

**Sri Lanka Institute of Information Technology**

Ticketing System for a Public Transport Network

**Case Studies in Software Engineering**

**Critical Reflection Report**

Group ID :- CSSE\_WE\_16

Submitted by:

1. IT16178700 – D.I.K. Rajapakshe
2. IT17018760 – M.P.P. Shamil
3. IT17152938 – S.K. Liyanage
4. IT17056212 – P.M.C.P. Paththinisekara

Date of submission

2019-09-29

**Table Of Content**

Contents

[Introduction to the System 3](#_Toc20616742)

[Major Changes 4](#_Toc20616743)

[1. Changes to the underlying software structure (Class Diagram) 4](#_Toc20616744)

[2. Changes in the interaction (sequence diagrams) 7](#_Toc20616745)

[3. Usage of UI designs as provided in the (wireframes) 12](#_Toc20616746)

# Introduction to the System

Our topic was to develop a system for ticketing of public transport Network. According to the given design document there Our team come up with plan for the ticketing for public transport network in Sri Lanka. But when we were studying the design document in order for implementation there were some incorrect and unimplementable points in the design. So, there were some changes had to be done in order for us to implement the assigned system correctly and practically. So mainly we made major changes in,

1. Class Diagram
2. Sequence Diagrams
3. Interfaces

These Changes had to be made specially in the class diagram and Interfaces. That will be explained next. These Changes have been made in both frontend and backend of the project.

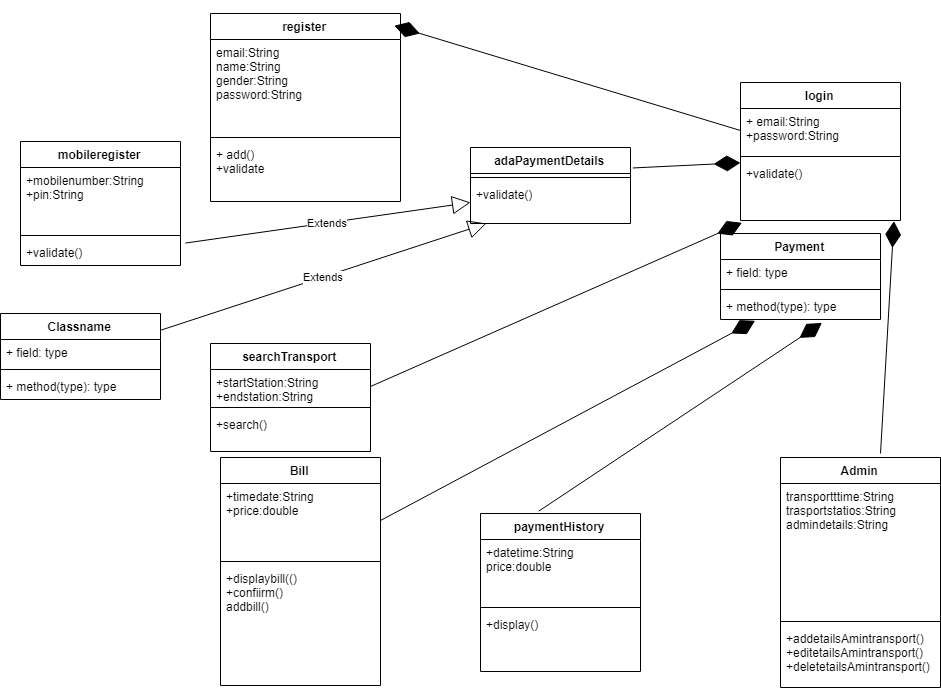
# Major Changes

We did major changes in both frontend and backend in throughout the implementation process. Specially in the class diagram we found it hard to implement because of its design. So we have explain the all the major changes below.

## Changes to the underlying software structure (Class Diagram)

We had to redesign most of the class diagram because the original design was mostly wrong.

