SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY


SE3070 – Case Studies in Software Engineering – 2021

# Critical Reflection Report for Ticketing System for Public Transport Network

## 2021S2_REG_WE_02

| **Student Registration Number** | **Name** |
|---|---|
| IT19004778 | W.G.M.S.Wickramathna |
| IT19006994 | K.H.K.L.De Silva |
| IT19111766 | U.L.V.M.Lekamalage |

## Contents

## Source Code URL

https://mysliit-my.sharepoint.com/:f:/g/personal/it19006994_my_sliit_lk/EkRBaCIQ0W5KjWZSKZi6oaABrt6kLWhd54x_Txov_jzgEQ?e=zruVaO

## GitHub URL

https://github.com/SE3070/implementation-walkthrough-2021s2_reg_we_02.git

*Please refer to branch FINAL_PRODUCT to see the source code for the Final Application.

# 1. Assumptions

- Issuing a new Smart Card is not a function that can process through the system as the particular user has to visit the relevant authority to obtain their card.

- Our complete implementation covers the three main business scenarios as given in the Design Document. They are as follows:
  1. Payment Management (Tap Smart Card)
  2. Smart Card Recharge
  3. Timetable Management

- As we do not cover the Hardware aspect of the application, scanning of the QR Code part has been eliminated and the Login function is handled by proving the Smart Card ID for validation.

## 2. Mistakes detected and suggested solutions to Design Changes

1. Absence of a list of Assumptions made -
   To enhance the understanding of the Development Team, Design documentation should explicitly state the assumptions made during the design process. However, such assumptions were not specified in the given document.

2. The appropriate justification was not provided for why certain business scenarios were chosen and why other key components were excluded.

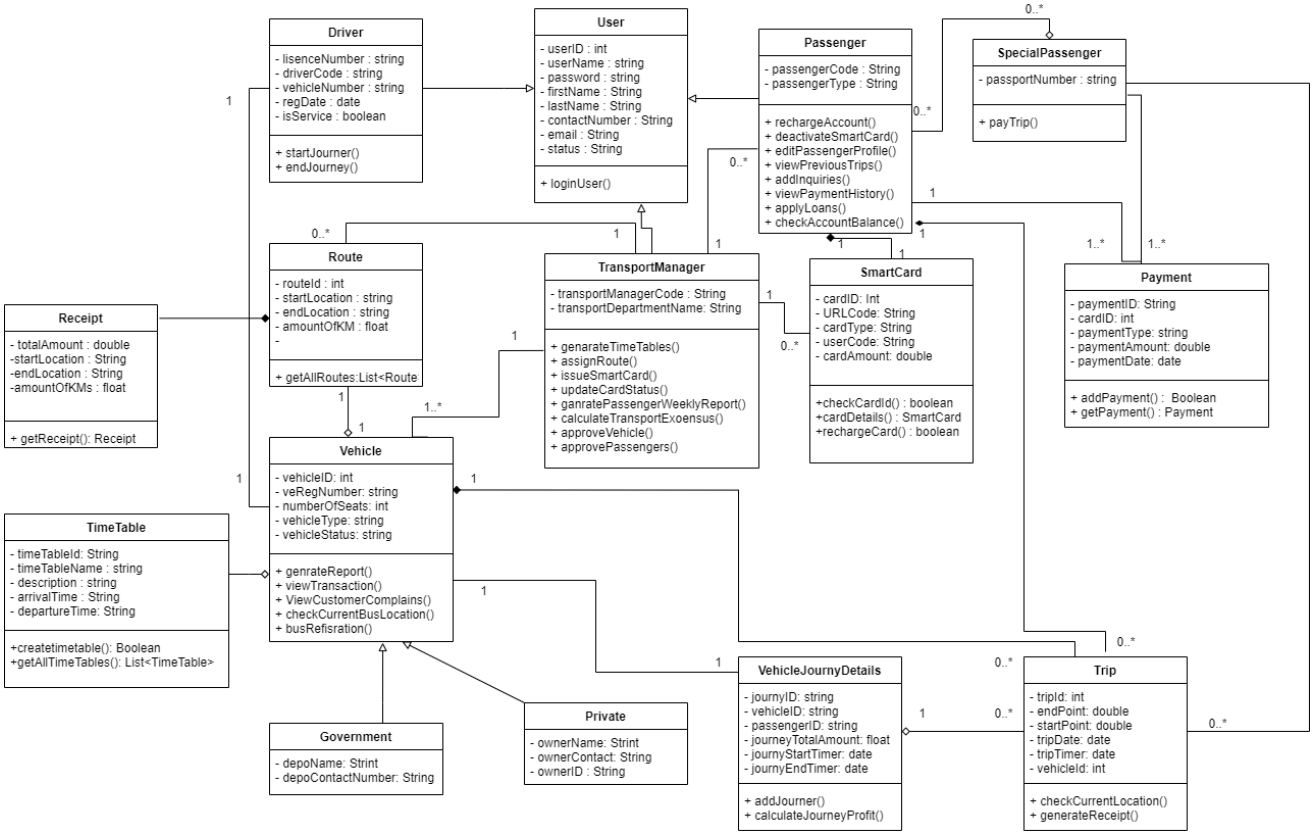
3. Some important Business Scenarios like Viewing Past Trips has not been included in the Design Document.

4. In High Fidelity Wireframes, some important wireframes were not included. (Refer to topic 6)

5. In Class Diagram some significant classes were not included while some unnecessary classes and relationships were included. (Refer to topic 3)

6. They have not shown the connection between the Time Table Generation and the rest of the functions of the system. That function seems to be isolated.

# 3. Modified Class Diagram



## Modifications made in the Class Diagram

- Return types for the methods have not been given in the class diagram.

- The location attribute is removed from the Payment class as it is an irrelevant attribute for that class.

- Route class and Vehicle class has an Aggregation relationship because each Vehicle must have a Route. Route can exists without the Vehicle.

- A new Timetable class was added to the class diagram because it is a mandatory requirement to implement the timetable generation function. Timetable class was not given in the initial design document.

- Timetable class and Vehicle class has an Aggregation relationship because each Vehicle must have a Timetable. The timetable can exist without the Vehicle.

- A New Receipt class is added to generate the receipt and get receipt details. There is a composition relationship between the Receipt and the Route. Receipt cannot exist without the Route.

- SmartCardReader interface was removed from the class diagram as it was unambiguous.

- Class ISP (Internet Service Provider) does not have a direct connection with the system, therefore it is removed from the Class Diagram.

## 4. Modified Use Case Scenarios

Modified By: IT19006994

| | | |
|---|---|---|
| Scenario No | Use Case 1 | |
| Scenario Name | Card validation from Reader | |
| Summary | Passengers have to tap the smart card to the reader in the bus and get the valid journey, generate the Receipt and make the Payment. | |
| Preconditions | Passenger has sufficient balance in smart card | |
| Postconditions | Passengers have successfully got paid for the journey. | |
| Primary Actor(s) | Passenger | |
| Trigger | Passengers have valid smart cards and have to tap the smart card to the reader | |
| Main Scenario | Step | Action |
| | 01 | The system will check if the card is valid or not |
| | 02 | The system directs the user to the Route page where they have to enter the Starting Location and the Destination |
| | 03 | When clicking the “Get Receipt” button user will navigate to the Receipt page |
| | 04 | User Enter the Route ID and click the “Get Receipt” button |
| | 05 | The system displays the total fare and generates the Receipt |
| | 06 | User clicks the Pay Now option and system displays the payment successful message |

| | | |
|---|---|---|
| | 07 | User will be navigated to the Home Page again |
| | 08 | Device Print the Ticket for User |
| | 09 | User logs out of the system |
| Extensions | 01a | If the Smart card is not valid, the system will display an error message |
| | 02a | If the Start Location and Destination fields are empty the system will prompt an error. |
| | 04a | If the Route ID field is empty the system will prompt an error |
| | 04b | If the Route ID is invalid, the system will prompt an error |
| | 06a | If the SmartCard does not have a sufficient amount to do the payment, the system will prompt an error |

Step 02 – In the Design Document, it was given that the System automatically captures the destination. Instead under step 02 user has to enter both Checked In and Checked Out Location.

Step 04 – As the QR Code scanning option is disabled user has to enter the Bus Route to generate their report and proceed with the Payment.

