**Requirements Specifications**

**P 07: HITCHERR**

**<team member names & ids>**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **23100040** | **Mustafa Afzal** |
| **23100333** | **Nashrah Shaukat** |
| **23100020** | **Ebad Ur Rehman** |
| **23100337** | **Irtza Tariq** |
| **22100021** | **Muhammad Muneeb Arshad** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Introduction & system actors | 5 | 5 |
| Use case diagram | 10 | 10 |
| Use case descriptions | 20 | 15 |
| Class diagram | 20 | 10 |
| Sequence diagram | 20 | 5 |
| State diagram | 5 | 5 |
| Non-functional requirements | 5 | 0 |
| Who did what | 5 | 0 |
| Review checklist | 5 | 0 |
| Overall formatting/template | 5 | 5 |
| Late submission penalty | -20 |  |
| Folder structure on github | -5 | -5 |
| **Total** | **100** | **50** |
| Review | 20 |  |
| **Grand Total** |  |  |

**Table of Contents**

[1. Introduction 3](#_Toc82791699)

[2. System Actors 4](#_Toc82791700)

[3. Use Cases 5](#_Toc82791701)

[3.1 Use Case Diagrams 5](#_Toc82791702)

[3.2 Description of Use Cases 5](#_Toc82791703)

[3.2.1 Withdraw cash 5](#_Toc82791704)

[3.2.2 Transfer funds 6](#_Toc82791705)

[4. Class Diagram 7](#_Toc82791706)

[4.1 Diagram 7](#_Toc82791707)

[4.2 Description 7](#_Toc82791708)

[5. Sequence Diagrams 8](#_Toc82791709)

[5.1 Use case Name e.g., Withdraw cash 8](#_Toc82791710)

[5.2 Use case Name e.g., Transfer funds 8](#_Toc82791711)

[6. State Diagrams 9](#_Toc82791712)

[6.1 Diagram details 9](#_Toc82791713)

[6.2 Diagram 9](#_Toc82791714)

[7. Non-functional Requirements / Quality Attributes 10](#_Toc82791715)

[8. Who Did What? 11](#_Toc82791716)

[9. Review checklist 11](#_Toc82791717)

# Introduction

Hitcherr (also referred as the company) is a ride sharing app that intends to solve the transportation problem of the country. June 2022 saw a 250% increase in petrol prices as compared to June 2021 in Pakistan. This increase in petrol price gave rise to increase in fare prices of the cab startups such as Careem and Uber. This rise made space for cheaper services like InDriver in the country which are unsafe and offer no safety features such as SOS to its customer, neither it has invested in any checks or police verification of the drivers. The need for a transportation service which is reliable, secure and cheaper is at an all-time high in this country. Hitcherr will be a mobile application which will allow the drivers to offer routes and customers will be able to select a route based on their pickup and drop-off points. The drivers will be able to pick a maximum of 3 or 4 customers based on the size of their car. This will divide the total cost of the route, hence will be a cheaper alternative to Careem, Uber, or any other conventional and non-conventional public transport services.

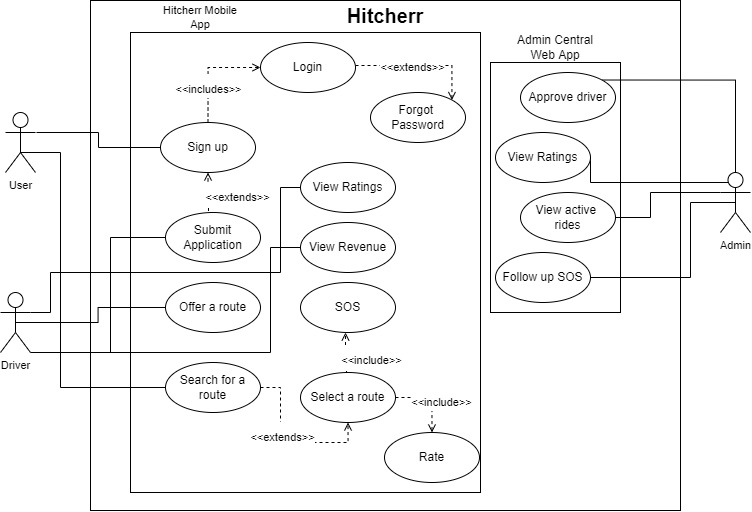
# System Actors

|  |  |
| --- | --- |
| **Actor Name** | **Description** |
| Driver | This individual will be offering routes from one place to another. They will be registered on the app after a process of verification from the company. |
| User/Rider | This individual will select a pick-up and drop-off location and will be able to see routes and their prices based on their overlap with the route offered by the driver. |
| Administrator | This individual will be a representative of the company who will have access to the admin panel web app of the system. |

# Use Cases

[Add more details in key use cases, otherwise it appears as if your system functionality is quite small. If use cases are actually quite simple, add some more features in your system. ]

## Use Case Diagrams



## Description of Use Cases

[**Select 20 most important use cases of your project and create their comprehensive descriptions.**]

