

**CARPOOLIFY**

MOBILE APPLICATION

Designed By:

Mainul Haque

Soojung Kwak

Anand Kasote

Table of Contents

[Table of Contents 1](#_Toc469752568)

[PART A – THE BUSINESS / DOMAIN MODEL 2](#_Toc469752569)

[Background 2](#_Toc469752570)

[Problem 2](#_Toc469752571)

[Scope 2](#_Toc469752572)

[Vision Statement 3](#_Toc469752573)

[System capabilities 3](#_Toc469752574)

[Business Benefits 4](#_Toc469752575)

[Workflows 5](#_Toc469752576)

[Use Case List 11](#_Toc469752577)

[Use Cases 14](#_Toc469752578)

[Prioritized, Consolidated list 17](#_Toc469752579)

[Use Case Description 17](#_Toc469752580)

[Use Case Brief Description 18](#_Toc469752581)

[Domain Class Diagram 21](#_Toc469752582)

[Technology Tools for Software Development 22](#_Toc469752583)

[Project Plan 23](#_Toc469752584)

[PART B – Software Design Specifications 24](#_Toc469752585)

[Refinement to Part A 24](#_Toc469752586)

[Use Case Description 26](#_Toc469752587)

[Use Case Brief Description 27](#_Toc469752588)

[Detail design class diagram 30](#_Toc469752589)

[sequence diagrams 32](#_Toc469752591)

[state diagrams 35](#_Toc469752592)

[project plan 36](#_Toc469752593)

[PART C – Design Specifications 37](#_Toc469752594)

[Refinement & Corrections 37](#_Toc469752595)

[High level architecture design 37](#_Toc469752596)

[Mock-up UI 38](#_Toc469752597)

[ERD Model & Database Schema 40](#_Toc469752598)

[Component & Deployment 41](#_Toc469752599)

[Skeletal or stub code 42](#_Toc469752600)

[Technology tools for Software Implementation 48](#_Toc469752601)

[Project Plan 49](#_Toc469752602)

# PART A – THE BUSINESS / DOMAIN MODEL

## Background

This is a mobile app which lets you find people going on the same route as you and allow you to connect with a car pooling community. After the user inputs the destination, this app will connect people with the app database and match their location and connect with users going to same destination.

## Problem

One of the biggest challenges in big cities is to improve people mobility. Thousands of people move at the same times every day, and that’s how they keep the system: they go to work, to school, and even run errands during the same hours. The majority of these people prefer to use their own vehicles because it is faster, more comfortable and convenient, however most are the only individual in the car. This creates a lot of additional traffic and increases pollution. The government incentives to carpool is not enough to change people’s mindset and reduce congestion. Sometimes people have hard time finding people going in same direction. However, currently there is no similar app for carpooling.

## Scope

## 

**Profile** - can access the basic profile information on a user account including their first name, email address and profile picture and so on.

**Places** – Save and retrieve user destination (Home and Work).

**Request** – Make requests for carpool on behalf of users.

**All Trips** – Get detail of the trip that the user is currently taking regardless of how it was requested. (via app or a third party app)

## Vision Statement

A group of three innovative software developers came together to develop a mobile app which will not only reduce the traffic on the roads during peak hours but also help to reduce the pollution. The idea is to encourage carpooling between individuals who are travelling to the same destination ensuring todays and next generation a more sustainable place.

## System capabilities

Our mobile application will have the following features:

1. Google Maps API will facilitate the discovery of users who are in the same region and are willing to carpool to the same destination along the route.
2. The use of Near Field Communication (NFC), will guarantee people who will carpool are in the same car, since it only works up to 10 cm of each other. The users will tap their phones at the beginning and at the end of the ride.
3. The mobile application will calculate the actual place through device geolocation – GPS, Wi-Fi or mobile network.
4. Authentication will be done with a valid driver’s license to prevent people that cannot drive from earning points just by sharing a ride.
5. Local business will exhibit their brands in multiple places in the app.
6. After tap the device at the final destination, the application will verify the similarity of the route and determine the points gained.
7. The points to be won will be multiplied according the number of people carpooling.
8. The earned points will be redeemed by products from the business partners who are exposing their logo in the app.

## Business Benefits

* Brands will have their logo exposed in the app;
* The brands will be known as “greener”, what increases the customer perception that they are concerned about the environment;
* Companies can choose this app to advertise their products.
* Customer retention - the accumulation of points toward a reward retain existing customers.
* Attract new customers.

## Workflows

1. **Customer Registration Process**
   1. Customer prompts sign in.
   2. System requests registration information from customer.
      1. If new customer, registers new customer into system.
      2. Customer provides registration information.
      3. System validates and verifies information and enters it into the system.
      4. System checks for pre-existing match to customer data.
   3. If existing customer, Customer provides Log in information.
   4. System validates information and enters it into the system.
   5. System allows log in.



1. **Application Redeem Points and Ride System**
2. Customer responds to prompt.
3. If customer wants to redeem or to ride.
4. Customer desire to redeem - the application verify if there are enough points.
   1. If there are not enough points, the system redemption menu will take an exit with an alert message.
   2. If there are enough points system request the customer to go to the participant stores.
   3. If there are enough point for the products customer wants to exchange for, points will be redeeming otherwise it will exit.
5. Customer desires to ride.
   1. Application will search for the location.
   2. If routes match it will connect the riders and driver, otherwise it will exit.
   3. The mobile application will connect the riders and the driver.
   4. Rides will be shared.
   5. Driver and Rider(s) will reach to the destination.
   6. Users will tap their phones to receive the points.
6. Customer exits.



1. **Payment system**
   1. Customer prompts to process payment.
   2. System provides payment option.
   3. Customer choose option for payment.
      1. System required to fill required information.
      2. Customer provides information.
      3. System validates information and enters it into the system.
      4. System process payment.
   4. Customer view payment history.
   5. Customer exists.