This was the original diagram,



***1.a*** ***Initial Class Diagram***

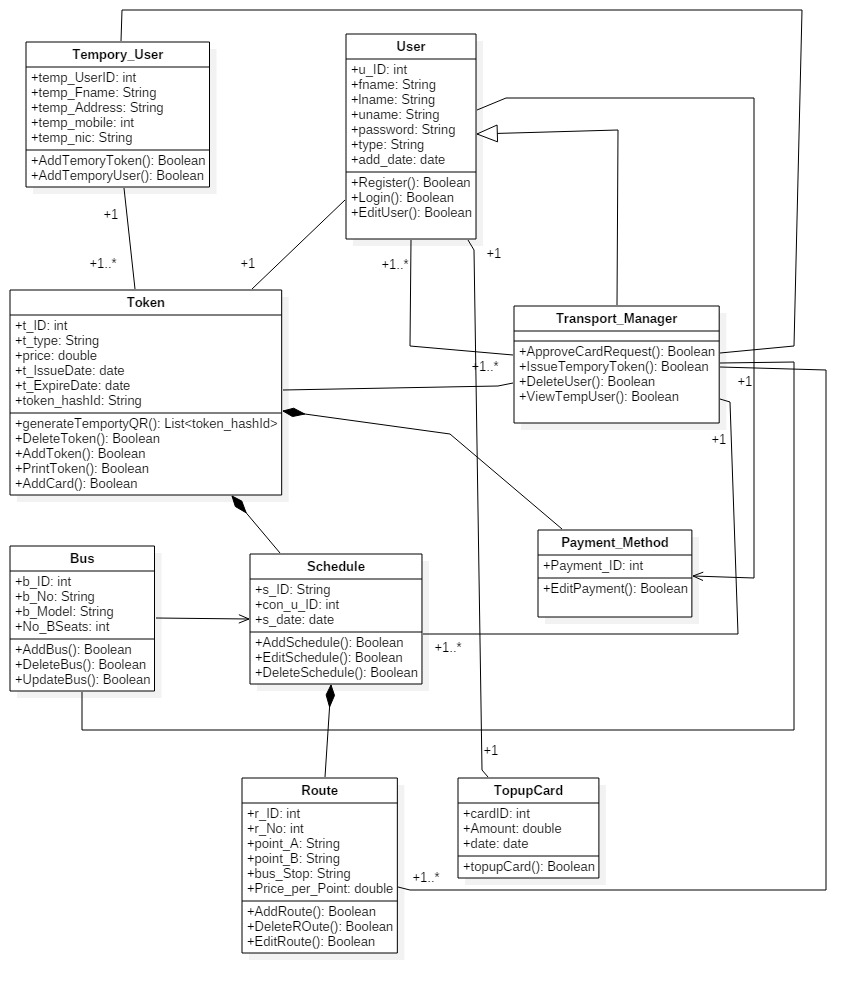
As you can see in the design of class diagram, they haven’t focused their attention on most of the major details.

**Changed classes: -**

1. **register/mobileregister/admin Classes**- There are two register classes in this diagram. They are register and mobile register. And also, there is an admin class. But according to our understanding all of these are fall under a one category which can be taken as the user. Creating and managing all of these classes are rater complicated and inefficient when implementing an application according to the standards. So, we replace all three classes with the **user** class. We also added an **TemporyUser** Class for unregistered users.
2. **paymentHistory/Payment Classes**- All of the data the exists in the paymentHistory class is already exists in the Payment class. And Thier Payment class not even a part of rest of the diagram. So, implementing the **Payment** class was enough. This was to prevent the duplication of codes.
3. **adaPaymentDetails Class** – It is hard to understand the meaning of this class. Considering it has no attributes and only have validate() method this could be the addPaymentDetails class . But we already have handled those functions in the **Payment** class. Either way there was no point of adding class with just the validate method.
4. **searchTransport Class –** This class is only connected to login class and there is no mentions of connection to the classes that contains tansport deatails.Obiously those classes don’t even exist in this diagram. So, there was no point of implementing a class with no valid details.

**Newly added Major classes: -**

1. **Bus/Route/Schedule Classes**- These major classes were added to manage the Buses, Routes and Schedule functions in the ticketing system. These classes are specially needed to handle the data needed for the other classes like payment for function.
2. **Token Class**- This class was used to handle the token(ticket) details.
3. **PaymentMethod** **Class** - We decided to keep this class as a separate class because this class handles all the methods of payments related to a specific customer.

****

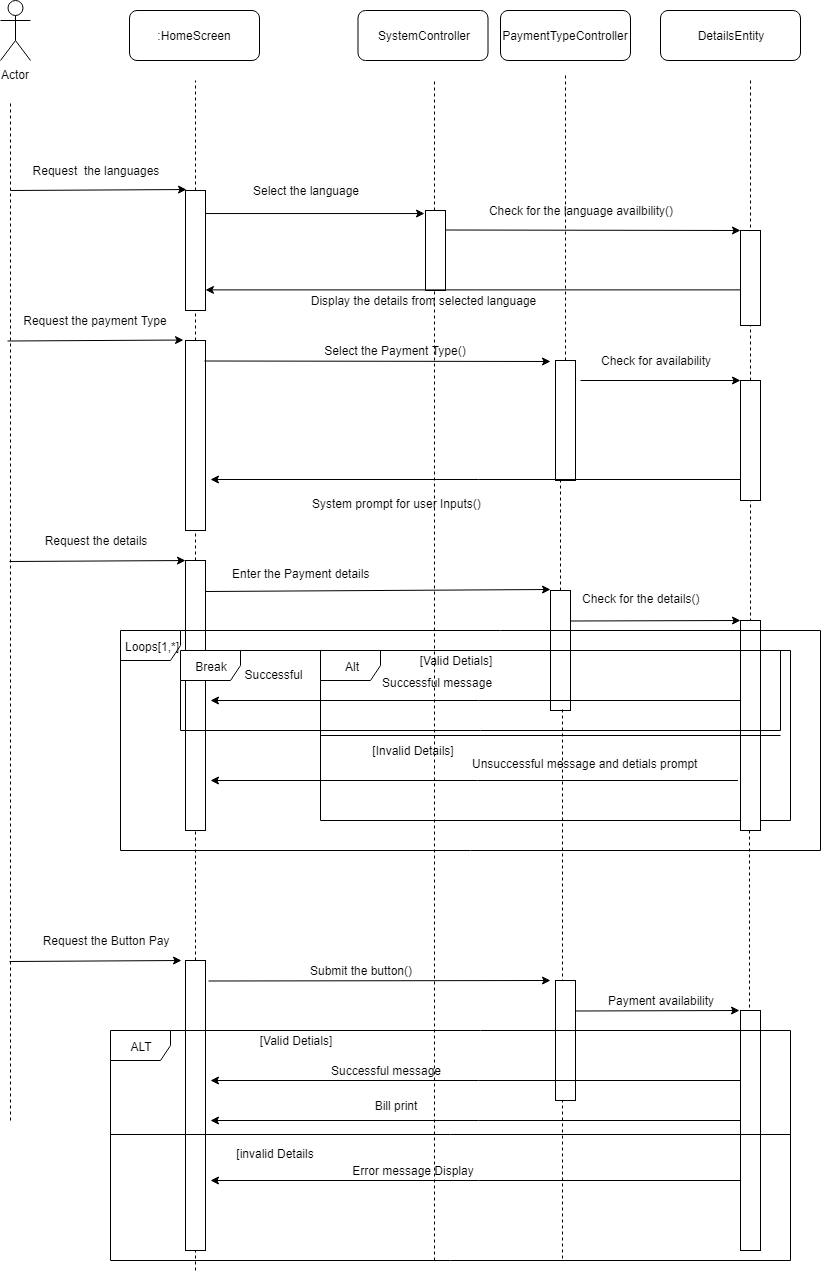
***1.b*** ***Redesigned Class Diagram***