<Write description of each use case separately using the template below.>

### Login

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-001 | |
| **Purpose** | | The user gets access to the system | |
| **Pre-conditions** | | A working application downloaded on user’s mobile phone | |
| **Post-conditions** | | The user is logged-in and can access the system | |
|  | | | |
| **Step #** | **Typical Course of Action** | |
|  | The user selects the login option on the application | |
|  | The user enter their mobile number and password then click on login button | |
|  | The credentials entered in step 2 are validated and upon successful validation the user is logged in and can access the system | |
|  | | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 2 if the credentials entered are not correct then the user is not allowed in. | |
| **Step #** | **Exception Paths** | |
|  | In step 8, if there are not sufficient funds, then an error message is displayed and execution proceeds to step 9. [Which is step 8 in this use case?] | |

### Driver Registration

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-001 |
| **Purpose** | | The driver wants to register themselves and applies for approval. The actor in this use case is the driver. |
| **Pre-conditions** | | The driver has a valid CNIC, driver’s license and car documents. |
| **Post-conditions** | | The application is successfully submitted, and the administrator receives the application on the admin side website. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The driver navigates to the registration screen for drivers on the application. | |
| 2. | The application asks for relevant information, including Name, phone number, email address, CNIC, driver’s license, car documents for the car the driver wants to register. | |
| 3. | The driver attaches all the relevant documents and enters the required information. | |
| 4. | The driver chooses the submit option to submit the application. | |
| 5. | The use case ends. | |
|  |  | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In steps 2 and 3 the driver can decide to cancel the application and go to step 5. | |
| **Step #** | **Exception Paths** | |
|  | In step 3, if the entered information in the text fields does not conform to the system requirements, an error is displayed till the user enters the correct information. | |
|  |  |  |

### Driver Sign up

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-002 |
| **Purpose** | | The driver wants to sign up by making an account. The actor in this use case is the driver. |
| **Pre-conditions** | | The driver has already submitted an application for registration and the application has been approved. |
| **Post-conditions** | | The driver successfully signs up the application. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The driver navigates to the confirmation email sent by the administrator which contains the link to the sign-up page. | |
| 2. | The driver opens the link and navigates to the sign-up page. | |
| 3. | The driver fills in the required information, including name, which would be visible to other users. | |
| 4. | The driver can upload a profile picture which would be visible to other users. | |
| 5. | The driver finishes signing up by clicking the ‘sign up’ button. | |
| 6. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In steps 3 and 4 the driver can decide to cancel the sign-up process by clicking the back button and go to step 6. | |
| **Step #** | **Exception Paths** | |
|  | In step 3, if the entered information in the text fields does not conform to the system requirements, an error is displayed till the user enters the information in correct format. | |
|  |  |  |

### Blocking a driver

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-003 |
| **Purpose** | | The administrator wants to block a certain driver. The actor in this use case is the administrator. |
| **Pre-conditions** | | The driver which the administrator wants to block is a valid driver and exists in the application database. |
| **Post-conditions** | | The driver is successfully blocked and cannot use the application to make ride requests. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The administrator navigates to the search bar. | |
| 2. | The administrator searches for the driver either by their name or their CNIC. | |
| 3. | The application displays the results of the search containing that specific driver. | |
| 4. | The administrator can then click on the ‘Block’ button next to the driver's profiles to block them. The block button is replaced by a ‘Unblock’ button. | |
| 5. | The use case ends. | |
|  |  | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 4, the administrator can choose to unblock the driver, by clicking the unblock button. The administrator can then repeat step 4 if need to or go to step 5. | |
| **Step #** | **Exception Paths** | |
|  | In step 2, if the entered name or CNIC is not valid the screen displays an error for the invalid information. | |
|  |  |  |

### User Sign-Up

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-004 |
| **Purpose** | | The user/rider wants to sign up to the application. The actor in this use case is the rider. |
| **Pre-conditions** | | The user has a valid email account to complete the sign-up process. |
| **Post-conditions** | | The user successfully signs up to the application and can book rides. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The user navigates to the sign-up page on the application. | |
| 2. | The application asks for the user’s information including name, email address, phone number and profile picture. Some of the information, including profile picture will be optional. | |
| 3. | The user enters the required information in the text fields. | |
| 4. | The user can then click on the ‘Sign up’ button to complete the sign-up process. | |
| 5. | The use case ends. | |
|  |  | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 2 and 3 the user can decide to cancel the sign-up process by clicking on the back button on the screen. The user goes to step 5. | |
| **Step #** | **Exception Paths** | |
|  | In step 3, if the entered information in the text fields does not conform to the system requirements, an error is displayed till the user enters the information in correct format. | |
|  |  |  |

### User Rating Driver

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-005 |
| **Purpose** | | The user wants to rate a driver based on their experience. |
| **Pre-conditions** | | The user is logged in and has just completed a ride |
| **Post-conditions** | | The user is returned to the dashboard. The driver can see his updated rating. The admin can see the new rating. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The ride has been completed and payment has been processed. | |
| 2. | The application prompts the user to enter a rating. | |
| 3. | The user selects a rating from 1 to 5. | |
| 4. | The user can add extra comments to go with the rating. | |
| 5. | The user presses the rate button, and the rating is added to the database. | |
| 6. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 2, the user can choose to not give a rating by pressing the close button. The user will go to step 6. | |
| **Step #** | **Exception Paths** | |
|  | N/A | |
|  |  |  |

