Project report For

VIT Bus Service Management System

Submitted as a Lab Component for the course

SOFTWARE ENGINEERING (CSE3001) B. Tech (CSE)

By

Navdeep Sureka 19BCE2679

Under the Guidance of Dr. AKILA VICTOR

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING JUNE 2022

SUMMER SEMESTER 2022-23

Abstract

VIT Vellore has a huge campus with many academic blocks and hostels. Students must rush from building to building for classes within 5-10 min. This was solved by having shuttle bus services connecting all buildings and hostels of the campus. The one major problem that this brought up was the bus payment method. It charges 15 rupees per ride. Most of the time, either the driver or the student doesn't have change and it takes up a lot of time for the money exchange. Sometimes, the drivers don't accept online payments and there are network problems too. This issue is what we have tried solving in my own way.

Requirements

Hardware

RFID-Reader(low frequency)

- F Range: 30 to 300KHz
- Common frequency 125 KHz or 134KHz
- Relative Cost :
- Read Range <=10 cm

Software

- Android Studio
- Firebase
- Google Cloud
- Mysql

Procedure

Since the payment service must be used offline and we can't add any electronics to the bus, we came up with the idea to use the driver's phone to scan the code from the student to make the payment process. This can be done using a RFID scanner through an app.

We will scan the code which will be the RFID for the registration number of that student (similar to what we have on student ID card).

The student riding will just have to show the RFID to the driver and the driver will scan the code or to the RFID reader which would be attached to the bus which will be added to the database (offline).

At the end of the day as the driver connects the phone and the reader to the internet the database will be transferred to the main server and the online bill will be generated for the students who took the ride on that particular day.

Advantages:

1. Cashless transaction
2. Unified framework
3. Corrupt free
4. User friendly
5. Time saving

Disadvantage:

1. Double payment
2. Wrong usage of ID card

Measures of reducing disadvantages:

1. Can use mobile application to block the ID card transactions
2. Can increase the response time of card

Process Model

Since the problem scenario is very well known and has been experienced over a long duration of time.. The time constraint is less and the client (Bus Operators and Owners) has given detailed information about requirements and problems. Hence, while analyzing and planning, the divided tasks and modules give enough confidence to go with the **Waterfall Model**.

1. Introduction

1.1 Purpose

VIT shuttles are one of the major modes of transport used by VIT for students/teachers to travel from one academic block to another. However, these shuttles take INR 15 per ride in form of paper money which is one of the hectic tasks in the modern cashless era. There is also a chance that these contract workers can manipulate the logs and the data in the workbook as there is no software to record all the students/teachers who entered/rode the shuttle. The current project would be useful to take down these challenges provided with a touch-pay mechanism.

1.2 Document Conventions

In this document, we used Calibri as the base font; Bold for headings and subheadings; font size 12 for the body, 14 for subheadings and 18 for headings.

1.3 Intended Audience and Reading Suggestions

This document is appropriate to read for developers, users, testers and documentation writers. This software requirement document is well organized by the table of contents which makes the reader easier to surf throughout the document and can find the specific portion of the document if needed.

1.4 Product Scope

To design and develop a effective, efficient, maintainable and secured web application and also the hardware product for scanning the IDs of the student/teacher and store the information in logs also the android application where a student/teacher can add the points (1point=15 INR)

along with the driver/admin who can view the logs and verify the student/teacher in the bus also perform the CRUD operations for the students/teachers.

1.5 References

Cab Management System SRS | MCA IGNOU GROUP

2. Overall Description

2.1 Product Perspective

VIT Vellore has a huge campus with many academic blocks and hostels. Students/teachers must rush from building to building for classes within 5-10 min. This was solved by having shuttle bus services connecting all buildings and hostels of the campus. The one major problem that this brought up was the bus payment method. It charges 15 rupees per ride. Most of the time, either the driver or the student/teacher doesn't have change and it takes up a lot of time for the money exchange. Sometimes, the drivers don't accept online payments and there are network problems too. The main aim of our work is to overcome this problem with the required hardware and software which allows safe and secure online transactions that no one can manipulate.

2.2 Product Functions

- **Payment from Student/Teachers (Riders) to the driver of the particular bus:**

  The rider can use his/her phone or RFID enabled ID card to interact with the RFID reader available in the bus. This will trigger the Pay-on-tap function which will detect one token (INR value ~ 15rs) and update the same value to driver's system after validating the transaction.

- **Adding funds into the In-app wallet to use for future transactions:**

  The function that allow users to add funds to the wallet in terms of token for future transactions making the transaction more seamless and quick. The function will allow the user to transfer the money from bank account to the wallet using pre-tested and off the shelf payment module (eg razerpay).

- **Verification and validation of transactions by the bus driver and administrator:**

  Each transaction will have a unique ID which will include the Timestamp of the transaction, details of the student who made the transaction and also the other related information. This data will be cross referenced to verify the transaction and the false transaction will be discarded and the driver will be alerted of the failed transaction.

- **Log for all transactions by both users and the bus drivers/ administrators:**

  After the validation and verification of the transaction the transaction will be appended to the transaction log and can be referred in the future to cross verify the number of transactions in the period of time and will give both user and the bus driver the transaction count over the period of operation.

- **Tracking of the Bus through real-time GPS Tracker for rider convenience:**

  The tracking function will keep track of the bus location using Google's open-source map API and the live location of the bus will be provided to the user in real time. The Tracking function will also include the next stop with the estimated time and route of each bus in advance.

2.3 User Classes and Characteristics

Since the payment service must be used offline and we can't add any electronics to the bus so, I came up with the idea to use the driver's phone to scan the code from the student to make the payment process. This can be done using a QR code scanner through an app. We will scan the code which will be the QR for the registration number of that student (similar to what we have on student ID card). The student riding will just have to show the QR code to the driver and the driver will scan the code, which will be added to the database (offline). At the end of the day as the driver connects the phone to the internet the database will be transferred to the main server and the online bill will be generated for the students who took the ride on that particular day.

2.4 Operating Environment

This is an extremely versatile application, and will work flawlessly on any android version 10.0 or greater. However, the device must be able to connect to the internet in order for it to connect to the database or perform other functions such as password change, payment, etc.

2.5 Design and Implementation Constraints

The resources are quite clear. Constraints that are going to be used are user and driver/admin authentication, student/teacher payment, student/teacher wallet update, bus tracking, bus history, invoice generation, bus tracking, user database, admin database, bus database. Based on the requests and reviews taken from admin the changes would be made to the project but the core functionality would remain the same.

2.6 User Documentation

The following components can be expected in the user documentation is:

- What the software is
- Features of software

- How to login/create new users
- Add card or other wallets to the account
- How to pair ID and mobile number
- Location of the bus
- Examples of above tasks in pictorial form
- Helpline number and guide

2.7 Assumptions and Dependencies

Resource Assumption:

- All the project team members are available and have the necessary skills and knowledge to work on the project.
- All the software's work flawlessly on our mobiles.

Budget Assumptions:

- Personnel costs will not change.
- Overall economic conditions will stay the same.

Scope Assumptions:

- Scope doesn't change.
- If it should; project will follow a change control approval process scheduling.

Assumptions/Constraints:

- Each member of the group is expected to work for 5-6 hours a week.
- The set deadlines and milestones are achievable and the project can be completed on time.
- Project is expected to be completed by 9th July.

All the studies have been carried out and assessed well beforehand and there are no high risk assumptions that would jeopardize the project.

A few factors we are dependent on are:

- Equal contribution from each project member.
- Continued faculty support
- Access to software used to develop the application.

3. External Interface Requirements

3.1 User Interfaces

The user interface of this application will include various logical characteristics such as:

- The loading screen: A graphic related to the project will be displayed along with the icon and watermark, as the application loads and connects to the server.
- Input method: Since it will be used exclusively on mobile devices, all the input methods will be standard mobile inputs(touch).
- Screen layout: The Log-in page of the app will feature the standard username and password input text fields, and a button for logging in and another button for 'forgot password'.
- Functions: The various functionalities of this app are as follows:

1. The payment module: The payment button will redirect to an online payment gateway such as PayPal or Paytm, which will be used to purchase credits.

2.Balance module: This button works by displaying the credits available to use in your account.

3.Transaction History button: This button displays the various instances where you have spent your credits to travel in a vit bus.

4.Help: This button displays contact information of the technical support staff for any queries or problems encountered by users. All text fields will be inputted by the on-screen keyboard in mobiles. Errors will appear for simple errors such as a transaction failure, or insufficient credits, or bad network environments etc.

3.2 Hardware Interfaces

The contactless payment technology known as RFID (Radio Frequency Identification Device) is used in this project. The RFID card used by the bus drivers will record and store transactions by reading our student/staff ID cards. The card then uses the internet to send these transactions to the central database. The customer interface only interacts with the database for updating the credits left in the account's balance.

3.3 Software Interfaces

The User Application will be available on both Android OS (versions 10.0v and higher) and IOS(versions 9.3 and higher). The Android App will be written in Java Language, and the IOS app will be written using Swift. The main data items received by the system will be the transaction records in the form of csv files, which will then be updated in the databases. Payments will also be recorded in case of the user application when the payment gateway is used to purchase credits. The sessions are also maintained by the servers to ensure that the usage is authorized properly and not misused.

3.4 Communications Interfaces

The main communication functions involved in the user application are:

1. The password reset page of the application will require a verification email containing an OTP sent to the user's email address.
2. The communications between the server and the app are encrypted by using REST-based on HTTP. The best protection method for this model of communication is the TLS/SSL standard. It can be combined with the HTTP protocol to create an encrypted variant called HTTPS. HTTPS ensures safe, encrypted communication between apps and server.
3. The data synchronization in the mobile applications can be achieved using Sync Service and Sync Adapter. A Sync Adapter is a plug-in that handles the background of the Android app sync data with the server. This plug-in is registered on the platform's Sync Manager, which is in charge of running it and can be triggered when needed, requested, or scheduled.

4. System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 System Feature 1

4.1.1 Description and Priority

Description: Cash should not be needed.(Main objective).

Priority: High

4.2 Wallet Facility

4.2.1 Description and Priority

Description: The wallet should have money, otherwise from student's caution money.

Priority: Medium

4.3 Network Independent

4.3.1 Description and Priority

Description: Should be network independent.

Priority: High

4.4 Bus Positioning

4.4.1 Description and Priority

Description: Could facilitate user, not necessity.

Priority: Low

4.5 Database Tracking

4.5.1 Description and Priority

Description: Database tracking should be easy to check errors.

Priority: Medium

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The performance requirements are:

- Android devices with android versions greater than 10.0

- A responsive database to store the credits done by the students/teachers, trips made by particular students/teachers and details of the bus. The database server has to be designed in such a way that it has minimum response time to provide the seamless working of the software without much lag.

5.2 Safety Requirements

The RFID reader has to be programmed in such a way that the student/teacher is able to scan the ID and make sure that it is done without any delay.

Also, the students/teachers and drivers have to be careful about who they share their login details with as they can be misused.

5.3 Security Requirements

The database server has to be as secure as possible so as to avoid any data breach or data leaks which can be harmful to the students/teachers as well as drivers. The most important part is to

properly encrypt the database access key so that no authorized personnel can access the private information of the other such as mobile numbers, passwords, card details, etc.

Also, the login screen has to be designed in such a way that any bystander cannot take a peek at the login details. This can be done in various ways:

- Masking the login password with * in place of all characters.

- Integrating a two-factor authentication system for both student/teacher and driver.

5.4 Software Quality Attributes

In the future, if the software has widespread use and applications, it can also be ported to other operating systems such as iOS, etc., so that it can be made available to a larger group of users.

Also, the software has to be designed so that it can be further customized to be useful in other fields apart from the targeted group. The database server will also have to be robust with as little downtime as possible so as to avoid any inconvenience for the user.

5.5 Business Rules

The main business rule is to complete the project before the deadline without running out of budget. This can be done be implementing the following model:

- Firstly, the payment for the development of the software has to be divided in parts so that the developers don't go out of budget. The payment can be divided into three parts: 40% upfront cost at the start of the project; 20% after the development of module 2 is finished; and the remaining 40% at the delivery and deployment of the project.

- Second, the developers should follow the project timetable at all costs so that the project is delivered on time.

- Last but not the least, the project should have some room for customization so that it can be made much more user friendly according to the requirements of the deployment field.

6. Other Requirements

The database software used for the databases required in this application will be MySQL. A commercial license will be required for using MySQL for production. Various databases such as Customer database, Driver Database, Admin database, Bus database are required for the app to have full functionality. Legal requirements involved will be the terms and conditions of Google Play Store and Apple store. The app must adhere to these terms and conditions for the app to be available on these platforms. The security mechanisms must be up to industrial standard since it involves online transactions and in-app purchases. Proper and sufficient technical support is a necessity for the application to be used widely.

Appendix A: Glossary

1. RFID - Radio Frequency Identification Device
2. ID - Student/staff Identity Card
3. CRUD Operations : create, read, update and delete.
4. OTP - One Time Password(Authentication method)
5. GPS tracker - Global Positioning System technology used to track a device

RISK MANAGEMENT

Impact

- HIGH
- MEDIUM
- LOW

Technology

- The database used in the system cannot process as many queries per second as expected. MEDIUM
- The API is not working properly or not performing with full functionality. HIGH

People

- It is harder to find skilled drivers to operate the RFID. LOW
- Key staff are ill and unavailable at critical times. MEDIUM
- Loophole so that drivers are cheating. HIGH
- Easy for students/drivers to sabotage the hardware device. MEDIUM

Organisational

- Bus Management is restructured. LOW

Tools

- The tools to configure the hardware are outdated. MEDIUM

Requirements

- Required modules fail to integrate. HIGH

- API is not working. HIGH
- Network is not working even when required to extract database. HIGH

Estimation

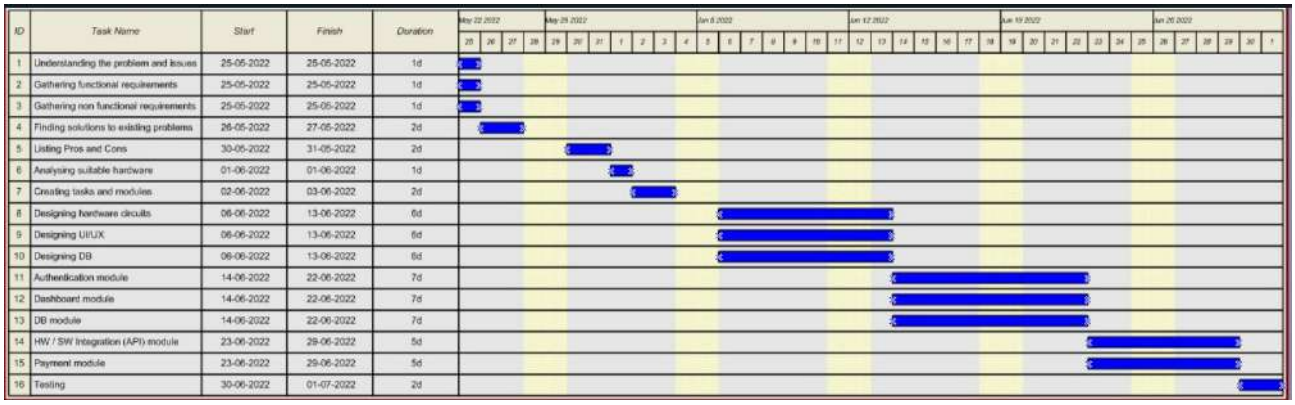
- The time required to develop this software is underestimated. MEDIUM

The price of hardware and installation is higher than expected. MEDIUM

TIMELINE CHARTS

GANTT CHART:

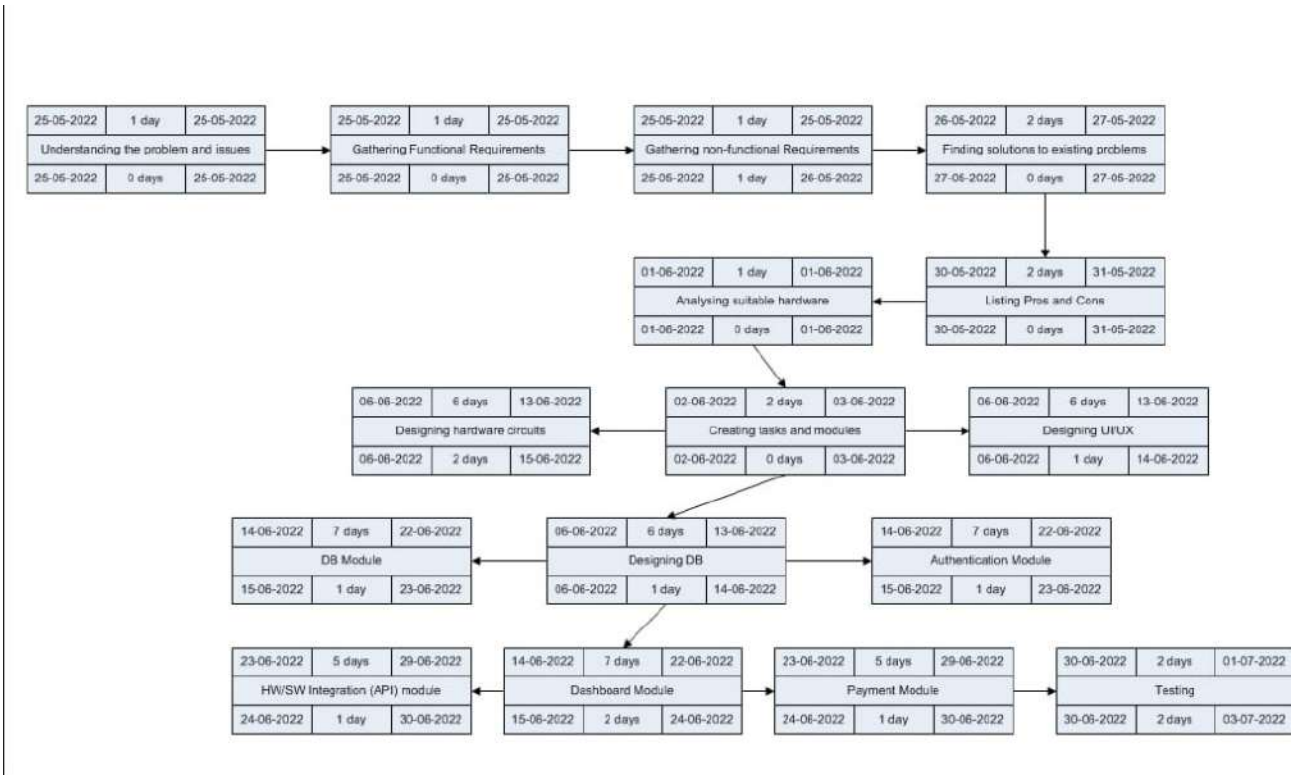
A Gantt chart is a type of bar chart that illustrates a project schedule. This chart lists the tasks to be performed on the vertical axis, and time intervals on the horizontal axis. The width of the horizontal bars in the graph shows the duration of each activity. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements constitute the work breakdown structure of the project. Modern Gantt charts also show the dependency (i.e., precedence network) relationships between activities. Gantt charts can be used to show current schedule status using percent-complete shadings and a vertical "TODAY" line.



ID	Task Name	Start	Finish	Duration
1	Understanding the problem and issues	25-05-2022	25-05-2022	1d
2	Gathering functional requirements	25-05-2022	25-05-2022	1d
3	Gathering non functional requirements	25-05-2022	25-05-2022	1d
4	Finding solutions to existing problems	26-05-2022	27-05-2022	2d
5	Listing Pros and Cons	30-05-2022	31-05-2022	2d
6	Analysing suitable hardware	01-06-2022	01-06-2022	1d
7	Creating tasks and modules	02-06-2022	03-06-2022	2d
8	Designing hardware circuits	06-06-2022	13-06-2022	6d
9	Designing UI/UX	06-06-2022	13-06-2022	6d
10	Designing DB	06-06-2022	13-06-2022	6d
11	Authentication module	14-06-2022	22-06-2022	7d
12	Dashboard module	14-06-2022	22-06-2022	7d
13	DB module	14-06-2022	22-06-2022	7d
14	HW / SW Integration (API) module	23-06-2022	28-06-2022	5d
15	Payment module	23-06-2022	29-06-2022	5d
16	Testing	30-06-2022	01-07-2022	2d

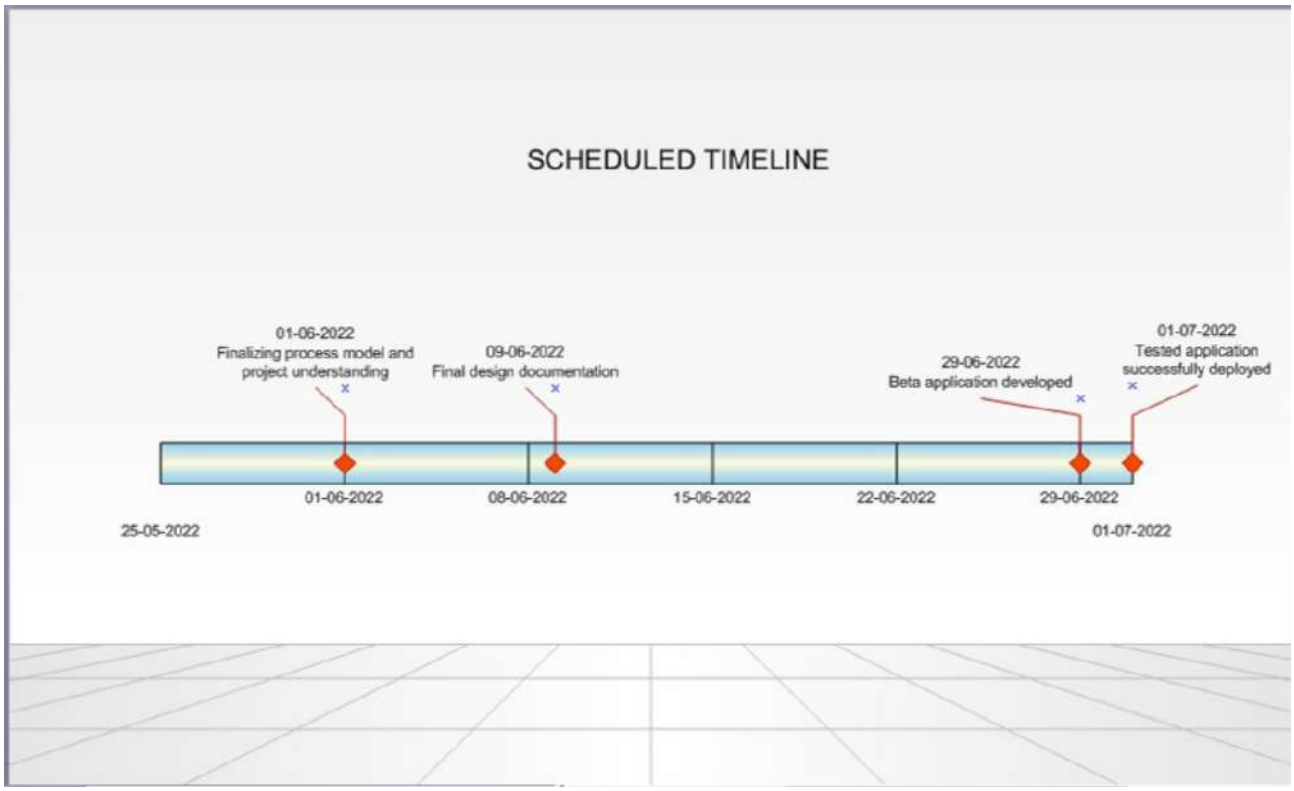
PERT CHART:

A pert chart, sometimes called a pert diagram, is a project management tool used to schedule, organize and coordinate tasks within a project. It provides a graphical representation of a project's timeline that enables project managers to break down each individual task in the project for analysis.



SCHEDULED TIMELINE:

The project timeline uses two different scatter chart series to display milestones above the timeline and tasks with durations below the timeline. The durations are created using X Error bars. The length of the leaderlines for the tasks can be defined by the user to show task dependencies. The vertical positions of the tasks and milestones are adjusted by the user. This may work great for some timelines, but if tasks exceed 20 it is better to opt for Gantt Chart.


SCHEDULED TIMELINE
01-06-2022
Finalizing process model and project understanding
09-06-2022
Final design documentation
29-06-2022
Beta application developed
01-07-2022
Tested application successfully deployed
01-06-2022
08-06-2022
15-06-2022
22-06-2022
29-06-2022
25-05-2022
01-07-2022

1. Introduction

1.1 Purpose

This design will detail the implementation of the requirements as defined in the Software Requirements Specification – Binder Workflow – Phase 2.

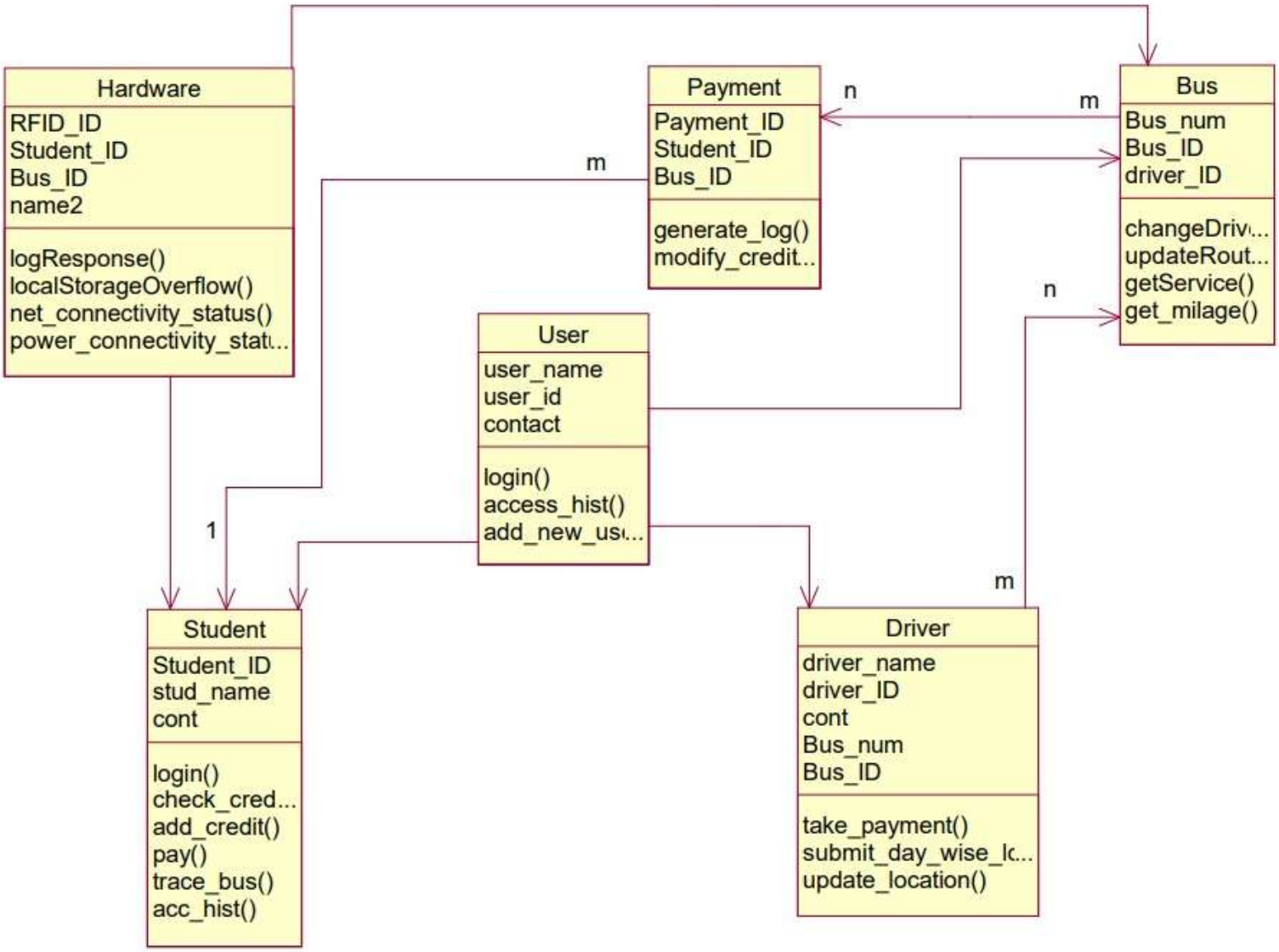
1.2 System Overview

This project extends the functionality of the VIT bus/shuttle system by making it convenient for the customers to travel freely without worrying about having the right amount of change of money with you, or worrying about when the bus will arrive at your location. The software and design requirements for the application will be discussed in this section. Metics and TaskView reports will be available for the application, but these will be implemented as the workflow reporting project and will not be included in the SDS.

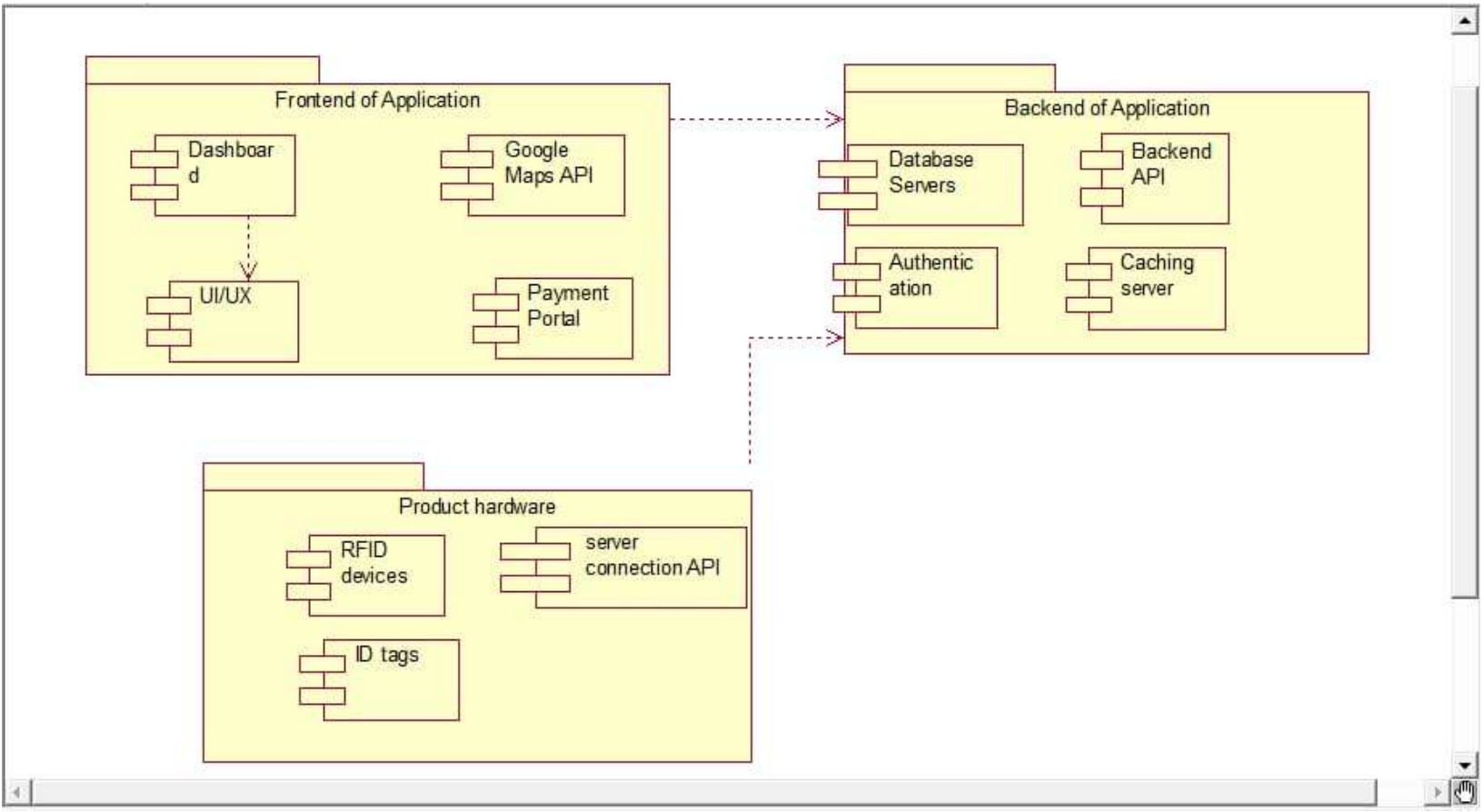
1.2.1 Workflow Reports

1) Class diagram
2) Component Diagram
3) Deployment Diagram
4) State machine diagram
5) Activity Diagram
6) Use case Diagram
7) Sequence Diagram
8) Communication/collaboration Diagram
9) DFD (Data Flow Diagram)
   a)Level 0
   b)Level 1