## Changes in the interaction (sequence diagrams)

1. Add details for payment diagram: -

**Changes to the diagram**: -

* **Language select**: - According to the diagram there is a language select function to select a language and check for language availability of the system. So that means there should be lot of languages to select. So, we had to disregard that option because practically we couldn’t add many languages to the system.
* **Payment Availability**: - According to the diagram this system check for available account balance in the user’s bank account. We technically do that without a gateway to the relevant bank so we had assumed that the necessary amount of cash available for the transaction and complete the transaction unless the user entered invalid details.
* **Boundary/Controllers/Entities**- Diagram doesn’t contain correct marking and labels for us to implement the function correctly. As an example, they have added SystemController and DeatailsEntity as parts of the sequence bus we can’t understand what they are.

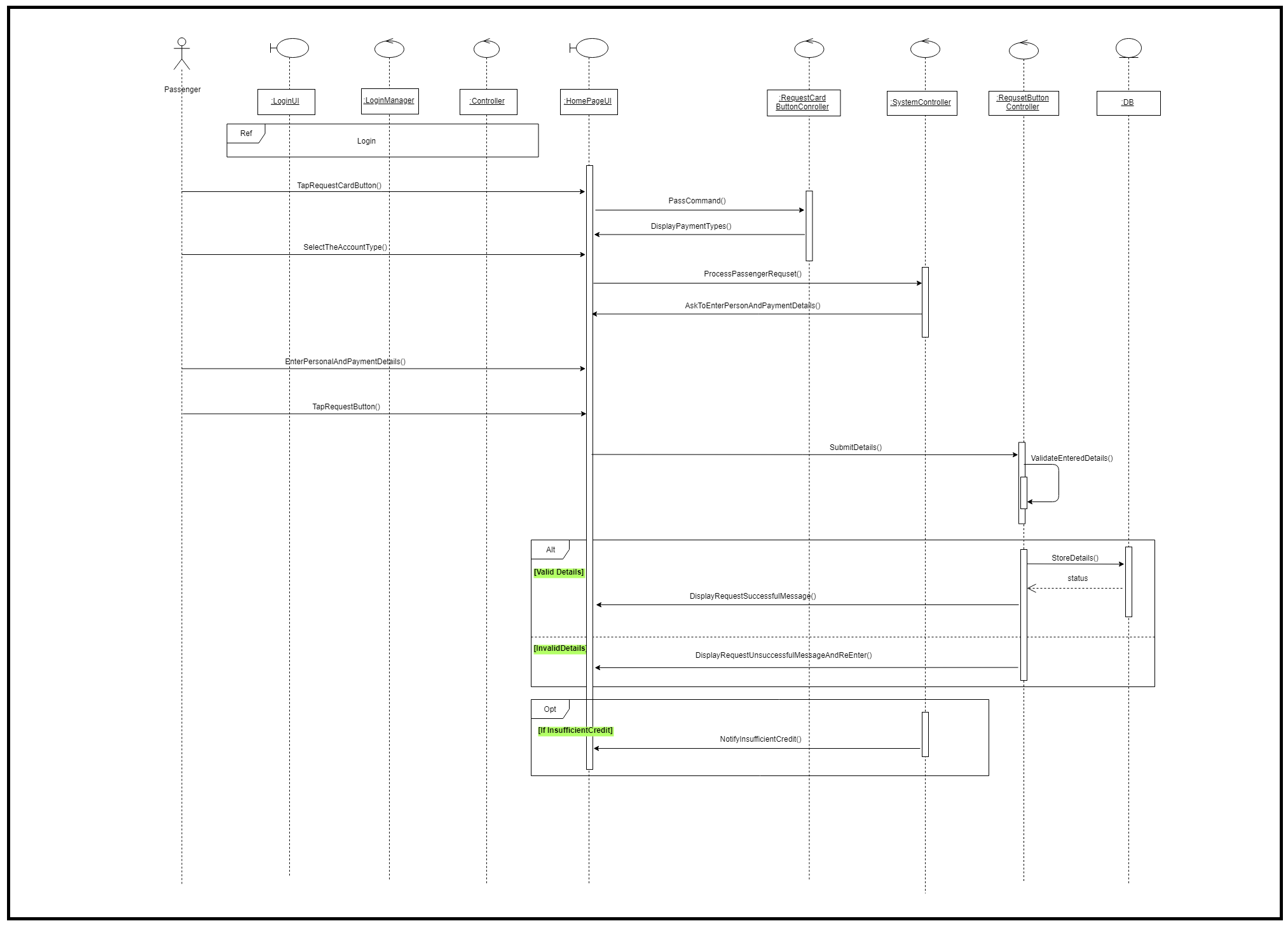


***2.a*** ***Initial*** ***add details for payment diagram.***

1. Request Passenger card diagram: -

**Changes to the diagram: -**

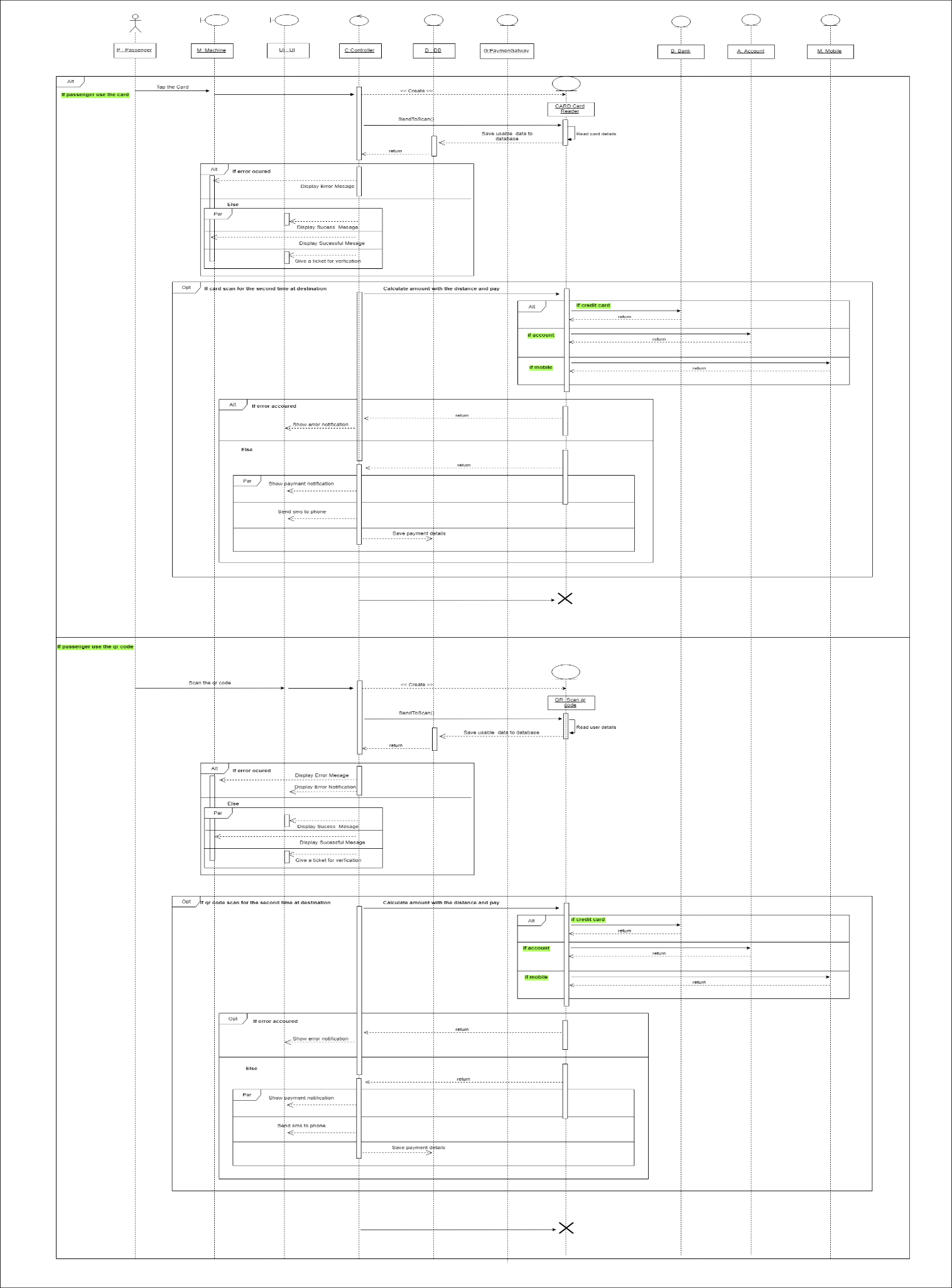
* **Enter personal details: -** According to the user interfaces in the design document there is a user registration function. So, when the user request for a passenger card that means users information are already in the system. So, there is no need to re enter the user’s personal information.



***2.b*** ***Initial*** ***request Passenger card diagram.***

1. Payment diagram: -

* **QR Code Scan (Changed this from passenger to conductor): -** The design document has given the scanning of QR code to passenger so they can scan QR code of the bus when both get in to the bus and at their destination. But practically it was practically unimplementable for us. Because in order to change the QR code according to busses location we would have get current GPS location of the bus and cross check it with the database. So, we implemented the functions so that the conductor can scan passenger cards QR code when get in the bus and manually enter the start point and the destination at the beginning of the journey.
* **Card Reader Scan: -** The design document has given tap the passenger card to card reader when get in to the bus to passengers but practically it is not implementable considering the hardware like card reader and card printer isn’t obtainable for us.