- Changes made are given in Red

Modified By: IT19111766

| | | |
|---|---|---|
| Scenario No | Use Case 2 | |
| Scenario Name | Recharge the smart card | |
| Summary | Due to insufficient Account balance passengers have to recharge their smartcard's account | |
| Preconditions | The passenger has a valid smart card | |
| Postconditions | Passengers have successfully recharged their smartcard's account | |
| Primary Actor(s) | Passenger | |
| Trigger | Users have to visit the transport service provider's Mobile App | |
| Main Scenario | Step | Action |
| | 01 | The system displays the login window ask smart card number for the login |
| | 02 | Users have to enter the smart card number to log in |
| | 03 | The system will display the home page |
| | 04 | Users have to select the recharge smart card option |
| | 05 | The system will display the recharging interface and ask for details for recharge (amount / smart card number/payment type) |
| | 06 | Users have to fill in the Details |
| | 07 | The system will display the payment method to do the payment (Paypal, VISA, eZ cash) |

| | | |
|---|---|---|
| | 08 | Users have to select the payment method and click "Recharge". fill required details based on the selection |
| | 09 | User will be navigated to the payment page and the user have to fill required details based on the selection |
| | 10 | The user clicks the 'Recharge' button. |
| | 11 | The system will check the payment is valid or not |
| | 12 | The system will display payment successful message and navigate the user to Home Page |
| | 13 | Users have to logout |
| Extensions | 02a | If the Smart card is not valid, the system will display an error message |
| | 07a | When the Papal option is selected user has to enter the PayPal Email |
| | 07b | When the VISA option is selected user must enter the VISA, Card Details |
| | 07c | When the eZ cash option is selected the User must enter the Owner's details and Mobile Number |
| | 11a | If the payment details are the incorrect system will prompt an error |
| | 11b | User has to re-enter payment details |

- Changes made are given in Red

Modified By: IT19004778

| | | |
|---|---|---|
| Scenario No | Use Case 3 | |
| Scenario Name | Manage Timetables | |
| Summary | Check the details and create timetables according to the report | |
| Preconditions | User must have valid login credentials | |
| Postconditions | Users can successfully manage the timetables for bus | |
| Primary Actor(s) | Admin | |
| Trigger | The user had logged to the transport service provider's website Backend as an Admin | |
| Main Scenario | Step | Action |
| | 01 | The system displays the admin dashboard |
| | 02 | Users have to select the manage timetables button |
| | 03 | The system will display the bus routes for manage timetables |
| | 04 | Users have to select the relevant bus route |
| | 05 | The system will display the list of details of Bus Routes |
| | 06 | Users have to analyse the data and select create time table button |
| | 07 | The system will display the form for fill |
| | 08 | Users have to fill in the details and click submit button |
| | 09 | The system will check the overlapping status |

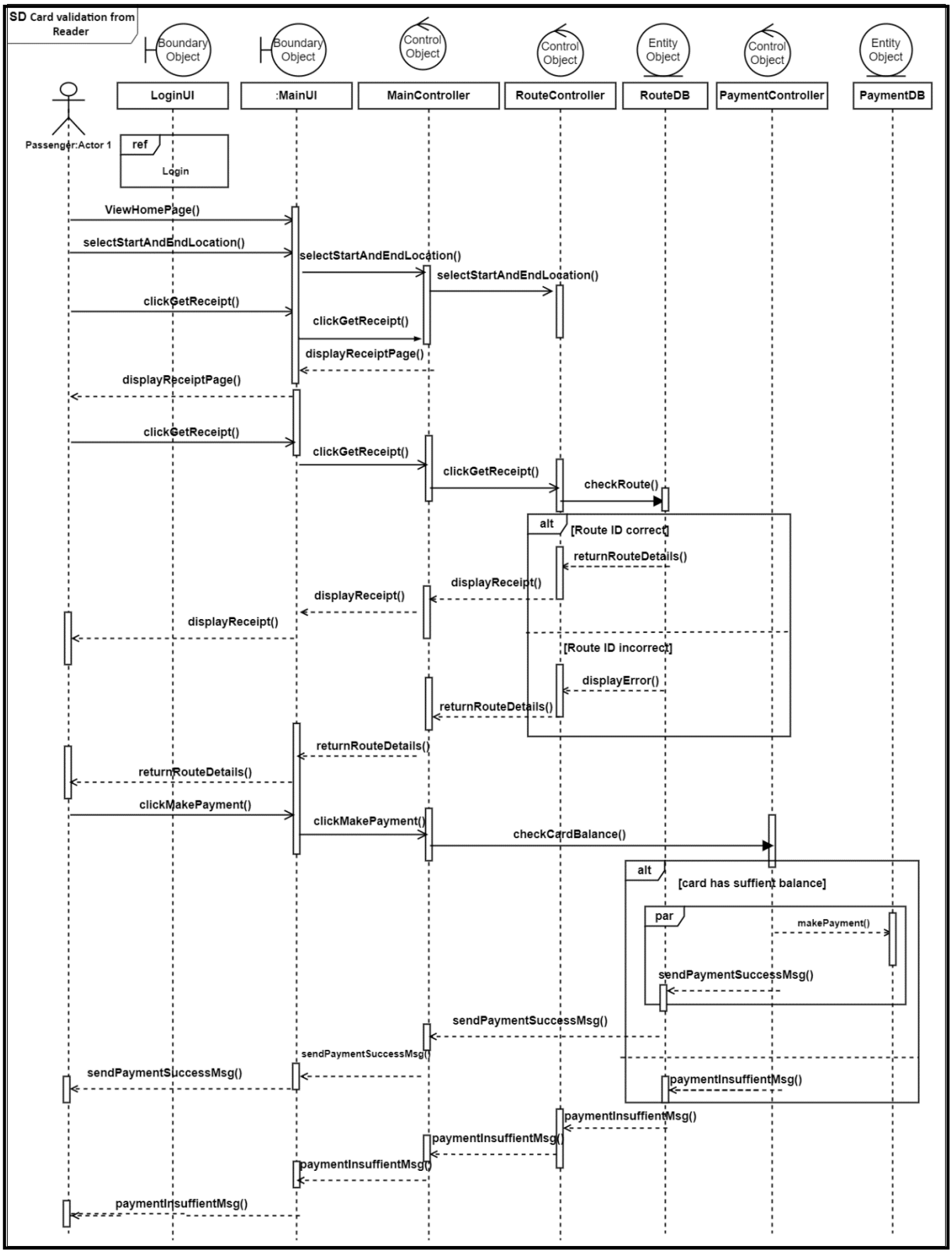
| | | |
|---|---|---|
| | 10 | The system will display the successful message to the user |
| | 11 | Users have to logout |
| Extensions | 08a | If the added details are incorrect it will prompt an error |
| | 09a | When a Bus already has a Timetable, it will prompt an error when adding a new Timetable to that. |
| | 09b | If a Bus already has a Timetable can update the existing one. |