### Driver Views Rating

### 

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-006 |
| **Purpose** | | The driver wants to see their rating. |
| **Pre-conditions** | | The driver is logged in. |
| **Post-conditions** | | The driver can see all their ratings, can continue normal use of the application. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The driver navigates to the account page. | |
| 2. | The driver sees his account information and alongside it his overall rating. | |
| 3. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | N/A | |
| **Step #** | **Exception Paths** | |
|  | In step 1, the driver has yet to receive a rating so he will have no overall rating. The driver will go to step 3. | |
|  |  |  |

### Admin Views Rating

### 

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-007 |
| **Purpose** | | The admin wants to see ratings given by the users |
| **Pre-conditions** | | The admin is logged into the admin webpage |
| **Post-conditions** | | The admin can take actions such as blocking a driver from this screen after seeing ratings. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The admin navigates to the ratings dashboard. | |
| 2. | The admin can see all ratings by sorted by recent. | |
| 3. | The admin can see which driver each rating was given to. | |
| 4. | The admin can see the overall rating of each driver who received the rating. | |
| 5. | The admin also sees visualizations of recent rating trends. | |
| 6. | The use case ends. | |
|  |  | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 2, the ratings can also be sorted by lowest rating, among other options. The admin goes to step 3. | |
| **Step #** | **Exception Paths** | |
|  | In step 2, if no ratings exist the page will be empty with a message displayed saying ‘No ratings exist’ | |

3.2.8 Driver Views Revenue

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-008 | |
| **Purpose** | | The driver wants to see revenue earned from a ride | |
| **Pre-conditions** | | The driver completed the ride successfully. | |
| **Post-conditions** | | The driver can see the revenue earned from that specific ride. | |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | Once the ride ends, the app will display the revenue the driver earned from that specific ride. | |
| **2.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | N/A | |
| **Step #** | **Exception Paths** | |
| 1. | If the ride has not been completed, the revenue will not be shown. [Considering the precondition of the use case, this should not happen.] | |

3.2.9 Admin Views Active Drivers

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-009 | |
| **Purpose** | | Admin wants to view the active drivers at any given point in time | |
| **Pre-conditions** | | Admin is logged into the admin webpage | |
| **Post-conditions** | | From here admin can request a driver's location and view their information such as name, phone number, car make and model, car registration number etc. | |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The admin navigates to the Driver dashboard | |
| **2.** | The admin can see all the active drivers at any given point. | |
| **3.** | The admin can click on the driver's name to get further information. | |
| **4.** | The use case ends | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| 1. | N/A | |
| **Step #** | **Exception Paths** | |
| 1. | If a driver is not active, they will not show up here. | |

3.2.10 Admin requests a driver’s location

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-010 |
| **Purpose** | | The admin wants to view a driver’s current location |
| **Pre-conditions** | | The admin is logged into the admin webpage. |
| **Post-conditions** | | The driver can see all the active drivers and request a driver’s current location |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The admin navigated to the Driver dashboard | |
| 2. | The admin clicks on the location button showing next to the driver’s name to view their current location. | |
| 3. | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | N/A | |
| **Step #** | **Exception Paths** | |
|  | If the driver is not active, the admin will not be able to view their location. | |

3.2.11 A user sends an SOS signal to the admin

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-011 |
| **Purpose** | | The user wants to send an SOS signal to the admin because there is an emergency onboard while they are taking the ride. |
| **Pre-conditions** | | The user is on a ride at that time.  The user has an active internet connection.  The admin is logged into the admin webpage. |
| **Post-conditions** | | The admin can contact the relevant authorities and inform them of the emergency and all the relevant details regarding the driver and his current location |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The user navigates to the active/current ride page. | |
| 2. | The user presses the SOS button. | |
| 3. | The user confirms that they want to send the SOS signal. | |
| 4. | The admin sees the SOS signal pop up. | |
| 5. | The admin clicks on the pop up and is taken to a screen where the details regarding the driver, their car and user are mentioned along with the current location of the driver. | |
| 6. | The admin contacts the relevant authorities and provides them with the information they got in step 5. | |
| 7. | The admin marks the SOS as resolved. | |
| 8. | The user gets confirmation of their SOS being received and relevant authorities being contacted. | |
| 9. | Use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 3, if the user does not confirm the SOS, it is not sent, and we come to Step 9. | |
| **Step #** | **Exception Paths** | |
|  | N/A | |
|  |  | |

3.2.12 Forgot Password

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-004 |
| **Purpose** | | The user/rider wants to change password. |
| **Pre-conditions** | | The user has an active account on the application. |
| **Post-conditions** | | The user will be able to log in using the new password. |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The user navigates to the log-in page on the application. | |
| 2. | The user clicks on ‘Forgot password’ button. | |
| 3. | The user is prompted to enter their email. | |
| 4. | The application will display a message that a link to renew password has been emailed. | |
| 5. | The user can access the link which will navigate to password renewal page. | |
| 6. | The user enters a new password. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 2 and 3 the user can decide to cancel the sign-up process by clicking on the back button on the screen. The user goes to step 5. | |
| **Step #** | **Exception Paths** | |
|  | In step 3, if the entered information in the text fields does not conform to the system requirements, an error is displayed till the user enters the information in correct format. | |
|  |  |  |