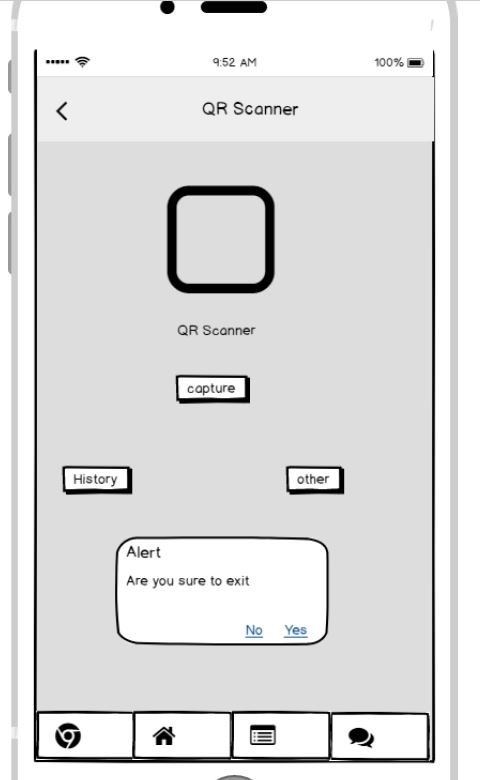


***2.c*** ***Initial*** ***payment diagram.***

## Usage of UI designs as provided in the (wireframes)

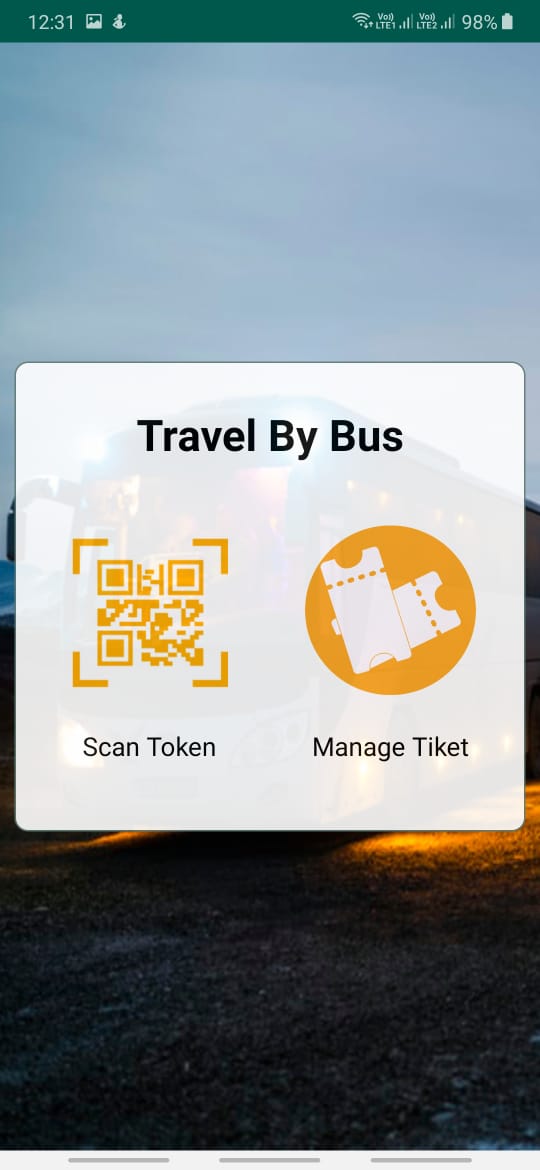
The Wireframes given in the design document haven’t been categorized according to the function and they are mixed together. And most of them are regarding to the user profile and register. Most of the wireframes regarding to the major business function are not in the design document. So we had to design them according to our own design.

1. Payment interfaces: -



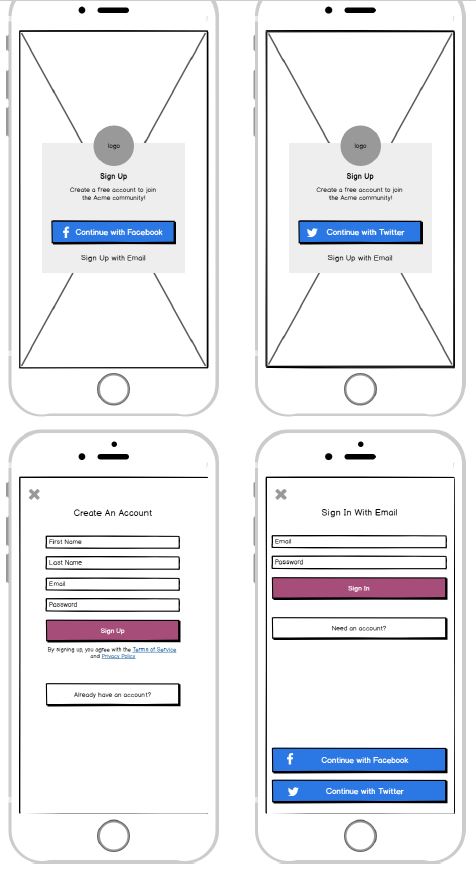
***3.a*** ***Initial*** ***payment interface***

* We Changed above payment slightly to make it bit user friendlier. And Instead of history button we added a wat to view the history and also manage ticket by adding Manage Ticket button.