- Changes made are given in Red

# 5. Modified Sequence Diagrams

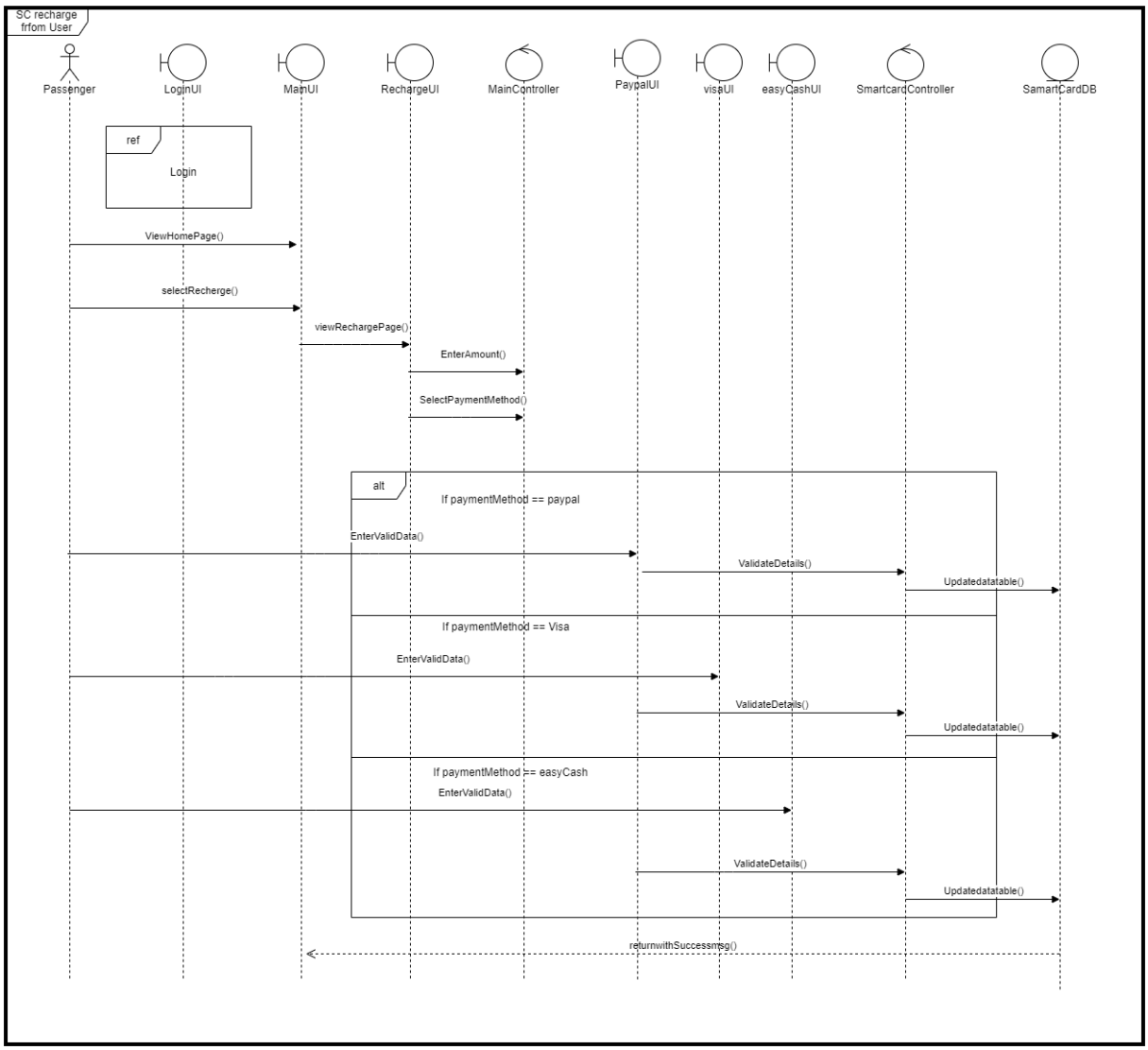
Modified By: IT19006994

**Modified Sequence Diagram for Card Validation from Reader**



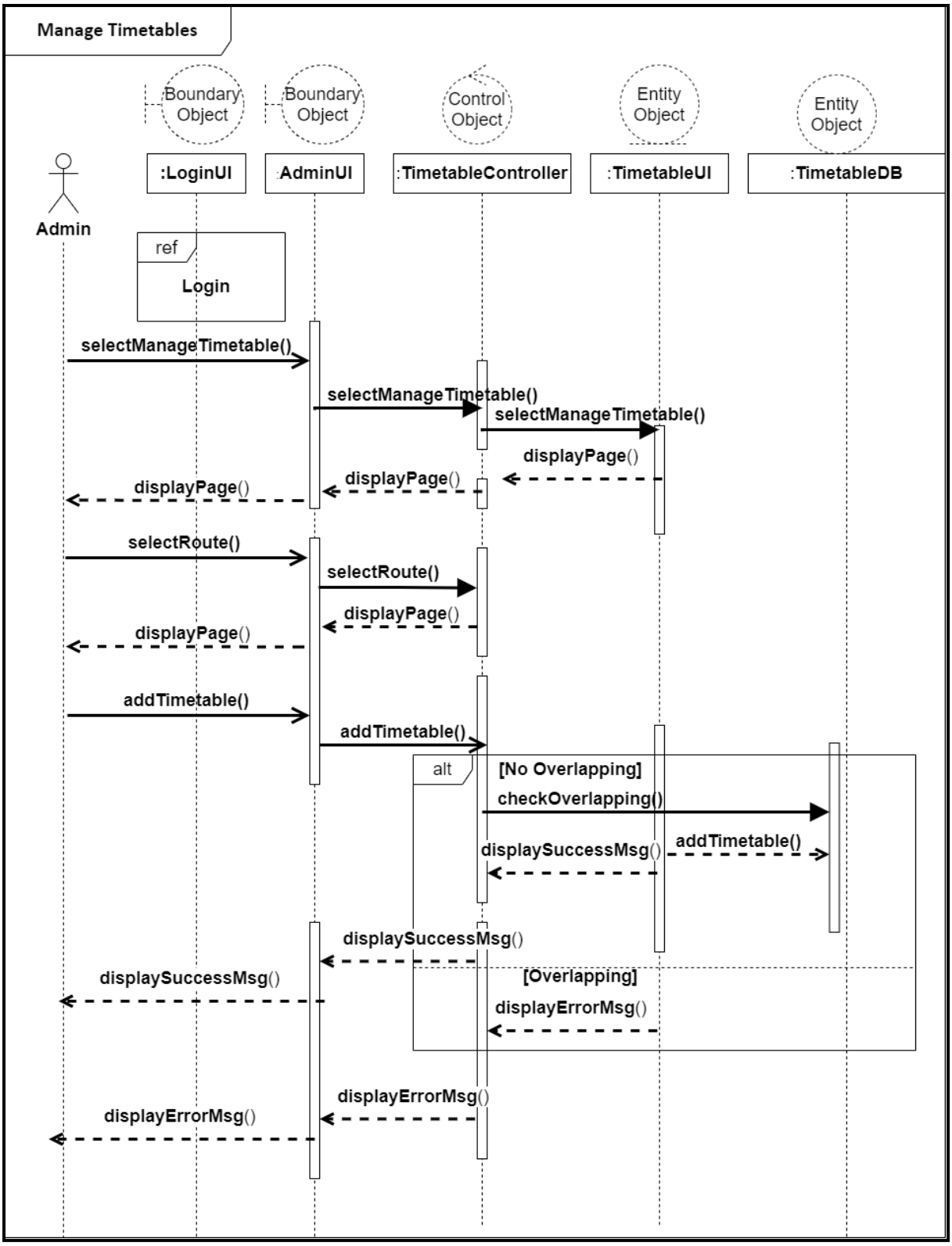
Modified By: IT19111766

## Modified Sequence Diagram for Recharge the smart card



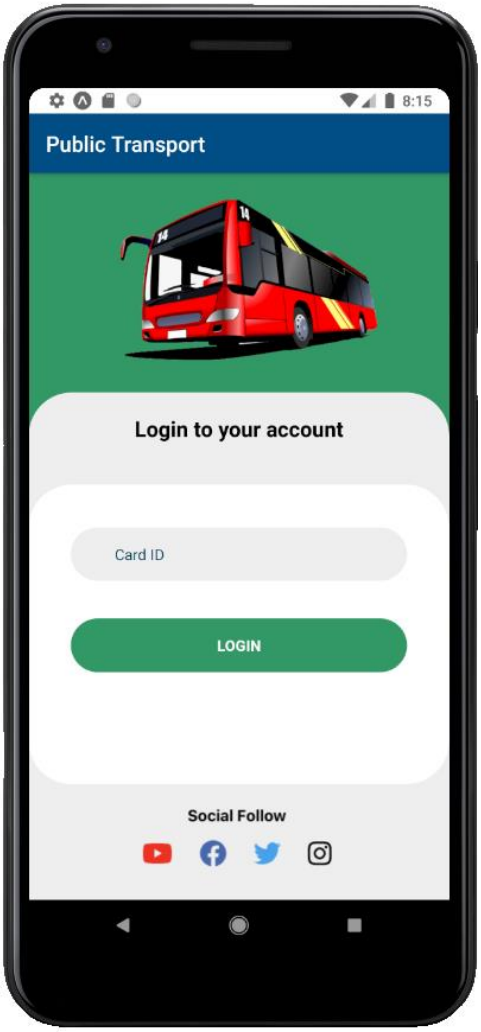
Modified By: IT19004778

**Modified Sequence Diagram for Manage Timetables**

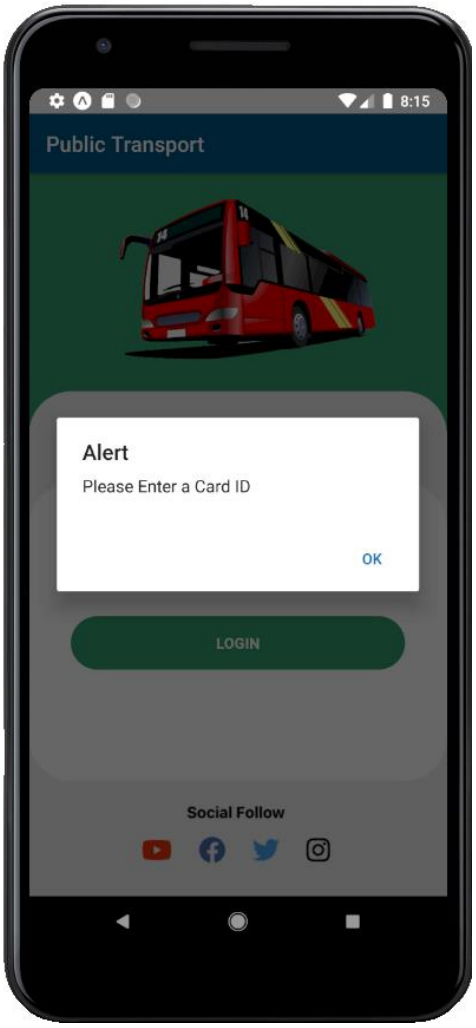


## 6. Modified High Fidelity Wireframes

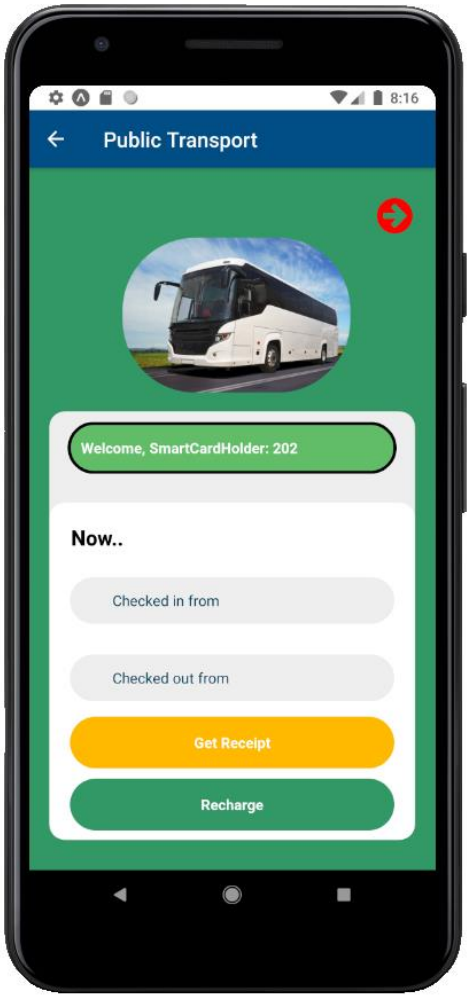