## Use Case List

1. **Customer Registration Process**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Event | Trigger | Source | Use Case | Response | Destination |
| 1.Customer wants to register | Customer requests for registration | Customer | Start registration Process | Deploy registration screen | Customer |
| 3. Customer wants to  sign in | Customer requests  sign in | Customer | Sign in process | Sign In confirmation | Customer |
| 2 Customer provides registration info | Customer fills in registration screen | Customer | Process registration | Provide account number and password | Customer |

1. **Customer Login**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Event | Trigger | Source | Use Case | Response | Destination |
| Customer wants to ride | Customer select ride option | Rider/Driver. | Open the ride option | Confirm ride  option | Rider/Driver. |
| Customer enter location | Searches the location | Riders/Driver | Matching Routes | Location  matches | Rider/Driver |
| Customer tap at start point (first tap) | Trip is started | Riders/Driver | Start the matched trip | Location is  matched | Rider/Driver |
| Customer reaches to destination (final tap) | Tapped phone | Rider/Driver. | Conclude the trip and assigned points. | Assign the  Points. | Rider/Driver. |
| Customer wants to Redeem points | Customer checks for the points balance | Rider/Driver. | Verify the amount of points | System confirms if there are enough points. | Rider/Driver |
| Customer wants to exchange points for products | Customer selects exchange option | Rider/Driver. | Exchange points for products | System confirms if there are enough points | Rider/Driver. |

1. **Customer Payment System**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Event | Trigger | Source | Use Case | Response | Destination |
| 1. Customer wants to pay | Customer select payment option | Customer | Open the payment option | Confirm Payment option | Customer |
| 2. Customer wants to select  Payment method | Customer select  Payment method | Customer | Open the payment option | Confirm Method option | Customer |
| 3. Customer provides required info | Customer fills in payment screen | Customer | Payment in process | Provide account number and password | Customer |
| 4. Customer wants to view payment history | Customer request for payment history | customer | Open customer  payment records | Check the payment records | Customer |

## Use Cases

1. **Customer Registration Process**



1. **Customer booking system**



1. **Customer payment system**

****

## Prioritized, Consolidated list

|  |  |
| --- | --- |
| **Use Case Goal** | **Priority** |
| Process registration | H |
| Exchange points for products | H |
| Open the ride option | H |
| Conclude the trip and assign points | H |
| Sign in process | M |
| Start registration Process | M |
| Open payment option | H |
| Start payment process | H |

## Use Case Description

**Register** - user will register in the application system.

**Sign in** - user provide information for signing in.

**Process registration** –application will process the information provided by the application.

**Open ride option** - user will select ride option in the application.

**Matching routes Application will** - user enter the destination location and search for the match trip.

**Start the match trip** – with the first tap the system will be updated as trip started on the matching routes.

**Conclude the match trip and assign the points** – with the second tap application will come to know that the trip has finished and the points will be given to customers.

**Verify the amount of points** – customer will go to participant stores to redeem the points if customer have enough points to redeem.

**Exchange points for products** – system will check enough points for the selected product by the customer.

**Open payment option** - user will select payment option for using the application.

**Process payment** –application will process the payment after user provides required information.

## Use Case Brief Description

1. **Process Registration**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | **Process Registration** | |
| Scenario: | Create a user account | |
| Triggering event: | A user downloaded the mobile application | |
| Brief description: | A user creates a username and a password. The system validates the entered name and password and logs the user into the system. | |
| Actors: | User | |
| Related use cases: | Start registration Process | |
| Stakeholders: | User, Mobile Developers | |
| Preconditions: | The system is in the registration state and has the login screen displayed. | |
| Post conditions: | Account created | |
| Flow of activities: | Actor | System |
| 1. User wants to register into the mobile application. 2. User create a username and password. | 1. System validates the received username and password. 2. System creates a new user. 3. System logs the actor into the system. |
| Exception conditions | 1. The user enters an invalid name and/or password. | |

1. **Exchange Points for Products**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | **Exchange Points for Products** | |
| Scenario: | Exchange points accumulated for ride. | |
| Triggering event: | Customer selects exchange option. | |
| Brief description: | A user can ride as a passenger or as a driver. The user accumulated enough points can exchange for rides. | |
| Actors: | User | |
| Related use cases: | Verify the amount of points | |
| Stakeholders: | User | |
| Preconditions: | User must have points to exchange | |
| Post conditions: | Rides exchanged. | |
| Flow of activities: | Actor | System |
| 1. User verifies the amount of points 2. User has reached the least amount of points. 3. User exchange the points for rides. | 1. System confirms if there are enough points to redeem. 2. System validates the points 3. System allow the exchange. |
| Exception conditions | 1. User did not reach the minimum points required. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | **Customer Payment System** | |
| Scenario: | Customers wants to pay/view payment history | |
| Triggering event: | Customer selects payment option | |
| Brief Description: | Customer pays for the ride by selection payment method from payment option or user can view payment history. | |
| Actors: | User | |
| Related Use Cases: | Record number of rides, calculate amount to be paid and verify the amount. | |
| Stakeholders: | User, mobile developers | |
| Preconditions: | The system is in the registration state and has the payment screen displayed. | |
| Post conditions: | Payment confirmed | |
| Flow activities: | Actor | System |
| * User pays for the service * User view payment history | * System validates the received payment information * System process payment |
| Exception conditions: | The user enters an invalid card number or name | |