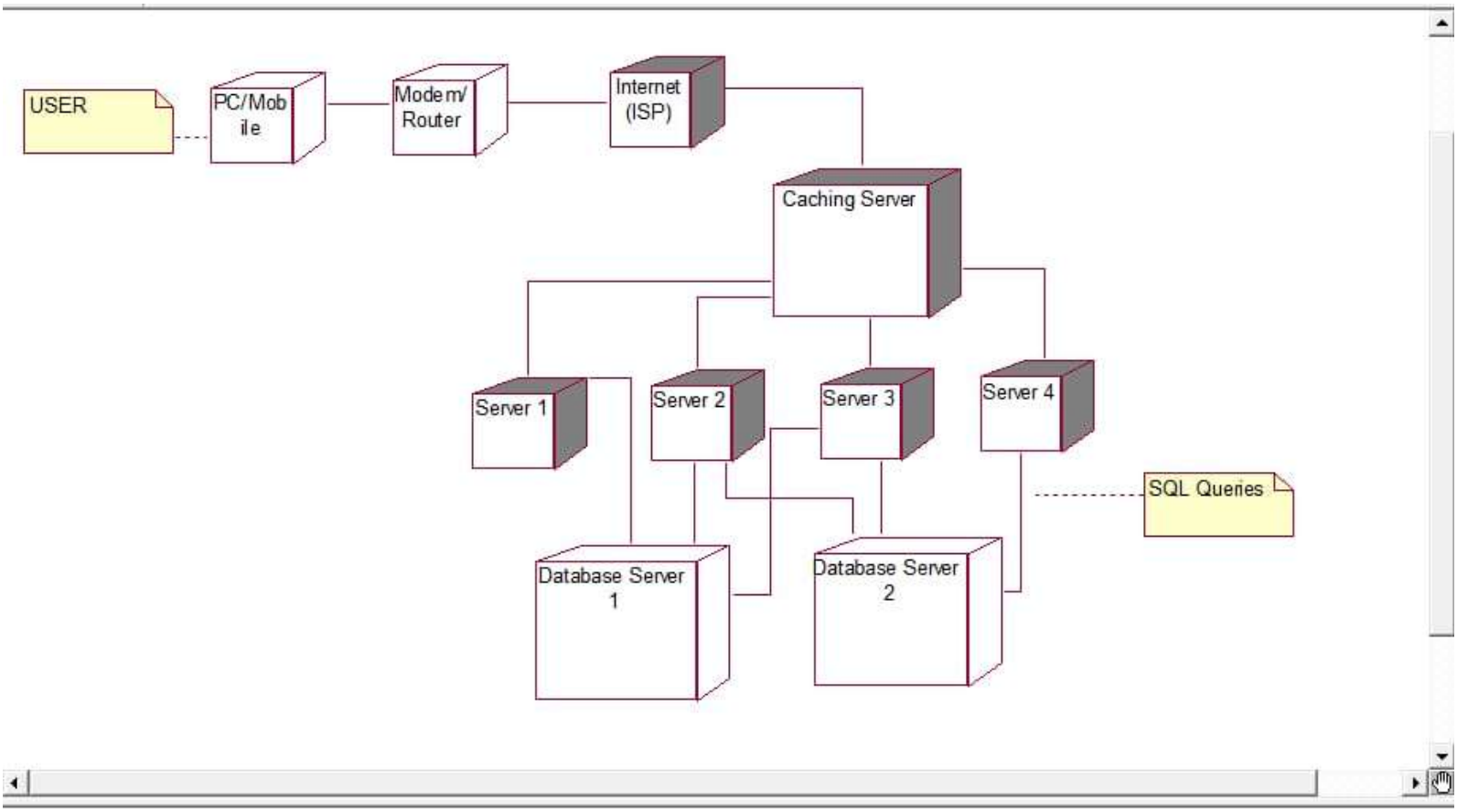
1) Class Diagram:



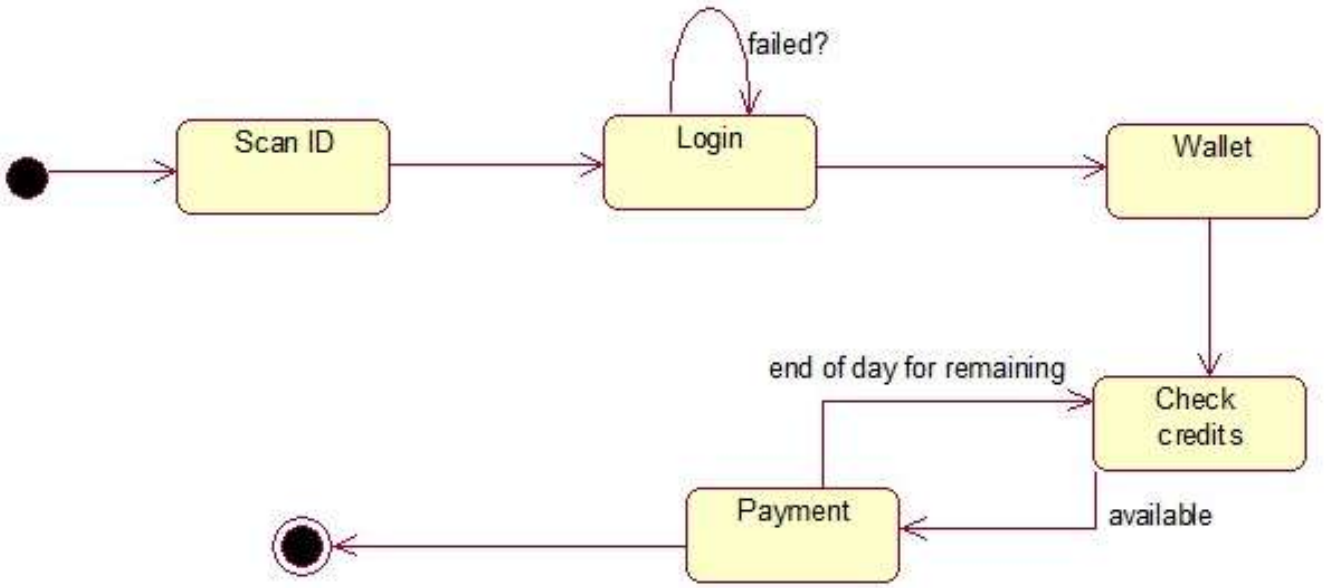
2) Component Diagram:



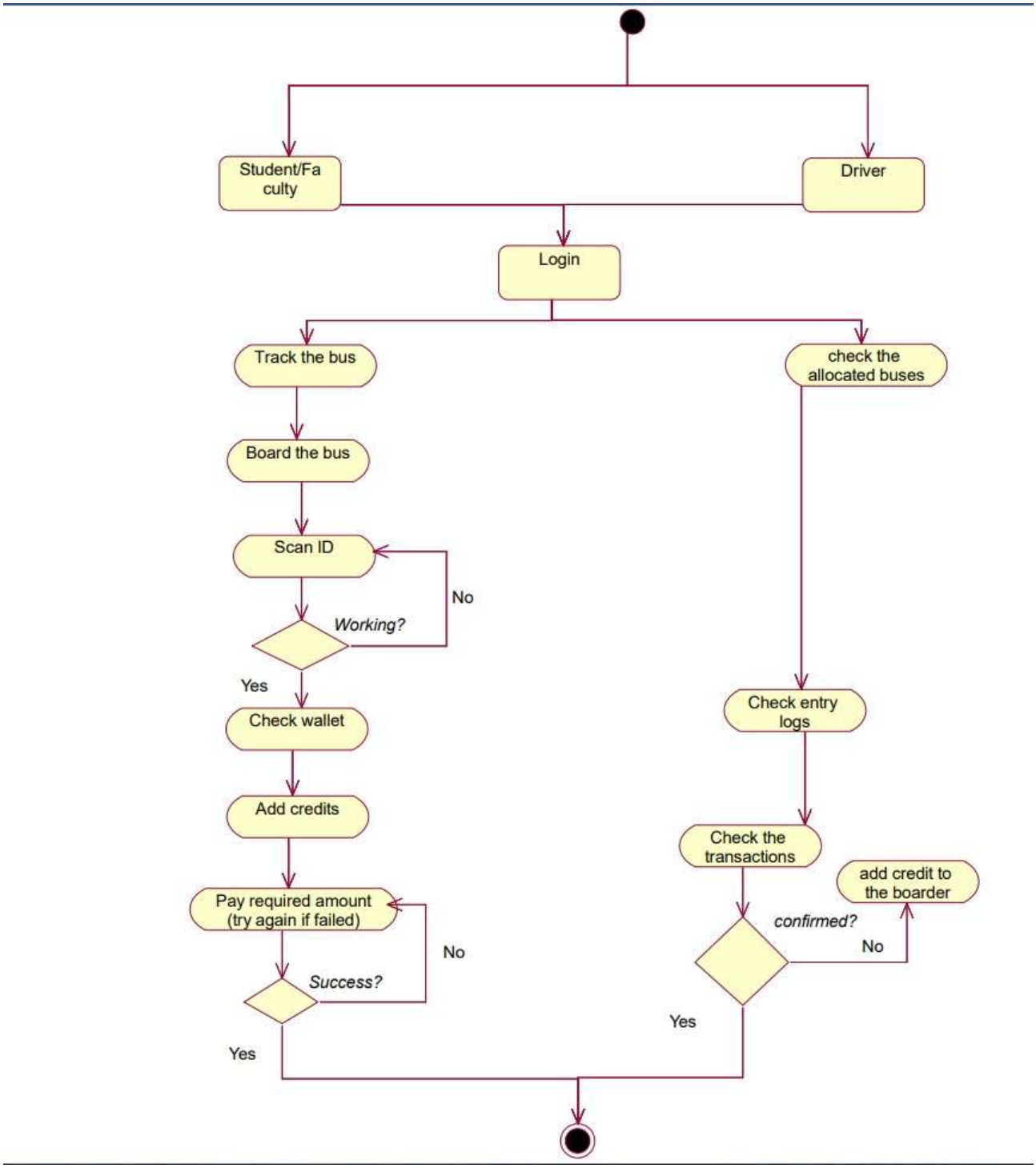
3) Deployment Diagram:



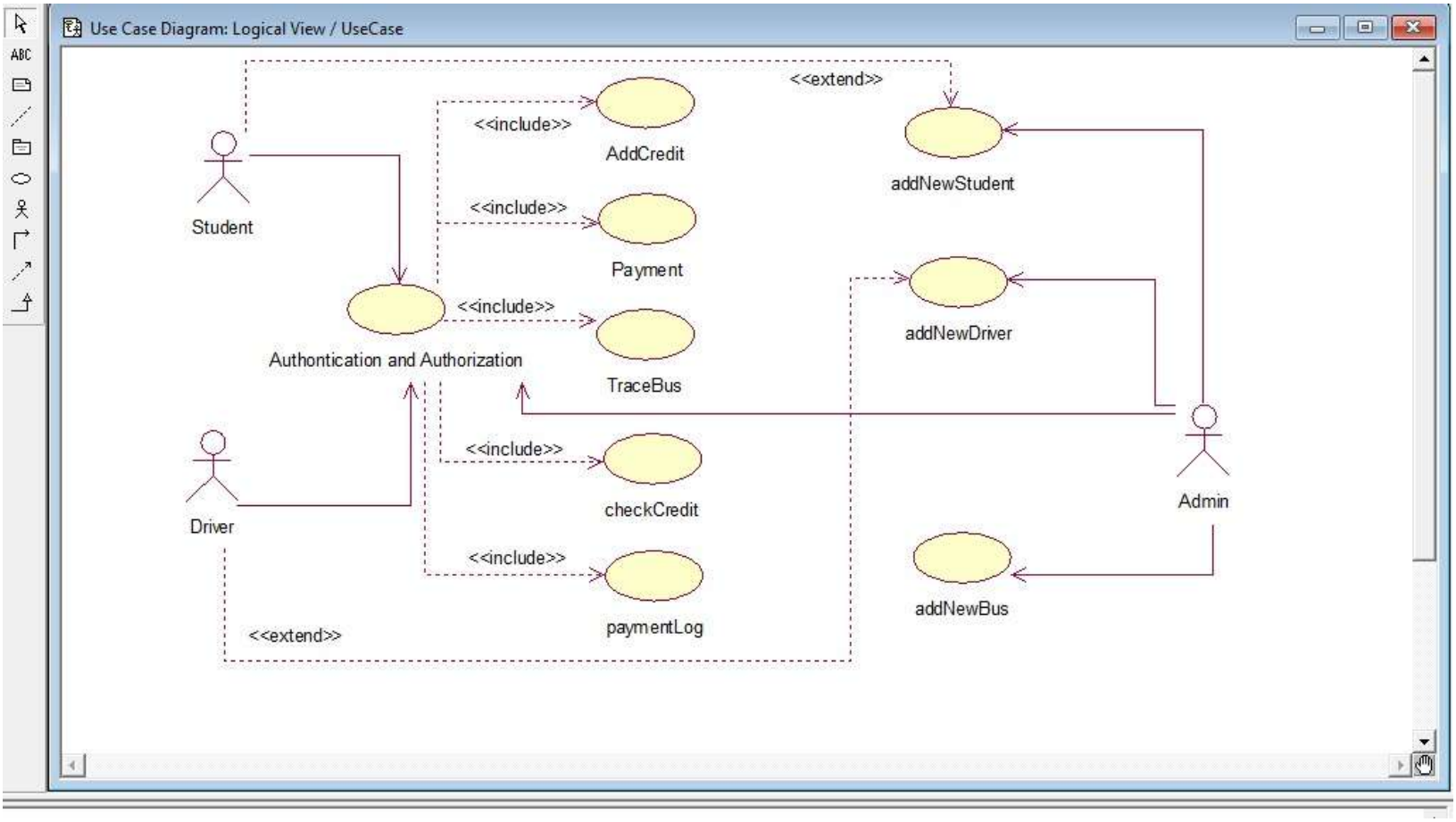
4) State Chart Diagram:



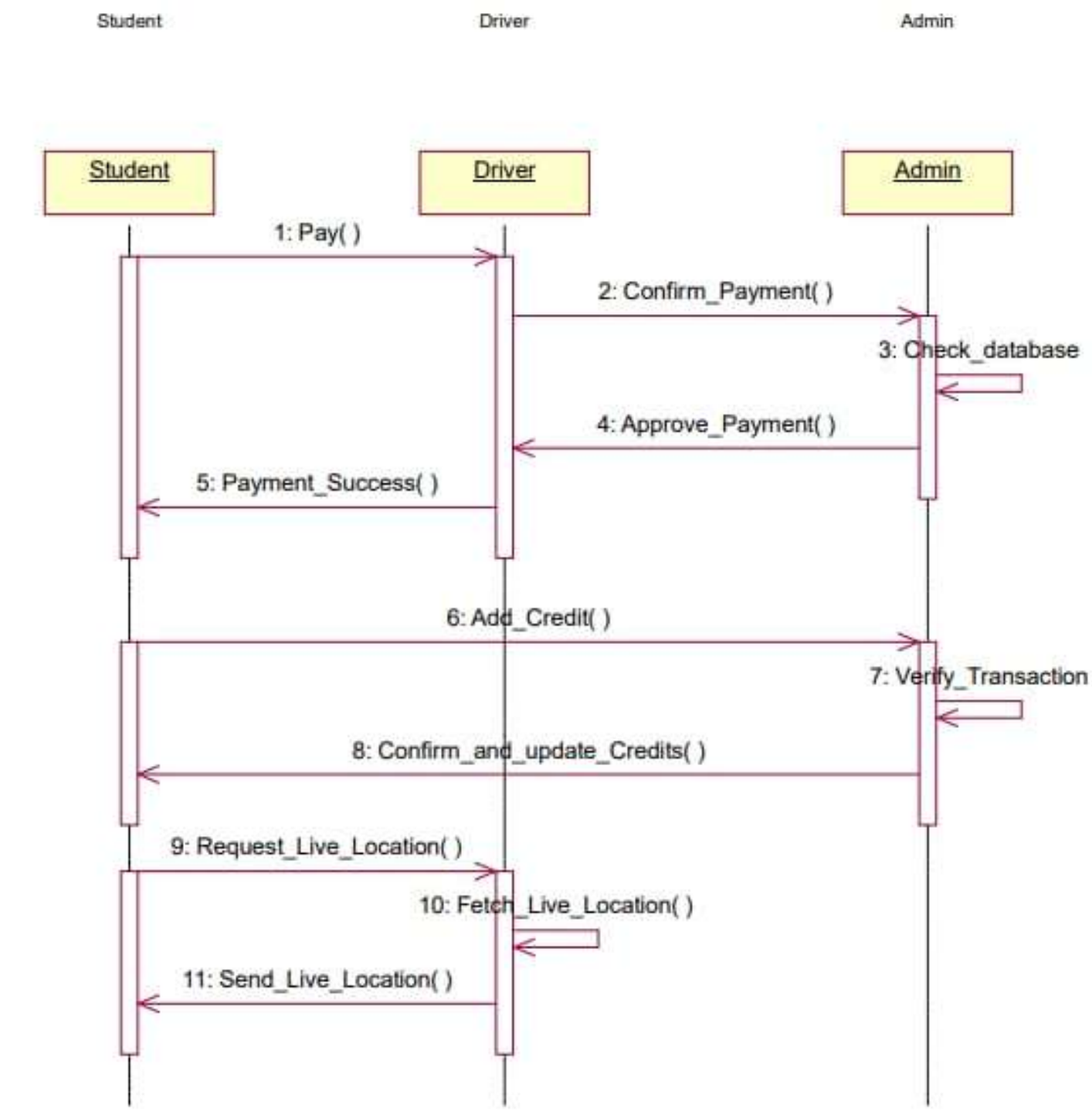
5) Activity Diagram :



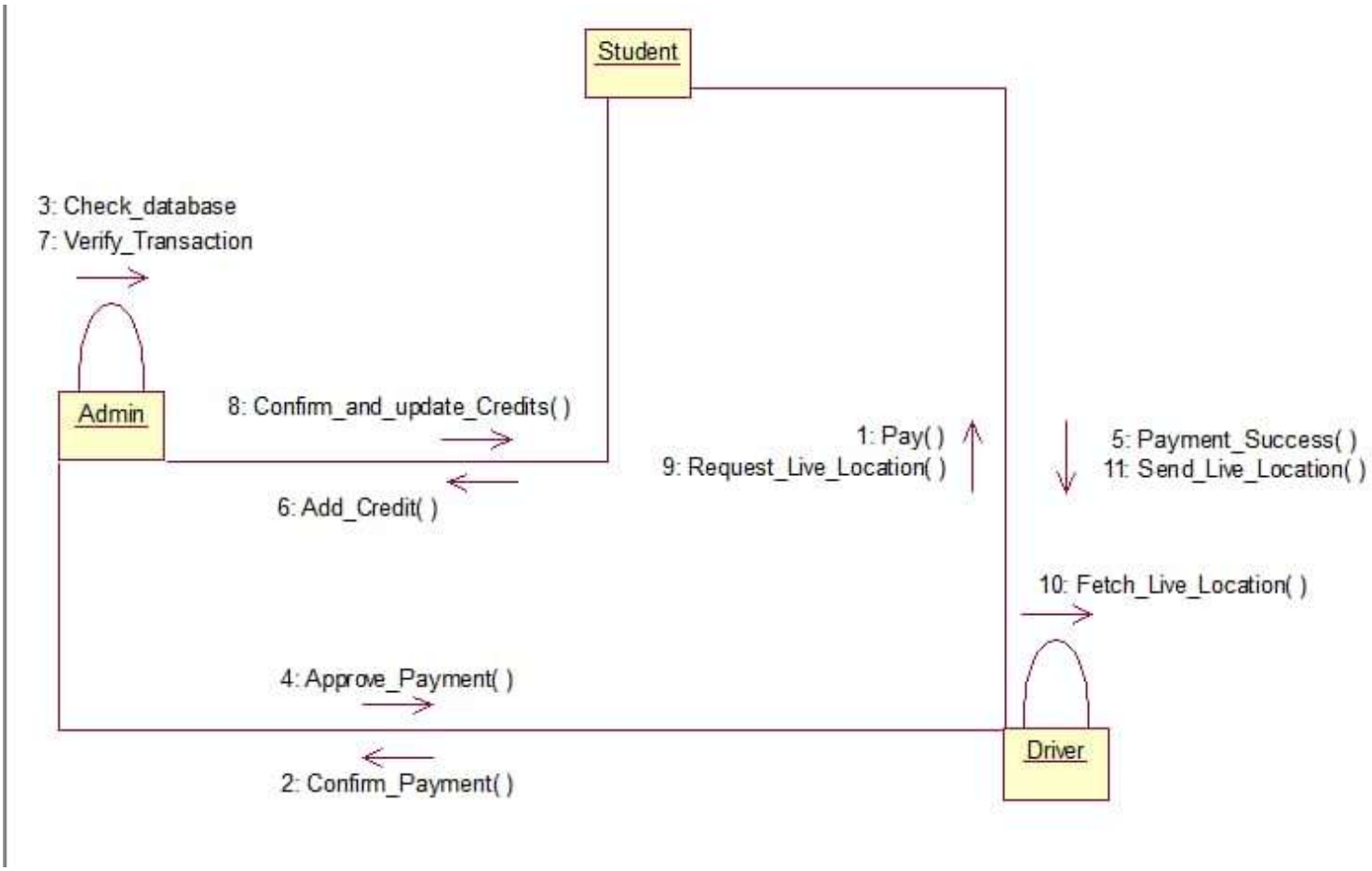
6) Use Case Diagram :



7) Sequence Diagram:

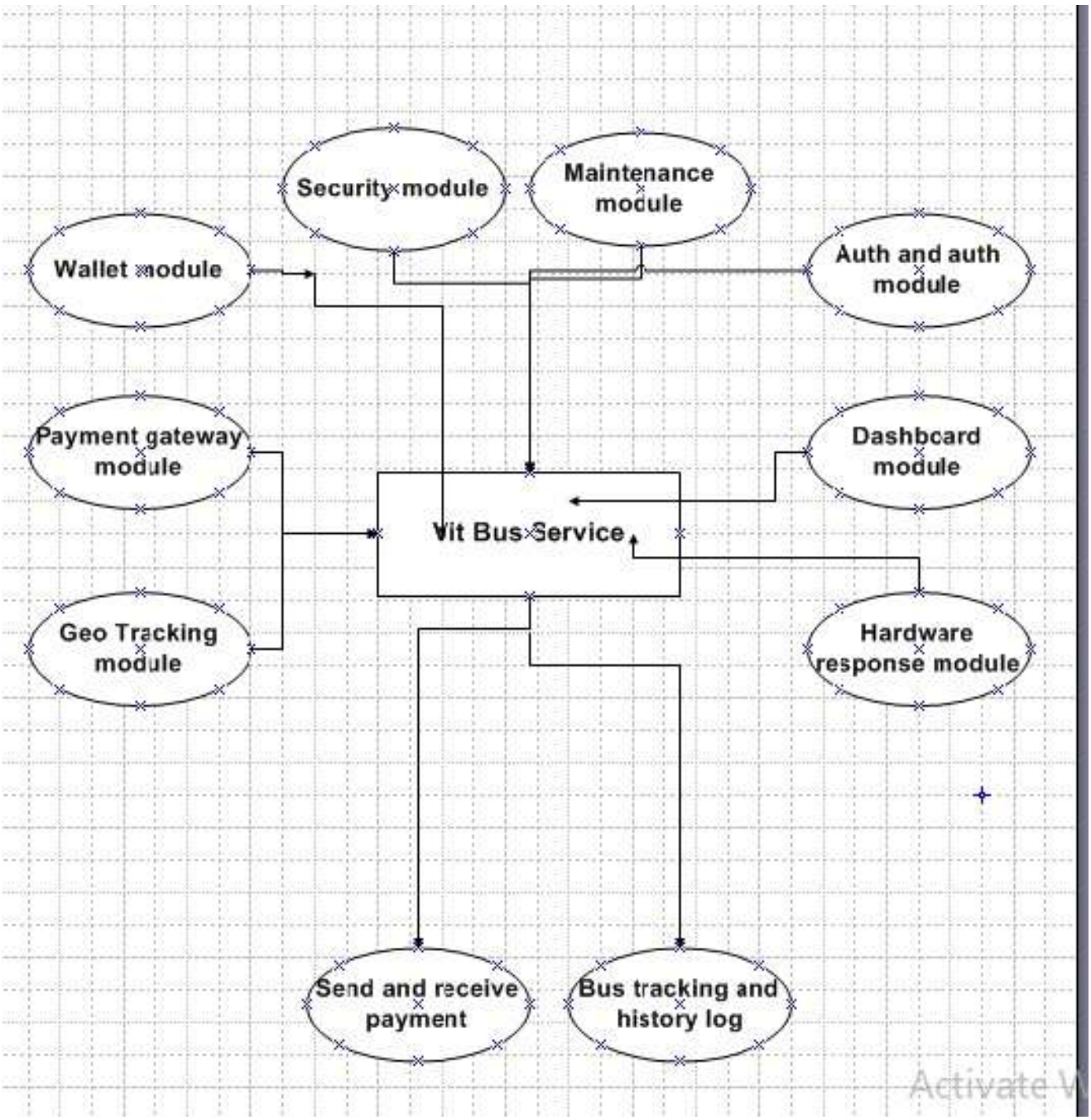


viii.) Communication/collaboration Diagram:

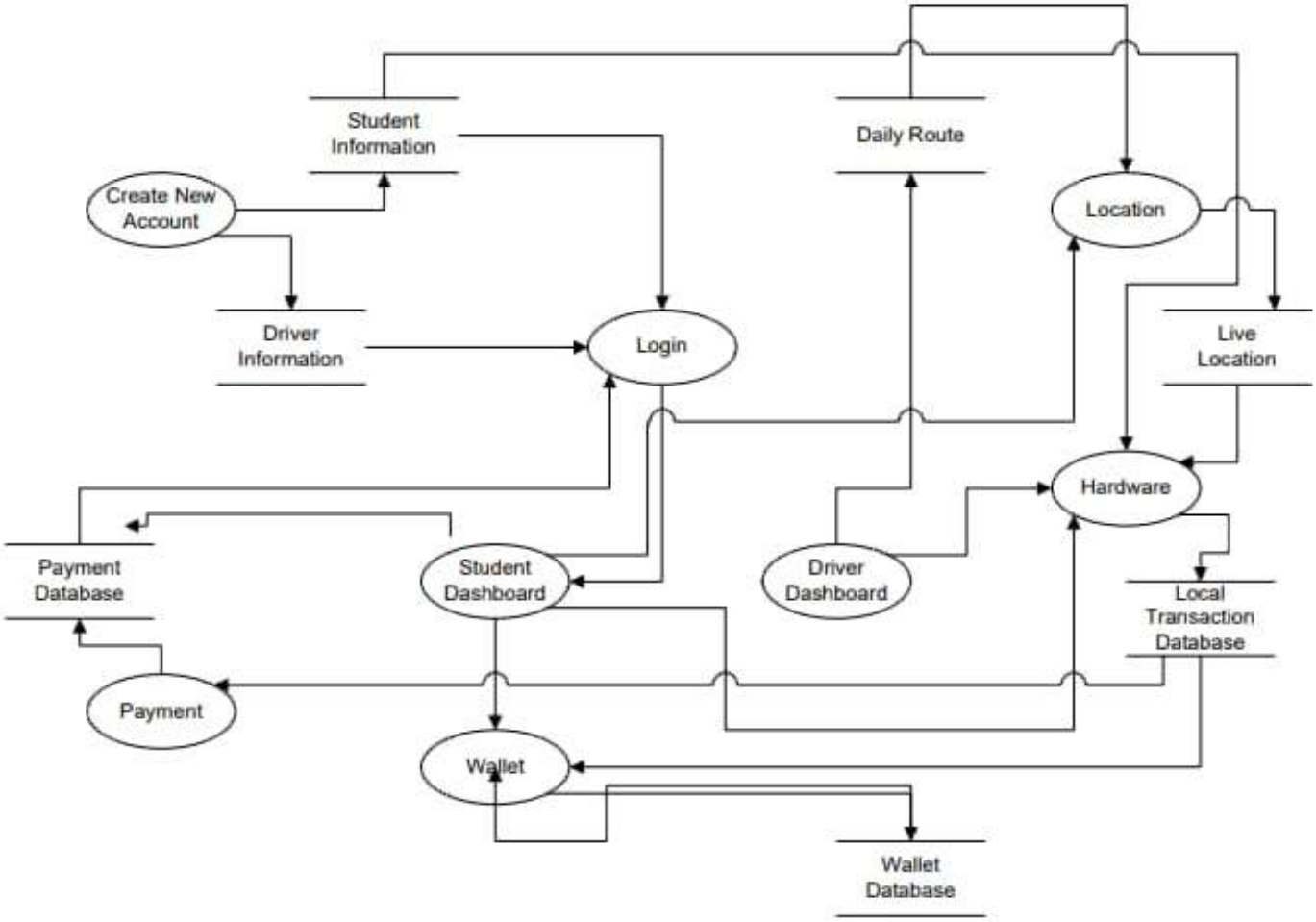


9) DFD (Data Flow Diagram)

a.) L0 :



b.) L1 :


Student Information
Daily Route
Create New Account
Location
Driver Information
Login
Live Location
Hardware
Payment Database
Student Dashboard
Driver Dashboard
Local Transaction Database
Payment
Wallet
Wallet Database

1. Hardware and Software Overview

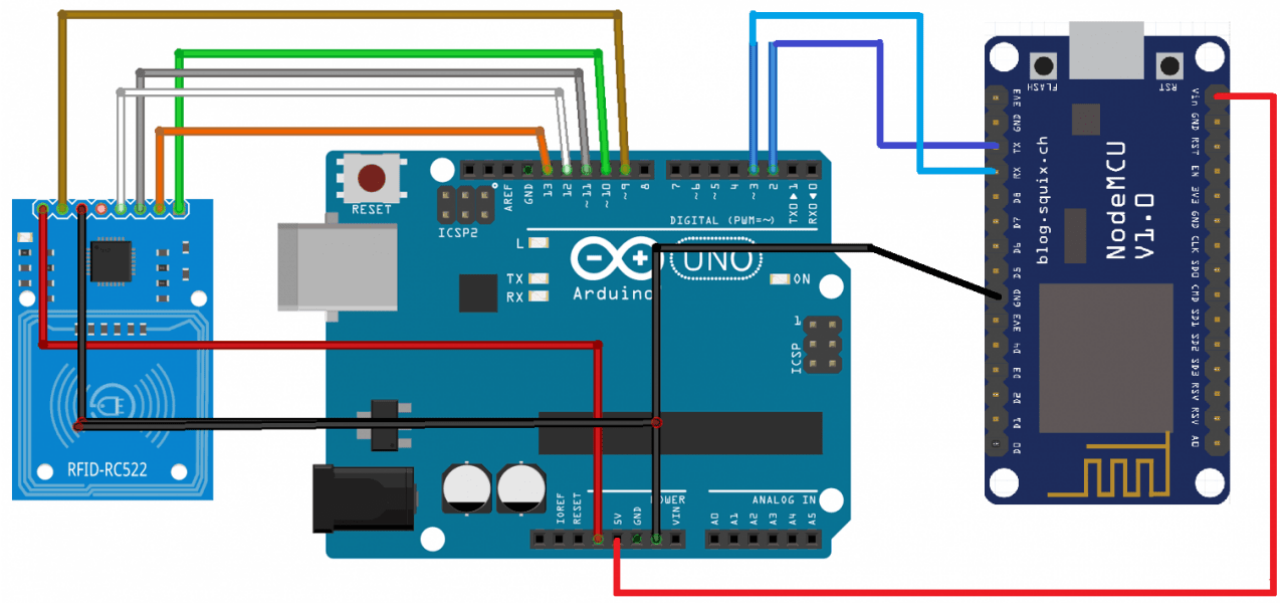
In this project, I will make IoT Based RFID Bus payment System using Arduino Node MCU ESP8266 Arduino using the MQTT broker. So we will use RFID MFRC522, Arduino Nano, and Node MCU ESP-12E Board. Arduino and RFID scanner scans the RFID cards and then logs the data to the cloud platform with the help of the ESP8266 Wi-Fi module. This information can be displayed in the dashboard and can be accessed by the required authorities to view and analyze the payments over the internet from anywhere at any time.