### High Fidelity Wireframes for Mobile Application

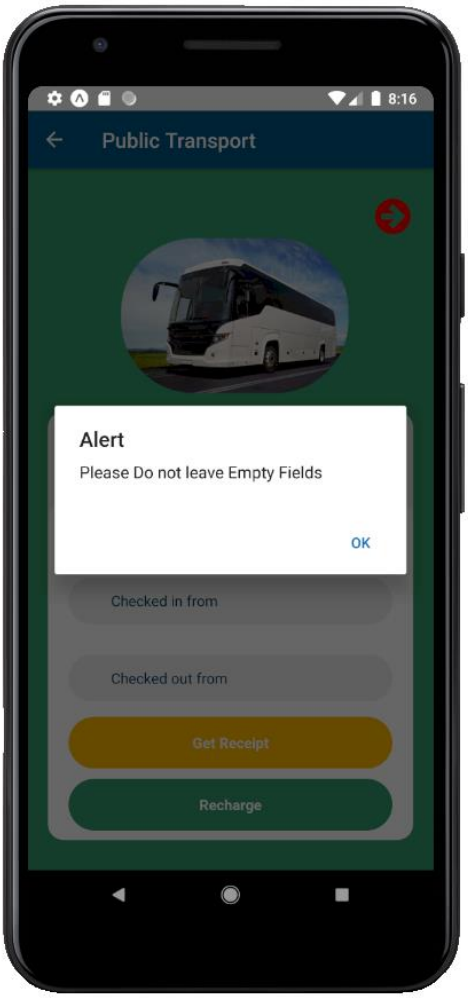



When a Passenger Logs into the Mobile App, they must enter the Smart Card ID to proceed.




When the Entered Smart Card ID is incorrect it will prompt an error



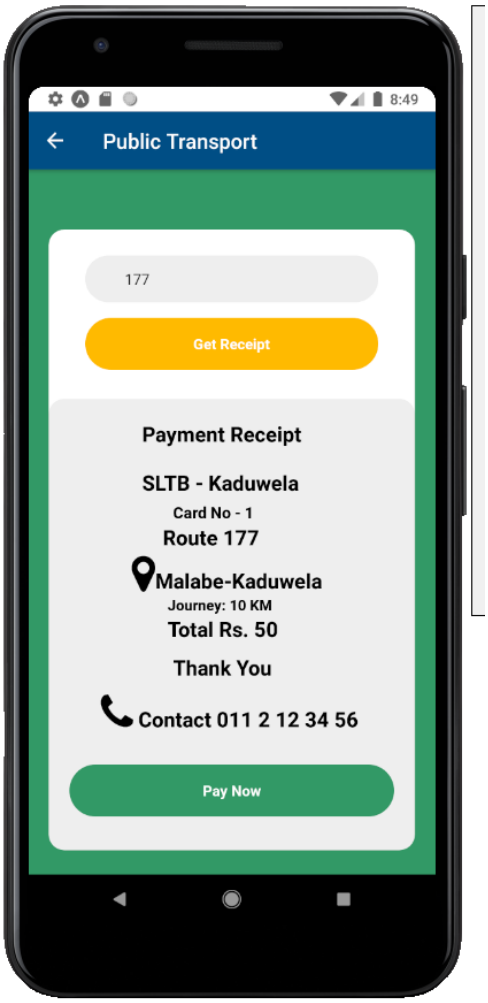




Upon successful login user will be navigated to the Home Page where they must select the Checked In (Start Location) and Checked out Location (Destination) to proceed. The smart card ID is displayed at the top.

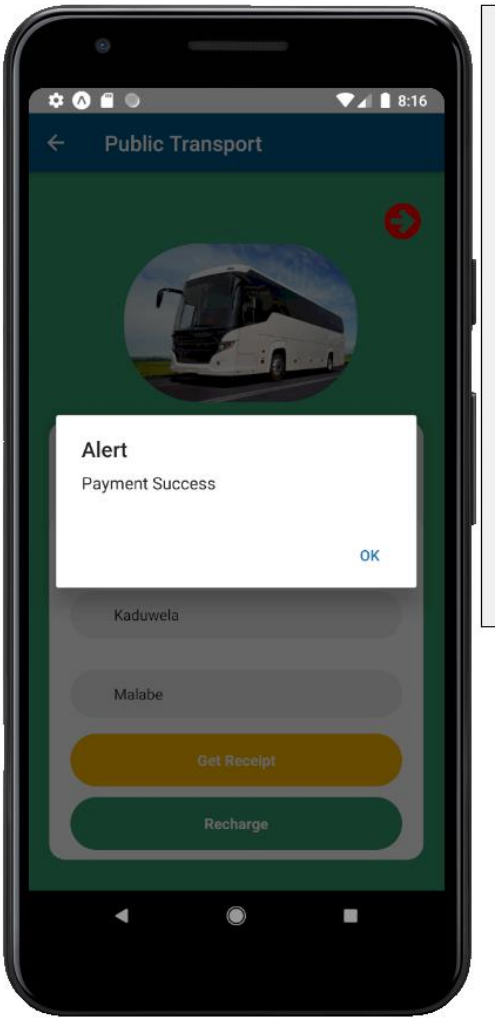
There are two options:

1. Get Receipt
2. Recharge

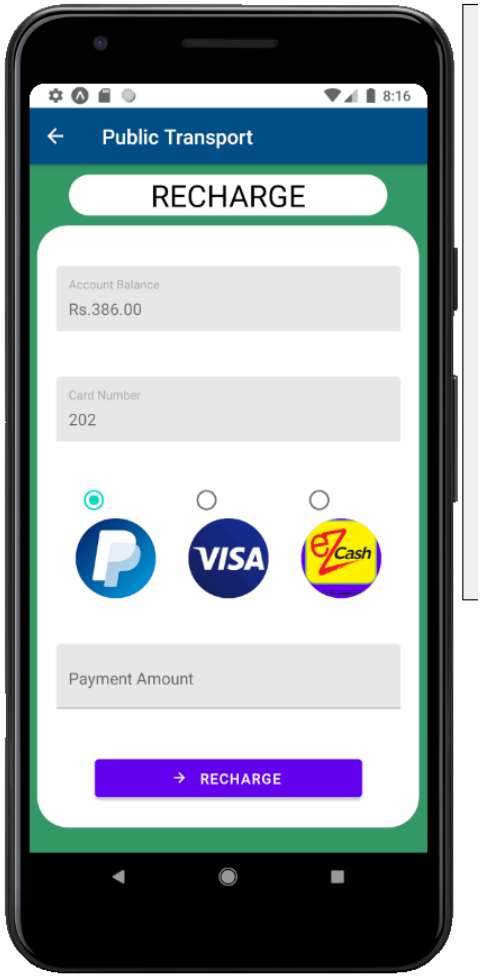
When user tries to proceed to Receipt page with empty fields it will prompt an error




After selecting the Start and End Locations and when Get Receipt Button is clicked, user will land on this page. Here the user must enter the Bus Route ID (Ex: 177) and click Get Receipt Button

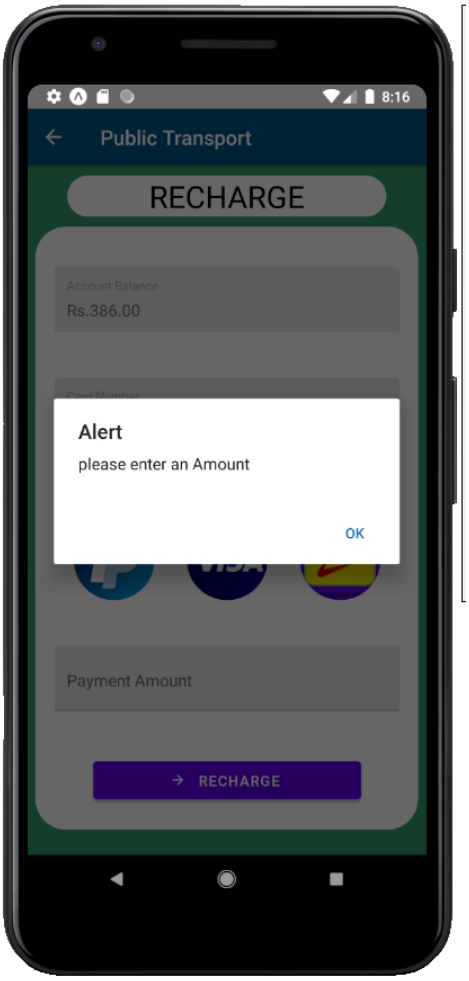



When the Receipt is generated, the user can click on Pay Now option, and it will prompt Payment Success Alert upon successful payment and will navigate to the Home Page again

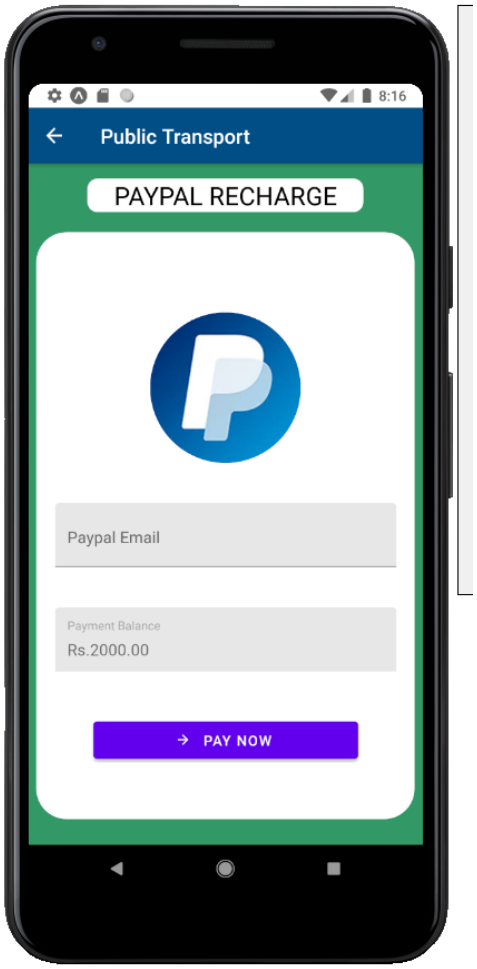



From the Home Page, when the Recharge option is selected, user will be navigated to this page. Here use must enter the Payment Amount to proceed. There are three main payment options as

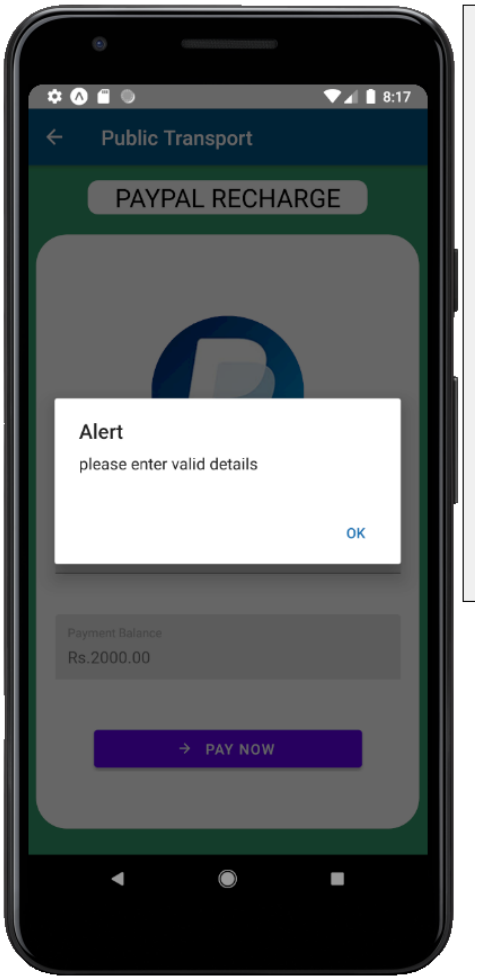
1. PayPal
2. VISA
3. eZ Cash



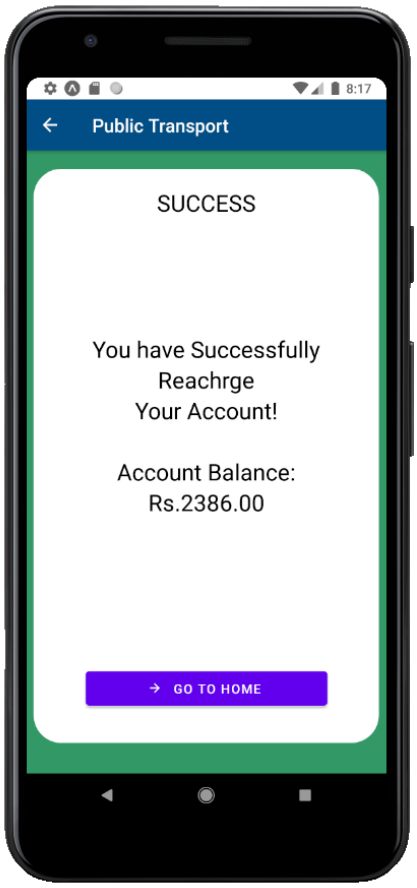

When user tries to proceed without entering the amount, an error will prompt



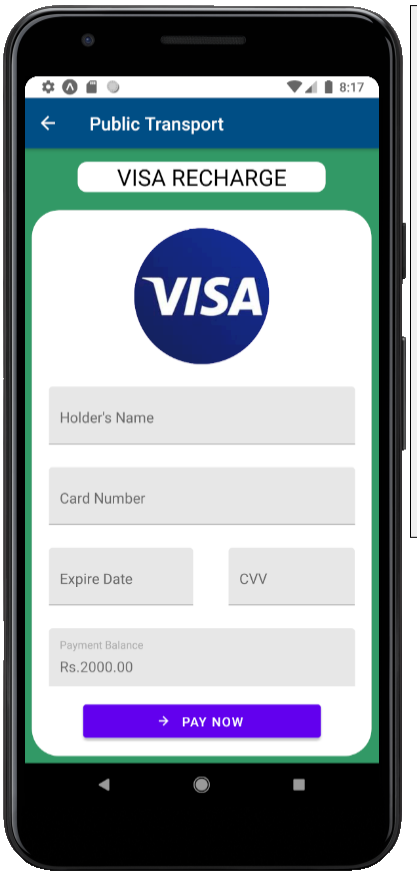

In the Recharge page when user selects PayPal option they will be directed into this page