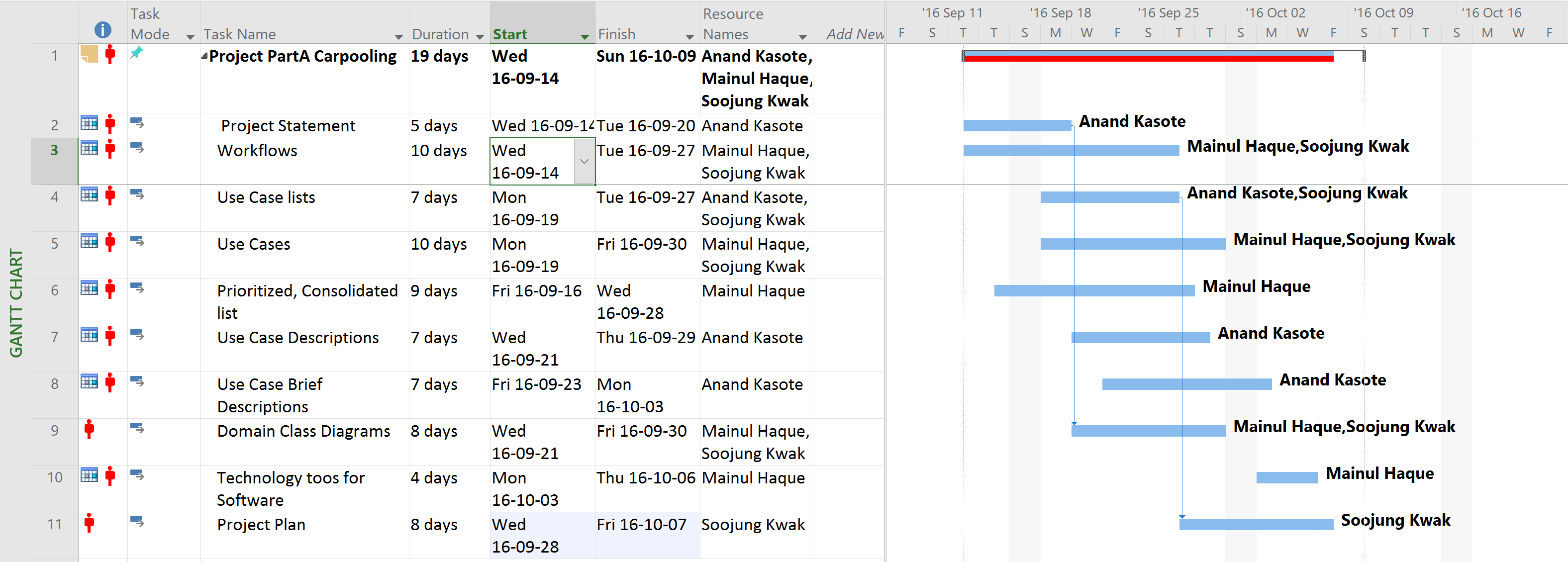
1. **Customer Payment System**

## Domain Class Diagram

## Technology Tools for Software Development

|  |  |  |
| --- | --- | --- |
| Documentation tools and 3 tier Architecture | Development tools | Descriptions |
| Documentation | Microsoft word,  Visual Paradigm,  Visio,  Google Drive | Tools used to prepare documentation, diagrams and sharing the work between our team mates |
| Presentation tier | HTML/XHTML/CSS JavaScript | This is where the user interacts with the system |
| Business tier | Java | Process commands, makes logical decisions, and performs calculations |
| Data tier | SQL Server | Information is stored and retrieved from System |

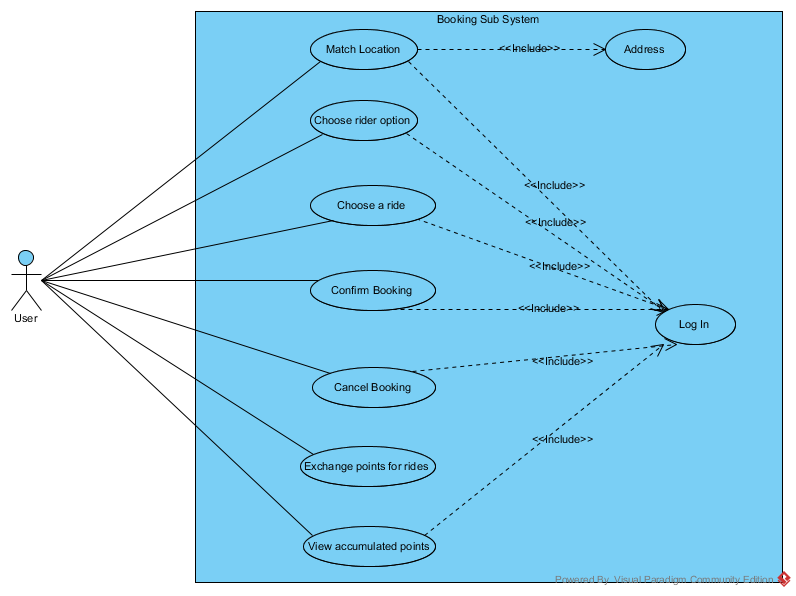
## Project Plan



# PART B – Software Design Specifications

## Refinement to Part A

**Booking System**



**Payment System**



## Use Case Description

**Register** - user will register in the application system.

**Sign in** - user provide information for signing in.

**Process registration** –application will process the information provided by the application.

**Choose ride option** - user will select ride option in the application.

**Matching routes Application will** - user enter the destination location and search for the match trip.

**Confirm booking** – application will confirm booking after user confirm for booking.

**Cancel booking** – user can cancel booking for any valid reason or to modify wrong input

**Verify the amount of points** – customer will go to participant stores to redeem the points if customer have enough points to redeem.

**Exchange points for products** – system will check enough points for the selected product by the customer.

**Check fare amount** - user will check fare amount for using the application.

**Pay fare**–application can for ride by using this application.

**Confirm payment**- After validating the user payment info system will user to confirm to process the payment.

**Cancel payment** - User will be able to cancel payment for rectify the wrong amount processed for payment.

**View payment history** - User will be able to view payment history using this application.

## Use Case Brief Description

1. **Process Registration**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | **Process Registration** | |
| Scenario: | Create a user account | |
| Triggering event: | A user downloaded the mobile application | |
| Brief description: | A user creates a username and a password. The system validates the entered name and password and logs the user into the system. | |
| Actors: | User | |
| Related use cases: | Start registration Process | |
| Stakeholders: | User, Mobile Developers | |
| Preconditions: | The system is in the registration state and has the login screen displayed. | |
| Post conditions: | Account created | |
| Flow of activities: | Actor | System |
| 1. User wants to register into the mobile application. 2. User create a username and password. | 1. System validates the received username and password. 2. System creates a new user. 3. System logs the actor into the system. |
| Exception conditions | 1. The user enters an invalid name and/or password. | |

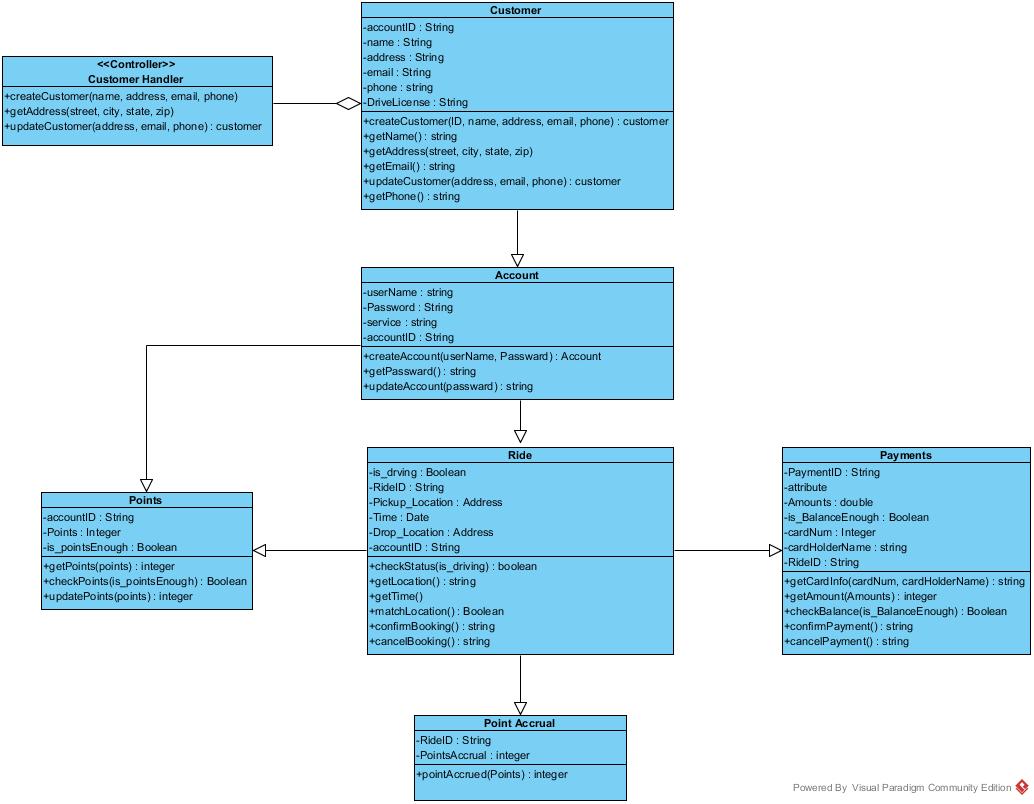
1. **Booking System**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | **Booking for ride** | |
| Scenario: | Customer will book the ride to reach destination | |
| Triggering event: | Customer provide destination location. | |
| Brief description: | A user can ride as a passenger or as a driver. The user accumulated enough points can exchange for rides. | |
| Actors: | User | |
| Related use cases: | Match the location to book ride | |
| Stakeholders: | User | |
| Preconditions: | User must provide valid address. | |
| Post conditions: | Points accumulation | |
| Flow of activities: | Actor | System |
| 1. User verifies the amount of points 2. User has reached the least amount of points. 3. User exchange the points for rides. | 1. System confirms if there are enough points to redeem. 2. System validates the points 3. System allow the exchange. |
| Exception conditions | 1. User did not reach the minimum points required. | |

