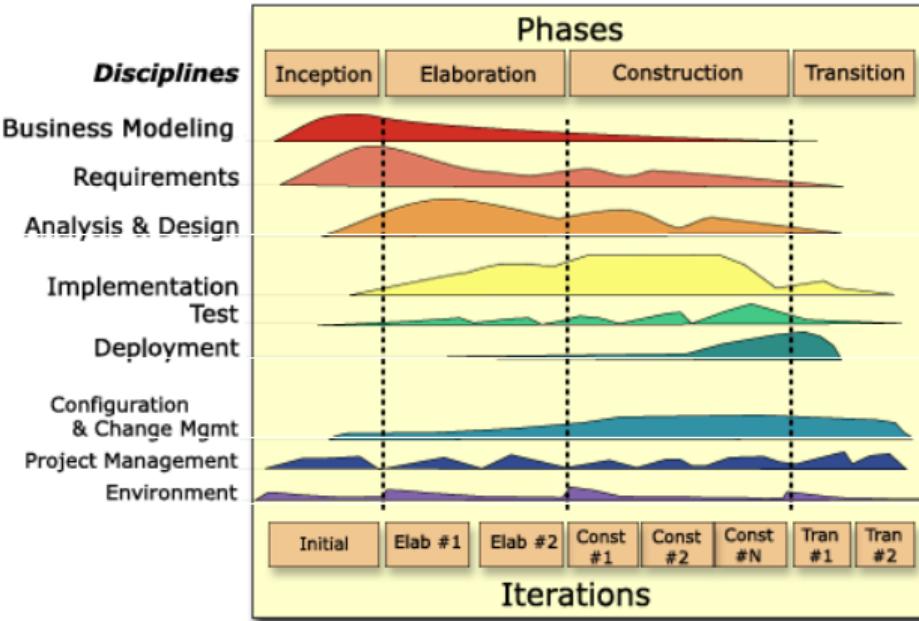


Figure 8 Phases of RUP (RationalSoftwareCorporation, 2002)

While developing the system, all the three aspects of my system; web, mobile and hardware need to be developed simultaneously which is possible using this methodology by overlapping two iterations of different features at the same time. This allows both the construction and testing of two or more features at once which is vital for this project as there is a lot of communication going on between all the three aspects.

##### 3.2.1.1 Justification for Using RUP

Table 11 Justification 1 for RUP

Case Study Scenario	The cost for the hardware purchase and deployment need to be known know the tentative budget.
Features	Cost Calculation
Justification	This methodology encourages to calculate cost way ahead of development which saves the project from going over the budget. Since this project does not intend to return any monetary value, staying under the budget is very important.

*Table 12 Justification 2 for RUP*

Case Study Scenario	In the project, there are a lot of features that can be changed over the time, this in turn can completely change the requirement for another related feature.
Features	Requirement Reassessment
Justification	In each iteration, the requirement for the feature needs to be looked at again. Although the time for reassessment needs to go lower with each iteration, any big changes in requirements can be easily identified during the development.

*Table 13 Justification 3 for RUP*

Case Study Scenario	There are a lot of features spread across the system that need to work with each other even during development for the completion of the system.
Features	Iterative and Incremental Development
Justification	With the idea of making a feature more robust with each iteration, having a model that adds more to a feature in each iteration helps it to communicate with any other feature in the system that relies on it.

*Table 14 Justification 4 for RUP*

Case Study Scenario	With a feature added to the system, it needs to be tested rigorously to prevent failure with the addition of a new feature.
Features	Embedded Testing
Justification	In this model, with each iteration, there needs to be testing done to the feature added in that iteration. Communication of a feature with other components of the system also needs to be tested after every construction phase.

*Table 15 Justification 5 for RUP*

Case Study Scenario	Some features that need to communicate through the web need to be deployed during construction to make the development of dependent features possible.
Features	Continuous Deployment
Justification	Unlike other models that deploy the complete system at the end of the development lifecycle, this model allows to ship the system with smaller features added to the system every time. This shipment or deployment can be done at the end of each iteration.

### 3.3 Phases of Methodology

#### 3.3.1 Inception

This is the first stage of this methodology where the idea of project is initiated. It mainly focuses on analyzing the requirements for the projects and the estimation of resources that come with the requirements. Tasks involved in inception are,

- Project Scope Identification
- Research Development Tools
- Planning Business Model Overview

#### 3.3.2 Elaboration

This second stage of this methodology further analyses the requirements that were assumed in the first stage of the development. Furthermore, architecture for development and planning with designing of the project is done in this stage of development. Tasks involved in elaboration phase are,

- Set up Development Tools
- System Design (UML diagrams, wireframes, prototypes)
- Iteration Planning

#### 3.3.3 Construction

This is the penultimate stage of the development methodology. As its name suggests, this stage includes the main development of the project. The outcome of this development should be ready to use and deployed to the required destination. This involves coding, documenting, and testing of the product. Tasks in construction are,

- System Development
- Incremental Development Testing
- Resource Management

### 3.3.4 Transition

This is the ultimate stage of this methodology which involves in the deployment of the product that comes out of the construction phase. Installation in case of software and teaching the users how to use the product falls under this category which is one of the simplest yet vital part of this methodology. Tasks involved in transition phase are,

- System Testing
- Unit Testing
- Deployment
- User Feedback

### 3.4 Survey Results

For the development of the application, and for the feedback of the developed application, surveys from commuters of varying demographic were conducted for requirement analysis and product feedback. Gist of the feedbacks from the participants have been documented in the upcoming section.

Pre survey has sample space of 72 participants whereas post survey form has sample space of 7 participants where all the participants of post survey are a subset of pre survey.

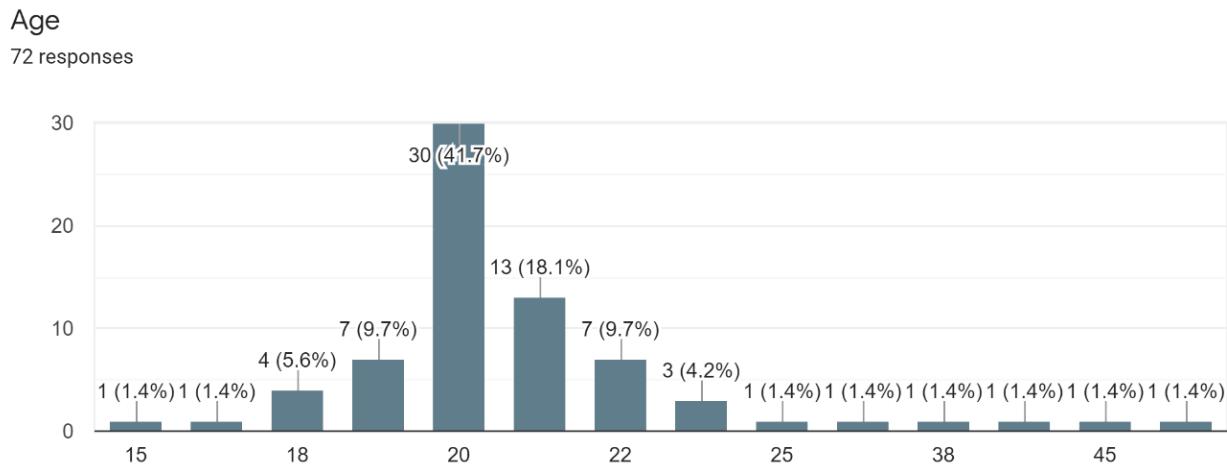


Figure 9 Age Demographic of Responders

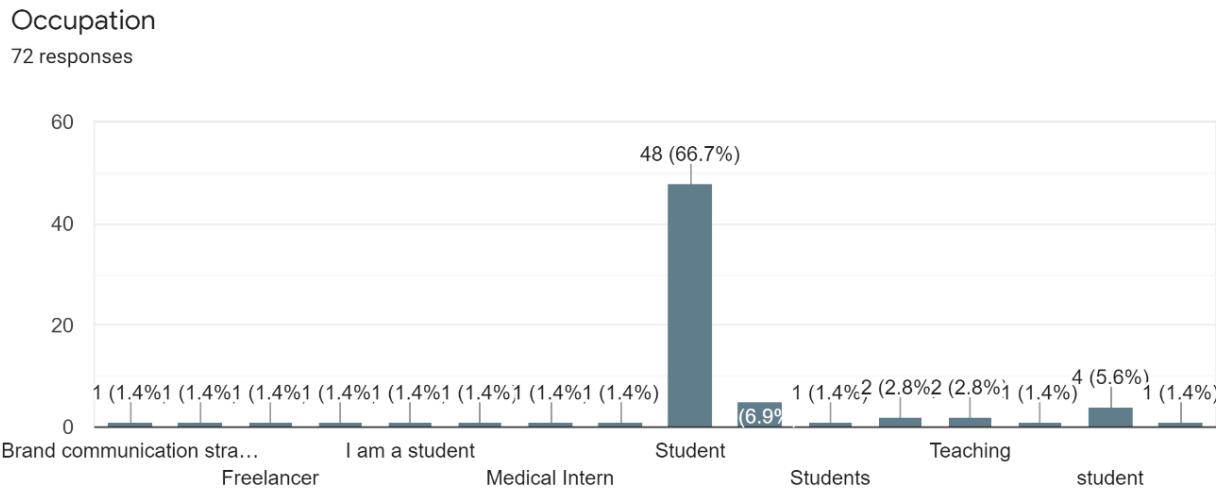


Figure 10 Occupation of Responders

### 3.4.1 Pre-Survey Results

- a) Majority of participants find the existing payment methods in transportation to be average or worse.
- b) Almost all of the participants believe that the addition of contactless payment system to public transportation would increase the quality of travel.
- c) Adding discount details without the need of showing institution issued identity card is acceptable for 88% of the participants, remaining percentage does not have a definitive answer.
- d) 97.2% of the responses show positive feedback for viewing their expenditure and travel statistics.
- e) Most of the people would not want to come in contact with a bus staff to tap their cards.

[Results of pre survey have been provided in the appendix section.](#)

### 3.4.2 Post-Survey Results

- a) Feedbacks for features suggests that users would want a 'view password' feature during login and signup.
- b) Majority of users do not see the need of having bus staff (conductors) in the vehicle if the system is implemented.
- c) There is a split decision if there needs to be a one to one replication of feature on mobile devices, this result could change with the increase of sample size.
- d) More than half of the responses believe that this payment system on public transportation could increase the quality of travel.

[Results of post survey have been provided in the appendix section.](#)

### 3.5 Requirement Analysis

These are the requirements required to develop and run the application.

#### 3.5.1 Feature Requirements

Features for this system are completely based on the solutions that have been mentioned for the given problem statements. These solutions can be implemented in the system by concealing them as features for the application. The feature requirements for the system are,

- a) Register User
- b) Login User
- c) Manage Data/Profile
- d) Write Card
- e) Read Card
- f) Payment
- g) View Data

Detailed description of all the use cases generated by the functional requirements have been explained in the [appendix section](#).

### 3.5.2 Software Requirements

Detailed description of the requirements has been described in SRS document.

#### 3.5.2.1 Planning and Development Tracking

- a) Notion
- b) Instagantt
- c) Asana

These applications were used to plan the development timeline along with making Gantt charts. The activity progress was then tracked along with development to generate project artefacts.

#### 3.5.2.2 IDE

- a) Visual Studio Code
- b) Thonny Python Editor

Visual Studio Code was used to edit code on windows OS whereas the latter one was used to edit python on raspberry pi.

#### 3.5.2.3 Web Application

- a) Python and Django
- b) Django-Rest
- c) HTML/CSS (bootstrap)
- d) JavaScript

#### 3.5.2.4 Mobile Application

- a) Flutter for front end
- b) Django for backend

#### 3.5.2.5 Desktop Application

- a) Python
- b) Tkinter Python

### 3.5.2.6 Web Plugins / SDK

asgiref==3.3.0 authy==2.2.6 cachetools==4.2.1 certifi==2020.12.5 cffi==1.14.5 chardet==4.0.0 dj-database-url==0.5.0 Django==3.1.3 django-filter==2.4.0 django-phonenumber-field==5.0.0 django-rest==0.8.6 django-storages==1.11.1 django-widget-tweaks==1.4.8 djangorestframework==3.1.2.2 djangorestframework-api-key==2.0.0 gunicorn==20.0.4	idna==2.10 packaging==20.9 phonenumbers==8.12.13 Pillow==8.0.1 protobuf==3.15.1 psycopg2==2.8.6 pyasn1==0.4.8 pyasn1-modules==0.2.8 pycparser==2.20 PyJWT==2.0.0 pyparsing==2.4.7 pytz==2020.4 requests==2.25.1 rsa==4.7.1 six==1.15.0 sqlparse==0.4.1 twilio==6.50.1 urllib3==1.26.2 whitenoise==5.2.0	google-api-core==1.26.0 google-auth==1.27.0 google-cloud-core==1.6.0 google-cloud-storage==1.36.0 google-crc32c==1.1.2 google-resumable-media==1.2.0 googleapis-common-protos==1.52.0
---	---	---

Figure 11 Required Plugins in Web Application

### 3.5.2.7 Mobile Plugins

- /cupertino\_icons: ^1.0.0
- /intl\_phone\_field: ^1.4.2
- /http: ^0.13.1
- /flutter\_secure\_storage: ^4.1.0
- /flutter\_screenutil: ^4.0.3+1

### 3.5.3 Hardware Requirements

Hardware required to make and program the card write and read functionalities have been listed below.

- [Raspberry Pi](#) to act as the brains of the operation which uses the sensors accordingly and store the data to the database.
- RFID cards to store the user ID which will be used for reading.
- [RFID module](#) to read the data present in the RFID cards to identify the user.
- [GPS module](#) to collect the position of the vehicles to identify where the user enters\_or exits the vehicle.
- Components to connect these modules (breadboard, wires, jumper wires) etc.
- Hardware OS: Raspbian
- Local machine to develop the application.

### 3.6 Design

Before constructing the application, designing all of the workflow for development is necessary. This helps in understanding features before development starts, which makes the development process easier. To completely understand the project, high level design for the whole project, and detailed design for its features have been produced in form of UML diagrams.

#### 3.6.1 Application Logo



*Figure 12 Application Logo*

This logo has been designed to signify 'Card Pay' as the intended title for the product in future. Instead of using generic wording for the whole logo, a logo that can be distinguished from others whilst having a minimal aesthetic has been developed. This design can be used in mobile application, web application and even in printing, due to its minimal nature.

### 3.6.2 System Design

#### 3.6.2.1 Use Case Diagram

Use case diagram is used to visualize the overall features that are required in the system including how the features relate to the people who use the application. People using the application are known as actors whereas the features themselves are known as use cases.

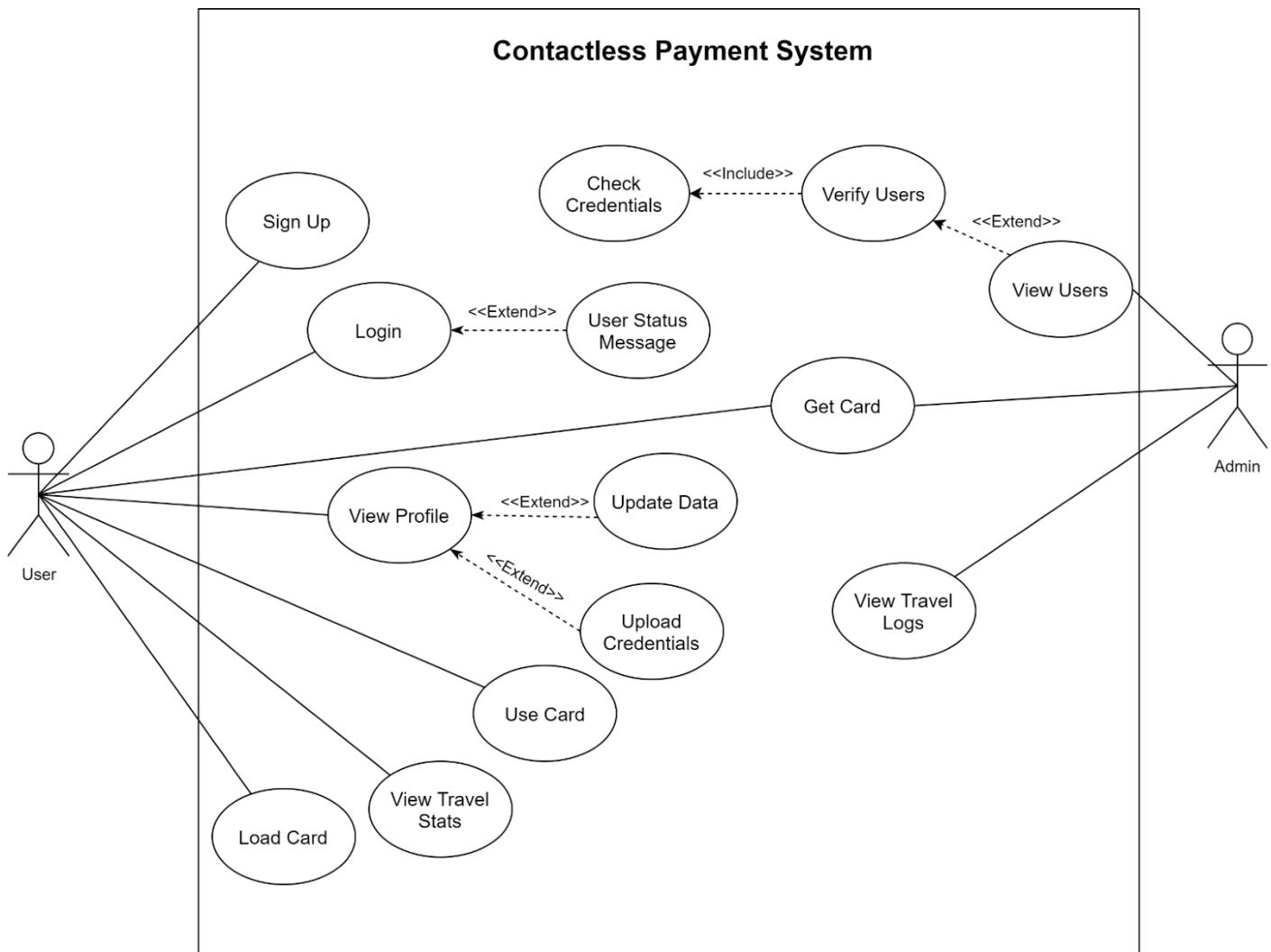


Figure 13 System Use Case

Iterations for the use case diagrams have been provided in the [appendix section](#).

### 3.6.2.2 ER Diagram

To define the structure of the database for the application, an entity relation diagram is required before defining the models of the application.

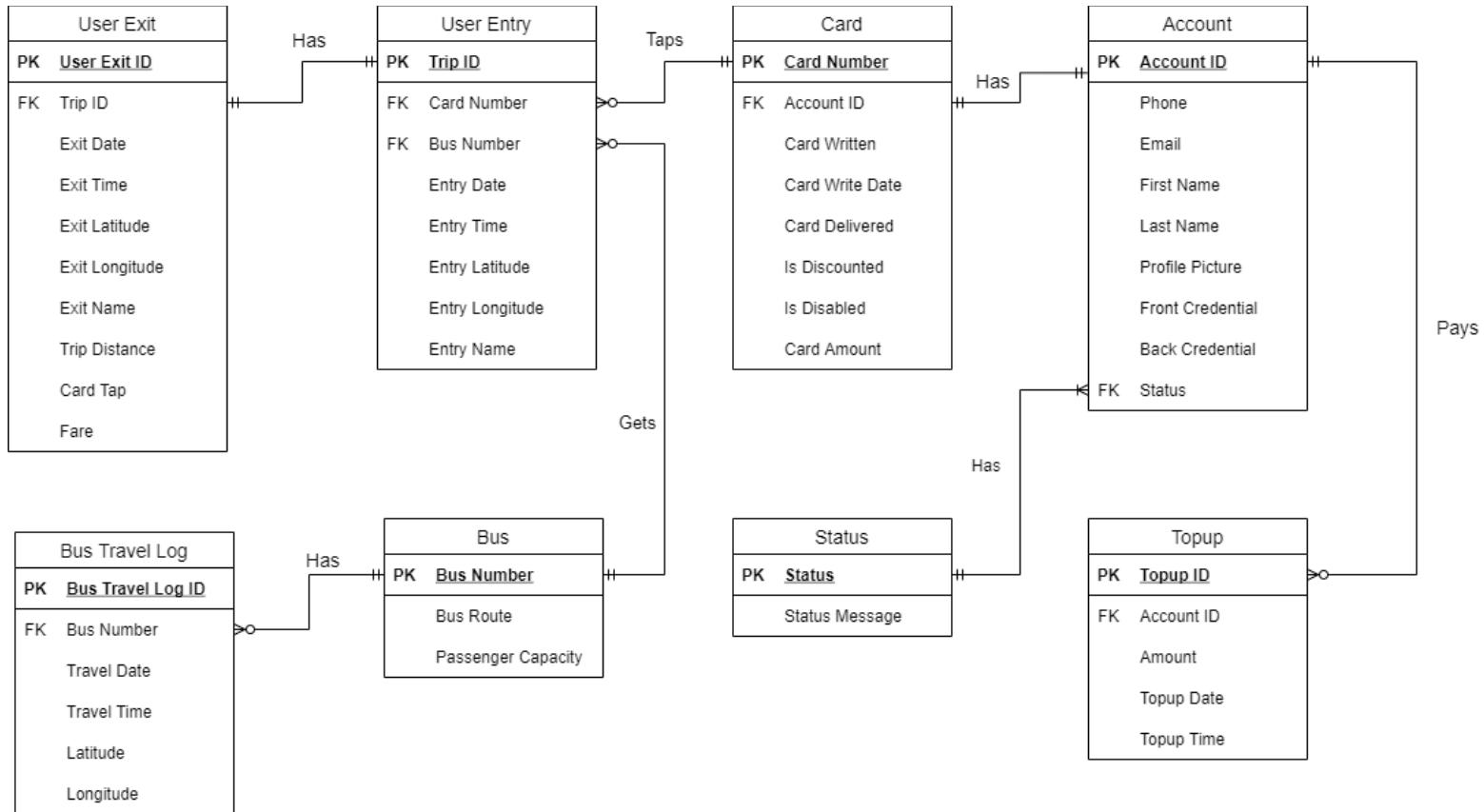


Figure 14 System ER Diagram

Iterations of the ER Diagrams have been given in the [appendix section](#).

### 3.6.2.3 Activity Diagram

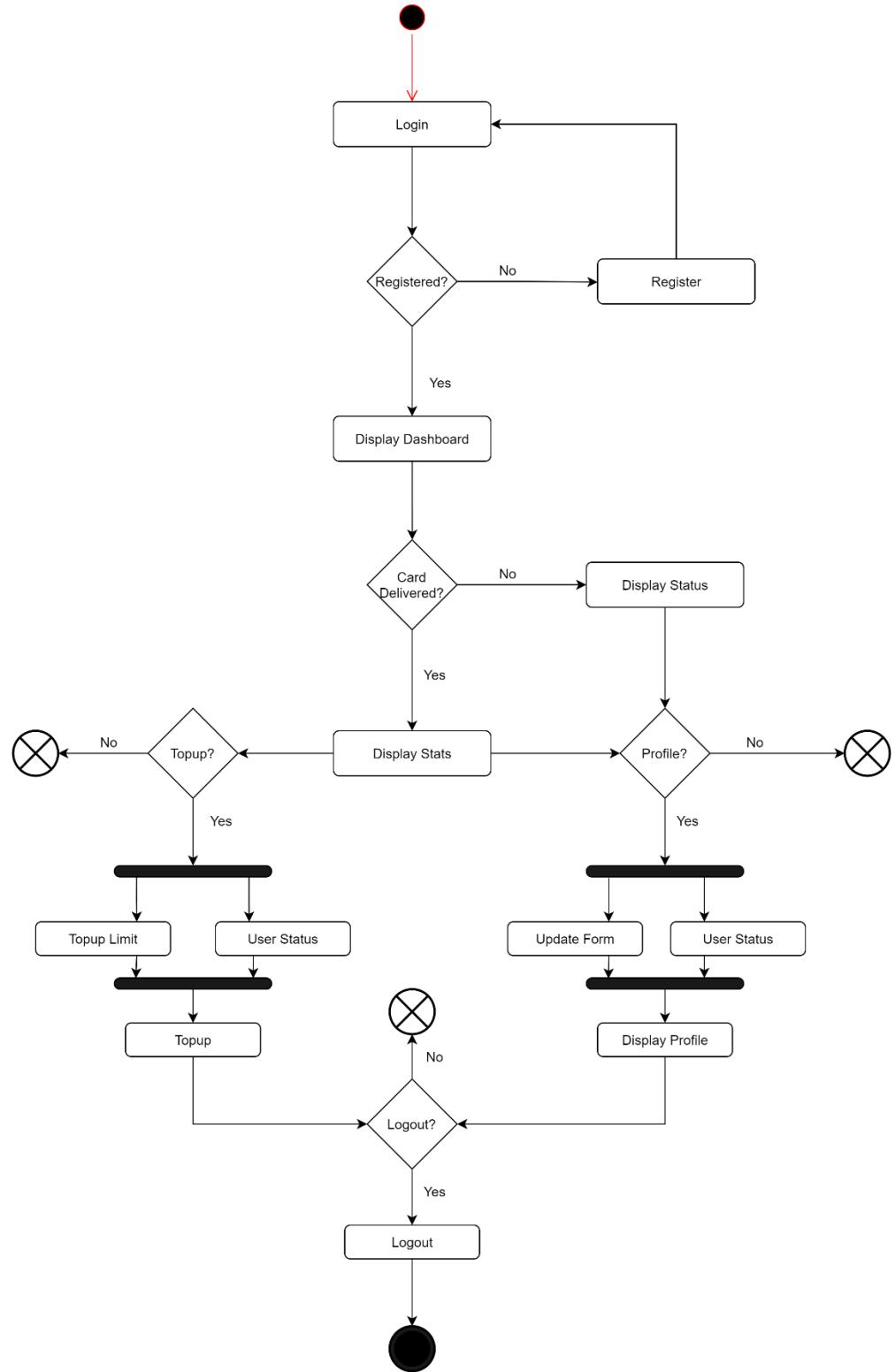


Figure 15 User Web Application Activity Diagram

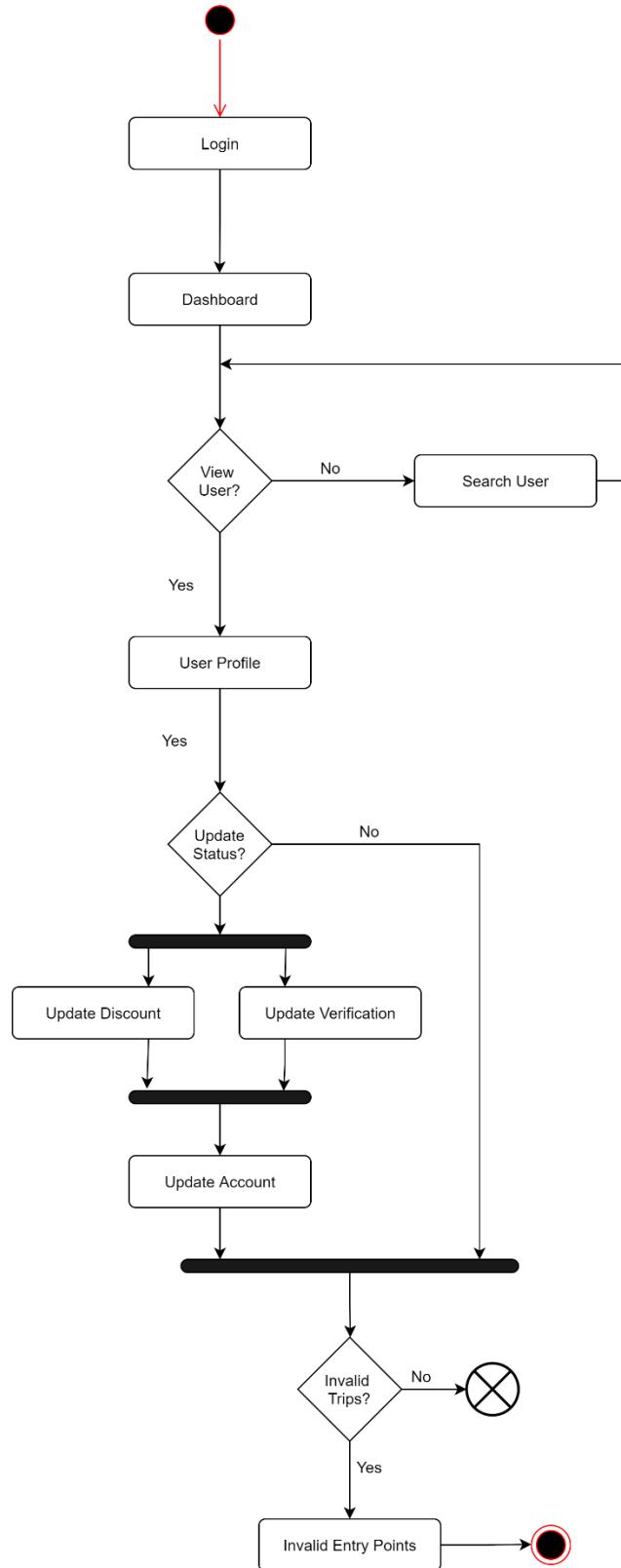


Figure 16 Admin Web Application Activity Diagram

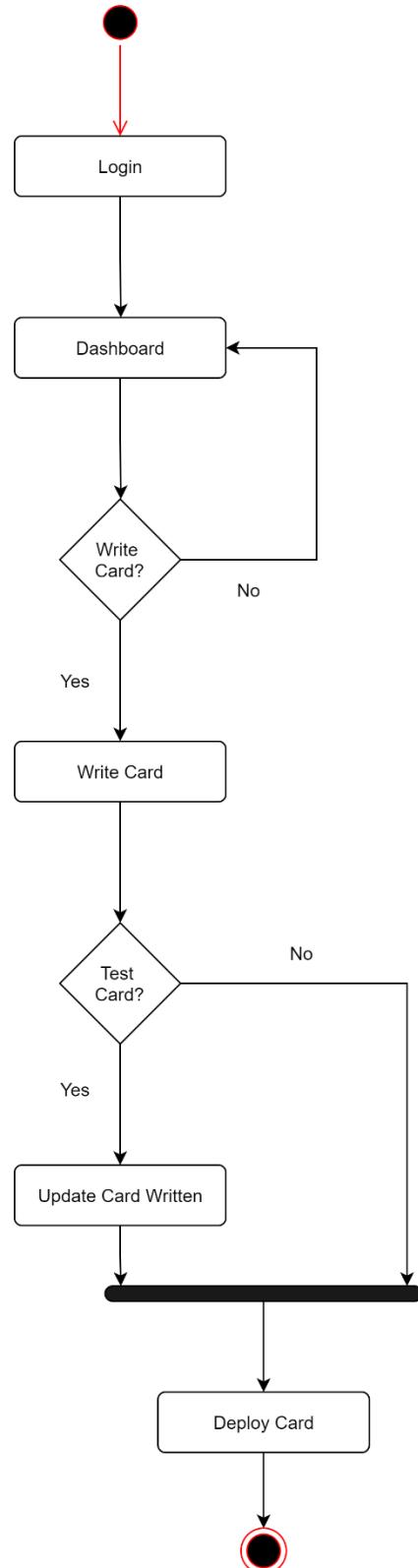


Figure 17 Admin Desktop Application Activity Diagram

### 3.6.3 Feature Design

Design for workings for individual core features of the system have been presented in the upcoming sections.

Level 0 and Level 1 DFD for the overall system have been given in the [appendix section](#).

#### 3.6.3.1 User Signup

First iteration of DFD for user signup is in [appendix section](#).