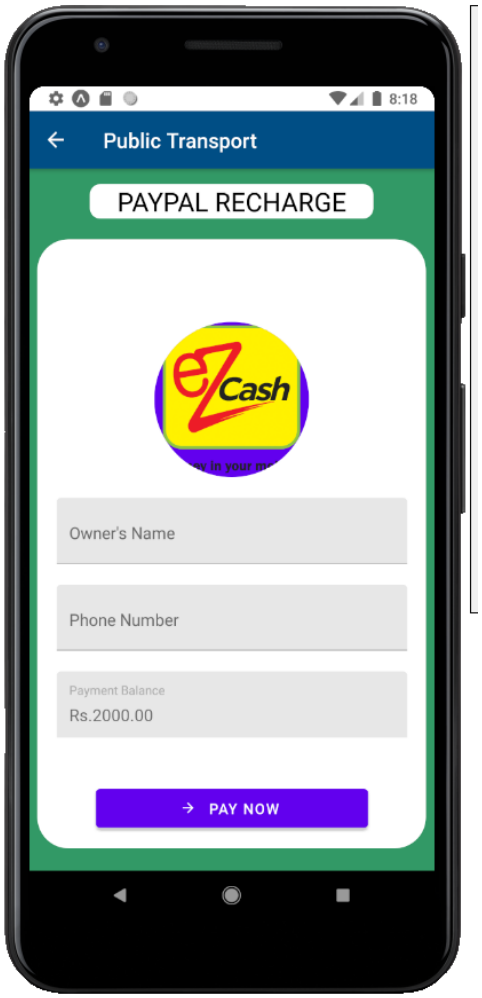

If user tries to proceed without adding details, it will prompt an error




Upon successful recharge, success message will be displayed as follows

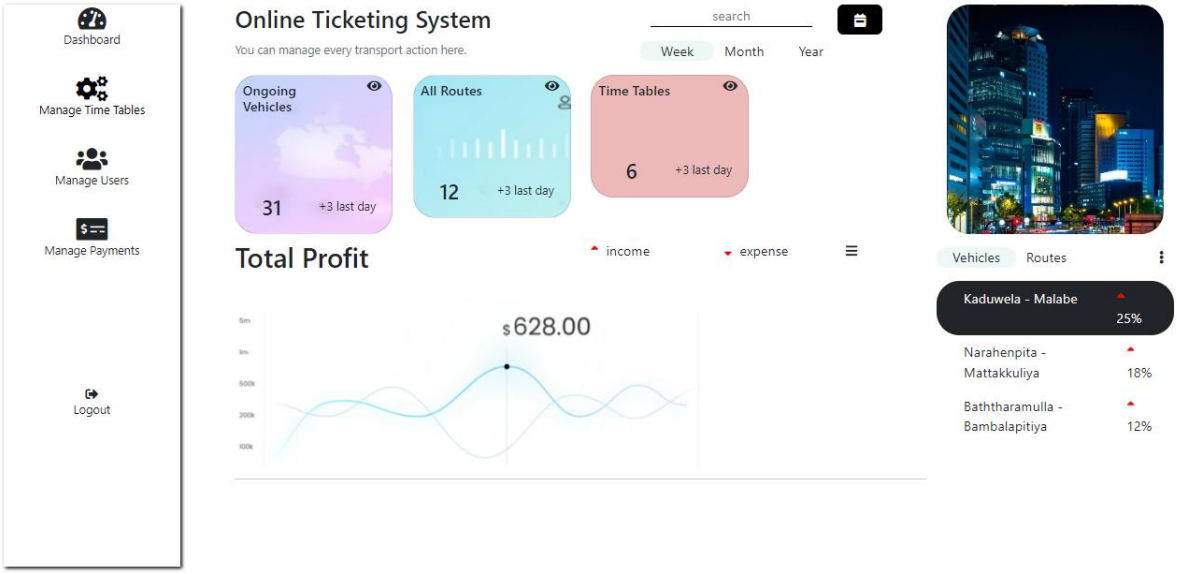



When user selects VISA option from the Recharge page, they will be prompted to this page.

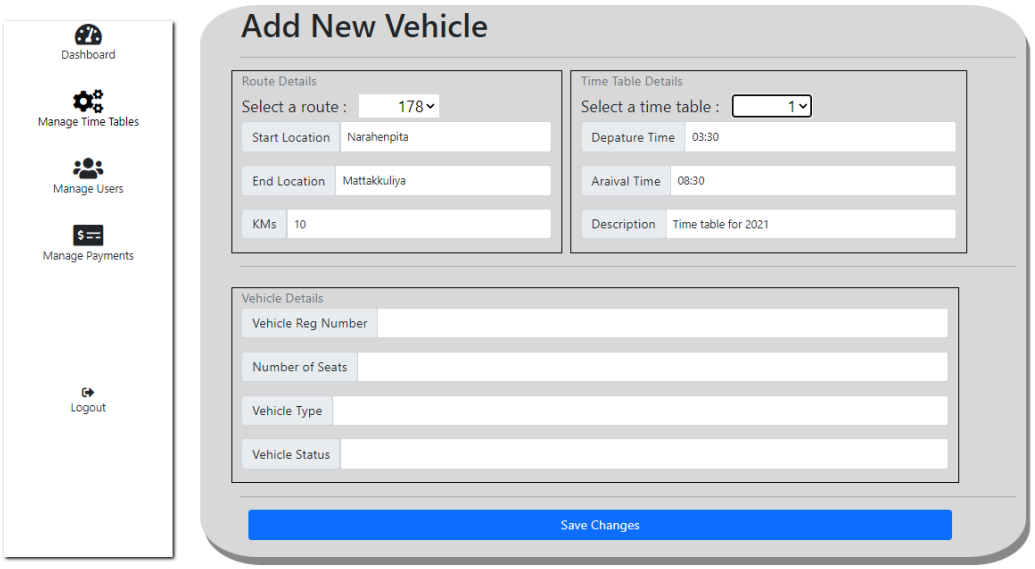



When user selects eZ cash option from the Recharge page, they will be prompted to this page.

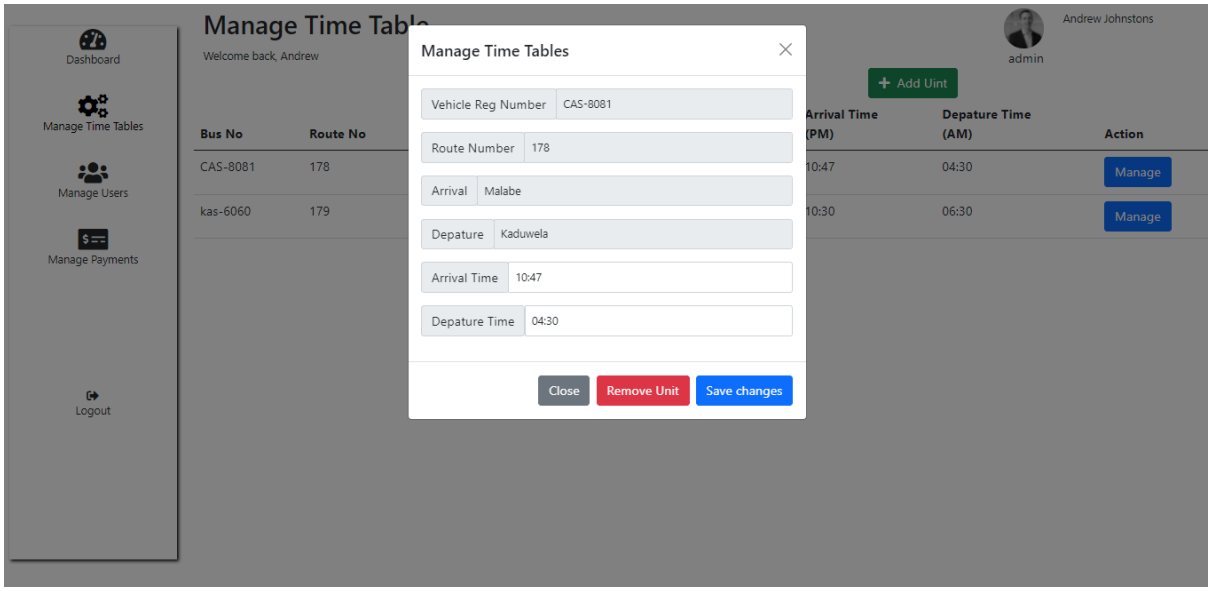
# High Fidelity Wireframes for Web Application