Components Required

S.N.	COMPONENTS	DESCRIPTION	QUANTITY
1	NodeMCU	ESP8266-12E Board	1
2	Arduino Board	Arduino UNO/Nano or any other Board	1
3	RFID Module	MFRC522 RFID SPI Module	1
4	RFID Cards	13.56 Mhz RFID Cards	5-10
5	Connecting Wires	Jumper Wires	10-20
6	Breadboard	-	1

1.1 Circuit Diagram & Connections

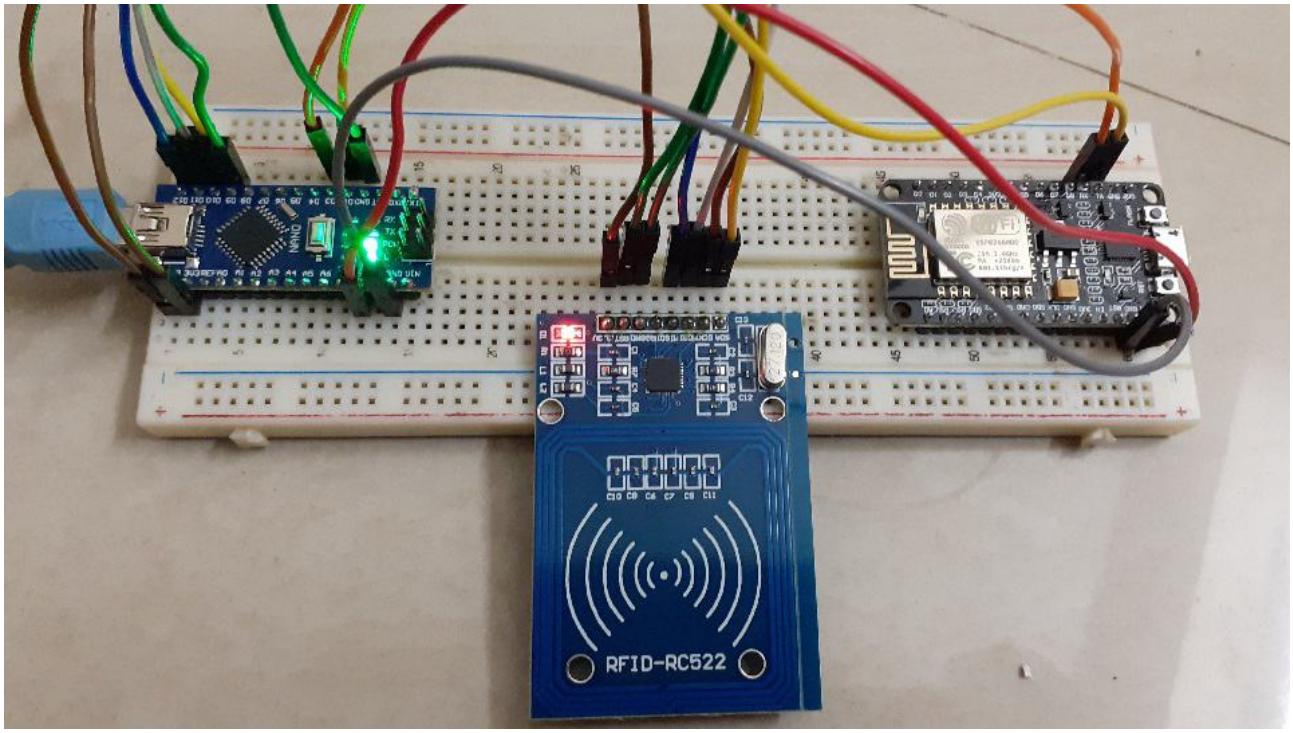
We just need Arduino Nano, NodeMCU Board, and MFRC522 RFID. So the connection diagram is exactly the same as shown in the figure.



Connect the SDA pin of RFID to Arduino Digital pin 10. Similarly Connect SCK to D13, MOSI to D11, MISO to D12, GND to GND, RST pin to D9 and supply 3.3V power using 3.3V Pins. IRQ is not connected. Similarly connect NodeMCU TX, RX pins to Arduino D2, D3 Pins respectively. Also, supply Power to Node MCU by connecting its VCC and GND Pins.

1.2 Hardware Setup

So here is our hardware complete setup. You can see Arduino Nano, Nodemcu board, and MFRC522 RFID scanner here.



All are connected as per the circuit diagram simply on a breadboard. MFRC522 is an SPI Module connected to Arduino. Nodemcu is connected via UART pins. There are few RFID cards, actually a total of 6 Cards for 6 people. All of these cards has a frequency of 13.56 MHz.

Source Code/program

The project is made using NodeMCU & Arduino Board. So, we have two codes one for arduino UNO/Nano and other for NodeMCU. You need to upload these both codes to Arduino and NodeMCU.

Code for Arduino Board

Download and install MFRC522 Library

<Code is in the appendix section 1>

Code for NodeMCU ESP-12E Board

Download and install MQTT Library.

<Code is in the appendix section 2>

2 Materials and methodology

2.1 Radio-frequency identification (RFID)

RFID has grown rapidly in recent decades [1] along with the demand from modern industry where data accuracy is required and efficiency improvements of a system, the supporting components of this electronic equipment are RFID reader and RFID Tags, where many kinds of tags have been manufactured according to industry needs. This technology has been applied to various sectors such as industry [2], airports [3], Payment monitoring systems [4–8], and with the use of IoT will be able to make this system more optimal [9– 11].

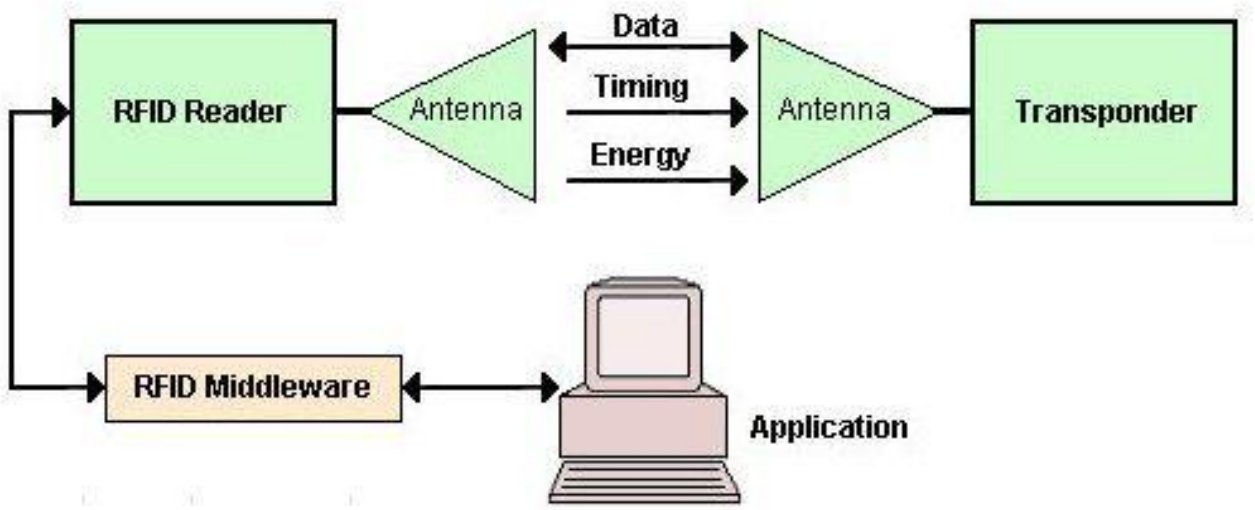


Fig. 1. Components of an RFID system [1].

Fig. 2. Type of an RFID tags.

2.2 RFID Payment technology

2.2.1 Magnetic Stripe

The use of magnetic strips can eliminate the deficiencies of barcodes, because in this technology magnetic card is required as an ID for each student, in the duplication of the magnetic card cannot be done easily because it requires a magnetic card reader and magnetic card writer, which because of the price quite expensive so it cannot be easily owned by students.

2.2.2 Biometric

The use of presences using biometric is an excellent solution, because as the ID of each student can be a finger print, retina, or face recognition. However, it has to consider the price of this equipment is very expensive.

Radio Frequency Identification (RFID) Payment monitoring system using RFID is highly recommended, because the required equipment is relatively cheap and the Payment process is very fast, so to make a monitoring system presence with this equipment can save time and budged.

2.3 Payment monitoring system with RFID

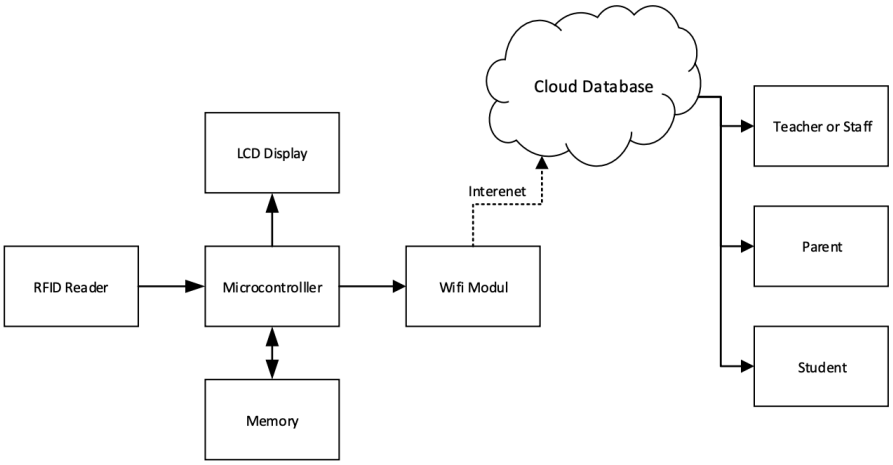


Fig. 3. Block diagram of payment system.

Block diagram in Figure 3. Show that the process of Payment is done by using RFID technology, in this system each student has an RFID Tag to do presence, student put RFID Tag near RFID Reader, then ID result from RFID reader will be sent to microcontroller and compare it with the student data stored in memory, memory serves to store the data of the student's name of the course, if the data ID is a lecture participant then the student's name will be displayed on the LCD Display as well if the student data is not listed it will be informed through LCD Display that the student unregistered, using Wi-Fi module microcontroller can send student Payment data to cloud database by using internet network, data already accommodated in cloud database can be seen in real time by teacher, student and even parent, so that student presences can be monitored from anywhere in real time using Internet of Things (IoT).

2.4 Flowchart presence using IoT-based RFID

Presence process as illustrated in the flowchart in Figure 4 starting from the RFID Tag scanning process using RFID Reader, the data obtained will be compared to the database, if the data match the database then the presence data will be stored in the cloud database, but if the data is not suitable then will be asked to scanning again RFID Tags, data stored in the cloud database is Student ID data, date and time of Payment, courses and on what week of Payment.

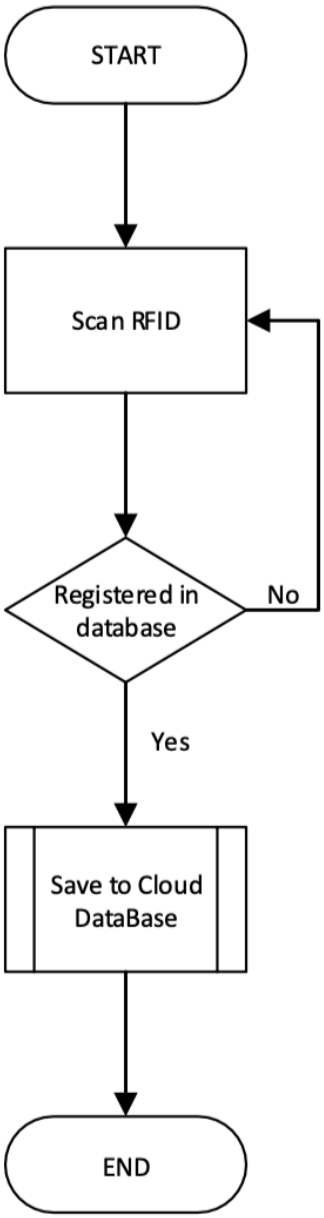


Fig. 4. Flowchart of RFID scan.

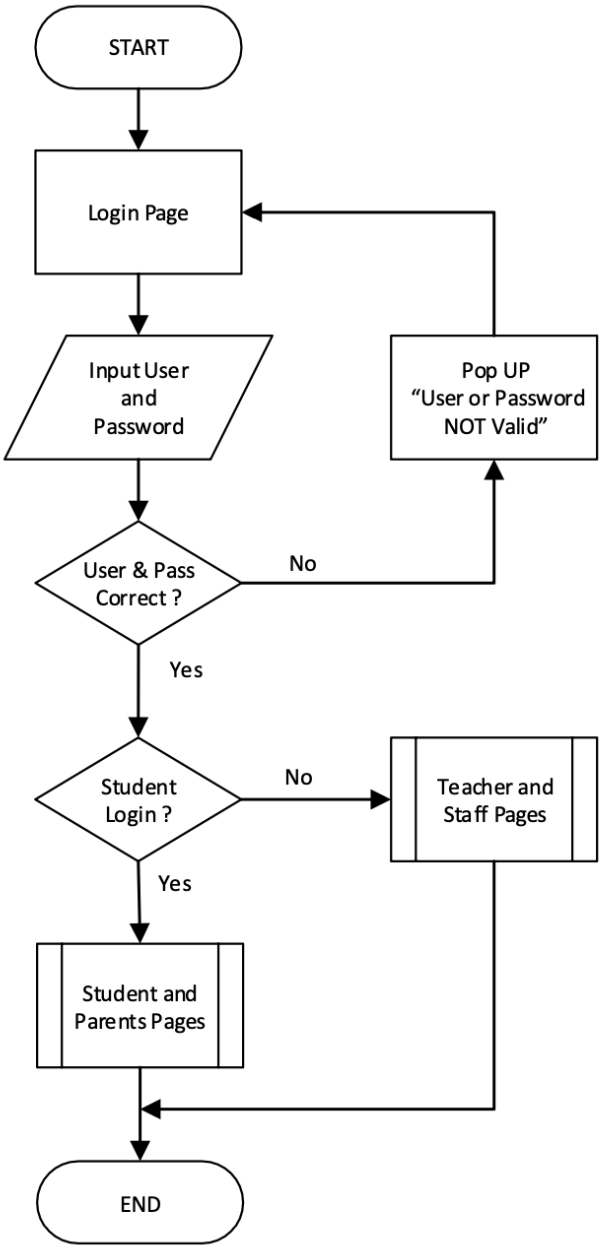
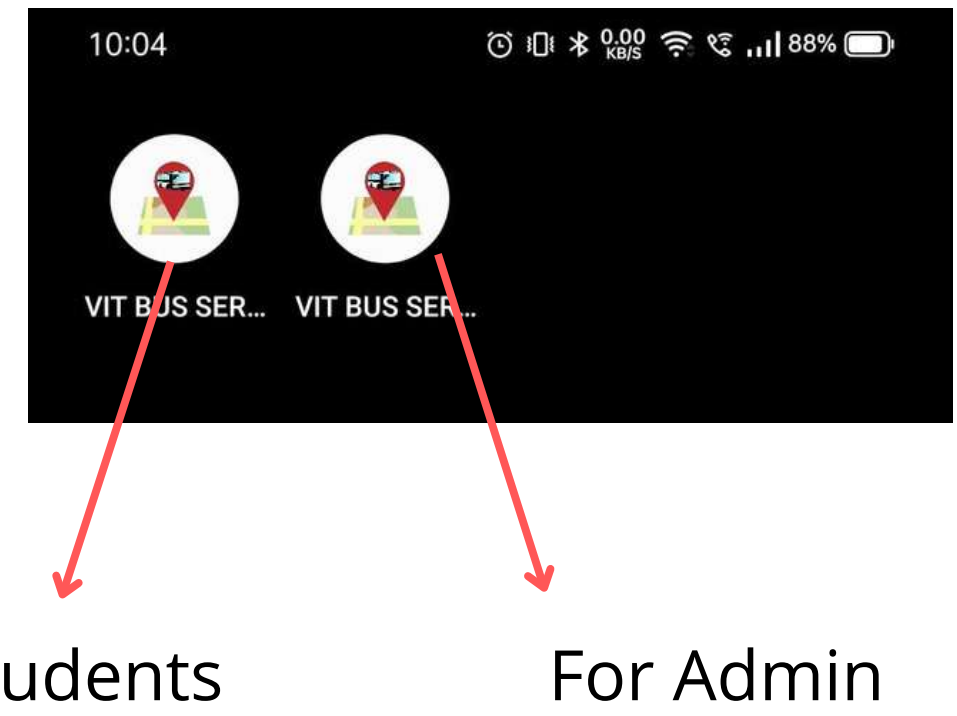


Fig. 5. Flowchart system

To display the results of presence data the user must login first, the login process determines the view that can be seen later, the user is divided into two kinds of login as a teacher or administration, and login as a student or parent, Payment data that has been stored in the cloud database can be displayed through two types of users, in Figure 6 is the output display for teachers and administration, where in this view Payment data from students are grouped by subject and class, while in Figure 7 is the output display for students and parents, it only displays data from students who login only, the name of the students displayed depending on the user's students and parents, so that each student and parents can only see their own presence data only. Figure 8 shows that the presence with RFID is faster than manual.