##### a) Data Flow Diagram

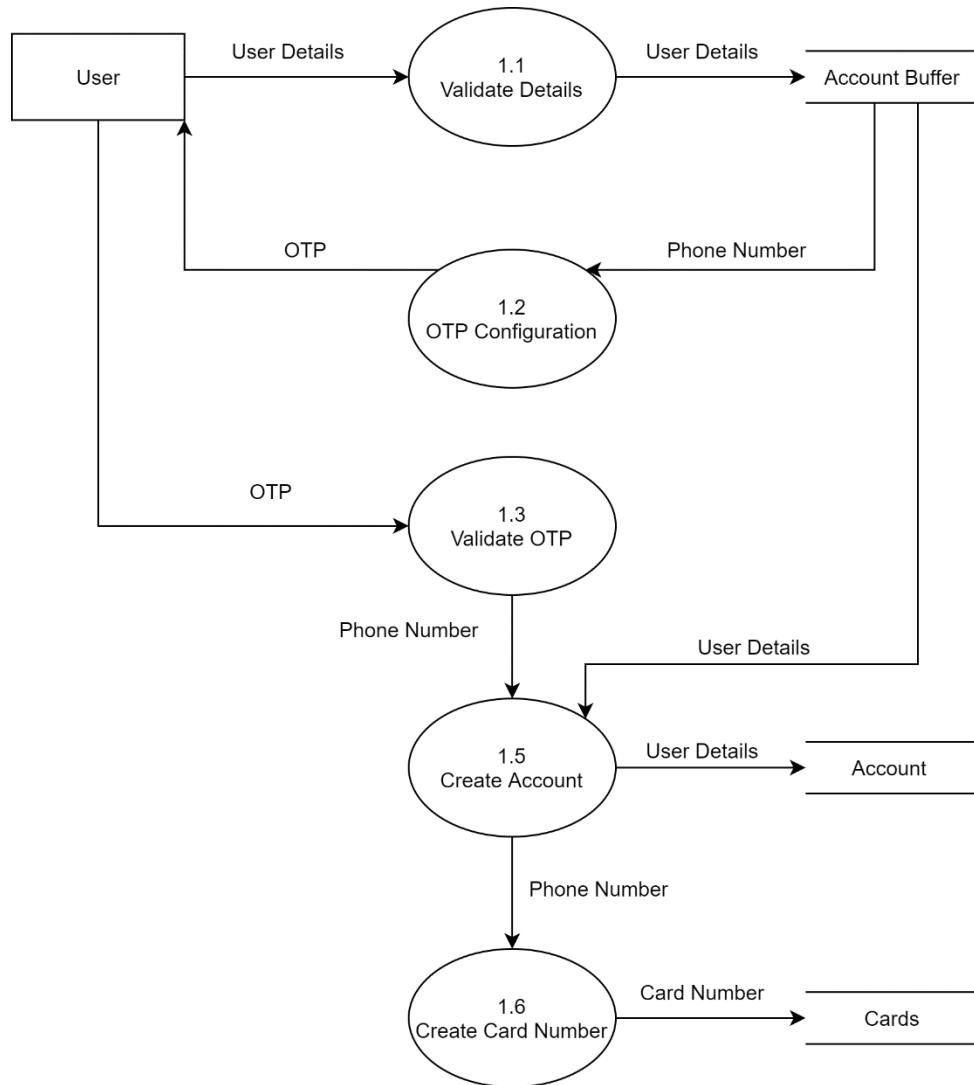


Figure 18 User Signup Level 2 DFD

### b) Communication Diagram

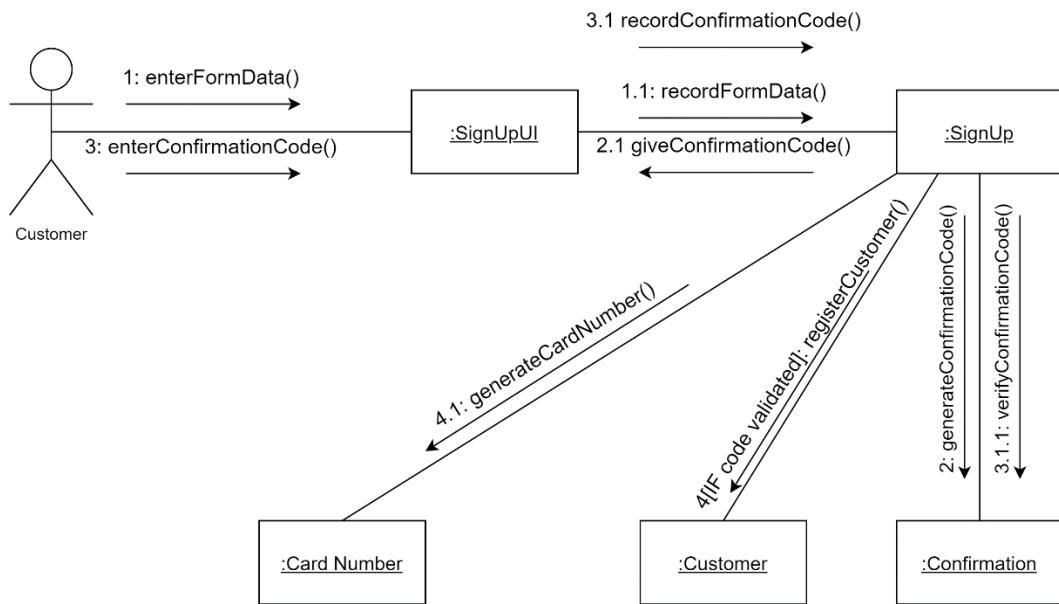


Figure 19 Sign Up User Communication Diagram

### c) Sequence Diagram

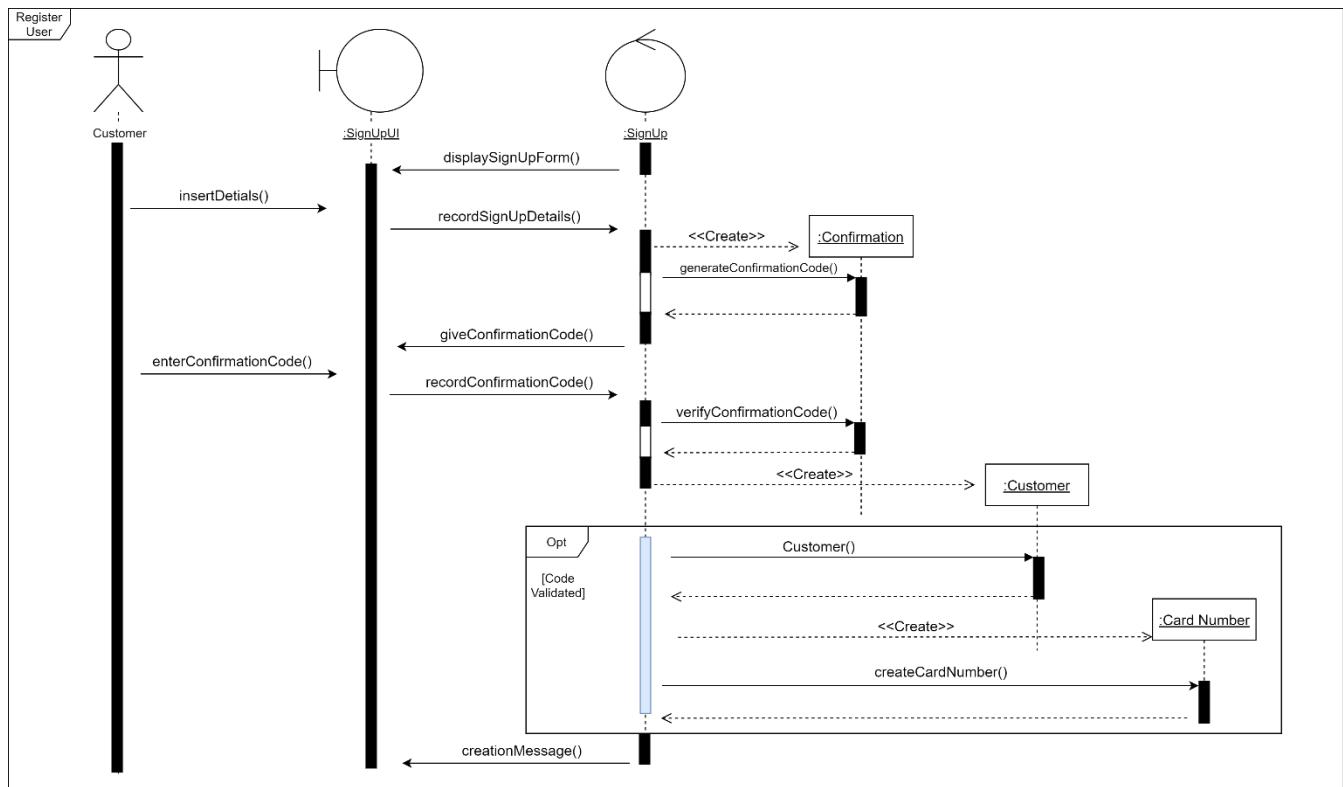


Figure 20 Sign Up User Sequence Diagram

### 3.6.3.2 Login User

#### a) Data Flow Diagram

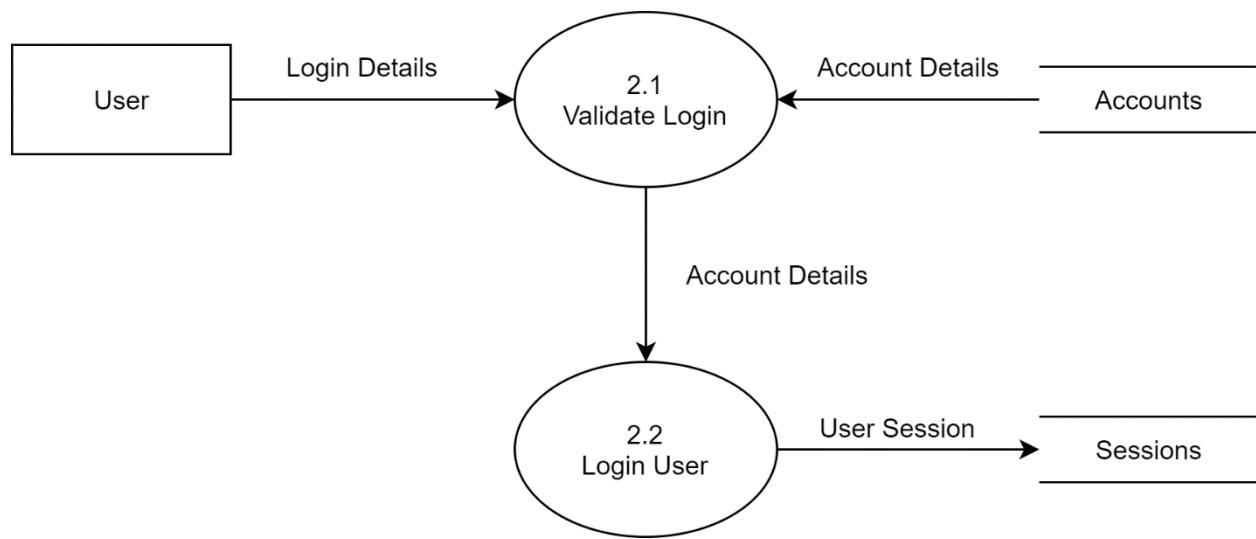


Figure 21 Login User Level 2 DFD

#### b) Communication Diagram

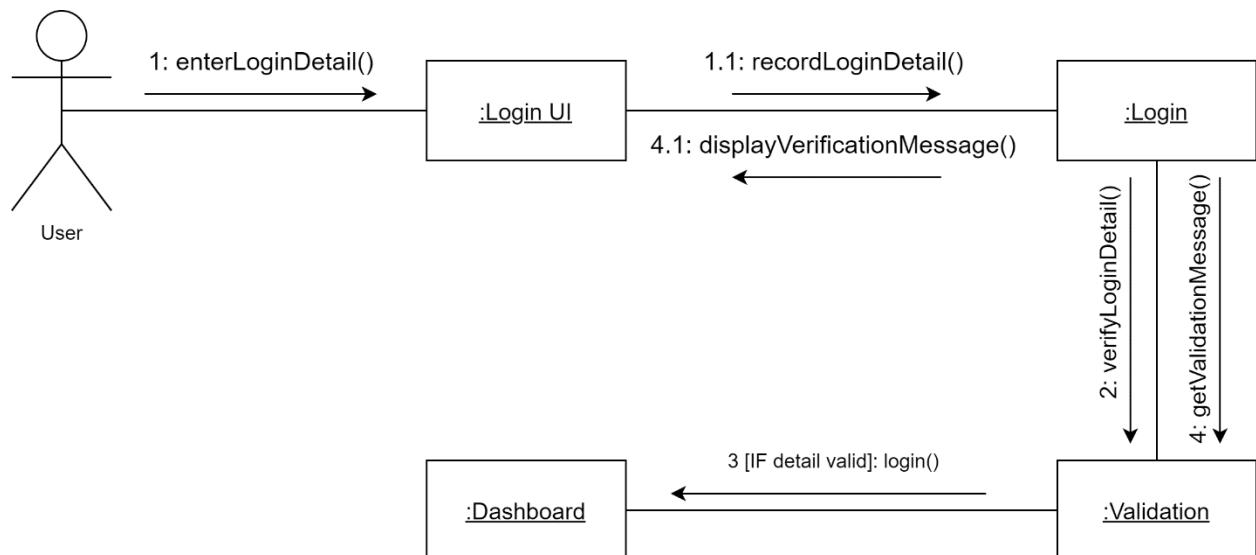


Figure 22 Login User Communication Diagram

## c) Sequence Diagram

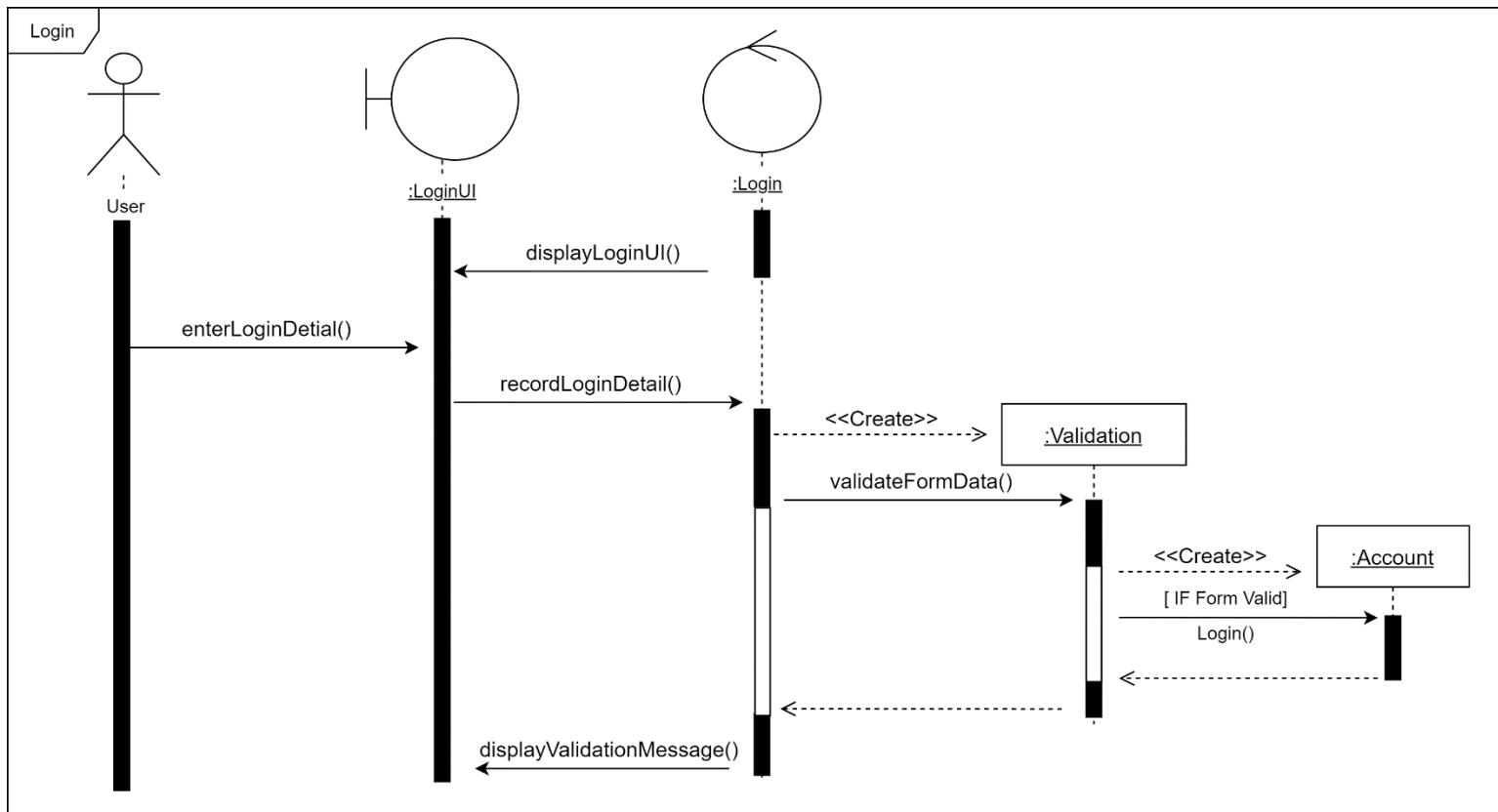


Figure 23 Login User Sequence Diagram

### 3.6.3.3 Manage Profile

#### a) Data Flow Diagram

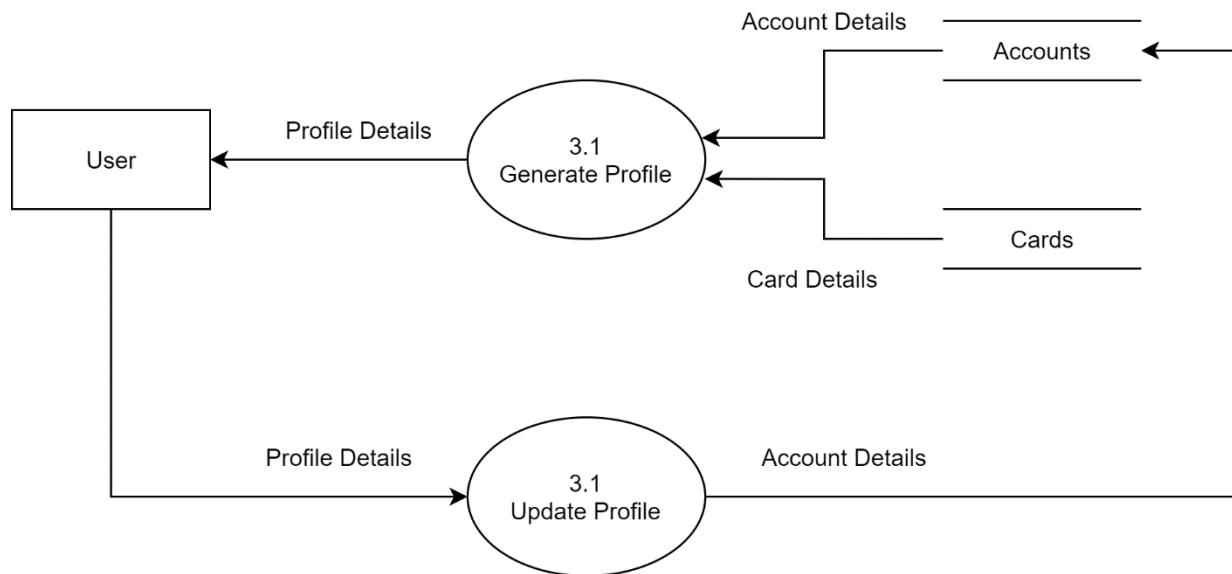


Figure 24 Manage Profile Level 2 DFD

#### b) Communication Diagram

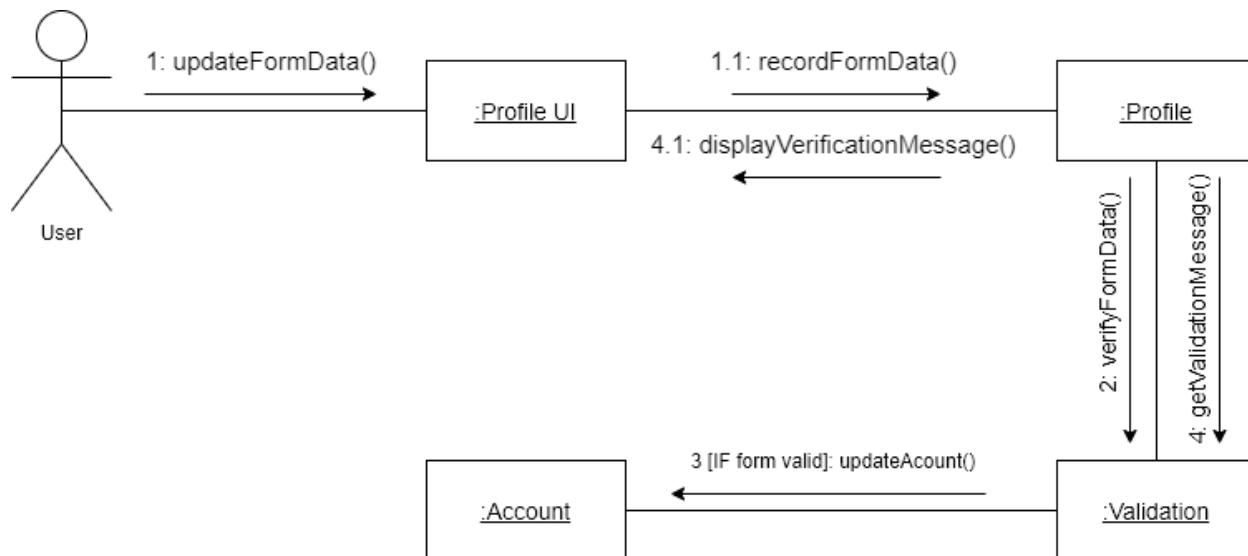


Figure 25 Manage Profile Communication Diagram

## c) Sequence Diagram

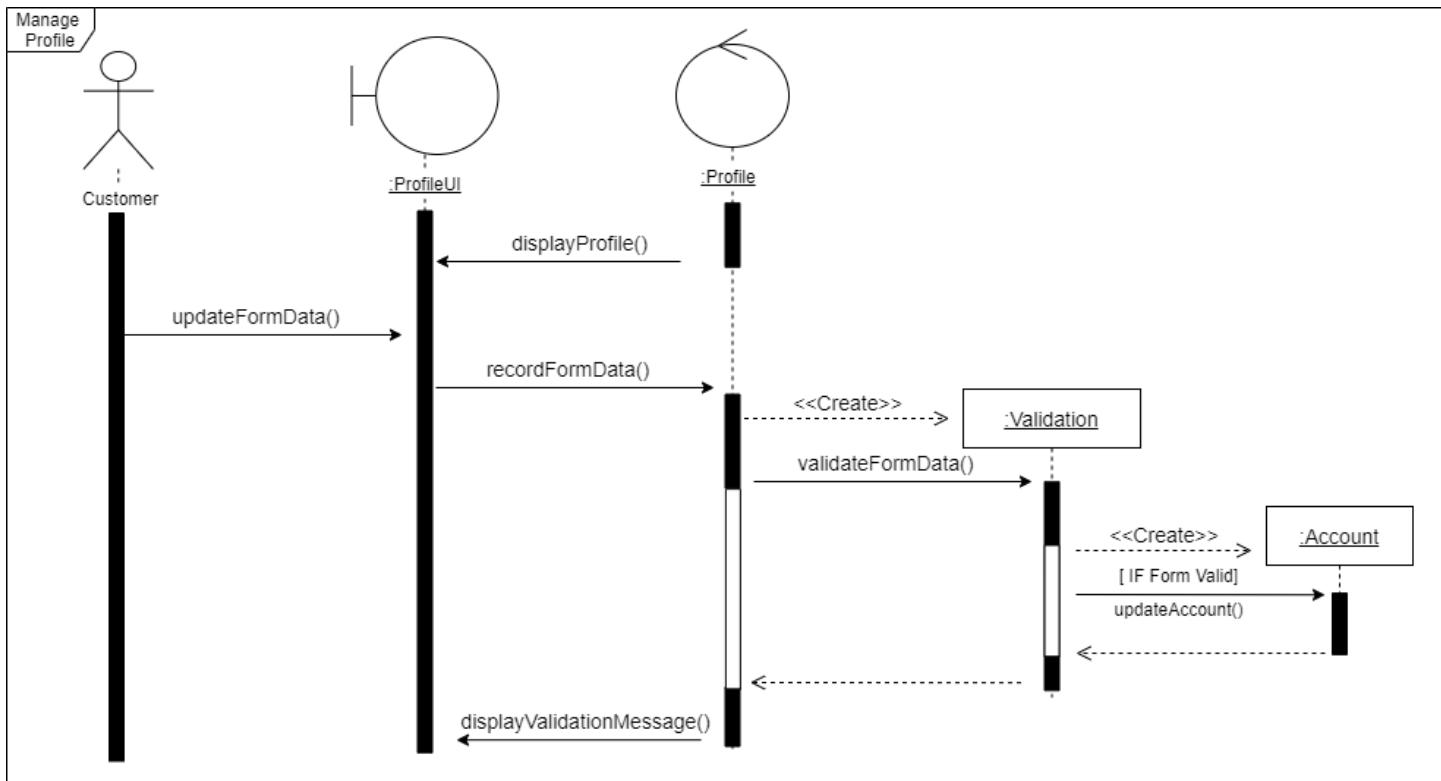


Figure 26 Manage Profile Sequence Diagram

### 3.6.3.4 Write Card

#### a) Data Flow Diagram

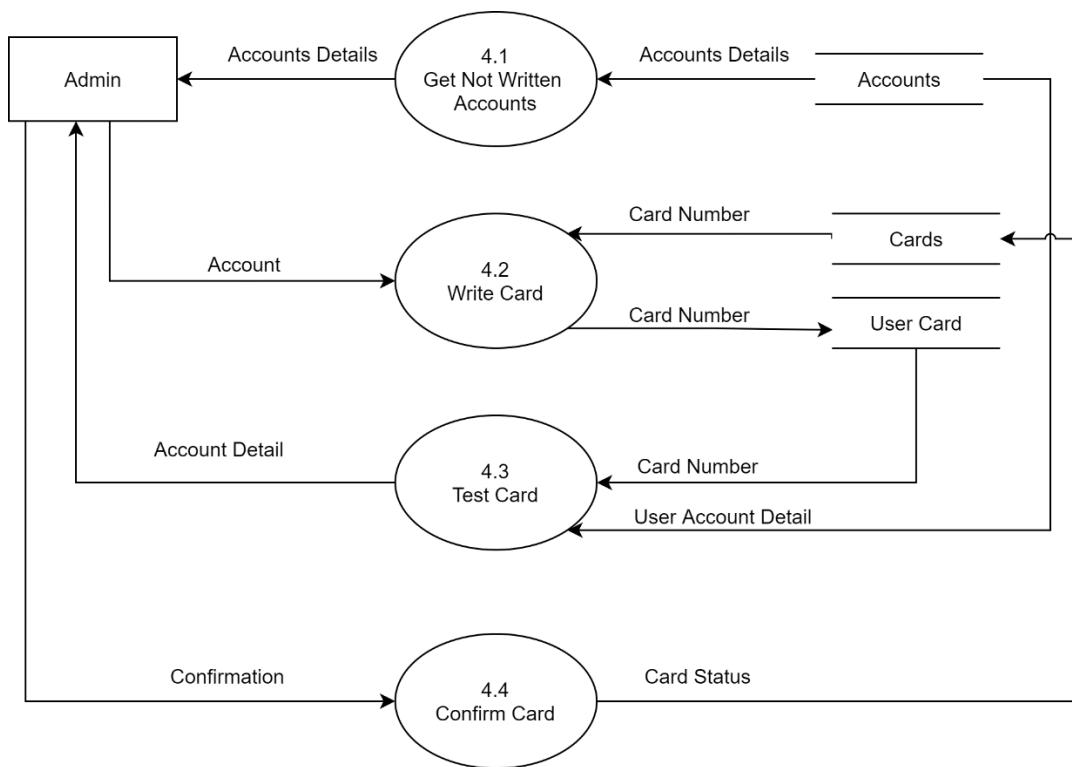


Figure 27 Write Card Level 2 DFD

#### b) Communication Diagram

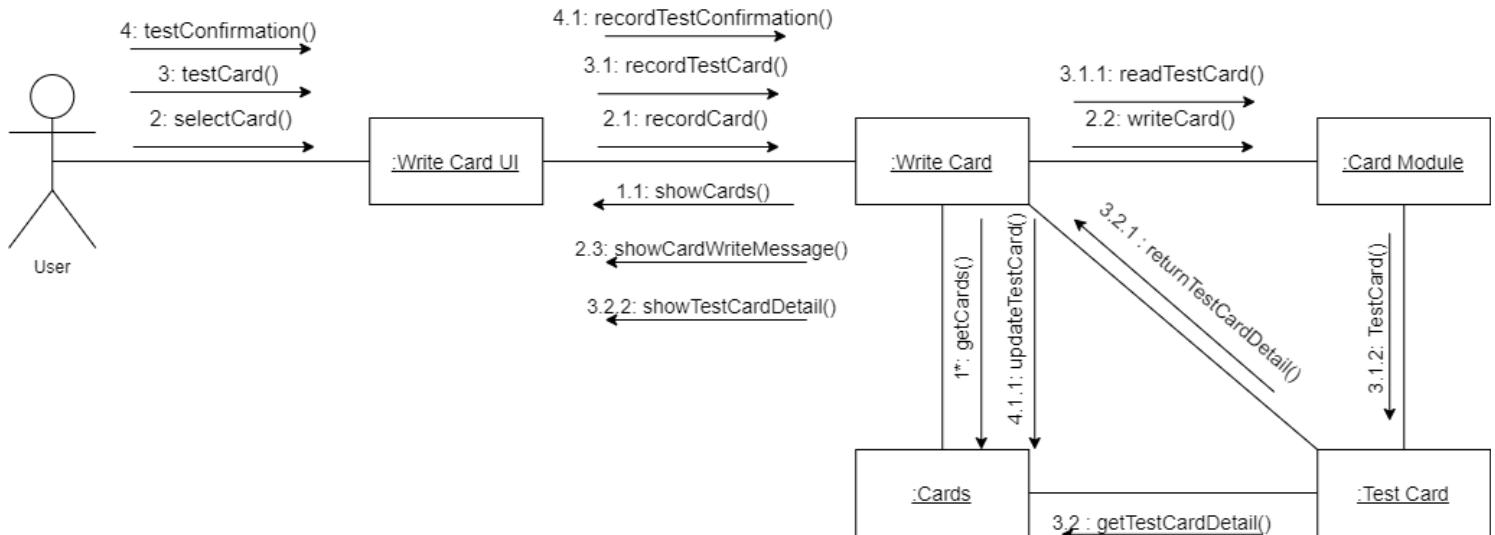


Figure 28 Write Card Communication Diagram

## c) Sequence Diagram

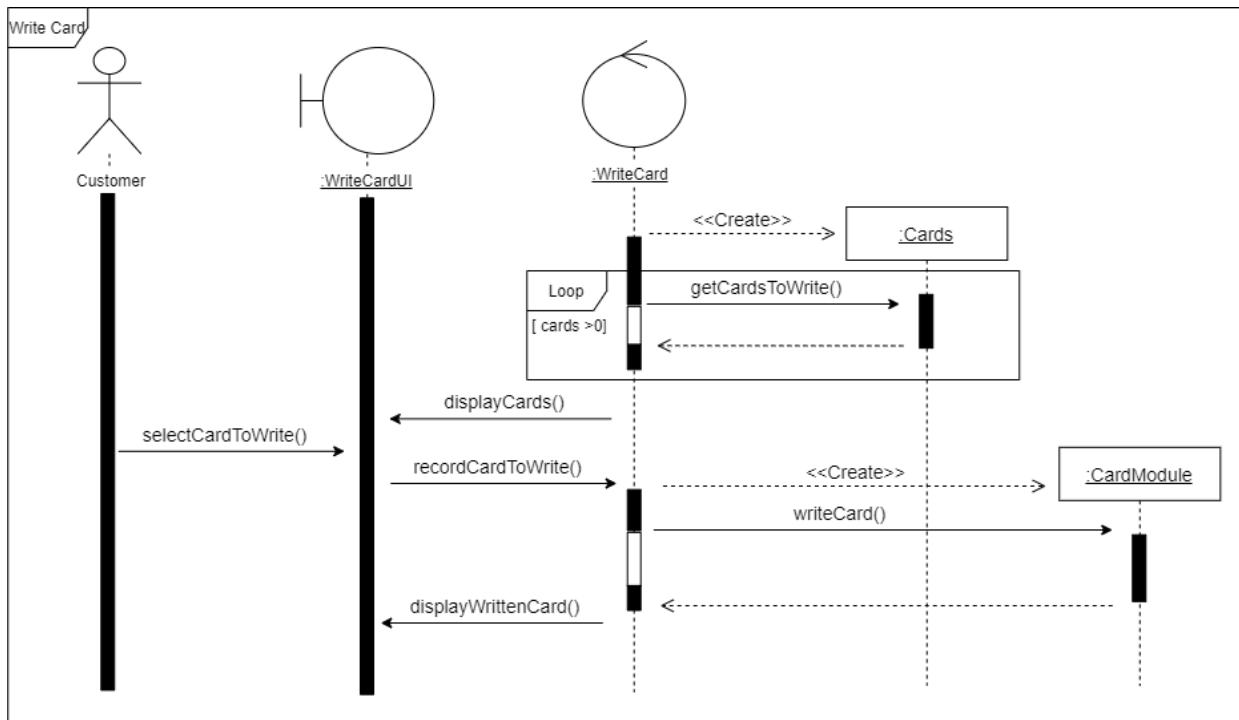


Figure 29 Write Card Sequence Diagram

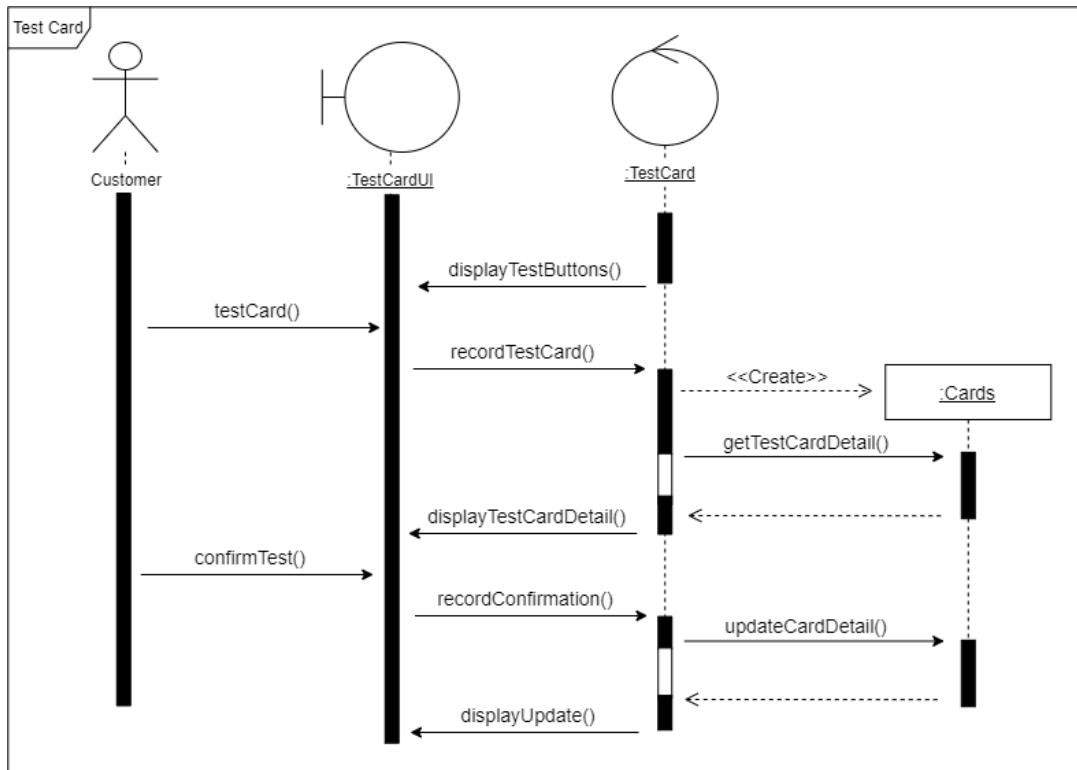


Figure 30 Test Card Sequence Diagram

### 3.6.3.5 Deploy Card

#### a) Data Flow Diagram

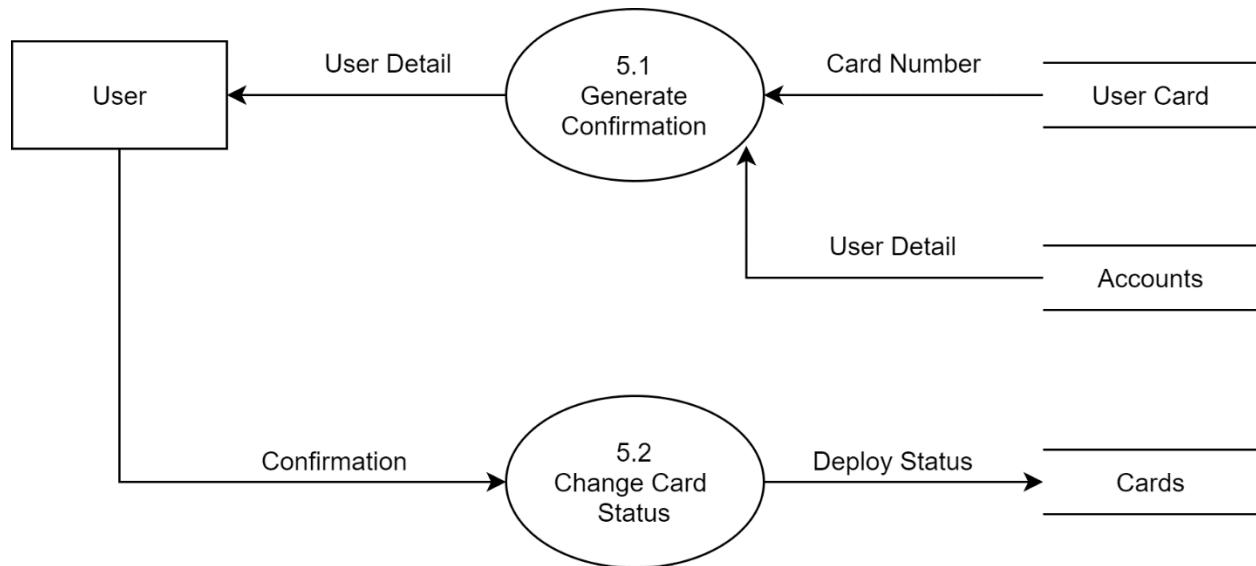


Figure 31 Deploy Card Level 2 DFD

#### b) Communication Diagram

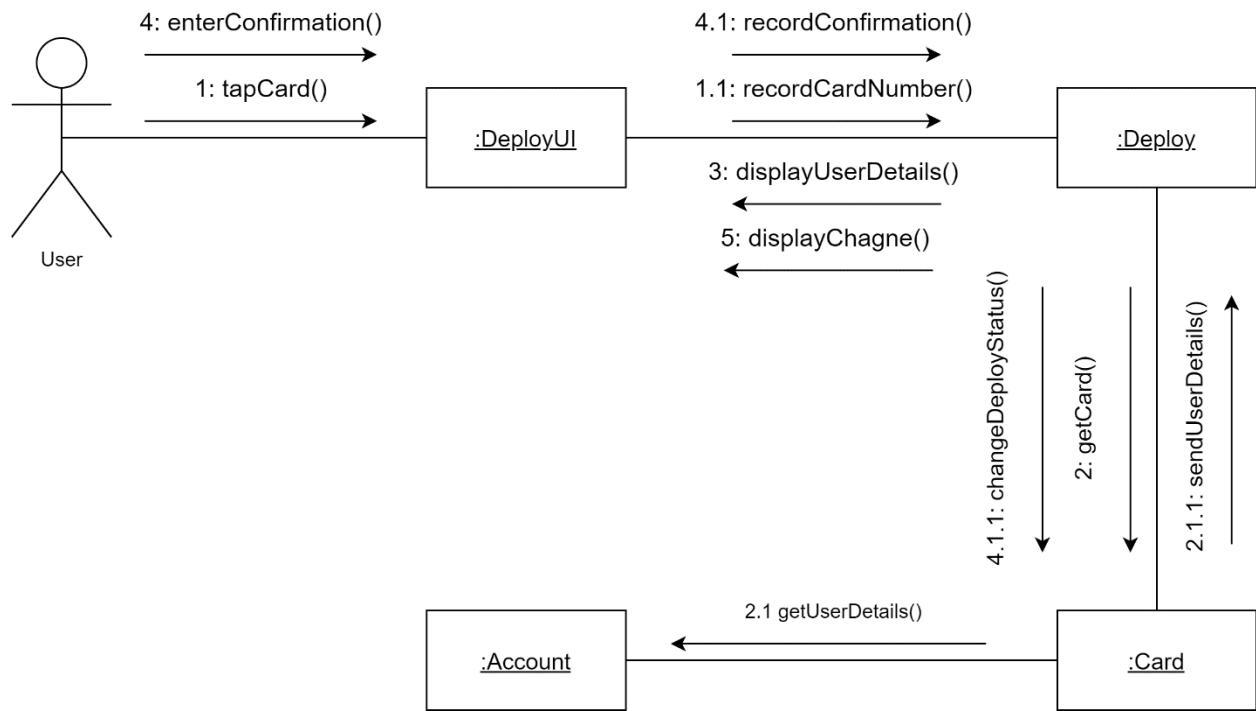


Figure 32 Deploy Card Communication Diagram

## c) Sequence Diagram

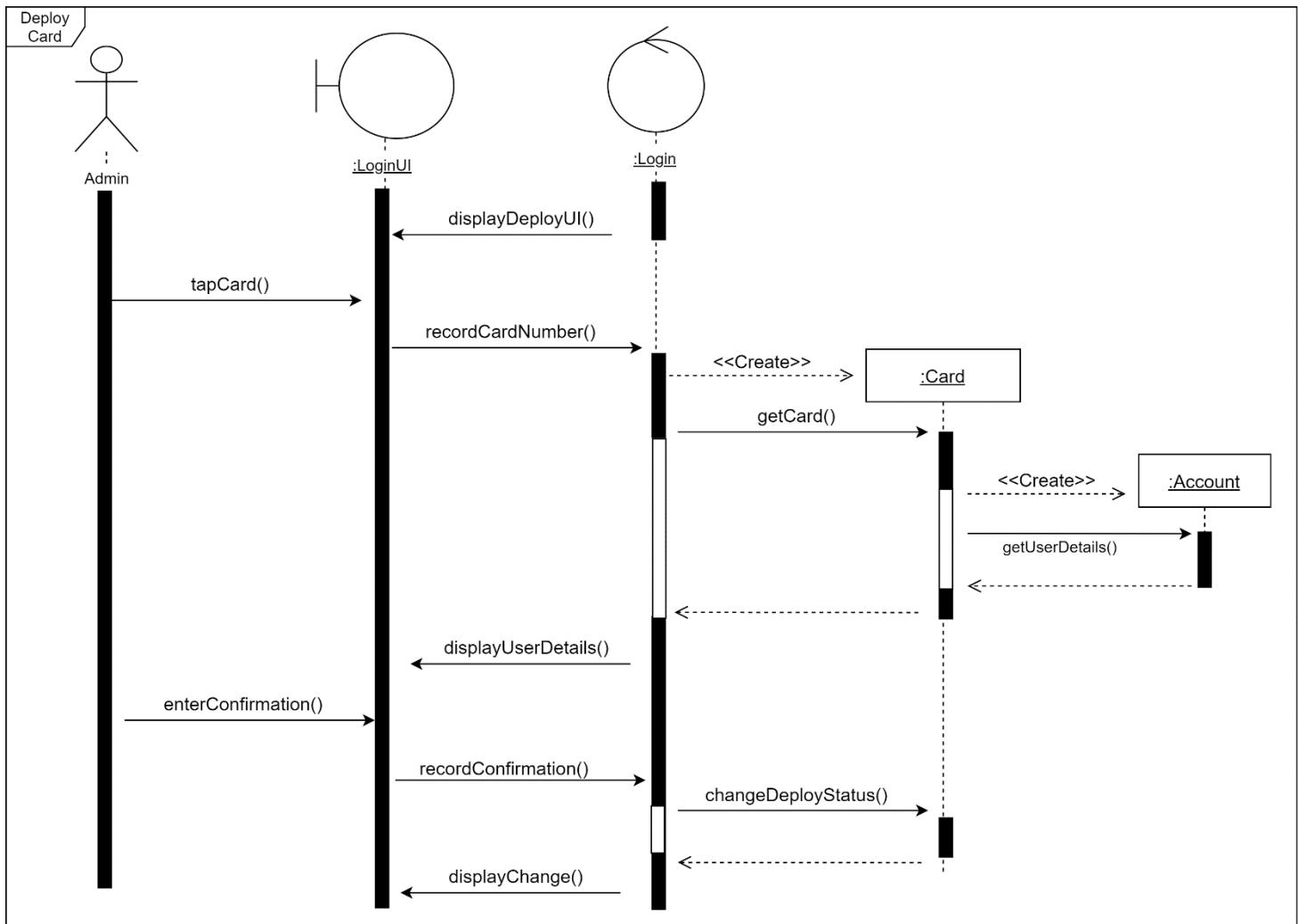


Figure 33 Deploy Card Sequence Diagram

### 3.6.3.6 Top-up

#### a) Data Flow Diagram

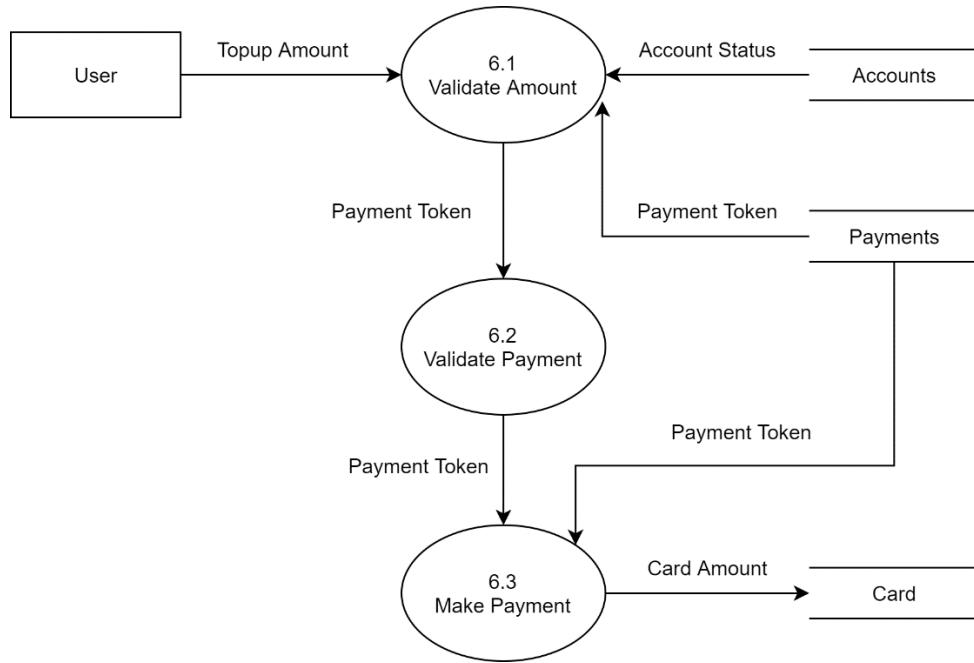


Figure 34 Top-up Level 2 DFD

#### b) Communication Diagram

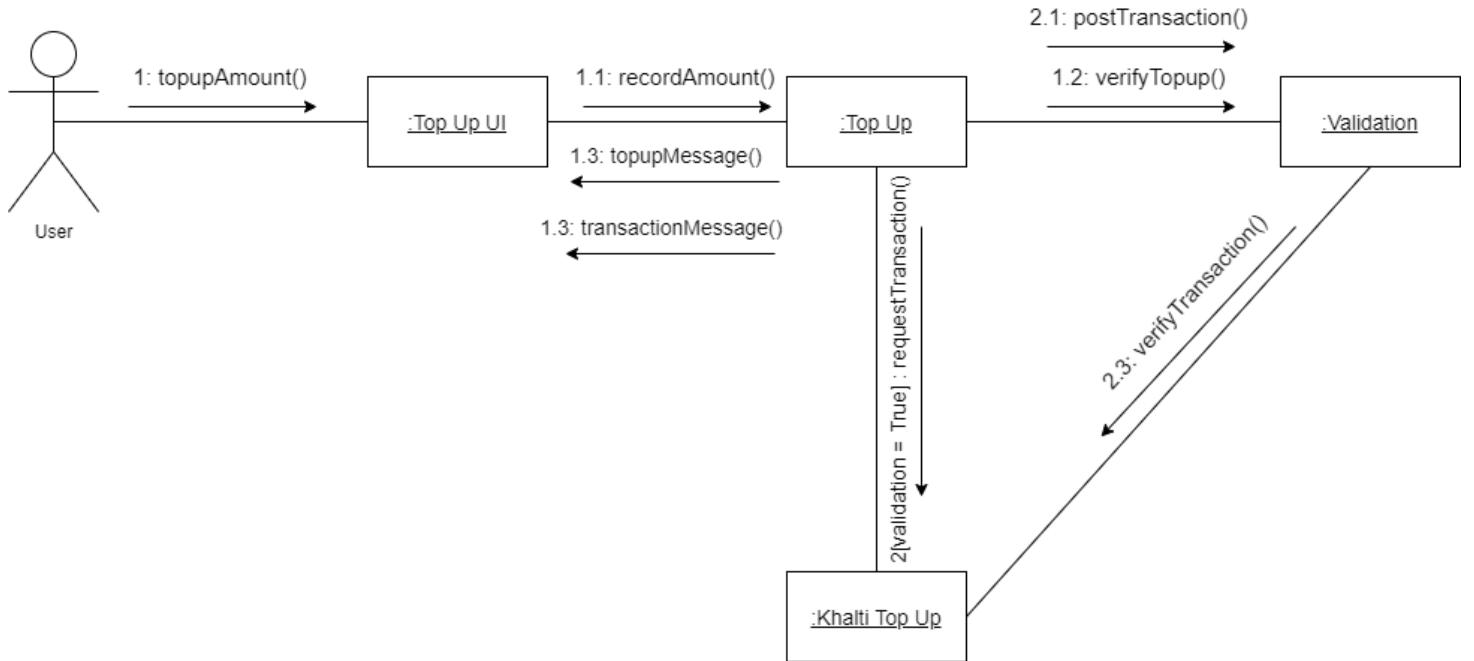


Figure 35 Top-up Communication Diagram

## c) Sequence Diagram

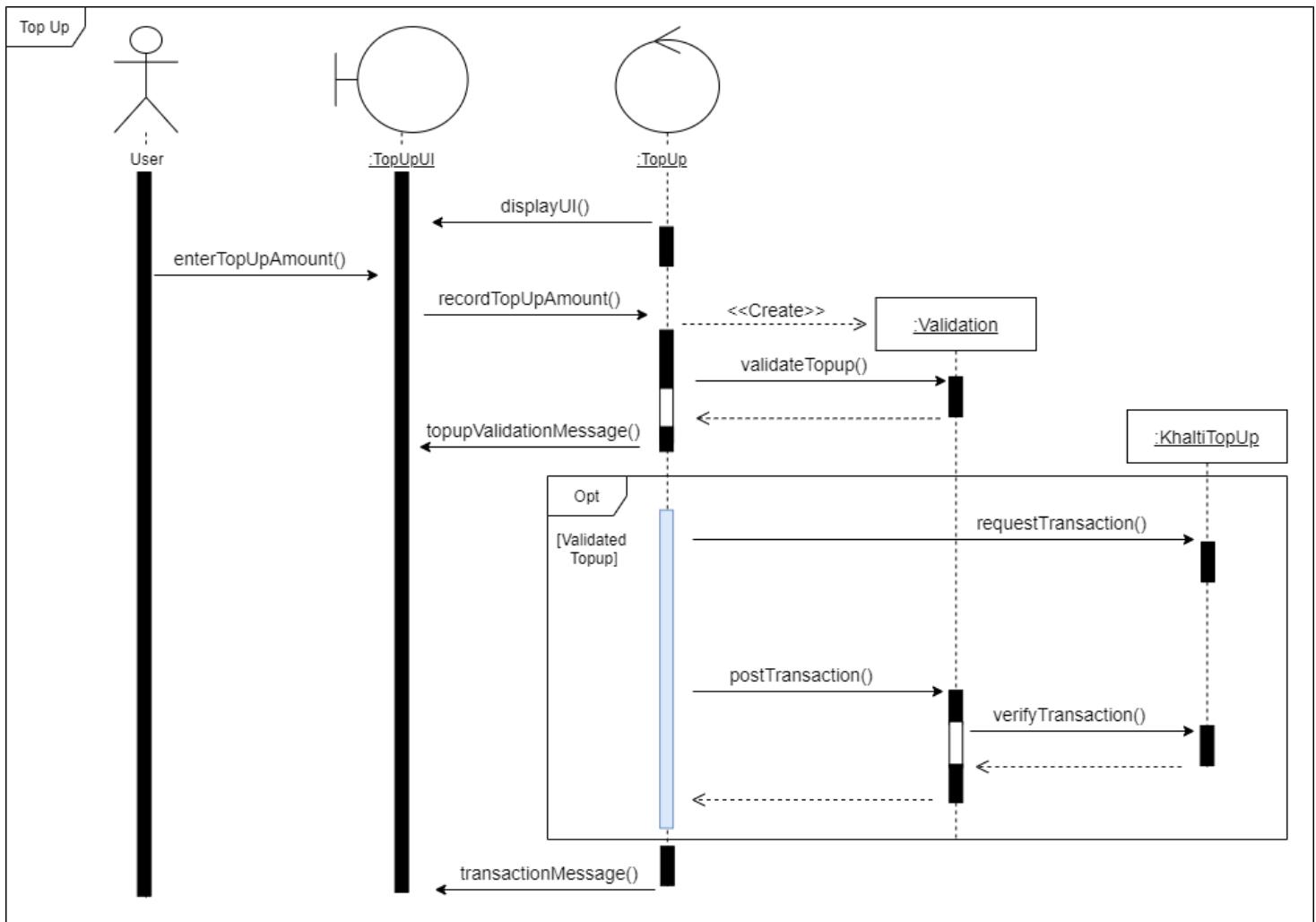


Figure 36 Top-up Sequence Diagram

## 3.6.3.7 Read Card

## a) Data Flow Diagram

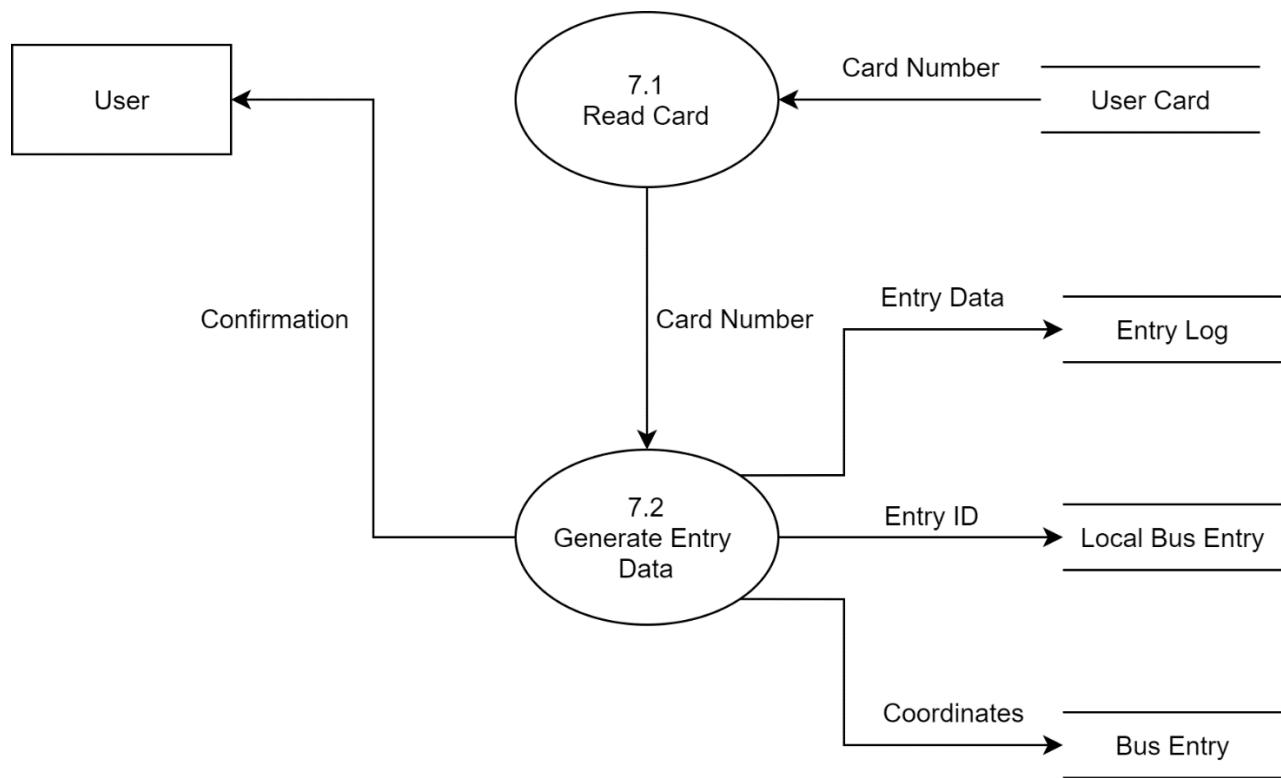


Figure 37 Entry Read DFD

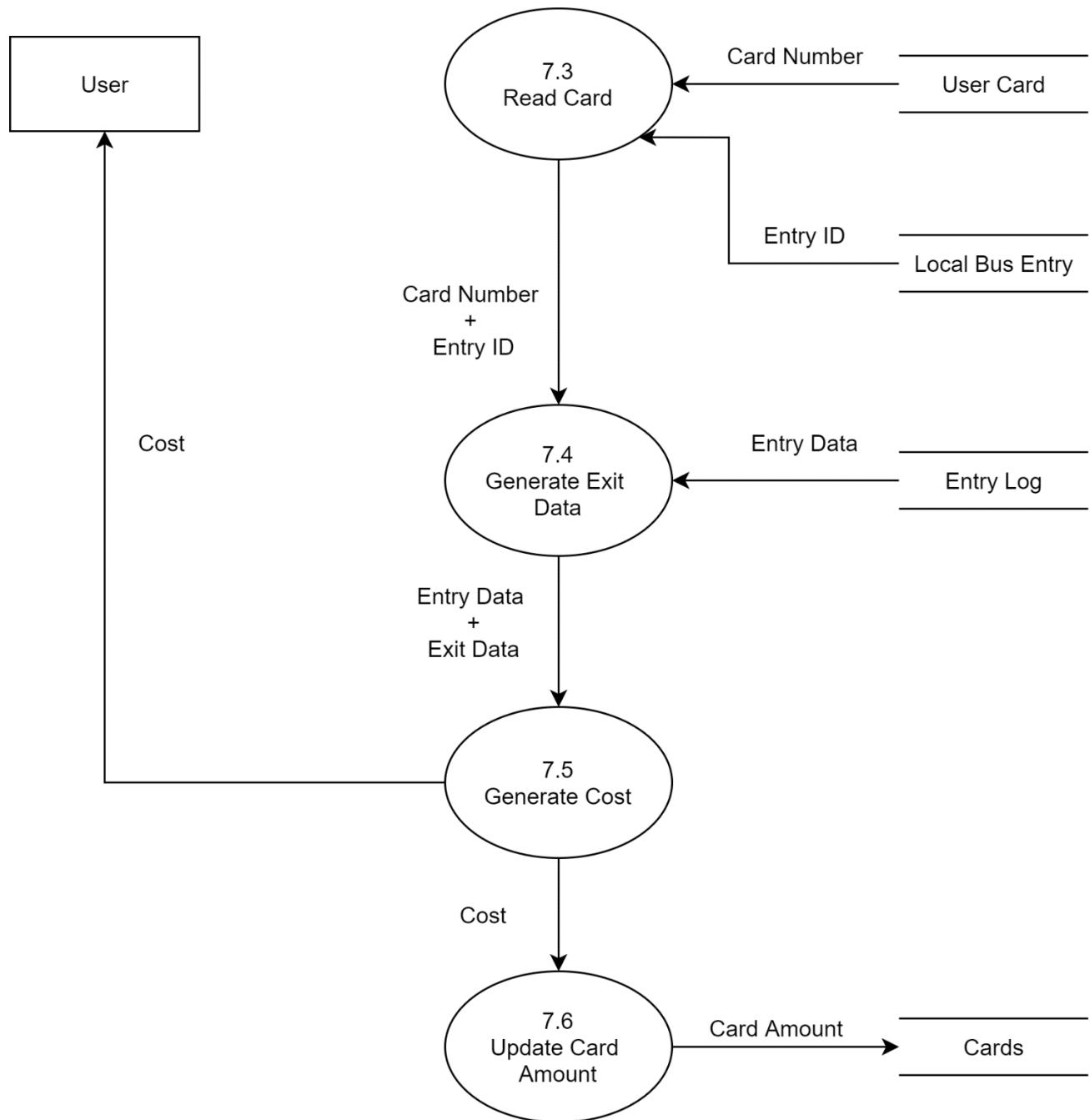


Figure 38 Exit Read DFD

## b) Communication Diagram

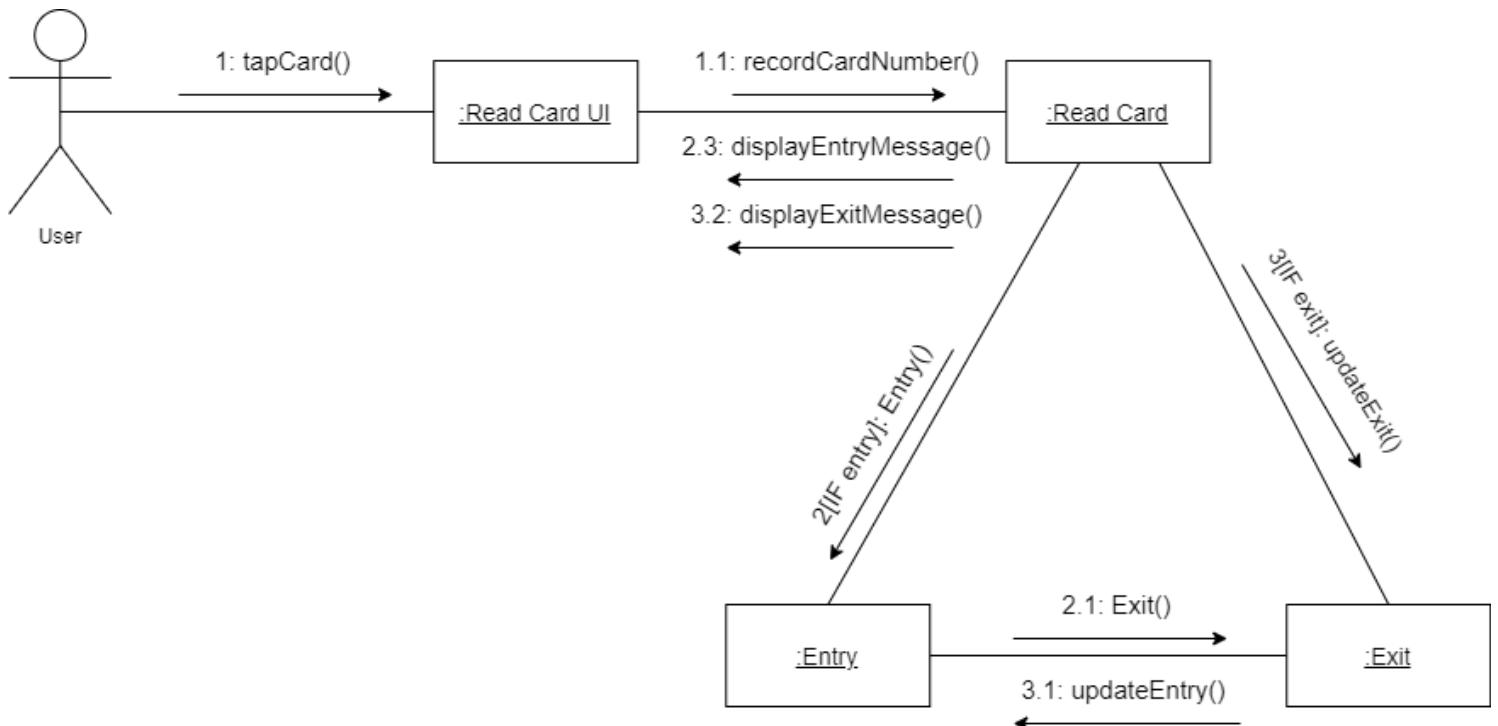


Figure 39 Read Card Communication Diagram

## c) Sequence Diagram

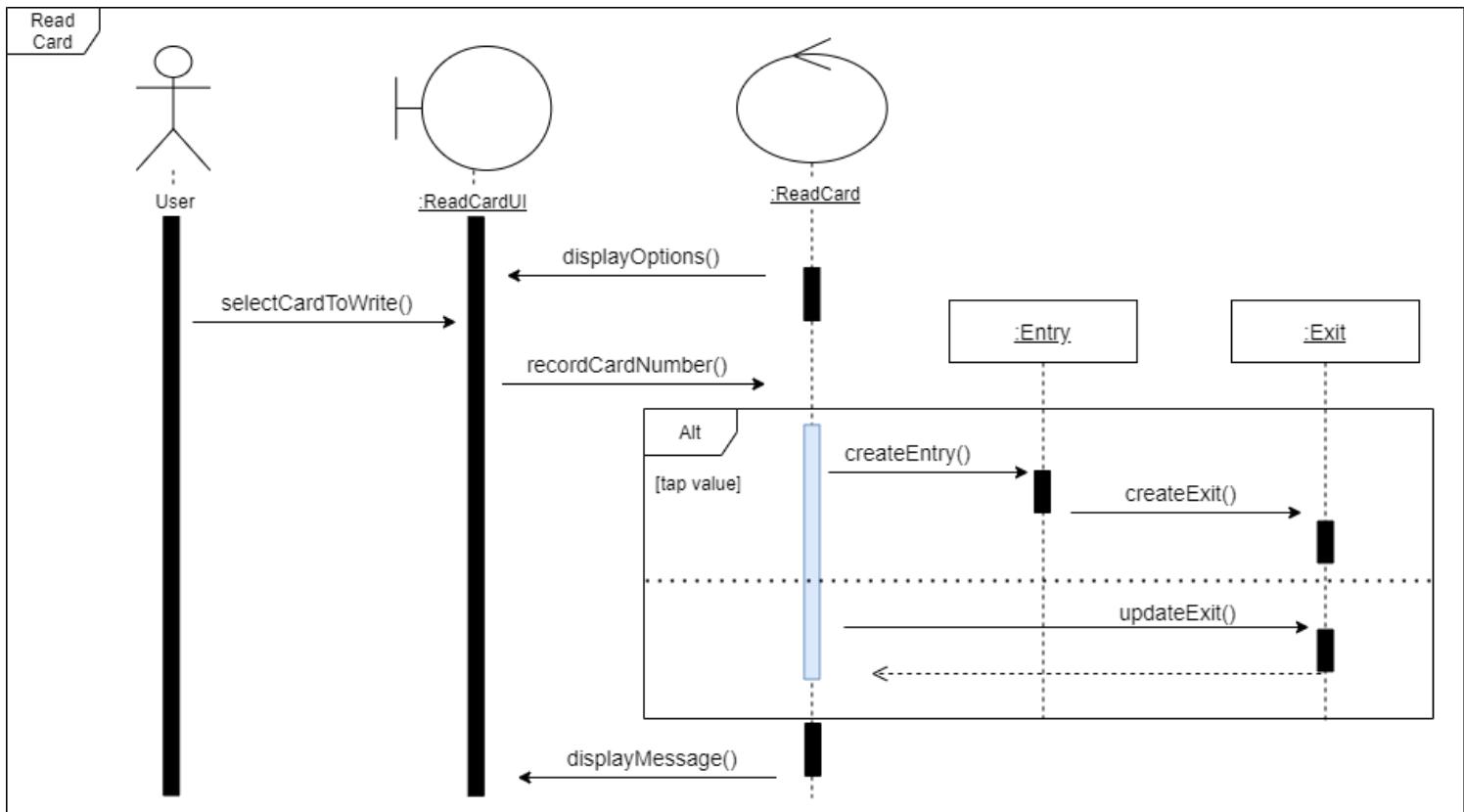


Figure 40 Read Card Sequence Diagram

### 3.6.3.8 View Travel Stats

#### a) Data Flow Diagram

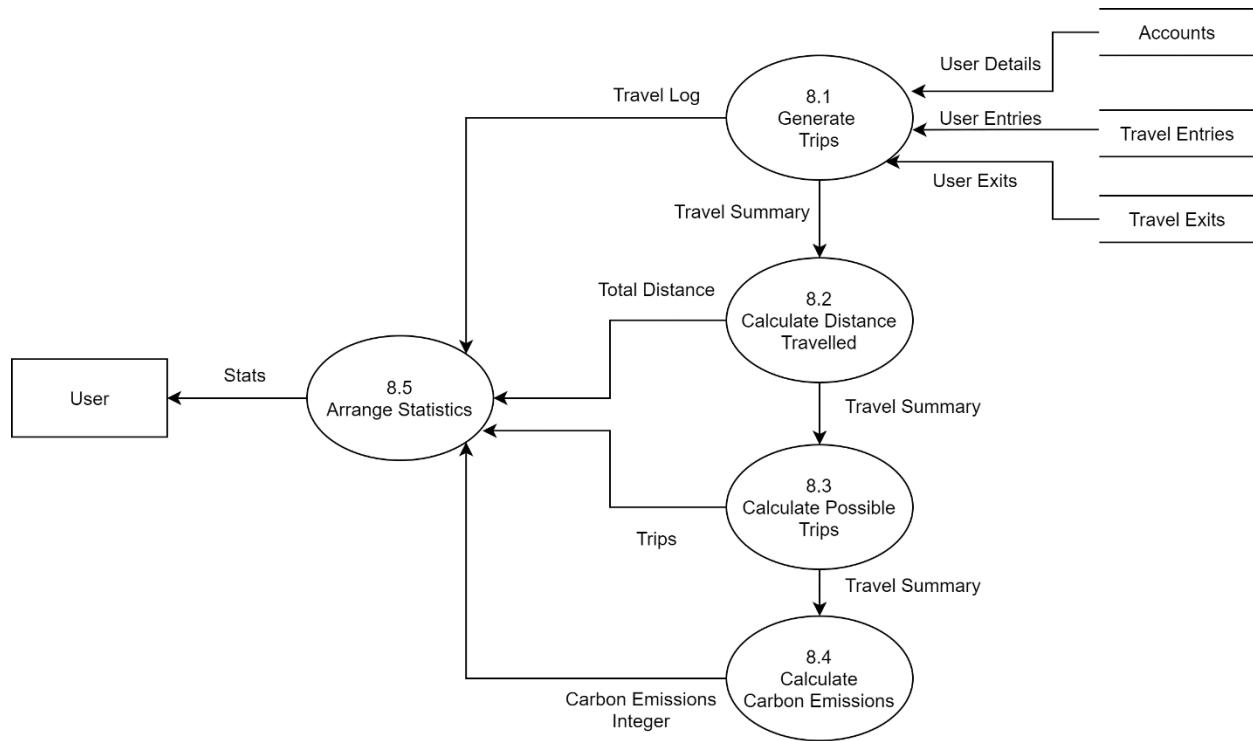


Figure 41 View Travel Stats Level 2 DFD

#### b) Communication Diagram

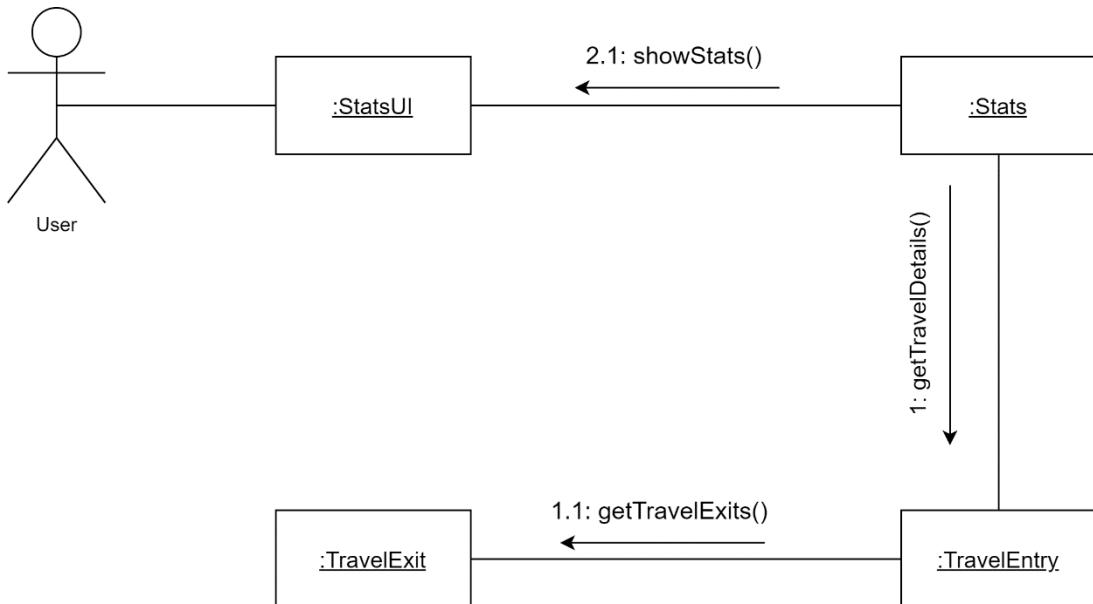


Figure 42 View Travel Stats Communication Diagram

## c) Sequence Diagram

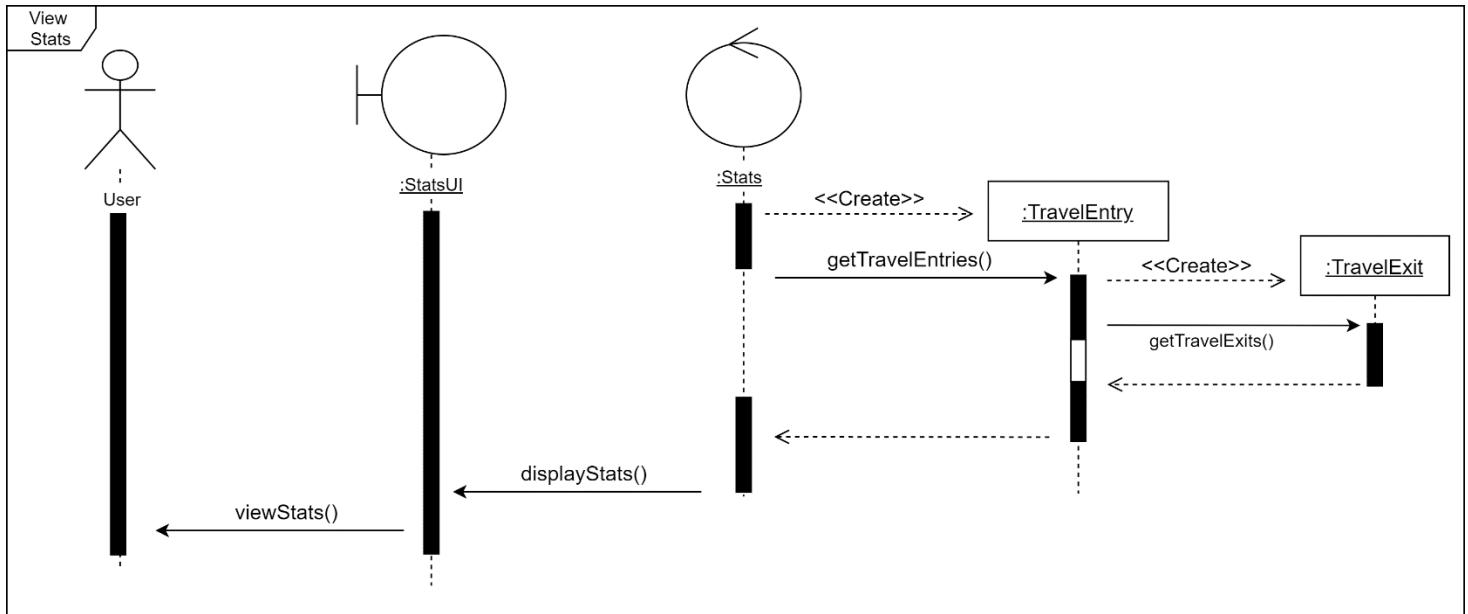


Figure 43 View Travel Stats Sequence Diagram

### 3.7 Implementation

Following RUP methodology, the implementation of this project has been done according to the timeline plotted as Gantt Chart shown in the [appendix section](#).

#### 3.7.1 Database Design

Using the ERD from the design phases and continuous iterations throughout the project builds, the models for the database of the application was built.

```
class Status(models.Model):
    status = models.PositiveIntegerField(primary_key=True, validators=[.MaxValueValidator(4)])
    status_message = models.CharField(null=True, blank=True, max_length=400)

    def __str__(self):
        return str(self.status_message)
```

Figure 44 Model for User Status

```
# custom user model to accomodate image fields and phone number as a login parameter
class Account(AbstractBaseUser, PermissionsMixin):
    phone      = PhoneNumberField(null=False, blank=False, unique=True)
    email      = models.EmailField(verbose_name="email", max_length=60, unique=True)

    first_name = models.CharField(max_length=30)
    last_name  = models.CharField(max_length=30)

    profile_pic = models.ImageField(default='default.png', null=False, blank=True, upload_to=upload_profile_pic)
    front_credential = models.ImageField(default='default.png', null=False, blank=True, upload_to=upload_front_cred)
    back_credential = models.ImageField(default='default.png', null=False, blank=True, upload_to=upload_back_cred)

    status      = models.ForeignKey(Status, default=0, null=False, on_delete=PROTECT)

    date_joined = models.DateTimeField(verbose_name='date joined', auto_now_add=True)
    last_login   = models.DateTimeField(verbose_name='last login', auto_now=True)
    is_admin     = models.BooleanField(default=False)
    is_active    = models.BooleanField(default=True)
    is_staff     = models.BooleanField(default=False)
    is_superuser = models.BooleanField(default=False)

    USERNAME_FIELD = 'phone'
    REQUIRED_FIELDS = ['email', 'first_name', 'last_name']

    objects = MyAccountManager()

    def __str__(self):
        return self.email

    def has_perm(self, perm, obj=None):
        return self.is_admin

    def has_module_perms(self, app_label):
        return True
```

Figure 45 Custom User Model

```

class MyAccountManager(BaseUserManager):
    def create_user(self, phone, email, first_name, last_name, profile_pic, front_credential, back_credential, status=0, password=None):
        if not phone:
            raise ValueError("Users must have a phone number")
        if not email:
            raise ValueError("Users must have an email")
        if not first_name:
            raise ValueError("Users must have a first name")
        if not last_name:
            raise ValueError("Users must have a last name")

        user = self.model(
            phone=phone,
            email=self.normalize_email(email),
            first_name=first_name,
            last_name=last_name,
            profile_pic=profile_pic,
            front_credential=front_credential,
            back_credential=back_credential,
            status=status
        )
        user.set_password(password)
        user.save(using=self._db)
        return user

    def create_superuser(self, phone, password, email, first_name, last_name):
        user = self.create_user(
            password=password,
            phone=phone,
            email=email,
            first_name=first_name,
            last_name=last_name,
            profile_pic="",
            front_credential="",
            back_credential="",
            status=Status.objects.get(status=3)
        )
        user.is_admin = True
        user.is_staff = True
        user.is_superuser = True

```

Figure 46 User Model Manager

```

class AccountCreationBuffer(models.Model):
    phone          = PhoneNumberField(null=False, blank=False)
    email          = models.EmailField(verbose_name="email", max_length=60)

    first_name     = models.CharField(max_length=30)
    last_name      = models.CharField(max_length=30)

    secret_word    = models.CharField(max_length=128)

    account_buffer_id = models.CharField(max_length=36)

```

Figure 47 Model for Account Creation

```

class Card(models.Model):
    user = models.OneToOneField(Account, null=True, blank=True, on_delete=models.CASCADE)

    card_number = models.CharField(max_length=200, primary_key=True)
    card_written = models.BooleanField(default=False, null=False)
    card_write_date = models.DateField(null=True, blank=True)
    card_delivered = models.BooleanField(default=False, null=False)

    is_discounted = models.BooleanField(default=False, null=False)
    is_disabled = models.BooleanField(default=False, null=False)

    card_amount = models.PositiveIntegerField(default=0, null=False)

    def __str__(self):
        return self.card_number

```

Figure 48 Model for Cards

```

class Topup(models.Model):
    user = models.ForeignKey(Account, null=True, on_delete=models.PROTECT)
    amount = models.PositiveIntegerField()
    topup_date = models.DateField(auto_now_add=True)
    topup_time = models.TimeField(auto_now_add=True)

```

Figure 49 Model for Payments

```

class Bus(models.Model):
    bus_number = models.CharField(max_length=30, primary_key=True)
    bus_route = models.CharField(max_length=60)
    passenger_capacity = models.PositiveIntegerField()

```

Figure 50 Model for Bus Data

```

class BusTravelLog(models.Model):
    bus_number = models.ForeignKey(Bus, on_delete=PROTECT)
    travel_date = models.DateField()
    travel_time = models.TimeField()
    latitude = models.CharField(max_length=60)
    longitude = models.CharField(max_length=60)

```

Figure 51 Model for Bus Travel Log

```
class UserEntry(models.Model):
    trip_id = models.CharField(max_length=60, primary_key=True)
    card_number = models.ForeignKey(Card, on_delete=PROTECT)
    bus = models.ForeignKey(Bus, on_delete=PROTECT)
    entry_date = models.DateField(blank=False)
    entry_time = models.TimeField(blank=False)
    entry_latitude = models.CharField(max_length=60, null=False, blank=False)
    entry_longitude = models.CharField(max_length=60, null=False, blank=False)
    entry_name = models.CharField(max_length = 200, null=True, blank=True)

    def get_location(self):
        location = ""
        location_list = self.entry_name.split(",")
        district_list = location_list[1].strip().split(",")
        district = district_list[0].split()
        try:
            num = int(district[-1])
        except:
            location = location + location_list[0] + ", "+ district_list[0]
        else:
            district.pop()
            location = location + location_list[0] + ", "+ district[0]

        return location
```

Figure 52 Model for User Entries

```
class UserExit(models.Model):
    entry = models.OneToOneField(UserEntry, on_delete=models.CASCADE)
    exit_date = models.DateField(null=True)
    exit_time = models.TimeField(null=True)
    exit_latitude = models.CharField(max_length=60, null=True)
    exit_longitude = models.CharField(max_length=60, null=True)
    exit_name = models.CharField(max_length = 200, null=True, blank=True)
    trip_distance = models.IntegerField(default=0, null=False)
    card_tap = models.BooleanField(default=False, null=False)
    fare = models.PositiveIntegerField(default=0, null=False)

    def get_location(self):
        location = ""
        location_list = self.exit_name.split(",")
        district_list = location_list[1].strip().split(",")
        district = district_list[0].split()
        try:
            num = int(district[-1])
        except:
            location = location + location_list[0] + ", " + district_list[0]
        else:
            district.pop()
            location = location + location_list[0] + ", " + district[0]

        return location
```

Figure 53 Model for User Exits

### 3.7.2 Web Application

#### 3.7.2.1 App Separation

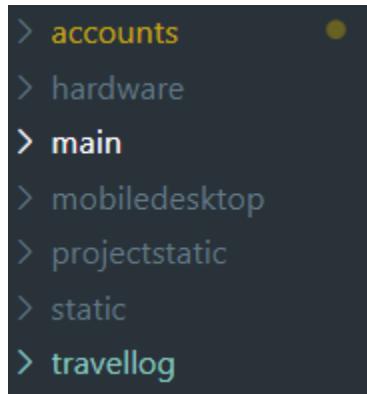


Figure 54 Project App Separation

To maintain parts of the overall project, the project was split into further 4 apps listed as, accounts, hardware, mobiledesktop and travellog.

### 3.7.2.2 Web Application Endpoints

```
urlpatterns = [
    path('', views.adminDash, name="adminDash"),
    path('home', views.userDash, name="home"),
    path('topup/', views.topup, name="topup"),
    path('topup-verify/', views.topupVerify, name="topupVerify"),
    path('password/', login_required(views.PasswordsChangeView.as_view()), name="password"),
    path('password-change-success/', views.PasswordsChangeSuccess, name="password_change_success"),

    path('reset_password/',
        auth_views.PasswordResetView.as_view(template_name="accounts/password_reset.html"),
        name="reset_password"),
    path('reset_password_sent/',
        auth_views.PasswordResetDoneView.as_view(template_name="accounts/password_reset_sent.html"),
        name="password_reset_done"),
    path('reset/<uidb64>/<token>/',
        auth_views.PasswordResetConfirmView.as_view(template_name="accounts/password_reset_form.html"),
        name="password_reset_confirm"),
    path('reset_password_complete/',
        auth_views.PasswordResetCompleteView.as_view(template_name="accounts/password_reset_done.html"),
        name="password_reset_complete"),
    path('register/', views.registerPage, name="register"),
    path('phoneverify/', views.token_validation, name="phoneverify"),
    path('login/', views.loginPage, name="login"),
    path('logout/', views.logoutUser, name="logout"),
    path('profile/', views.manageProfile, name="profile"),
    path('credentials/', views.manageCredentials, name="credentials"),
    path('customer/<str:pk>', views.customerPage, name="customerPage"),
    path('foul/', views.foulPage, name="foulPage"),
    path('foul/<str:pk>', views.foulCustomerPage, name="foulCustomerPage"),
]
```

Figure 55 URLs for Accounts App

```
urlpatterns = [
    path('get-cards/', views.GetCards.as_view(), name='getCards'),
    path('write-card/<str:phone>', views.WriteCard.as_view(), name='writeCard'),
    path('test-set-card/<str:card_number>', views.TestSetCard.as_view(), name='testSetCard'),
    path('deploy-card/<str:card_number>', views.DeployCard.as_view(), name='deployCard'),
]
```

Figure 56 URLs for Raspberry Pi

```
urlpatterns = [
    path('applogin', obtain_auth_token, name="applogin"),
    path('get-travel-log', views.getTravelLog.as_view(),name='getTravelLog'),
]
```

Figure 57 URLs for Login and Travel History on Mobile Devices

```
urlpatterns = [
    path('entry-read/',views.EntryRead.as_view(),name='entryRead'),
    path('exit-read/<str:trpnum>',views.ExitRead.as_view(),name='exitRead'),
    path('bus-route/',views.BusRoute.as_view(),name='busRoute')
]
```

Figure 58 URLs for Making Travel Log

### 3.7.2.3 User Signup

This version of registration of users uses phone number for registration and authentication without top-up validation and front end version of choosing country code.

```
<body>
    <h1>Register Account</h1>

    <form method="POST" action="">
        {% csrf_token %}
        First Name
        {{form.first_name}} <br>
        Last Name
        {{form.last_name}} <br>
        Phone
        {{form.phone}} <br>
        Email
        {{form.email}} <br>
        Password
        {{form.password1}} <br>
        Confirm Password
        {{form.password2}} <br>
        <button type="submit">Register</button>

        {% if form.errors %}
            {% for field in form %}
                {% for error in field.errors %}
                    <p> {{ error }} </p>
                {% endfor %}
            {% endfor %}
        {% endif %}
    </form>
</body>
```

Figure 59 Basic Signup Form Code

```
def registerPage(request):
    context = {}

    if request.POST:
        form = RegistrationForm(request.POST)
        if form.is_valid():

            form.save()

            name = form.cleaned_data.get('first_name')

            messages.success(request, 'Account was created for ' + name)

            return redirect('login')
        else:
            context['form'] = form
    else:
        form = RegistrationForm()
        context['form'] = form

    return render(request, 'accounts/register.html', context)
```

Figure 60 User Creation without OTP Validation

### 3.7.2.4 User Signup with OTP

The figure consists of two screenshots of a mobile application. The left screenshot, titled 'Card Pay Register', shows a registration form with fields for First Name, Last Name, Phone number (+977 984-1234567), Email, Enter Password, Re-enter Password, and a Register Account button. It also includes a link to log in if already registered. The right screenshot, titled 'Verify Phone', shows a verification screen with a Verify Token button and a placeholder for entering the code sent via SMS.

Figure 61 Registration UI

Figure 62 Verify Phone UI

```
class RegistrationForm(UserCreationForm):
    phone = PhoneNumberField()
    class Meta:
        model = Account
        fields = ('first_name', 'last_name', 'phone', 'email', 'password1', 'password2')
```

Figure 63 User Registration Model Form

```
@unauthenticated_user
def registerPage(request):
    context = {}

    if request.POST:
        form = RegistrationForm(request.POST)
        if form.is_valid():
            form.cleaned_data['phone'] = request.POST['phone']
            cl = form.cleaned_data

            buffer_token = str(uuid.uuid1())

            AccountCreationBuffer.objects.create(phone=cl['phone'], email=cl['email'], first_name=cl['first_name'], last_name=cl['last_name'], secret_word=cl['password2'], account_buffer_id=buffer_token)

            request.session['bufferToken'] = buffer_token
            request.session['phone_number'] = request.POST['phone']

            verification = twilio_client.verify.verifications.create(phone=cl['phone'], method='sms')
            return redirect('phonerverify')
        else:
            context['form'] = form
    else:
        form = RegistrationForm()
        context['form'] = form

    return render(request, 'accounts/register.html', context)
```

Figure 64 Registration Backend Buffer

```
@unauthenticated_user
def token_validation(request):
    if request.session.get('bufferToken', False):
        if request.method == 'POST':
            form = TokenForm(request.POST)
            if form.is_valid():
                verification = twilio_client.verify.verifications.create(phone=request.session['phone_number'], token=form.cleaned_data['tokenfield'])

                if verification.status == 'approved':
                    # if form.is_valid():
                    buffAcc = AccountCreationBuffer.objects.get(account_buffer_id=request.session.get('bufferToken'))
                    obj = Account.objects.create(first_name=buffAcc.first_name, last_name=buffAcc.last_name, phone=buffAcc.phone, email=buffAcc.email)
                    obj.set_password(buffAcc.secret_word)
                    obj.save()
                    buffAcc.delete()
                    del request.session['bufferToken']
                    del request.session['phone_number']
                    messages.success(request, 'Your account has been created!')

                    return redirect('login')
                else:
                    form.add_error(None, 'The token did not match, try again!')
            else:
                form.add_error(None, 'The token is not valid, try again!')
        else:
            form = TokenForm()
        return render(request, 'accounts/token_validation.html', {'form': form})
    else:
        return redirect('register')
```

Figure 65 Registration Backend

### 3.7.2.5 Login (Web and Device API)

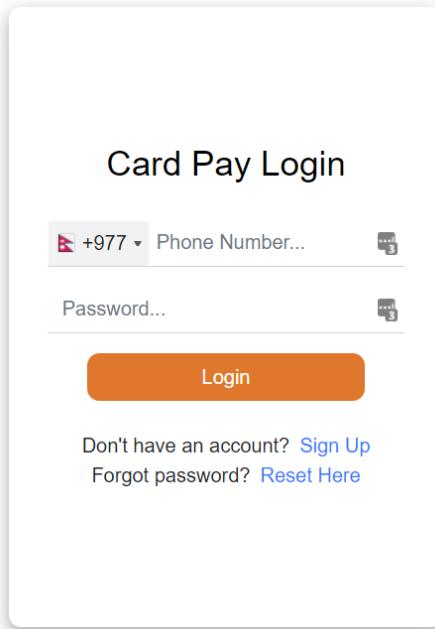


Figure 66 Login UI

```

<div class="container h-100">
  <div class="d-flex justify-content-center h-100">
    <div class="user_card">
      <div class="d-flex justify-content-center">
        <h3 id="form-title">Card Pay Login</h3>
      </div>
      <div class="d-flex justify-content-center form_container">
        <form id = "userLoginForm" method="POST" action="" >
          {% csrf_token %}
          <div class="input-group mb-3">
            <input type="tel" id = "phone" name="phone" placeholder="Phone Number..." class="form-control" >
          </div>
          <div class="input-group mb-2">
            <input type="password" name="password" placeholder="Password..." class="form-control" >
          </div>
          <div class="d-flex justify-content-center mt-3 login_container">
            <input name="loginbutton" class="btn login_btn" type="submit" value="Login">
          </div>
        </form>
      </div>
      {% for message in messages %}
        <p id="messages">{{ message }}</p>
      {% endfor %}
      <div class="mt-4">
        <div class="d-flex justify-content-center links">
          Don't have an account? <a href="{% url 'register' %}" class="ml-2">Sign Up</a>
        </div>
        <div class="d-flex justify-content-center links">
          Forgot password? <a href="{% url 'reset_password' %}" class="ml-2">Reset Here</a>
        </div>
      </div>
    </div>
  </div>
</div>

```

Figure 67 Login Front End Code

```

@unauthenticated_user
def loginPage(request):
    if request.POST:
        phone = request.POST.get('phone')
        password = request.POST.get('password')
        user = authenticate(request, phone=phone, password=password)

        if user is not None:
            login(request, user)
            if request.user.groups.all()[0].name == 'busadmin':
                return redirect('adminDash')
            return redirect('home')
        else:
            messages.info(request, 'Username or password is incorrect')
            return render(request, 'accounts/login.html')

    return render(request, 'accounts/login.html')

```

Figure 68 Login Backend

```

urlpatterns = [
    path('applogin', obtain_auth_token, name="applogin"),
    path('get-travel-log', views.getTravelLog.as_view(), name='getTravelLog'),
]

```

Figure 69 Mobile/Desktop Login Backend

### 3.7.2.6 Logout

```
def logoutUser(request):  
    logout(request)  
    return redirect('login')
```

Figure 70 Logout Backend

### 3.7.2.7 Admin Dashboard

The screenshot shows the Admin Dashboard interface. At the top, there is a dark header bar with a logo on the left and navigation links: Home, Invalid Trips, and Logout. Below the header, the title "Admin Dash" is displayed. The dashboard features three summary cards: "Accounts To Verify" (0), "Foul Trips" (5), and "Users Last 30 Days" (2). A horizontal line separates this from the "User Details" section. The "User Details" section contains a table with columns: Unverified, Resubmission Required, Pending Review, and Verified. The table lists five users with their first names, last names, phone numbers, and "Action" links (View).

Unverified	Resubmission Required	Pending Review	Verified
First Name	Last Name	Phone	Action
prithivi	maharjan	+9779860317206	<a href="#">View</a>
Prajna	Subedi	+9779860479271	<a href="#">View</a>
Sachhyam	Kaji Shakya	+9779845763331	<a href="#">View</a>
Report	Test	+9779861367410	<a href="#">View</a>

Figure 71 Admin Dashboard Table

## Search Customer

Phone	Email	Status	
-----	-----	-----	<input type="button" value="Search"/>

First Name	Last Name	Phone	Email	Status	Action
Bibek	Maharjan	+9779841877861	bibek.maharjan@islingtoncollege.edu.np	Account Verified	<a href="#">View</a>
Pranjali	Test2	+9779861367460	testpranjal@gmail.com	Account Verified	<a href="#">View</a>
prithivi	maharjan	+9779860317206	prithivi.maharjan@islingtoncollege.edu.np	Account Unverified	<a href="#">View</a>
Sodip	Thapa	+9779816809696	sodip99@gmail.com	Account Verified	<a href="#">View</a>
Sadikshya	Luitel	+9779860236211	sadikshyaluitel16@gmail.com	Account Verified	<a href="#">View</a>
test	1	+9779861234567	test@gmail.com	Needs Resubmission	<a href="#">View</a>
Prajna	Subedi	+9779860479271	itsmeprajna.subedi@gmail.com	Account Unverified	<a href="#">View</a>

Figure 72 Admin Dashboard Search

```

@login_required(login_url='login')
@admin_only
def adminDash(request):
    accounts_list = Account.objects.all()

    # filtering user types according to status and group
    unverified_customers = accounts_list.filter(groups__name='customer', status=0)
    pending_review_customers = accounts_list.filter(groups__name='customer', status=1)
    resubmission_required_customers = accounts_list.filter(groups__name='customer', status=2)
    verified_customers = accounts_list.filter(groups__name='customer', status=3)

    # calculating foul trips numbers
    user_exit_list = UserExit.objects.all()
    foul_trip_list = user_exit_list.filter(card_tap = False)
    foul_trip_number = foul_trip_list.count()

    # calculating time delta for given number of days
    current_date = datetime.datetime.now()
    active_days = 30
    one_month_ago_date = current_date - datetime.timedelta(days=active_days)

    # Joining cards, card entries, card exits where card entry between dates, card exit's card_tap property is true and grouped by card number
    active_user_number = Card.objects.filter(Q(userentry__entry_date__range=(one_month_ago_date, current_date)) & Q(userentry__userexit__card_tap=True)).annotate(count=Count('card_number')).count

    filter_accounts_list = accounts_list
    filter_form = AccountFilter(request.GET, queryset=filter_accounts_list)
    filter_accounts_list = filter_form.qs

    context = {
        'unverified_customers':unverified_customers,
        'pending_review_customers':pending_review_customers,
        'resubmission_required_customers':resubmission_required_customers,
        'verified_customers':verified_customers,
        'foul_trip_number':foul_trip_number,
        'active_user_number':active_user_number,
        'active_days':active_days,
        'filter_form':filter_form,
        'filter_data':filter_accounts_list
    }
    return render(request, 'accounts/admindash.html', context)

```

Figure 73 Admin Dashboard Backend

### 3.7.2.8 User Dashboard

## Dash Board

Card Amount	Distance Travelled	Total Trips
Rs 120	51.83 KM	28

### Travel History

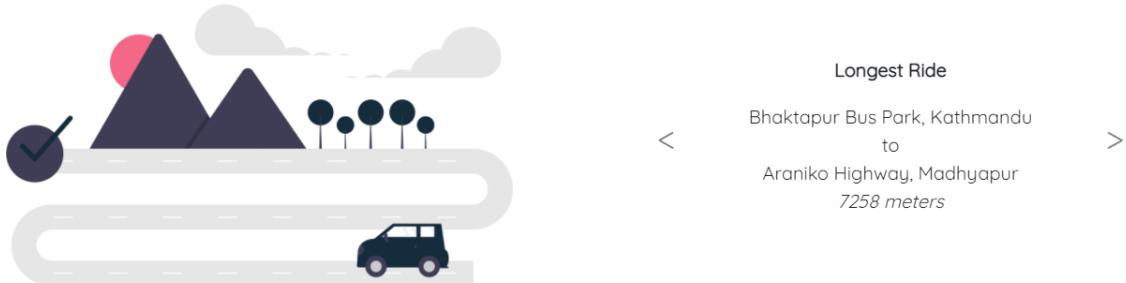
Entry Location	Exit Location	Entry Time	Exit Time	Distance (meters)	Cost (Rs)
Chandan Marg, Kathmandu	Chandan Marg, Kathmandu	April 2, 2021 3:59 p.m.	April 2, 2021 3:59 p.m.	1	0
Dillibazar Pipalko Bot Bus Stop, Dilli Bazaar Rd	Bag Bazar Sadak, Kathmandu	March 12, 2021 11:45 a.m.	March 12, 2021 12:03 p.m.	1084	7
Pashupati Rd, Kathmandu	Pashupati Rd, Kathmandu	March 11, 2021 11:49 a.m.	March 11, 2021 12:05 p.m.	1724	12
Dilli Baazar Sadak, Kathmandu	Sinamangal Rd, Kathmandu	March 11, 2021 5:43 p.m.	March 11, 2021 6:03 p.m.	2194	15
Battisputali Rd, Kathmandu	Maiju Bahal Bus Stop, Gangalal Way	March 6, 2021 8:30 a.m.	March 6, 2021 9:05 a.m.	1931	13

« [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) »

Figure 74 User Dashboard

## Your Travel Statistics

## Longest and Shortest Ride



## Possible Cycling Trips

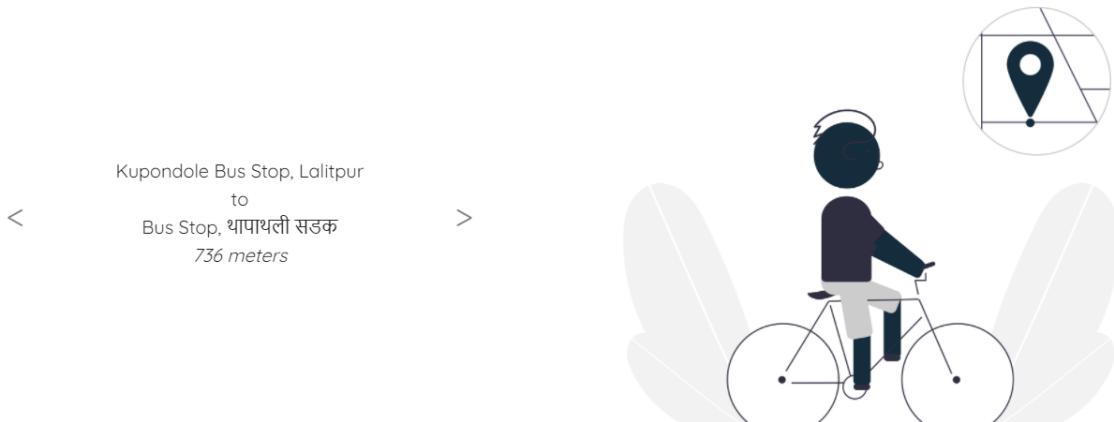


Figure 75 User Stat board

```

@login_required(login_url='login')
@allowed_users(allowed_roles=['customer'])
def userDash(request):

    userProfile = request.user
    userCard = Card.objects.get(user=userProfile)

    # getting trip valid trips for a user i.e. card has been tapped bot on entry and exit
    valid_trips_list = UserExit.objects.filter(entry_card_number=userCard, card_tap=True).order_by('-exit_date')

    # paginating trip objects in table
    page = request.GET.get('page',1)
    paginator = Paginator(valid_trips_list,5)

    try:
        valid_trips = paginator.page(page)
    except PageNotAnInteger:
        valid_trips = paginator.page(1)
    except EmptyPage:
        valid_trips = paginator.page(paginator.num_pages)

    # calculating total distance travelled by user in lifetime
    total_distance = 0

    for trip in valid_trips_list:
        total_distance = total_distance + trip.trip_distance
    total_distance = total_distance/1000

    total_trips = valid_trips_list.count

    # Longest and Shortest Ride
    longest_ride = valid_trips_list.order_by('-trip_distance').first()
    shortest_ride = valid_trips_list.order_by('trip_distance').first()

    cycling_trips = valid_trips_list.filter(trip_distance__lte=1000)
    walking_trips = valid_trips_list.filter(trip_distance__lte=500)
    carbon_emission = int((total_distance * 63)/1000)

    context = {'userProfile':userProfile,
               'userCard':userCard,
               'valid_trips':valid_trips,
               'total_distance':total_distance,
               'total_trips':total_trips,
               'longest_ride':longest_ride,
               'shortest_ride':shortest_ride,
               'cycling_trips':cycling_trips,
               'walking_trips':walking_trips,
               'carbon_emission':carbon_emission}

    # rendering status based templates
    if userCard.card_written and userCard.card_delivered:
        return render(request, 'accounts/user_dash_true.html', context)
    else:
        return render(request, 'accounts/user_dash_false.html', context)

```

Figure 76 User Dashboard and Stat board Backend

### 3.7.2.9 User Profile

Your current account status is: Account Verified

## Update Profile

First Name  
Pranjali

Last Name  
Test2

Email  
testpranjal@gmail.com

Phone  
+9779861367460

Change password [here...](#)

Figure 77 User Profile from User Page

## Credentials

Upload the front and the back side of a government provided credential. To apply for discount, upload both sides of school/university provided ID.