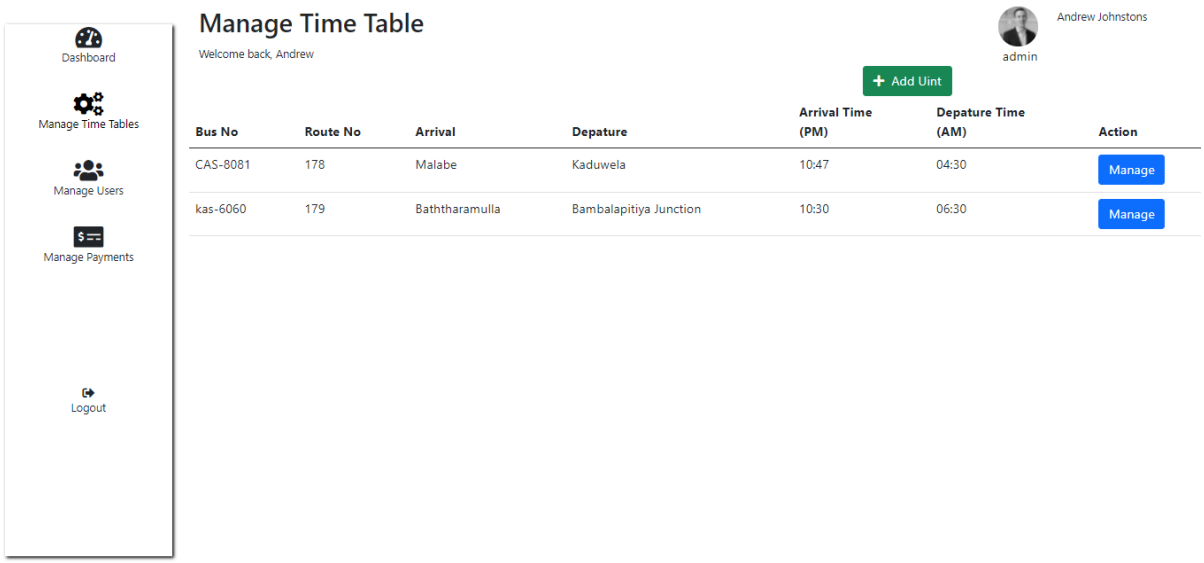

When an Admin Logs into the Online Ticketing Web Application, the respective user will land on the given Dashboard.




Here when the User Selects the Add New Vehicle option, they can add a new vehicle, Get Route Details by selecting the Route Number and Get Timetable Details based on the Timetable Number.




When Add New Time Table Option is selected, the User can Add Timetable details and can Add a new Timetable to the System.




From the Manage Timetable Option, User can view the specific Timetable details and can Update the existing details as well.

## 7. Unit Testing – Manual

| Test Case ID | Test Case Description | Test Steps | Test Data | Actual Result | Expected Result | Pass/Fail | Created By |
|---|---|---|---|---|---|---|---|
| TC01 | Check when Valid Data is entered in Adding a New Vehicle | 1. Visit Add New Vehicle Interface in the Web Application<br>2. Enter Valid Data<br>3. Click Save Changes | Vehicle Reg No: CA-6567 Number of Seats: 20 Vehicle Type: Bus Vehicle Status: Public | Display "Success" Message Alert | Vehicle added successfully | Pass | IT19004778 |
| TC02 | Check when Adding a Vehicle with empty fields | 1. Visit Add New Vehicle Interface in the Web Application<br>2. Leave Empty Input Fields<br>3. Click Save Changes | Vehicle Reg No: - Number of Seats: - Vehicle Type: - Vehicle Status: - | Display "Please fill the empty input fields" alert | Pop up an error | Pass | IT19004778 |
| TC03 | Check when Searching the Bus Route by Valid Route ID | 1. Visit the Receipt Page of the Mobile App<br>2. Enter Valid Route ID<br>3. Click Get Receipt Button | Route ID: 177 | Display Route Details and Generate Receipt | Display Route Details and Generate Receipt | Pass | IT19006994 |
| TC04 | Check when Searching the Bus Route by Empty Input Field for Route ID | 1. Visit the Receipt Page of the Mobile App<br>2. Leave empty Input Field for Route ID<br>3. Click Get Receipt Button | Route ID: - | Display "Please fill the empty input fields" alert | Pop up an error | Pass | IT19006994 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **TC05** | Check when Recharging the Smart Card with valid Data | 1. Visit the Recharge Page of the Mobile App<br>2. Enter Valid Details<br>3. Click Recharge Button | Payment Amount: 2000 | Display “Success ” Message | Card Recharge Successfully | Pass | IT19111766 |
| **TC06** | Check when Recharging the Smart Card with Empty Fields | 1. Visit the Recharge Page of the Mobile App<br>2. Leave Input Fields Empty<br>3. Click Recharge Button | Payment Amount: - | Display “Please fill the empty input fields” alert | Pop up an error | Pass | IT19111766 |

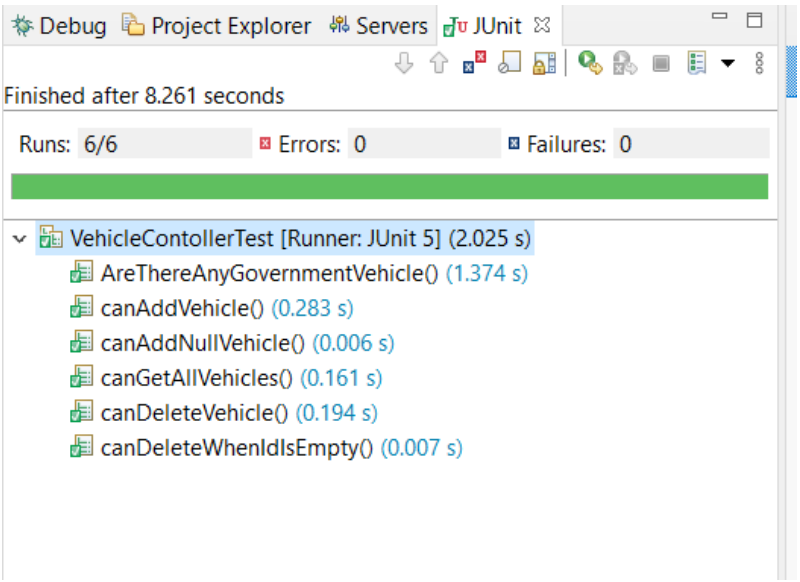
# 8. Unit Testing – Automated

IT19004778

VehicleContollerTest.java

```
50
51      //Test case for add a new vehicle
52      @Test
53      @Order(2)
54      void canAddVehicle() {
55          PrivateVehicle vehicle = new PrivateVehicle();
56          vehicle.setVehicleRegNumber("Test01");
57          vehicle.setVehicleType("Car");
58          vehicle.setNumberOfSeats(5);
59          underTest.addNewVehicle(vehicle);
60          assertNotNull(underTest.getVehicleById("Test01"));
61      }
62
63      //Test case for add null vehicle
64      @Test
65      @Order(3)
66      void canAddNullVehicle() {
67          boolean res = underTest.addNewVehicle(null);
68          assertEquals(false, res);
69      }
70
71      //Test case for check whether there is any private vehicle in the database
72      @Test
73      @Order(4)
74      void canGetAllVehicles() {
75          List<PrivateVehicle> vehicleList = underTest.getAllVehicles();
76          assertThat(vehicleList).size().isGreaterThan(0);
77
78      }
79      //Test case for delete particular vehicle
80      @Test
81      @Order(5)
82      void canDeleteVehicle() {
83          vehicleRepository.deleteById("Test01");
84          assertThat(vehicleRepository.existsById("Test01")).isEqualTo(false);
85      }
86
```

```
    //Test case for delete particular vehicle passing empty parameters
    @Test
    @Order(6)
    void canDeleteWhenIdIsEmpty() {
        boolean res =underTest.deleteVehicle("", "");
        assertEquals(false, res);
    }
}
```