UI / UX Prototyping and Designing of the mobile application

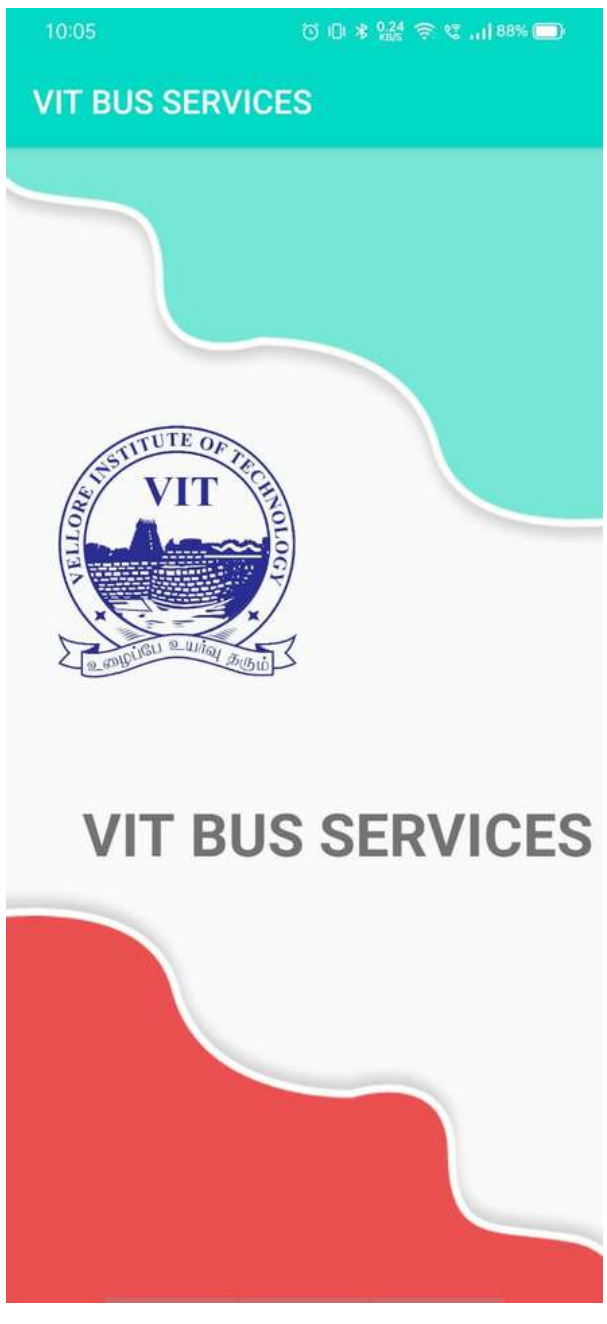
App Icon



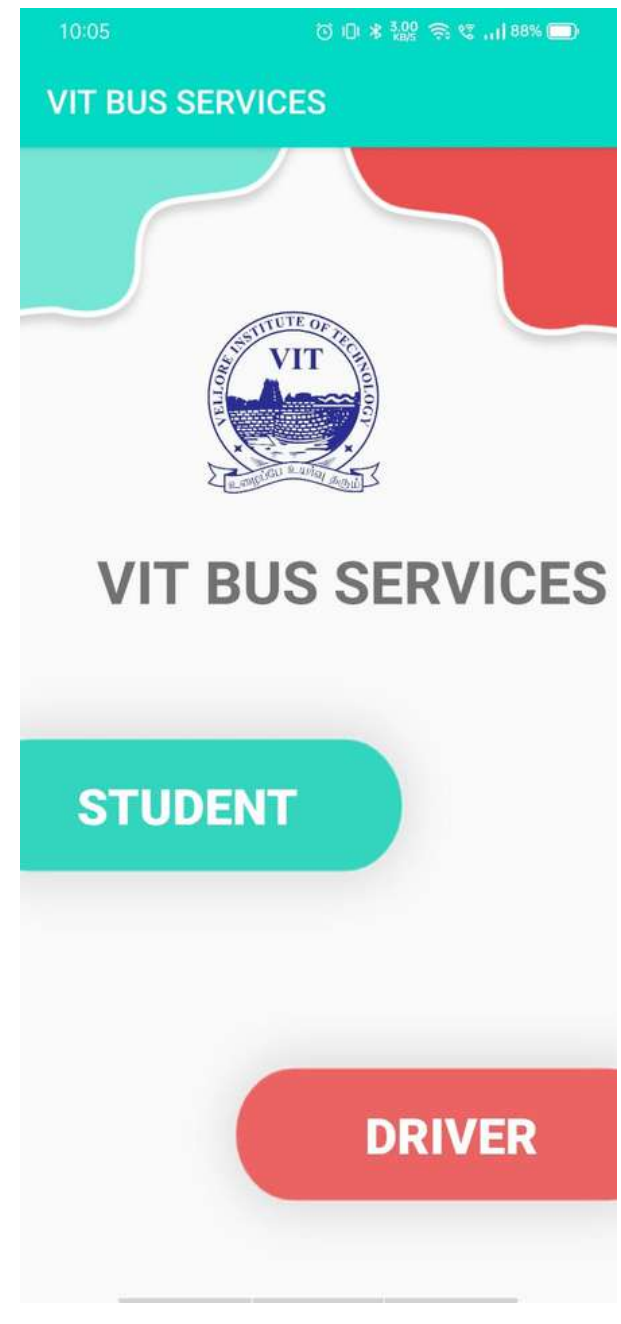
For students and drivers

For Admin

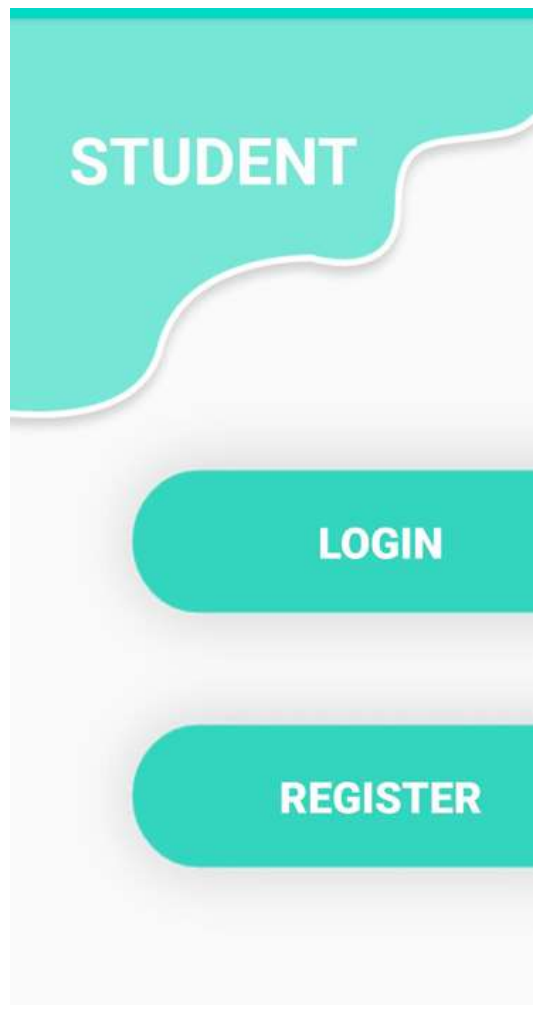
Home page



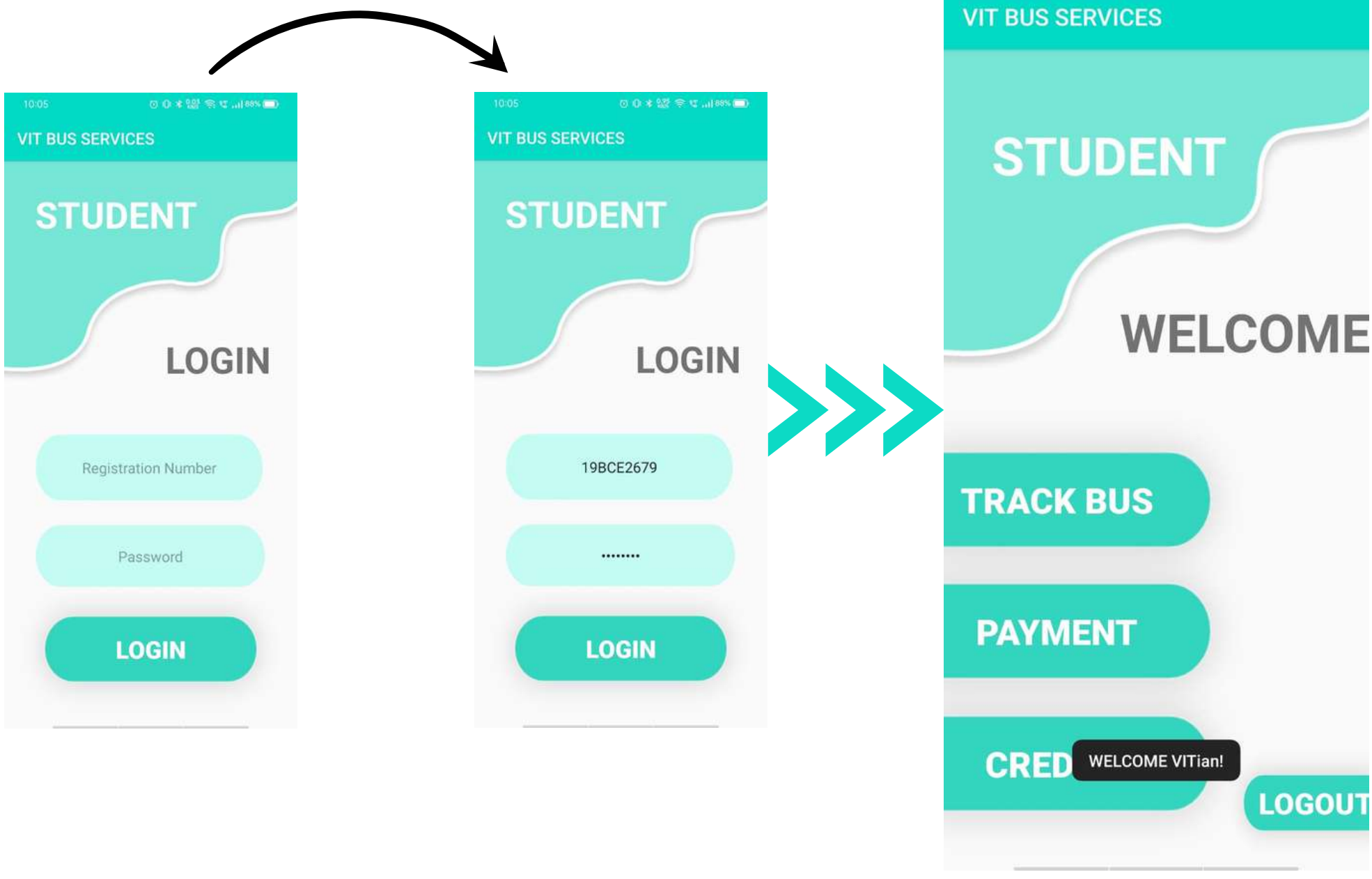
Transition to selection section



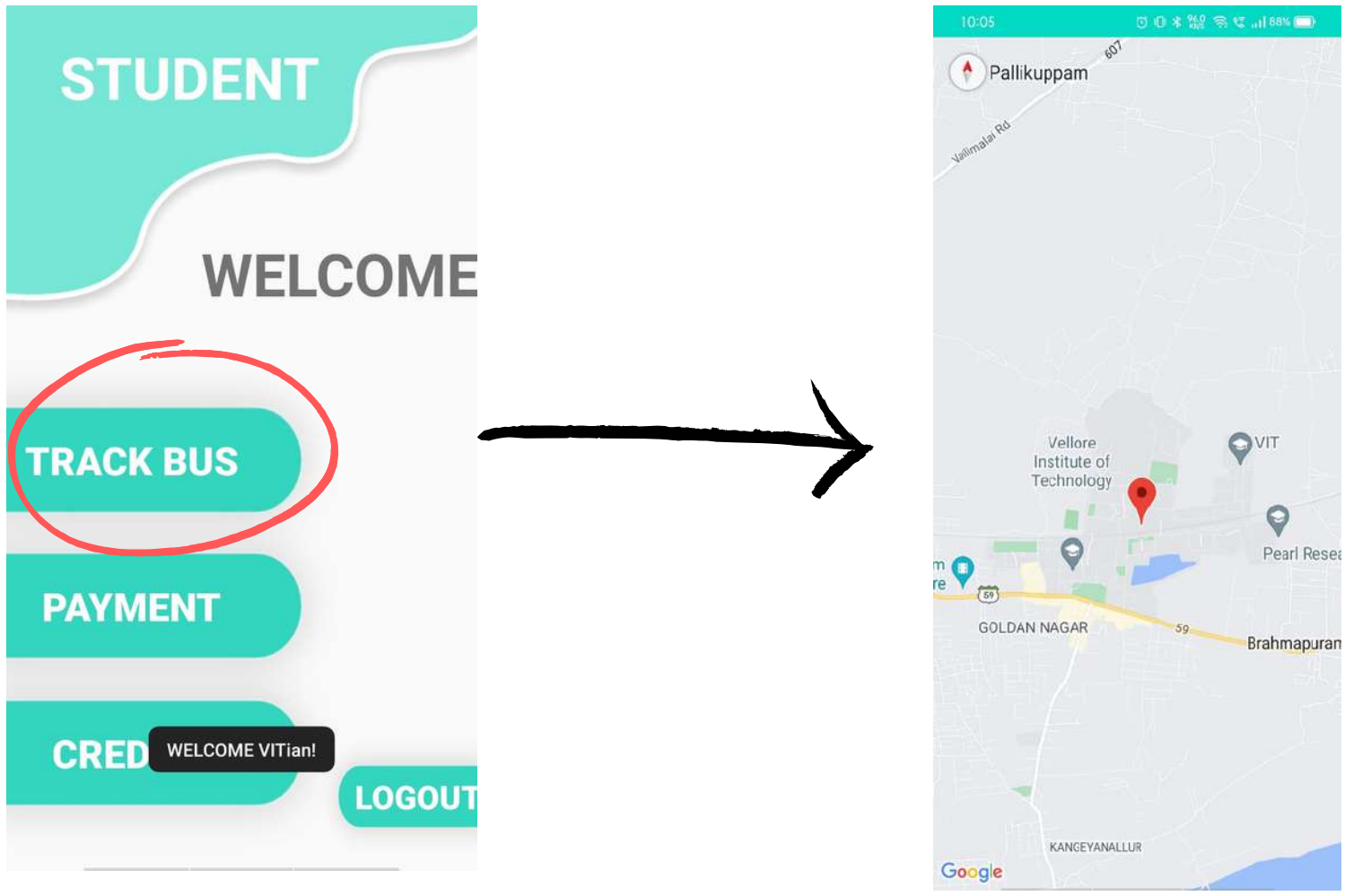
Student page



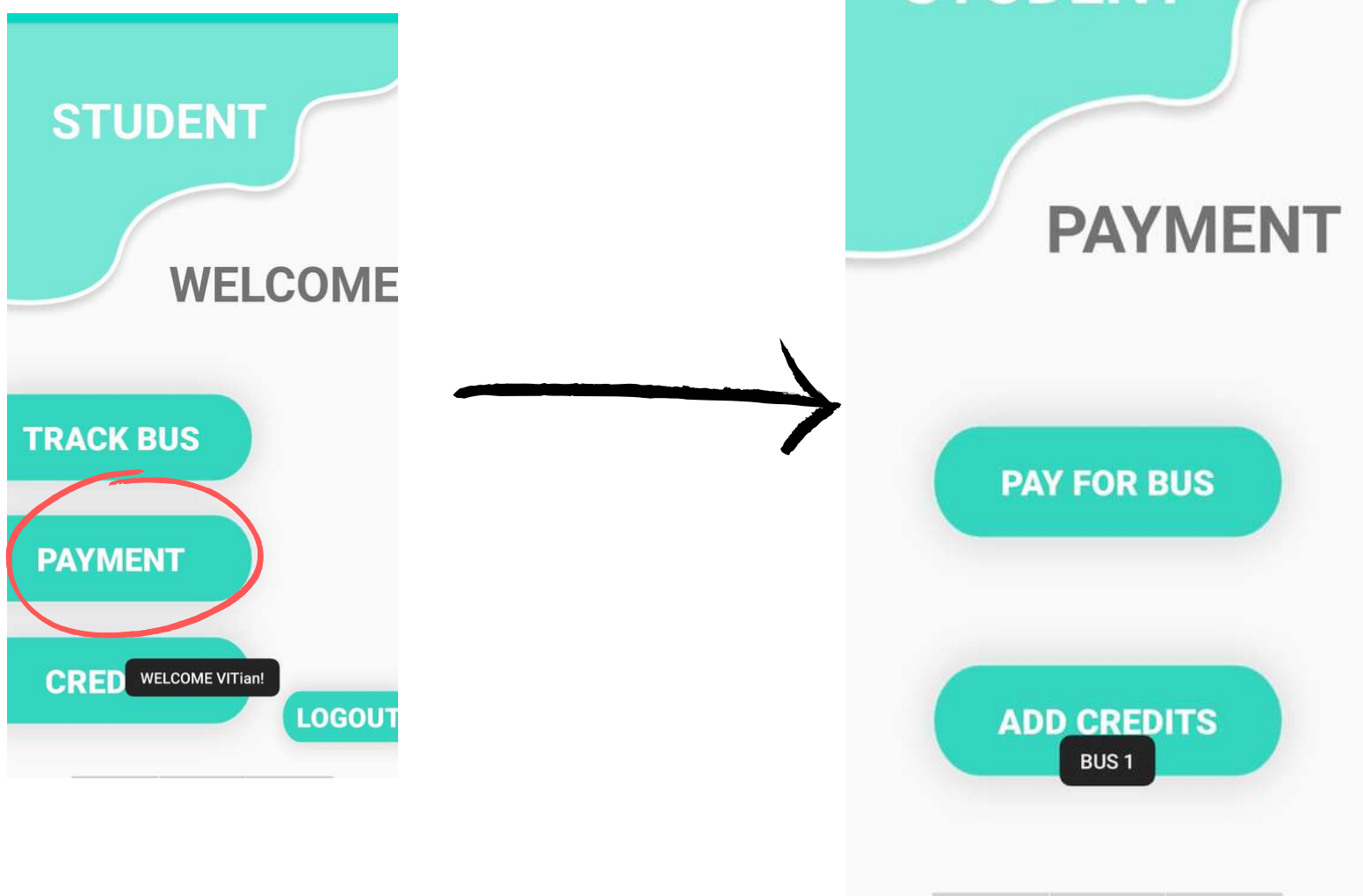
Student Login page



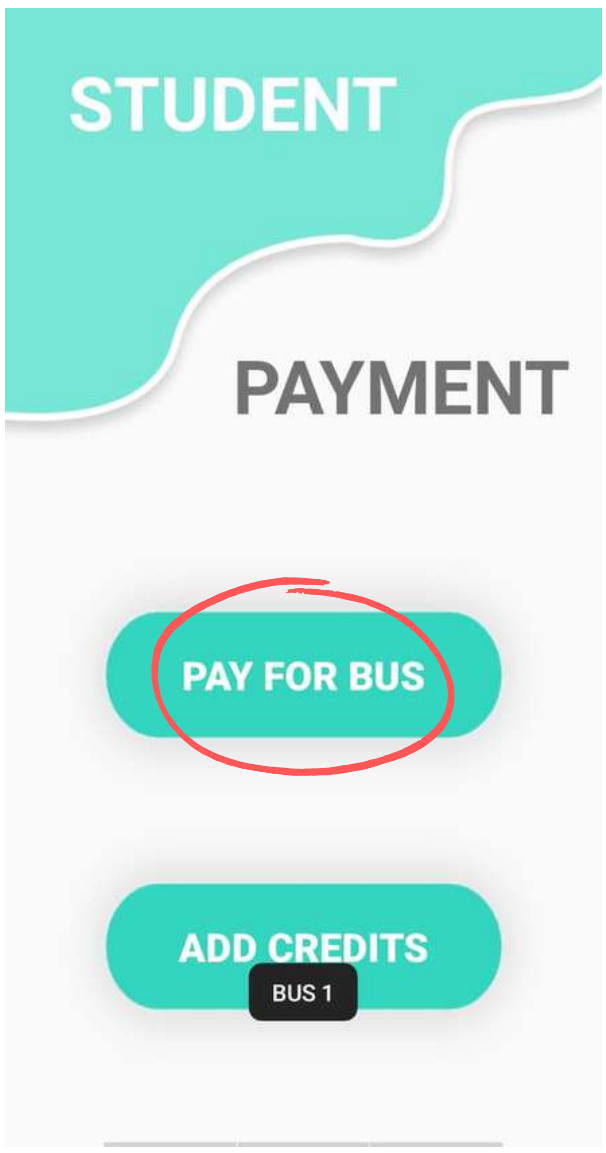
Student Bus Tracking



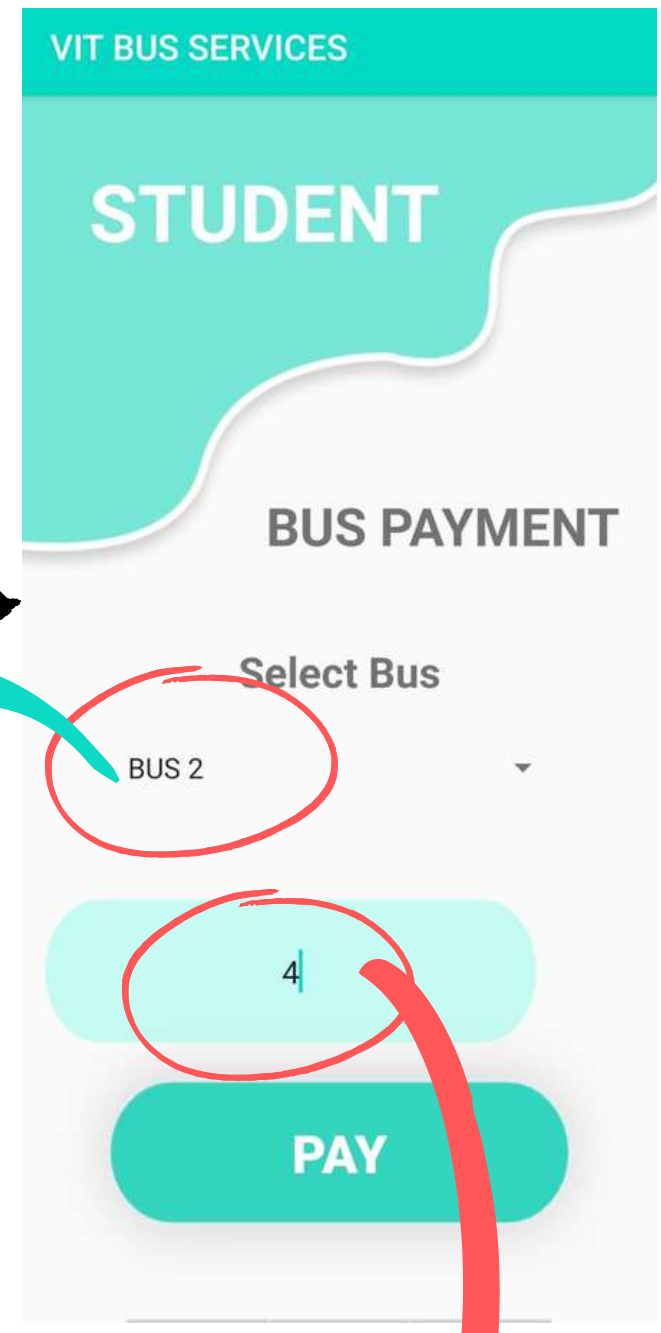
Student Payment to bus



Student bus Selection and pay

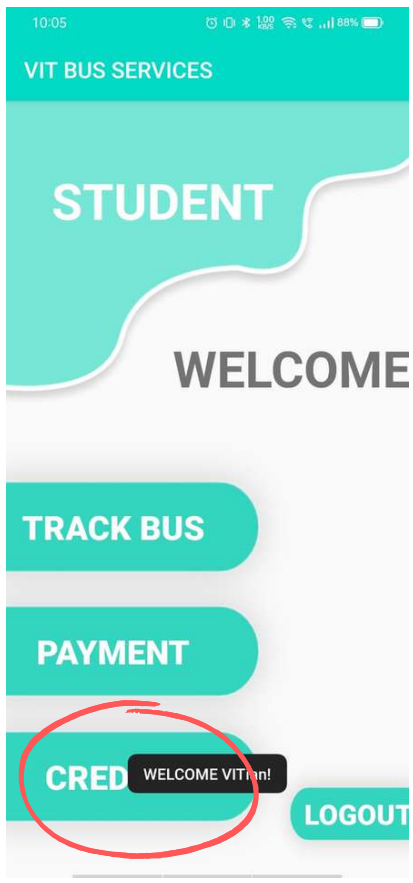


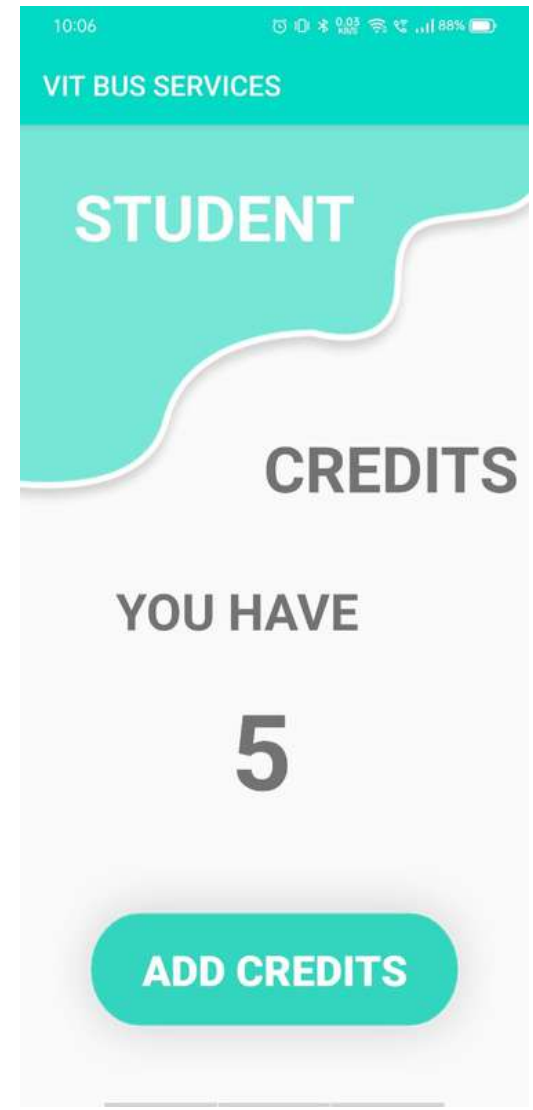
Select the Bus number



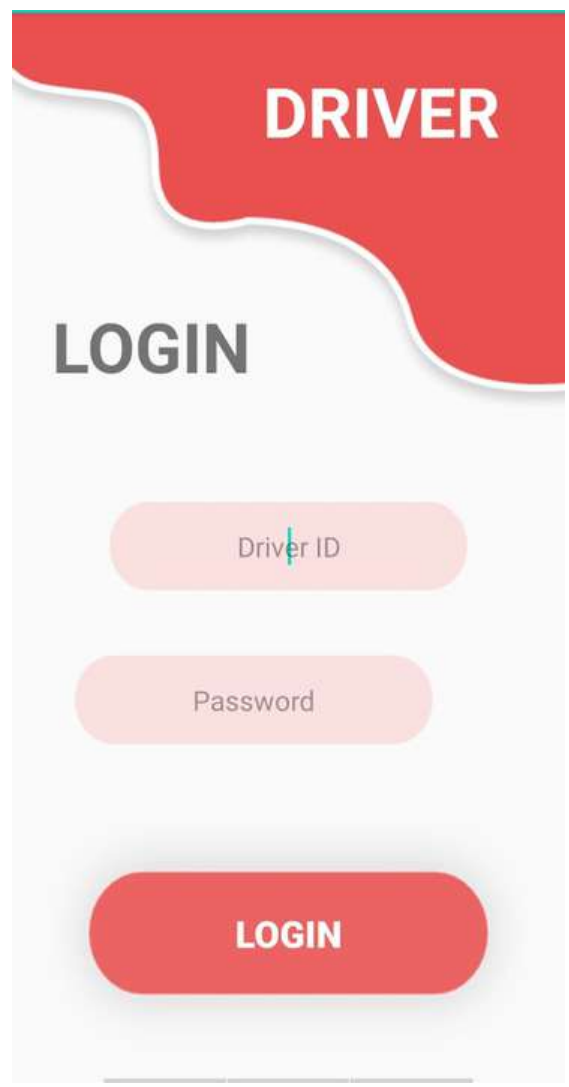
Number of credits to pay

Student check and add credits



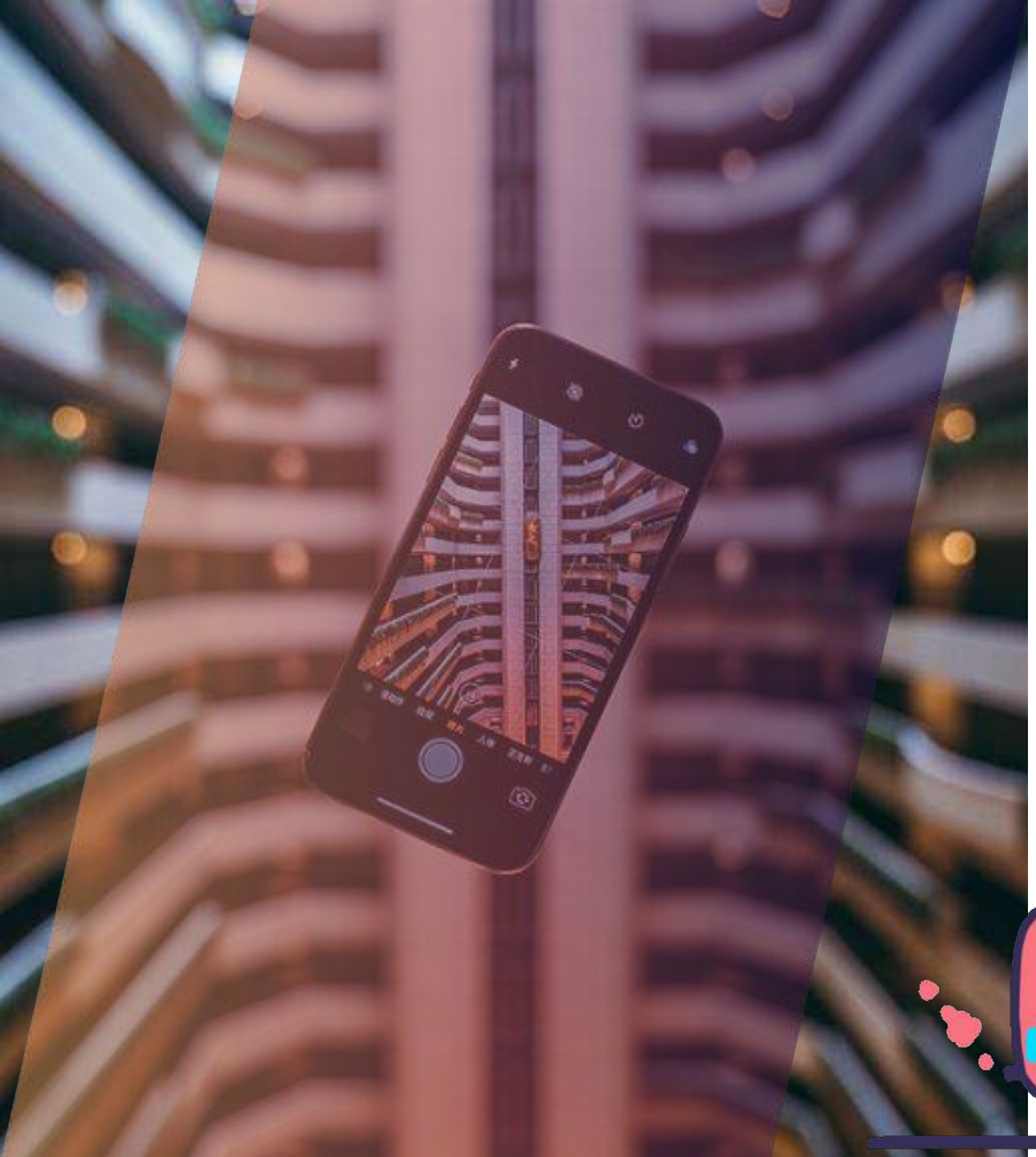


Drivers section



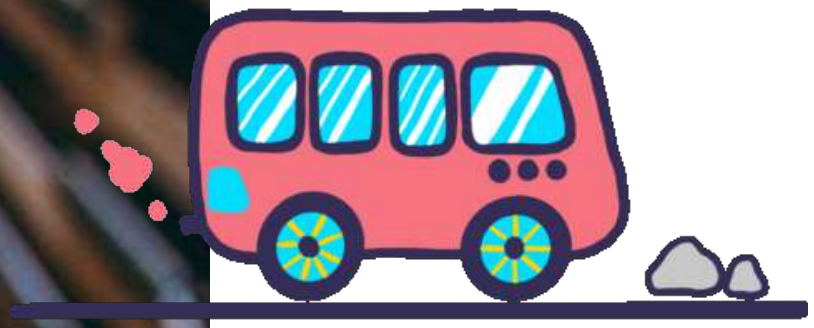
VIT BUS SERVICES

Software Engineering
PROJECT



PROJECT DEFINITION

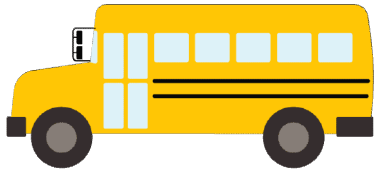
Basic idea behind the project



PROJECT MOTIVATION

ABSTRACT

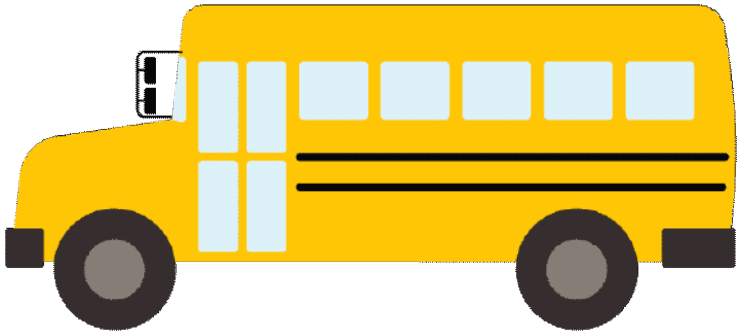
VIT Vellore has a huge campus with many academic blocks and hostels. Students must rush from building to building for classes within 5-10 min. This was solved by having shuttle bus services connecting all buildings and hostels of the campus. The one major problem that this brought up was the bus payment method. , it charges 15 rupees per ride. Most of the time, either the driver or the student doesn't have change and it takes up a lot time for the money exchange. Sometimes, the drivers don't accept online payments and there is network problem too. This issue is what I have tried solving in my own way.

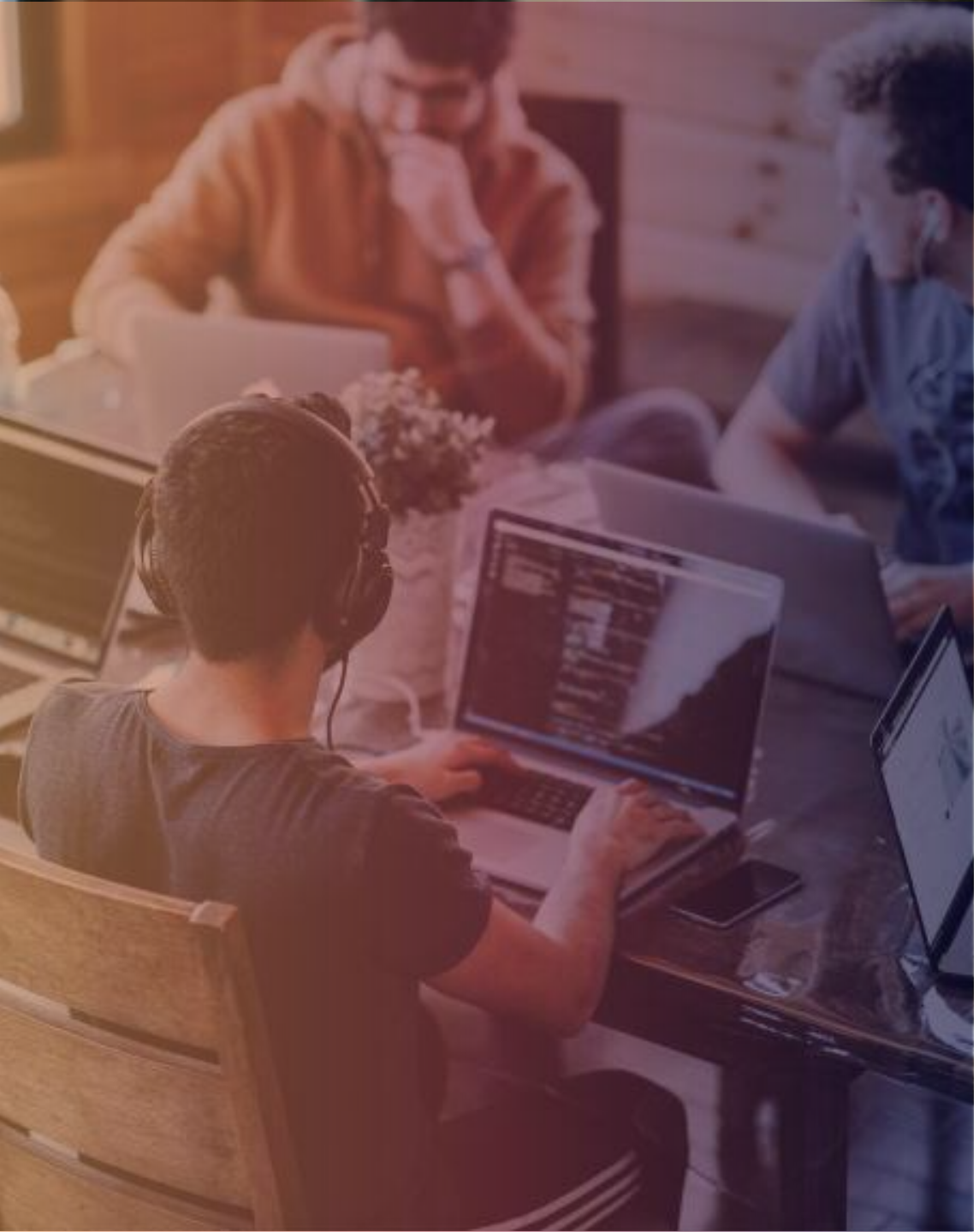


? ?

? SOLUTION ?

Solved through Software Engg. concepts





APPROACH TO SOLVE

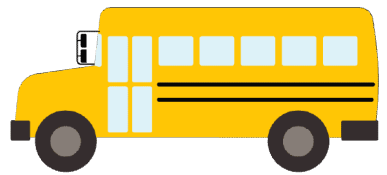
Overview of the solution

Since the payment service must be used offline and we can't add any electronics to the bus so, I came up with the idea to use the driver's phone to scan the code from the student to make the payment process. This can be done using a QR code scanner through an app.

We will scan the code which will be the QR for the registration number of that student (similar to what we have on student ID card).

The student riding will just have to show the QR code to the driver and driver will scan the code, which will be added to the database (offline).

At the end of the day as the driver connect the phone to the internet the database will be transferred to the main server and the online bill will be generated for the students who took ride on that particular day.



CODE

```
// isempty

bool isempty() {
   if (front==-1 && rear==-1)
      return true;
   else
      return false;
}

//isfull

bool isfull() {
   if ((rear+1)%500==front)
      return true;
   else
      return false;
}

//Enqueue

void enqueue(int index,string regNo,int credit) {
   if (isfull())
   {
      return;  //This will terminate the function without any changes

   }
   else if (isempty()){