Figure 78 User Credential from User Page

```

@login_required(login_url='login')
@allowed_users(allowed_roles=['customer'])
def manageProfile(request):
    userProfile = request.user
    profileForm = ProfileForm(instance=userProfile)
    credForm = CredentialForm(instance=userProfile)

    phone = userProfile.phone

    status = userProfile.status
    context = {'userProfile':userProfile, 'profileForm':profileForm, 'credForm':credForm, 'phone':phone, 'status':status}

    if request.POST:
        form = ProfileForm(request.POST, request.FILES, instance=userProfile)
        if form.is_valid():
            form.save()
            context['profileForm'] = ProfileForm(instance=userProfile)
            messages.info(request, 'Profile Updated', extra_tags='profile')
        else:
            context['profileForm'] = form

    return render(request, 'accounts/profile.html', context)

@login_required(login_url='login')
@allowed_users(allowed_roles=['customer'])
def manageCredentials(request):
    userProfile = request.user
    profileForm = ProfileForm(instance=userProfile)
    credForm = CredentialForm(instance=userProfile)

    phone = userProfile.phone

    status = userProfile.status

    context = {'userProfile':userProfile, 'profileForm':profileForm, 'credForm':credForm, 'phone':phone, 'status':status}

    if request.POST:
        form = CredentialForm(request.POST, request.FILES, instance=userProfile)
        if form.is_valid():
            request.user.status = Status.objects.get(status=1)
            form.save()
            context['status'] = userProfile.status
            context['credForm'] = CredentialForm(instance=userProfile)
            messages.info(request, 'Credentials Updated', extra_tags='credentials')
        else:
            context['credForm'] = form

    return render(request, 'accounts/profile.html', context)

```

Figure 79 User Profile Backend

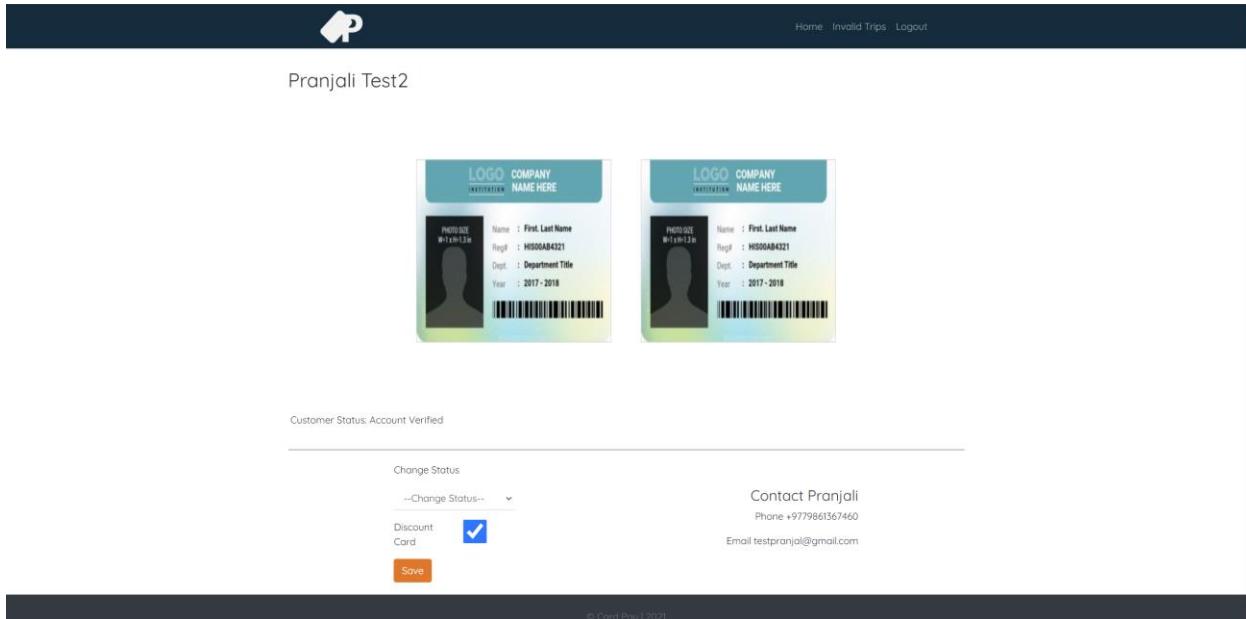


Figure 80 User Profile from Admin View

```

@login_required(login_url='login')
@allowed_users(allowed_roles=['busadmin'])
def customerPage(request, pk):
    customer = Account.objects.get(id=pk)
    form = SetUserStatusForm()
    discount_form = SetUserDiscountForm(instance=Card.objects.get(user=customer))

    context = {'customer':customer, 'form':form, 'discount_form':discount_form}

    if request.POST:
        customer = Account.objects.get(id=pk)

        discount_form = SetUserDiscountForm(request.POST, instance=Card.objects.get(user=customer))

        new_status = request.POST['change_status_field']
        customer.status = Status.objects.get(status=new_status)
        if discount_form.is_valid():
            discount_form.save()
        customer.save()

        context['customer'] = Account.objects.get(id=pk)
        context['discount_form'] = SetUserDiscountForm(instance=Card.objects.get(user=Account.objects.get(id=pk)))

        messages.success(request, 'Details Updated!')
        return render(request, 'accounts/customer.html', context)

    return render(request, 'accounts/customer.html', context)

```

Figure 81 User Profile from Admin Backend

### 3.7.2.10 Top-up

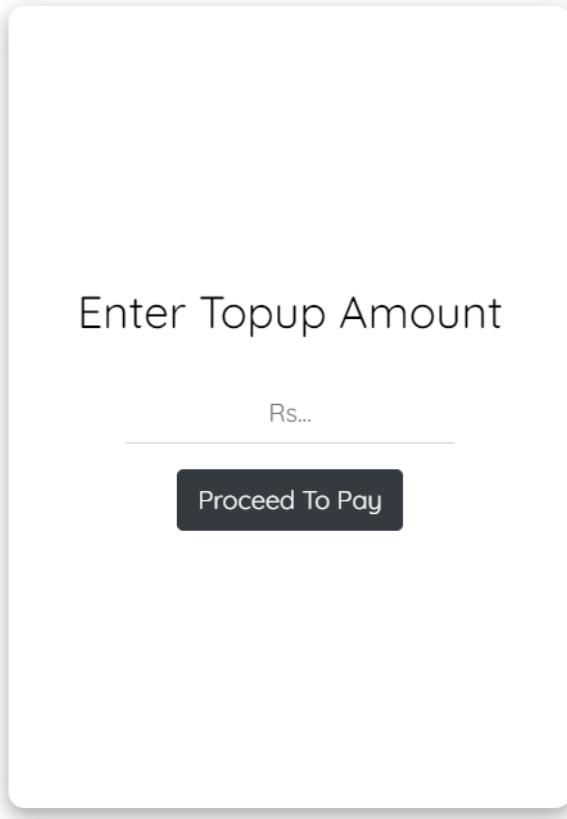


Figure 82 Top-up UI

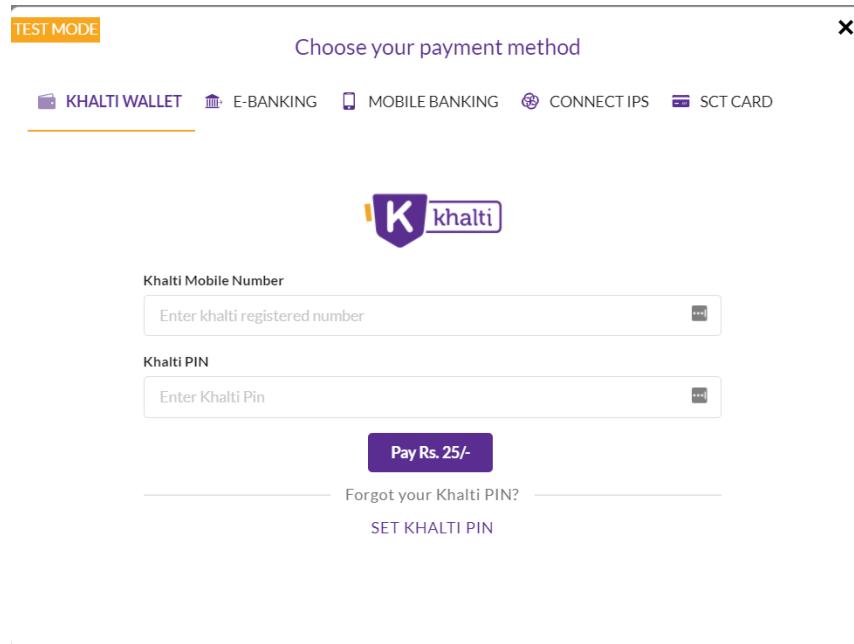


Figure 83 Khalti Integration in Top-up

```

@login_required(login_url='login')
@allowed_users(allowed_roles=['customer'])
def topup(request):
    account = request.user
    context = {}
    context['pay'] = False
    context['user_id'] = request.user.id
    if request.POST:
        amount = request.POST.get('topupamount')
        try:
            amount_paisa = int(amount) * 100
        except:
            messages.info(request, 'Amount should be positive number')
    else:
        if amount_paisa < 150:
            messages.info(request, 'Amount should be atleast Rs 15')
            return render(request,'accounts/topup.html')
        else:
            if account.status.status != 3:
                if amount_paisa > 15000:
                    messages.info(request, 'Please verify account to load more than Rs 150')
                    return render(request,'accounts/topup.html',context)
                elif ((amount_paisa + (account.card.card_amount*100)) > 15000):
                    message = "Verify account to raise topup limit <br/> Possible Load: {}".format(str(150 - account.card.card_amount))
                    messages.info(request, message=mark_safe(message))
                    return render(request,'accounts/topup.html',context)
                else:
                    context['pay'] = True
                    context['amount'] = amount_paisa
            else:
                if amount_paisa > 100000:
                    messages.info(request, 'Amount should not exceed Rs 1,000')
                    return render(request,'accounts/topup.html',context)
                elif ((amount_paisa + (account.card.card_amount*100)) > 200000):
                    message = "Your total topup balance exceeds Rs 2000 <br/> Possible Load: {}".format(str(2000 - account.card.card_amount))
                    messages.info(request, message=mark_safe(message))
                    return render(request,'accounts/topup.html',context)
                else:
                    context['pay'] = True
                    context['amount'] = amount_paisa
    return render(request,'accounts/topup.html',context)

```

Figure 84 Payment form Backend

```

@csrf_exempt
def topupVerify(request):
    if request.is_ajax() and request.method== "POST":
        data = request.POST
        token = data['token']
        amount = data['amount']
        user_id = data['user_id']
        account = Account.objects.get(pk=user_id)
        url = "https://khalti.com/api/v2/payment/verify/"

        payload = {
            "token":token,
            "amount":amount,
        }
        headers = {
            "Authorization": "Key test_secret_key_f157e5e9a2774ed9b80434192070ad0f"
        }

        response = requests.post(url,payload, headers = headers)

        response_data = response.json()
        status_code = str(response.status_code)

        if status_code == '400':
            server_response = JsonResponse({'status':'false','message':response_data['detail']},status=500)
            return server_response
        else:
            topup_amount = int(amount)/100
            new_topup = Topup(user=account, amount=topup_amount)
            new_topup.save()
            new_card_amount = account.card.card_amount + topup_amount
            Card.objects.filter(user=account).update(card_amount = new_card_amount)
            return JsonResponse(f"Topup Complete",safe=False)

```

Figure 85 Khalti Verification Backend

### 3.7.2.11 Invalid Trips

#### Invalid Trips

Users without exit taps

First Name	Last Name	Phone	Email	Action
prithivi	maharjan	+9779860317206	prithivi.maharjan@islingtoncollege.edu.np	<a href="#">View</a>
Pranjali	Test2	+9779861367460	testpranjal@gmail.com	<a href="#">View</a>
Bibek	Maharjan	+9779841877861	bibek.maharjan@islingtoncollege.edu.np	<a href="#">View</a>

Figure 86 Invalid Trips Table UI

## Bibek Maharjan

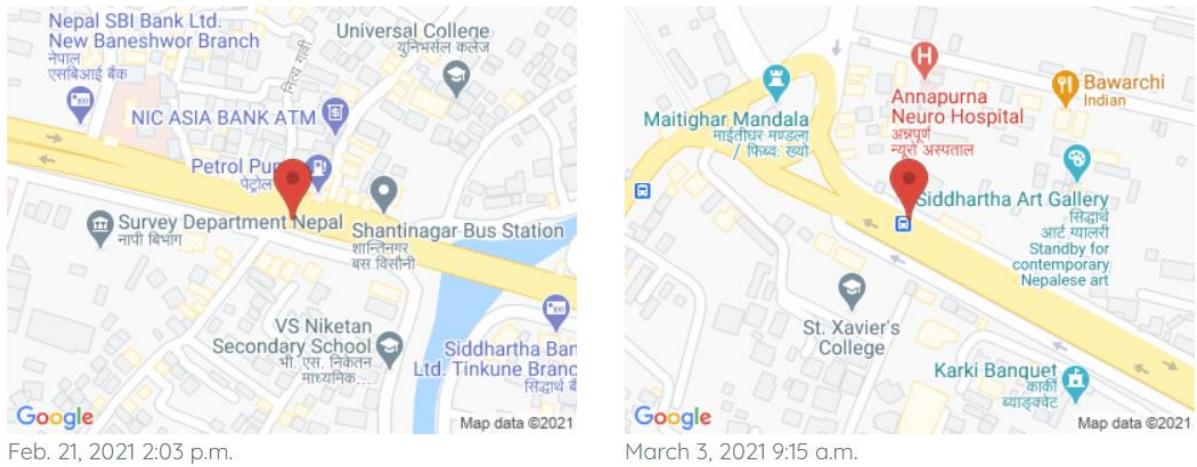


Figure 87 Invalid Trips UI

```
@login_required(login_url='login')
@allowed_users(allowed_roles=['busadmin'])
def foulCustomerPage(request,pk):
    account = Account.objects.get(id=pk)
    invalid_trips = UserEntry.objects.filter(Q(card_number__user=account) & Q(userexit__card_tap=False))
    context = {
        'account':account,
        'invalid_trips':invalid_trips
    }
    return render(request, 'accounts/foulcustomer.html',context)
```

Figure 88 Invalid Trips Backend

```
{% extends 'accounts/home.html' %}
{% load widget_tweaks %}
{% block content %}
<div class="container">
<div class="p-3 m-5">
    <h1>{{account.first_name}} {{account.last_name}}</h1>
</div>
<div class="d-flex flex-wrap justify-content-around m-3 p-3" style="min-height:75vh">
    {% for trip in invalid_trips %}
        <div>
            
            <p class="text-muted">{{trip.entry_date}} {{trip.entry_time}}</p>
        </div>
    {% endfor %}
</div>
{% endblock content %}
```

Figure 89 Invalid Trips Front End

### 3.7.3 Desktop Application

#### 3.7.3.1 Write Card

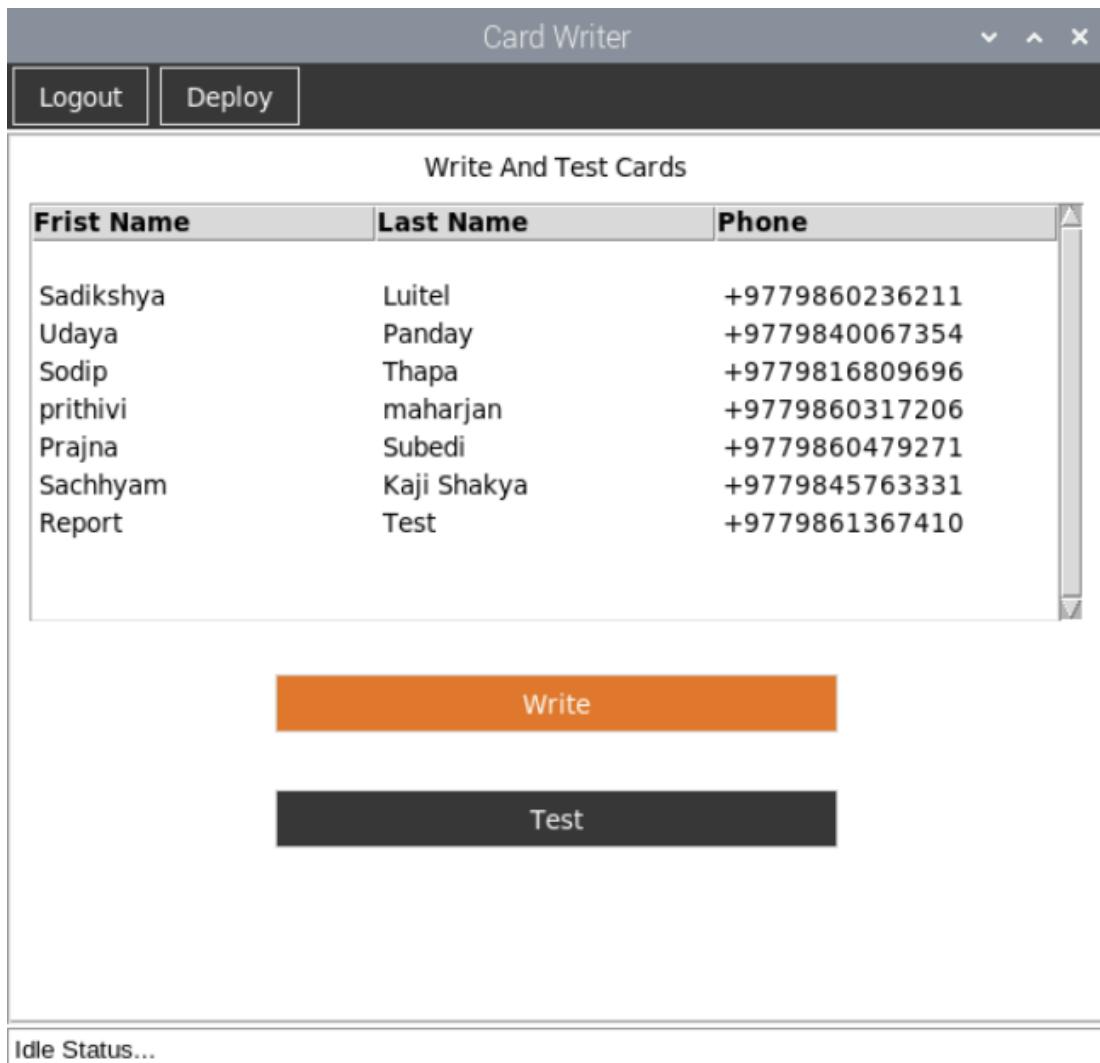


Figure 90 Write Card GUI

```

def loadTable(self):
    data = getCards(getToken())

    if isinstance(data,dict):
        from login import LoginGui
        self.root.destroy()
        LoginGui()
    else:
        self.account_tree.insert(parent='',index='end', iid='space', text="", values=('','',''))
        for i in range(len(data)):
            self.account_tree.insert(parent='',index='end', iid=i, text="", values=(data[i]['first_name'],data[i]['last_name'],data[i]['phone']))

def writeAction(self):
    # select row number
    selected = self.account_tree.focus()

    if selected != 'space':
        try:
            # select row items
            row = self.account_tree.item(selected,'values')
            # select phone
            phone = row[2]
        except:
            self.statusLbl.config(text="No row selected")
        else:
            self.statusLbl.config(text="Place card on reader")

            # write card button event:
            if writeCardEvent(token=getToken(),phone=phone):
                self.statusLbl.config(text="Card Written for "+str(phone))
                if len(self.account_tree.selection()) > 0:
                    self.account_tree.selection_remove(self.account_tree.selection()[0])
                    self.account_tree.focus("space")
            else:
                self.statusLbl.config(text="Write Unsuccessful, Please try again in a while")

    else:
        self.statusLbl.config(text="No row selected")

```

*Figure 91 Write Card Application Backend*

Imported [card](#) function in appendix.

```
class GetCards(APIView):
    authentication_classes = [TokenAuthentication]
    permission_classes = [HasGroupPermission]
    required_groups = {
        'GET': ['busadmin']
    }
    def get(self, request):
        to_write_accounts = Account.objects.filter(groups__name = 'customer', card__card_written =False)
        serializer = AccountSerializer(to_write_accounts, many=True)
        return Response(serializer.data)

class WriteCard(APIView):
    authentication_classes = [TokenAuthentication]
    permission_classes = [HasGroupPermission]
    required_groups = {
        'GET': ['busadmin']
    }
    def getObject(self,phone):
        try:
            return Account.objects.get(phone = phone)
        except Account.DoesNotExist:
            raise Http404

    def get(self,request,phone):
        card_to_write = Card.objects.get(user=self.getObject(phone))
        serializer = CardNumberSerializer(card_to_write)
        return Response(serializer.data)
```

Figure 92 Write Card Server Backend

## 3.7.3.2 Test Card

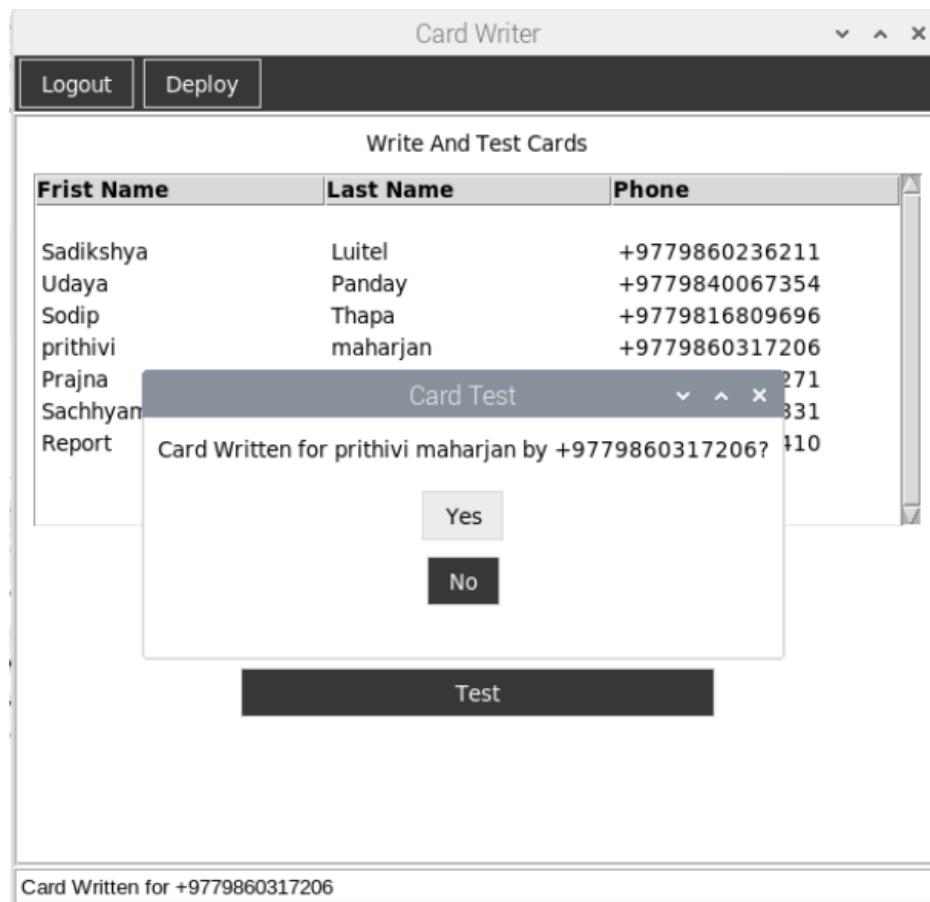


Figure 93 Test Card GUI

```

def testAction(self):
    # send api request of card to get user details and display it on popup
    data = testCardEvent(token=getToken())
    if 'detail' not in data.keys():
        self.testActionWindow(data)
    else:
        self.statusLbl.config(text=data['detail'])

def testActionWindow(self,data):
    global pop
    pop = Toplevel(self.root)
    pop.title("Card Test")
    pop.geometry("250x150")
    pop.minsize(400,150)
    pop.config(bg="white")

    popLbl = Label(pop,text="Card Written for "+data['first_name']+ ' '+data['last_name']+ ' by '+data['phone']+'?',bg="white")
    popLbl.pack(pady=10,anchor=CENTER)

    popYesBtn = tk.Button(pop,text="Yes",relief='flat',bg="#E87721",fg='white',command=lambda: self.setCardWritten(card_num=data['card_number'],written=True))
    popNoBtn = tk.Button(pop,text="No",relief='flat',bg="#373737",fg='white',command=lambda: self.setCardWritten(card_num=data['card_number'],written=False))

    popYesBtn.pack(pady=5)
    popNoBtn.pack(pady=5)

def setCardWritten(self,card_num,written):
    response = setCardWrittenEvent(token=getToken(),card_num=card_num,written=written)
    if 'status' in response.keys():
        self.statusLbl.config(text=response['status'])
        pop.destroy()
    else:
        self.statusLbl.config(text="Unable to change status at the moment...")
        pop.destroy()
    self.account_tree.delete(*self.account_tree.get_children())
    self.loadTable()

```

Figure 94 Test Card Application Backend

```

class TestSetCard(APIView):

    authentication_classes = [TokenAuthentication]
    permission_classes = [HasGroupPermission]
    required_groups = {
        'GET': ['busadmin'],
        'PUT': ['busadmin']
    }

    def get_object(self, card_number):
        try:
            return Card.objects.get(card_number = card_number)
        except Account.DoesNotExist:
            raise Http404

    def get(self, request, card_number):
        card_to_test = self.get_object(card_number)
        serializer = CardNumberSerializer(card_to_test)
        return Response(serializer.data)

    def put(self, request, card_number):
        card_to_set = self.get_object(card_number)
        serializer = CardWrittenSerializer(card_to_set, request.data)

        data = {}
        if serializer.is_valid():
            serializer.save()
            data["status"] = "Card written status changed!"
            return Response(data=data)
        else:
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

```

Figure 95 Test Card Server Backend

### 3.7.3.3 Deploy Card

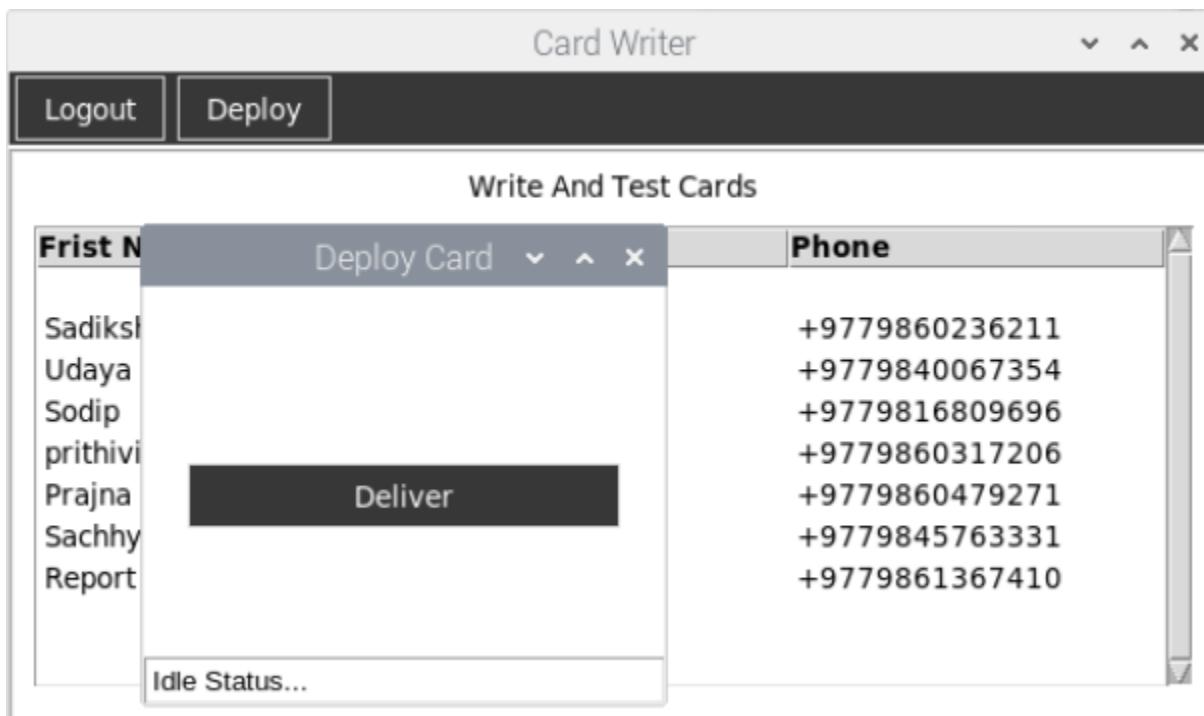


Figure 96 Deploy Card GUI

```
def setCardDeployEvent(token):
    func_response = {}
    try:
        card_num = Read.readCard()
    except:
        func_response["status"] = "Card Not Readable"
    else:
        deploy_status_data = get_deploy_status(token, card_num)
        deploy_status = deploy_status_data["card_delivered"]
        if deploy_status == "Not found":
            func_response["status"] = "Card Not Found"
        else:
            r = change_status_data = set_deploy_status(token, card_num, deploy_status)
            func_response["status"] = r["status"]
    finally:
        return func_response
```

Figure 97 Deploy Card Application Backend

```

class DeployCard(APIView):

    authentication_classes = [TokenAuthentication]
    permission_classes = [HasGroupPermission]
    required_groups = {
        'GET': ['busadmin'],
        'PUT': ['busadmin']
    }

    def get_object(self, card_number):
        try:
            return Card.objects.get(card_number = card_number)
        except Account.DoesNotExist:
            raise Http404

    def get(self, request, card_number):
        card_to_deploy = self.get_object(card_number)
        serializer = CardDeploySerializer(card_to_deploy)
        return Response(serializer.data)

    def put(self, request, card_number):
        card_to_set = self.get_object(card_number)
        serializer = CardDeploySerializer(card_to_set, request.data)

        data = {}
        if serializer.is_valid():
            serializer.save()
            if card_to_set.card_delivered:
                data["status"] = "Card delivered to user."
            else:
                data["status"] = "Card not delivered to user."
            return Response(data=data)
        else:
            return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

```

Figure 98 Deploy Card Server Backend

## 3.7.3.4 Read Card

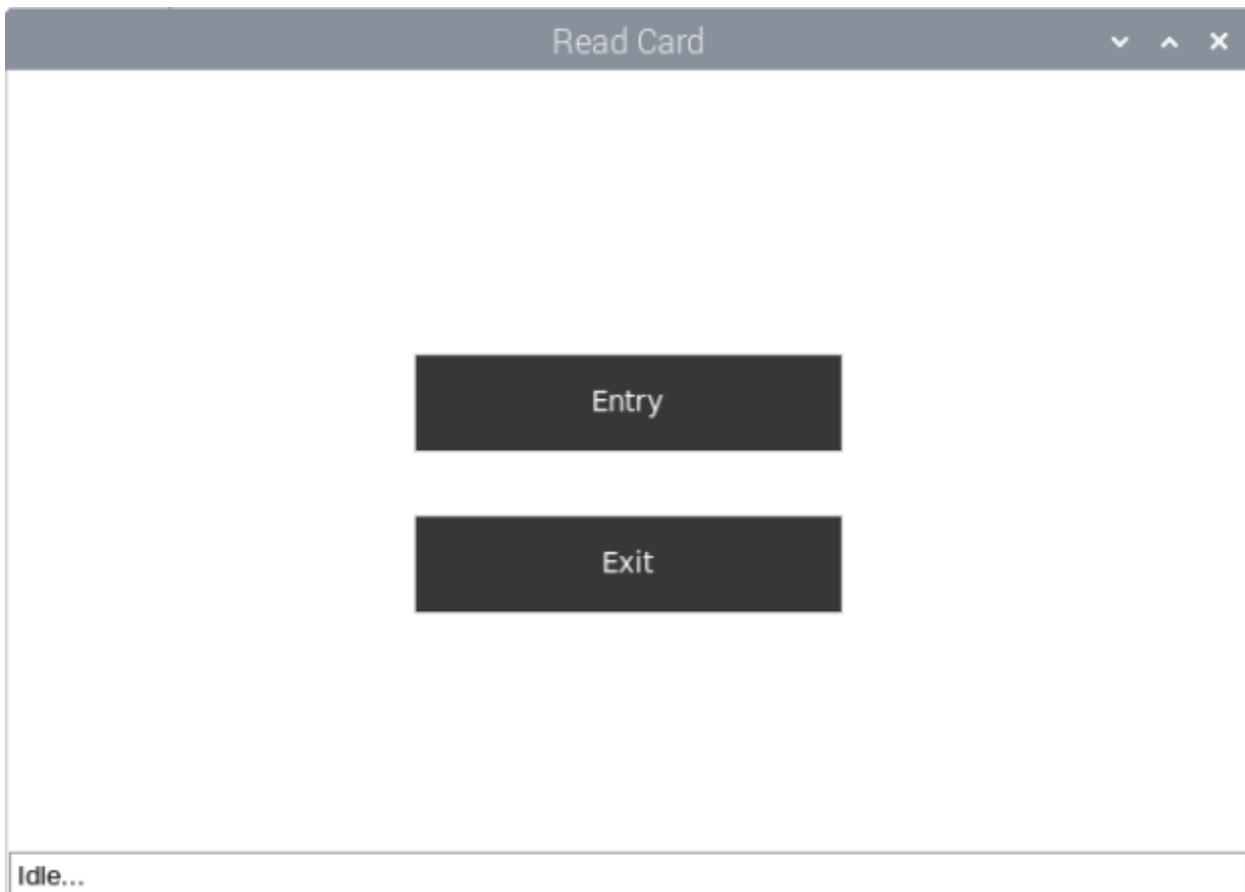


Figure 99 Read Card GUI

```

def entryBtnAction(self):
    global bus
    global reader
    card_num_rsp = btne.getCardNumber(reader)

    if card_num_rsp['status'] == 'OK':
        card_num = card_num_rsp['card_num']

        travel_id_rsp = btne.makeTravelID(card_num)

        if travel_id_rsp['status'] == 'OK':
            travel_id = travel_id_rsp['travel_id']

            lat_lon_rsp = btne.getLatitude()
            date_time_rsp = btne.getDateTimeEntry(card_num)

            if lat_lon_rsp['status'] == 'OK':
                if date_time_rsp['status'] == 'OK':
                    latitude = lat_lon_rsp['latitude']
                    longitude = lat_lon_rsp['longitude']
                    date = date_time_rsp['date']
                    time = date_time_rsp['time']

                    server_response = srequest.entryRequest(
                        trip_id = travel_id,
                        card_number = card_num,
                        bus = bus,
                        entry_date = date,
                        entry_time = time,
                        entry_latitude = latitude,
                        entry_longitude = longitude)
                if server_response['status'] == 'OK':
                    self.statusLbl.config(text='Card Entry Complete')
                else:
                    btne.deleteRow(card_num)
                    self.statusLbl.config(text=server_response['status'])
            else:
                btne.deleteRow(card_num)
                self.statusLbl.config(text=date_time_rsp['message'])
        else:
            btne.deleteRow(card_num)
            self.statusLbl.config(text=lat_lon_rsp['message'])
    else:
        self.statusLbl.config(text=travel_id_rsp['message'])

self.statusLbl.config(text=card_num_rsp['message'])

```

Figure 100 Entry Tap Application Backend

Imported [card function in appendix.](#)

```

def exitBtnAction(self):
    global bus
    global reader
    date_time_obj = datetime.datetime.now()
    date = date_time_obj.strftime("%Y-%m-%d")
    time = date_time_obj.strftime("%H:%M:%S")

    card_num_rsp = btne.getCardNumber(reader)

    if card_num_rsp['status'] == 'OK':
        card_num = card_num_rsp['card_num']

    lat_lon_rsp = btne.getLatitude()
    travel_id_rsp = btne.getTravelIdExit(card_num)

    if lat_lon_rsp['status'] == 'OK':
        if travel_id_rsp['status'] == 'OK':
            latitude = lat_lon_rsp['latitude']
            longitude = lat_lon_rsp['longitude']
            travel_id = travel_id_rsp['travel_id']

            server_response = srequest.exitRequest(trip_id = travel_id,exit_date = date,exit_time = time,exit_latitude = latitude,exit_longitude = longitude)
            if server_response['status'] == 'OK':
                self.statusLbl.config(text='Card Exit Complete' + ' Fare: '+str(server_response['fare']))
                btne.deleteRow(card_num)
            elif server_response['status'] == 'INSUFFICIENT_BALANCE':
                lblmessage = 'Insufficient Balance on Card, take Rs {}'.format(str(server_response['due_fare']))
                self.statusLbl.config(text=lblmessage)
                btne.deleteRow(card_num)
            else:
                self.statusLbl.config(text=server_response['status'])
        else:
            self.statusLbl.config(text=travel_id_rsp['message'])
    else:
        self.statusLbl.config(text=lat_lon_rsp['message'])
else:
    self.statusLbl.config(text=card_num_rsp['message'])


```

Figure 101 Exit Tap Application Backend

```

# Class to create record if user enters the bus after tapping card
class CreateRecordView:
    permission_classes = [HasAPIKey]
    response_dict = {}
    def post(self,request):
        serializer = EntrySerializer(data = request.data)
        self.response_dict['status'] = 'INVALID_DATA'
        if serializer.is_valid():
            card_num = serializer.validated_data['card_number']
            try:
                card = Card.objects.get(card_number = card_num)
            except:
                self.response_dict['status'] = 'INVALID_CARD_NUMBER'
            else:
                # Checking if card is not disabled or Card hasn't Card does not have # balance
                if not card.is_disabled and card.card_amount and card.card_delivered != 0:
                    serializer.save()
                    # Add record to bus routing after saving serializer
                    bus_serializer_data = {'bus_number':serializer.data['bus'],'travel_date':serializer.data['entry_date'],'travel_time':serializer.data['entry_time'],'latitude':serializer.data['entry_latitude'],'longitude':serializer.data['entry_longitude']}
                    bus_serializer = BusRouteSerializer(data = bus_serializer_data)
                    if bus_serializer.is_valid():
                        bus_serializer.save()
                        self.response_dict['status'] = 'OK'
                        return Response(self.response_dict)
                    else:
                        self.response_dict['status'] = 'NOT ELIGIBLE'
                        return Response(self.response_dict)
            else:
                return Response(serializer.errors)


```

Figure 102 Entry Read Server Backend

```

def mapRequest(coordinates):
    return_dict = {}
    if coordinates is not None:
        formatted_entry_exit_coordinates=entry_exit_coordinates.format(origin='.'.join(coordinates[0]), destination='.'.join(coordinates[1]))
        if len(coordinates)>2:
            waypoint_pre_format=''
            for i in range(2,len(coordinates)):
                if i > 2:
                    waypoint_pre_format = waypoint_pre_format+'|'+'.'.join(coordinates[i])
                else:
                    waypoint_pre_format = waypoint_pre_format+','.join(coordinates[i])
            formatted_waypoints = waypoints.format(waypoints=waypoint_pre_format)
        else:
            formatted_waypoints = waypoints.format(waypoints='')
    try:
        r = requests.get(url+formatted_entry_exit_coordinates+formatted_waypoints+access_key)
        r = r.json()
    except:
        return_dict['status'] = ['error']
        return return_dict
    else:
        return r
    else:
        return_dict['status'] = ['error']
        return return_dict

def getTripDistance(coordinates):
    r = mapRequest(coordinates=coordinates)
    response_dict = {}
    if r['status'] == 'OK':
        response_dict['status'] = 'OK'
        trip_distance = 0
        legs = r['routes'][0]['legs']
        for leg in legs:
            trip_distance = trip_distance + int(leg['distance']['value'])
        response_dict['trip_distance'] = trip_distance
        response_dict['entry_name'] = legs[0]['start_address']
        response_dict['exit_name'] = legs[-1]['end_address']
    else:
        response_dict['status'] = 'ERROR'
    return response_dict

```

Figure 103 Google Maps Distance Backend

```

# Class to create record if user exits the bus after tapping card
class ExitRead(APITableView):
    permission_classes = [HasAPIKey]
    response_dict = {}
    def put(self,request,trpnum):
        # 1 Check if card exists
        try:
            entry_row = UserEntry.objects.get(pk=trpnum)
        except:
            self.response_dict['status'] = 'No entry record'
        else:
            # 2 Checking if card is not disabled in Card model and Card does not have 0 balance
            if not entry_row.card_number.is_disabled and entry_row.card_number.card_amount and entry_row.card_number.card_delivered != 0:
                exit_row = UserExit.objects.get(entry=entry_row)
                serializer = ExitSerializer(exit_row,request.data)
                if serializer.is_valid():
                    serializer.save()

                    # making coordinates List by getting coordinates from user entry and user exit models
                    pre_coordinates = []
                    entry_lat = str(entry_row.entry_latitude)
                    entry_lon = str(entry_row.entry_longitude)
                    exit_lat = str(exit_row.exit_latitude)
                    exit_lon = str(exit_row.exit_longitude)
                    pre_coordinates.append((entry_lat,entry_lon))
                    pre_coordinates.append((exit_lat,exit_lon))
                    bus_num = entry_row.bus

                    way_points = getBusCoordinates(bus_num=bus_num, entry_date=entry_row.entry_date, entry_time=entry_row.entry_time, exit_date=exit_row.exit_date, exit_time=exit_row.exit_time)
                    coordinates = pre_coordinates + way_points
                    # getting trip details from GOOGLE's MAP API from imported function
                    details = getTripDistance(coordinates=coordinates)

```

Figure 104 Exit Read Backend (1)

```

if details['status'] == 'OK':
    # Retrieving required data from API response
    distance = int(details['trip_distance'])
    fare = int((distance/1000) * 7)
    entry_name = details['entry_name']
    exit_name = details['exit_name']

    # Reduce Fare if card is discounted
    if entry_row.card_number.is_discounted:
        fare = int(fare - (fare * 0.15))

    ... Actions if card has sufficient balance,
    If sufficient then reduce amount while updating required attributes
    If not sufficient, deduct remaining amount and respond with due amount in response (should not be able to tap card on entry next time)
    ...
    if entry_row.card_number.card_amount >= fare:
        self.response_dict['status'] = 'OK'
        self.response_dict['fare'] = fare
        UserEntry.objects.filter(pk=trpnum).update(entry_name=entry_name)
        UserExit.objects.filter(entry=entry_row).update(exit_name=exit_name, trip_distance=distance, fare=fare, card_tap=True)
        new_amount = entry_row.card_number.card_amount - fare
    else:
        new_amount = 0
        due_fare = fare - entry_row.card_number.card_amount
        self.response_dict['status'] = 'INSUFFICIENT_BALANCE'
        UserEntry.objects.filter(pk=trpnum).update(entry_name=entry_name)
        UserExit.objects.filter(entry=entry_row).update(exit_name=exit_name, trip_distance=distance, fare=fare, card_tap=False)
        self.response_dict['fare'] = fare
        self.response_dict['due_fare'] = due_fare

    Card.objects.filter(card_number = entry_row.card_number).update(card_amount = new_amount)
else:
    self.response_dict['status'] = 'ERROR'
    return Response(self.response_dict)
else:
    return Response(serializer.errors)
else:
    self.response_dict['status'] = 'NOT_ELIGIBLE'
    return Response(self.response_dict)

```

Figure 105 Exit Read Backend – Fare Calculation (2)

### 3.7.4 Mobile Application

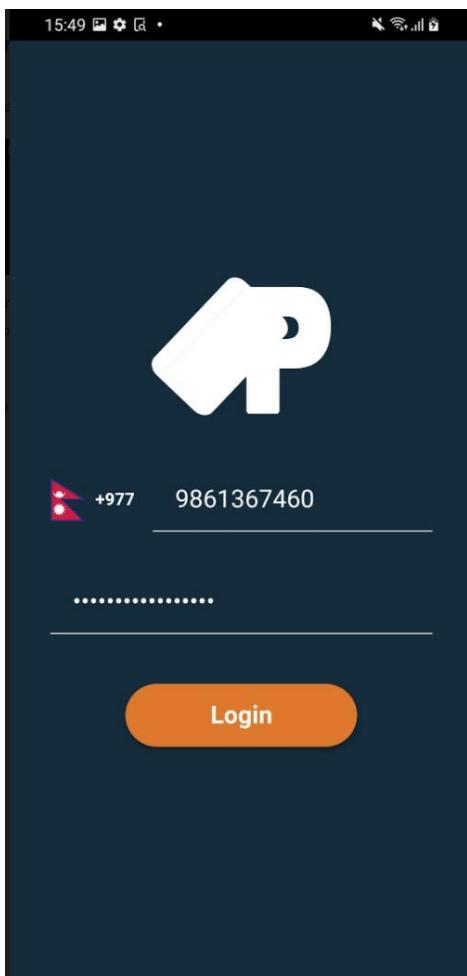


Figure 106 Mobile Login UI

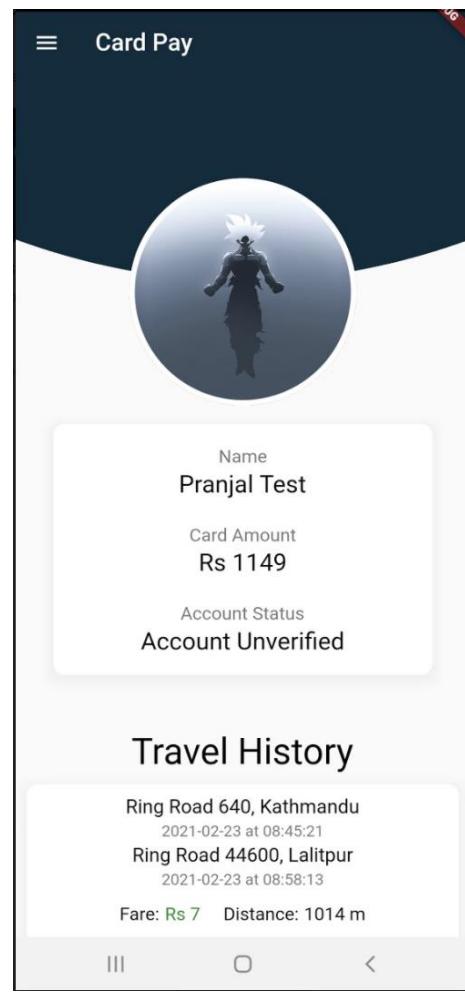


Figure 107 Mobile Dashboard UI

```
class getTravelLog(APIView):
    authentication_classes = [TokenAuthentication]
    permission_classes = [HasGroupPermission]
    required_groups = {
        'GET': ['customer']
    }

    def get(self, request):
        userProfile = request.user

        userCard = Card.objects.get(user=userProfile)

        serializer = UserDataSerializer(userCard)

        return Response(serializer.data)
```

Figure 108 Mobile Travel Log Backend

```

class TravelLogSerializer(serializers.ModelSerializer):
    entry_name = serializers.SerializerMethodField()
    exit_date = serializers.SerializerMethodField()
    exit_time = serializers.SerializerMethodField()
    exit_name = serializers.SerializerMethodField()

    trip_distance = serializers.SerializerMethodField()
    fare = serializers.SerializerMethodField()

    class Meta:
        model = UserEntry
        fields = ['entry_name', 'entry_date', 'entry_time', 'exit_name', 'exit_date', 'exit_time', 'trip_distance', 'fare']

    def get_entry_name(self, obj):
        return obj.get_location()

    def get_exit_date(self, obj):
        return obj.userexit.exit_date

    def get_exit_time(self, obj):
        return obj.userexit.exit_time

    def get_exit_name(self, obj):
        return obj.userexit.get_location()

    def get_trip_distance(self, obj):
        return obj.userexit.trip_distance

    def get_fare(self, obj):
        return obj.userexit.fare

```

Figure 109 Travel Log Serializer

```

class UserDataSerializer(serializers.ModelSerializer):

    profile_pic = serializers.SerializerMethodField()
    user_name = serializers.SerializerMethodField()
    user_status = serializers.SerializerMethodField()

    # making method field to access previously declared serializer's data
    log = serializers.SerializerMethodField()
    class Meta:
        model = Card
        fields = ['profile_pic', 'user_name', 'user_status', 'card_amount', 'log']

    def get_profile_pic(self, obj):
        profile_pic = obj.user.profile_pic.url
        return profile_pic

    def get_user_name(self, obj):
        user_name = obj.user.first_name + " " + obj.user.last_name
        return user_name

    def get_user_status(self, obj):
        return obj.user.status.status_message

    def get_log(self, obj):
        # getting trips where card_tap is true
        valid_trips_list = UserEntry.objects.filter(card_number=obj, userexit__card_tap=True).order_by('-entry_date')
        serializer = TravelLogSerializer(valid_trips_list, many=True)
        return serializer.data

```

Figure 110 User Data Serializer

```

final userContainer = FutureBuilder(
  future: userFuture,
  builder: (BuildContext context, AsyncSnapshot snapshot) {
    User user = snapshot.data;
    switch (snapshot.connectionState) {
      case ConnectionState.done:
        return ListView(children: [
          Stack(alignment: Alignment.center, children: [
            CustomPaint(
              child: Container(
                width: MediaQuery.of(context).size.width,
                height: MediaQuery.of(context).size.height / 2.75,
              ), // Container
              painter: HeaderCurvedContainer(),
            ), // CustomPaint
            Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                Padding(padding: EdgeInsets.all(40)),
                Container(
                  padding: EdgeInsets.all(10.0),
                  width: MediaQuery.of(context).size.width / 2,
                  height: MediaQuery.of(context).size.width / 2,
                  decoration: BoxDecoration(
                    border: Border.all(color: Colors.white, width: 5),
                    shape: BoxShape.circle,
                    color: Colors.white,
                    image: DecorationImage(
                      fit: BoxFit.cover,
                      image: NetworkImage(user.profilePic),
                    )), // DecorationImage // BoxDecoration
                ) // Container
              ],
            ), // Column
          ]), // Stack
          Container(
            //height: 475,
            width: double.infinity,
            margin: EdgeInsets.symmetric(horizontal: 10),
            child: Column(
              //mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                UserContainer(Column(
                  children: [
                    UserTile('Name', user.userName),
                    UserTile('Card Amount',
                      "Rs " + user.cardAmount.toString()), // UserTile
                    UserTile('Account Status', user.userStatus),
                  ],
                )), // Column // UserContainer
              ],
            ), // Column
          ), // Container
        ]);
  }
);

```

Figure 111 Travel Log Front End (1)

```
user.log.length > 0
    ? LogContainer(
        "Travel History",
        ListView.builder(
            shrinkWrap: true,
            physics: ClampingScrollPhysics(),
            itemCount: user.log.length,
            itemBuilder: (BuildContext context, int index) {
                return LogTile(user.log[index]);
            },
        )) // ListView.builder // LogContainer
    : Center(
        child: Padding(
            padding: EdgeInsets.all(30),
            child:
                Text("Use your card to view your travel data!"), // Padding
        ) // Center
    ]); // ListView
default:
    return _buildLoadingScreen();
}
}); // FutureBuilder
```

Figure 112 Travel Log Front End (2)

## Chapter 4. Testing and Analysis

### 4.1 Test Plan

Testing for the complete application is done using two approaches, unit testing and system testing. Each test cases and their outcomes have been documented and analyzed in the upcoming section.

#### 4.1.1 Unit Testing, Test Plan

In this approach, every function of the application in detailed form is tested. Optimal and suboptimal actions for the functions have been performed to check capabilities of the function to handle varying input.

The documentation of testing will be split into functions of the system. Since there are three parts (web, mobile, desktop) in the system, tests for overlapping functions will be done under the same function while being subcategorized as parts.

Standalone functions, that are only available in a single part of the system will not be categorized and varying tests for all the functionalities will be performed.

#### 4.1.2 System Testing, Test Plan

In this approach, functionality of the project, that help in completing multiple functions of the overall application are tested. This involved data passing, builds of the application along with the core permissions for the application.

Builds of all the application will be shown in the test cases while the core APIs of the system that enable other features of the application will be shown as well, which involved endpoints for the desktop application and the mobile application.

## 4.2 Unit Testing

### 4.2.1 User Signup

#### 4.2.1.1 Form Validation

Table 16 User Signup From Validation Description

Action Performed	Invalid data types for defined fields is entered.
Expected Output	Field errors should be displayed for appropriate form fields.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows a mobile application interface titled "Card Pay Register". The form has several fields: "Report" (with a file icon), "Test" (highlighted in orange), "Phone Number" (containing "+977 dddd" with a flag icon and a dropdown arrow), "Name" (containing "xyz"), "Address" (containing "..."), and "Email" (containing "..."). Below the form is a large orange button labeled "Register Account". At the bottom of the screen, there is a link "Already have an account? [Login](#)".

Figure 113 Adding Invalid Data to Fields

## Card Pay Register

Report

Test

+977  xyz

Please include an '@' in the email address. 'xyz' is missing an '@'.

Register Account

Already have an account? [Login](#)

This screenshot shows a registration form titled 'Card Pay Register'. It includes fields for 'Report' and 'Test' (both with print icons), a dropdown for phone number (+977) and a placeholder 'dddd', and an email input field containing 'xyz'. A yellow warning box with an exclamation mark states: 'Please include an '@' in the email address. 'xyz' is missing an '@''. Below the form is an orange 'Register Account' button and a link to 'Login' for existing users.

Figure 114 JavaScript Validation on Signup

## Card Pay Register

Report

Test

+977 xyz

reporttest@gmail.com

Enter Password...

Re-enter Password...

Register Account

- Enter a valid phone number (e.g. +12125552368).
- This password is too short. It must contain at least 8 characters.

Already have an account? [Login](#)

This screenshot shows a registration form titled 'Card Pay Register'. It includes fields for 'Report' and 'Test' (both with print icons), a phone number input ('+977 xyz') with a flag icon, an email input ('reporttest@gmail.com'), and two password inputs ('Enter Password...' and 'Re-enter Password...'). Below the form is an orange 'Register Account' button. A pink callout box contains two validation errors: 'Enter a valid phone number (e.g. +12125552368)' and 'This password is too short. It must contain at least 8 characters'. At the bottom is a link to 'Login' for existing users.

Figure 115 Server Side Signup Validation

#### 4.2.1.2 Database Unique Constraints (duplicate phone and email)

Table 17 Database Unique Constraint Test Description

Action Performed	Existing email and phone number in the database is entered for signup.
Expected Output	Errors should be displayed with appropriate message.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows a Django admin interface for managing accounts. On the left, there's a sidebar with categories like ACCOUNTS, API KEY PERMISSIONS, AUTH TOKEN, and AUTHENTICATION AND AUTHORIZATION. Under AUTHENTICATION AND AUTHORIZATION, there are sections for Groups, Bus travel logs, Buss, User entries, and User exits. The main content area shows a user profile for a user named Pranjal Kanel. The user has a phone number (+9779861367463) and an email (pranjaldeepkanel@gmail.com). The status is set to 'Account Verified'. A list of user permissions is visible on the right, including auth\_token, token, Can add token, Can change token, Can delete token, and Can view token. There are also contenttypes, rest\_framework, and API key permissions listed.

Figure 116 Existing Data in Database

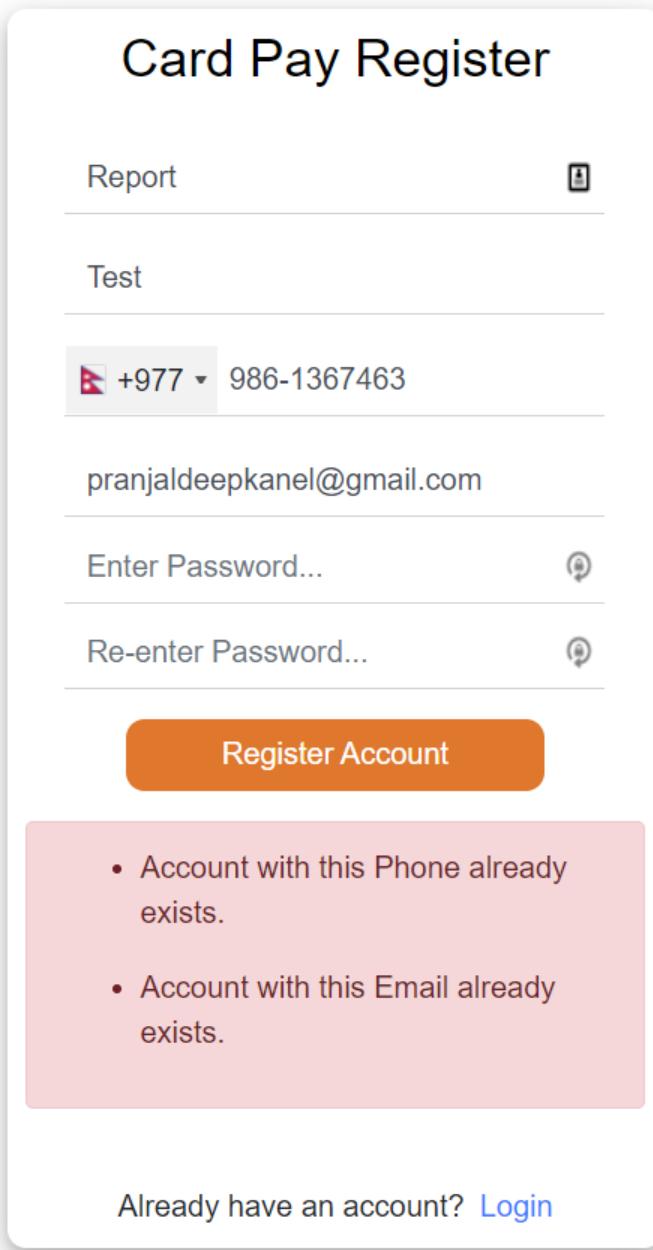


Figure 117 Error Messages for Unique Constraint Violation

#### 4.2.1.3 Valid Signup

Table 18 Signup User with Valid Data Test Description

Action Performed	Register form for the application is filled using valid data.
Expected Output	User creation with authentication of phone number should be performed.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows the 'Card Pay Register' application interface. At the top, there is a header section with the title 'Card Pay Register'. Below the header, there are two input fields: 'Report' and 'Test'. Under 'Report', there is a phone number field containing '+977 9861367460'. Under 'Test', there is an email field containing 'reporttest@gmail.com'. Below these fields are two password input fields, each consisting of a masked password ('.....') and a visibility icon. At the bottom of the screen is a large orange button labeled 'Register Account'. At the very bottom, there is a link for existing users: 'Already have an account? [Login](#)'.

Figure 118 Valid User Data in Signup Form

Verify Phone

Verify Token

Enter the code sent via SMS

Your twofact verification code is:  
712646

Figure 119 Entering Phone Verification Code

Figure 120 SMS Code on Entered Mobile Phone

Card Pay Login

+977 ▾ Phone Number...

Password...

Login

Your account has been created!

Don't have an account? [Sign Up](#)  
Forgot password? [Reset Here](#)

Figure 121 Redirect To Login Page after Account Creation

Select account to change

Action:	-----	Go	0 of 11 selected
<input type="checkbox"/>	ACCOUNT		
<input type="checkbox"/>	reporttest@gmail.com		
<input type="checkbox"/>	sachhyamcrook@gmail.com		
<input type="checkbox"/>	itsmeprajna.subedi@gmail.com		
<input type="checkbox"/>	bibek.maharjan@islingtoncollege.edu.np		
<input type="checkbox"/>	prithivi.maharjan@islingtoncollege.edu.np		
<input type="checkbox"/>	testpranjal@gmail.com		
<input type="checkbox"/>	sodip99@gmail.com		
<input type="checkbox"/>	udaya.panday@gmail.com		
<input type="checkbox"/>	sadikshyaluitel16@gmail.com		
<input type="checkbox"/>	test@gmail.com		
<input type="checkbox"/>	pranaldeepkanel@gmail.com		

Figure 122 New Signup in Database Dashboard

## 4.2.2 User Login

### 4.2.2.1 Web Application

#### 4.2.2.1.1 Invalid Login

Table 19 Invalid Login Test Description for Web Application

Action Performed	Invalid username and password are entered in the web application login.
Expected Output	Appropriate message for invalid login should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

Card Pay Login

+977 9861367463

.....