IT19006994

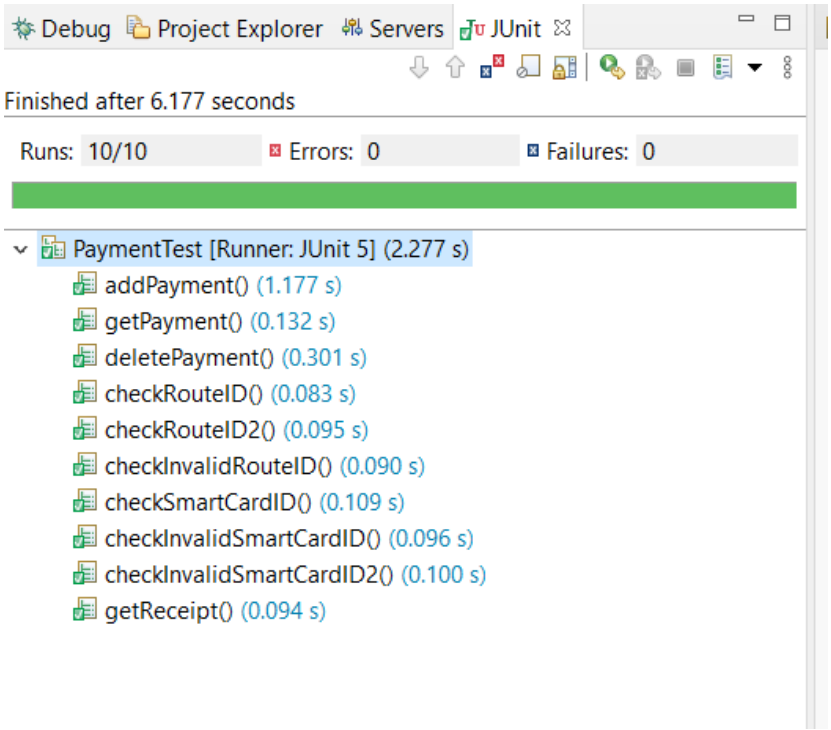
VehicleContollerTest.java | PaymentTest.java

```
42    @Test
43    @Order(1)
44    void addPayment() {
45        Payment payment = new Payment();
46        payment.setCardId(100);
47        payment.setPaymentAmount(1000);
48        Payment obj=paymentService.addPayment(payment);
49        assertNotNull(paymentRepository.findById(obj.getPaymentId()));
50    }
51
52    @Test
53    @Order(2)
54    void getPayment() {
55        List<Payment> paymentList = paymentService.getAllPayments();
56        assertThat(paymentList).size().isGreaterThan(0);
57    }
58
59    @Test
60    @Order(3)
61    void deletePayment() {
62        List<Payment> paymentList = paymentRepository.findAll();
63        Payment payment = null;
64        for(Payment p : paymentList) {
65            if(p.getCardId() == 100) {
66                payment = p;
67                break;
68            }
69        }
70        paymentRepository.deleteById(payment.getPaymentId());
71        assertThat(paymentRepository.existsById(payment.getPaymentId())).isEqualTo(false);
72    }
```

```
74    //Check Valid Route ID
75    @Test
76    @Order(4)
77    void checkRouteID() {
78        Route route = routeController.getRouteDetails(177);
79        boolean res = route == null ? false : true;
80        assertEquals(true,res);
81    }
82
83    //Check Invalid Route ID
84    @Test
85    @Order(5)
86    void checkRouteID2() {
87        Route route = routeController.getRouteDetails(190);
88        boolean res = route == null ? false : true;
89        assertEquals(false,res);
90    }
91    //Check Invalid Route ID
92    @Test
93    @Order(6)
94    void checkInvalidRouteID() {
95        Route route = routeController.getRouteDetails(190);
96        boolean res = route == null ? false : true;
97        assertNotEquals(true,res);
98    }
99
100   //Check Valid SmartCard ID
101   @Test
102   @Order(7)
103   void checkSmartCardID() {
104       SmartCard card = smartCardController.cardDetails(1);
105       boolean res = card == null ? false : true;
106       assertEquals(true,res);
107   }
```

```java
//Check invalid SmartCard ID
@Test
@Order(8)
void checkInvalidSmartCardID() {
    SmartCard card = smartCardController.cardDetails(200);
    boolean res = card == null ? false : true;
    assertEquals(false,res);
}
//Check invalid SmartCard ID
@Test
@Order(9)
void checkInvalidSmartCardID2() {
    SmartCard card = smartCardController.cardDetails(300);
    boolean res = card == null ? false : true;
    assertNotEquals(true,res);
}

//Get Receipt for Valid Route ID
@Test
@Order(10)
void getReceipt() {
    Receipt receipt = routeController.getReceipt(178);
    boolean res = receipt == null ? false : true;
    assertEquals(true,res);
}
```



IT19111766

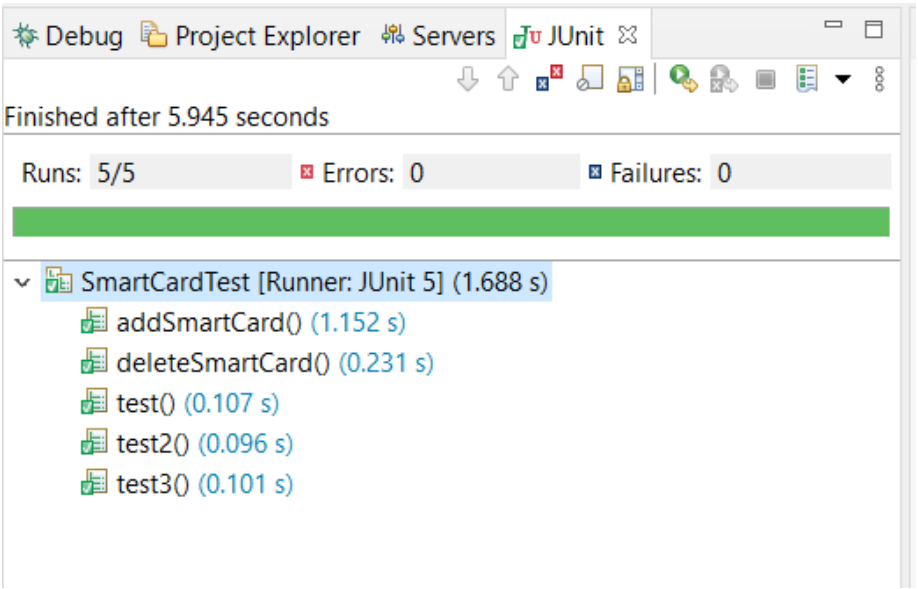
```java
//Add Smart Card
@Test
@Order(1)
void addSmartCard() {
    SmartCard smartcard = new SmartCard();
    smartcard.setCardId(111);
    smartcard.setUrlCode("1111");
    smartcard.setCardType("new");
    smartcard.setUserCode("1111");
    smartcard.setCardAmount(111);
    smartCardController.addDetails(smartcard);
    assertNotNull(smartCardRepository.findById(111));
}

//Delete Smart Card
@Test
@Order(2)
void deleteSmartCard() {
    smartCardRepository.deleteById(111);
    assertThat(smartCardRepository.existsById(111)).isEqualTo(false);
}

//Get All Payments
@Test
@Order(3)
void test() {
    int count = paymentController.getAllPayments().size();
    boolean res = count > 0 ? true : false;
    assertEquals(true, res);
}

//Get payment by Type
@Test
void test2() {
    int count = paymentController.findByType("debit").size();
    boolean res = count > 0 ? true : false;
    assertEquals(true, res);
}
```

```java
71
72      //Find Payment Type and ID
73      @Test
74      void test3() {
75          int count = paymentController.findByTypeAndID(202, "debit").size();
76          boolean res = count > 0 ? true : false;
77          assertEquals(true, res);
78      }
79
80  }
```



End of the Document