      rear=0;
      front=0;
      busTotalCreditColloction+=credit;
      arr[rear].regNo=regNo;
      arr[rear].credit=credit;
      itemcount++;

   }

   else{
      busTotalCreditColloction+=credit;
      rear=(rear+1)%500;
      arr[rear].regNo=regNo;
      arr[rear].credit=credit;
      itemcount++;
   }
}

      //dequeue

void dequeue() {
   if (isempty()){

   }
   else if(rear==front){  //This is true only when both front and rear is equal to 0...
      //rear==0
      //front==0
      //raer==front
      //when we remove the element we have to set that place as 0
      rear =-1;
      front=-1; //after removing it again go back to one place back
      itemcount--;


   }
   else
   {

      front=(front+1)%n;
      itemcount--;

   }
}

   //count
   // It have to be changed for circular queue
   int count() {
      return itemcount;

   }
};


busCirQueue queueBus[6];



int noOfReg=0; // No of students registered on the app
bool online=false;   // status of driver's phone


string * regList=new string[1000]; //list of student registeration number for easy search
string * nameList=new string[1000];  //list of student name for easy search
string * driverIDList=new string[10];  //list of driver emp number for easy search



// payment gateway function


bool payment(int amo){
   cout<<endl;
   cout<<"Pay "<<amo<<endl;
   char con='Y';
   cout<<"Press Y to confirm or N to deny"<<endl;
   cin>>con;
   cout<<endl;
   if(con=='Y'|| con=='y'){
      cout<<"Payment successful"<<endl;
```

VIT BUS SERVICES CODE SAMPLE OUTPUTS

Apendix

Section 1

```
#include <SPI.h>
#include <MFRC522.h>
#include "SoftwareSerial.h"
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.
SoftwareSerial ser(2,3); // RX, TX
void setup()
{
  Serial.begin(9600);   // Initiate a serial communication
  ser.begin (115200);
  SPI.begin();      // Initiate  SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522
  Serial.println("Put RFID Card to Scan...");
  Serial.println();

}
void loop()
{
  // Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
  }
  //Show UID on serial monitor

  String content= "";
  byte letter;
```

```
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {

     content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  Serial.println();
  Serial.print("User No./Name:    ");
  content.toUpperCase();
  if (content.substring(1) == "E9 9C B0 E3" ) //change here the UID of the card/
cards that you want to give access
  {
    Serial.println("1-Alex Newton");
    ser.write(1);
    Serial.println();

    delay(3000);
  }
   if (content.substring(1) ==  "01 15 C9 1F" ) //change here the UID of the
card/cards that you want to give access
  {
    Serial.println("2-Tracy Witney");
    ser.write(2);
    Serial.println();

    delay(3000);
  }
   if (content.substring(1) == "B9 BF 62 1B" ) //change here the UID of the
card/cards that you want to give access
  {
    Serial.println("3-Lucinda Ambrey");
    ser.write(3);
    Serial.println();

    delay(3000);
  }
   if (content.substring(1) == "C9 17 AF E3" ) //change here the UID of the
card/cards that you want to give access