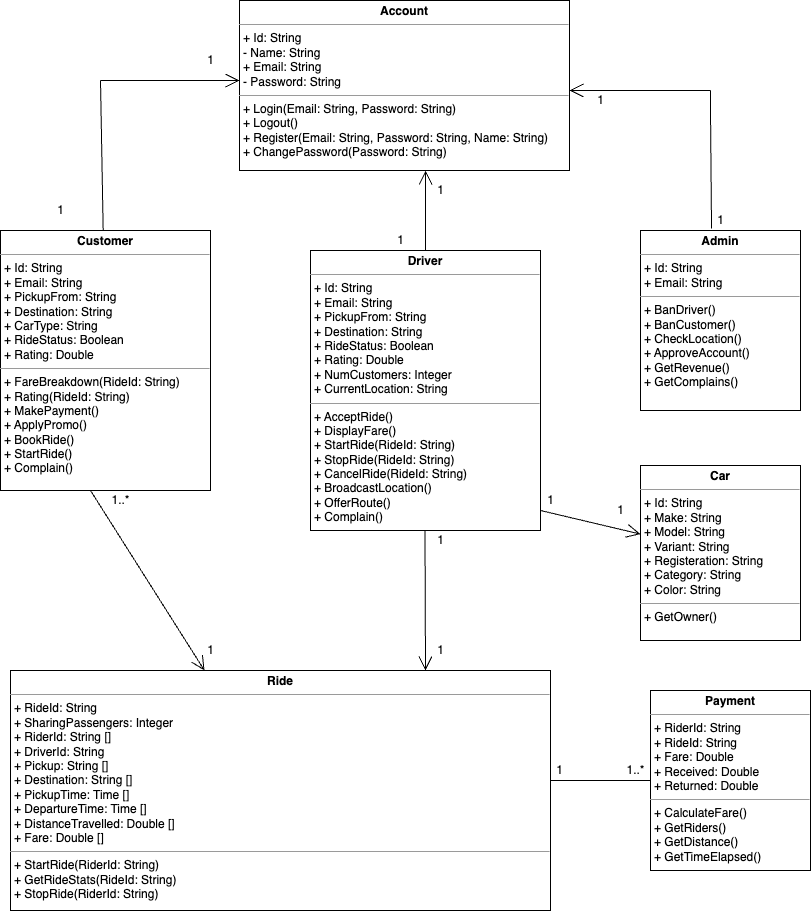
3.2.13 Offer a route

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-013 |
| **Purpose** | | Driver offers a ride |
| **Pre-conditions** | | The user is on a ride at that time.  The user has an active internet connection.  The admin is logged into the admin webpage. |
| **Post-conditions** | | The admin can contact the relevant authorities and inform them of the emergency and all the relevant details regarding the driver and his current location |
|  | | |
| **Step #** | **Typical Course of Action** | |
| 1. | The user navigates to the active/current ride page. | |
| 2. | The user presses the SOS button. | |
| 3. | The user confirms that they want to send the SOS signal. | |
| 4. | The admin sees the SOS signal pop up. | |
| 5. | The admin clicks on the pop up and is taken to a screen where the details regarding the driver, their car and user are mentioned along with the current location of the driver. | |
| 6. | The admin contacts the relevant authorities and provides them with the information they got in step 5. | |
| 7. | The admin marks the SOS as resolved. | |
| 8. | The user gets confirmation of their SOS being received and relevant authorities being contacted. | |
| 9. | Use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
|  | In step 3, if the user does not confirm the SOS, it is not sent, and we come to Step 9. | |
| **Step #** | **Exception Paths** | |
|  | N/A | |
|  |  | |

# Class Diagram

[More classes can be added for routes, ratings, active rides, vehicle location, paths etc. The current model does not give complete picture of your system. This will also help you identify some more use cases.]

## Diagram



## Description

There are a total of seven classes in the class diagram. The classes are named as follows: Account, Customer, Driver, Ride, Car, Admin and Payment. The Account class manages the unique IDs, names, passwords and email addresses of all users. There are three classes that are associated with the Account class. Customer, Driver and Admin have a one to one relationship with the Account class. This shows that one customer, driver or admin member will only have one account.

The customer class further focuses on the functionality of the customer. It contains variables such as the location from where the customer will be picked up, their destination, a boolean variable showing whether the customer has an active ride status, the car type they have demanded and their rating in the community. The boolean variable is significant in our application as it acts as a flag that makes more routes available to other customers. This class has a special method to start a ride which will be used to ensure that the customer acknowledges the starting of a ride to abort any frauds or deceptions.

The driver class focuses on similar aspects to the customer. It also focuses on the number of customers that the driver is driving in order to split the fare. The driver may also choose to accept a ride, cancel it or end it according to the need. The driver may also broadcast their location on demand of the admin.

The admin class has functionalities limited to monitoring the entire situation. The admin may ban a user from using the application by monitoring complaint status and ratings. They are also responsible for approving accounts when they request to register.

The ride class holds a crucial position in the application. It contains a unique riding ID that will be used to save past rides of users and lodge complaints and ratings. The ride class consists of lists of areas where rides started, ended and the time duration they were active for. This data is used to split the fare among customers. The class contains a list for fares that would index fares for different customers according to their index in the list of RiderId.

The car class contains information of the vehicle driven by the Driver. It consists of the make, model, variant and registration number of a car so that a customer may be able to easily identify the car thus acting as a security measure for the customer.