Login

Don't have an account? [Sign Up](#)  
Forgot password? [Reset Here](#)

Figure 123 Invalid Data in Web Login

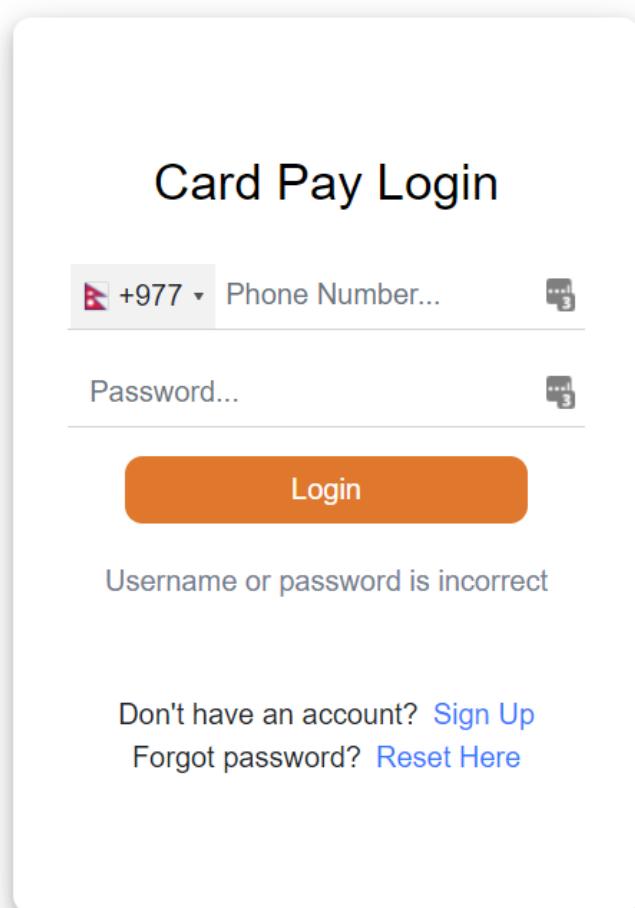


Figure 124 Login Failure Message

#### 4.2.2.1.2 Valid Login

Table 20 Valid Login Test Description for Web Application

Action Performed	Valid username and password are entered in the web application login.
Expected Output	User should be logged in and redirected to appropriate page.
Actual Output	The output was as expected.
Conclusion	Successful

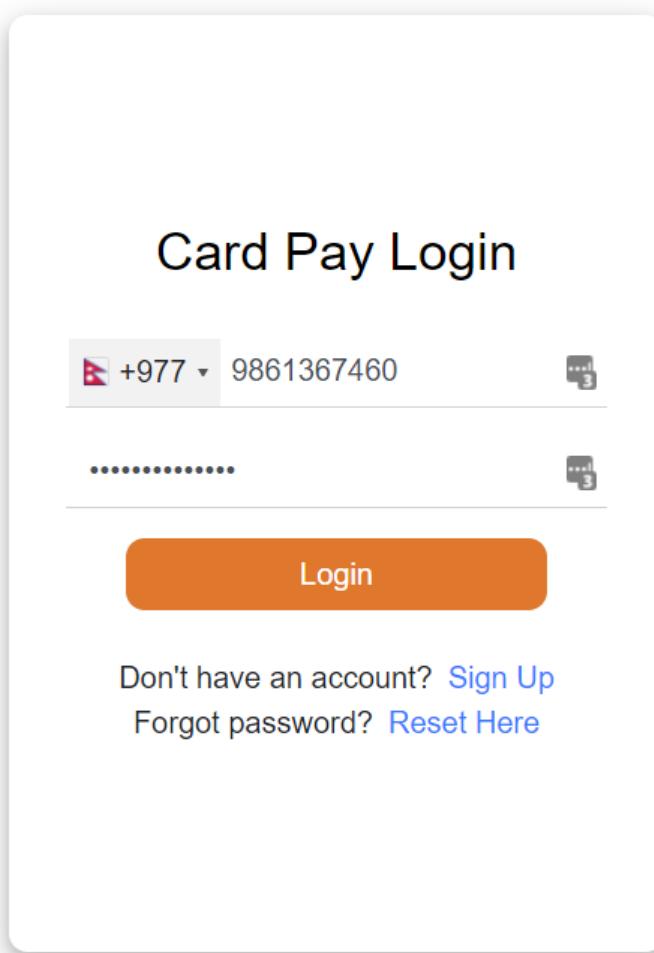


Figure 125 Valid Data Entry in Web Login

The screenshot shows a user interface for a travel dashboard. At the top, there is a dark header bar with a logo on the left and navigation links "Home", "Profile", "Topup", and "Logout" on the right. Below the header, the word "Dash Board" is displayed in a large, bold, black font. Underneath this, there are three rectangular boxes with rounded corners, each containing a title and a value: "Card Amount" (Rs 1149), "Distance Travelled" (51.83 KM), and "Total Trips" (29). A horizontal line separates this section from the main content. The main content is titled "Travel History" and features a table with six columns: "Entry Location", "Exit Location", "Entry Time", "Exit Time", "Distance (meters)", and "Cost (Rs)". The table contains five rows of data. At the bottom of the table, there is a navigation bar with page numbers (1, 2, 3, 4, 5, 6) and arrows for navigating through the pages.

Entry Location	Exit Location	Entry Time	Exit Time	Distance (meters)	Cost (Rs)
Chandan Marg, Kathmandu	Chandan Marg, Kathmandu	April 2, 2021 3:56 p.m.	April 2, 2021 3:56 p.m.	0	0
Chandan Marg, Kathmandu	Chandan Marg, Kathmandu	April 2, 2021 3:59 p.m.	April 2, 2021 3:59 p.m.	1	0
Dillibazar Pipalko Bot Bus Stop, Dilli Bazaar Rd	Bag Bazar Sadak, Kathmandu	March 12, 2021 11:45 a.m.	March 12, 2021 12:03 p.m.	1084	7
Pashupati Rd, Kathmandu	Pashupati Rd, Kathmandu	March 11, 2021 11:49 a.m.	March 11, 2021 12:05 p.m.	1724	12
Dilli Baazar Sadak, Kathmandu	Sinamangal Rd, Kathmandu	March 11, 2021 5:43 p.m.	March 11, 2021 6:03 p.m.	2194	15

« 1 2 3 4 5 6 »

Figure 126 User Redirected to Dashboard After Login

#### 4.2.2.2 Mobile Login

##### 4.2.2.2.1 Invalid Login

Table 21 Invalid Login Test Description for Mobile Application

Action Performed	Invalid username and password are entered in the mobile application login.
Expected Output	Appropriate message for invalid login should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

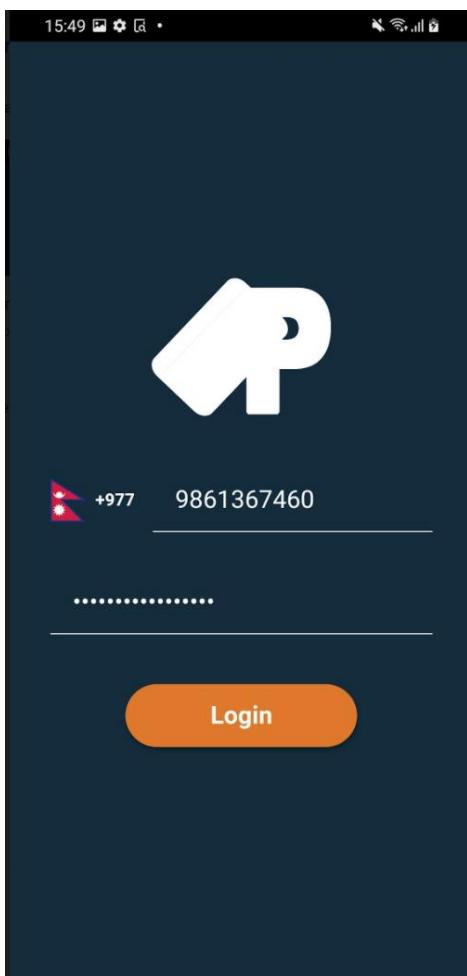


Figure 127 Invalid Data in Mobile Application

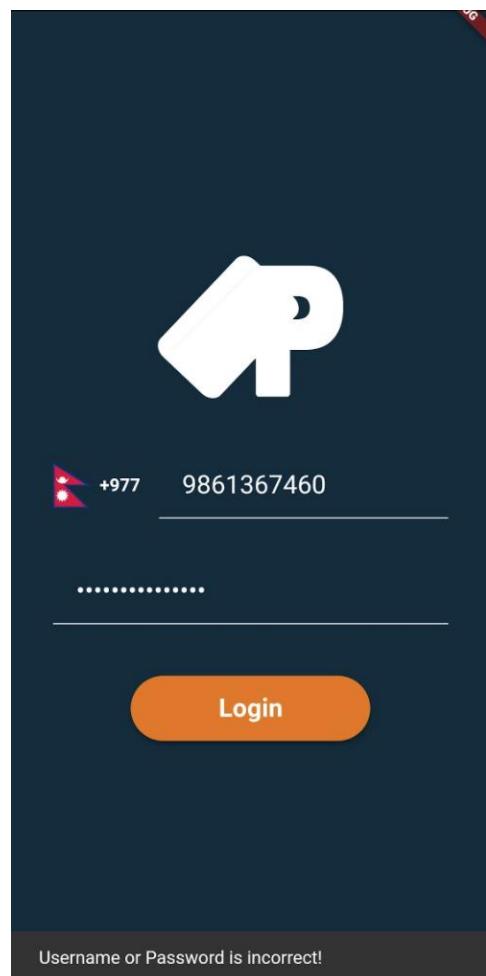


Figure 128 Invalid Login Message in Mobile App

#### 4.2.2.2.2 Valid Login

Table 22 Valid Login Test Description for Mobile Application

Action Performed	Invalid username and password are entered in the mobile application login.
Expected Output	Appropriate message for invalid login should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

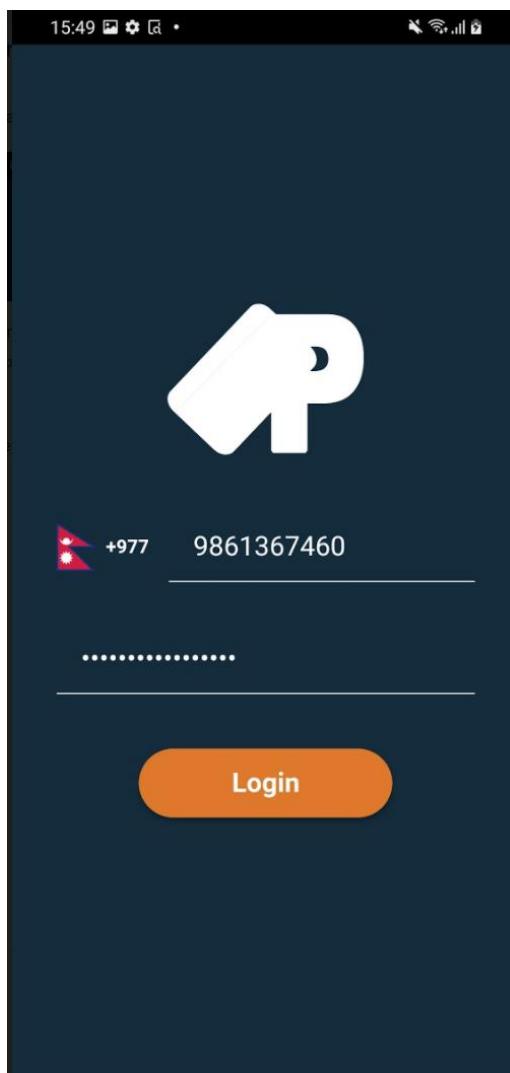


Figure 129 Valid Data Entry in Mobile Login

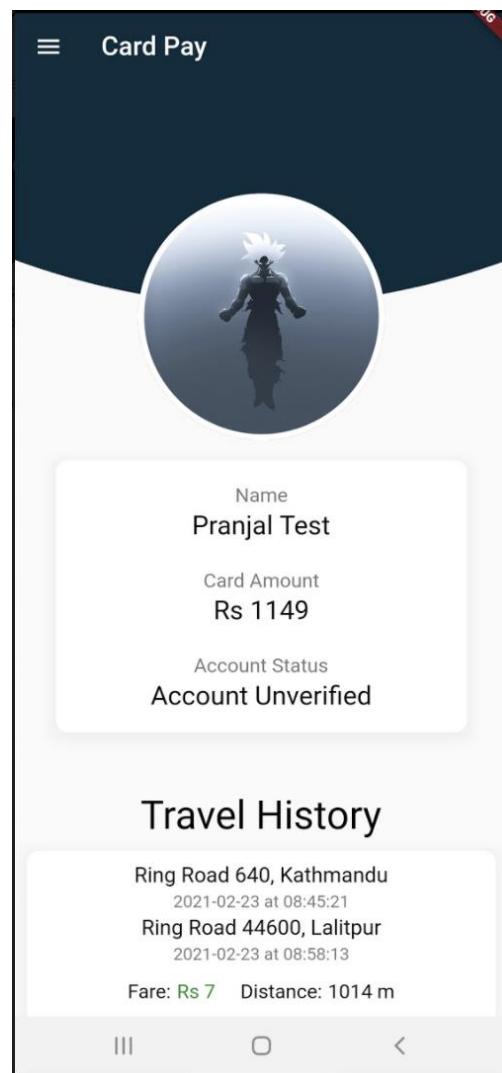


Figure 130 Redirect to Dashboard After Login

### 4.2.2.3 Desktop Login

#### 4.2.2.3.1 Invalid Login

Table 23 Invalid Login Test Description for Desktop

Action Performed	Invalid username and password are entered in the desktop application login.
Expected Output	Appropriate message for invalid login should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows a desktop application window titled "Card Writer Login". Inside, there are two text input fields labeled "Admin Id" and "Password", both of which are currently empty. Below these fields is a "Login" button. A red error message at the bottom of the window reads "Both fields must be filled".

Figure 131 Empty Field Validation on Desktop Login

The screenshot shows the same "Card Writer Login" window. The "Admin Id" field now contains the value "9861367463" and the "Password" field contains four asterisks ("\*\*\*\*"). When the "Login" button is clicked, a red error message appears below the fields stating "{Unable to log in with provided credentials.}".

Figure 132 Wrong Credential Validation on Desktop Login

#### 4.2.2.3.2 Valid Login

Table 24 Valid Login Test Description for Desktop

Action Performed	Invalid username and password are entered in the mobile application login.
Expected Output	Appropriate message for invalid login should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

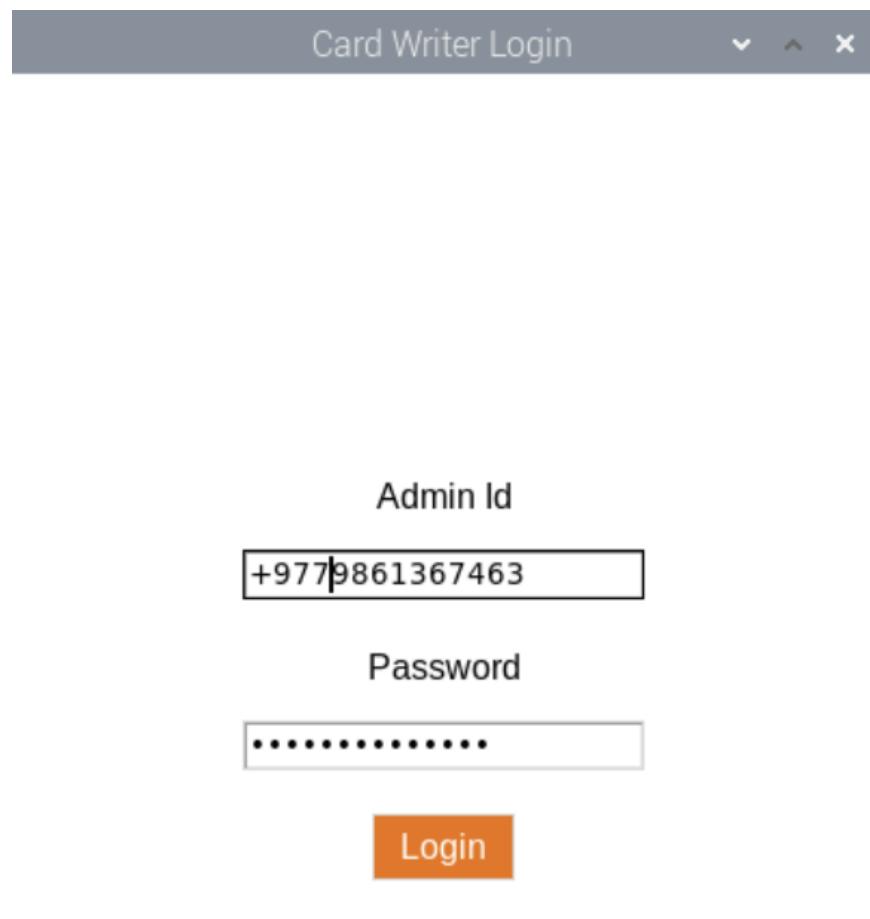


Figure 133 Valid Credentials in Desktop Login

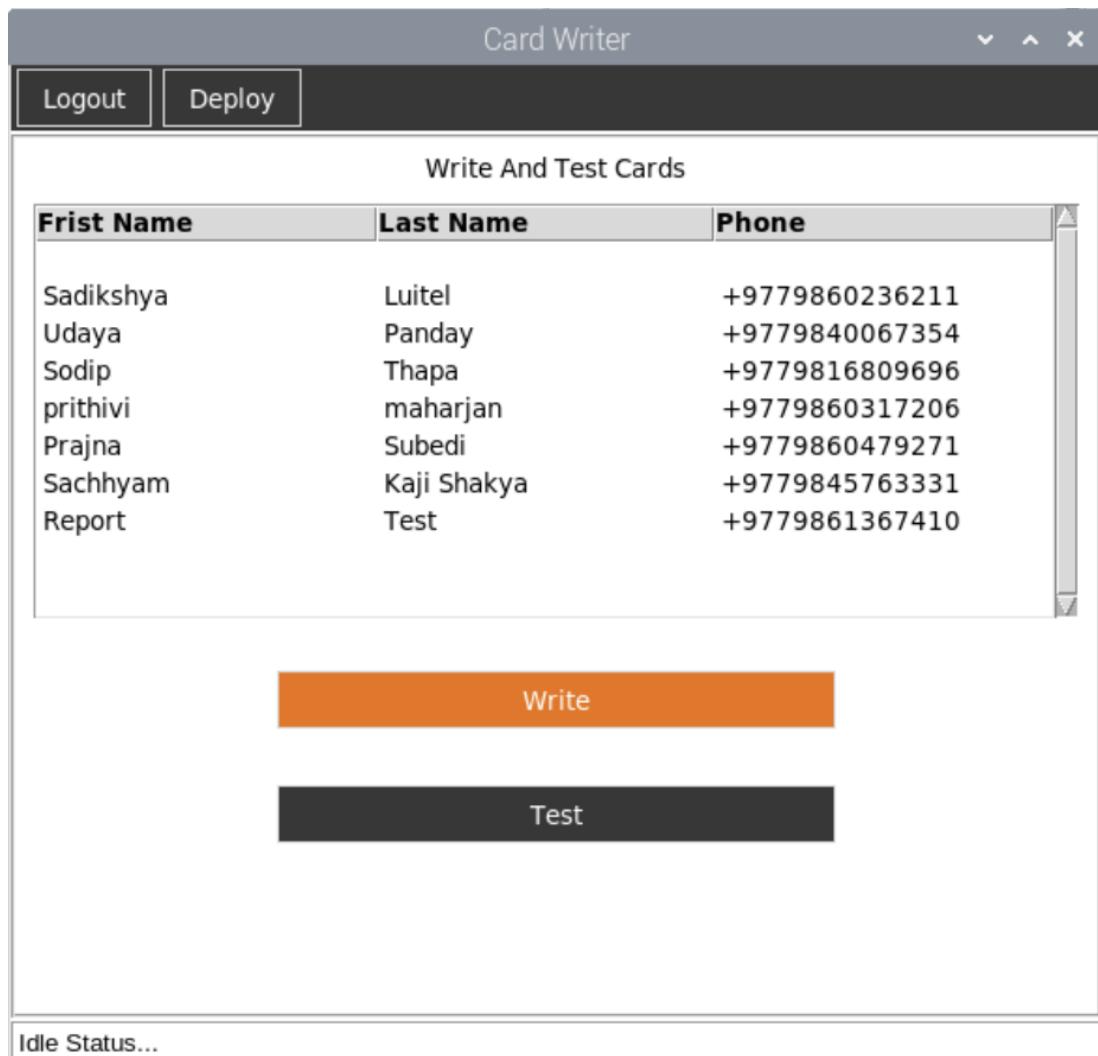


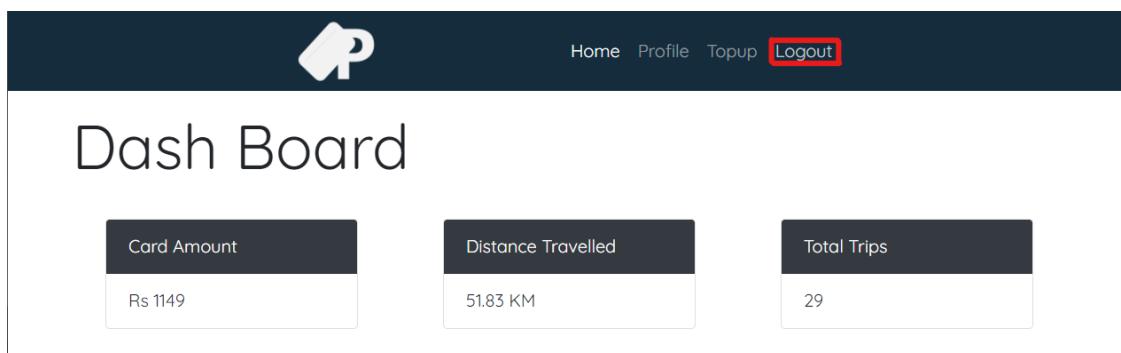
Figure 134 Desktop Login Successful

### 4.2.3 User Logout

#### 4.2.3.1 Web Logout

*Table 25 Web Logout Test Description*

Action Performed	For a logged in user account, log out button is clicked.
Expected Output	User should be redirected to login page.
Actual Output	The output was as expected.
Conclusion	Successful



*Figure 135 Pressing Web Logout Button*

*Figure 136 Login Page Redirect*

#### 4.2.3.2 Mobile Logout

Table 26 Mobile Logout Action Description

Action Performed	For a logged in user account, log out button is pressed.
Expected Output	User should be redirected to login screen.
Actual Output	The output was as expected.
Conclusion	Successful

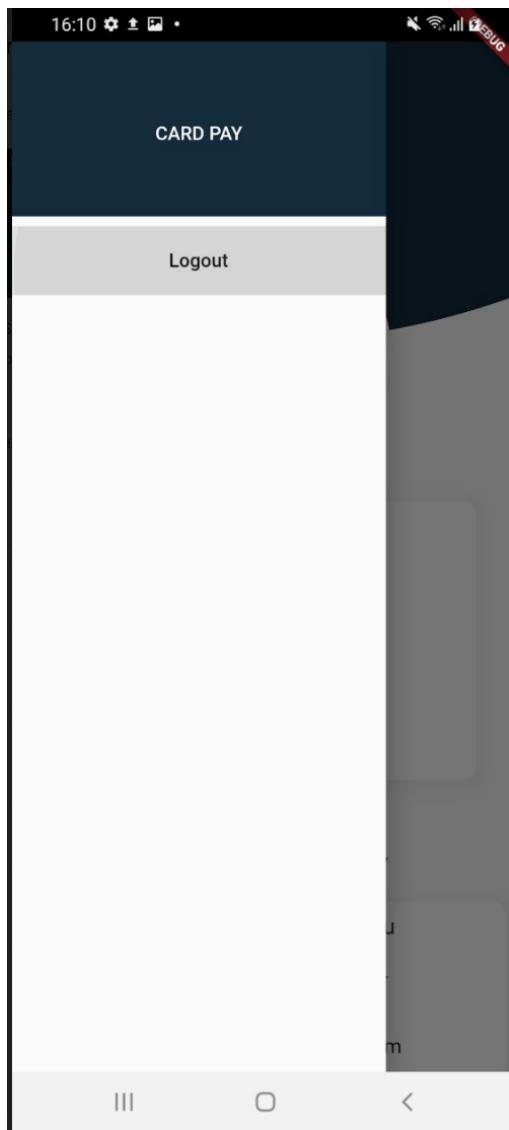


Figure 137 Mobile Logout Action

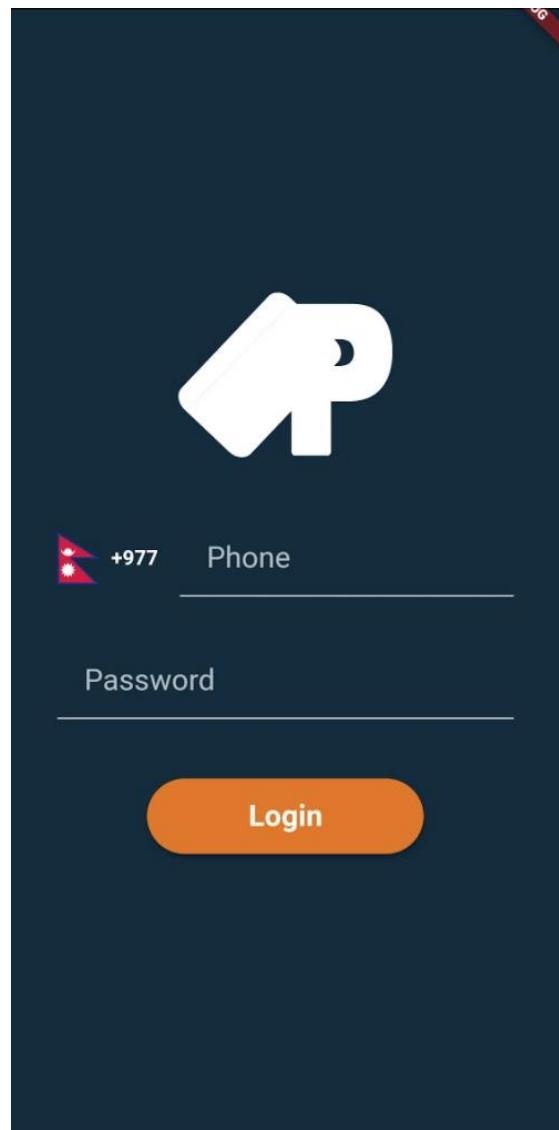
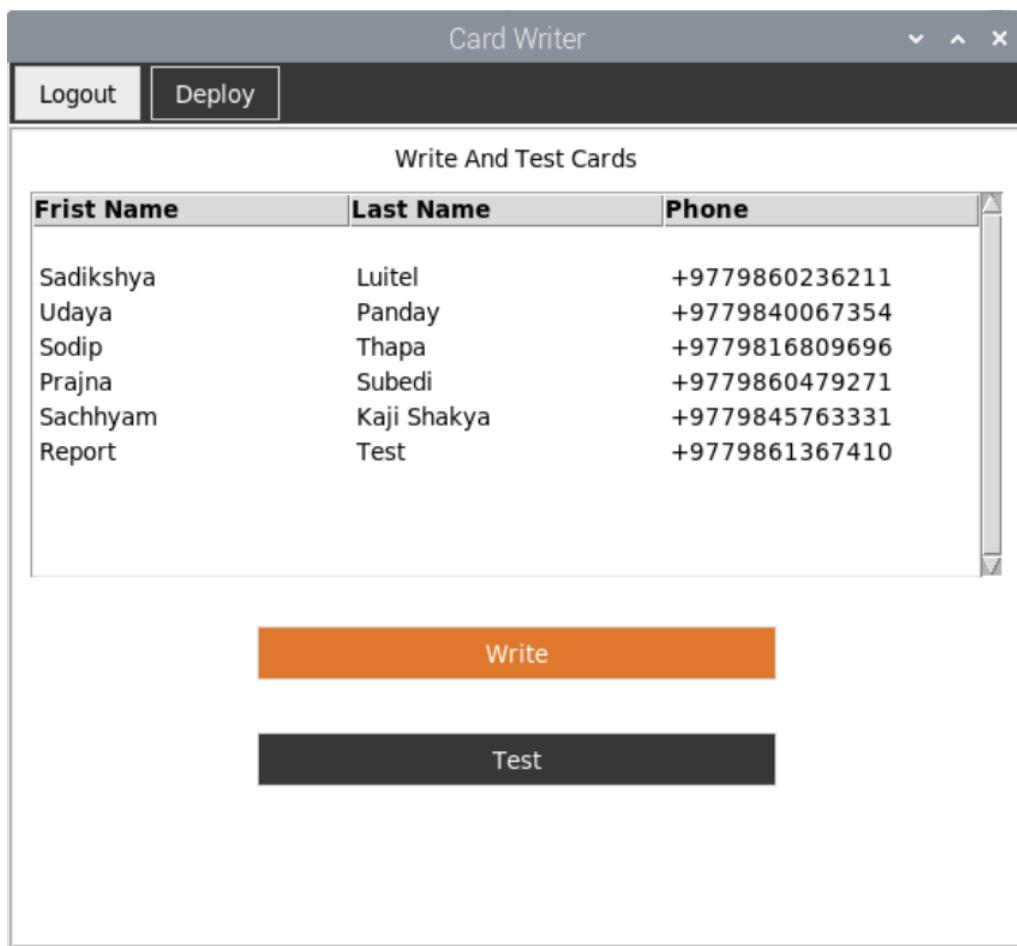


Figure 138 Mobile Redirect after Logout

#### 4.2.3.3 Desktop Logout

*Table 27 Desktop Logout Test Description*

Action Performed	For a logged in user account, log out button is clicked.
Expected Output	User should be redirected to login screen.
Actual Output	The output was as expected.
Conclusion	Successful



*Figure 139 Clicking Desktop Logout*

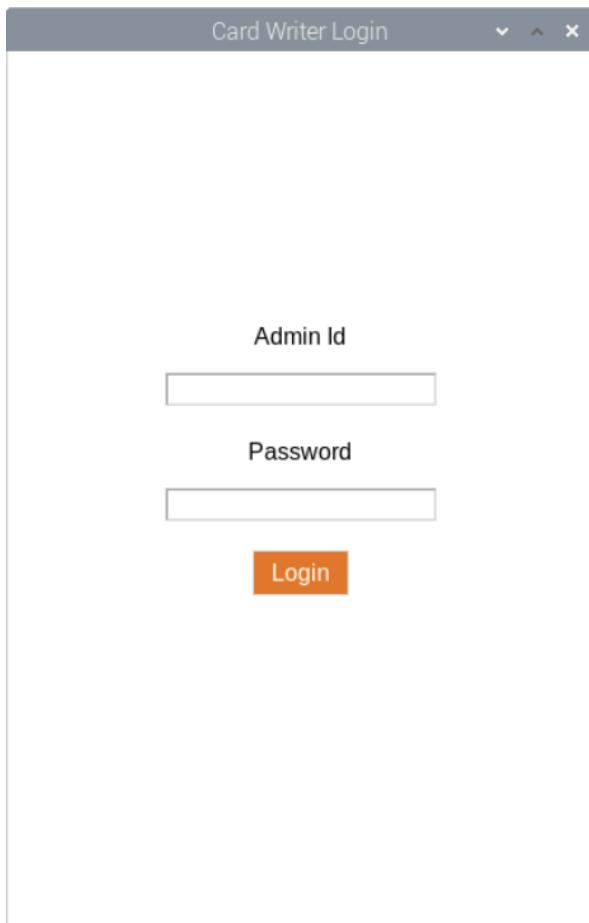


Figure 140 Successful Logout

#### 4.2.4 View Dashboard / Stats

##### 4.2.4.1 Web Application

###### 4.2.4.1.1 View Dashboard and Stats

*Table 28 Web Dashboard Stats Test Description*

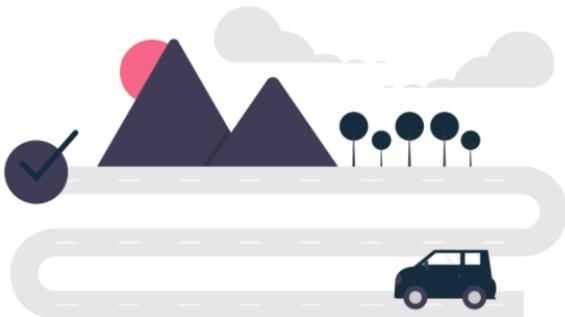
Action Performed	For a logged in user, check if all the logs and stats.
Expected Output	All valid stats and travel history for the user should be displayed in the application.
Actual Output	Every aspect of the dashboard was functioning except for walking distance stat board, which did not show the possible walking distance.
Conclusion	Test Failed

Entry Location	Exit Location	Entry Time	Exit Time	Distance (meters)	Cost (Rs)
Chandan Marg, Kathmandu	Chandan Marg, Kathmandu	April 2, 2021 3:56 p.m.	April 2, 2021 3:56 p.m.	0	0
Chandan Marg, Kathmandu	Chandan Marg, Kathmandu	April 2, 2021 3:59 p.m.	April 2, 2021 3:59 p.m.	1	0
Dillibazar Pipalkot Bus Stop, Dilli Bazaar Rd	Bog Bazar Sadak, Kathmandu	March 12, 2021 11:45 a.m.	March 12, 2021 12:03 p.m.	1084	7
Pushupati Rd, Kathmandu	Pushupati Rd, Kathmandu	March 11, 2021 11:49 a.m.	March 11, 2021 12:05 p.m.	1724	12
Dilli Bazaar Sadak, Kathmandu	Sinamangal Rd, Kathmandu	March 11, 2021 5:43 p.m.	March 11, 2021 6:03 p.m.	2194	15

*Figure 141 Web Dashboard with Travel History*

## Your Travel Statistics

## Longest and Shortest Ride



## Longest Ride

Bhaktapur Bus Park, Kathmandu  
to  
Araniko Highway, Madhyapur  
7258 meters

&lt;

&gt;

## Possible Cycling Trips

<  
Chandan Marg, Kathmandu  
to  
Chandan Marg, Kathmandu  
1 meters  
>

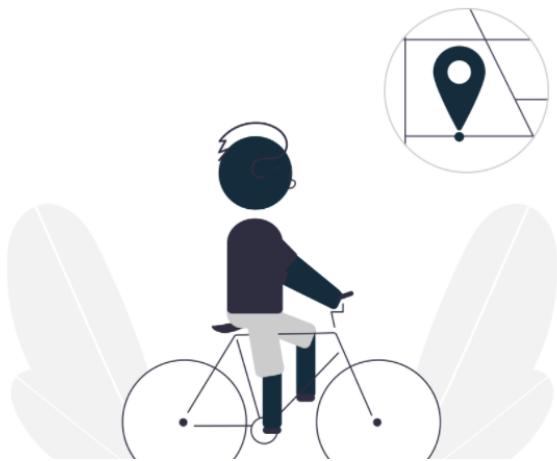


Figure 142 Travel Stats related to Travel Distances

## Possible Walking Trips



Looks like all of your trips have been long...

We could not find a trip you could walk through!

## Your Carbon Emissions

Hey Pranjali, with 51.83 Km that you have travelled,  
you have contributed in 3Kg of CO<sub>2</sub> emission.  
Planting a single tree could offset this amount of emission.  
For a better environment, consider planting a tree!



Figure 143 Travel Stats for Walking Distance and Carbon Emission

#### 4.2.4.1.2 Change Travel Log Buttons

Table 29 Web Travel Log Buttons Test

Action Performed	For a logged in user, change travel history sections in the table pagination.
Expected Output	The table should change and display older data.
Actual Output	The output was as expected.
Conclusion	Successful

## Travel History

Entry Location	Exit Location	Entry Time	Exit Time	Distance (meters)	Cost (Rs)
Chandan Marg, Kathmandu	Chandan Marg, Kathmandu	April 2, 2021 3:56 p.m.	April 2, 2021 3:56 p.m.	0	0
Chandan Marg, Kathmandu	Chandan Marg, Kathmandu	April 2, 2021 3:59 p.m.	April 2, 2021 3:59 p.m.	1	0
Dillibazar Pipalko Bot Bus Stop, Dilli Bazaar Rd	Bag Bazar Sadak, Kathmandu	March 12, 2021 11:45 a.m.	March 12, 2021 12:03 p.m.	1084	7
Pashupati Rd, Kathmandu	Pashupati Rd, Kathmandu	March 11, 2021 11:49 a.m.	March 11, 2021 12:05 p.m.	1724	12
Dilli Bazaar Sadak, Kathmandu	Sinamangal Rd, Kathmandu	March 11, 2021 5:43 p.m.	March 11, 2021 6:03 p.m.	2194	15



Figure 144 Clicking on Change Button

## Travel History

Entry Location	Exit Location	Entry Time	Exit Time	Distance (meters)	Cost (Rs)
Maiju Bahal Bus Stop, Gangalal Way	Ring Road, Kathmandu	March 6, 2021 4:42 p.m.	March 6, 2021 5:05 p.m.	1129	7
Battisputali Rd, Kathmandu	Maiju Bahal Bus Stop, Gangalal Way	March 6, 2021 8:30 a.m.	March 6, 2021 9:05 a.m.	1931	13
Araniko Highway, Kathmandu	Opposite to Civil Bank, Devkota Sadak	March 1, 2021 12:30 p.m.	March 1, 2021 12:40 p.m.	1891	13
Pashupati Rd, Kathmandu	Bishal Nagar Marg, Kathmandu	Feb. 28, 2021 4:15 p.m.	Feb. 28, 2021 4:30 p.m.	1715	12
Dilli Bazaar Sadak, Kathmandu	Sinamangal Rd, Kathmandu	Feb. 28, 2021 8:25 a.m.	Feb. 28, 2021 8:55 a.m.	1119	7

« 1 2 3 4 5 6 »

Figure 145 Changed Travel History After Button Click

#### 4.2.4.2 Mobile Application

Table 30 Mobile Dashboard Test Description

Action Performed	For a logged in user, check all travel history.
Expected Output	All valid travel history for the user should be displayed in the application.
Actual Output	The output was as expected.
Conclusion	Successful

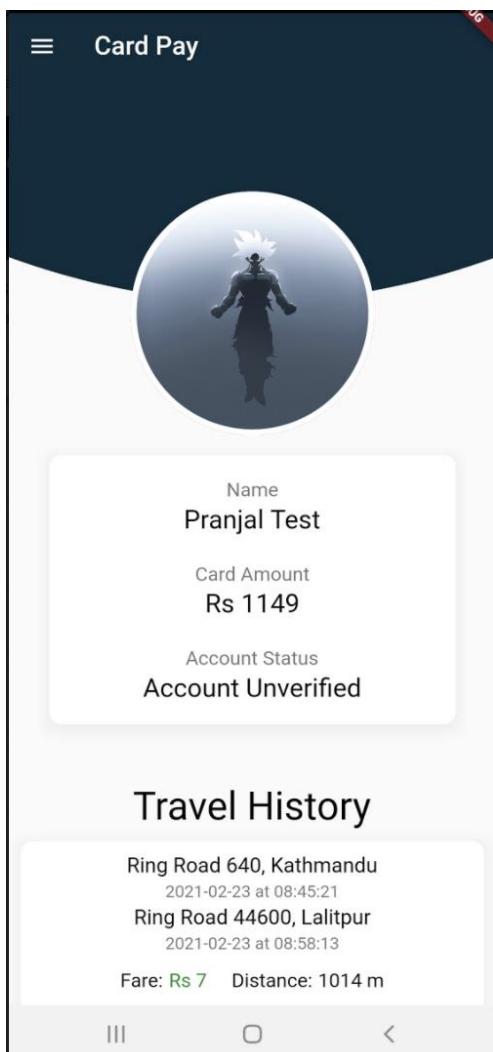


Figure 146 Mobile Dashboard for Logged In User

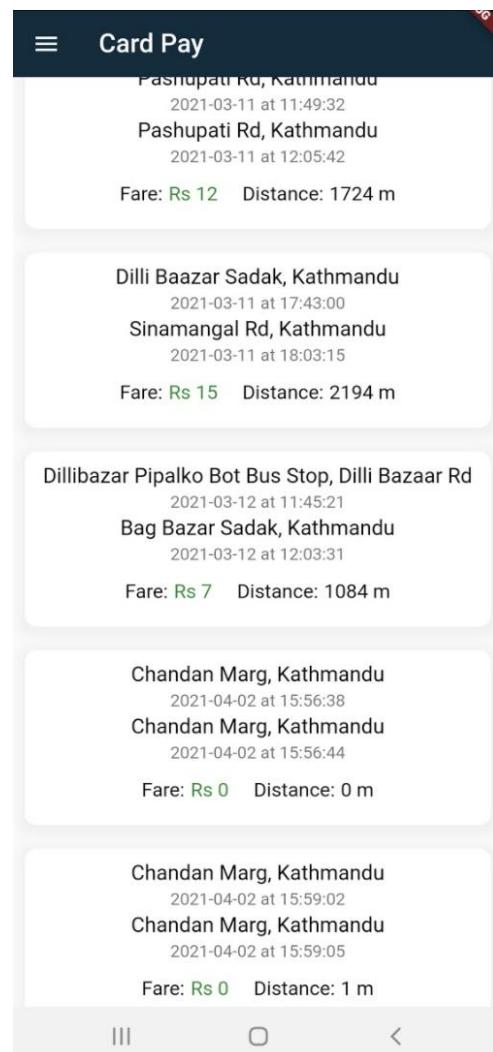


Figure 147 Mobile Travel Log

## 4.2.5 Manage Profile

### 4.2.5.1 Invalid Data Upload

Table 31 Invalid Profile Data Upload Test Description

Action Performed	For field validation, invalid data was inserted to see if input type has been validated or not.
Expected Output	An error message should be displayed before or after form submission.
Actual Output	The output was as expected.
Conclusion	Successful

Figure 148 JavaScript Field Validation

#### 4.2.5.2 Unique Key Violation

Table 32 Unique Key Constraint Violation in Profile Test Description

Action Performed	Email of the logged in user is changed to an email already existing in database.
Expected Output	Errors should be displayed with appropriate message.
Actual Output	The output was as expected.
Conclusion	Successful

## Update Profile



First Name  
Pranjal

Last Name  
Test

Email  
pranjaldeepkanel@gmail.com

Phone  
+9779861367460

[Change password here...](#)

Figure 149 Adding Existing Email in Profile

## Update Profile

First Name



Pranjal

Last Name

Test

Email

pranjaldeepkanel@gmail.com

Phone

+9779861367460

[Change password here...](#)

Account with this Email already exists.

Figure 150 Error Message for Database Constraint Violation in Update Profile

#### 4.2.5.3 Valid Data Change

*Table 33 Valid Data Update Test Description*

Action Performed	Valid data is entered into the fields and profile is updated.
Expected Output	The existing form data should change, and proper success message should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

## Update Profile



First Name  
Pranjali

Last Name  
Test2

Email

Phone  
+9779861367460

Change Profile Picture

[Change password here...](#)

*Figure 151 Valid Data Update in User Profile*

## Update Profile



First Name

Pranjali



Last Name

Test2

Email

testpranjal@gmail.com

Phone

+9779861367460

[Change Profile Picture](#)

[Browse](#)

[Save Changes](#)

[Change password \*here...\*](#)

Profile Updated

Figure 152 Profile Data Update Success

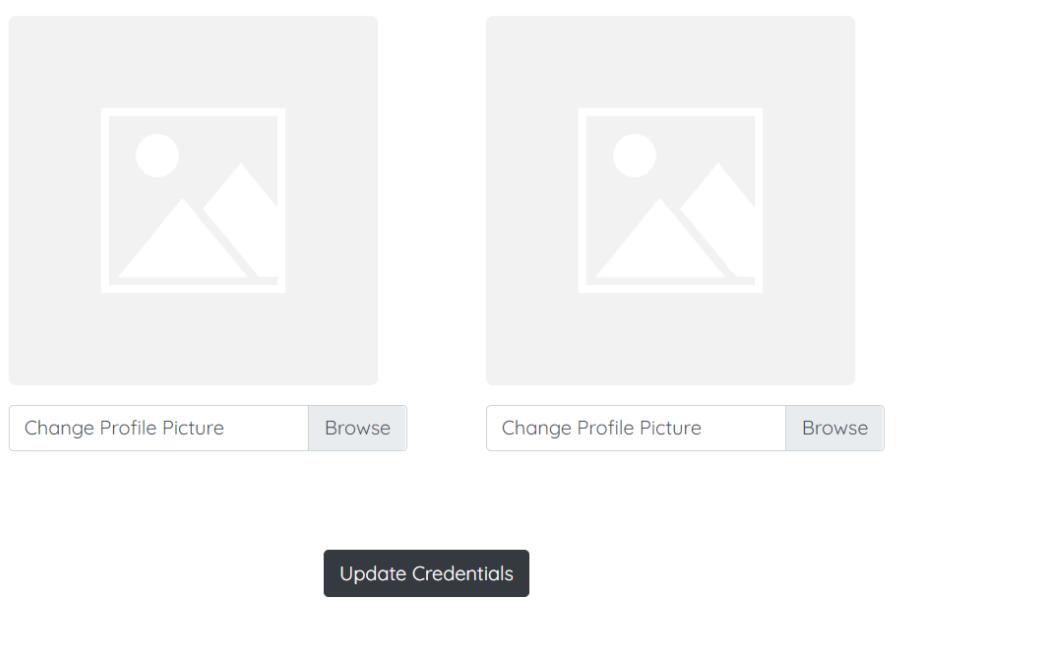
#### 4.2.5.4 Update Credentials

*Table 34 Update Credentials Test Description*

Action Performed	Default picture provided for credentials in the application is changed and submitted.
Expected Output	The photo should show preview before submitting and change the credential pictures after submitting.
Actual Output	The output was as expected.
Conclusion	Successful

## Credentials

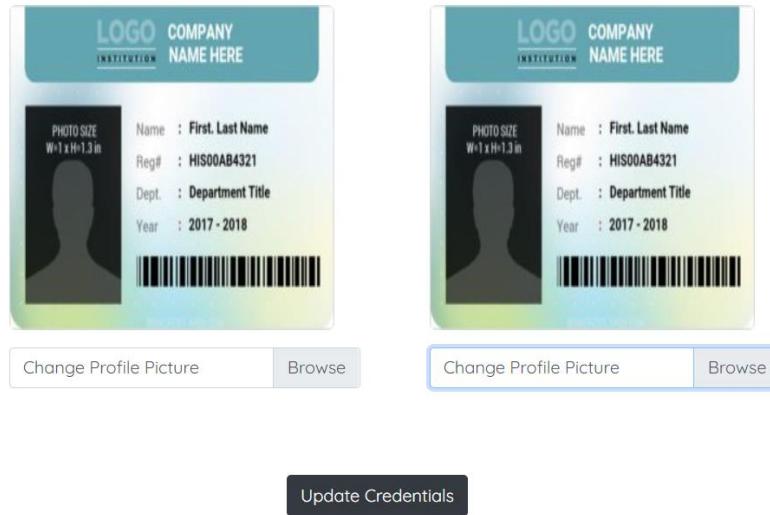
Upload the front and the back side of a government provided credential. To apply for discount, upload both sides of school/university provided ID.



*Figure 153 Default Credentials*

## Credentials

Upload the front and the back side of a government provided credential. To apply for discount, upload both sides of school/university provided ID.



*Figure 154 Uploaded Credentials Preview*

---

## Credentials

Upload the front and the back side of a government provided credential. To apply for discount, upload both sides of school/university provided ID.

Name : First. Last Name  
Reg# : HIS00AB4321  
Dept. : Department Title  
Year : 2017 - 2018

Change Profile Picture      Browse

Name : First. Last Name  
Reg# : HIS00AB4321  
Dept. : Department Title  
Year : 2017 - 2018

Change Profile Picture      Browse

**Update Credentials**

Credentials Updated

*Figure 155 Updated Credentials*

#### 4.2.5.5 Password Change

*Table 35 Password Change Test Description*

Action Performed	Change password link is clicked and form is filled to change the existing password of the user.
Expected Output	Error messages should be displayed for incorrect password whereas correct password should be changed if old password is correct.
Actual Output	The output was as expected.
Conclusion	Successful

Change Your Password

Old password

.....

New password

.....

New password confirmation

.....

Change Password

Forgot password? [Reset Here](#)

© Card Pay | 2021

*Figure 156 Entering Incorrect Old Password*

### Change Your Password

Old password

---

New password

---

New password confirmation

---

**Change Password**

- Your old password was entered incorrectly. Please enter it again.
- This password is too short. It must contain at least 8 characters.

Forgot password? [Reset Here](#)

Figure 157 Error Messages for Incorrect Passwords

### Change Your Password

Old password

---

.....

New password

---

.....

New password confirmation

---

.....

**Change Password**

Figure 158 Entering Different Passwords

### Change Your Password

Old password

New password

New password confirmation

**Change Password**

• The two password fields didn't match.

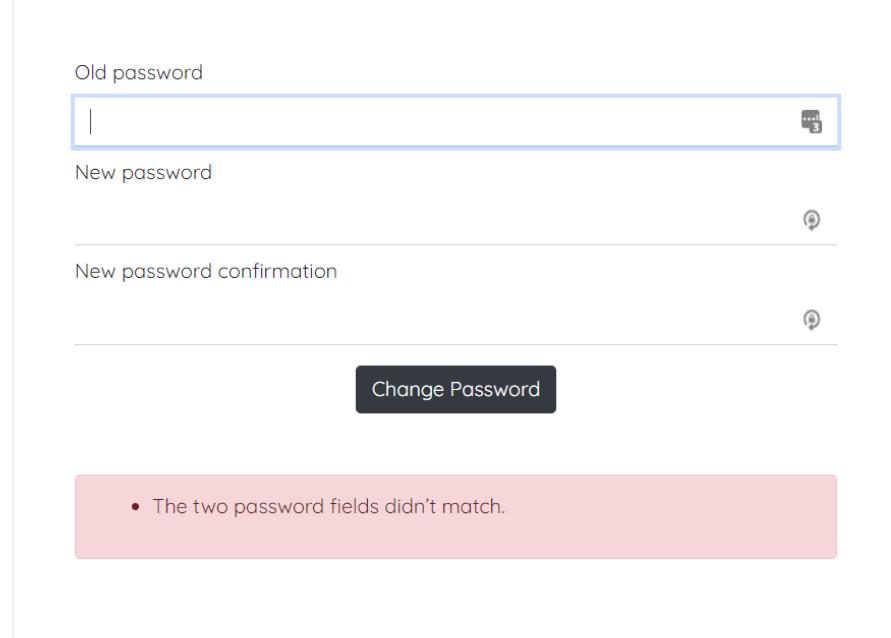


Figure 159 Error Message for Password Mismatch

### Change Your Password

Old password

New password

New password confirmation

**Change Password**

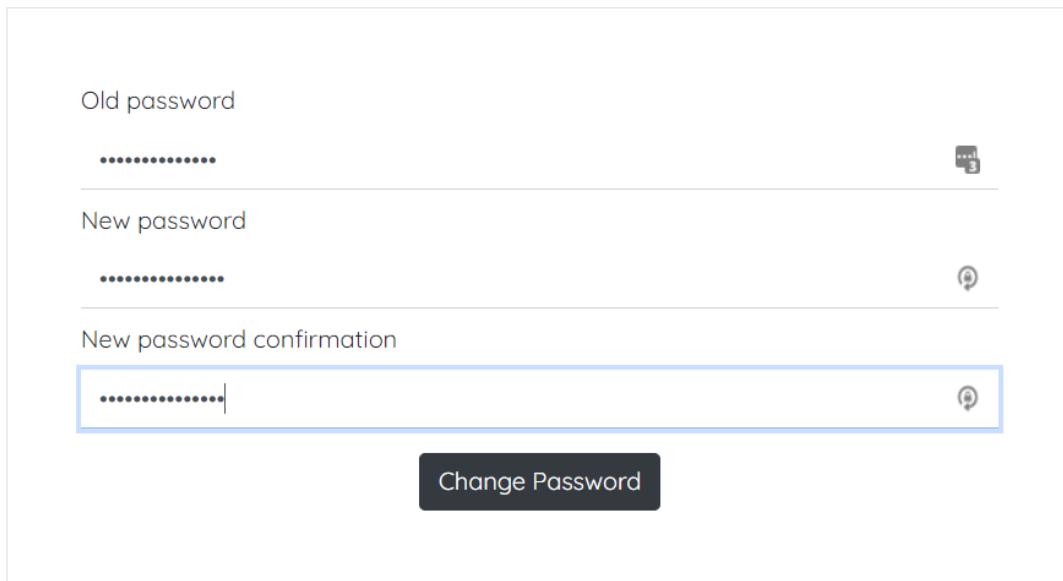


Figure 160 Entering Valid Data

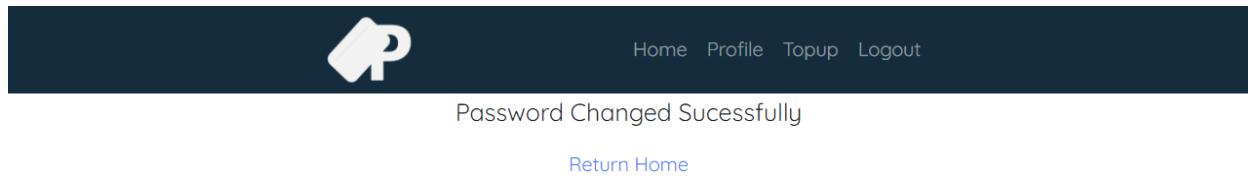


Figure 161 Password Change Message

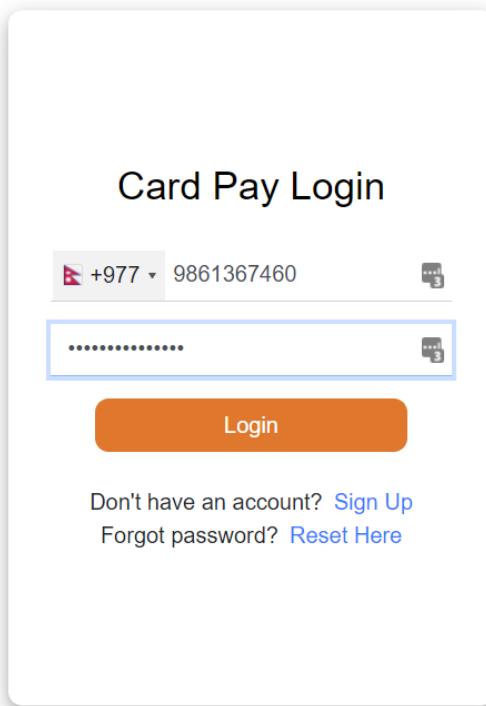


Figure 162 Logging In With New Password

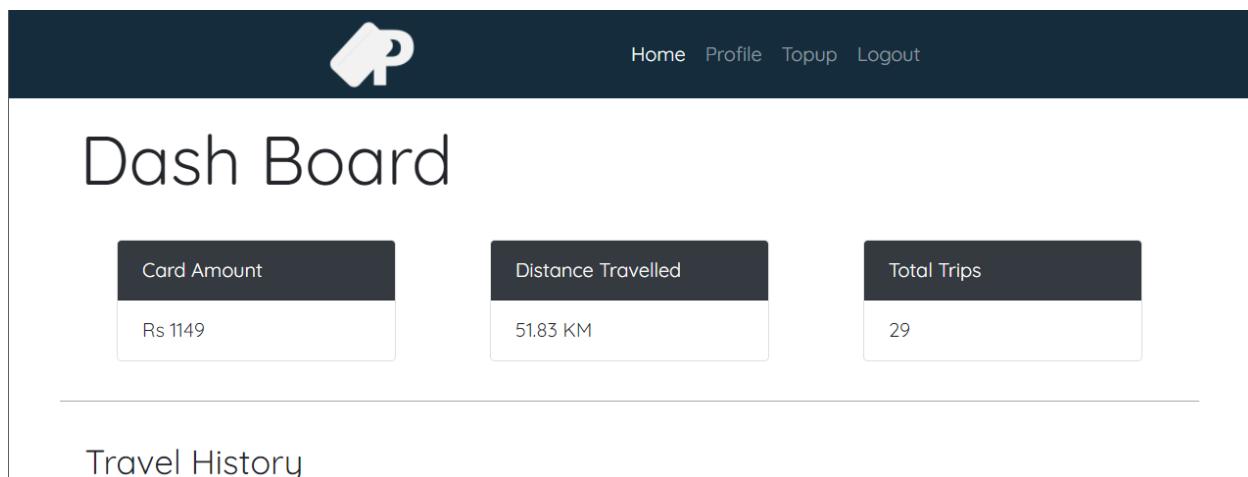


Figure 163 Login Successful

#### 4.2.5.6 Password Reset

*Table 36 Password Reset Test Description*

Action Performed	Password reset link is clicked and filled to get password reset link. The form filled should change the password.
Expected Output	Password reset link should be sent via email and the link should open a form similar to change password, which should change the current password upon submission.
Actual Output	The output was as expected.
Conclusion	Successful

**Change Your Password**

Email

Change Password

*Figure 164 Entering Valid Email for Password Reset*

## Reset Link Sent

Check your entered email. You will receive an email if it is registered with us.

[Home](#)

*Figure 165 Password Reset Link Sent Message*

Password reset on contactlesspay.herokuapp.com

Inbox ✖

 fyp@pranjalkanel.com 5:41 PM (1 minute ago)    

to me ▾

You're receiving this email because you requested a password reset for your user account at [contactlesspay.herokuapp.com.](https://contactlesspay.herokuapp.com/)

Please go to the following page and choose a new password:

<https://contactlesspay.herokuapp.com/reset/MTY/akw15w-58260d264e71fa77d6e2a32c5a3e3518/>

Your username, in case you've forgotten: +9779861367463

Thanks for using our site!

The [contactlesspay.herokuapp.com](https://contactlesspay.herokuapp.com) team

Figure 166 Password Reset Link via Email

## Reset Your Password

New password

---

New password confirmation

---



Figure 167 Password Reset Form

## Reset Your Password

The screenshot shows a password reset interface. At the top, the title "Reset Your Password" is displayed. Below it are two input fields labeled "New password" and "New password confirmation". Both fields are currently empty, indicated by placeholder text and small icon indicators. A large "Change Password" button is positioned below the input fields.

Figure 168 Invalid Password Entry in Reset

## Reset Your Password

The screenshot shows a password reset interface. It features two input fields for "New password" and "New password confirmation", both of which are empty. Below these fields is a large "Change Password" button. A prominent red error message box at the bottom contains the text "• The two password fields didn't match.", indicating that the user must enter matching passwords.

Figure 169 Error Message for Invalid Password in Reset

New password

.....

New password confirmation

.....

Change Password

Figure 170 Resetting Password with Proper Data

# Password Changed!

Your password has been changed successfully, you can now login.

Login

Figure 171 Password Change Message

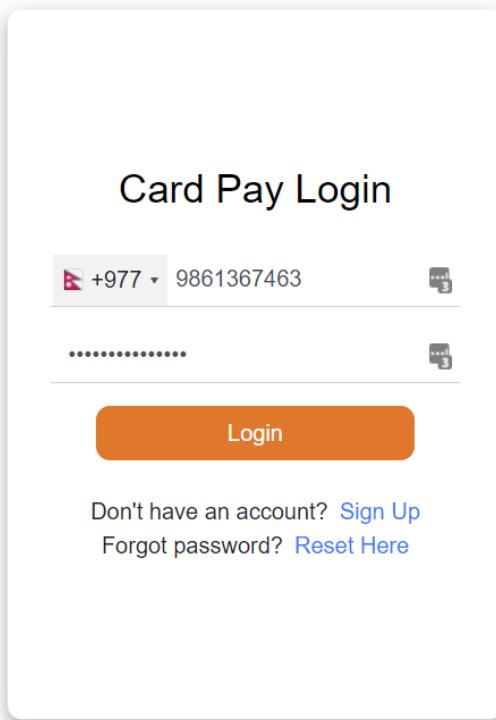


Figure 172 Logging In With New Password

The image shows the 'Admin Dash' interface. At the top, there is a dark header bar with a logo and navigation links: 'Home', 'Invalid Trips', and 'Logout'. Below the header, the title 'Admin Dash' is displayed. There are three cards: 'Accounts To Verify' (1), 'Foul Trips' (5), and 'Users Last 30 Days' (2). A horizontal line separates this from the 'User Details' section. The 'User Details' section includes a table with columns: 'Unverified', 'Resubmission Required', 'Pending Review', and 'Verified'. The table lists five users:

First Name	Last Name	Phone	Action
Sodip	Thapa	+9779816809696	<a href="#">View</a>
prithivi	maharjan	+9779860317206	<a href="#">View</a>
Prajna	Subedi	+9779860479271	<a href="#">View</a>
Sachhyam	Koji Shakya	+9779845763331	<a href="#">View</a>
Report	Test	+9779861367410	<a href="#">View</a>

Figure 173 User Logged In

#### 4.2.6 Top-up (Load Card)

These tests are carried out when a user's card amount is Rs 100.

##### 4.2.6.1 Invalid Amount

*Table 37 Load Invalid Amount on Card Test Description*

Action Performed	On paying amount, a string is entered first and then negative amount is entered.
Expected Output	Appropriate error message should be shown to users that should clarify the amount to be entered.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows a mobile application interface. At the top, it says "Enter Topup Amount". Below that is an input field containing the text "string". At the bottom is a dark button labeled "Proceed To Pay". A message at the bottom of the screen states "Amount should be positive number".

Figure 174 String Validation for Payment

The screenshot shows a mobile application interface. At the top, it says "Enter Topup Amount". Below that is an input field containing the text "-20". At the bottom is a dark button labeled "Proceed To Pay". A message at the bottom of the screen states "Amount should be atleast Rs 15".

Figure 175 Negative Amount Validation for Payment

#### 4.2.6.2 Exceeding Payment Limit

*Table 38 Higher Payment Amount Test Description*

Action Performed	Amount exceeding the payment limit for an unverified user is entered.
Expected Output	Appropriate message with the limit that can be payed should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows a white rectangular input field containing the number "2000". Below the input field is a dark grey button labeled "Proceed To Pay". At the bottom of the screen, there is a message in a light blue font that reads: "Please verify account to load more than Rs 150".

*Figure 176 Exceeding Payment Limit*

The screenshot shows a white rectangular input field containing the number "70". Below the input field is a dark grey button labeled "Proceed To Pay". At the bottom of the screen, there is a message in a light blue font that reads: "Verify account to raise topup limit Possible Load: 50".

*Figure 177 Exceeding Top-up Amount Limit*

#### 4.2.6.3 Right Amount

Table 39 Right Payment Amount Test Description

Action Performed	Right payment amount for an unverified user is entered.
Expected Output	A payment gateway that allows for payment should be opened.
Actual Output	The output was as expected.
Conclusion	Successful

The image shows a mobile application screen with a light gray background. At the top center, the text "Enter Topup Amount" is displayed in a large, dark font. Below this, there is a horizontal text input field containing the number "20". Underneath the input field is a dark rectangular button with the white text "Proceed To Pay". The overall design is clean and modern.