```

```
  {
    Serial.println("4-Simon Jones");
    ser.write(4);
    Serial.println();

    delay(3000);
  }
   if (content.substring(1) == "D9 4D 0C 1B" ) //change here the UID of the
card/cards that you want to give access
  {
    Serial.println("5-Dimitri Levrock");
    ser.write(5);
    Serial.println();

    delay(3000);
  }
    if (content.substring(1) ==  "3B 06 A9 1B" ) //change here the UID of the
card/cards that you want to give access
  {
    Serial.println("6-Jasmine Joseph");
    ser.write(6);
    Serial.println();

    delay(3000);
  }
  }
```

Section 2

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
// WiFi parameters
#define WLAN_SSID       "Your Wifi SSID"
#define WLAN_PASS       "Your Wifi Password"

// Adafruit IO
#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883
#define AIO_USERNAME    "Your AIO Username"
#define AIO_KEY         "Your AIO Key"  // Obtained from account info on
io.adafruit.com

// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;

// Setup the MQTT client class by passing in the WiFi client and MQTT server
and login details.
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT,
AIO_USERNAME, AIO_KEY);

Adafruit_MQTT_Publish Attendance = Adafruit_MQTT_Publish(&mqtt,
AIO_USERNAME "/feeds/RFID_Attendance");

char ID;
/***************************** Sketch Code ************************************/

void setup() {
  Serial.begin(115200);
  Serial.println(F("Adafruit IO Example"));

  // Connect to WiFi access point.
  Serial.println(); Serial.println();
  delay(10);
```

```
  Serial.print(F("Connecting to "));
  Serial.println(WLAN_SSID);

  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(F("."));
  }
  Serial.println();

  Serial.println(F("WiFi connected"));
  Serial.println(F("IP address: "));
  Serial.println(WiFi.localIP());

  // connect to adafruit io
  connect();

}

// connect to adafruit io via MQTT
void connect() {
  Serial.print(F("Connecting to Adafruit IO... "));
  int8_t ret;
  while ((ret = mqtt.connect()) != 0) {
    switch (ret) {
      case 1: Serial.println(F("Wrong protocol")); break;
      case 2: Serial.println(F("ID rejected")); break;
      case 3: Serial.println(F("Server unavail")); break;
      case 4: Serial.println(F("Bad user/pass")); break;
      case 5: Serial.println(F("Not authed")); break;
      case 6: Serial.println(F("Failed to subscribe")); break;
      default: Serial.println(F("Connection failed")); break;
    }

    if(ret >= 0)
      mqtt.disconnect();

    Serial.println(F("Retrying connection..."));
```

```
    delay(5000);
  }
  Serial.println(F("Adafruit IO Connected!"));
}

void loop() {
  // ping adafruit io a few times to make sure we remain connected
  if(! mqtt.ping(3)) {
    // reconnect to adafruit io
    if(! mqtt.connected())
      connect();
  }
   if ( Serial.available() ) { // Update and send only after 1 seconds
   char a = Serial.read();
   ID = a;

   if (! Attendance.publish(ID)) {                    //Publish to Adafruit
      Serial.println(F("Failed"));
    } else {
      Serial.println(F("Sent!"));
    }
}
}
```