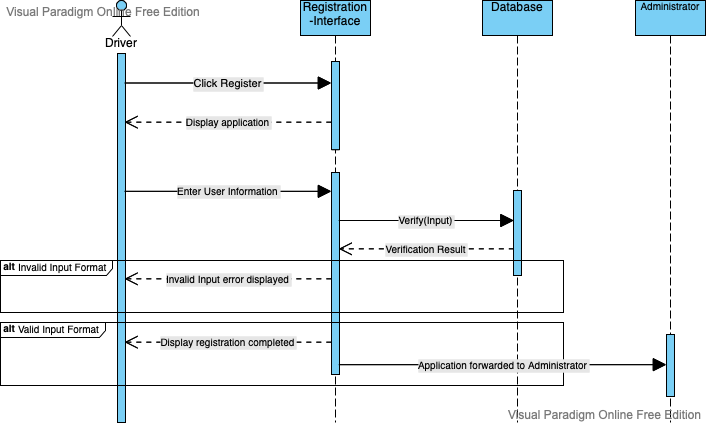
The payment class deals with the splitting of fares keeping an accounting balanace of the inflows and outflows of a ride. This will also be used to determine the cut of the company in a ride and provide incentives to drivers.

All classes comprise of a one to one relationship with each other except for Ride that has a one to many relationship with Customer and Payments as one ride may have multiple customers hence multiple payments.

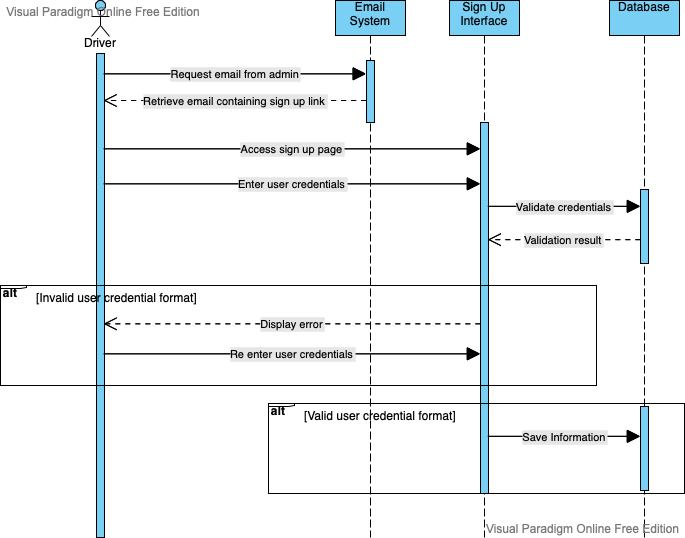
# Sequence Diagrams

The sequence diagrams must have classes from the Class diagram you created earlier in this document. These diagrams must show the interactions among the system classes. In all of the sequence diagrams that you have created, there is hardly any classes from your class model. There is also lack of consistency—multiple different formats have been used for these diagrams.