Figure 178 Entering Valid Amount

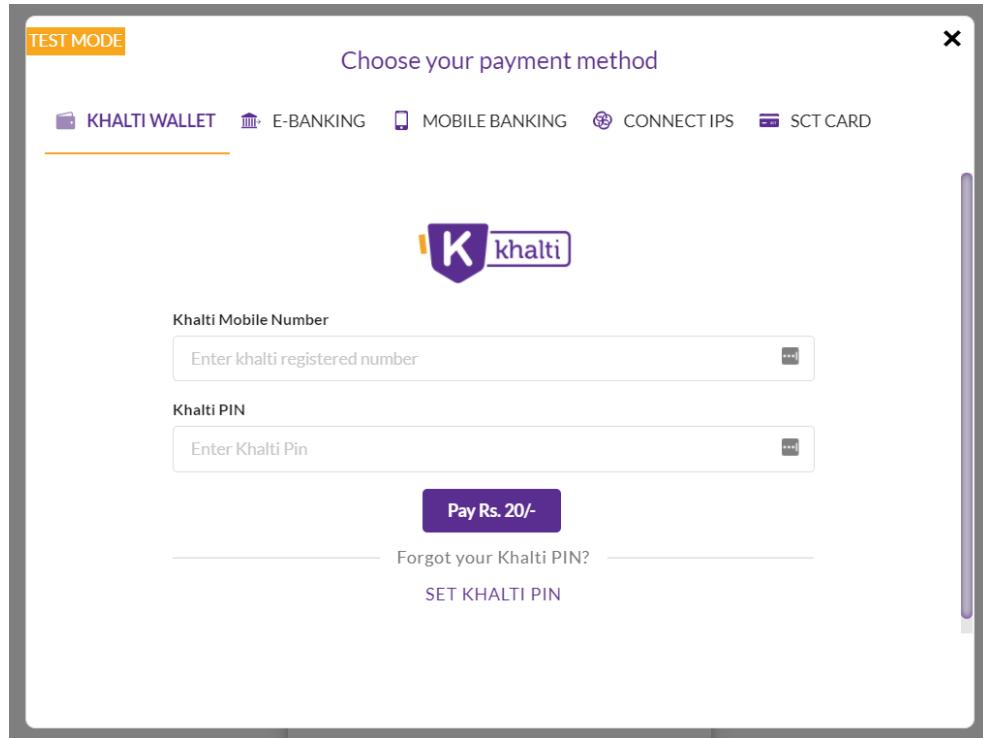


Figure 179 Payment Portal with Entered Amount

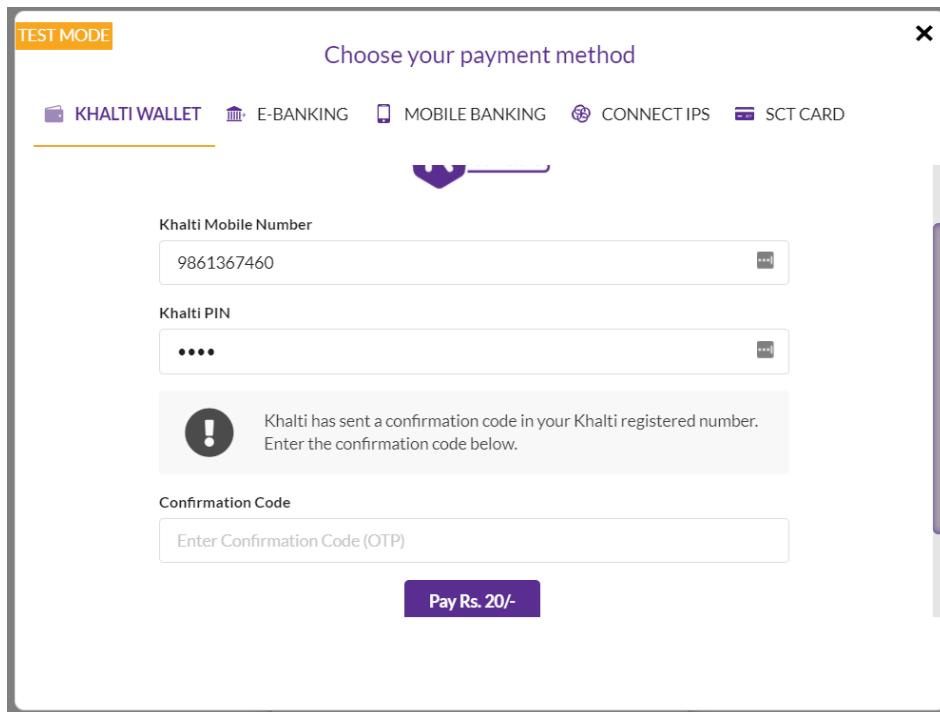


Figure 180 Confirmation Code for Verification

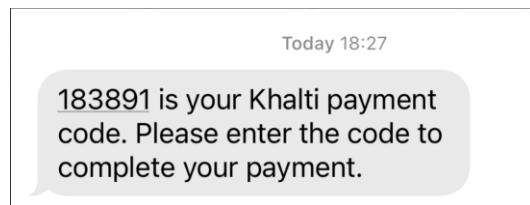


Figure 181 Payment Confirmation Code

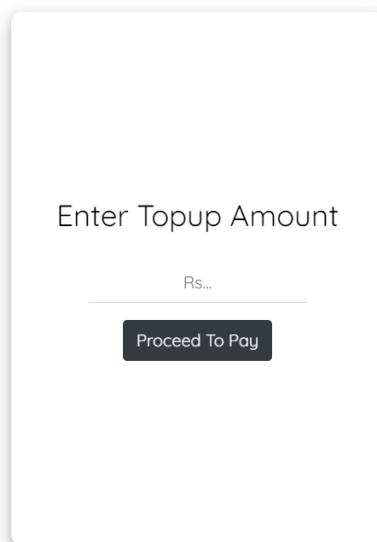
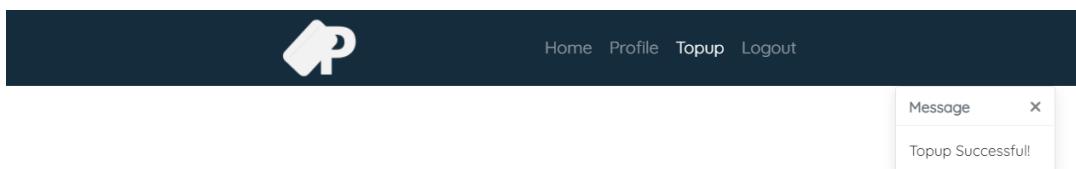


Figure 182 Top-up Successful Message

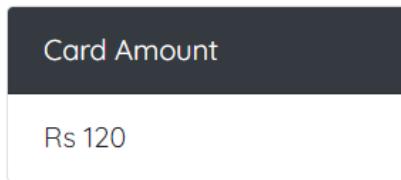


Figure 183 Updated Card Amount

## 4.2.7 Manage Users

### 4.2.7.1 View User

*Table 40 View User Test Description*

Action Performed	Table with navigation pane on dashboard is navigated and a user is viewed.
Expected Output	Various types of users should be displayed in different panes, user details should be shown when a user is viewed.
Actual Output	The output was as expected.
Conclusion	Successful

## User Details

Unverified	Resubmission Required	Pending Review	Verified	
First Name	Last Name	Phone		Action
prithivi	maharjan	+9779860317206		<a href="#">View</a>
Prajna	Subedi	+9779860479271		<a href="#">View</a>
Sachhyam	Kaji Shakya	+9779845763331		<a href="#">View</a>
Report	Test	+9779861367410		<a href="#">View</a>

*Figure 184 Unverified Users*

Unverified	Resubmission Required	Pending Review	Verified	
First Name	Last Name	Phone		Action
test	1	+9779861234567		<a href="#">View</a>

*Figure 185 Resubmission Required Users*

Unverified	Resubmission Required	Pending Review	Verified
First Name	Last Name	Phone	Action
Sadikshya	Luitel	+9779860236211	<a href="#">View</a>
Udaya	Panday	+9779840067354	<a href="#">View</a>
Sodip	Thapa	+9779816809696	<a href="#">View</a>
Pranjali	Test2	+9779861367460	<a href="#">View</a>
Bibek	Maharjan	+9779841877861	<a href="#">View</a>

*Figure 186 Verified Users*

Unverified	Resubmission Required	Pending Review	Verified
First Name	Last Name	Phone	Action
Pranjali	Test2	+9779861367460	<a href="#">View</a>

*Figure 187 Users with Pending Review*

#### 4.2.7.2 Search User

Table 41 Search User Test Description

Action Performed	Search fields provided on home page are tested with matching and unmatching data.
Expected Output	Relevant user should be shown if search criteria matches.
Actual Output	The output was as expected.
Conclusion	Successful

#### Search Customer

Phone	Email	Status
-----	-----	Search

First Name	Last Name	Phone	Email	Status	Action
Bibek	Maharjan	+9779841877861	bibek.maharjan@islingtoncollege.edu.np	Account Verified	<a href="#">View</a>
prithivi	maharjan	+9779860317206	prithivi.maharjan@islingtoncollege.edu.np	Account Unverified	<a href="#">View</a>
Sodip	Thapa	+9779816809696	sodip99@gmail.com	Account Verified	<a href="#">View</a>
Sadikshya	Luitel	+9779860236211	sadikshyaluitel16@gmail.com	Account Verified	<a href="#">View</a>
test	1	+9779861234567	test@gmail.com	Needs Resubmission	<a href="#">View</a>
Prajna	Subedi	+9779860479271	itsmeprajna.subedi@gmail.com	Account Unverified	<a href="#">View</a>
Report	Test	+9779861367410	reporttest@gmail.com	Account Unverified	<a href="#">View</a>

Figure 188 Customer Search Field

Phone	Email	Status	
xyz	-----	v	<button>Search</button>

First Name	Last Name	Phone	Email	Status	Action

Figure 189 Unmatching Search Field

Phone	Email	Status	
977	pranjal	Review Pending	v <button>Search</button>

First Name	Last Name	Phone	Email	Status	Action
Pranjali	Test2	+9779861367460	testpranjal@gmail.com	Review Pending	<a href="#">View</a>

Figure 190 Matching Search Field

#### 4.2.7.3 Change User Status / Give Discount

Table 42 User Status Change Test Description

Action Performed	Status of user is changed along with the discounted card attribute.
Expected Output	Status for the user should be changed along with discount details for the user.
Actual Output	The output was as expected.
Conclusion	Successful

Unverified	Resubmission Required	Pending Review	Verified
First Name	Last Name	Phone	Action
Pranjali	Test2	+9779861367460	<a href="#">View</a>

Figure 191 Viewing Pending Review User

Pranjali Test2



Customer Status: Review Pending

Change Status

--Change Status-- ▾

Discount  
Card

[Save](#)

Contact Pranjali

Phone +9779861367460

Email testpranjal@gmail.com

Figure 192 Account Review Form

Phone:	+9779861367460
Email:	testpranjal@gmail.com
First name:	Pranjali
Last name:	Test2
Profile pic:	Currently: user_25/profile_pic/933854.jpg <input type="checkbox"/> Clear Change: <input type="button" value="Choose File"/> No file chosen
Front credential:	Currently: user_25/front_cred/Staff-ID-Card-8-CRC-300x189.jpg <input type="checkbox"/> Clear Change: <input type="button" value="Choose File"/> No file chosen
Back credential:	Currently: user_25/back_cred/Staff-ID-Card-8-CRC-300x189.jpg <input type="checkbox"/> Clear Change: <input type="button" value="Choose File"/> No file chosen
Status:	Review Pending <input type="button" value="▼"/>

Figure 193 Current User Verification Status

## Change card

User:	testpranjal@gmail.com <input type="button" value="▼"/>
Card number:	Z977986136746002n17n2021
<input checked="" type="checkbox"/> Card written	
Card write date:	2021-04-02 <input type="button" value="Today   Calendar icon"/>
Note: You are 5.75 hours ahead of server time.	
<input checked="" type="checkbox"/> Card delivered	
<input type="checkbox"/> Is discounted	
<input type="checkbox"/> Is disabled	
Card amount:	120

Figure 194 Current Card Discount Status

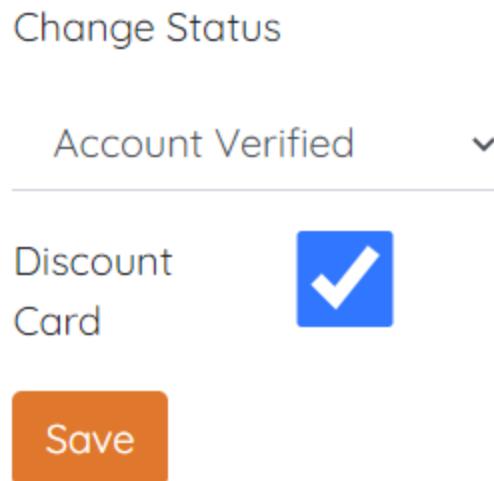


Figure 195 Changing User Status

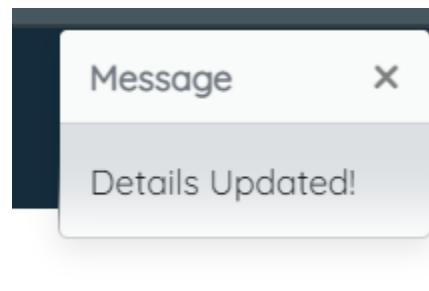


Figure 196 Update Confirmation

Phone:	+9779861367460
Email:	testpranjal@gmail.com
First name:	Pranjali
Last name:	Test2
Profile pic:	Currently: user_25/profile_pic/933854.jpg <input type="checkbox"/> Clear Change: <input type="button" value="Choose File"/> No file chosen
Front credential:	Currently: user_25/front_cred/Staff-ID-Card-8-CRC-300x189.jpg <input type="checkbox"/> Clear Change: <input type="button" value="Choose File"/> No file chosen
Back credential:	Currently: user_25/back_cred/Staff-ID-Card-8-CRC-300x189.jpg <input type="checkbox"/> Clear Change: <input type="button" value="Choose File"/> No file chosen
Status:	Account Verified <input type="button" value="▼"/> <input type="button" value="Pencil"/> <input type="button" value="Plus"/> <input type="button" value="X"/>

Figure 197 Account Status Changed

User:	testpranjal@gmail.com <input type="button" value="▼"/> <input type="button" value="Pencil"/> <input type="button" value="Plus"/>
Card number:	Z977986136746002n17n2021
<input checked="" type="checkbox"/> Card written	
Card write date:	2021-04-02 <input type="button" value="Today   Calendar"/> Note: You are 5.75 hours ahead of server time.
<input checked="" type="checkbox"/> Card delivered	
<input checked="" type="checkbox"/> Is discounted	<input type="button" value="Red Line"/>
<input type="checkbox"/> Is disabled	
Card amount:	120

Figure 198 Card Discounted

#### 4.2.7.4 View Invalid Trips

Table 43 View Invalid Test Description

Action Performed	Invalid trips option is clicked in the navigation pane and invalid trips for a given user is viewed.
Expected Output	Users with invalid trip should be displayed with their entry point details.
Actual Output	The output was as expected.
Conclusion	Successful

#### Invalid Trips

Users without exit taps

First Name	Last Name	Phone	Email	Action
prithivi	maharjan	+9779860317206	prithivi.maharjan@islingtoncollege.edu.np	<a href="#">View</a>
Pranjali	Test2	+9779861367460	testpranal@gmail.com	<a href="#">View</a>
Bibek	Maharjan	+9779841877861	bibek.maharjan@islingtoncollege.edu.np	<a href="#">View</a>

Figure 199 Invalid Trips Dashboard

#### Bibek Maharjan

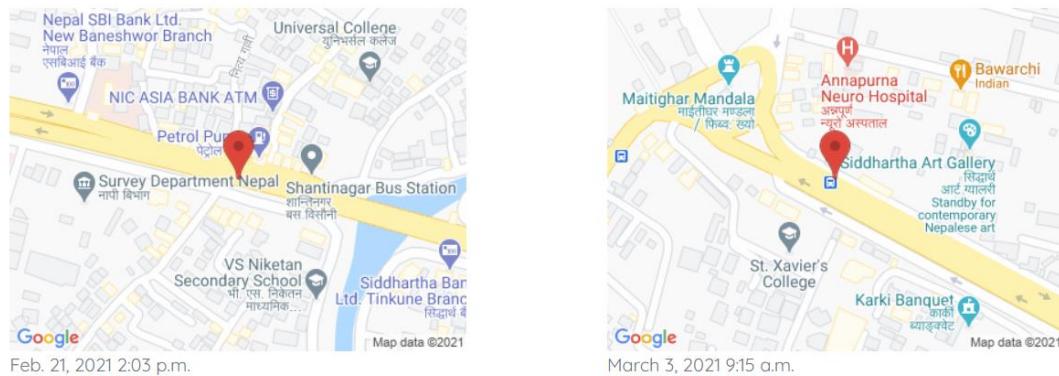


Figure 200 Entry Points for Invalid Trips



## 4.2.8 Write Card

### 4.2.8.1 Initial Write

*Table 44 Initial Write Test Description*

Action Performed	A user is selected from the given table and the card is placed on the reader.
Expected Output	The card number for the given user should be written on the card.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows a software application window titled "Card Writer". At the top, there are "Logout" and "Deploy" buttons. Below the title bar is a toolbar with three small icons. The main area contains a table titled "Write And Test Cards" with columns for "First Name", "Last Name", and "Phone". The table lists several entries. At the bottom of the application window, there are two large buttons: "Write" (light gray background) and "Test" (dark gray background). A yellow status bar at the very bottom displays the message "No row selected".

First Name	Last Name	Phone
Sadikhya	Luitel	+9779860236211
Udaya	Panday	+9779840067354
Sodip	Thapa	+9779816809696
prithivi	maharjan	+9779860317206
Prajna	Subedi	+9779860479271
Sachhyam	Kaji Shakya	+9779845763331
Report	Test	+9779861367410

*Figure 201 Write Without Selecting User*

Write And Test Cards		
Frist Name	Last Name	Phone
Sadikshya	Luitel	+9779860236211
Udaya	Panday	+9779840067354
Sodip	Thapa	+9779816809696
prithivi	maharjan	+9779860317206
Prajna	Subedi	+9779860479271
Sachhyam	Kaji Shakya	+9779845763331
Report	Test	+9779861367410

No row selected

[Write](#)

[Test](#)

Figure 202 User Selection to Write Card

User:

---

Card number:

---

Card written

---

Card write date:  Today |   
Note: You are 5.75 hours ahead of server time.

---

Card delivered

---

Is discounted

---

Is disabled

---

Card amount:

Figure 203 Current Card Status of Selected User

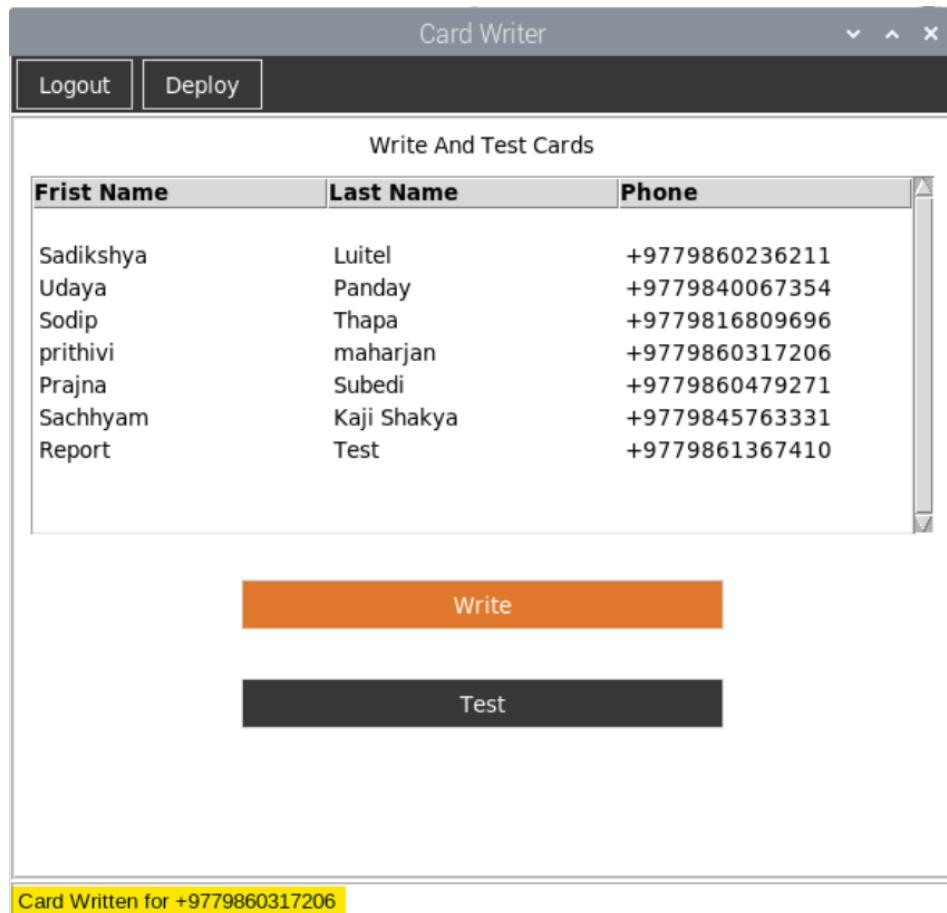
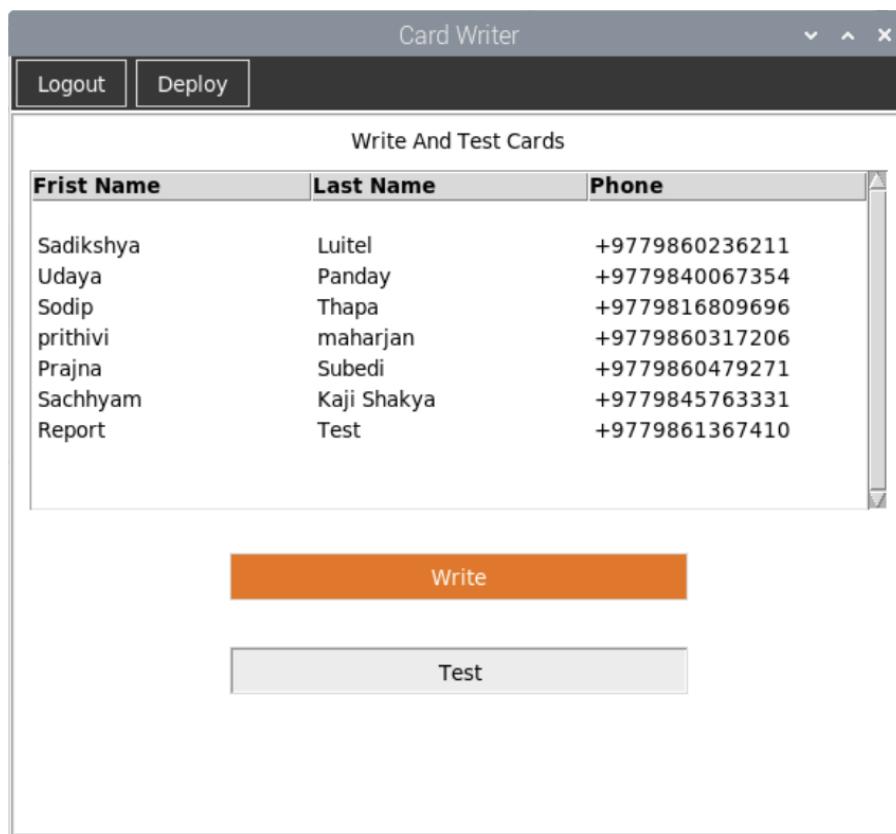


Figure 204 Card Written for Selected User

#### 4.2.8.2 Confirm Card Write

*Table 45 Card Test Description*

Action Performed	For a user whose card has been written, the written card is placed on the testing screen.
Expected Output	The status of the user's card should be changed to written.
Actual Output	The output was as expected.
Conclusion	Successful



*Figure 205 Running Test Card Operation*

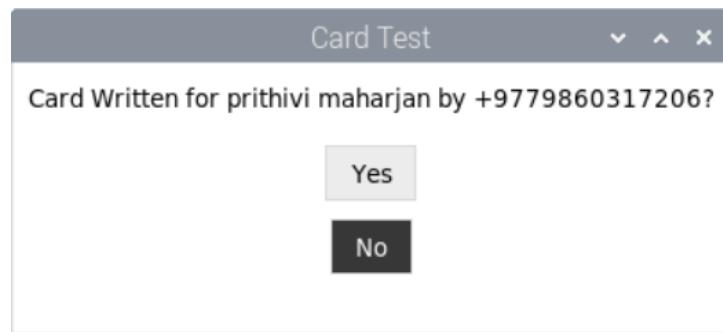


Figure 206 Card Write Confirmation Dialog Box

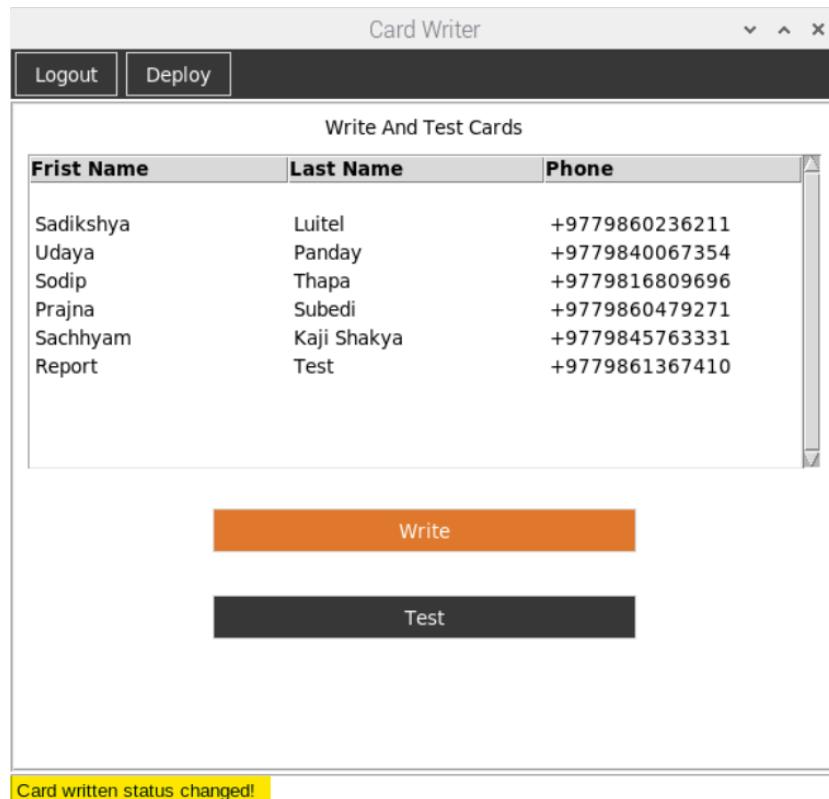


Figure 207 Card Written Status Change Confirmation

User:

---

Card number:

Card written

Card write date:  Today |

Note: You are 5.75 hours ahead of server time.

---

Figure 208 User Card Status After Confirmation

## 4.2.9 Deploy Card

### 4.2.9.1 Correct Deploy

Table 46 Deploy Card Test Description

Action Performed	Card is tapped on the deploy screen to simulate distribution of card to the user.
Expected Output	The status of user's card delivery should be changed to delivered.
Actual Output	The output was as expected.
Conclusion	Successful

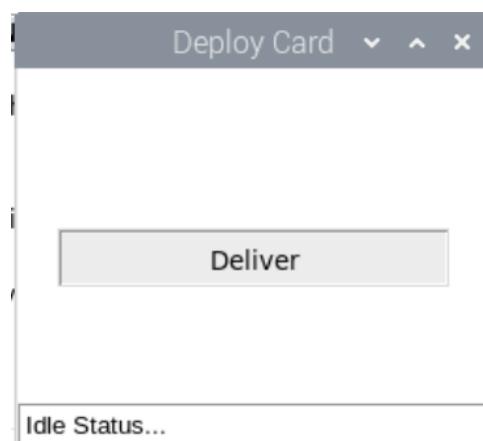


Figure 209 Card Deploy Action

 A screenshot of a mobile application interface. At the top, there is a user selection field showing "prithivi.maharjan@islingtoncollege.edu.np" with a dropdown arrow, a pencil icon, and a green plus icon. Below this is a card number field containing "Z977986031720602n21n2021". Underneath are two sections: one for "Card written" with a checked checkbox and one for "Card write date" showing "2021-04-16" with a "Today" button and a calendar icon. A note below the date says "Note: You are 5.75 hours ahead of server time." At the bottom is a section for "Card delivered" with an unchecked checkbox.

Figure 210 Current Delivery Status

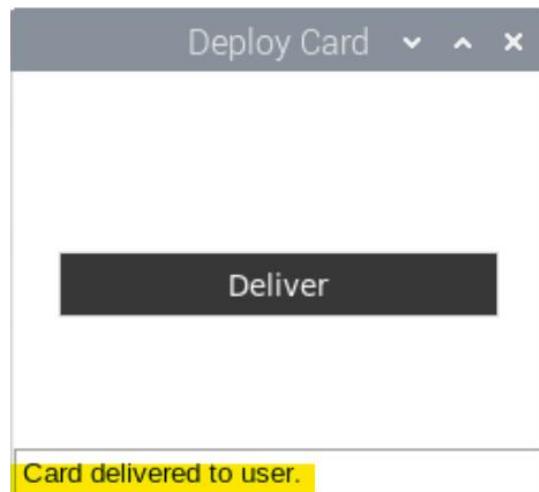


Figure 211 Card Deliver Confirmation

User:  ✎ +

---

Card number:

---

Card written

---

Card write date:  Today |   
Note: You are 5.75 hours ahead of server time.

---

Card delivered

Figure 212 Updated Delivery Status

#### 4.2.9.2 Un-deploy Card

Table 47 Un-Deploy Card Test Description

Action Performed	To simulate wrong distribution, deployed card is tapped again on the deploy screen.
Expected Output	The status of user's card delivery should be changed to not delivered.
Actual Output	The output was as expected.
Conclusion	Successful



Figure 213 Card Un-deploy Action

User:

prithivi.maharjan@islingtoncollege.edu.np

---

Card number:

Z977986031720602n21n2021

---

Card written

---

Card write date:

2021-04-16

Today

Note: You are 5.75 hours ahead of server time.

---

Card delivered

Figure 214 Current Card Delivery Status



Figure 215 Deploying Card for 'card-delivered' User

User: prithivi.maharjan@islingtoncollege.edu.np ▾

---

Card number: Z977986031720602n21n2021

---

Card written

---

Card write date: 2021-04-16 Today |

Note: You are 5.75 hours ahead of server time.

---

Card delivered

Figure 216 Updated Card Delivery Status

#### 4.2.10 Read Card

##### 4.2.10.1 Tap With 0 Balance

Action Performed	Card is tapped on entry when top-up balance is 0.
Expected Output	Message should be displayed to the user for invalid entry.
Actual Output	The output was as expected.
Conclusion	Successful

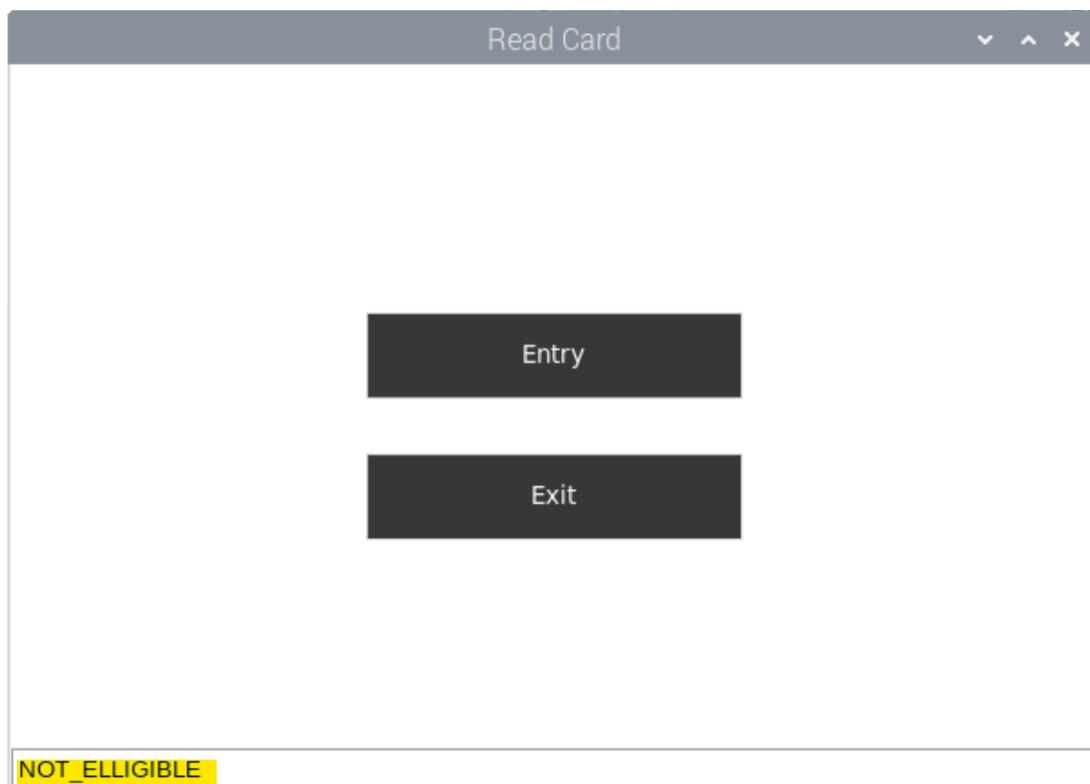


Figure 217 Error Message for Invalid Entry Tap

#### 4.2.10.2 Tap Twice on Entry

Table 48 Tap Twice on Entry Test Description

Action Performed	Card is tapped twice on entry screen.
Expected Output	Message for invalid number of taps should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

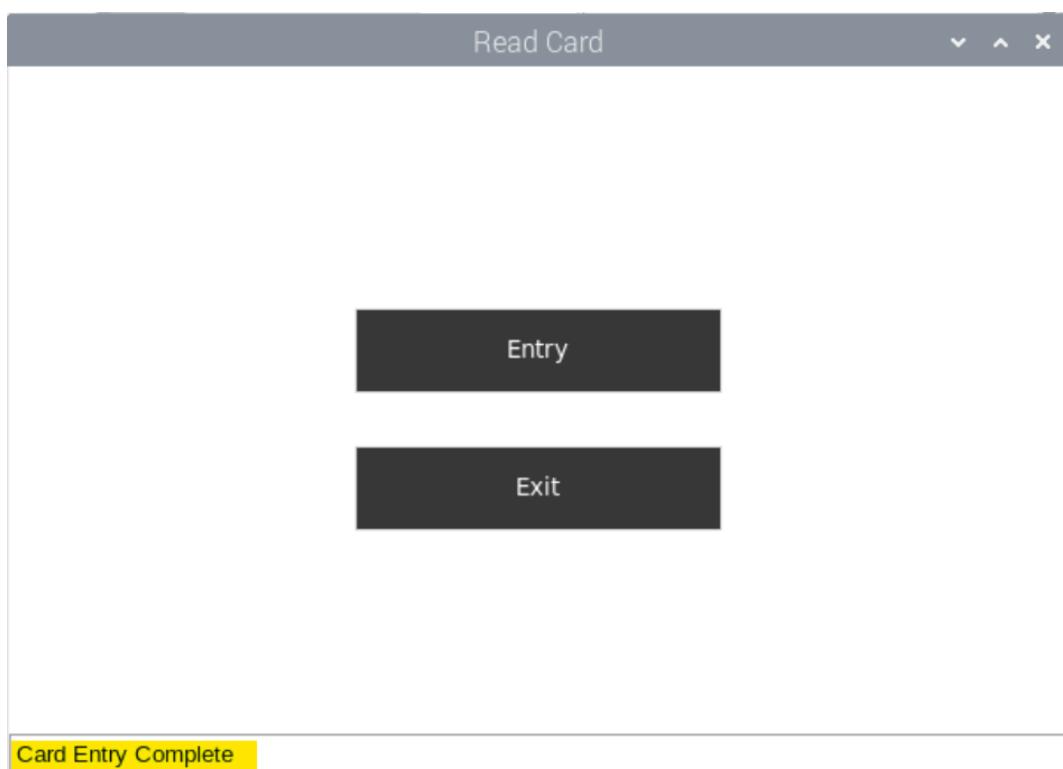


Figure 218 First Valid Entry

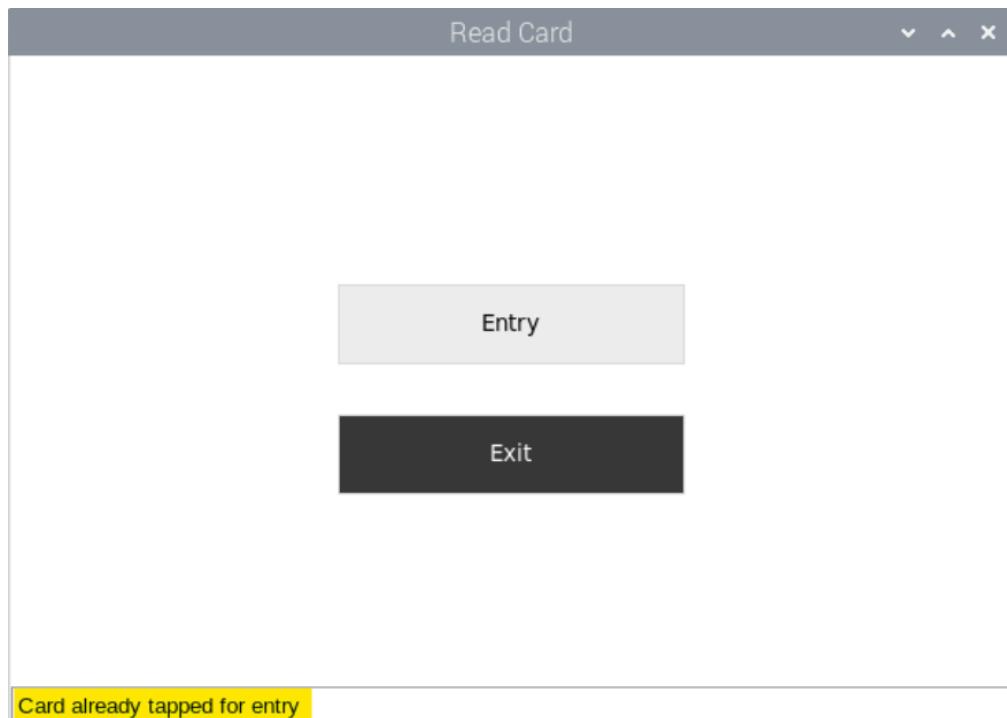


Figure 219 Second Tap on Entry

#### 4.2.10.3 Tap Exit Before Entry

Table 49 Tap on Exit before Entry Description

Action Performed	Card is tapped on exit before tapping on entry.
Expected Output	The system should prompt the user to tap on entry before tapping on exit.
Actual Output	The output was as expected.
Conclusion	Successful

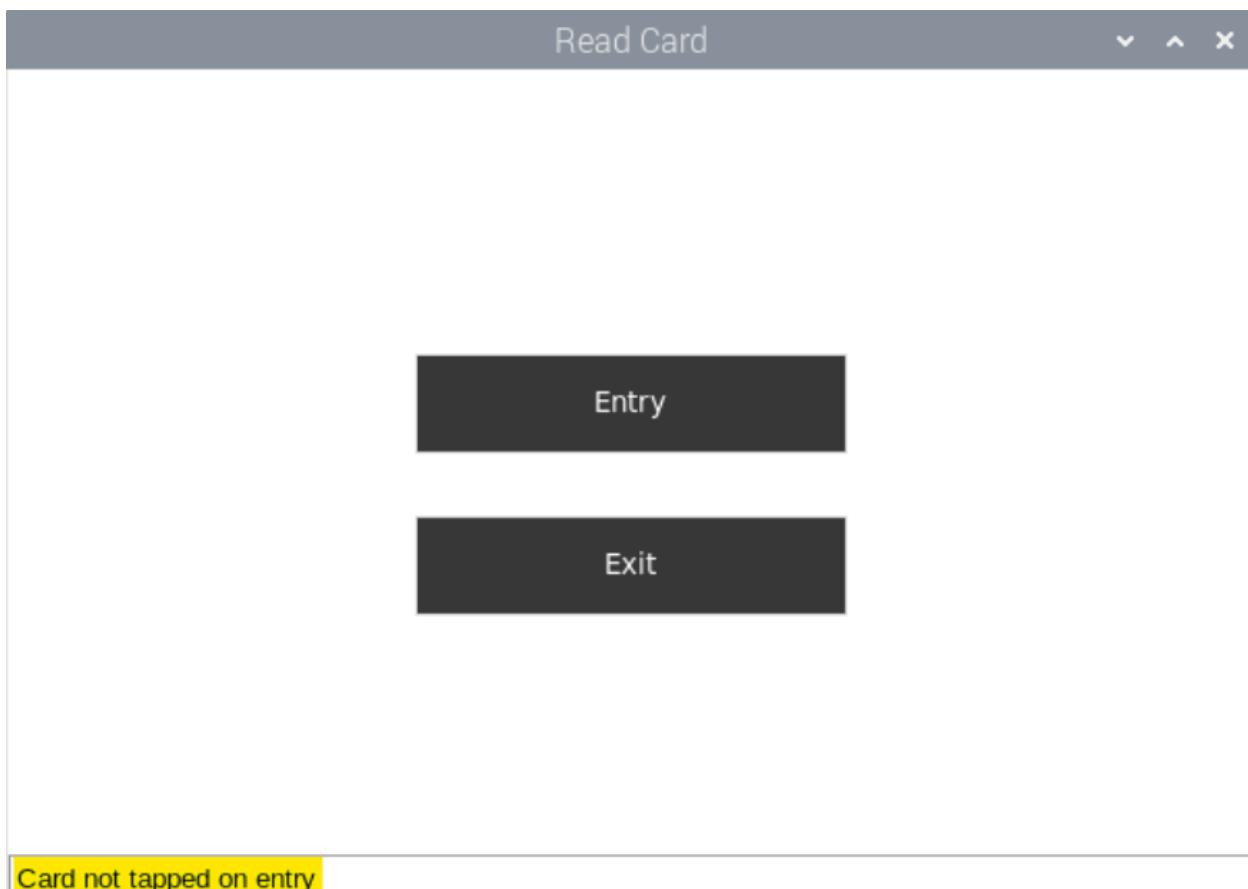


Figure 220 Error Message for Exit Tap Before Entry Tap

#### 4.2.10.4 Tap on Entry and Exit

Table 50 Valid Taps Test Description

Action Performed	Card is tapped on entry first and then on exit to simulate valid trip.
Expected Output	The travel data should be saved on user's travel history and the amount from the card needs to be deducted.
Actual Output	The dialog box showed 'ERROR' on exit tap.
Conclusion	Test Failed

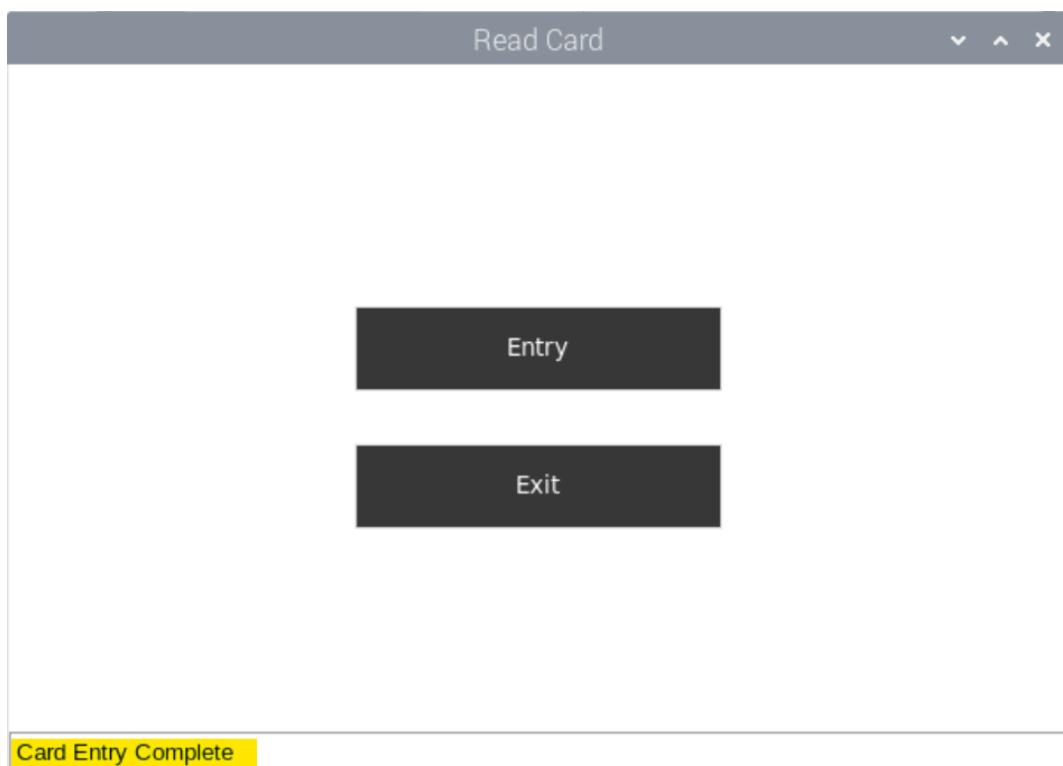


Figure 221 Successful Card Entry for Valid Trip

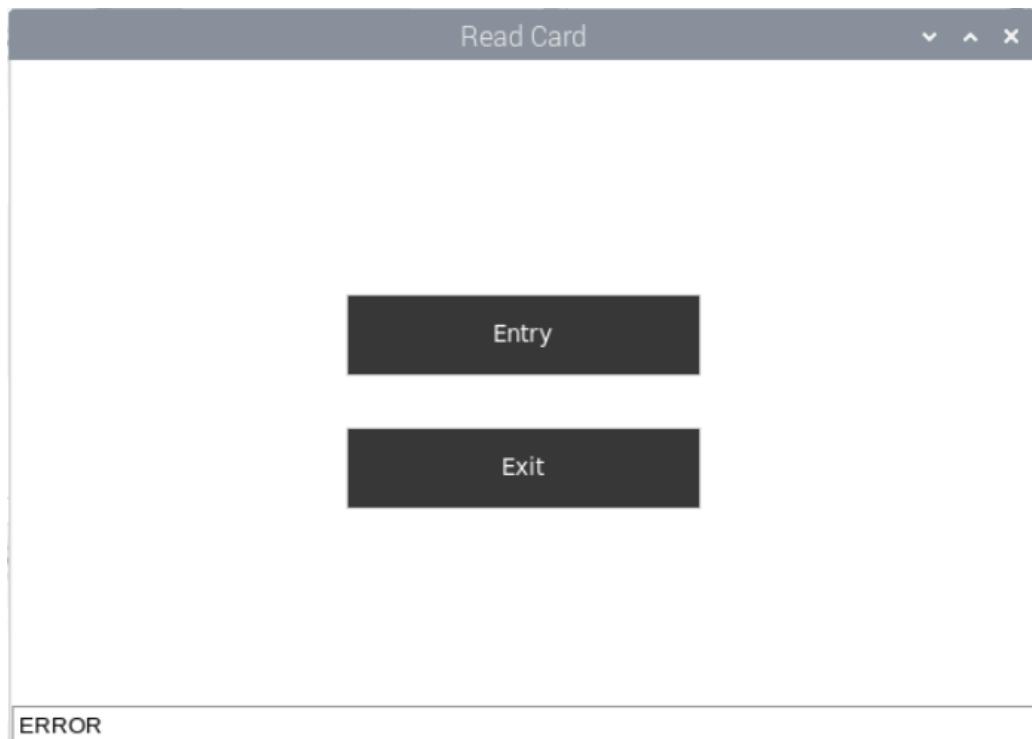


Figure 222 Unsuccessful Exit Tap

**Retest**

Action Performed	Card is tapped on entry first and then on exit to simulate valid trip.
Expected Output	The travel data should be saved on user's travel history and the amount from the card needs to be deducted.
Actual Output	The output was as expected.
Conclusion	Successful

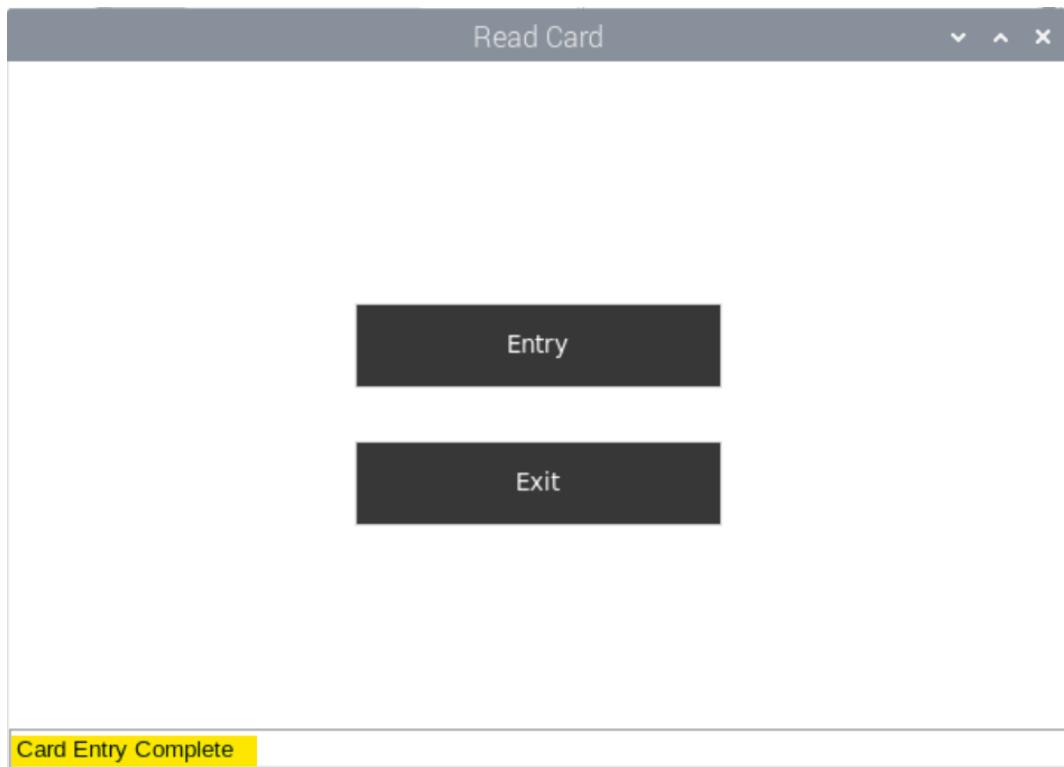


Figure 223 Successful Card Entry for Valid Trip (2)

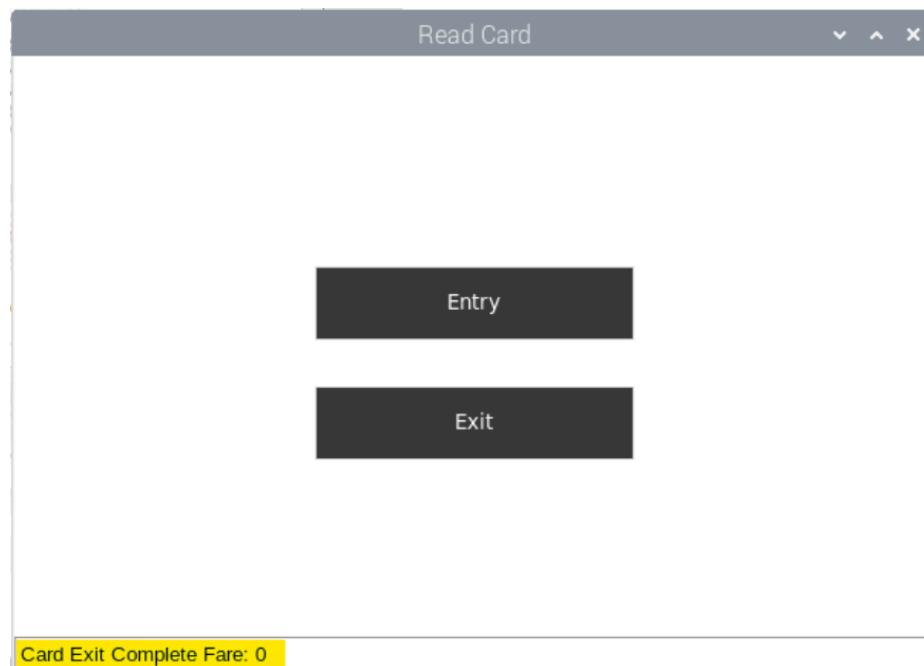


Figure 224 Card Exit Successful for Valid Entry

Trip id:	7b4a1858-9eb1-11eb-95cf-dca63211b649
Card number:	Z977986031720602n21n2021
Bus:	Bus object (BA-4-KHA-8848)
Entry date:	2021-04-16 Note: You are 5.75 hours ahead of server time.
Entry time:	18:29:27 Note: You are 5.75 hours ahead of server time.
Entry latitude:	27.702146666666668
Entry longitude:	85.33066833333334
Entry name:	Chandan Marg, Kathmandu 44600, Nepal

Figure 225 Trip Entry Data

Entry: UserEntry object (7b4a1858-9eb1-11eb-95cf-dca63211b649)

---

Exit date: 2021-04-16 Today |

Note: You are 5.75 hours ahead of server time.

---

Exit time: 18:29:32 Now |

Note: You are 5.75 hours ahead of server time.

---

Exit latitude: 27.702151

---

Exit longitude: 85.330677

---

Exit name: Chandan Marg, Kathmandu 44600, Nepal

---

Trip distance: 0

---

Card tap

---

Fare: 0

Figure 226 Trip Exit Data

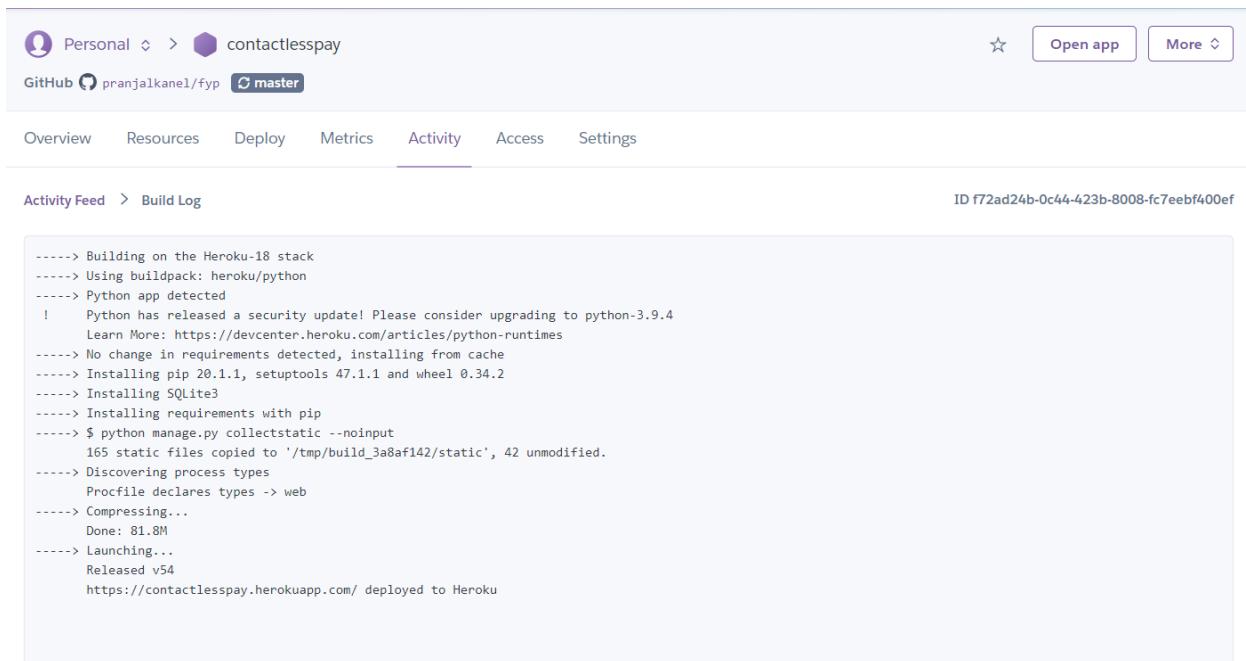
## 4.3 System Testing

### 4.3.1 Application Build Tests

#### 4.3.1.1 Web Build

*Table 51 Web Build Test Description*

Action Performed	Locally developed web application is built on Heroku web server.
Expected Output	The application build should be successful and launch on URL.
Actual Output	The output was as expected.
Conclusion	Successful



The screenshot shows a GitHub repository page for a project named 'contactlesspay'. The repository has 1 star and is set to 'Open app'. The navigation bar includes links for Overview, Resources, Deploy, Metrics, Activity (which is underlined), Access, and Settings. Below the navigation bar, it says 'Activity Feed > Build Log' and provides a build ID: f72ad24b-0c44-423b-8008-fc7eebf400ef. The build log itself displays the command-line output of the deployment process:

```

-----> Building on the Heroku-18 stack
-----> Using buildpack: heroku/python
-----> Python app detected
!   Python has released a security update! Please consider upgrading to python-3.9.4
  Learn More: https://devcenter.heroku.com/articles/python-runtimes
-----> No change in requirements detected, installing from cache
-----> Installing pip 20.1.1, setuptools 47.1.1 and wheel 0.34.2
-----> Installing SQLite3
-----> Installing requirements with pip
-----> $ python manage.py collectstatic --noinput
    165 static files copied to '/tmp/build_3a8af142/static', 42 unmodified.
-----> Discovering process types
  Procfile declares types -> web
-----> Compressing...
  Done: 81.8M
-----> Launching...
  Released v54
  https://contactlesspay.herokuapp.com/ deployed to Heroku

```

*Figure 227 Latest Build Log on Heroku*

[Application available here.](#)

### 4.3.1.2 Phone Build

Table 52 Phone Build Test Description

Action Performed	The mobile application is built to return an APK.
Expected Output	Installable APK for android devices should be generated.
Actual Output	The output was as expected.
Conclusion	Successful

```
Calling mockable JAR artifact transform to create file: C:\Users\PREDATOR\.gradle\caches\transforms-2\files-2.1\899ad92750e1801bdd89e15d3b996076\android.jar with input E:\Android\platforms\android-29\android.jar
Calling mockable JAR artifact transform to create file: C:\Users\PREDATOR\.gradle\caches\transforms-2\files-2.1\95bb85f161de960dcba14b3f1e7bf7b\android.jar with input E:\Android\platforms\android-28\android.jar
Removed unused resources: Binary resource data reduced from 54KB to 46KB: Removed 1dK
Running Gradle task 'assembleRelease'...
Running Gradle task 'assembleRelease'... done
Built build/appoutputs/flutter-apk/app-release.apk (18.1MB).
```

Figure 228 Flutter Build Successful

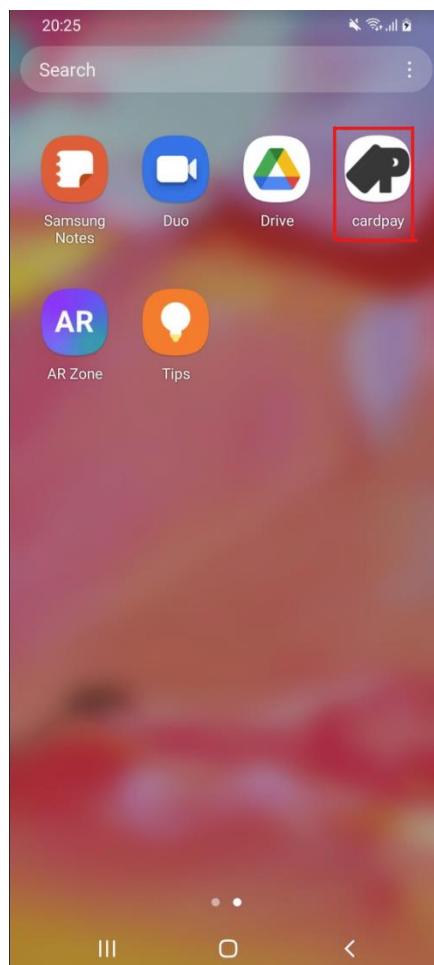


Figure 229 Application Installed on Android Device

#### 4.3.1.3 Desktop Build

Table 53 Desktop Build Test Description

Action Performed	Desktop application is run on debug mode.
Expected Output	The application should run on device containing the required python files.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows a desktop environment with two windows. The top window is titled 'Read Card' and contains a single button labeled 'Entry'. The bottom window is a terminal window titled 'Shell' with the following text:

```
dy in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.
GPIO.setup(pin_RST, GPIO.OUT)
>>>
```

Figure 230 Read Card Application

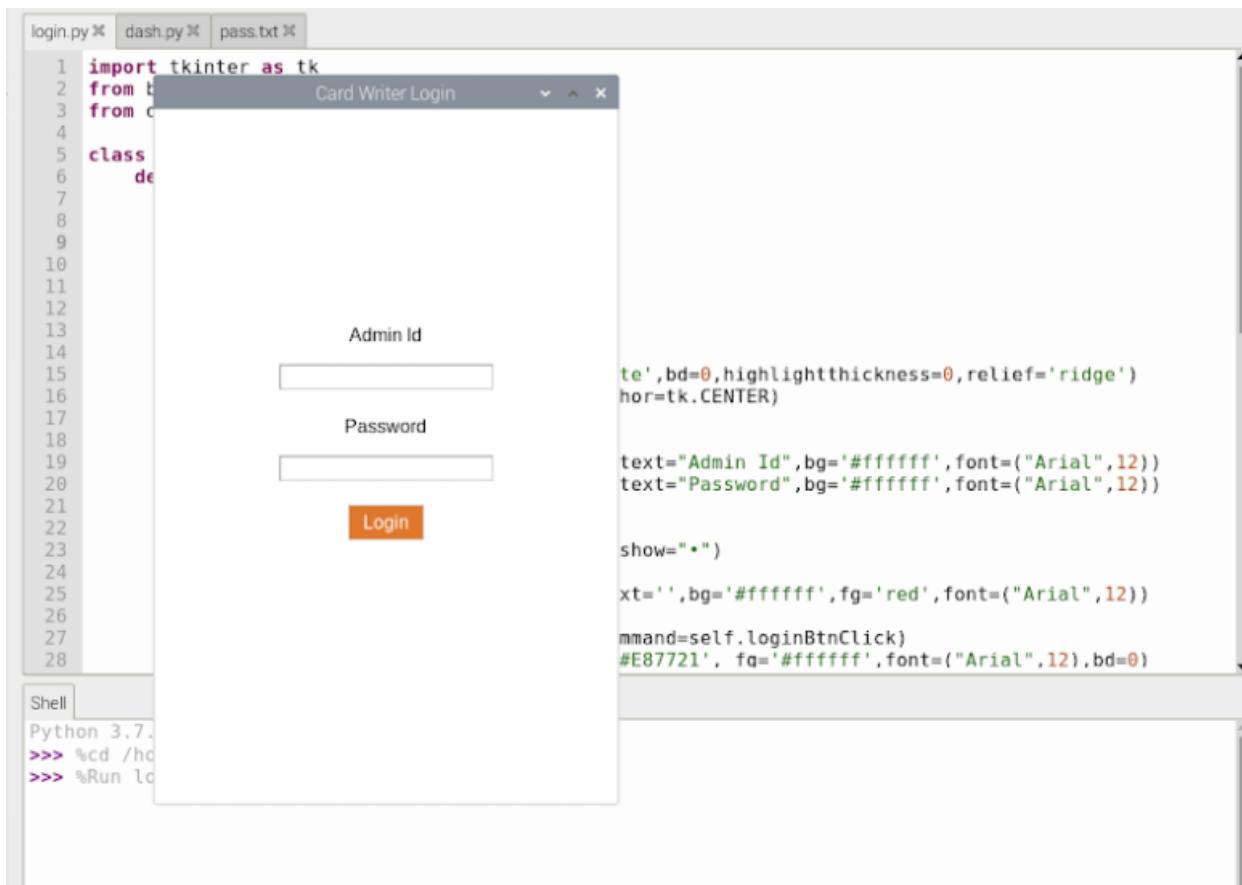


Figure 231 Write Card Application

### 4.3.2 API Data Tests

#### 4.3.2.1 Card-Not-Written User Data

Table 54 User Data API Test Description

Action Performed	Using postman, request is sent to the web server to fetch data of users who don't have their card written.
Expected Output	User data of people with card written status set to false should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows a Postman API test configuration for a GET request to `http://contactlesspay.herokuapp.com/hardware/get-cards`. The Headers tab is selected, displaying the following key-value pairs:

Header	Value
Accept	application/json
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	Token 27f755bdfe6eca3fba6aa1a94b5bbe2af2abb4e1
Key	Value

The Body tab is selected, showing the JSON response received from the API. The response is a list of user cards, each containing a phone number, first name, and last name. The JSON is formatted as follows:

```

17  [
18    {
19      "phone": "+9779860479271",
20      "first_name": "Prajna",
21      "last_name": "Subedi"
22    },
23    {
24      "phone": "+9779845763331",
25      "first_name": "Sachhyam",
26      "last_name": "Kaji Shakya"
27    },
28    {
29      "phone": "+9779861367410",
30      "first_name": "Report",
31      "last_name": "Test"
32    }
]
  
```

Figure 232 Card-Not-Written User Details for Admin

### 4.3.2.2 Travel Log Data

Table 55 Travel Log Data API Test Description

Action Performed	Travel log for users is tested using postman.
Expected Output	Travel history of a given user should be displayed.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** https://contactlesspay.herokuapp.com/apps/get-travel-log ...
- Headers (10):**
  - Authorization (checked): token ecaf50c4c1aea3398059f85c99134ee740471f53
  - Other entries: token 5U3T5/4U889U/D3AC849DUE9AEB3D3//Z91U/UZ, token 27f755bdfe6eca3fba6aa1a94b5bbe2af2abb4e1, token 4561fe67acdc1f3a55fee9f57e48b3ade62f8d0a
- Body:** JSON response (Pretty) showing travel log data:

```

1  {
2      "profile_pic": "https://storage.googleapis.com/fyp-contactlesspay/default.png",
3      "user_name": "Bibek Maharjan",
4      "user_status": "Account Verified",
5      "card_amount": 1648,
6      "log": [
7          {
8              "entry_name": "Dillibazar Pipalko Bot Bus Stop, Dilli Bazaar Rd",
9              "entry_date": "2021-03-12",
10             "entry_time": "11:45:21",
11             "exit_name": "Bag Bazar Sadak, Kathmandu",
12             "exit_date": "2021-03-12",
13             "exit_time": "12:03:31",
14             "trip_distance": 1084,
15             "fare": 7
16         },

```

Figure 233 Successful API Response for Travel Log

### 4.3.3 API Permissions Tests

#### 4.3.3.1 Travel Log API for Admin

Table 56 Travel Log API Permission Test Description

Action Performed	API for retrieving travel history is entered using admin credential.
Expected Output	An error message should be displayed for unauthorized use.
Actual Output	The output was as expected.
Conclusion	Successful

The screenshot shows a Postman collection interface. At the top, a GET request is defined with the URL <https://contactlesspay.herokuapp.com/apps/get-travel-log>. Below the request, the 'Headers' tab is selected, showing a table with four rows. The first row has an unchecked checkbox under 'Authorization' and a value 'token 5U3T5/4U88YU/D3ac84yD'. The second row has a checked checkbox under 'Authorization' and a value 'token 27f755bdfe6eca3fba6aa1e'. The third and fourth rows have unchecked checkboxes under 'Authorization' and values 'token ecaf50c4c1aea3398059f8' and 'token 4561fe67acdc1f3a55fee9f' respectively. The 'Body' tab is also visible at the bottom.

Key	Value
Authorization	token 5U3T5/4U88YU/D3ac84yD
Authorization	token 27f755bdfe6eca3fba6aa1e
Authorization	token ecaf50c4c1aea3398059f8
Authorization	token 4561fe67acdc1f3a55fee9f

At the bottom, the response body is displayed in JSON format:

```

1  {
2      "detail": "You do not have permission to perform this action."
3  }

```

Figure 234 Denied Permission for Admin Travel Log

#### 4.3.3.2 Card Write Data API for User

Table 57 Card Write API Permission Test Description

Action Performed	API for retrieving user data with card written false is entered using user credential.
Expected Output	An error message should be displayed for unauthorized use.
Actual Output	The output was as expected.
Conclusion	Successful

GET http://contactlesspay.herokuapp.com/hardware/get-cards

Params	Authorization	Headers (7)	Body	Pre-request Script	Tests	Settings
Accept		Accept		/*		
		Accept-Encoding	gzip, deflate, br			
		Connection	keep-alive			
		Authorization	Token 4561fe67acdc1f3a55fee9f57e48b3ade62f8d0a			
		Key	Value			Description

Body Cookies Headers (11) Test Results Status: 403 Forbidden

Pretty Raw Preview Visualize JSON

```

1 {
2   "detail": "You do not have permission to perform this action."
3 }
```

Figure 235 Permission Denied for User on Admin API

## 4.4 Critical Analysis

### 4.4.1 Working Under Development Methodology

This being the first project that I have worked under a development methodology, helped me understand a lot of development techniques. Development of software does not constitute primarily of coding; it is a form of implementation that is brought upon with a lot of planning and designing the system beforehand. Upcoming points summarize my experience of working under the RUP methodology,

- Time management is a vital skill to complete a project.
- Planning and designing features of a project is very important aspect of software engineering since coding without representations of data flow prolongs the development time. This is easily dealt by planning a feature.
- Iterations during development enriches the overall system and adds stability to the existing feature.
- User feedbacks before project construction and after project completion is necessary to gain knowledge about requirements gathering and changing existing requirements.