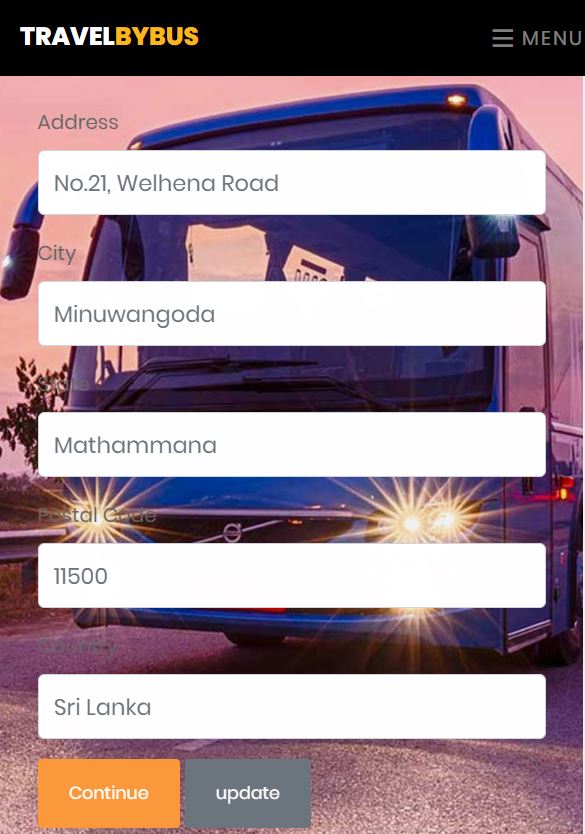
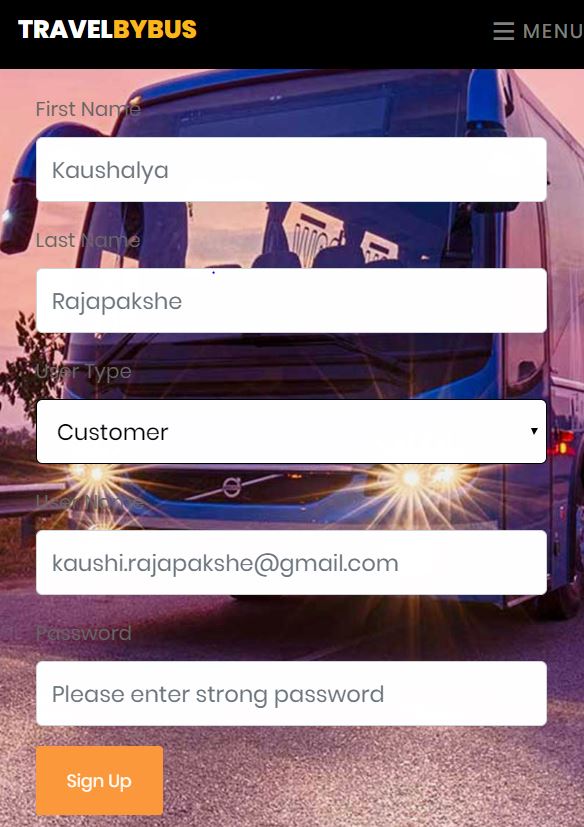
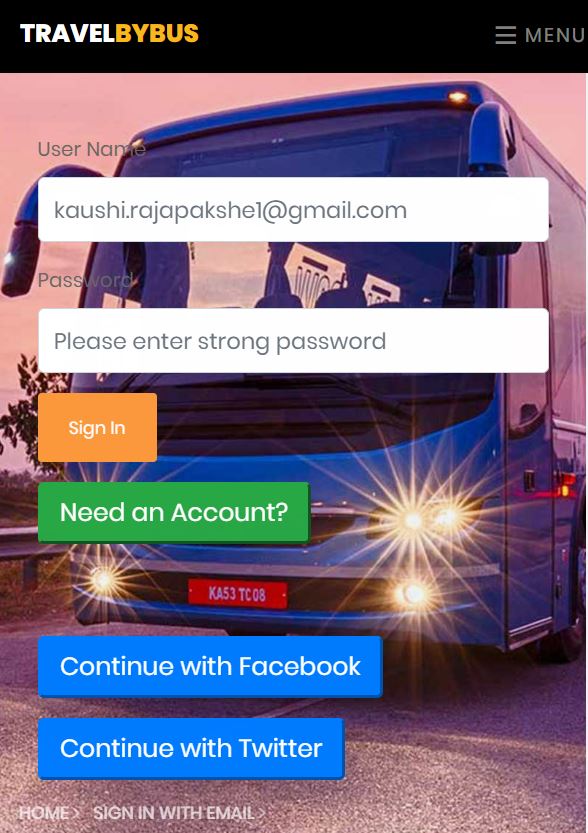
***3.b*** ***Redesigned*** ***payment interface***