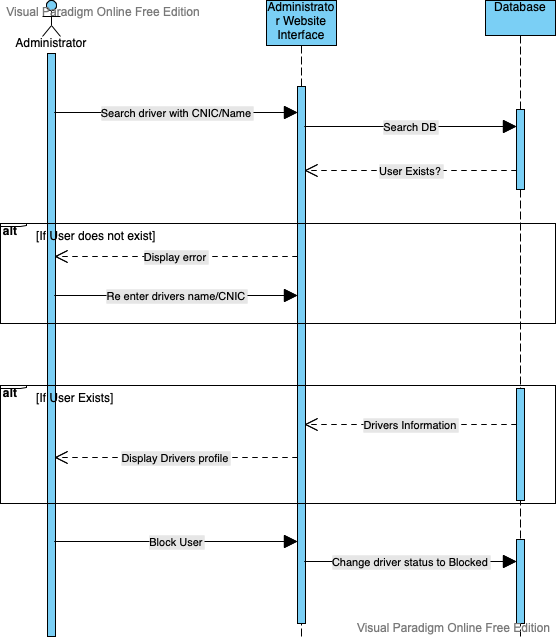
## Driver Registration UC- 001



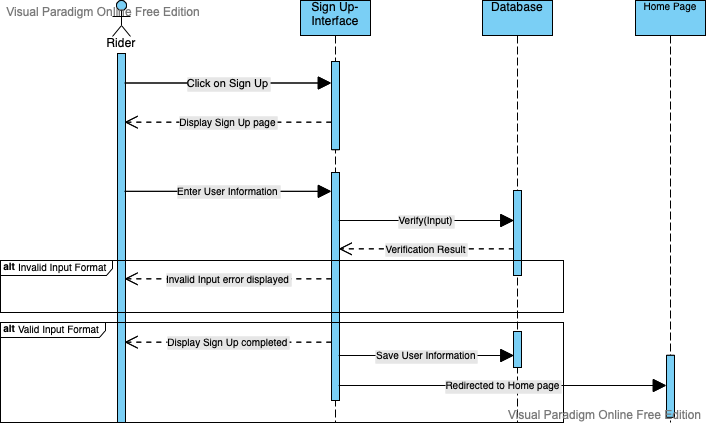
## Driver Sign Up UC-002



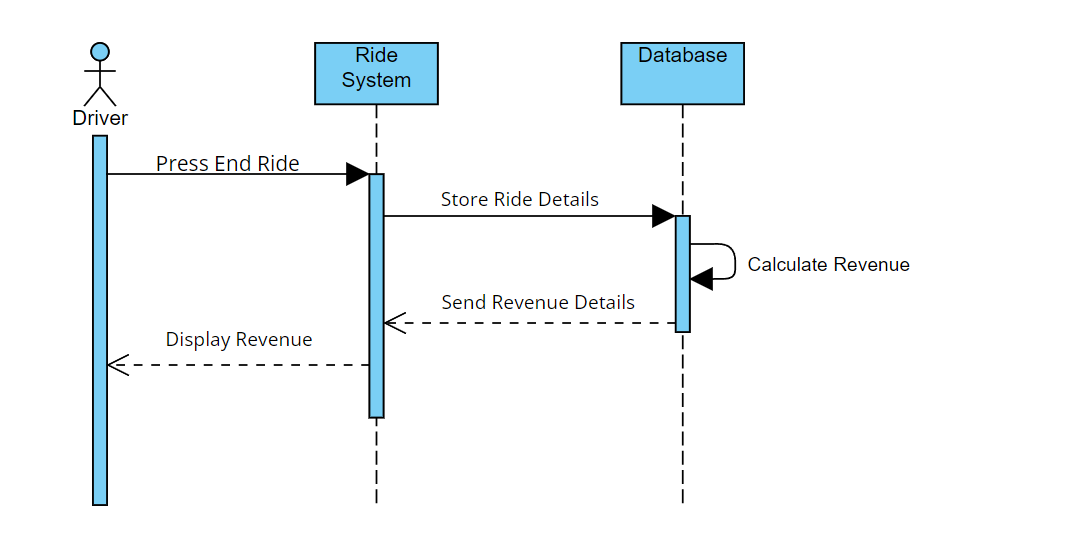
## Blocking a driver UC-003



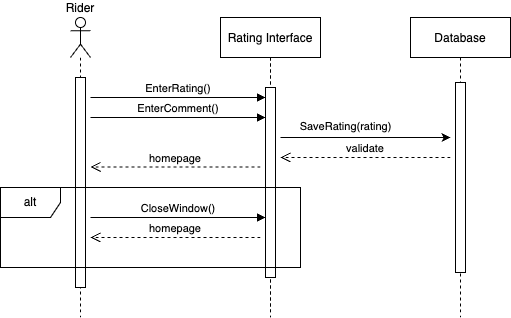
## User Sign Up UC-004



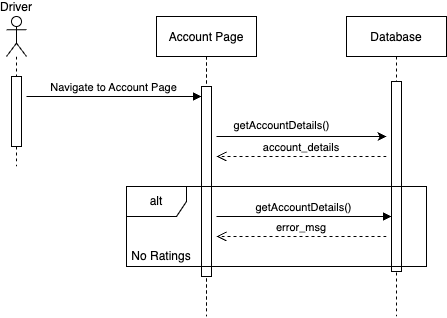
## Driver Views Revenue UC-008



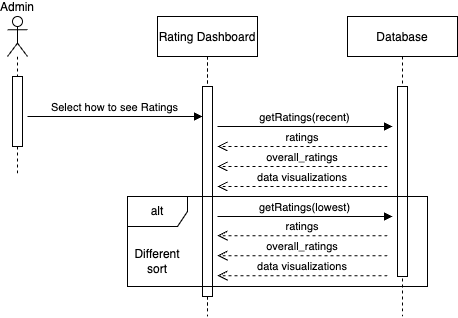
## User Rating Driver



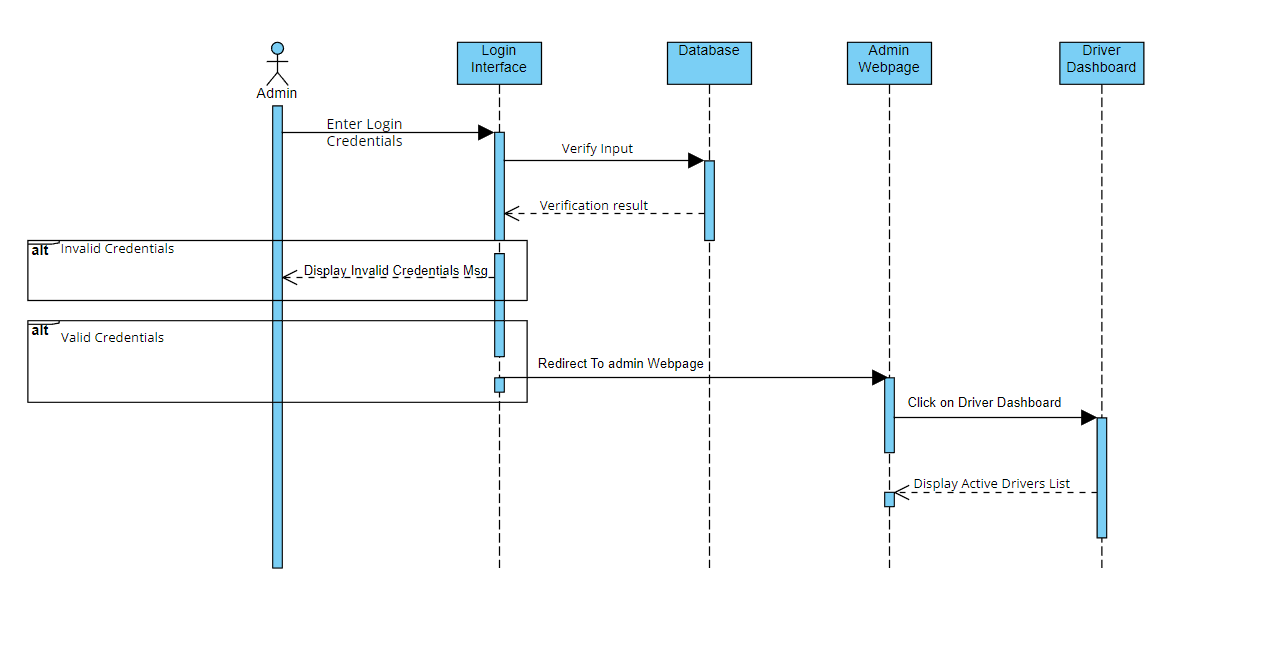
## Driver Views Rating



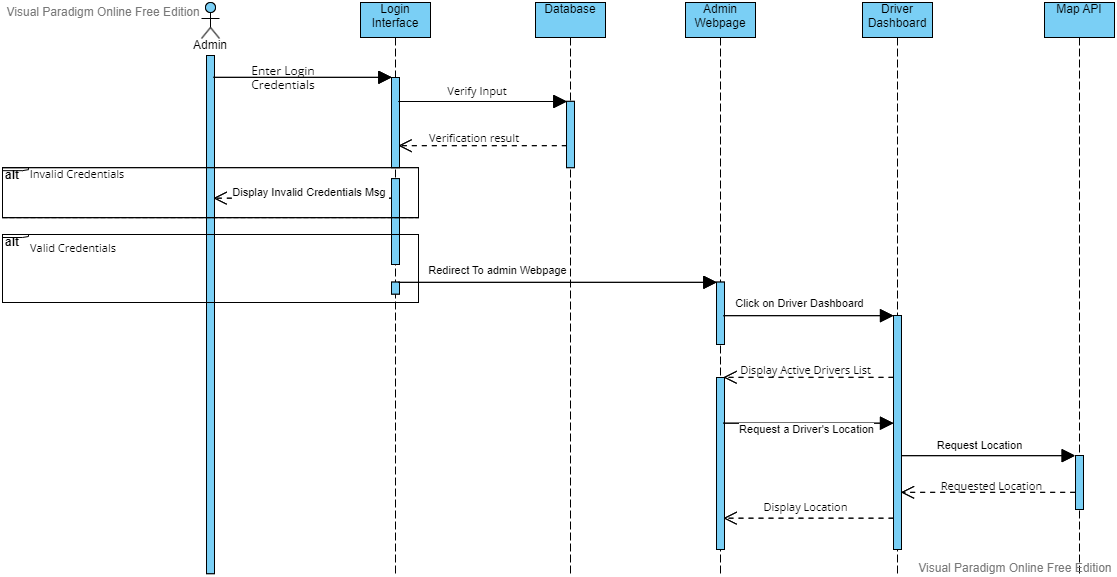
## Admin Views Rating



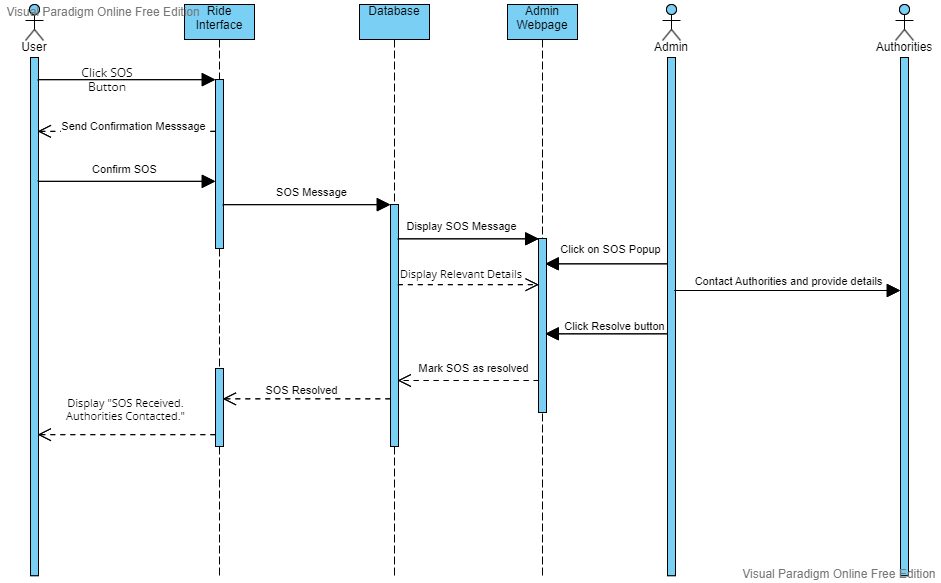
## Admin Views Active Drivers UC-009



## Admin Requests a Driver’s Location UC-010



## A User Sends an SOS Signal to the Admin UC-011

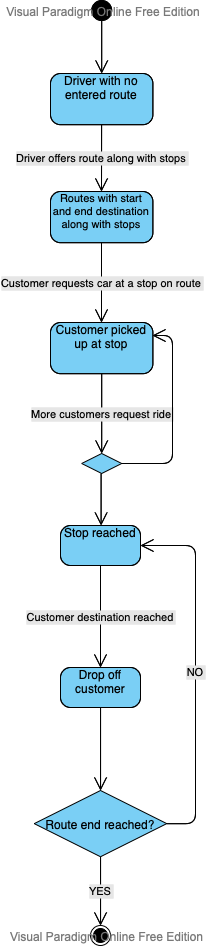


# State Diagrams

## Driver’s ride details

The following diagram shows the states of the ride as the status of the ride progresses. The state starts as the driver declares a route with stops and a start and end point. After the route is declared a customer can request a ride with pickup along the route and