|  |  |  |
| --- | --- | --- |
| Use Case Name: | **Customer Payment System** | |
| Scenario: | Customers wants to pay/view payment history | |
| Triggering event: | Customer selects payment option | |
| Brief Description: | Customer pays for the ride by selection payment method from payment option or user can view payment history. | |
| Actors: | User | |
| Related Use Cases: | Record number of rides, calculate amount to be paid and verify the amount. | |
| Stakeholders: | User, mobile developers | |
| Preconditions: | The system is in the registration state and has the payment screen displayed. | |
| Post conditions: | Payment confirmed | |
| Flow activities: | Actor | System |
| * User pays for the service * User view payment history | * System validates the received payment information * System process payment |
| Exception conditions: | The user enters an invalid card number or name | |

**Process payment:**

## Detail design class diagram



**CRC Cards**

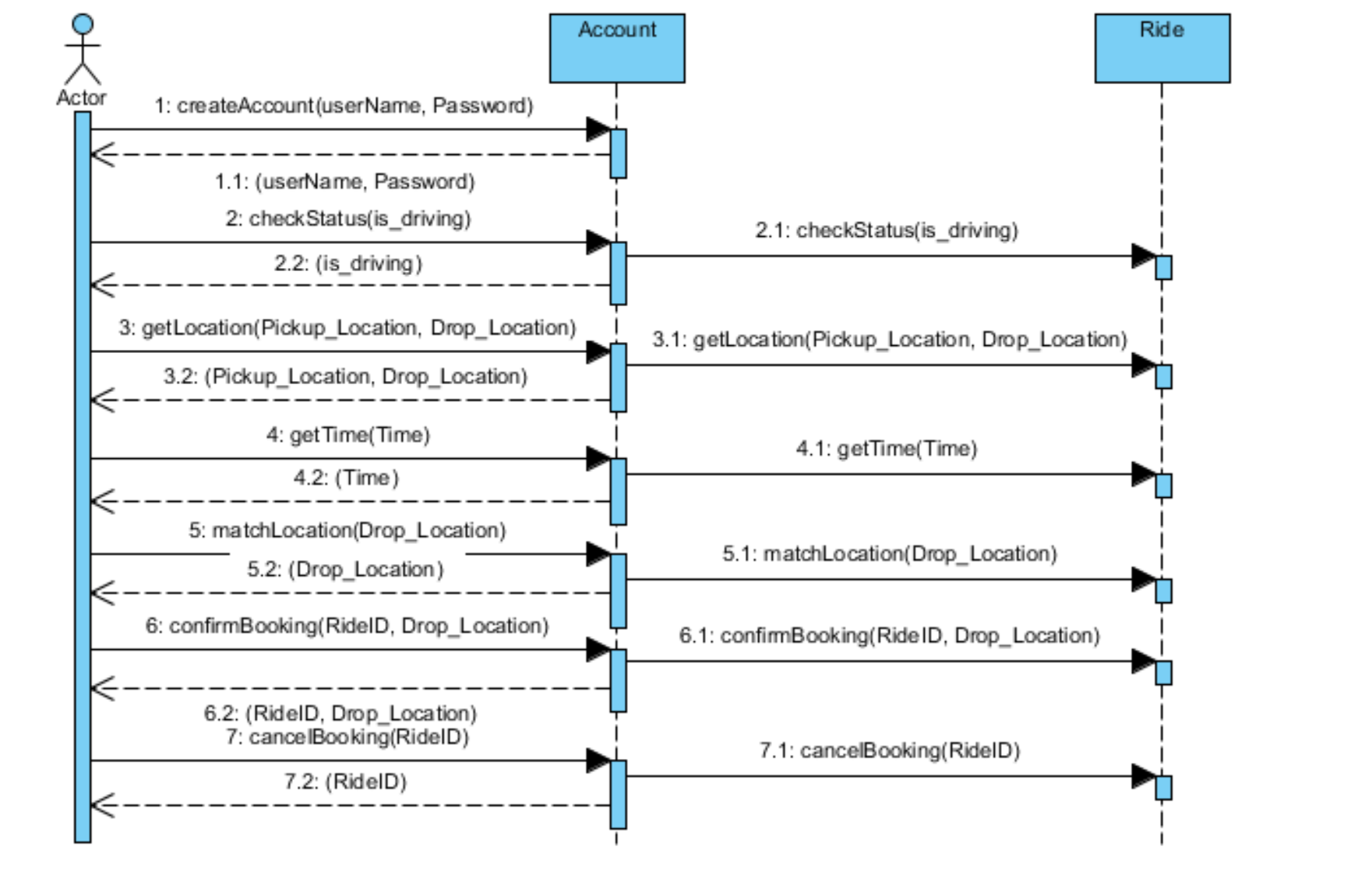