1. User details interfaces: -



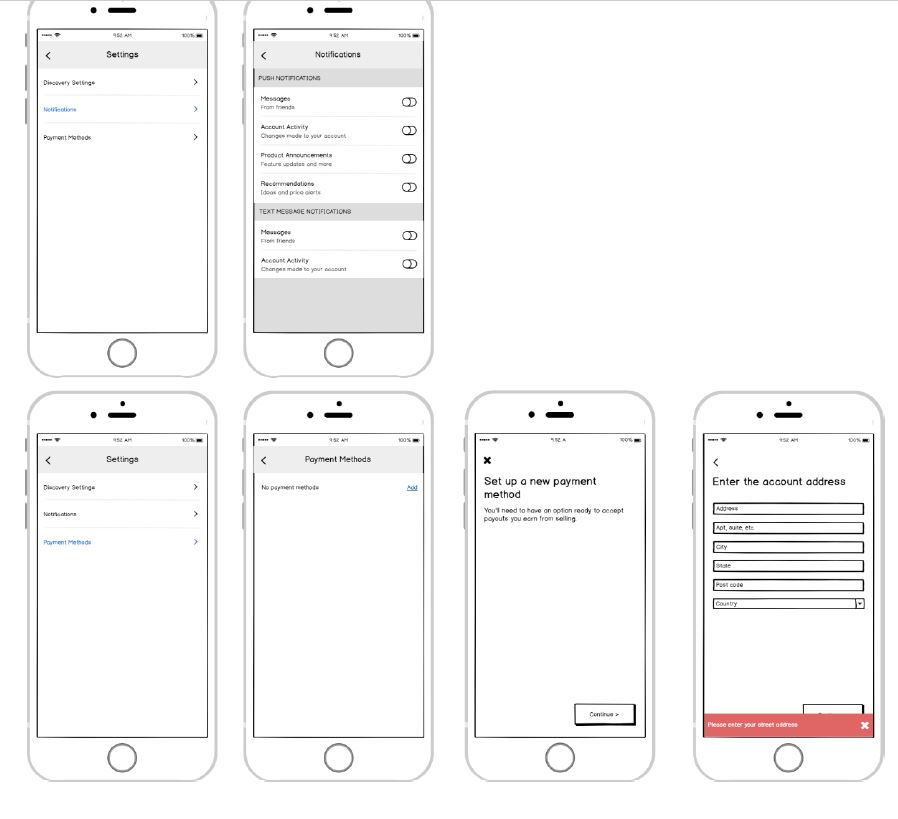
***3.c*** ***Initial*** ***user details interface***

* All of the above interfaces just for user registering but there is no need for 4 interfaces so we covered all those function in 2 interfaces;



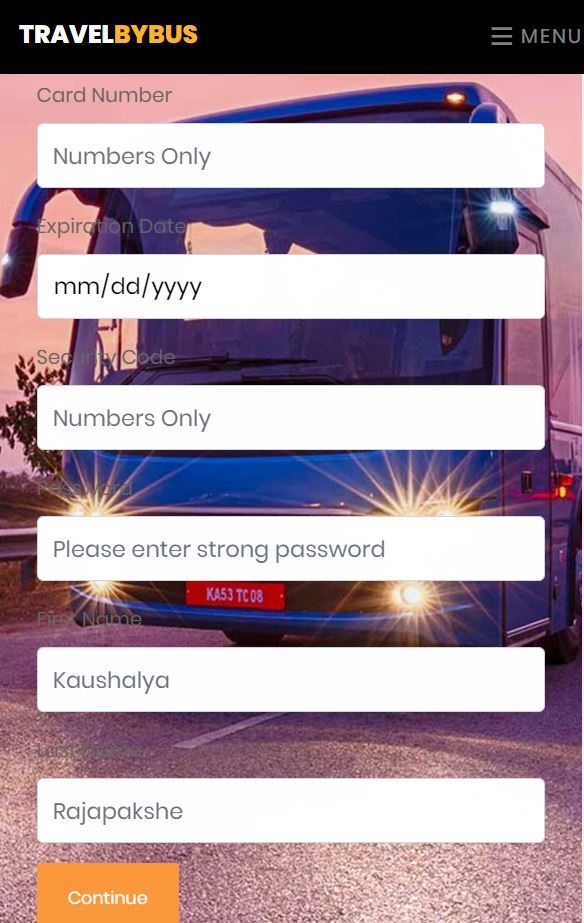
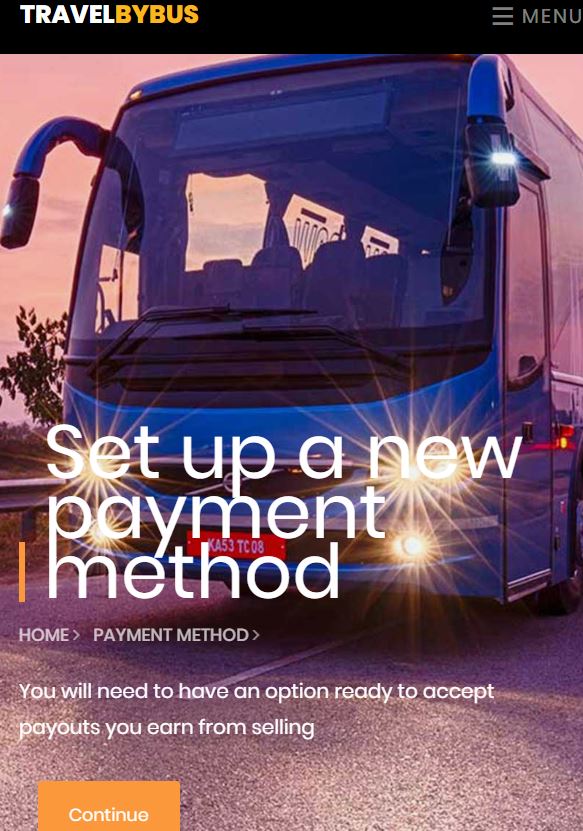
***3.d*** ***Redesigned*** ***user details interface***