drop off along the route. The customer is picked up along the route and if there are more customers requesting the ride and the vehicle has available space, other customers are picked up as well and dropped off at their respective drop off location. The route ends once the end point has reached and all customers have been dropped off at their destinations.

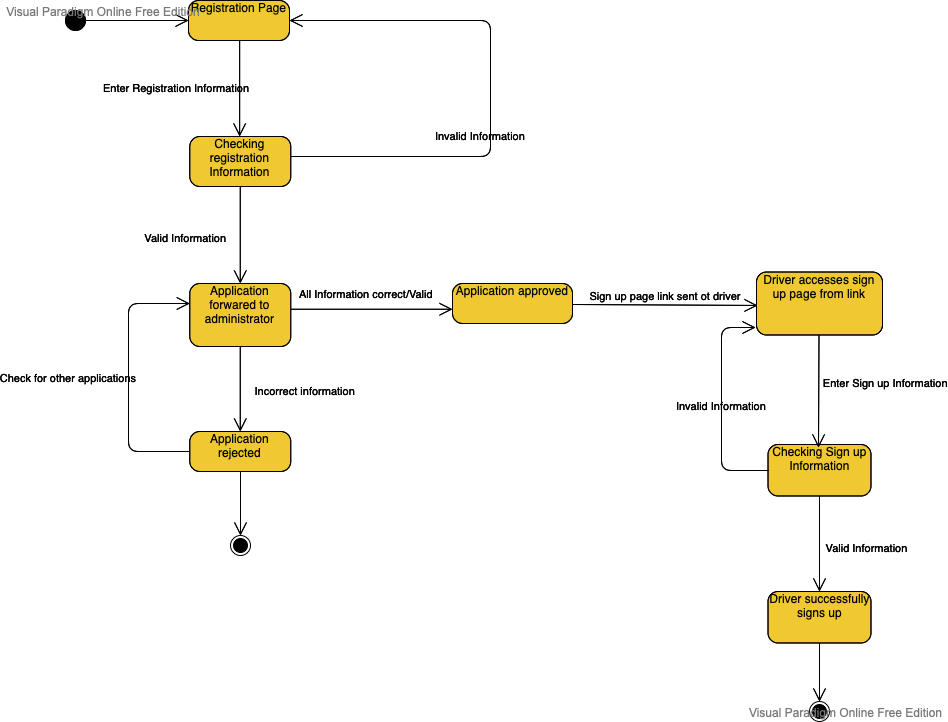
## Diagram



## Driver’s registration/sign up process

The following diagram illustrates the entire sign-up process for a driver. The first part of it is the registration process, which involves the user navigating to the registration page where they enter their personal information and provide documentation for their vehicle. After the application is submitted, it is forwarded to the administrator who reviews the application and approves or rejects it. If the application is rejected the state exits. If the application is approved the driver receives an email, which includes the link to the sign-up page. The driver can open the sign-up page and proceed to sign up by entering relevant information.