#### 4.4.2 Analyzing Failed Test Cases

In this section, analysis related to the test cases completed in the previous section is documented. This contains addressing failed test cases and test cases that passed but could have a better working in the system.

##### 4.4.2.1 User Dashboard Test

For [this test case](#), the error was identified as a mismatch of spelling of a variable in the server and the client template. Due to this mismatch in spelling, the correct query of getting possible walking trips was not shown in the front end.

```
context = {'userProfile':userProfile,
           'userCard':userCard,
           'valid_trips':valid_trips,
           'total_distance':total_distance,
           'total_trips':total_trips,
           'longest_ride':longest_ride,
           'shortest_ride':shortest_ride,
           'cycling_trips':cycling_trips,
           'walking_rides':walking_trips,
           'carbon_emission':carbon_emission}
```

Figure 236 Variable Name in Server

```
{% for trip in walking_trips %}
  {% if forloop.counter == 1 %}
    <div class="carousel-item active text-center p-4">
      click right or left arrow to view your trips!
    </div>
  {% endif %}
```

Figure 237 Variable Name in Template

This error was solved just by making both variable's name same, which in turn rendered the correct stat board for the user.

```
context = {'userProfile':userProfile,  
          'userCard':userCard,  
          'valid_trips':valid_trips,  
          'total_distance':total_distance,  
          'total_trips':total_trips,  
          'longest_ride':longest_ride,  
          'shortest_ride':shortest_ride,  
          'cycling_trips':cycling_trips,  
          'walking_trips':walking_trips,  
          'carbon_emission':carbon_emission}
```

Figure 238 Fixing Variable Spelling



Figure 239 Fixed Stat Board for Walking Trips

#### 4.4.2.2 Travel Entry/Exit Test

For [this test case](#), the error was identified as the result of GIGO (garbage in, garbage out). Occasionally, the GPS module fails to identify the correct geo location of the bus and returns both the latitude and longitude as 0. This causes no routes to be found from the distance matrix API on exit tap, which causes the dashboard of the read-card application to show error.

Trip id:	cf9003ce-9eb0-11eb-86ca-dca63211b649 
Card number:	Z977986031720602n21n2021   
Bus:	Bus object (BA-4-KHA-8848)   
Entry date:	2021-04-16 Today  Note: You are 5.75 hours ahead of server time.
Entry time:	18:24:39 Now  Note: You are 5.75 hours ahead of server time.
Entry latitude:	0.0
Entry longitude:	0.0
Entry name:	

Figure 240 Wrong Coordinates on Entry

Although this test case has not been fixed due to time constraints for the project submission, the solution for this error is to set checks for these types of garbage data which would not allow a faulty user entry to occur. The error only appeared during the testing phase of the application which caused the fix for the bug to be applied in future work.

#### 4.4.3 Test Summary

The overall core features proposed for the payment system have been completely developed. Not more than one bug in the system could be identified as unfixable for the given time period for development, the existing bug can be fixed in future where the time for development is extended.

The summary of all the tests have been given below,

##### 4.4.3.1 Strong Aspects of the System

- The signup process works as intended.
- The login process in all the applications handle exceptions on par to industry standards.
- Permission management for all the applications work as intended.
- The UI on all the devices are responsive and carry the same theme which makes productivity for all types of users to be better.
- With occasional hiccups, the card payment feature works great and the data storage along with data management works better than originally expected.

##### 4.4.3.2 Weak Aspects of the System

- Due to lack of individual storage buckets for individual users, their credentials can be seen by other users if the link for the image is accessible. This vulnerability has been placed on purpose to save costs of the purpose and for demonstration purposes.
- Loading screen and proper notifications on the desktop application for the admin seems to be below par, although the system works as expected.
- The hardware of for payment seems to be fragile for deployment and very expensive for individual buses. It fits well for demonstration purposes but could be better for deployment.

The overall system was a success as shown by all the test cases, with room for improvement, further development for this project should be a deployable, fix-all solution for contactless payment system.

## Chapter 5. Conclusion

Working under RUP methodology, the development of the proposed payment system has been completed. The developed system provides all the core functionalities as proposed that aim to solve the problems in the current scenario. The developed system aims to promote travel using public transportation by bringing ease of use to the end users who can sign up from all around the world. The system also aims to provide valid data of the customers for the bus companies which can be used to stay in contact with the commuters.

The development process of the system included a lot of planning, implementing the plans and testing the developed product. With the approach of incremental development, the development of system is documented accordingly.

## 5.1 Legal, Social and Ethical Issues

The upcoming section discusses about the legal, social, and ethical issues that could arise during the use of the system and how the system is equipped to handle these issues.

### 5.1.1 Legal Issues

#### 5.1.1.1 User Privacy

The application only takes limited amount of data from the user and the collected data is stored safely in AWS servers. Credentials uploaded by the users are stored safely in google cloud buckets which needs a permission configuration [addressed in the future work](#). With these checks and balances in place, user information cannot be accessed by any other parties other than the users themselves.

#### 5.1.1.2 Card Data Breach

The headers of the cards which have been deployed can be stored in the reader's local database so that external card scamming mechanisms would not work. Although this feature has not been implemented, it has been [addressed in the future work for this project](#). This would prevent the card reader from being spoofed a fake card containing a valid card number from a user.

#### 5.1.1.3 Limited Tracking

Users are not tracked by the application by any means. The sole way of tracking a user during trips by using the GPS module equipped in the means of transportation and not in the card. The application does not take data from the user anywhere else other than paying for their fare on the bus.

## 5.1.2 Social Issues

### 5.1.2.1 Travel Syndicate

Nepal's transportation industry is heavily crippled by transportation syndicate (Ghimire, 2018). For the developed system to be deployed in the Nepalese market, replacing, or removing the existing market is next to impossible. Constant reluctance towards change in prosperous development has been an issue in the Nepalese transportation industry. To deal with this issue, the business model needs to fit in certain way that the developed system should fit existing bus systems as well as newer bus companies.

Excluding companies with old buses is not an option since travel industry is not open to changes, so the product is designed in with the thinking of implementation in all types of buses and organizations which excludes bus stops and proprietary bus readers by opting for universal reading technology.

### 5.1.2.2 Replacing Bus Staff

As explained in the previous section for travel syndicate, same goes for bus staff (conductors) as well. Since the introduction of the project will bring the role of the bus staff to a near end, the only solution to peacefully displace the existing manpower is by shifting their roles in the bus. In current scenarios, bus staff collect money and return back the change to the commuters, with the implementation of this payment system, the bus staff can be taught how to fix and repair the technology equipped by the bus.

Shifting roles for the existing manpower is the only viable solution to have a better deployment experience for the overall system.

### 5.1.3 Ethical Issues

#### 5.1.3.1 Card Scams

People having wrong intentions can use the card not as the system intends it to be used. People might could tap the card on entry to get into the bus and not tap while exiting the bus. The same situation goes when people might do this on number of buses to get 'free rides'. But the system detects these anomalies and shows it to the administration side as 'invalid trips'. To handle these types of issues that an individual might possess, all the checks for this issue has been dealt in the [read-card tests](#).

#### 5.1.3.2 System Exploit (Distance Calculation)

There are individuals with a lot of thinking capacity that can de-code how the distance calculation for the system works. For a bus moving in clockwise direction in its route, it gets to point B from point A as shown below.

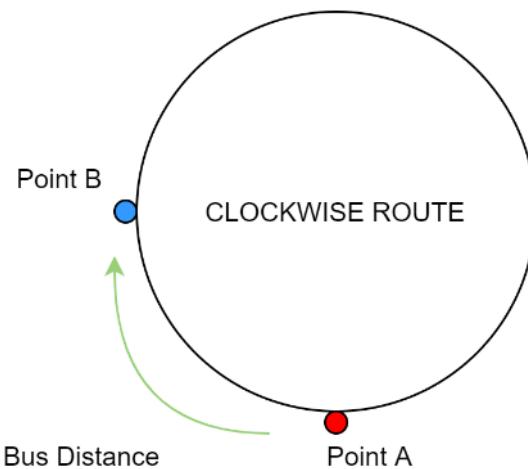
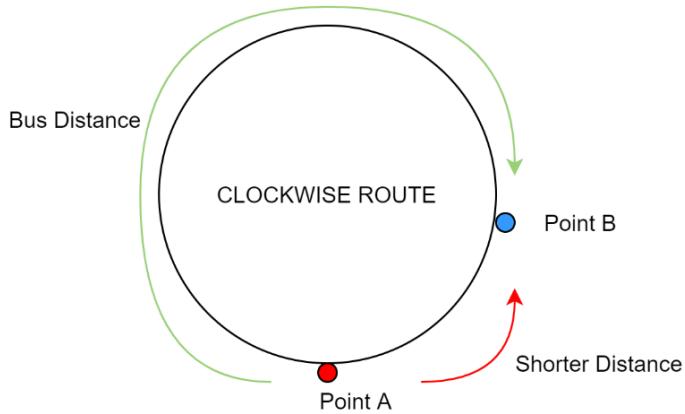


Figure 241 Bus Route Simulation

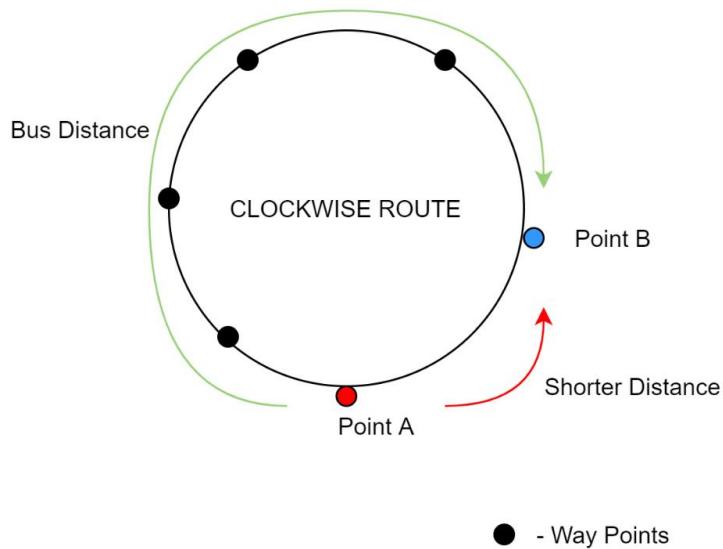
Since the distance is calculated using the google maps directions API, the default distance given by the API is the shortest distance for two points. If an individual were to recognize this flaw in the system design, they could easily travel more than halfway through the route for a lesser fare as shown below.



*Figure 242 Bus Shorter Distance Exploit*

If the cost is calculated using the shorter distance in every trip, individuals with degrading mindset could easily cause harm to the overall business model by encouraging passengers to travel longer distances in a route.

To address this issue, way-points are introduced in cost calculation. Way-points are coordinates between given origin (Point A) and destination point (Point B). By storing the path through which the bus travels, the distance for origin and destination is calculated through the path where the bus has gone through. This eliminates the possibility of cost calculation using shorter distance.



*Figure 243 Way Points in Bus Route*

## 5.2 Advantages

The advantages brought by this project are the direct result of solving the outlined problems in the existing payment systems. The advantages in the implemented system of this project have been listed below.

### 5.2.1 Faster Payment

In ideal conditions, the developed payment system outperforms the traditional cash based payments. This is the direct result of not needing external manpower to handle all the cash and return back cash to the commuter. Removing this labor intensive step also removes the tiring nature of the job, which reduces the chances of error during processing cash. Using card for payment on entry and exit sets the habit of entry and exit through terminals of the buses which causes the capacity of the bus to circulate better.

### 5.2.2 In Built Discount

This feature is aimed at students who commute regularly. With the removal of showing school-provided ID every time on their trip, the overall experience for regular commuters elevates significantly. This removes the need of handling cash and card together and merges them both to unify the payment method whilst applying the discount.

### 5.2.3 Expenditure Tracking

One of the biggest drawbacks of using cash is that there is no tracking of expenditure. With the developed system in place, flow of money in and out of the payment system can be tracked easily. With the reports of travel history along with money required for the trip displayed in the dashboard for the user, it gives the user a better idea of their travelling expenses for public transportation.

### 5.2.4 Customer Relation Establishment

In existing scenario, bus companies have no records of who their customers are, the only existing relation between a public transportation company and their consumers is monetary. With the establishment of this system, companies will have a way of knowing their consumers along with a way of contacting them if necessary.

### 5.2.5 Eco Friendliness Promotion

The user stat board shows the total distance travelled by a commuter using busses. This data also provides the contribution of the commuter in carbon emissions, which is harmful to the environment. With the target of increased travel using public transportation, responsibility for using the technologies derived from nature should be assessed as well. Which is why the commuters get to see their impact in global carbon emissions and their actions to mitigate it.

### 5.2.6 Eradication of Slower Systems

Implementing payment system in transportation is only a start for a better future. With the implementation of this system, it will start a movement for digitalizing aspects of our functioning that are still slow. These systems could include, stock management, user management and customer relationship. Different aspects of this projects can be implemented in other sectors of business in varying degree. This includes contactless payments in stores, libraries, and places where a lot of people need quick transaction whereas the tracking of user could be built in customer relationship management software which can be implemented in most of the type of businesses.

### 5.3 Limitations

Although the problems stated in the project proposal have been addressed, due to multiple reasons, some limitations in the system exist. These existing limitations in the system and reasoning for their existence have been listed and explained in upcoming section.

#### 5.3.1 Phone Number Change

In the current system, the card number for the user is partly generated by using the verified phone number. Changing the phone number would create inconsistencies in identifying a consumer in the overall system. If an old phone number for a user is used to register into the system again, it could bring unwary problems in the system which cannot be assessed in the given time frame for this project.

In addition to this problem, limiting the change of phone number saves drastic amount of money for SMS client. Changing a phone number would need the user to remove existing phone number and verify new phone number again. This includes receiving new SMS again which cannot be sustained in the current budget.

#### 5.3.2 Cash Payment Tracking

During payment of fare in the exit of a trip, if the card has insufficient amount, there is currently no way of determining if the user has paid the remaining amount or not. This would require a system of receipt or physical identifiers for payment.

Since this part of the project has not been included in the overall use case and there is no implemented logic of physical payment identifiers, this limitation exists in the system which cannot be fixed in the given development context.

#### 5.3.3 Locating Exit for Invalid Trips

When a user taps on entry but does not on exit, the entry location is easily seen by the admin and can contact the user. While this system is in place, there is no actual method of verifying the location where the person has got off the bus. Also with the cash payment limitation, knowing if the user has paid in cash or exited in the right location cannot be verified.

#### 5.3.4 Transaction Time (Server Location)

The current server location is in USA for free hosting. Although the application runs as intended and in time, the physical location of the data center and the application server makes the overall system a bit slower than expected. This limitation can be multiplied by slow internet connections for the card readers which bring us to the next limitation of this system.

#### 5.3.5 Mobile Internet for Buses

In Nepal, the only existing bus proprietary internet is Nepal Telecom internet for buses which is based on Wi-Fi with data limits (Nepali Telecom, 2017). These data limits and internet based for commuters is rather limiting for the payment system. Due to lack of proper internet connection in buses, the implementation of the system is not quite ready and is limited only in testing environment.

## 5.4 Future Work

Research for future work have been provided in the [appendix section](#).

### 5.4.1 Addressing Limitations

#### 5.4.1.1 Phone Number Change Feature

With additional forms and increased budget of SMS client, the phone number change feature can be easily implemented. With this in mind, the card number generation method should also be taken into consideration for change if phone numbers were allowed to be changed in the first place.

#### 5.4.1.2 Track Cash Payment

This limitation can be addressed by adding receipts for the user which is generated on cash payment. This receipt should include an identifier that the user should be allowed to enter scan through the mobile app or enter in the web application which removes verifies the user's payments using card.

#### 5.4.1.3 Implementing AI

To locate the user's exit in a trip, the only viable way in current scenario is to check surveillance footage present in the bus. This scenario is extremely unlikely to be properly implemented in the current scenario, which brings to this solution.

Since the capacity of users in a bus is known, and data for all the users is flowing through the system, an estimated location for the invalid trip can be generated. This location can then be verified with the user through the admin which can then update the record for that trip.

#### 5.4.1.4 Stable Internet Connection for Bus

Since the current internet in bus is commuter focused Wi-Fi, there needs to be an establishment of proper internet connection on buses for mobile transport monitoring. This feature could eliminate the need of external GPS module since the Nepal Telecom MODEM could provide the geolocation of the bus during stable connection period.

#### 5.4.2 Asynchronous Programming in Desktop Application

During testing and critical analysis of the project, conclusions were drawn indicating lack of proper messaging in the desktop application. During the development of the application, used programming language's support for async programming was unknown. In the future, these existing features can be added to the program which will have progress indicators and appropriate messages that notify the admin about the progress of current process.

#### 5.4.3 Cancel Card Operations

In the current 'card read and write' operation, there is no way of cancelling an operation like writing a card if the 'write button' has already been clicked. Without being able to cancel the action, this brings problem in reading the card where actuating the 'entry' function of the card would make it impossible to change to 'exit' function. Although this functionality falls under asynchronous programming as well, its implementation adds to the solution for another feature.

#### 5.4.4 Tap to Pay Using Apple or Google Pay

Apple Pay and Google Pay include the feature of adding bank cards in their wallet where tapping through their application completed the monetary transaction. Adding the current card and binding it to these services would greatly increase the quality of payment methods as users would be able to tap and pay using their phones and eliminating the need of card itself.

#### 5.4.5 Using Better Card Validation

Adding EMV to the NFC communication could make the monetary transactions more secure. Since EMV and NFC are companion technologies, increasing the security of cards using a better technology would mean a better and reliable source of payment in public transportation.

#### 5.4.6 Individual Buckets for Users

Users with links to the credentials of other users can view the credentials of other users. This can be mitigated by adding individual buckets for individual users. This system is not in place right now to save cost for buckets storage provided by google. Adding individual buckets for the users would make the application more secure and restrict the data to the internet.

#### 5.4.7 Web Features in Mobile Application with Notifications

The mobile application currently lacks heavily in features. The ability to signup, pay for card and pay for fare directly through the mobile application would make the system complete and ready for use for the general public. With the shift of usage from desktop to mobile, this work is necessary to capitalize the market.

With proper notifications to the user in mobile application using Firebase Cloud Messaging, this should be easy to implement in the near future.

#### 5.4.8 Disable Card if Stolen

The feature of disabling card payments if the card is stolen is a must. This feature can be applied in both the admin and user dashboard where the user can manually disable their card by providing their ID and password for verification or by contacting the bus-administration. The required checks for restricting disabled card and the disabling card from backend has already been implemented. The remaining future work is to allow users to disable their cards by themselves.

#### 5.4.9 Discount Plans

In the current system, the admin only checks a box to specify if a card is discounted or not. Every discounted card is given 15% discount of the calculated fare. In future, administration should be allowed to add greater discount plans for greater types of users that the administration recognizes. Adding discount plans to the application would allow groups of users to have variable discount rates. For example, bus staffs travelling on another bus could have a 100% discount whereas discount plans for festivals or seasons could be introduced.

## Chapter 6. References

- Abel, M. (2014) *Stack Exchange* [Online]. Available from:  
<https://electronics.stackexchange.com/questions/78013/can-i-use-a-nfc-ring-as-an-oyster-card#:~:text=Oyster%20cards%2C%20using%20the%20MiFare,essentially%20standard%20encrypted%20nfc%20tags>. [Accessed 2 December 2020].
- Agile Alliance. (2020) *Scrum* [Online]. Available from:  
[https://www.agilealliance.org/glossary/scrum/#q=~\(infinite~false~filters~\(postType~\(~'page~'post~'aa\\_book~'aa\\_event\\_session~'aa\\_experience\\_report~'aa\\_glossary~'aa\\_research\\_paper~'aa\\_video\)~tags~\(~'scrum\)\)~searchTerm~'~sort~false~sortDirection~'asc~page~1\)](https://www.agilealliance.org/glossary/scrum/#q=~(infinite~false~filters~(postType~(~'page~'post~'aa_book~'aa_event_session~'aa_experience_report~'aa_glossary~'aa_research_paper~'aa_video)~tags~(~'scrum))~searchTerm~'~sort~false~sortDirection~'asc~page~1)) [Accessed 7 September 2020].
- Chhetri, S. (2017) *Kathmandu Post* [Online]. (2) Available from:  
<https://kathmandupost.com/valley/2017/03/17/sajha-yatayat-introduces-smart-card-payment-system> [Accessed 11 December 2020].
- Durkin, A. (2018) *Trade Vistas* [Online]. Available from: <https://tradevistas.org/toward-a-global-cashless-economy/> [Accessed 31 August 2020].
- EMV Connection. (2015) *EMV and NFC: Complementary Technologies Enabling Secure Contactless Payments* [Online]. Available from: <https://www.emvconnection.com/emv-and-nfc-complementary-technologies-enabling-secure-contactless-payments/> [Accessed 20 March 2021].
- Evening Standard. (2014) *Evening Standard* [Online]. Available from:  
<https://www.standard.co.uk/news/transport/oyster-card-users-overcharged-more-than-60-million-9041393.html> [Accessed 30 November 2020].
- EZ-Link. (2020) *EZ-Link* [Online]. Available from: <https://www.ezlink.com.sg/index.php> [Accessed 4 December 2020].
- GadgetsNowBureau. (2019) *Gadgets Now* [Online]. Available from:  
<https://www.gadgetsnow.com/slideshows/one-nation-one-card-launched-how-to-get-it-what-you-can-do-and-more/the-rupay-contactless-card-with-national-common-mobility-card-ncmc-support-will-be-available-with-over-25-banks-including-sbi-pnb-and-others/photolis> [Accessed 3 December 2020].

- Ghimire, R. (2018) *Driving syndicate off: More needs to be done* [Online]. Available from: <https://thehimalayantimes.com/opinion/driving-syndicate-off-more-needs-to-be-done> [Accessed 10 April 2021].
- Google. (2021) *Set up a JavaScript Firebase Cloud Messaging client app* [Online]. Available from: <https://firebase.google.com/docs/cloud-messaging/js/client> [Accessed 10 April 2021].
- Gundaniya, N. (2019) *Digipay* [Online]. Available from: <https://www.digipay.guru/blog/towards-a-cashless-society-major-benefits/#:~:text=Transaction%20speed&text=Sweetgreen%20and%20Salad%20chains%20also,than%20that%20of%20card%20transactions>. [Accessed 15 August 2020].
- Gus. (2017) *PiMyLifeUp* [Online]. Available from: <https://pimylifeup.com/raspberry-pi-rfid-rc522/> [Accessed 30 October 2020].
- Hackster. (2019) *Hackster* [Online]. Available from: <https://www.hackster.io/bhushanmapari/interfacing-u-blox-neo-6m-gps-module-with-raspberry-pi-3d15a5> [Accessed 22 October 2020].
- Hallen, J. (2018) *Combining Tkinter and Asynchronous I/O with Threads* [Online]. Available from: <https://www.oreilly.com/library/view/python-cookbook/0596001673/ch09s07.html> [Accessed 23 March 2021].
- Khabarhub. (2020) *Khabarhub* [Online]. Available from: <https://english.khabarhub.com/2020/03/116909/> [Accessed 4 August 2020].
- Laurance, K. (2020) *Recently Heard* [Online]. Available from: <https://recentlyheard.com/2020/11/18/when-is-the-scrum-methodology-not-ideal/> [Accessed 22 November 2020].
- Lewis, S. (2019) *Tech Target* [Online]. Available from: <https://searchsoftwarequality.techtarget.com/definition/waterfall-model> [Accessed 25 November 2020].
- Lohani, M. (2018) *Explainer: What is the ‘syndicate’ controversy all about?* [Online]. Available from: <https://english.onlinekhabar.com/explainer-what-is-the-syndicate-controversy-all-about.html> [Accessed 10 April 2021].

- Lucidchart. (2018) *Lucid Chart* [Online]. Available from: <https://www.lucidchart.com/blog/rapid-application-development-methodology> [Accessed 26 November 2020].
- Nepali Telecom. (2017) *Ntc bus WiFi Internet service during bus travel; Free WiFi and more* [Online]. Available from: <https://www.nepalitelecom.com/2017/06/ntc-bus-wifi-internet-service-internet-bus-travel.html> [Accessed 2 April 2021].
- Neupane, A. (2018) *New Business Age* [Online]. Available from: [https://www.newbusinessage.com/MagazineArticles/view/2339#:~:text=\(NCHL\)%2C%20excluding%20cheques%2C,rural%20population%20are%20prime%20factors.](https://www.newbusinessage.com/MagazineArticles/view/2339#:~:text=(NCHL)%2C%20excluding%20cheques%2C,rural%20population%20are%20prime%20factors.) [Accessed 31 August 2020].
- O'Carroll, B. (2020) *Codebots* [Online]. Available from: <https://codebots.com/app-development/what-is-rapid-application-development-rad> [Accessed 23 November 2020].
- Pettinger, T. (2020) *Economics Help* [Online]. Available from: <https://www.economicshelp.org/blog/164246/economics/pros-and-cons-of-a-cashless-society/#:~:text=There%20are%20several%20advantages%20of,an%20with%20bad%20credit%20histories.> [Accessed 31 August 2020].
- RationalSoftwareCorporation. (2002) *Rational Unified Process: Overview* [Online]. Rational Software Coorporation Available at: <https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/> [Accessed 22 November 2020].
- Red Apple PR Agency. (2015) *Red Apple PR Agency* [Online]. Available from: <http://www.redapplepragency.com/nuspay-launches-first-tap-pay-nfc-smart-payment-card-in-nepal.html> [Accessed 10 December 2020].
- Republica. (2017) *NT launches Internet service for bus passengers* [Online]. Available from: <https://myrepublica.nagariknetwork.com/news/nt-launches-internet-service-for-bus-passengers/> [Accessed 22 February 2021].
- Rojko, M. (2021) *fcm-django* [Online]. Available from: <https://github.com/xtrinch/fcm-django> [Accessed 18 March 2021].
- Singh, A. (2019) *Capterra* [Online]. Available from: <https://blog.capterra.com/what-is-rapid-application->

[development/#:~:text=Rapid%20Application%20Development%20\(RAD\)%20is,strict%20planning%20and%20requirements%20recording.](#) [Accessed 24 November 2020].

Square Up. (2016) *EMV vs. NFC Payments* [Online]. Available from: <https://squareup.com/us/en/townsquare/emv-vs-nfc-whats-the-difference> [Accessed 18 March 2021].

TechTerms. (2005) *TechTerms* [Online]. Available from: <https://techterms.com/definition/rup> [Accessed 24 April 2020].

TheEconomicTimes. (2012) *The Economic Times* [Online]. Available from: <https://economictimes.indiatimes.com/news/news-by-industry/banking/finance/banking/npcis-rupay-debit-cards-to-rival-visa-and-mastercard/articleshow/12424537.cms> [Accessed 01 December 2020].

TheHindu. (2019) *The Hindu* [Online]. Available from: <https://www.thehindu.com/news/national/pm-launches-one-nation-one-card/article26433523.ece#> [Accessed 3 December 2020].

TransLink. (2020) *TransLink* [Online]. Available from: <https://translink.com.au/about-translink/who-we-are> [Accessed 6 December 2020].

VikasPedia. (2019) *Vikaspedia* [Online]. Available from: <https://vikaspedia.in/e-governance/online-citizen-services/government-to-citizen-services-q2c/transport-related-services/national-common-mobility-card> [Accessed 2 December 2020].

Vitali, M. (2019) *Barcelona Checkin* [Online]. Available from: [https://www.barcelonacheckin.com/en/r/barcelona\\_tourism\\_guide/articles/bicing-rental-scheme](https://www.barcelonacheckin.com/en/r/barcelona_tourism_guide/articles/bicing-rental-scheme) [Accessed 7 December 2020].

Xiao, Y. (2020) *WeForum* [Online]. Available from: <https://www.weforum.org/agenda/2020/05/digital-payments-cash-and-covid-19-pandemics/> [Accessed 20 August 2020].

## Chapter 7. Appendix

### 7.1 Appendix A: Pre-Survey

#### 7.1.1 Pre-Survey Form

The image shows a survey form with four sections:

- Name \***: A text input field labeled "Short-answer text".
- Age \***: A text input field labeled "Short-answer text".
- Occupation \***: A text input field labeled "Short-answer text".
- A question: "Before the pandemic, how often did you use public transportation in a week? \*". Below it are three radio button options:
  - Never
  - Moderately
  - Almost daily

Figure 244 Pre Survey Form (1)

How do you rate the experience of paying with cash in public transportation? \*

1      2      3      4      5

Bad                                    Good

On a scale from 1 to 5, how often do you use contactless payment methods or e banking? \*

1      2      3      4      5

Never                                    For most of my transactions.

On a scale from 1 to 5, how much do you think implementation of contactless payment system on public transportation would increase the quality of travel? \*

1      2      3      4      5

Not helpful                                    Very helpful

Do you think providing discount details for students in their card would help students during travel? (Not needing to show college cards to conductors) \*

Yes  
 No  
 Maybe

Figure 245 Pre Survey Form (2)

Would you be comfortable to sign up to this service using an application where you can view all \*  
of your travel details, expenditure and other statistics?

Yes

No

Would it encourage you to signup to the service if you didn't have to come in contact with \*  
vehicle staff to pay, reducing the risk of transmission during the pandemic?

Yes

No

Maybe

Would you rather load your card before travelling or pay your card after travelling? \*

Load card before travelling

Pay card after travelling

Use existing bank card

How likely are you to sign up for this contactless payment service? (This could expand to other \*  
use cases as well)

1

2

3

4

5

Figure 246 Pre Survey Form (3)

### 7.1.2 Sample of Filled Pre-Survey Forms

Name *					
Basant Tamang					
Age *					
23					
Occupation *					
Student					
Before the pandemic, how often did you use public transportation in a week? *					
<input type="radio"/> Never					
<input type="radio"/> Moderately					
<input checked="" type="radio"/> Almost daily					
How do you rate the experience of paying with cash in public transportation? *					
1                  2                  3                  4                  5					
Bad <input type="radio"/> <input type="radio"/> <input checked="" type="radio"/> <input type="radio"/> Good					

Figure 247 Pre Survey Form Sample (1)

On a scale from 1 to 5, how often do you use contactless payment methods or e banking? \*

1      2      3      4      5

Never                                    For most of my transactions.

On a scale from 1 to 5, how much do you think implementation of contactless payment system on public transportation would increase the quality of travel? \*

1      2      3      4      5

Not helpful                                    Very helpful

Do you think providing discount details for students in their card would help students during travel? (Not needing to show college cards to conductors) \*

Yes  
 No  
 Maybe

Would you be comfortable to sign up to this service using an application where you can view all of your travel details, expenditure and other statistics? \*

Yes  
 No

Figure 248 Pre Survey Form Sample (2)

Would it encourage you to signup to the service if you didn't have to come in contact with vehicle staff to pay, reducing the risk of transmission during the pandemic? \*

Yes  
 No  
 Maybe

Would you rather load your card before travelling or pay your card after travelling? \*

Load card before travelling  
 Pay card after travelling  
 Use existing bank card

How likely are you to sign up for this contactless payment service? (This could expand to other use cases as well) \*

1      2      3      4      5

Submitted 25/11/2020, 16:36

Figure 249 Pre Survey Form Sample (3)

### 7.1.3 Pre-Survey Result

This report is based on preliminary survey taken among regular commuter to find out the likeliness of the service to take off among commuters and see their ratings for the 'solutions' mentioned in the introduction portion. The following statistics are from 72 total responses. If any outcomes can be determined from the project, it will be listed below the figure of response.

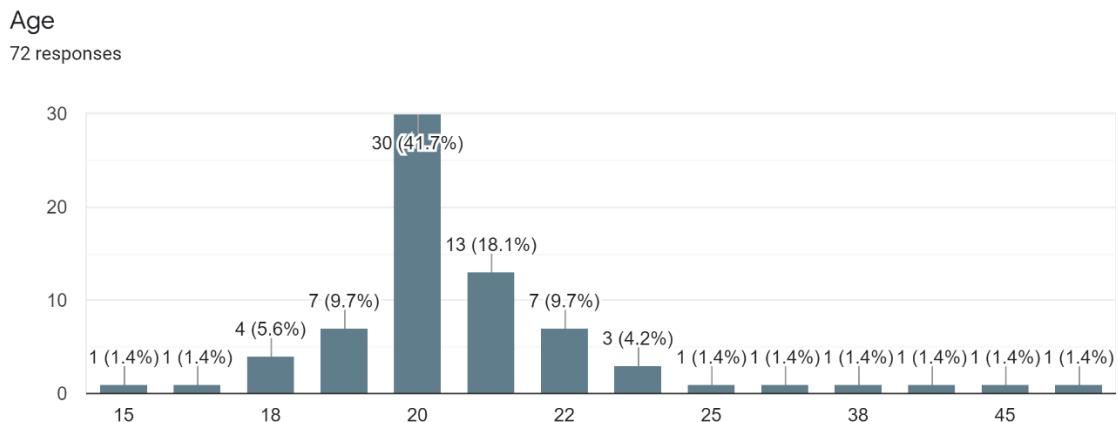


Figure 250 First Question Response

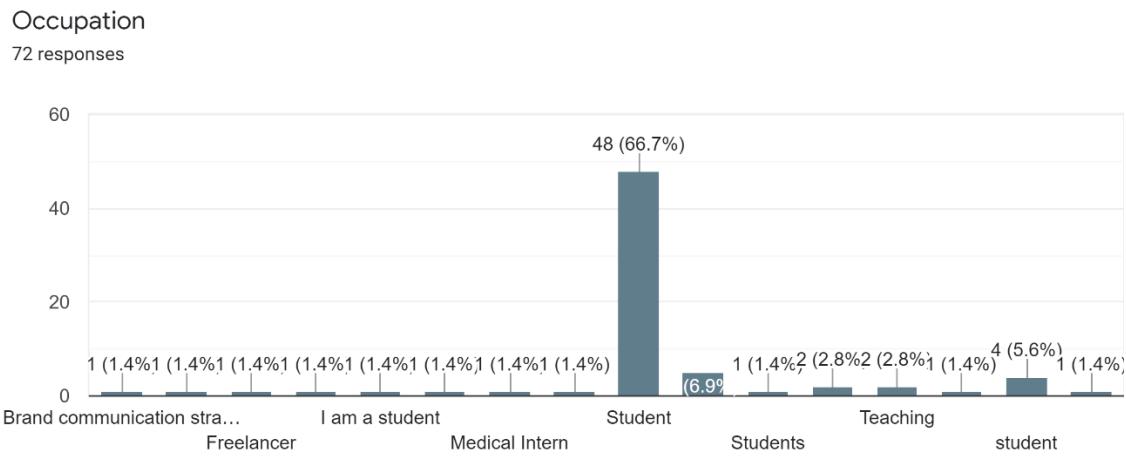
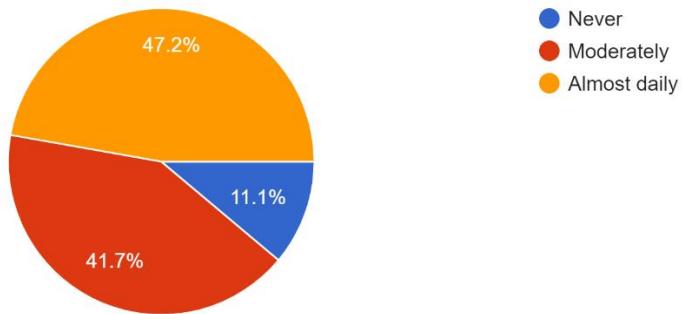


Figure 251 Second Question Response

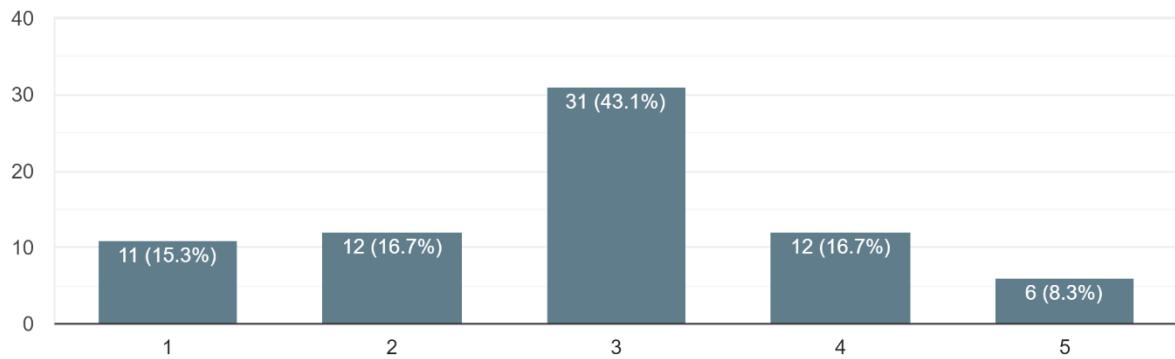
Before the pandemic, how often did you use public transportation in a week?  
72 responses



*Figure 252 Third Question Response*

This response shows that the majority of the participants are commuters which helps to legitimize the outcome from the upcoming questions.

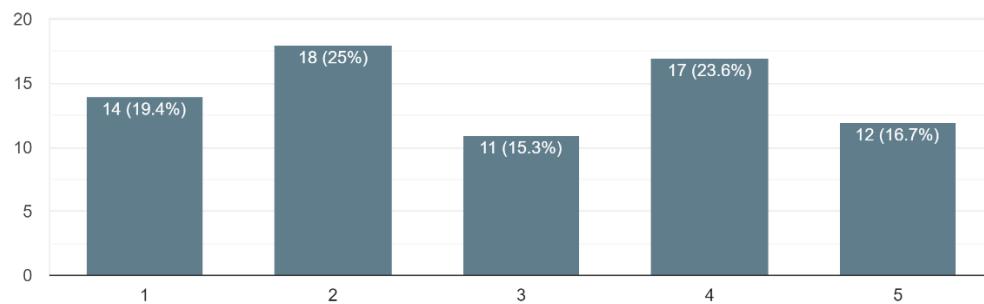
How do you rate the experience of paying with cash in public transportation?  
72 responses



*Figure 253 Fourth Question Response*

These ratings show the user experience with current cash based payments in transportation which leans towards the negative side of the spectrum with most of the participants having average customer experience.

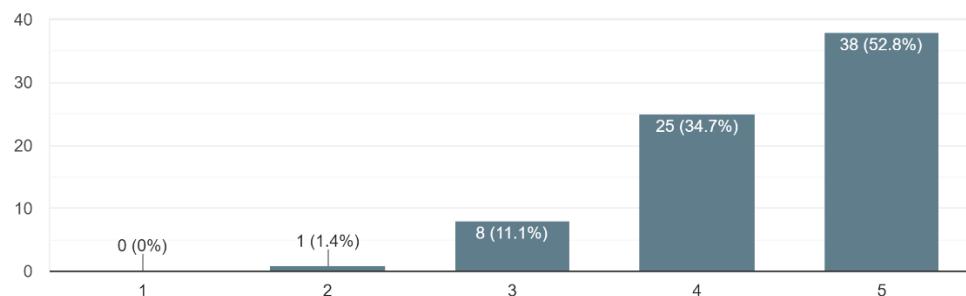
On a scale from 1 to 5, how often do you use contactless payment methods or e banking?  
72 responses



*Figure 254 Fifth Question Response*

This graph shows a very diverse response from the participants with slight skew towards non e-banking users which means that most of the participants could have new experience to pay for the card if they sign up for the service. The built-in graph does not show the correlation between regular commuters and e-banking users, so the future assumptions are purely based on survey experience

On a scale from 1 to 5, how much do you think implementation of contactless payment system on public transportation would increase the quality of travel?  
72 responses

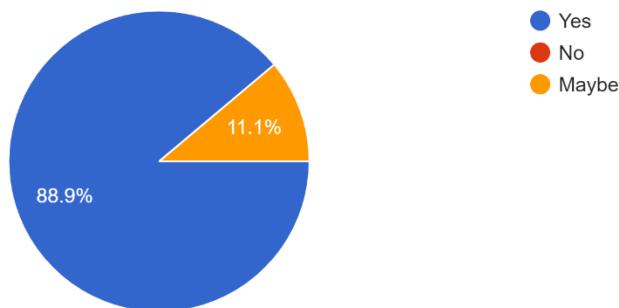


*Figure 255 Sixth Question Response*

This graph shows that the almost all of the participants believe that contactless payment system could increase the quality of travel. This response is positive for the continuity of development for the project.

Do you think providing discount details for students in their card would help students during travel? (Not needing to show college cards to conductors)

72 responses

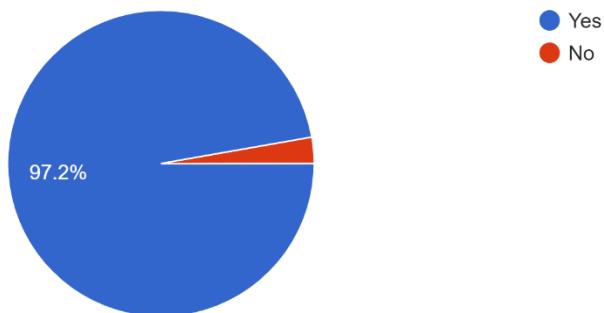


*Figure 256 Seventh Question Response*

With none of the users opposing the idea of integrated discount system on the card, this response solidifies the features for integrated discount cards which must be present in the system.

Would you be comfortable to sign up to this service using an application where you can view all of your travel details, expenditure and other statistics?

72 responses

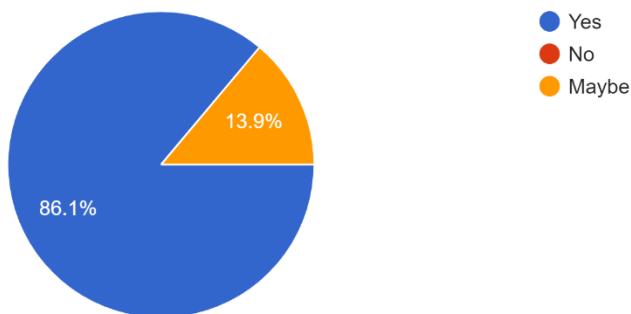


*Figure 257 Eighth Question Response*

With 97.2% of the responses showing positive feedback for viewing their expenditure and travel statistics, this legitimizes the development of web application and mobile application as an actual solution for the project.

Would it encourage you to signup to the service if you didn't have to come in contact with vehicle staff to pay, reducing the risk of transmission during the pandemic?

72 responses

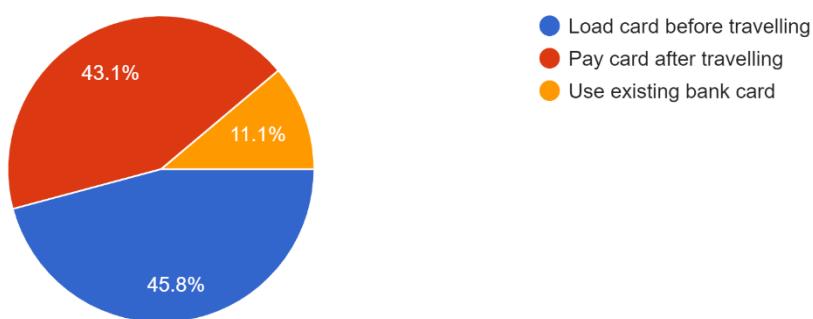


*Figure 258 Ninth Question Response*

This figure shows that most of the people would not want to come in contact with a bus staff to tap their cards. This answer could change the use case of the project as a bus staff will not be involved in payment and the users will need to tap their cards when entering and leaving the transportation for payment.

Would you rather load your card before travelling or pay your card after travelling?

72 responses

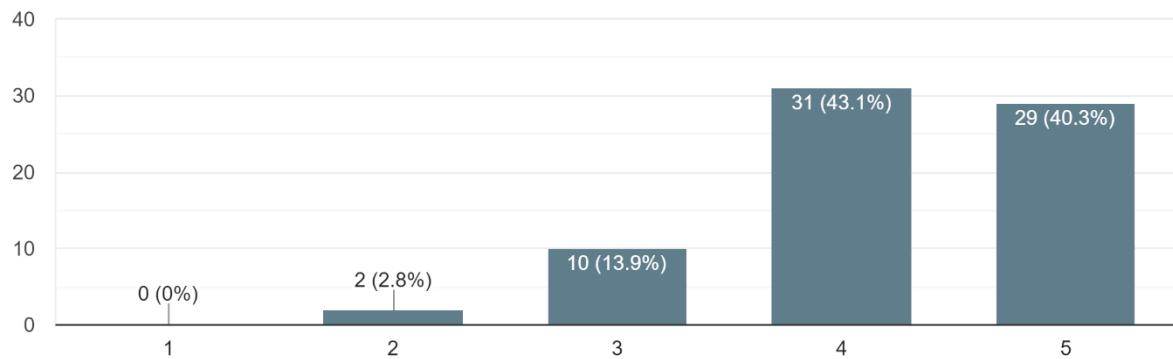


*Figure 259 Tenth Question Response*

With even distribution with the payment methods, top up for the cards would be the optimal solution for this project as implemented in other proven-to-work international contactless payment projects.

How likely are you to sign up for this contactless payment service? (This could expand to other use cases as well)

72 responses



*Figure 260 Eleventh Question Response*

With most of the response being positive towards the end goal of the project, this helps the goal of the project to be clear and worth working on.

## 7.2 Appendix C: Sample Codes

### 7.2.1 Sample Code of the UI

#### 7.2.1.1 Desktop Tree View

```

self.account_tree = ttk.Treeview(self.treeframe, yscrollcommand=self.treescroll.set)

# configuring scroll bar
self.treescroll.config(command=self.account_tree.yview)

self.account_tree['columns'] = ("First Name", "Last Name", "Phone")
self.account_tree.column("#0", width=0, stretch=NO)
self.account_tree.column("First Name", anchor=W, width=120)
self.account_tree.column("Last Name", anchor=W, width=120)
self.account_tree.column("Phone", anchor=W, width=120)

# headings for columns
self.account_tree.heading("#0", text="", anchor=W)
self.account_tree.heading("First Name", text="First Name", anchor=W)
self.account_tree.heading("Last Name", text="Last Name", anchor=W)
self.account_tree.heading("Phone", text="Phone", anchor=W)
# adding data
self.loadTable()

# buttons
self.writeBtn = tk.Button(self.buttonframe, text="Write", relief='flat', bg="#E87721", fg='white', width=30, command=self.writeAction)
self.testBtn = tk.Button(self.buttonframe, text="Test", relief='flat', bg="#373737", fg='white', width=30, command=self.testAction)
# adding to screen
self.mainframe.pack(pady=2, fill=BOTH, expand=True)
self.treeframe.pack(pady=2, fill=X)
self.account_tree.pack(fill=X, pady=5)
self.buttonframe.pack(fill=X, pady=5)

```

Figure 261 Tree View for Desktop

```

def loadTable(self):
    data = getCards getToken()

    if isinstance(data, dict):
        from login import LoginGui
        self.root.destroy()
        LoginGui()
    else:
        self.account_tree.insert(parent='', index='end', iid='space', text="", values=( '', ' ', ' '))
        for i in range(len(data)):
            self.account_tree.insert(parent='', index='end', iid=i, text="", values=(data[i]['first_name'], data[i]['last_name'], data[i]['phone']))

```

Figure 262 Load Tree View in GUI

### 7.2.1.2 Mobile Data Tiles

```
@override
Widget build(BuildContext context) {
    return Container(
        padding: EdgeInsets.all(10.0),
        margin: EdgeInsets.only(top: 10.0, bottom: 10.0),
        decoration: BoxDecoration(
            color: Colors.white,
            borderRadius: BorderRadius.only(
                topLeft: Radius.circular(10),
                topRight: Radius.circular(10),
                bottomLeft: Radius.circular(10),
                bottomRight: Radius.circular(10)),
            boxShadow: [
                BoxShadow(
                    color: Colors.grey.withOpacity(0.1),
                    spreadRadius: 5,
                    blurRadius: 7,
                    offset: Offset(0, 3), // changes position of shadow
                ),
            ],
        ),
        child: Column(
            children: [
                Text(logData["entry_name"], style: nameStyle),
                Padding(
                    padding: EdgeInsets.all(3),
                    child: Text(
                        logData["entry_date"] + " at " + logData["entry_time"],
                        style: dateStyle,
                    ),
                ),
                Text(logData["exit_name"], style: nameStyle),
                Padding(
                    padding: EdgeInsets.all(3),
                    child: Text(
                        logData["exit_date"] + " at " + logData["exit_time"],
                        style: dateStyle,
                    ),
                ),
            ],
        ),
    );
}
```

Figure 263 Log Tile (1)

```
        Row(
            mainAxisAlignment: CrossAxisAlignment.center,
            mainAxisSize: MainAxisSize.center,
            children: [
                Padding(
                    padding: EdgeInsets.all(10),
                    child: Row(
                        mainAxisAlignment: CrossAxisAlignment.center,
                        children: [
                            Text("Fare: ", style: distanceStyle),
                            Text("Rs " + logData["fare"].toString(),
                                style: fareStyle)
                        ],
                    ),
                Padding(
                    padding: EdgeInsets.all(10),
                    child: Row(
                        mainAxisAlignment: CrossAxisAlignment.center,
                        children: [
                            Text("Distance: ", style: distanceStyle),
                            Text(logData["trip_distance"].toString() + " m",
                                style: distanceStyle)
                        ],
                    )));
            ],
        );
    },
);
});
```

*Figure 264 Log Tile (2)*

```
class LogContainer extends StatelessWidget {
    final String title;
    final Widget logTileCollection;

    LogContainer(this.title, this.logTileCollection);

    final TextStyle titleStyle =
        TextStyle(fontFamily: 'Montserrat', fontSize: 30.0, color: Colors.black);

    @override
    Widget build(BuildContext context) {
        return Container(
            margin: EdgeInsets.only(top: 30),
            padding: EdgeInsets.all(10),
            child: Column(
                children: <Widget>[
                    Text(
                        title,
                        style: titleStyle,
                    ),
                    logTileCollection,
                ],
            ),
        );
    }
}
```

*Figure 265 Log Tile Container*

### 7.2.1.3 Web Forms

```
<h2 class="mt-3">{{ customer.first_name }} {{customer.last_name}}</h2>
<div class="m-1 p-1 d-flex flex-wrap justify-content-center align-items-center" style="min-height:50vh">
    <div class="p-2 mr-3">
        
    </div>
    <div class="p-2 ml-3">
        
    </div>
</div>
<div class="mt-2 p-1">
    <p>Customer Status: {{ customer.status }} </p>
</div>
<hr>
<div class="d-flex flex-wrap justify-content-around align-items-center" >
    <div class="p-1">
        <form action="" method="POST">
            {% csrf_token %}
            {% for field in form %}
                <p class="form-label">{{field.label}}</p>
                {{field|add_class:"form-control"}}
            {% endfor %}
            {% for field in discount_form %}
                <div class="d-flex">
                    <p class="form-label">{{field.label}}</p>{{field|add_class:"form-control"|add_class:"shadow-none"}}
                </div>
            {% endfor %}
            <input class="btn" type = "submit" value = "Save" style="background-color:#E87721;color:white">
        </form>
    </div>
    <div class="d-flex flex-column justify-content-start p-1" style="text-align:right">
        <h4>Contact {{customer.first_name}}</h4>
        <p>Phone {{customer.phone}}</p>
        <p>Email {{customer.email}}</p>
    </div>
</div>
```

Figure 266 General Form Template in Web Application

#### 7.2.1.4 Web Tables

```
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item">
    <a class="nav-link" id="unverified-tab" data-toggle="tab" href="#unverified" role="tab" aria-controls="unverified" aria-selected="true">Unverified</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" id="resubmission-tab" data-toggle="tab" href="#resubmission" role="tab" aria-controls="resubmission" aria-selected="false">Resubmission Required</a>
  </li>
  <li class="nav-item">
    <a class="nav-link active" id="pending-tab" data-toggle="tab" href="#pending" role="tab" aria-controls="pending" aria-selected="false">Pending Review</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" id="verified-tab" data-toggle="tab" href="#verified" role="tab" aria-controls="verified" aria-selected="false">Verified</a>
  </li>
</ul>
```

Figure 267 Table Tabs in Table

```
<div class="tab-content" id="myTabContent">
  <div class="tab-pane fade " id="unverified" role="tabpanel" aria-labelledby="unverified-tab">
    <table class="table table-sm">
      <tr>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Phone</th>
        <th>Action</th>
      </tr>
      {% for unverified_customer in unverified_customers %}
      <tr>
        <td>{{unverified_customer.first_name}}</td>
        <td>{{unverified_customer.last_name}}</td>
        <td>{{unverified_customer.phone}}</td>
        <td><a href="{% url 'customerPage' unverified_customer.id %}">View</a></td>
      </tr>
      {% endfor %}
    </table>
  </div>
  <div class="tab-pane fade" id="resubmission" role="tabpanel" aria-labelledby="resubmission-tab">
    <table class="table table-sm">
      <tr>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Phone</th>
        <th>Action</th>
      </tr>
      {% for resubmission_customer in resubmission_customers %}
      <tr>
        <td>{{resubmission_customer.first_name}}</td>
        <td>{{resubmission_customer.last_name}}</td>
        <td>{{resubmission_customer.phone}}</td>
        <td><a href="{% url 'customerPage' resubmission_customer.id %}">View</a></td>
      </tr>
      {% endfor %}
    </table>
  </div>
</div>
```

Figure 268 Table Tabs in Admin Dashboard

## 7.2.2 Sample Code of the Applications

### 7.2.2.1 Read / Write Cards

```
import RPi.GPIO as GPIO
# import location
from mfrc522 import SimpleMFRC522

def writeCard(cardNum):
    reader = SimpleMFRC522()
    try:
        reader.write(cardNum)
    except:
        return False
    else:
        return True
```

Figure 269 Write Card

```
import RPi.GPIO as GPIO
from mfrc522 import SimpleMFRC522

reader=SimpleMFRC522()

def readCard():
    text = None
    try:
        id, text = reader.read()
    except:
        return text
    else:
        return str(text).strip()
```

Figure 270 Write Card

### 7.2.2.2 Server Requests using Python

```
# method to make header for requests
def makeheader(token):
    tokenheader = 'Token {}'.format(token)
    header = {'Authorization': tokenheader}
    return header

# request to get login token
def loginAdmin(username,password):
    data = {'username':username,'password':password}
    r = requests.post(url = login_url, data = data)
    r = r.json()
    return r

# request that gets account details for people without card written
def getCards(token):
    header = makeheader(token)
    r = requests.get(url = get_card_users_url, headers = header)
    r = r.json()
    return r

# request that gets the card number for a selected account
def getCardNumber(token, phone):
    header = makeheader(token)
    r = requests.get(url = write_cards_url+'/'+phone, headers = header)
    try:
        r = r.json()
    except:
        r = {'detail':'Card not found'}
    return r

# request that gets account detail for a given card number
def checkCorrectUser(token, card_num):
    header = makeheader(token)
    r = requests.get(url = test_set_card_url + '/' + card_num, headers=header)
    try:
        r = r.json()
    except:
        r = {'detail':'Card not found'}
    return r
```

Figure 271 Python Server Requests

## 7.3 Appendix D: Designs

### 7.3.1 Gantt Chart

#### 7.3.1.1 Final Gantt Chart

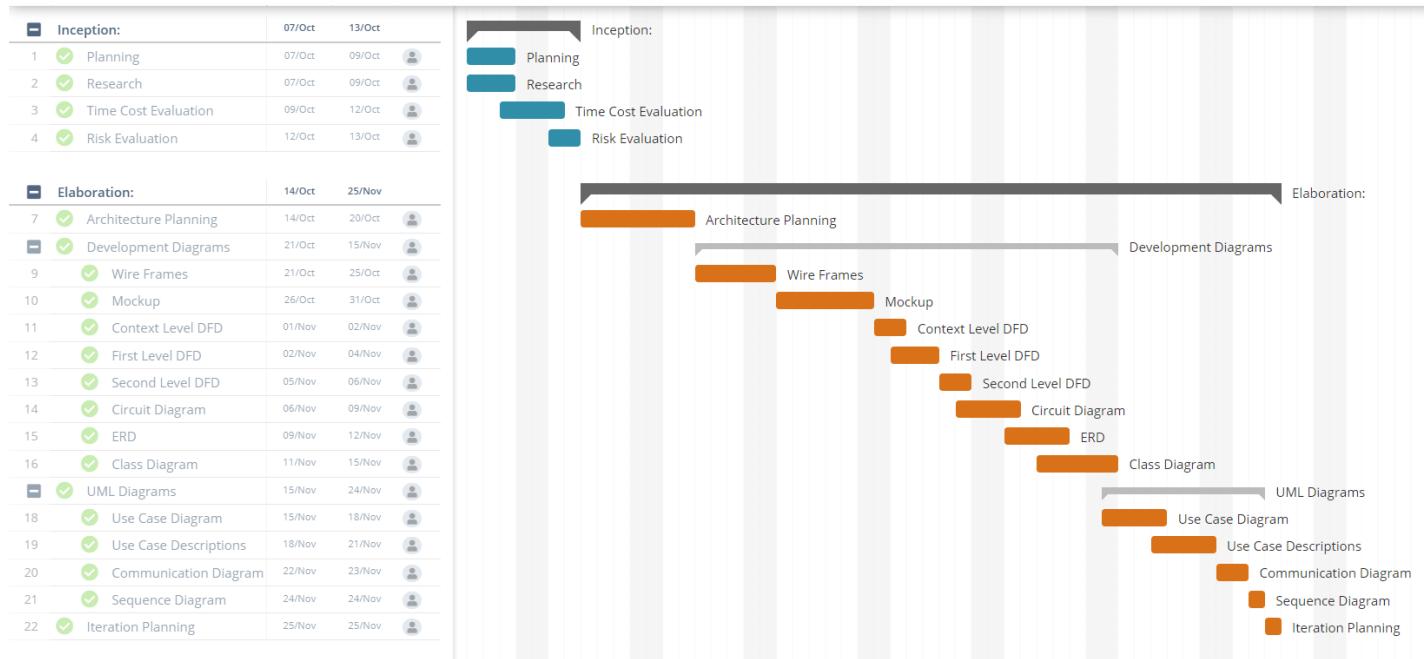


Figure 272 Final Gantt Chart for Inception and Elaboration

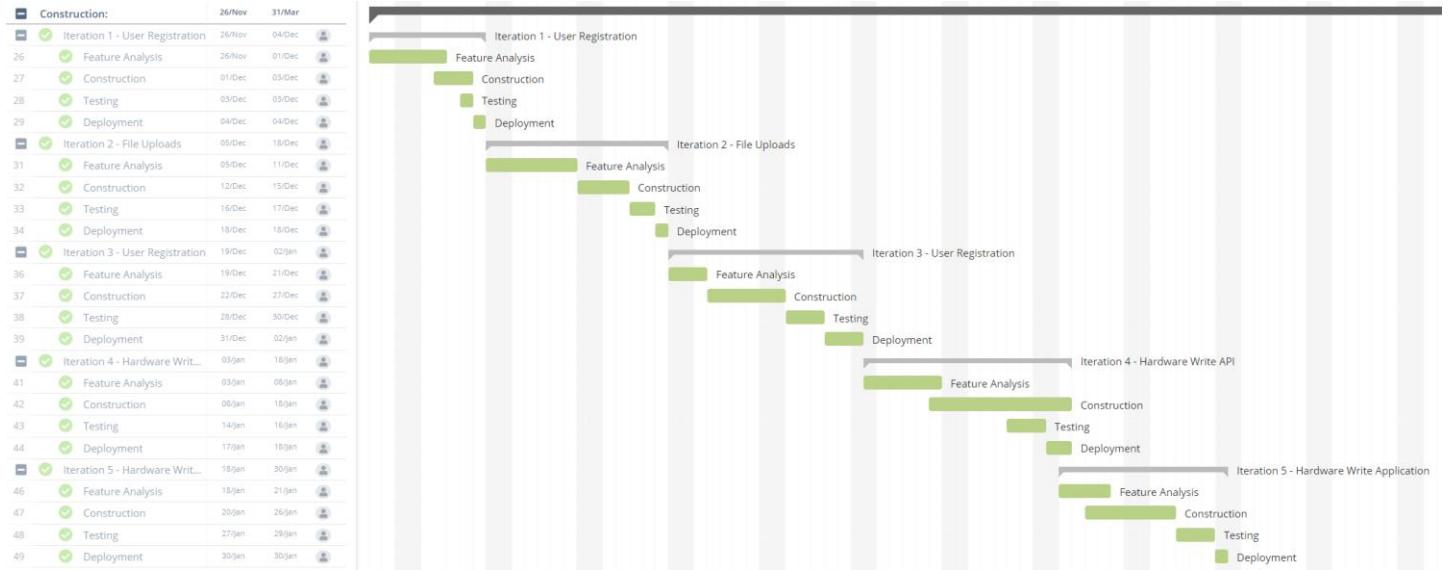


Figure 273 Final Gantt Chart for First 5 Iterations

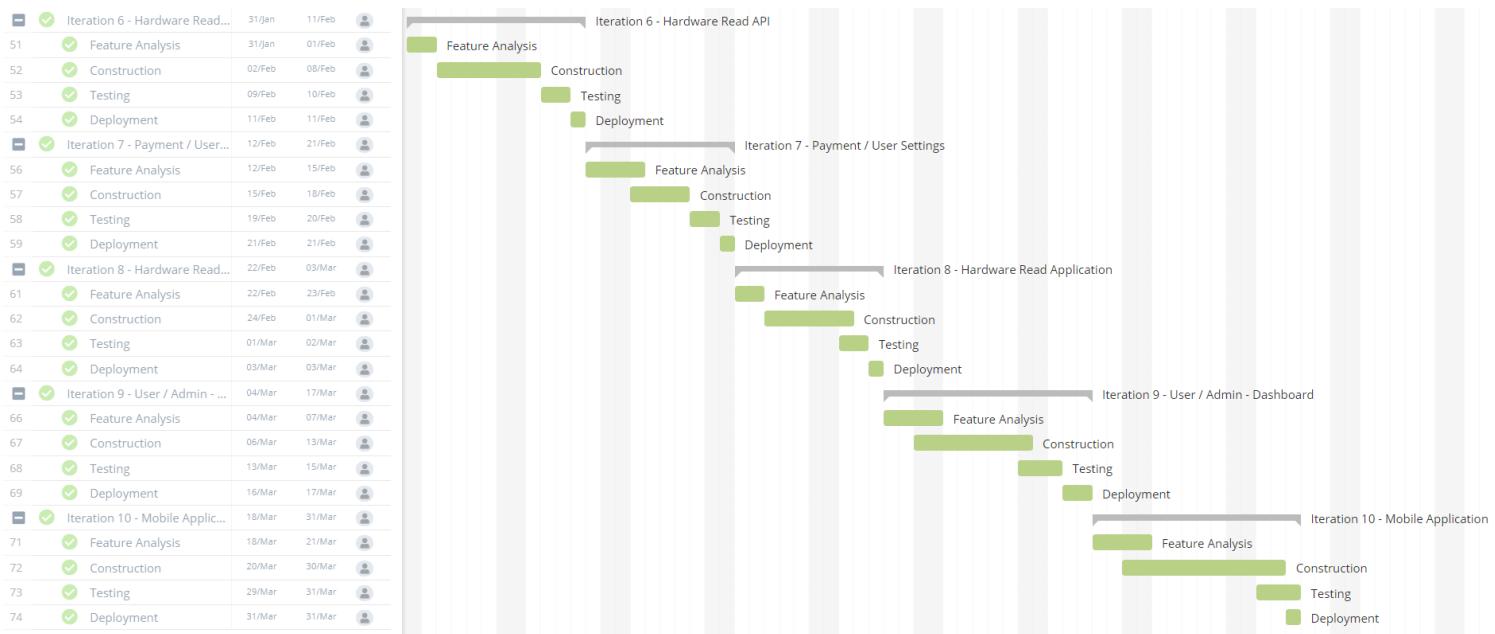


Figure 274 Final Gantt Chart for Last 5 Iterations

Transition:	01/Apr	17/Apr
77 ✓ Deployment Execution	01/Apr	04/Apr
78 ✓ Test Deliverable Product	05/Apr	13/Apr
79 ✓ User Feedback	13/Apr	17/Apr



Figure 275 Final Gantt Chart for Transition

### 7.3.1.2 Gantt Chart Iterations

These are the revisions of Gantt Chart that led to the final chart.