## 

## sequence diagrams

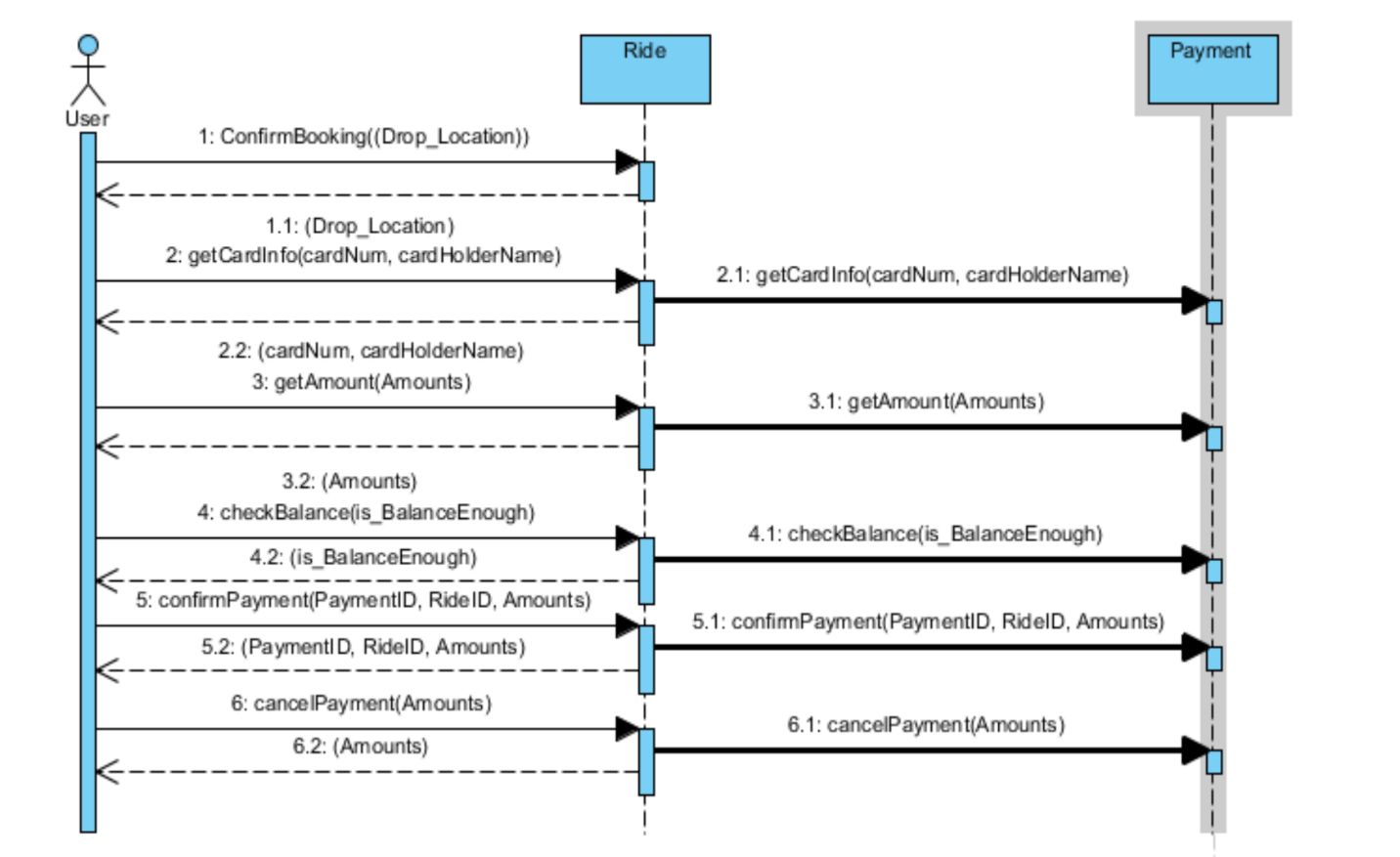
* Registration



* Booking



* Payment



## state diagrams

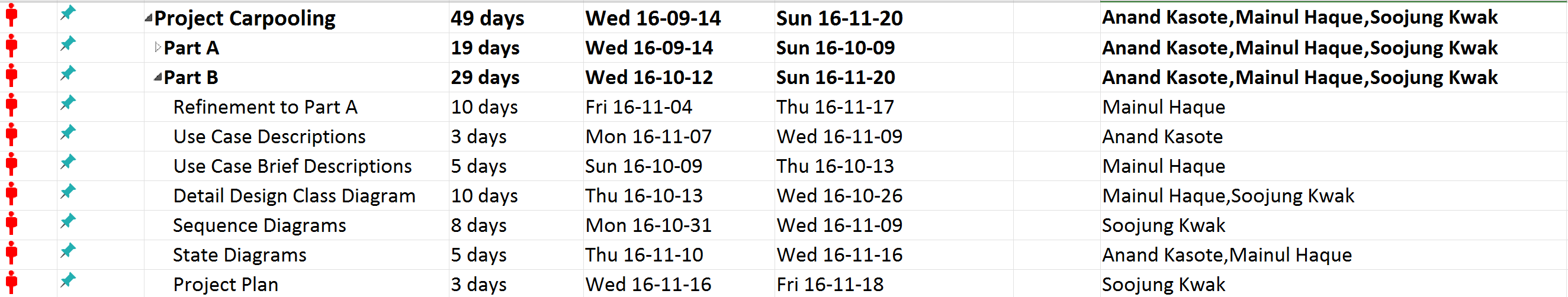


Payment Class



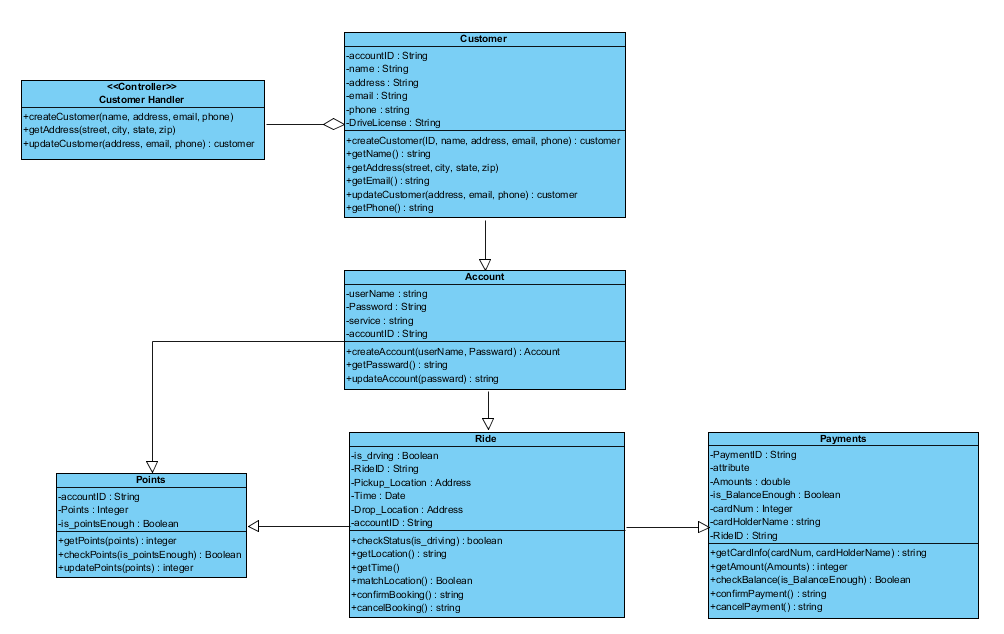


## project plan

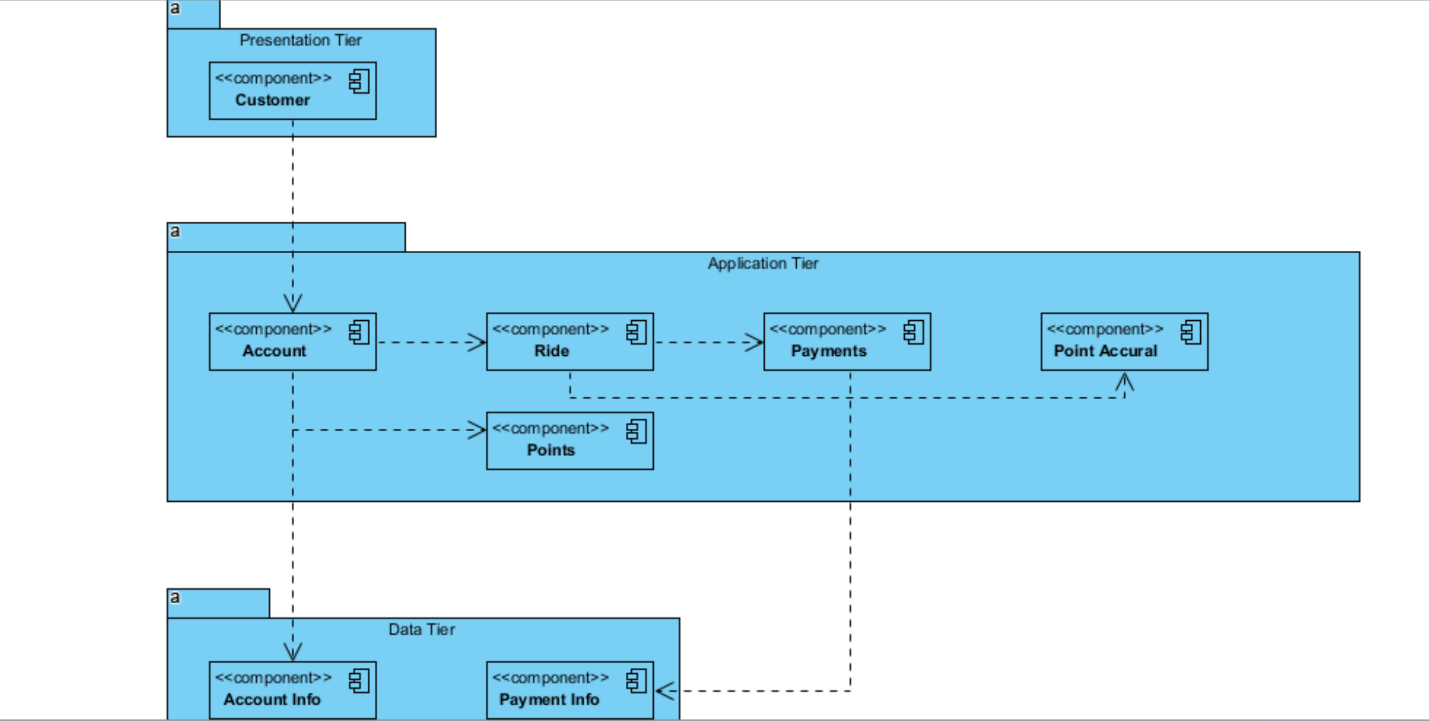


# PART C – Design Specifications

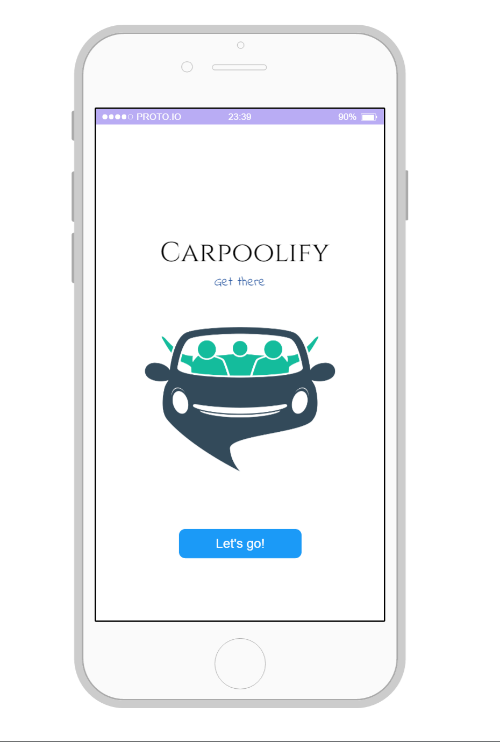
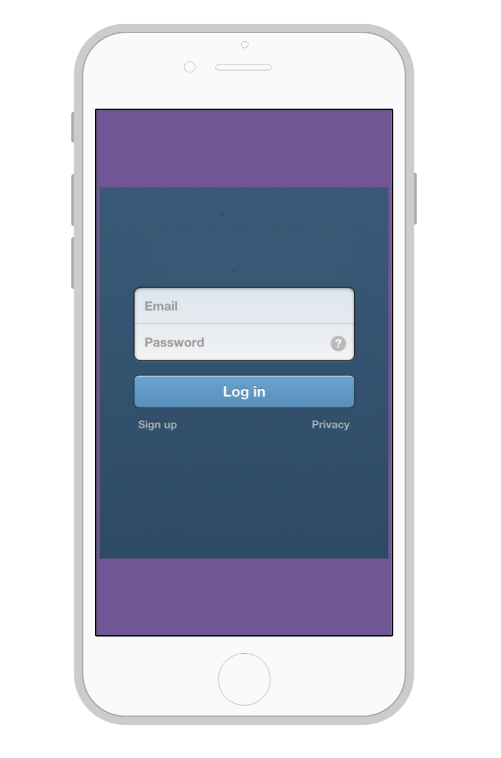
## Refinement & Corrections