1. Payment details interfaces: -



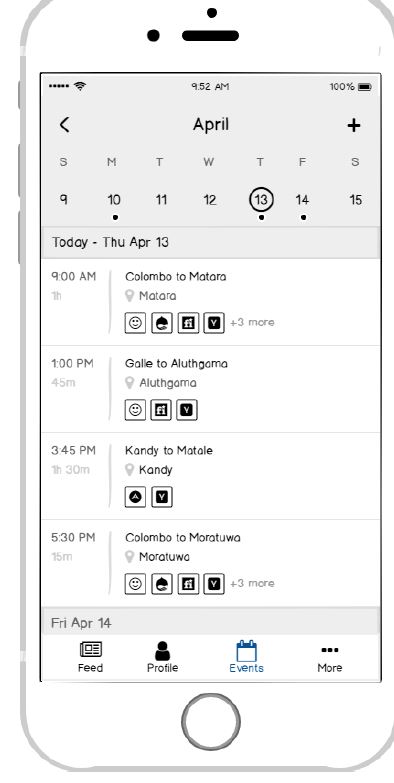
***3.e*** ***initial*** ***payment details interface***

* The path for go to payment method is for settings which not a part main business scenario. So, we added it to the navigation bar.



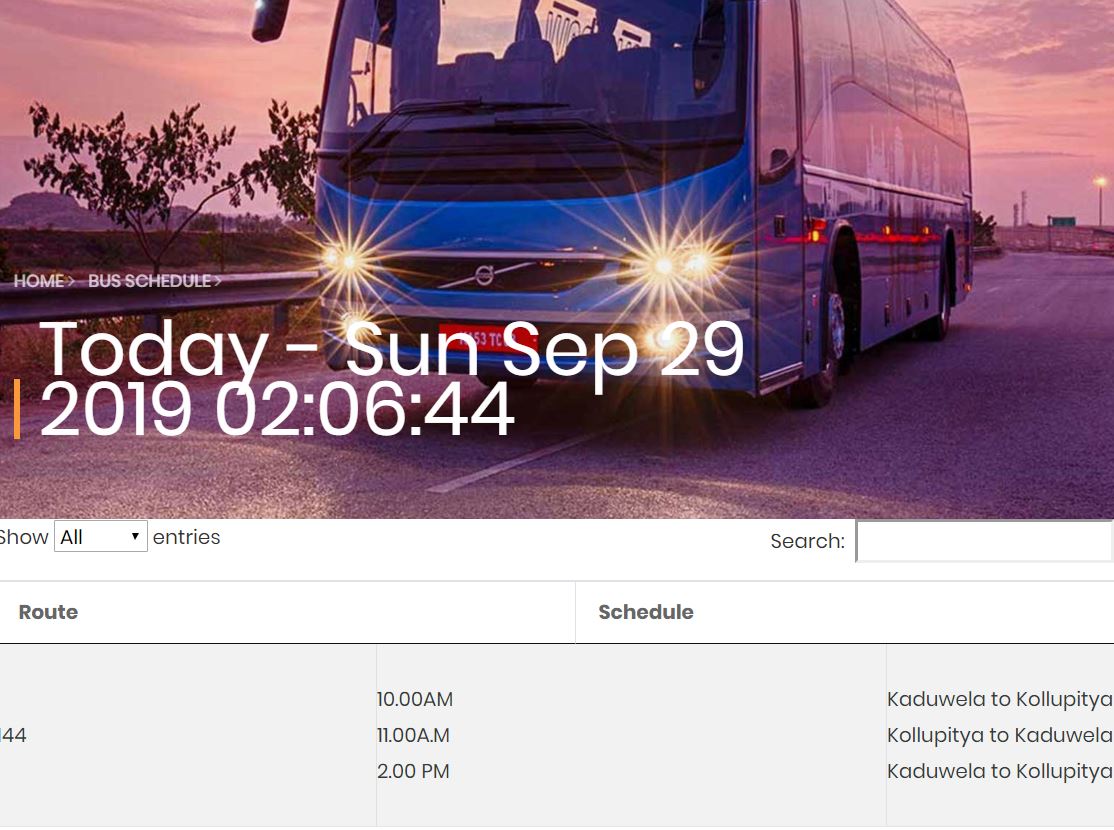
***3.f*** ***Redesigned*** ***payment details interface***

1. Schedule interfaces: -



***3.g*** ***initial*** ***schedule interface***

* The initial schedule document shows the current location of the bus. But practically we couldn’t implement that function so we had to remove it from the design and it also showed some weird icon below the route that didn’t explain in the document so we couldn’t implement those icons.

****

***3.h*** ***Redesigned*** ***schedule interface***

* None of these design document’s interfaces covers the Request Passenger card functionality so we had to design it according to our design.
* And also, there is no interfaces to manage buses, routes and schedules even though there is a bus schedule interface. But there must be in order for it to show in the schedule. So, we also added those interfaces as well;