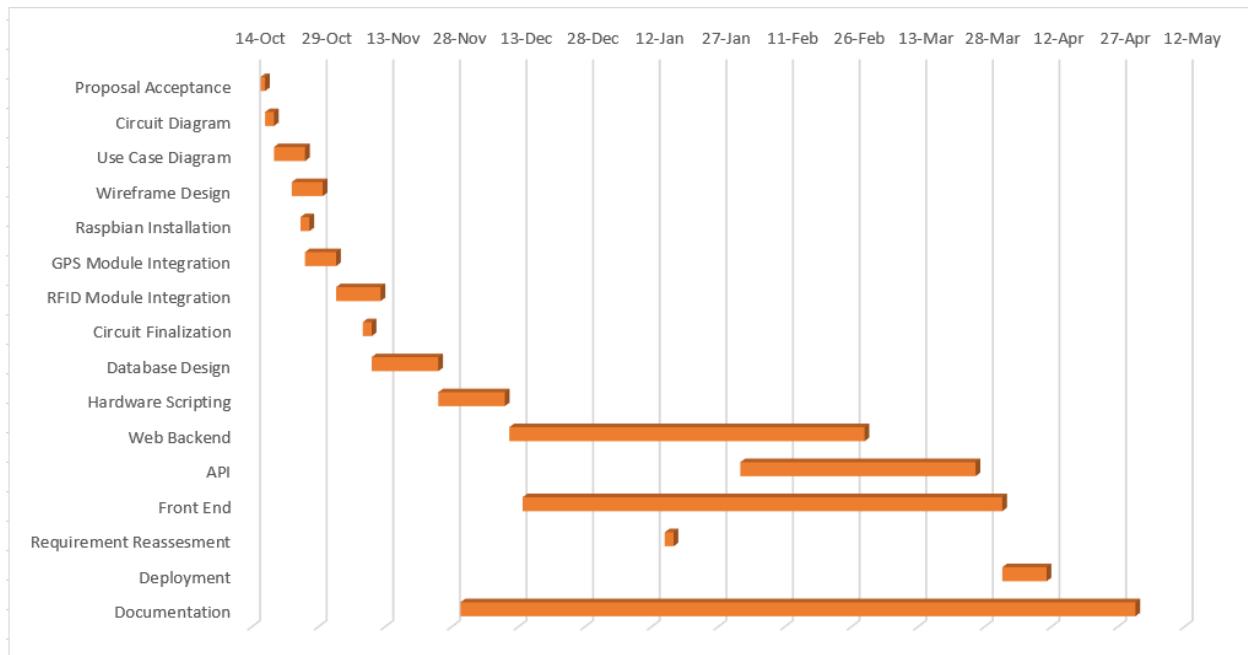


Figure 276 First Gantt Chart Revision

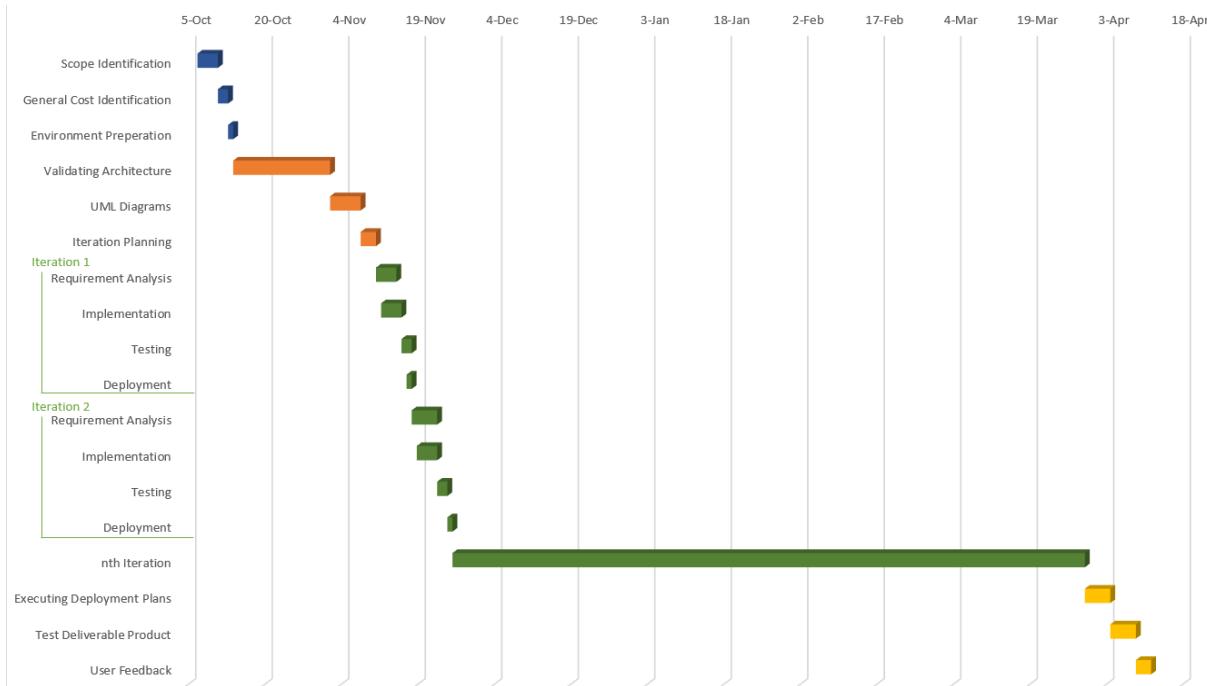


Figure 277 Second Gantt Chart Revision

### 7.3.2 Work Breakdown Structure

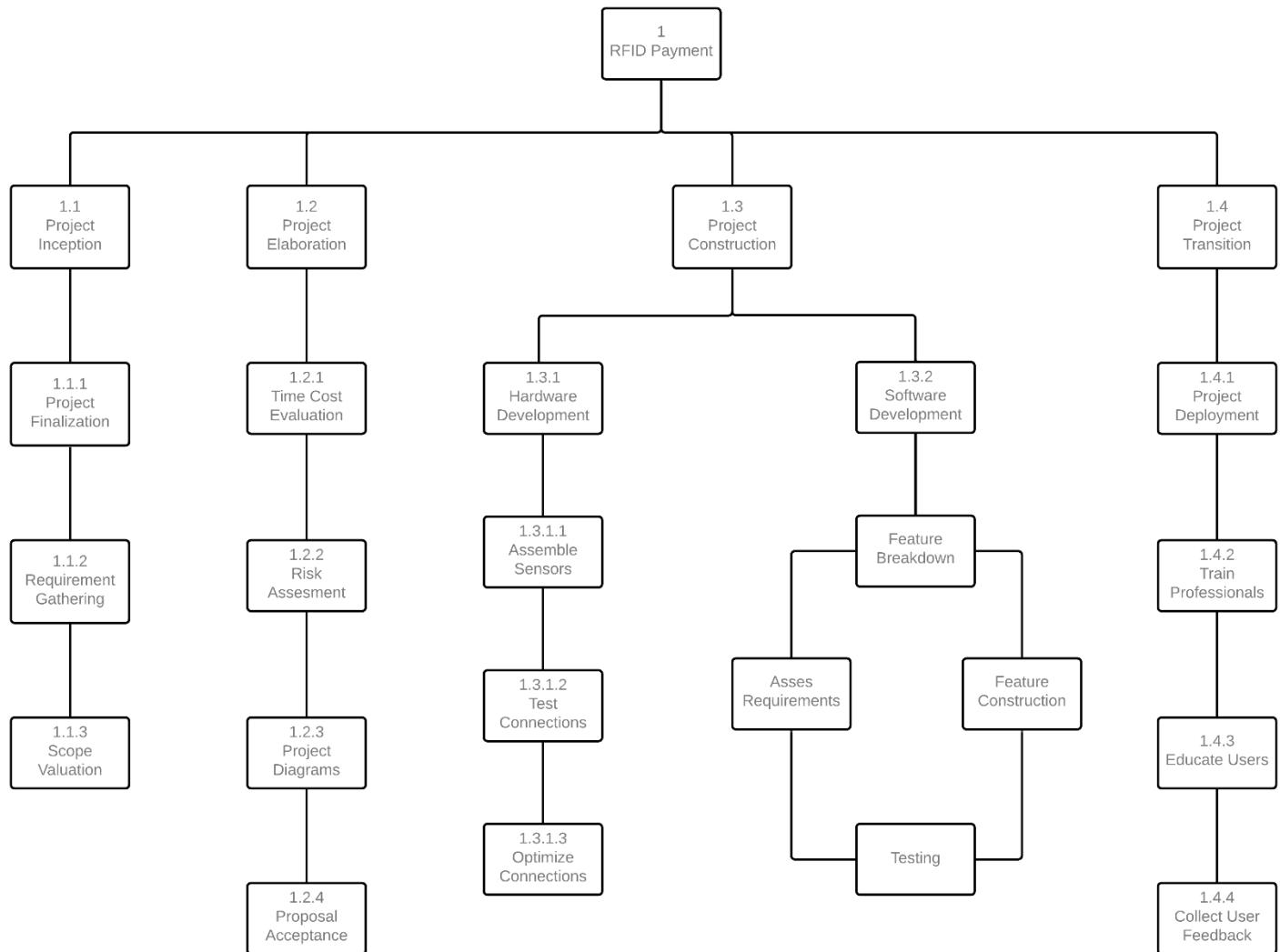


Figure 278 Work Breakdown Structure Chart

### 7.3.3 Hardware Architecture

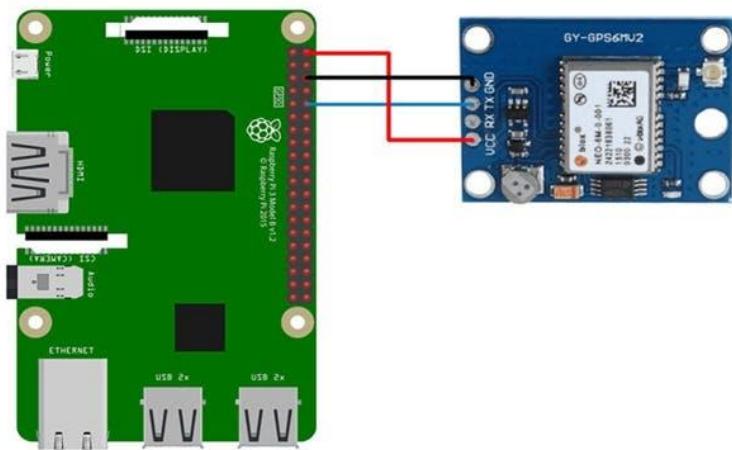
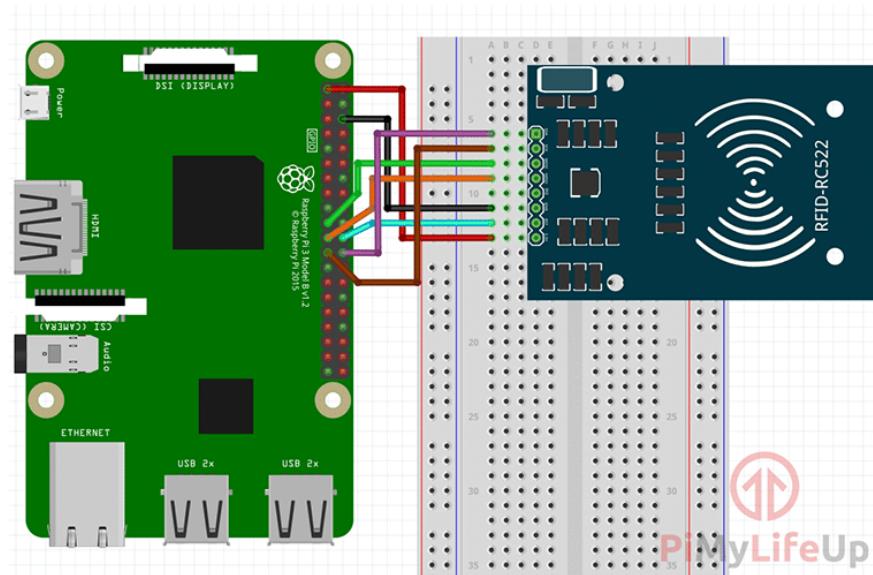


Figure 279 Circuit Diagram for GPS Module (Hackster, 2019)



### 7.3.4 ERD Iterations

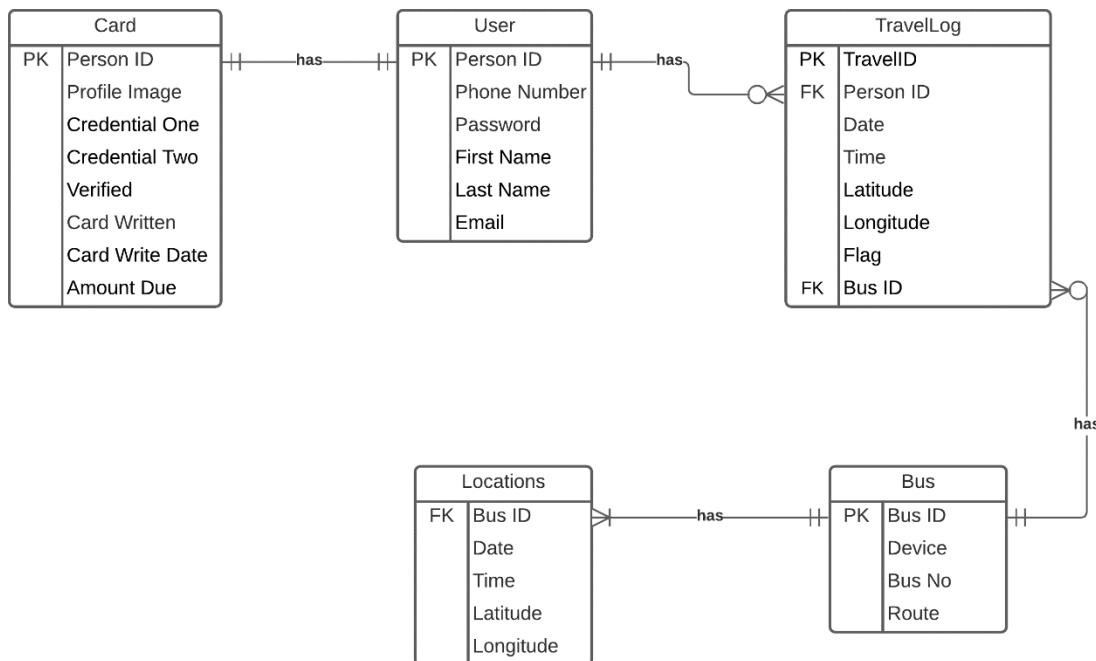


Figure 281 ER Diagram First Iteration

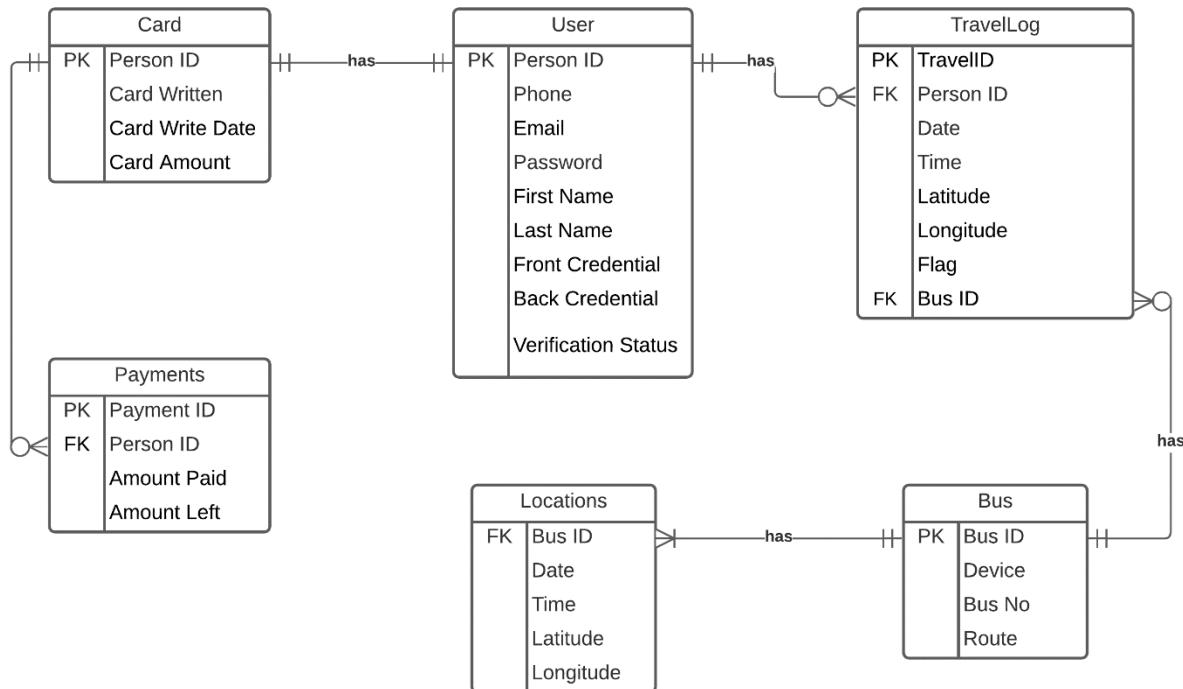


Figure 282 ER Diagram Second Iteration

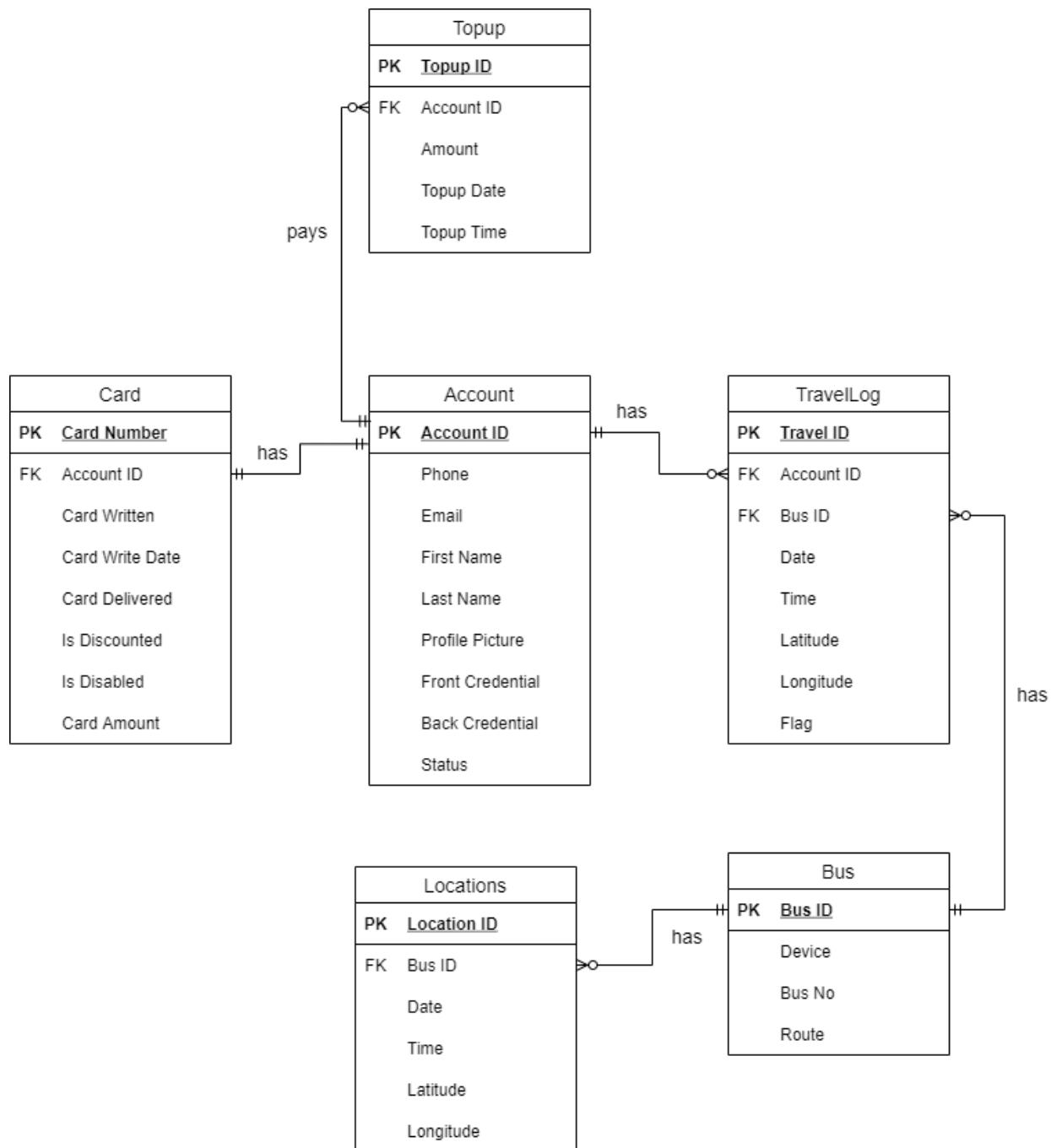


Figure 283 ER Diagram Third Iteration

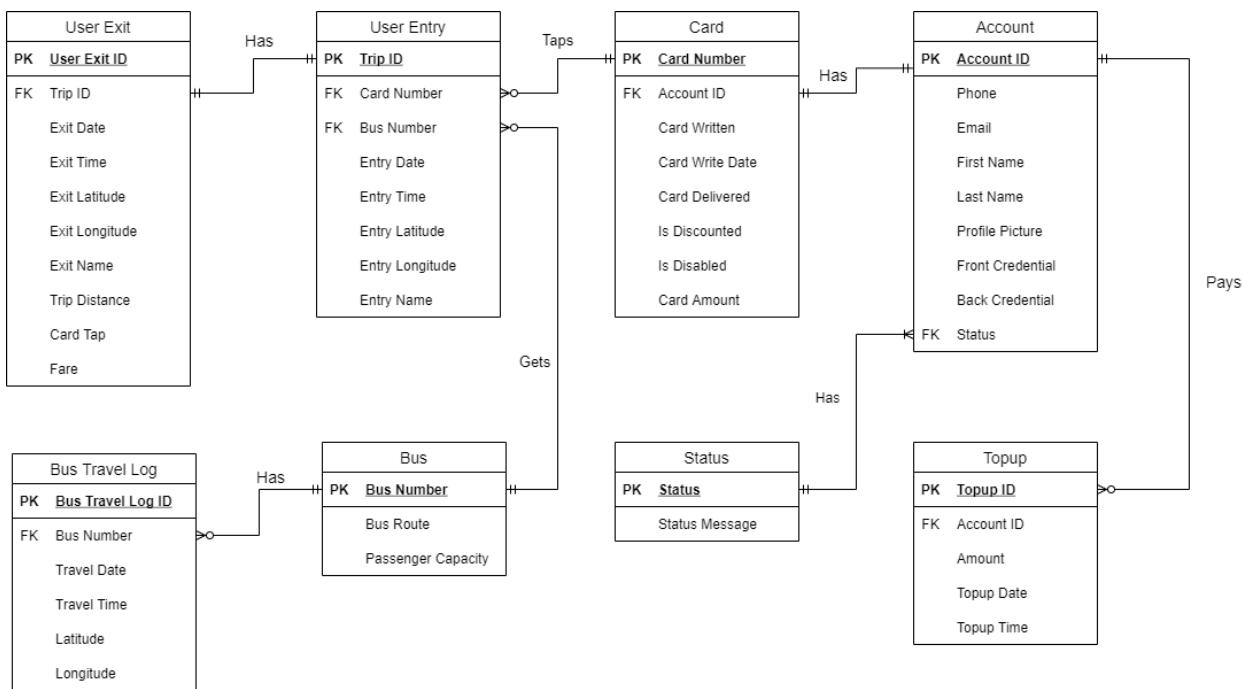


Figure 284 ER Diagram Final Iteration

### 7.3.5 Data Flow Diagrams

To understand the overall flow of data throughout the project components, data flow diagrams of various levels have been created. These diagrams have helped to shape the necessary development on how one module communicates with another.

#### 7.3.5.1 Context Level DFD

This diagram represents the most abstract form of view for the project which defines how entities of the project give or take data.

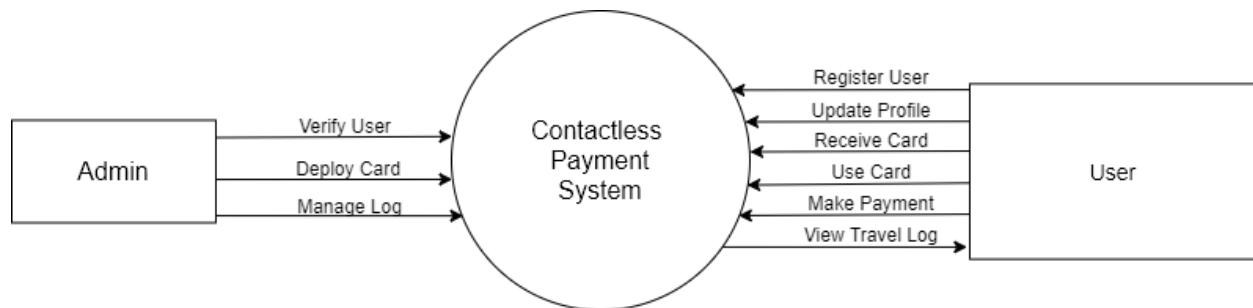


Figure 285 Context Level DFD

Level 2 DFD Diagrams have been shown in the [design section of the report](#).

### 7.3.5.2 Level 1 DFD

Expanding the context level data flow diagram into a more structured approach we get,

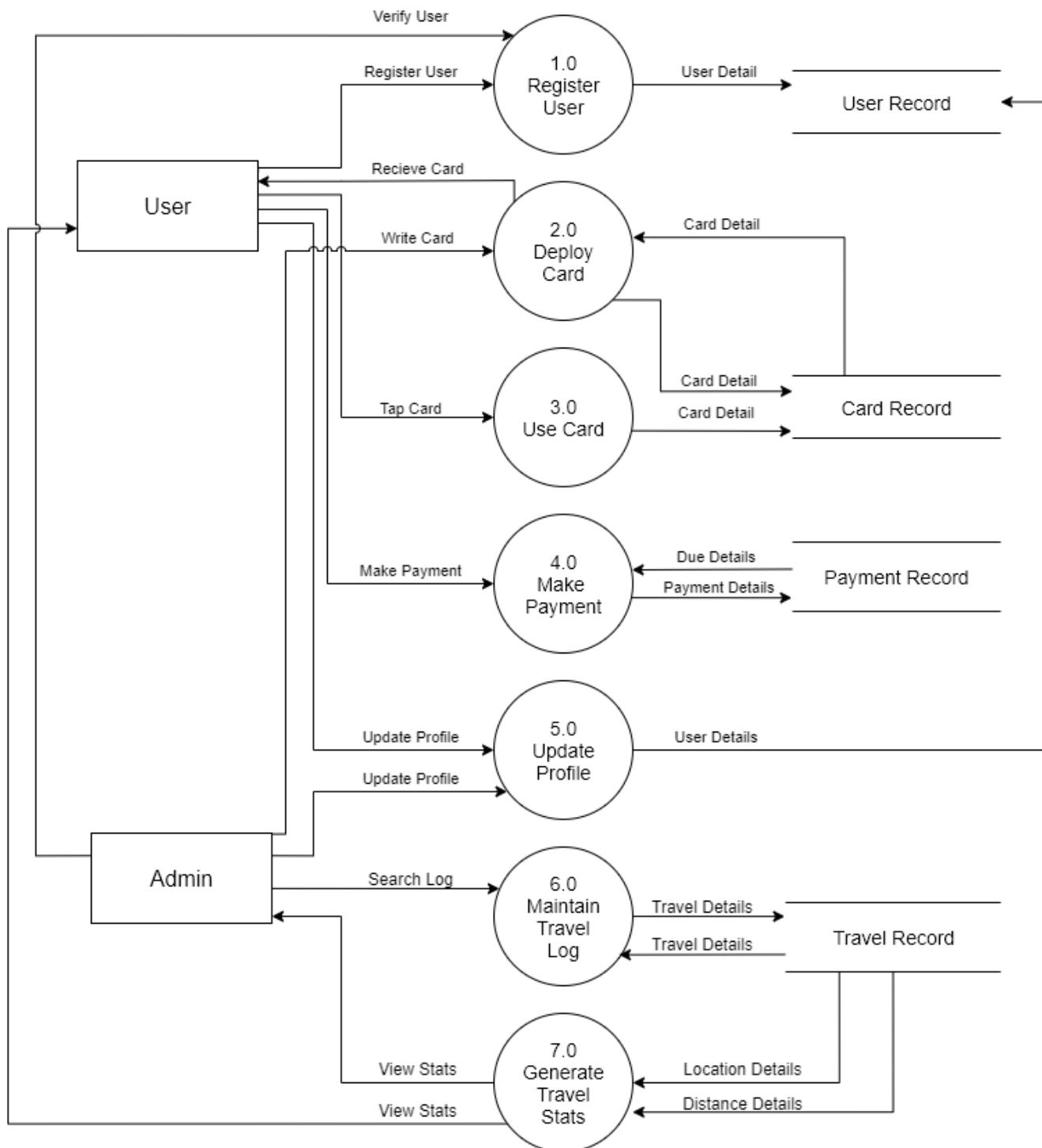


Figure 286 Level 1 DFD

### 7.3.5.3 Level 2 DFD Iterations

#### 7.3.5.3.1 Sign Up User

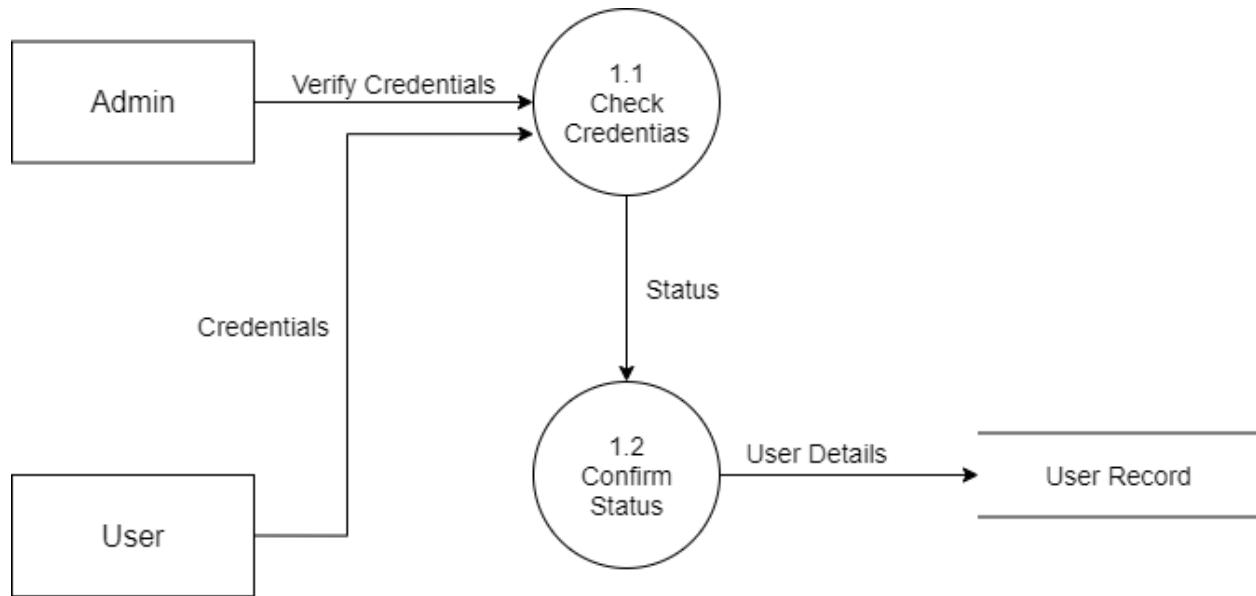


Figure 287 Sign Up User Level 2 DFD First Iteration

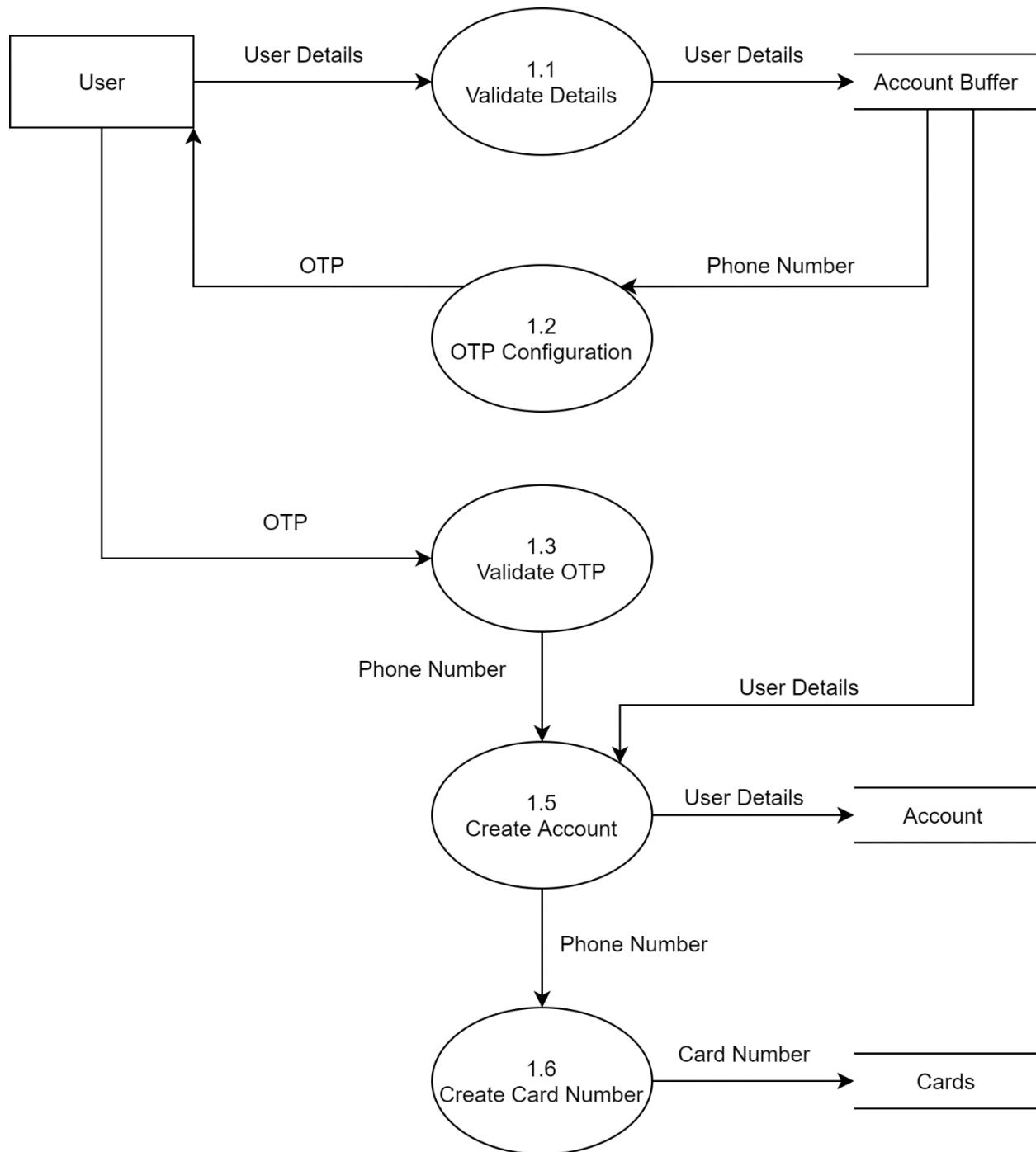


Figure 288 Sign Up User Level 2 DFD Second Iteration

### 7.3.6 Use Case Diagrams

#### 7.3.6.1 Final Use Case Diagram

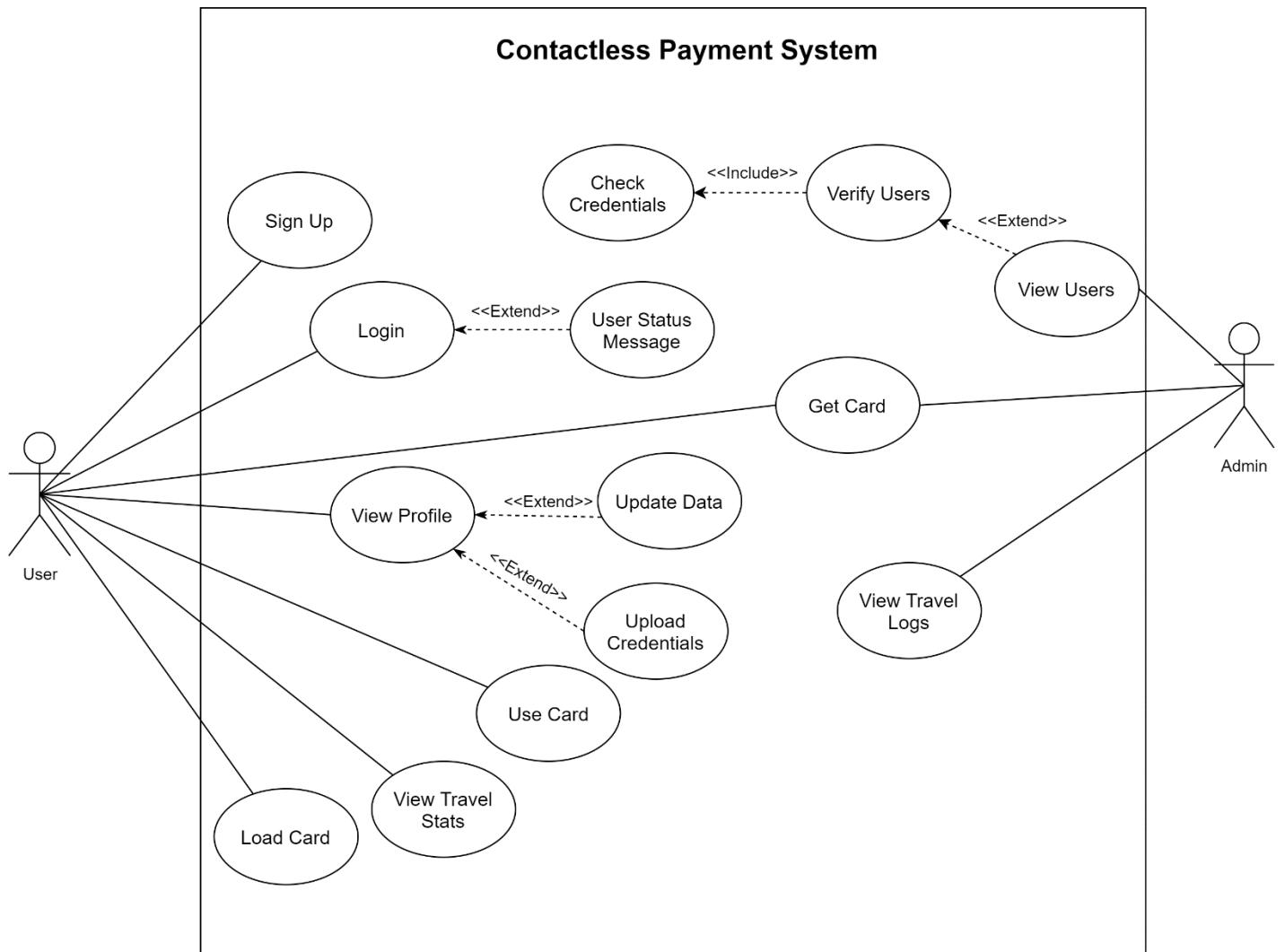


Figure 289 Final Use Case Diagram

### 7.3.6.2 Use Case Iterations

Iterations of use case diagram throughout development excluding final diagram.

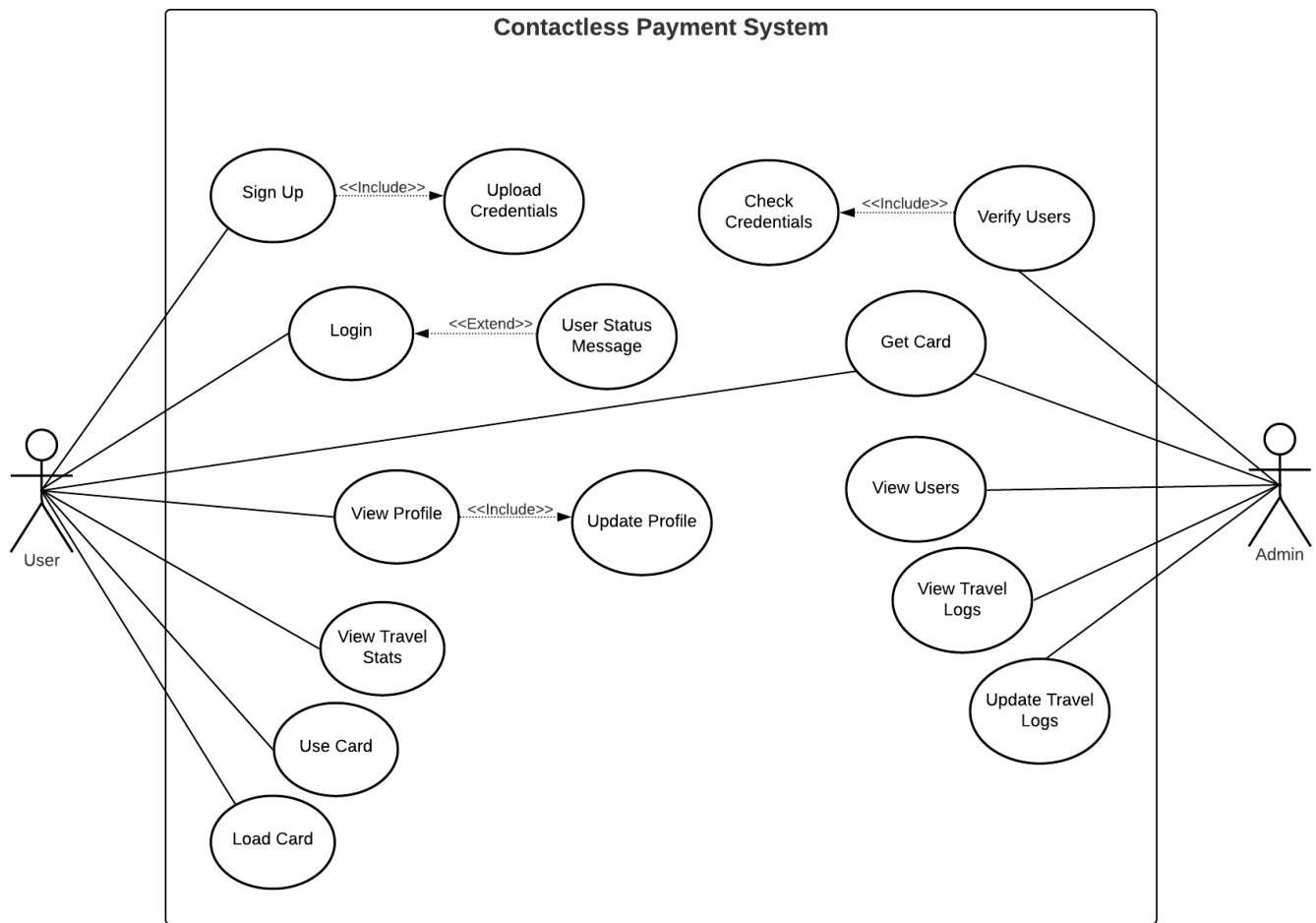


Figure 290 Use Case Diagram First Iteration

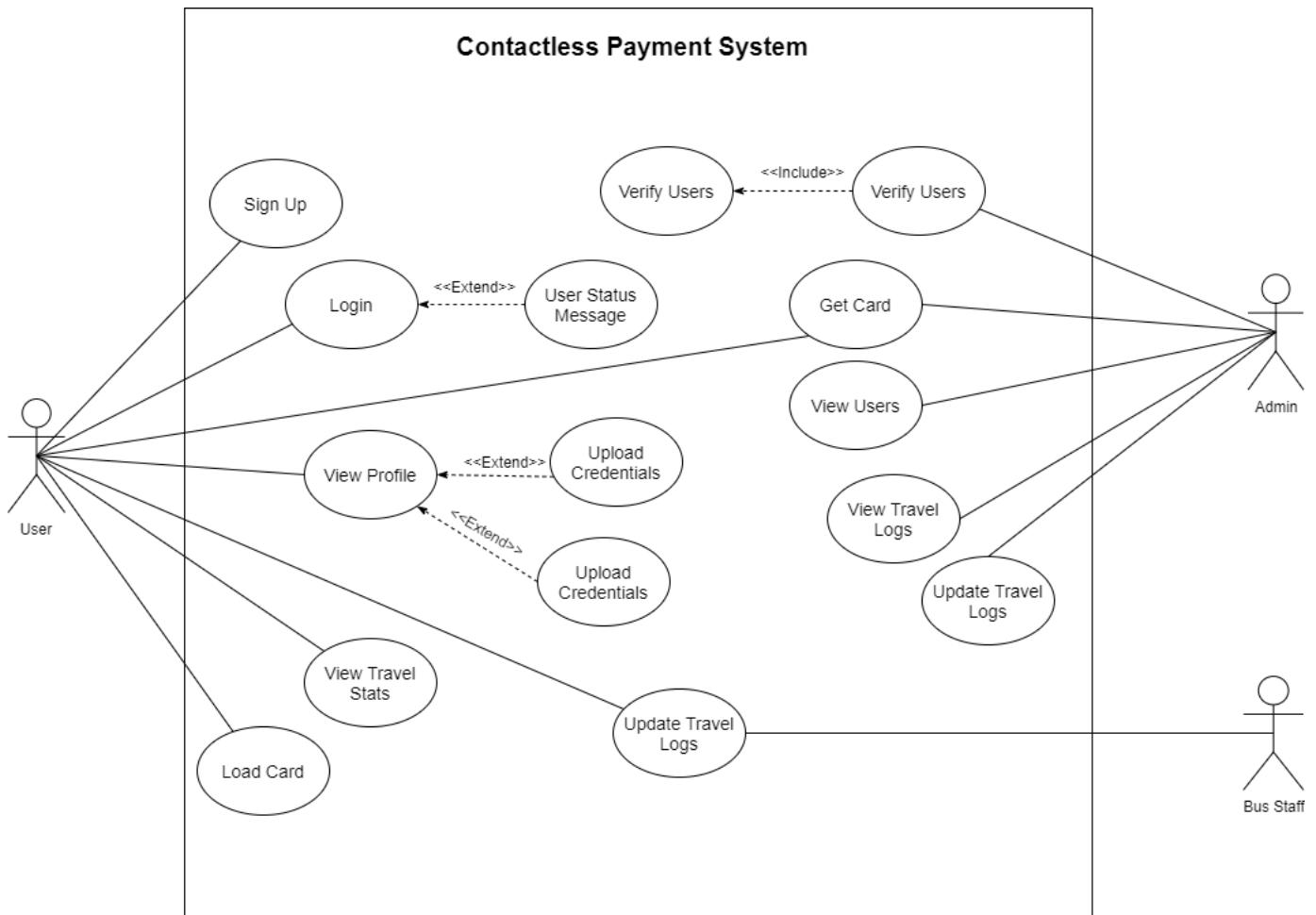


Figure 291 Use Case Diagram Second Iteration

### 7.3.6.3 High Level Use Case Description

Following are the high level use case description that includes all the base cases required for the contactless payment system to function.

- Case 1
  - Name: Sign Up
  - Actor(s): User
  - Description: Any user, who wants to start using the system should first sign up through the web application which includes them verifying their phone number for registration.
- Case 2
  - Name: Login
  - Actor(s): User
  - Description: After registration, users can log into the service to view their travel data or if they're not verified, they can view if their verification status.
- Case 3
  - Name: Get Card
  - Actor(s): User, Admin
  - Description: After registration, the user is queued for card write. The administration writes the card for the user and contacts the user to get it.
- Case 4
  - Name: View Profile
  - Actor(s): User
  - Description: A user who has started using the payment service can view their details and submitted credentials. They can update their profile if they deem it necessary.
- Case 5
  - Name: View Travel Stats
  - Actor(s): User

- Description: Any user who has been travelling and paying through their card can view their travel history and meaningful statistics in the web or mobile application.
- Case 6
  - Name: Use Card
  - Actor(s): User
  - Description: Users can tap the card on entry and exit through the card reader screen.
- Case 7
  - Name: Load Card
  - Actor(s): User
  - Description: To pay for bus fares, users need to prepay their cards or add top up to their cards which runs out as they pay for their travel fares.
- Case 8
  - Name: Verify Users
  - Actor(s): Admin
  - Description: For a user to add significant amount top-up to their card, they need to provide their credentials in the web application which will be verified by the administration.
- Case 9
  - Name: View Users
  - Actor(s): Admin
  - Description: When required, the admin is able to view user profiles which contains their details and their submitted credentials.
- Case 10
  - Name: View Travel Logs
  - Actor(s): Admin
  - Description: The admin can view abstract form of travel log to monitor the performance and the data being generated by all the users using the card

to pay, this can be then used to add new features or change existing features.

- Case 11
  - Name: Update Travel Logs
  - Actor(s): Admin
  - Description: The admin gets to see faulty travel records in the system which then should be updated by the admin after contacting the user to verify the specific details of the travel record.

### 7.3.7 Wireframes

Necessary wireframes for the base pages for web and mobile version of the application have been given below.

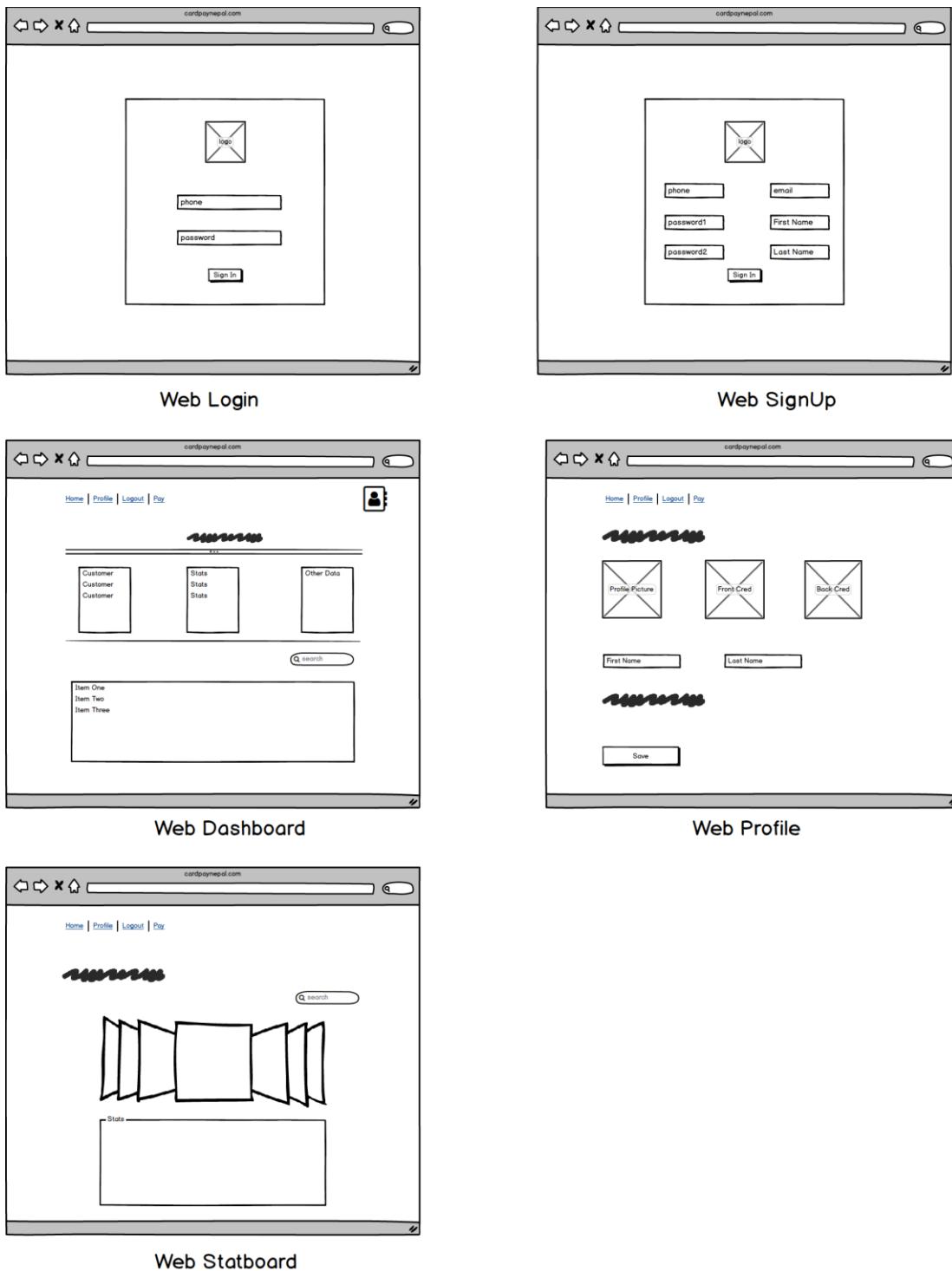
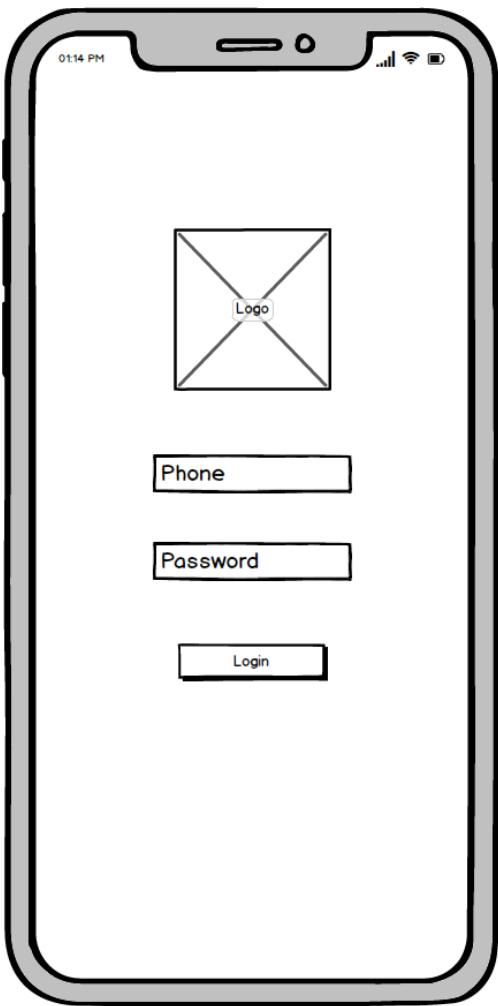
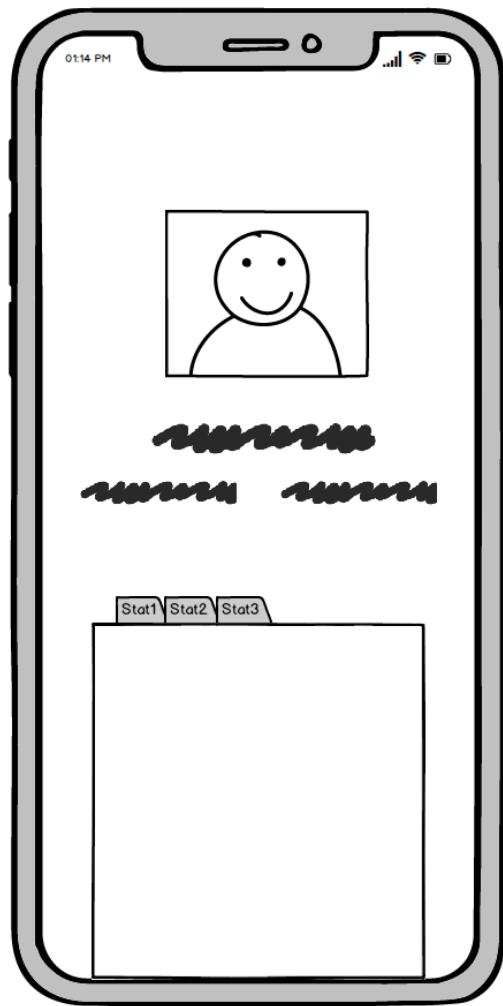


Figure 292 Wire Frames for Base Web Pages



Mobile Login



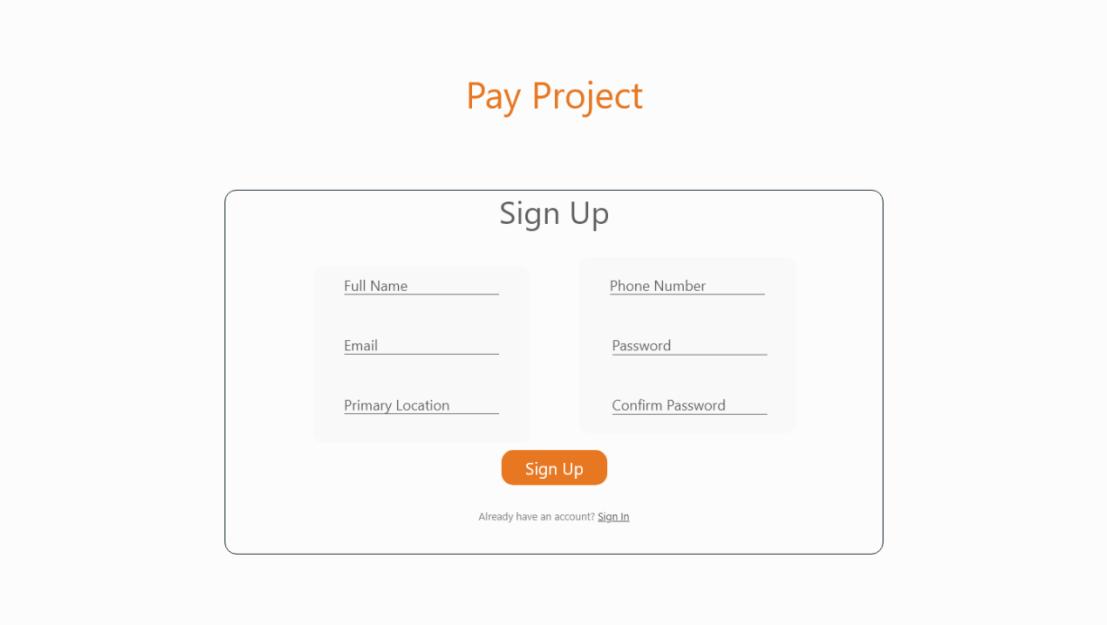
Mobile Dash/Stat board

Figure 293 Wireframes for Mobile Application

### 7.3.8 Mock-ups

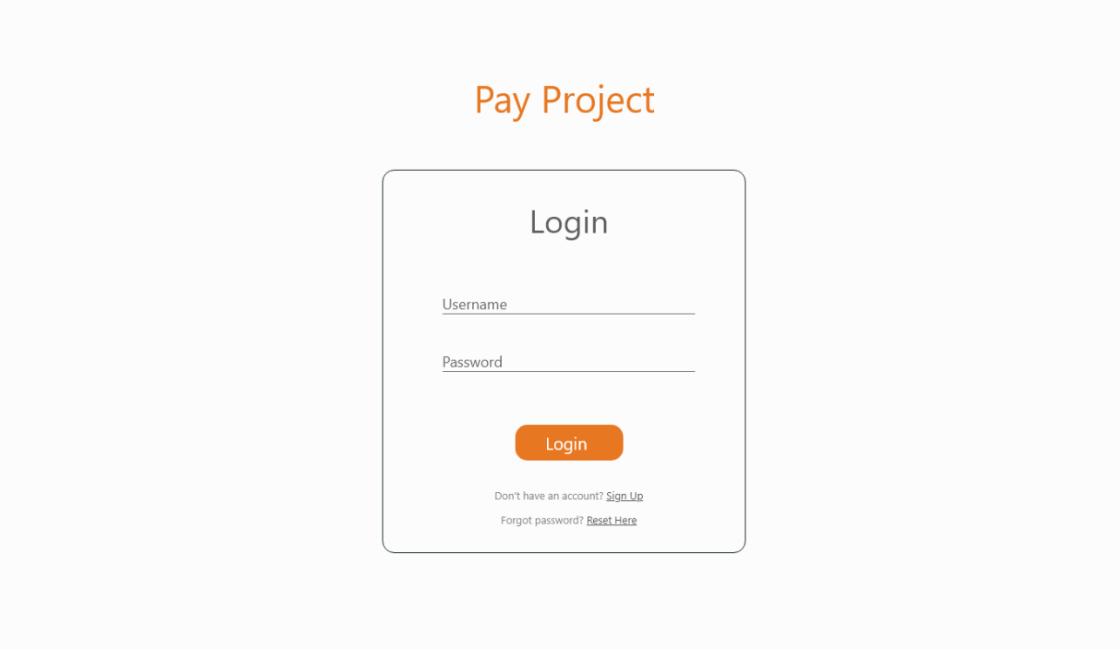
Basic mockups for the web and mobile application have been given below.

#### 7.3.8.1 Web Mockups



The image shows a web-based sign-up form titled "Pay Project". The form is contained within a rounded rectangular container. At the top center, it says "Sign Up". Below this, there are two rows of input fields. The first row contains "Full Name" and "Phone Number". The second row contains "Email" and "Password". In the third row, there are "Primary Location" and "Confirm Password". A large orange "Sign Up" button is centered at the bottom of the form. Below the button, small text indicates "Already have an account? [Sign In](#)".

Figure 294 Web Signup Mockup



The image shows a web-based login form titled "Pay Project". The form is contained within a rounded rectangular container. At the top center, it says "Login". Below this, there are two input fields: "Username" and "Password". A large orange "Login" button is centered at the bottom of the form. Below the button, small text indicates "Don't have an account? [Sign Up](#)" and "Forgot password? [Reset Here](#)".

Figure 295 Web Login Mockup

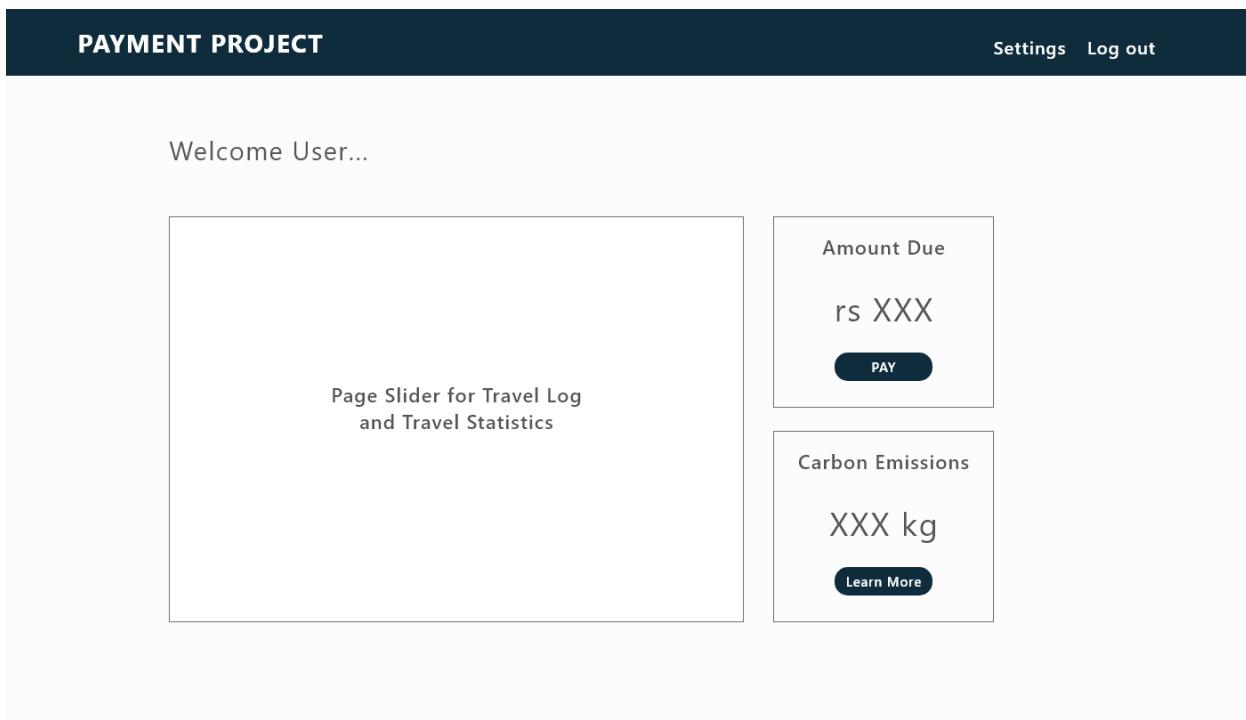


Figure 296 Web User Dashboard Mockup

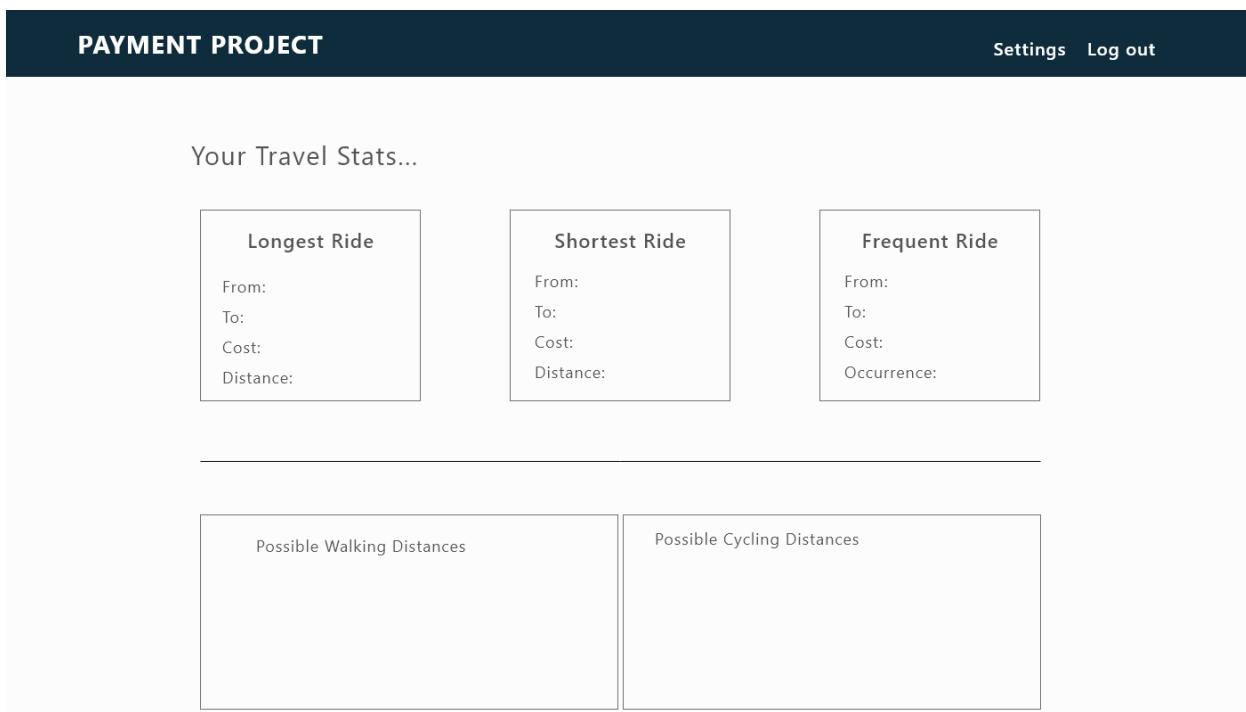


Figure 297 Web User Travel Stat board Mockup

PAYMENT PROJECT

[Settings](#) [Log out](#)

Travel Log From xx/xx/xx - xx/xx/xx

Time	From	To	Minutes	Distance	Cost

[January](#)

[February](#)

[March](#)

[April](#)

[May](#)

[June](#)

[July](#)

[August](#)

[September](#)

[October](#)

[November](#)

[December](#)

*Figure 298 Web User Travel Log Mockup*

PAYMENT PROJECT

[Settings](#) [Log out](#)

User's Settings

A large, empty rectangular box with rounded corners, representing a placeholder for a user's profile picture. At the bottom center of this box is the text "Change Image".