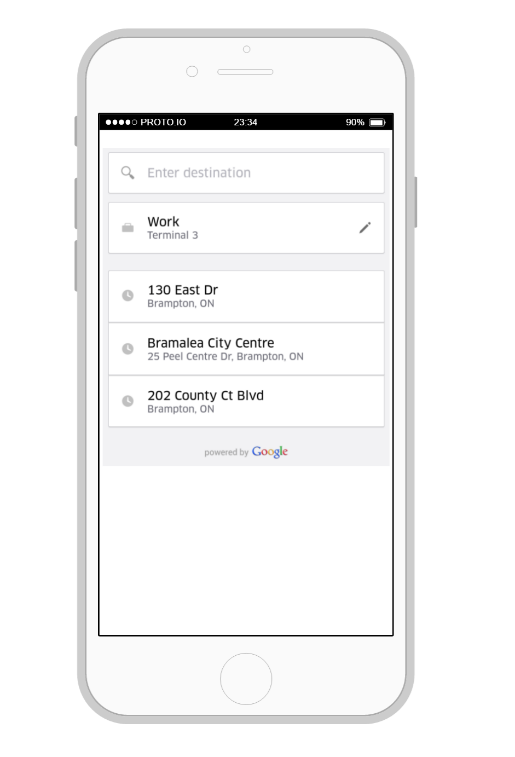
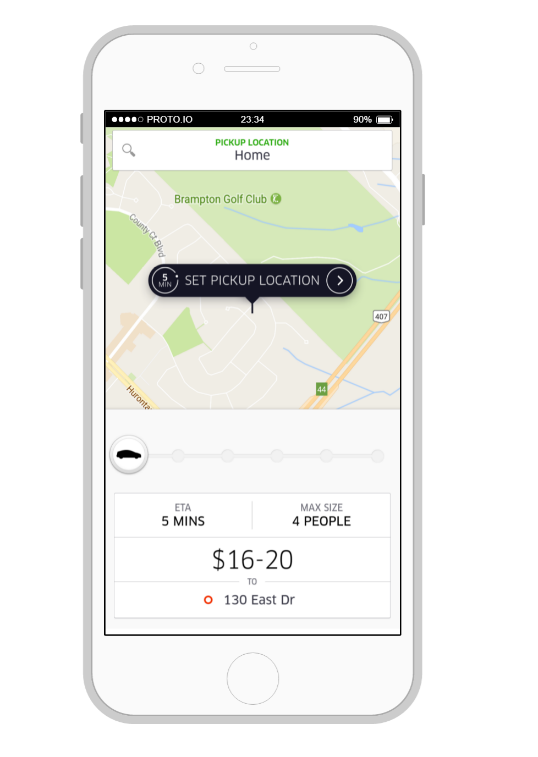