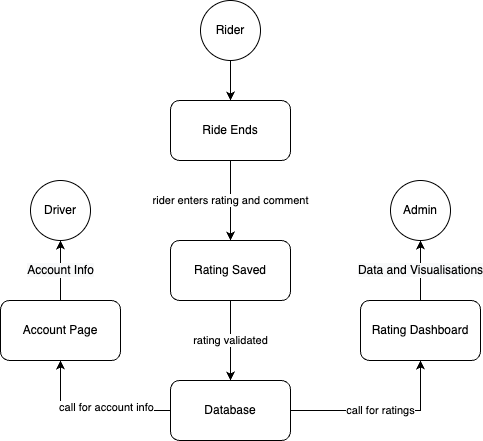
## Diagram



6.5 Ratings process

The ratings will be created by the user and attached to each driver, creating an overall rating this will be stored and updated in the database. Drivers and Admins can access this information through the Account page and Admin Dashboard.

6.6 Diagram



# Non-functional Requirements / Quality Attributes

<Requirements must be testable>

|  |  |
| --- | --- |
| **Sr#** | **Requirements** |
| 1 | The system should not utilize more than 1 GB of memory at any time during its execution.  <Do NOT write “*The system should utilize minimal memory*”> |
| 2 | The system should not fail more than 3 times every 24 hours. In case of a failure, the system should restore to normal operations within 5 minutes of a failure.  <Do NOT write “*The system should not fail often*”> |
| 3 |  |
| 4 |  |

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Irtza Tariq | Class Diagram and Class Di |
|  |  |
|  |  |
|  |  |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
|  |  |
|  |  |
|  |  |
|  |  |