Full Name \_\_\_\_\_

Email \_\_\_\_\_

Phone Number \_\_\_\_\_

[Change Password](#)

[Submit Changes](#)

*Figure 299 Web User Profile Mockup*

**PAYMENT PROJECT**

**Log out**

### Today's stats

Total Users  
  
X

Verifications Left  
  
X

Card Writes Left  
  
X

---

Search

Option 1

Search Category Field GO

*Figure 300 Web Admin Dashboard Mockup*

**PAYMENT PROJECT**

**Log out**

### Your users

All Users	Registered	Verification Left	Card Writes Left		
Card No	Name	Status	No of Trips	Due	Action
xxx	xxx	Verification Left	None	None	<b>Verify</b>
xxx	xxx	Registered	xxx	xxx	<b>View</b>

*Figure 301 Web Admin View User Mockup*

## PAYMENT PROJECT

[Log out](#)

User Name

Credentials

Img1

Img2

Travel Log From xx/xx/xx - xx/xx/xx

Time	From	To	Minutes	Distance	Cost

JanuaryFebruaryMarchAprilMayJune

JulyAugustSeptemberOctoberNovemberDecember

Figure 302 Web Admin View User Page Mockup

### 7.3.8.2 Mobile Mockups

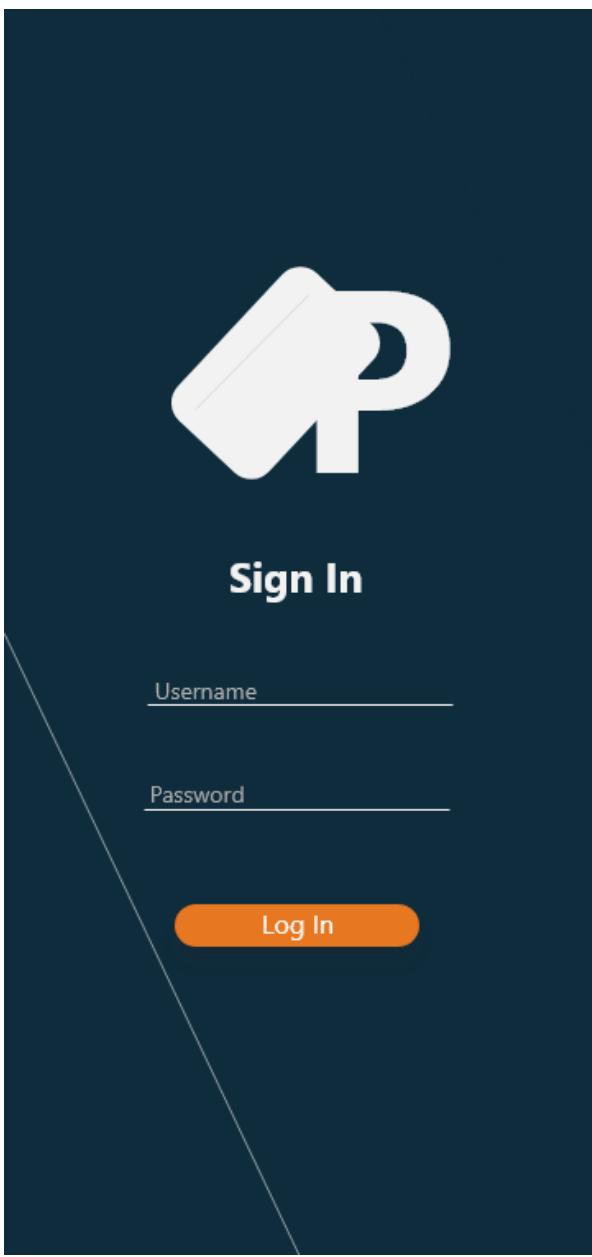


Figure 303 Mobile Login Screen Mockup

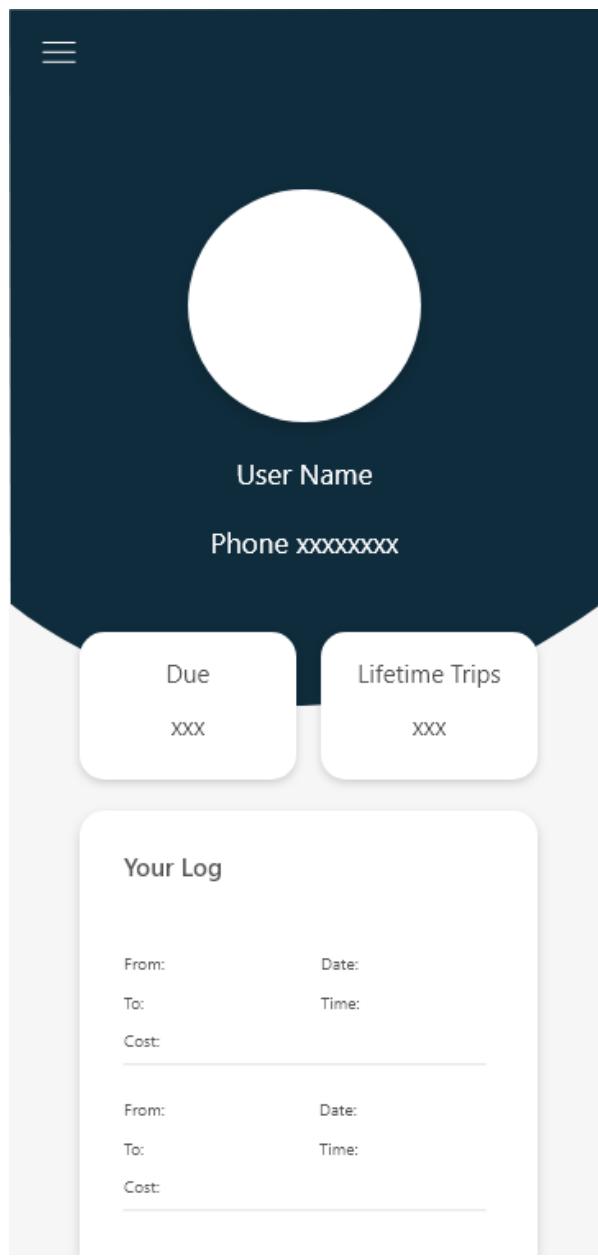


Figure 304 Mobile Dashboard Screen Mockup

## 7.4 Appendix E: Screenshots of the System

### 7.4.1 Web Application

#### 7.4.1.1 Customer

**Card Pay Register**

First Name...

Last Name...

 +977

Email...

Enter Password...

Re-enter Password...

**Register Account**

Already have an account? [Login](#)

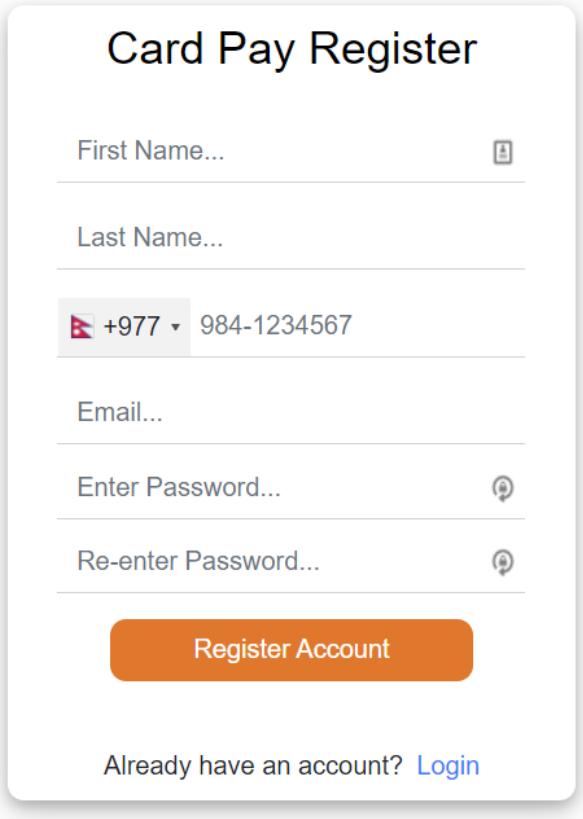


Figure 305 Registration UI

**Verify Phone**

**Verify Token**

Enter the code sent via SMS

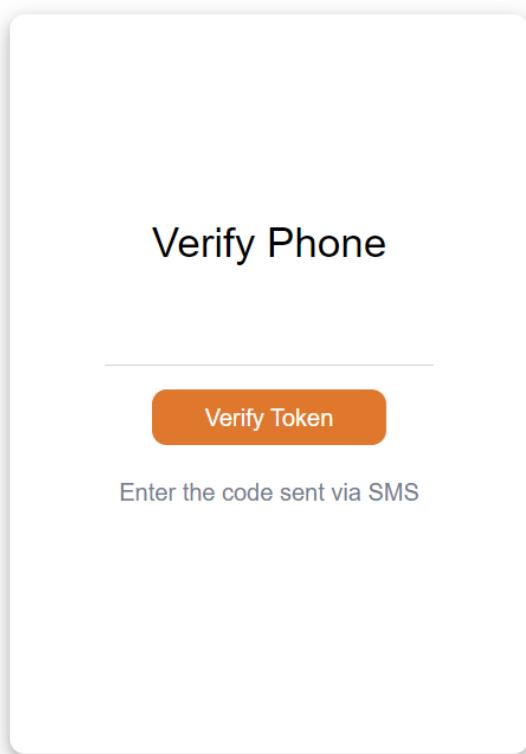


Figure 306 Verify Phone UI

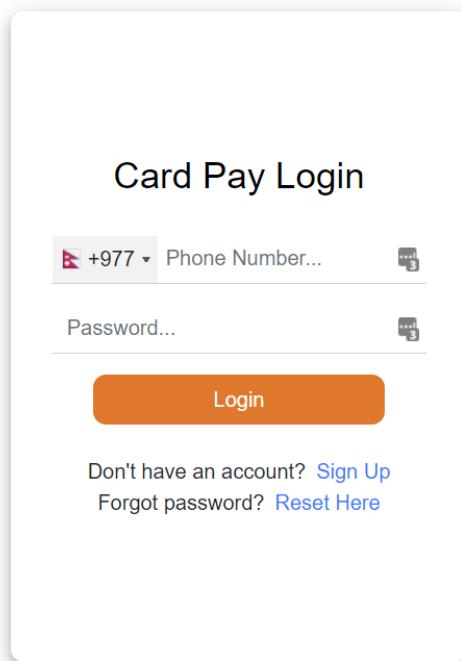


Figure 307 Login UI

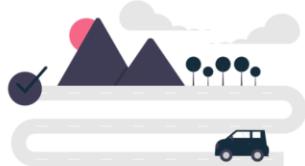
 A screenshot of the user dashboard. The top navigation bar includes a logo, 'Home', 'Profile', 'Topup', and 'Logout'. The main section has a title 'Dash Board' with three summary boxes: 'Card Amount' (Rs 120), 'Distance Travelled' (51.83 KM), and 'Total Trips' (28). Below this is a 'Travel History' table with five rows of data. At the bottom, there is a page navigation bar with buttons for '«', '1', '2', '3', '4', '5', '6', '»'.
 

Entry Location	Exit Location	Entry Time	Exit Time	Distance (meters)	Cost (Rs)
Chandan Marg, Kathmandu	Chandan Marg, Kathmandu	April 2, 2021 3:59 p.m.	April 2, 2021 3:59 p.m.	1	0
Dillibazar Pipalko Bot Bus Stop, Dilli Bazaar Rd	Bag Bazar Sadak, Kathmandu	March 12, 2021 11:45 a.m.	March 12, 2021 12:03 p.m.	1084	7
Pushupati Rd, Kathmandu	Pushupati Rd, Kathmandu	March 11, 2021 11:49 a.m.	March 11, 2021 12:05 p.m.	1724	12
Dilli Baazar Sadak, Kathmandu	Sinamangal Rd, Kathmandu	March 11, 2021 5:43 p.m.	March 11, 2021 6:03 p.m.	2194	15
Battisputali Rd, Kathmandu	Mayju Bahal Bus Stop, Gangalal Way	March 6, 2021 8:30 a.m.	March 6, 2021 9:05 a.m.	1931	13

Figure 308 User Dashboard

## Your Travel Statistics

## Longest and Shortest Ride



Shortest Ride  
Chandan Marg, Kathmandu  
to  
Chandan Marg, Kathmandu  
1 meters



## Possible Cycling Trips

Thapathali Bus Stand, Thapathali Road  
to  
Kupondole Bus Stop, Lalitpur  
848 meters

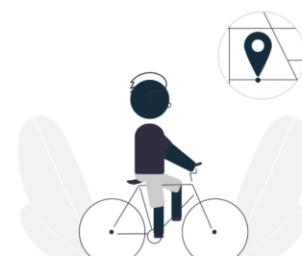


Figure 309 User Stat Board (1)

## Possible Walking Trips



< Click right or left arrow to view your trips! >

## Your Carbon Emissions

Hey Pranjali, with 51.83 Km that you have travelled,  
you have contributed in 3Kg of CO2 emission.  
Planting a single tree could offset this amount of emission.  
For a better environment, consider planting a tree!



© Card Pay | 2021

Figure 310 User Stat Board (2)

Your current account status is: Account Verified

### Update Profile

First Name  
Pranjali

Last Name  
Test2

Email  
testpranjali@gmail.com

Phone  
+9779861367460

Change Profile Picture

[Change password here...](#)

### Credentials

Upload the front and the back side of a government provided credential. To apply for discount, upload both sides of school/university provided ID.



Figure 311 User Profile

### Credentials

Upload the front and the back side of a government provided credential. To apply for discount, upload both sides of school/university provided ID.



© Card Pay | 2021

Figure 312 User Credentials

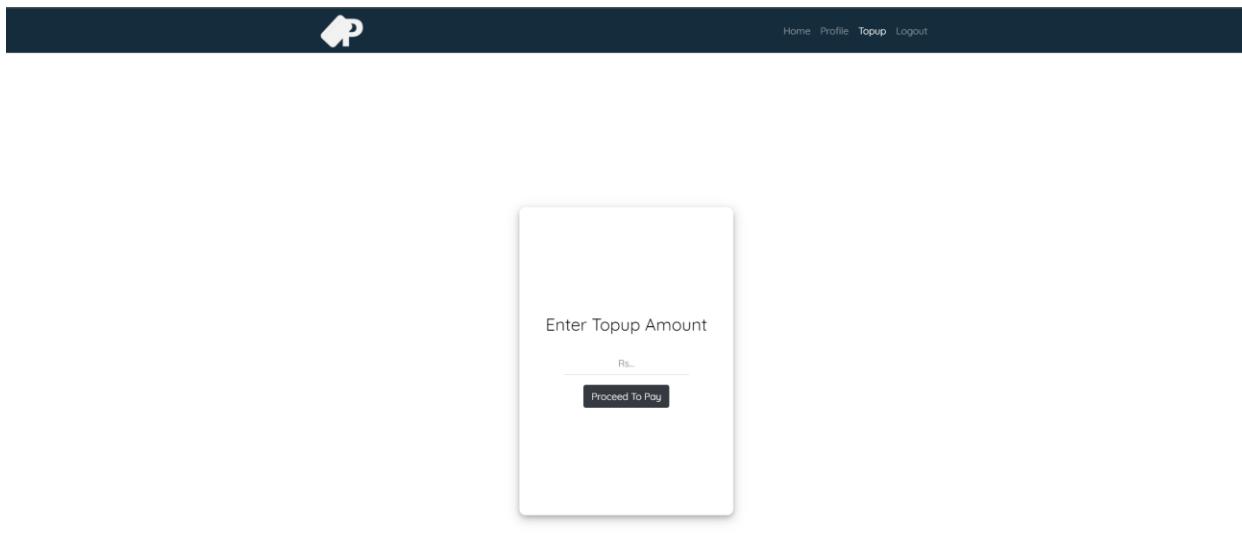


Figure 313 User Top-up

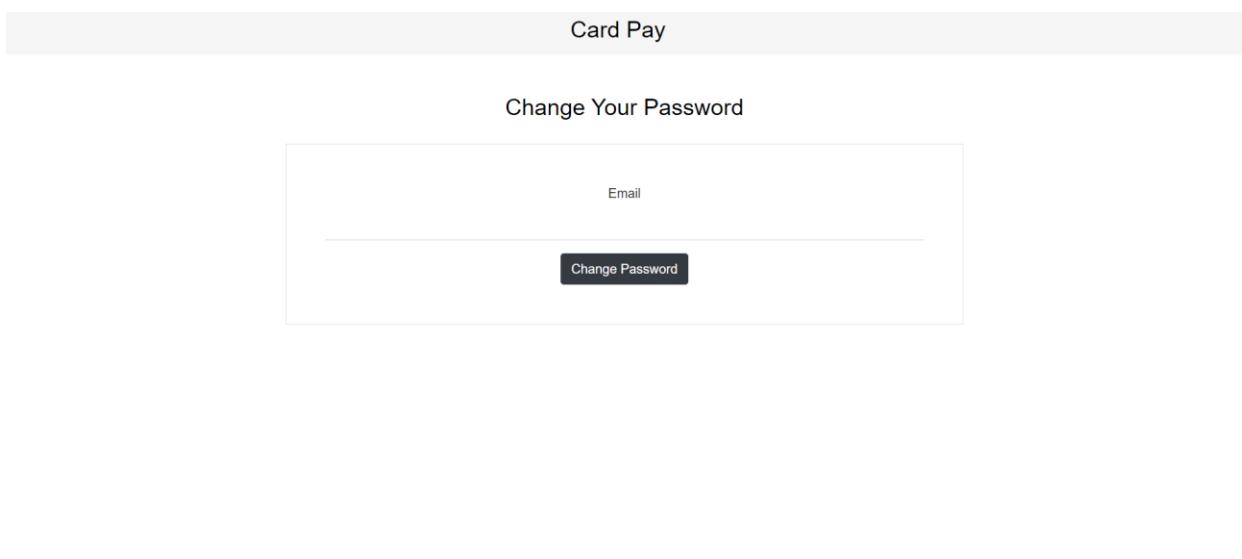


Figure 314 Reset Password Screen

## Card Pay

### Reset Link Sent

Check your entered email. You will receive an email if it is registered with us.

[Home](#)

© Contactless Pay | 2021

*Figure 315 Reset Link Sent Screen*

The screenshot shows a mobile application interface for 'Card Pay'. At the top, there is a navigation bar with a back arrow icon and the text 'Card Pay'. Below the navigation bar, the main content area has a light gray background. The title 'Reset Your Password' is centered at the top of the content area. Below the title, there are two input fields: 'New password' and 'New password confirmation', both represented by horizontal lines with placeholder text. At the bottom of the content area is a dark blue rectangular button labeled 'Change Password' in white text.

*Figure 316 Password Change Screen*

Card Pay

## Password Changed!

Your password has been changed successfully, you can now login.

Login

© Contactless Pay | 2021

*Figure 317 Password Changed Screen*

### 7.4.1.2 Admin

The screenshot shows the Admin Dashboard with the following components:

- Header:** A dark header bar with a logo on the left and navigation links "Home", "Invalid Trips", and "Logout" on the right.
- Summary Metrics:** Three cards at the top: "Accounts To Verify" (0), "Foul Trips" (5), and "Users Last 30 Days" (2).
- User Details:** A table titled "User Details" with columns: First Name, Last Name, Phone, and Action. The table lists six users with their respective details and "View" links.

Figure 318 Admin Dashboard

### Search Customer

Phone	Email	Status	Action
-----	-----	-----	Search

First Name	Last Name	Phone	Email	Status	Action
Bibek	Maharjan	+9779841877861	bibek.maharjan@islingtoncollege.edu.np	Account Verified	<a href="#">View</a>
prithivi	maharjan	+9779860317206	prithivi.maharjan@islingtoncollege.edu.np	Account Unverified	<a href="#">View</a>
Sodip	Thapa	+9779816809696	sodip99@gmail.com	Account Verified	<a href="#">View</a>
Sadikshya	Luitel	+9779860236211	sadikshyaluitel16@gmail.com	Account Verified	<a href="#">View</a>
Pranjali	Test2	+9779861367460	testpranjal@gmail.com	Account Verified	<a href="#">View</a>
test	1	+9779861234567	test@gmail.com	Needs Resubmission	<a href="#">View</a>
Prajna	Subedi	+9779860479271	itsmeprajna.subedi@gmail.com	Account Unverified	<a href="#">View</a>

© Card Pay | 2021

Figure 319 Admin Search User UI

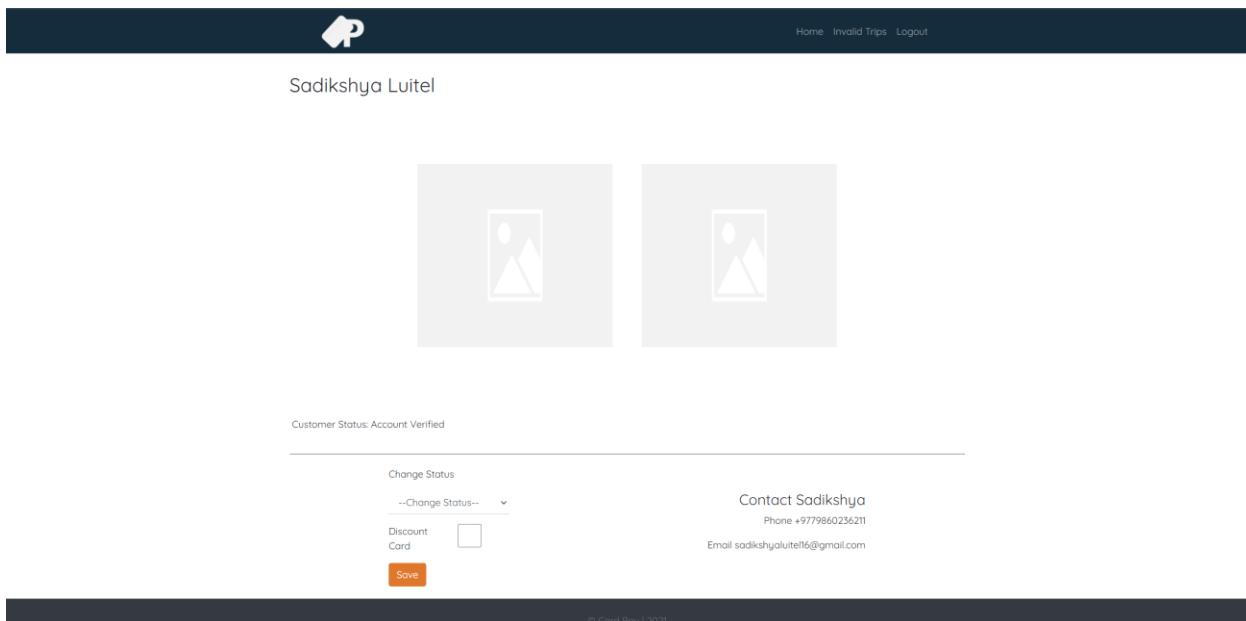


Figure 320 Admin View User

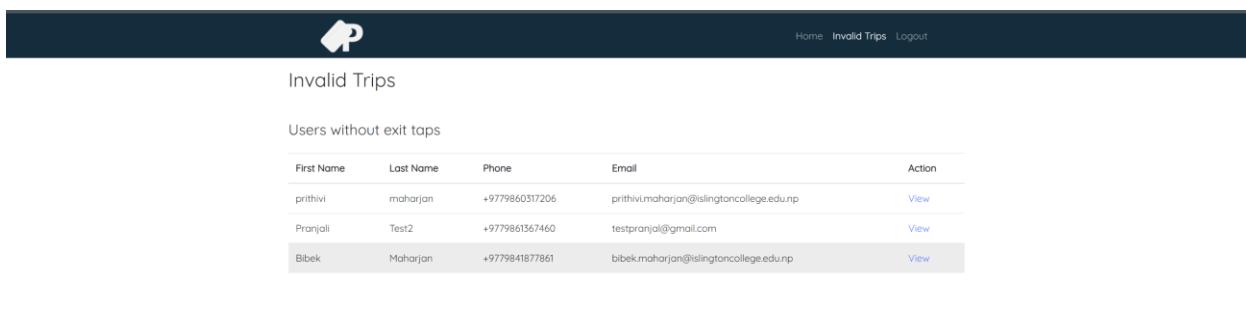


Figure 321 Admin View Invalid Trips Users

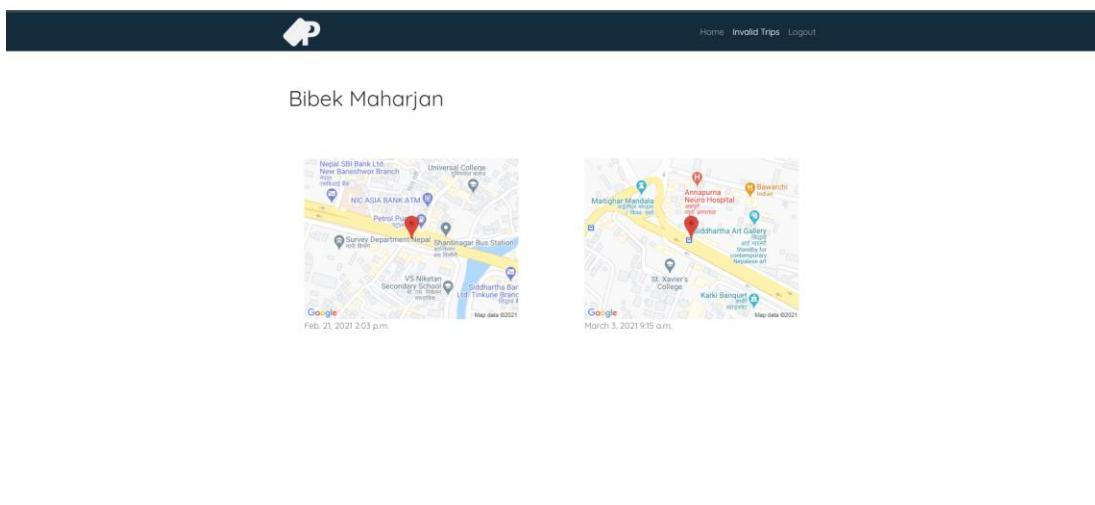


Figure 322 Admin View Invalid Trip

### 7.4.2 Desktop Application

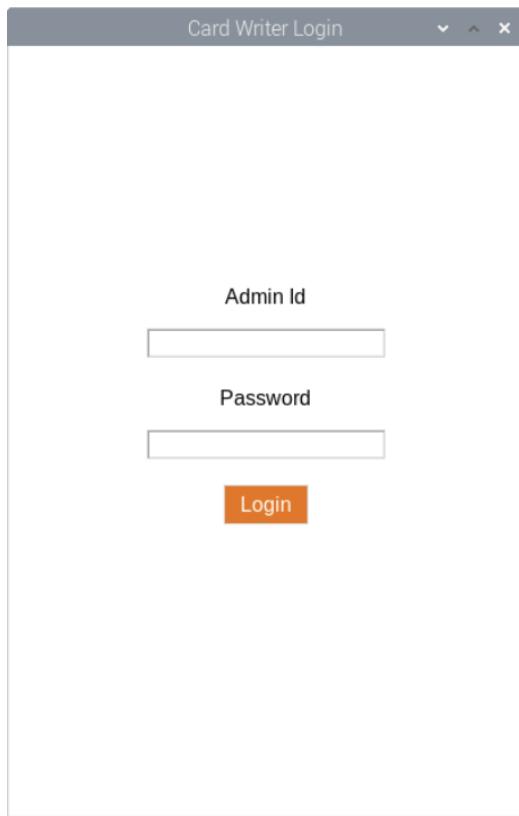


Figure 323 Desktop Login Screen

The screenshot shows a desktop application window titled "Card Writer". The window has a toolbar with "Logout" and "Deploy" buttons. The main area is titled "Write And Test Cards" and contains a table with the following data:

Frist Name	Last Name	Phone
Sadikshya	Luitel	+9779860236211
Udaya	Panday	+9779840067354
Sodip	Thapa	+9779816809696
Prajna	Subedi	+9779860479271
Sachhyam	Kaji Shakya	+9779845763331
Report	Test	+9779861367410

Below the table are two buttons: a large orange "Write" button and a smaller dark grey "Test" button.

Figure 324 Desktop Dashboard

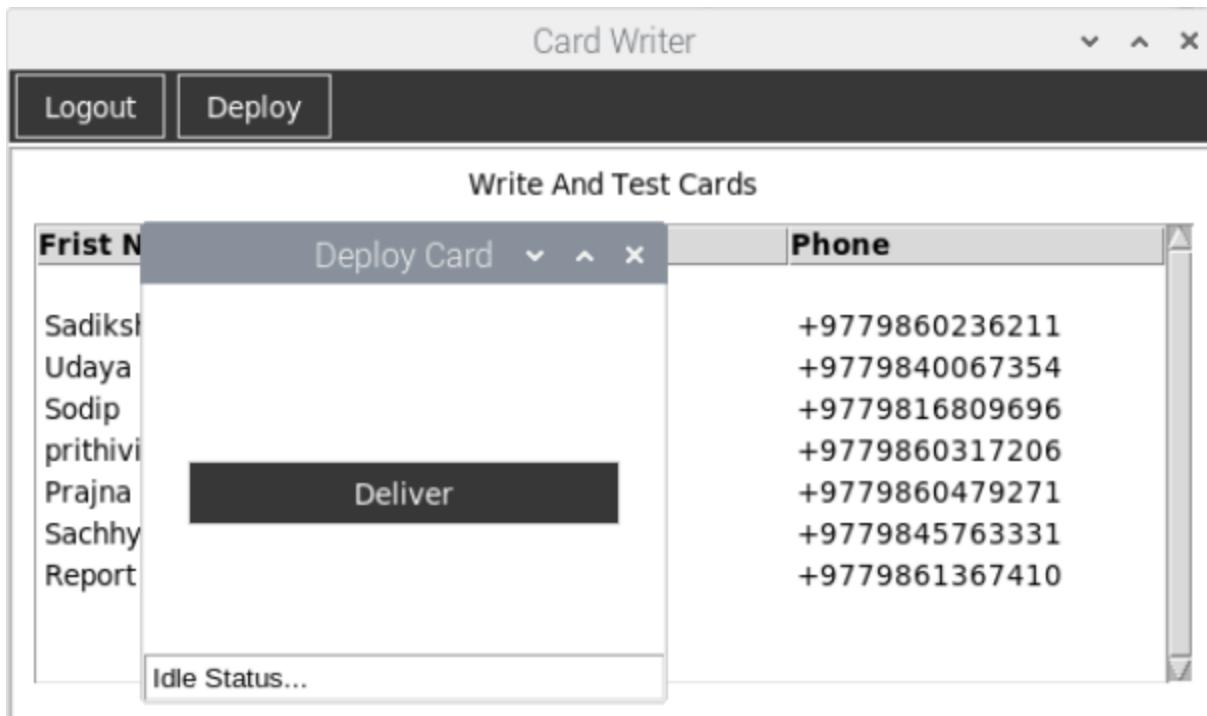


Figure 325 Desktop Deploy Card

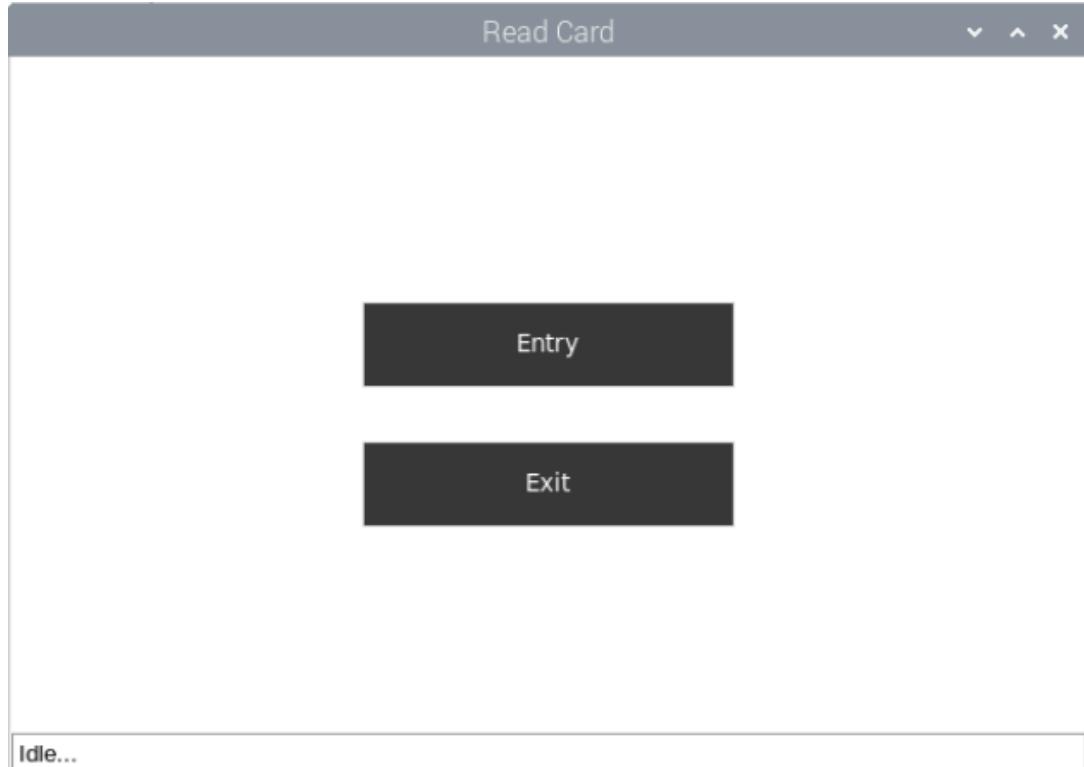


Figure 326 Desktop Read Card

### 7.4.3 Mobile Application

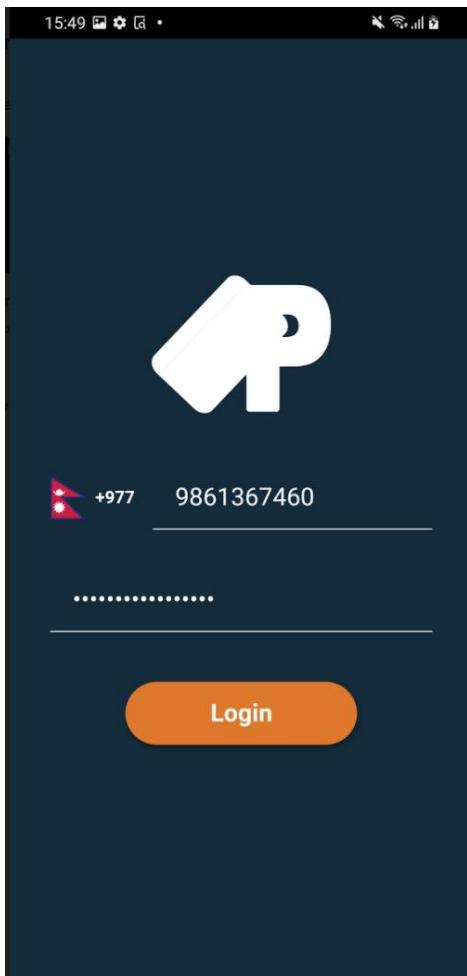


Figure 327 Mobile Login

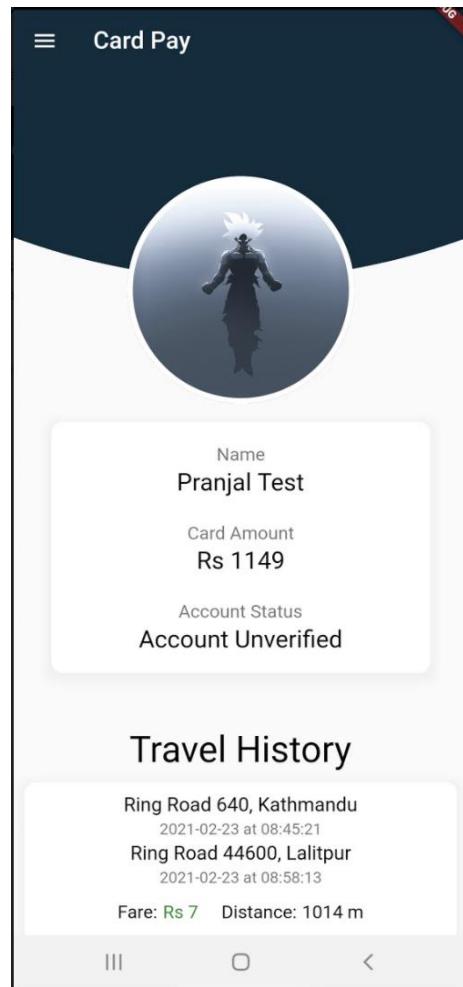


Figure 328 Mobile Dashboard

## 7.5 Appendix F: User Feedback

### 7.5.1 User Feedback Form

Your Name \*

Short-answer text

Your Age \*

Short-answer text

Under regular circumstances, how would you describe yourself as a commuter? \*

Regular Commuter  
 Occasional Commuter  
 Don't commute unless necessary.

Overall experience of using the system. \*

1      2      3      4      5

Rate the sign up and verification process. \*

1      2      3      4      5

Figure 329 User Feedback Form (1)

Rate the "Pay using Card Feature". \*

1      2      3      4      5

How likely do you think this system could be implemented in real vehicles? \*

1      2      3      4      5

Not Likely           Very Likely

Would one to one replication of website features on mobile application be useful? \*

Yes  
 No

Do you think implementing this system would encourage travel using public transportation? \*

Yes  
 No  
 Maybe

Figure 330 User Feedback Form (2)

Do you think a bus staff (conductor) would be required for people using this system to pay for their fares? \*

- Yes
- No
- Maybe

Suggest some features you would want in the over all system.

Long-answer text

---

*Figure 331 User Feedback Form (3)*

### 7.5.2 Sample of Filled User Feedback Forms

Your Name *	Sodip Bikram Thapa			
Your Age *	22			
Under regular circumstances, how would you describe yourself as a commuter? *				
<input checked="" type="radio"/> Regular Commuter <input type="radio"/> Occasional Commuter <input type="radio"/> Don't commute unless necessary.				
Overall experience of using the system. *				
1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Rate the sign up and verification process. *				
1	2	3	4	5
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>

Figure 332 User Feedback Sample (1)

Rate the "Pay using Card Feature". \*

1      2      3      4      5

How likely do you think this system could be implemented in real vehicles? \*

1      2      3      4      5

Not Likely      Very Likely

Would one to one replication of website features on mobile application be useful? \*

Yes  
 No

Do you think implementing this system would encourage travel using public transportation? \*

Yes  
 No  
 Maybe

Figure 333 User Feedback Sample (2)

Do you think implementing this system would encourage travel using public transportation? \*

- Yes
- No
- Maybe

Do you think a bus staff (conductor) would be required for people using this system to pay for their fares? \*

- Yes
- No
- Maybe

Suggest some features you would want in the over all system.

would be nice to have 'View Password' option as a feature

Submitted 15/04/2021, 17:07

Figure 334 User Feedback Sample (3)

### 7.5.3 User Feedback Result

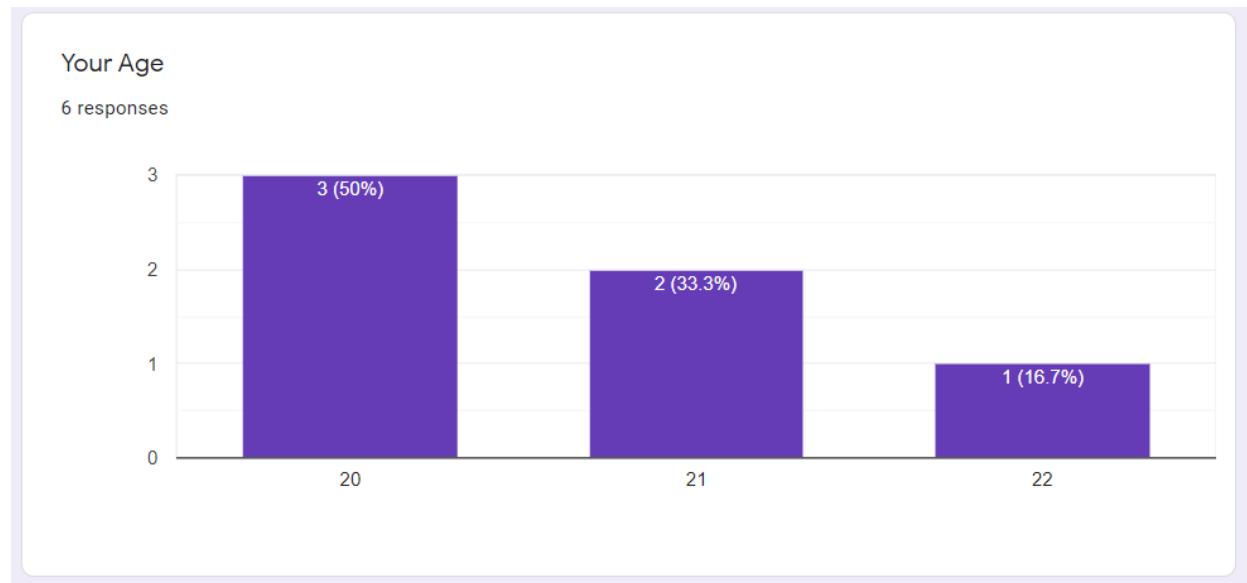


Figure 335 Average Age Group for Sample Space

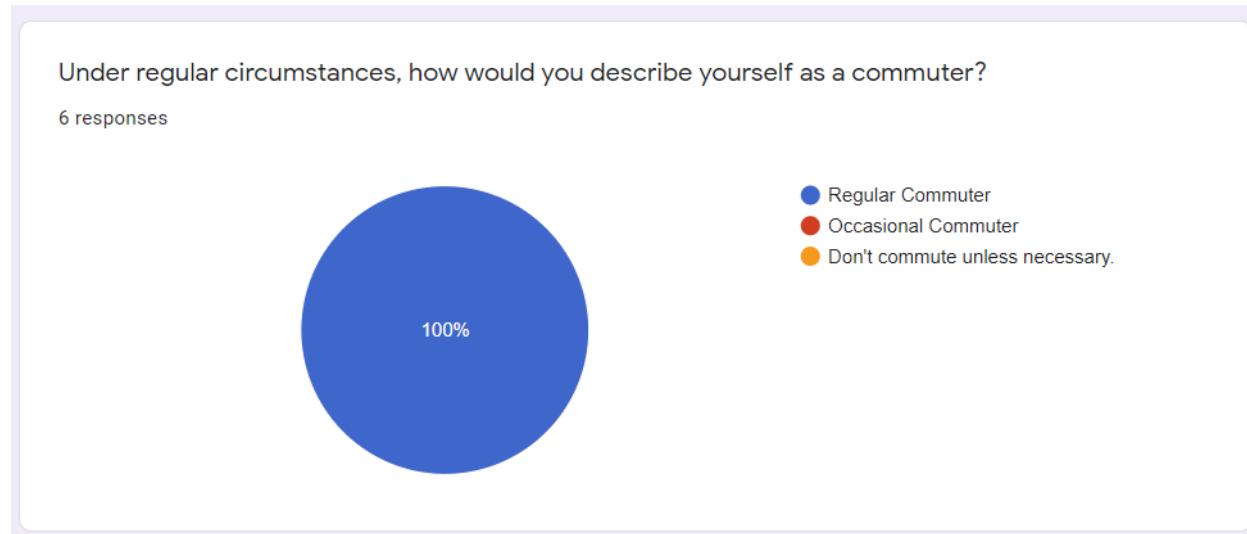


Figure 336 Percentage of Commuters

Overall experience of using the system.

6 responses

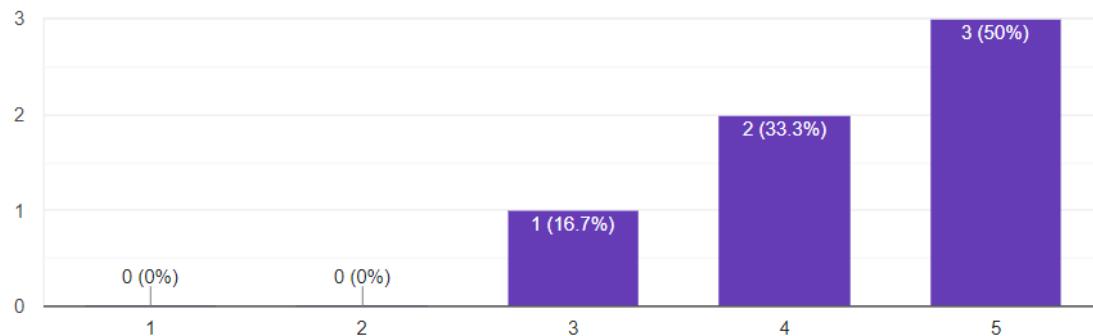


Figure 337 System User Experience Feedback

Rate the sign up and verification process.

6 responses

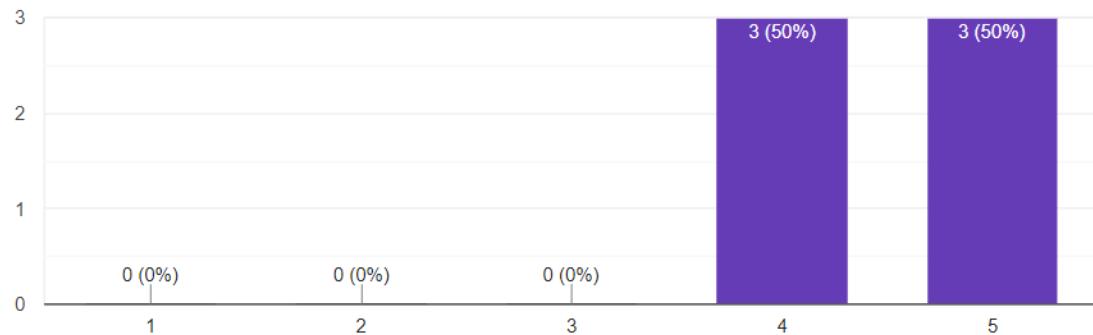


Figure 338 User Signup Feedback

Rate the "Pay using Card Feature".

6 responses

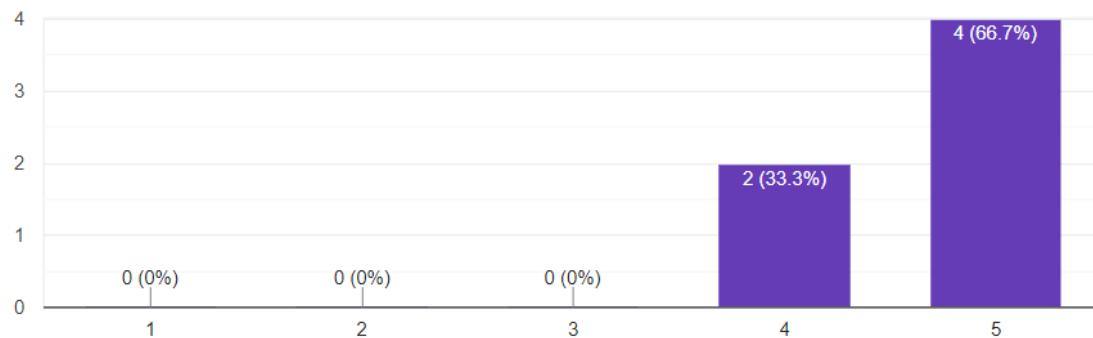


Figure 339 Read Card Feedback

How likely do you think this system could be implemented in real vehicles?

6 responses

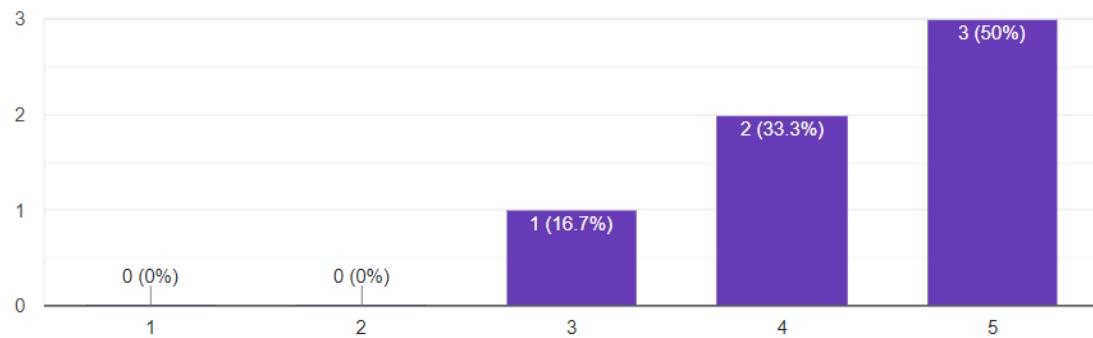


Figure 340 Implementation Expectation Feedback

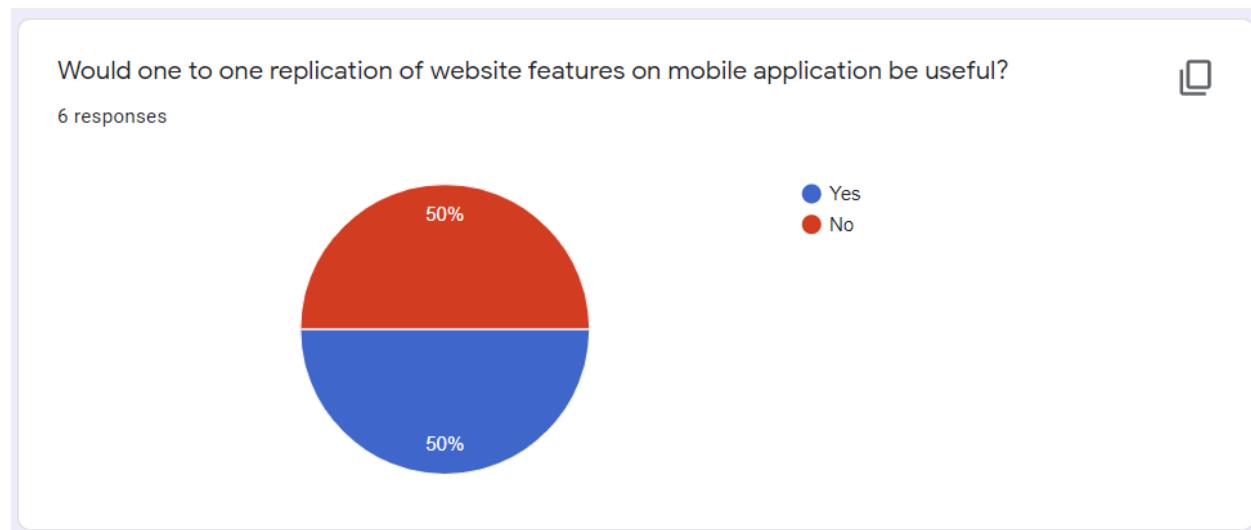


Figure 341 Mobile Feature Replication Feedback

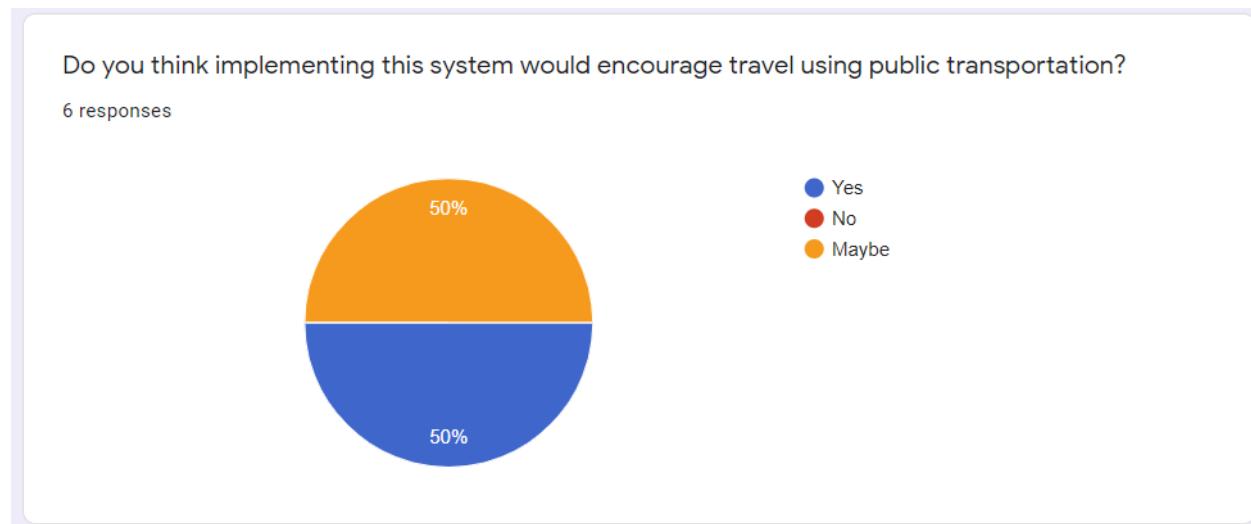


Figure 342 Quality of Transportation Feedback

Do you think a bus staff (conductor) would be required for people using this system to pay for their fares?

6 responses

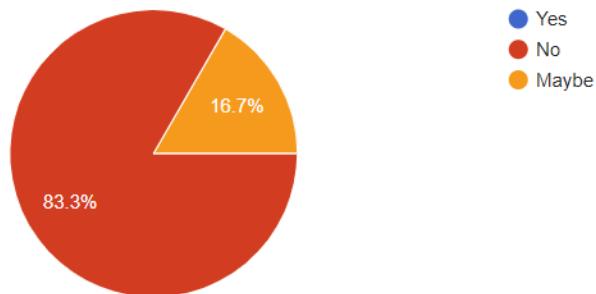


Figure 343 Buss Staff Replacement Feedback

Suggest some features you would want in the over all system.

2 responses

would be nice to have 'View Password' option as a feature

it would be helpful if their was an option to view my password

Figure 344 Extra Feature Request Feedback

## 7.6 Appendix G: Future Work

### 7.6.1 Readings for Future Work

#### 7.6.1.1 Firebase Cloud Messaging

Add Firebase to your JavaScript project

If you haven't already, [add Firebase to your JavaScript project](#).

If you are currently using FCM for web and want to upgrade to SDK 6.7.0 or later, you must enable the [FCM Registration API](#) for your project in the Google Cloud Console. When you enable the API, make sure you are logged in to Cloud Console with the same Google account you use for Firebase, and make sure to select the correct project. New projects adding the FCM SDK have this API enabled by default.

### Configure Web Credentials with FCM

The FCM Web interface uses Web credentials called "Voluntary Application Server Identification," or "VAPID" keys, to authorize send requests to supported web push services. To subscribe your app to push notifications, you need to associate a pair of keys with your Firebase project. You can either generate a new key pair or import your existing key pair through the Firebase Console.

#### Generate a new key pair

1. Open the [Cloud Messaging](#) tab of the Firebase console **Settings** pane and scroll to the **Web configuration** section.
2. In the **Web Push certificates** tab, click **Generate Key Pair**. The console displays a notice that the key pair was generated, and displays the public key string and date added.

*Figure 345 Firebase for Web Setup (Google, 2021)*

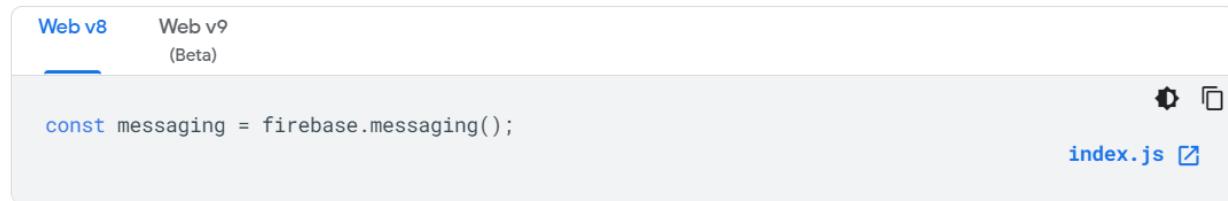
### Import an existing key pair

If you have an existing key pair you are already using with your web app, you can import it to FCM so that you can reach your existing web app instances through FCM APIs. To import keys, you must have owner-level access to the Firebase project. Import your existing public and private key in base64 URL safe encoded form:

1. Open the [Cloud Messaging](#) tab of the Firebase console **Settings** pane and scroll to the **Web configuration** section.
2. In the **Web Push certificates** tab, find and select the link text, "import an existing key pair."
3. In the **Import a key pair** dialog, provide your public and private keys in the corresponding fields and click **Import**.  
The console displays the public key string and date added.

For instructions on how to add the key to your app, see [Configure Web credentials in your app](#). For more information about the format of the keys and how to generate them, see [Application server keys](#).

### Retrieve a messaging object ↗



The screenshot shows the 'Web v8' tab selected in the Firebase Cloud Messaging settings. A modal dialog is open, prompting for the import of an existing key pair. The code snippet inside the dialog is:

```
const messaging = firebase.messaging();
```

At the bottom right of the dialog, there is a file icon labeled 'index.js' with a checkmark.

Figure 346 Receiving Firebase Messages on Web App (Google, 2021)

## fcm-django

---



Django app for Firebase Cloud Messaging. Used as an unified platform for sending push notifications to mobile devices & browsers (android / ios / chrome / firefox / ...).

### *FCMDevice model fields*

- *registration\_id* (required - is FCM token)
- *name* (optional)
- *active* (default: true)
- *user* (optional)
- *device\_id* (optional - can be used to uniquely identify devices)
- *type* ('android', 'web', 'ios')

### *Functionality:*

- all necessary migrations
- model admins for django admin
- admin actions for testing single and bulk notification sending
- automatic device pruning: devices to which notifications fail to send are marked as inactive
- devices marked as inactive will not be sent notifications
- Django rest framework viewsets

Figure 347 FCM Package for Django (Rojko, 2021)

### 7.6.1.2 EMV Card

#### What EMV means

EMV stands for “[Europay, MasterCard, and Visa](#).” (We know, not a particularly informative acronym). Here’s what EMV means in practice: a transaction between a chip-enabled credit card (as opposed to a magnetic-stripe-only card) and an EMV-enabled payment terminal or ATM. And here’s why you need to know about it: EMV will soon become the technological standard for credit card processing in the United States. ([It’s already the standard throughout most of the world](#)).

We’re switching over to EMV cards (aka chip cards) because they’re [leagues more secure than the magnetic-stripe cards we currently carry](#). EMV cards contain a tiny, dynamic computer chip that talks back and forth with the payments terminal to make sure you’re not a fraudster. The technology works. In other countries that have adopted EMV as the standard, counterfeit fraud has dramatically declined.

*Figure 348 Introduction to EMV (Square Up, 2016)*

Simultaneously with the U.S. move to EMV chip card payments, Near NFC technology is emerging as a useful accessory for consumer transactions. NFC is not a payment technology; it is a set of standards that enables proximity-based communication between consumer electronic devices such as mobile phones, tablets, and personal computers. NFC supports an extremely simple man-machine interface, facilitating its use for a number of functions, including mobile payment. NFC technology is compatible with the current contactless payment acceptance infrastructure—an NFC-compliant mobile device can communicate with a point-of-sale (POS) system that currently accepts contactless payment cards.

***EMV is a payments technology.***

***NFC is a communications technology that enables contactless EMV.***

NFC and EMV are companion technologies. NFC applies to how devices communicate; EMV applies to payments made with contact and contactless chip cards or with a mobile NFC device emulating a contactless chip card. Contactless payment transactions made using mobile NFC devices use the same infrastructure as contact and contactless EMV chip card transactions.

The white paper describes the current status of the U.S. payment issuance and acceptance infrastructure, identifies impacts, benefits and key considerations for contactless payment migration. Topics covered include:

- The current status of the EMV chip migration and predictions for contactless payments in the U.S., including the number of chip cards issued, the number of merchant locations accepting chip cards, and predictions for contactless cards over the next two years
- The intersection EMV contact, contactless and dual interface chip cards and mobile NFC devices for payment describing how chip cards are provisioned, how they're used in stores, how payment account credentials are provisioned into mobile NFC devices, and how mobile NFC devices are used for contactless payments
- Key considerations for contactless payment implementations, such as how to migrate from contactless magnetic stripe card acceptance to contactless EMV chip card acceptance, what types of transactions to accept, how contactless payment adoption is affected by mobile NFC device availability, and other usability factors

Figure 349 EMV Complementary with NFC (EMV Connection, 2015)

### 7.6.1.3 Asynchronous Programming (Tkinter)

#### Discussion

This recipe shows the easiest way of handling access to sockets, serial ports, and other asynchronous I/O ports while running a Tkinter-based GUI. Note that the recipe's principles generalize to other GUI toolkits, since most of them make it preferable to access the GUI itself from a single thread, and all offer a toolkit-dependent way to set up periodic polling as this recipe does.

Tkinter, like most other GUIs, is best used with all graphic commands in a single thread. On the other hand, it's far more efficient to make I/O channels block, then wait for something to happen, rather than using nonblocking I/O and having to poll at regular intervals. The latter approach may not even be available in some cases, since not all data sources support nonblocking I/O. Therefore, for generality as well as for efficiency, we should handle I/O with a separate thread, or more than one. The I/O threads can communicate in a safe way with the main, GUI-handling thread through one or more Queues. In this recipe, the GUI thread still has to do some polling (on the Queues), to check if something in the Queue needs to be processed. Other architectures are possible, but they are much more complex than the one in this recipe. My advice is to start with this recipe, which will handle your needs over 90% of the time, and explore the much more complex alternatives only if it turns out that this approach cannot meet your performance requirements.

*Figure 350 Async Programming Introduction for Tkinter (Hallen, 2018)*

```
class ThreadedClient:  
    """  
    Launch the main part of the GUI and the worker thread. periodicCall and  
    endApplication could reside in the GUI part, but putting them here  
    means that you have all the thread controls in a single place.  
    """  
  
    def __init__(self, master):  
        """  
        Start the GUI and the asynchronous threads. We are in the main  
        (original) thread of the application, which will later be used by  
        the GUI as well. We spawn a new thread for the worker (I/O).  
        """  
  
        self.master = master  
  
        # Create the queue  
        self.queue = Queue.Queue()  
  
        # Set up the GUI part  
        self.gui = GuiPart(master, self.queue, self.endApplication)  
  
        # Set up the thread to do asynchronous I/O  
        # More threads can also be created and used, if necessary  
        self.running = 1  
        self.thread1 = threading.Thread(target=self.workerThread1)  
        self.thread1.start()  
  
        # Start the periodic call in the GUI to check if the queue contains  
        # anything  
        self.periodicCall()
```

Figure 351 Threading in Python (Hallen, 2018)

```
def workerThread1(self):
    """
    This is where we handle the asynchronous I/O. For example, it may be
    a 'select( )'. One important thing to remember is that the thread has
    to yield control pretty regularly, by select or otherwise.
    """
    while self.running:
        # To simulate asynchronous I/O, we create a random number at
        # random intervals. Replace the following two lines with the real
        # thing.
        time.sleep(rand.random( ) * 1.5)
        msg = rand.random( )
        self.queue.put(msg)

def endApplication(self):
    self.running = 0
```

Figure 352 Ending Functions in Async (Hallen, 2018)

## 7.7 Similar Projects Continued

### 7.7.1 Go Card Queensland



Figure 353 Tiers of Go Card

Go card is a smart card payment service currently deployed in TransLink public transport network operating in South East Queensland, Australia. It focuses on contactless payment service for transportation where users need to tap for payment before and after boarding a vehicle. Go Card has been available in retail stores throughout Brisbane from February 2008 (TransLink, 2020).

**Platform:** NFC Cards, Web Application, Mobile Application (iOS, android)

#### Features:

- Web Application to setup payment
- Tap to Pay
- Top-ups for card

### 7.7.2 Bicing Barcelona



Figure 354 Bicing Card in Operation

Bicing is a bicycle sharing service in Barcelona, Spain that focuses on contactless payment with travel statistics focusing on environmental friendliness. Statistics provided by Bicing focuses on level of happiness and carbon dioxide emissions similar to my project. Bicing has been in operation since 2007 with primary aim to encourage commuters to travel small distance (Vitali, 2019).

**Platform:** NFC Cards, Web Application, Mobile Application (iOS, android)

#### Features:

- Mobile Application for statistics
- Tap to Pay
- Top-ups for card