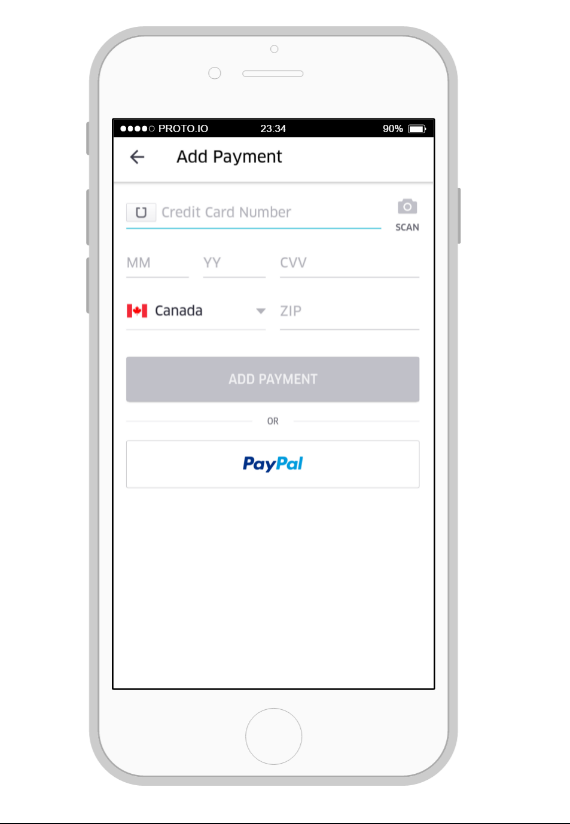
## High level architecture design



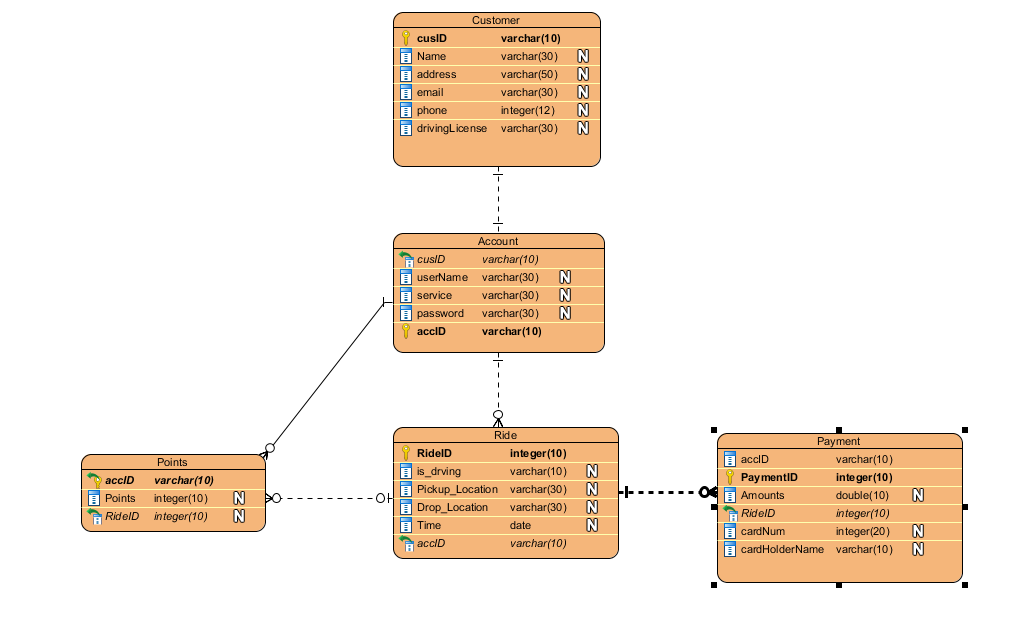
## Mock-up UI

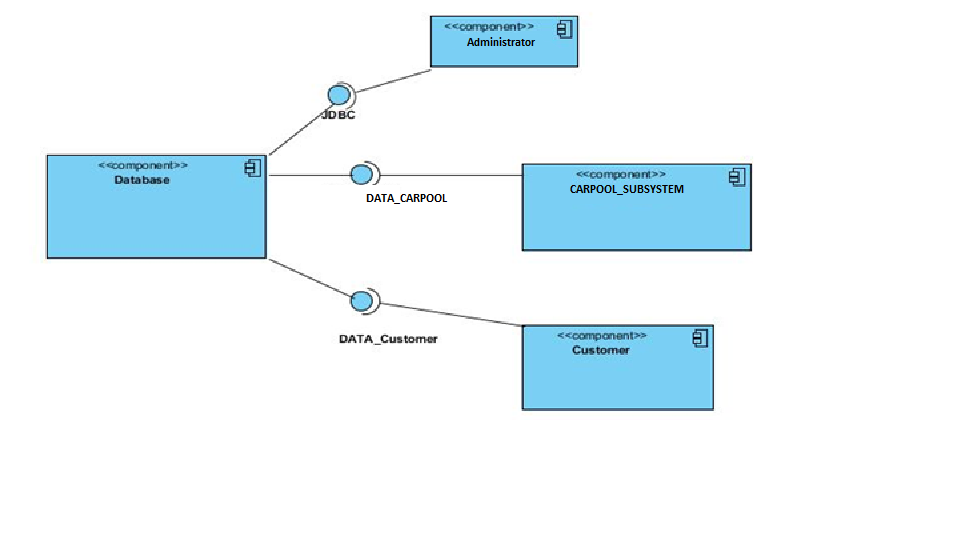
 

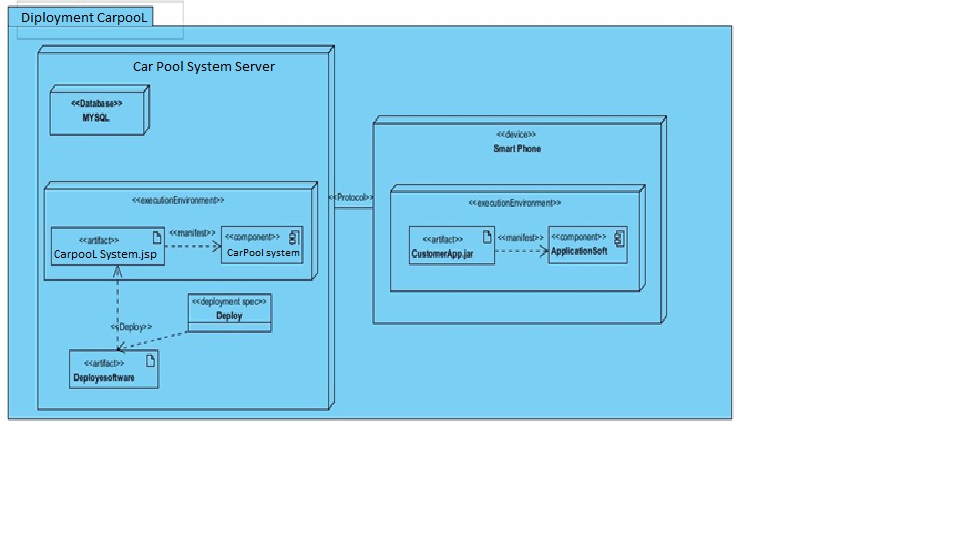


## ERD Model & Database Schema



## Component & Deployment





## Skeletal or stub code

public class Customer {

private String \_accountID;

private String \_name;

private String \_address;

private String \_email;

private String \_phone;

private String \_driveLicense;

public Account \_unnamed\_Account\_;

public customer createCustomer(Object aID, Object aName, Object aAddress, Object aEmail, Object aPhone) {

throw new UnsupportedOperationException();

}

public String getName() {

return this.\_name;

}

public void getAddress(Object aStreet, Object aCity, Object aState, Object aZip) {

throw new UnsupportedOperationException();

}

public String getEmail() {

return this.\_email;

}

public customer updateCustomer(Object aAddress, Object aEmail, Object aPhone) {

throw new UnsupportedOperationException();

}

public String getPhone() {

return this.\_phone;

}

}

public class Account {

private String \_userName;

private String \_password;

private String \_service;

private String \_accountID;

public Customer \_unnamed\_Customer\_;

public Points \_unnamed\_Points\_;

public Ride \_unnamed\_Ride\_;

public Account createAccount(Object aUserName, Object aPassward) {

throw new UnsupportedOperationException();

}

public String getPassward() {

throw new UnsupportedOperationException();

}

public String updateAccount(Object aPassward) {

throw new UnsupportedOperationException();

}

}

public class Ride {

private Boolean \_is\_drving;

private String \_rideID;

private Address \_pickup\_Location;

private Date \_time;

private Address \_drop\_Location;

private String \_accountID;

public Account \_unnamed\_Account\_;

public Points \_unnamed\_Points\_;

public Payments \_unnamed\_Payments\_;

public Point\_Accrual \_unnamed\_Point\_Accrual\_;

public boolean checkStatus(Object aIs\_driving) {

throw new UnsupportedOperationException();

}

public String getLocation() {

throw new UnsupportedOperationException();

}

public void getTime() {

throw new UnsupportedOperationException();

}

public Boolean matchLocation() {

throw new UnsupportedOperationException();

}

public String confirmBooking() {

throw new UnsupportedOperationException();

}

public String cancelBooking() {

throw new UnsupportedOperationException();

}

}

public class Payments {

private String \_paymentID;

private Object \_attribute;

private double \_amounts;

private Boolean \_is\_BalanceEnough;

private Integer \_cardNum;

private String \_cardHolderName;

private String \_rideID;

public Ride \_unnamed\_Ride\_;

public String getCardInfo(Object aCardNum, Object aCardHolderName) {

throw new UnsupportedOperationException();

}

public integer getAmount(Object aAmounts) {

throw new UnsupportedOperationException();

}

public Boolean checkBalance(Object aIs\_BalanceEnough) {

throw new UnsupportedOperationException();

}

public String confirmPayment() {

throw new UnsupportedOperationException();

}

public String cancelPayment() {

throw new UnsupportedOperationException();

}

}

public class Points {

private String \_accountID;

private Integer \_points;

private Boolean \_is\_pointsEnough;

public Account \_unnamed\_Account\_;

public Ride \_unnamed\_Ride\_;

public integer getPoints(Object aPoints) {

throw new UnsupportedOperationException();

}

public Boolean checkPoints(Object aIs\_pointsEnough) {

throw new UnsupportedOperationException();

}

public integer updatePoints(Object aPoints) {

throw new UnsupportedOperationException();

}

}

## Technology tools for Software Implementation

|  |  |  |
| --- | --- | --- |
| Documentation tools and 3 tier Architecture | Development tools | Descriptions |
| Documentation | Microsoft word,  Visual Paradigm,  Visio,  Google Drive | Tools used to prepare documentation, diagrams and sharing the work between our team mates |
| Presentation tier | HTML/XHTML/CSS JavaScript | This is where the user interacts with the system |
| Business tier | Java | Process commands, makes logical decisions, and performs calculations |
| Data tier | SQL Server | Information is stored and retrieved from System |

## Project Plan

