

**REPUBLIQUE DU CAMEROUN**

**\*\*\*\*\*\*\*\***

**Paix-Travail-Patrie**

**\*\*\*\*\*\*\***

**UNIVERSITE DE BUEA**

**REPUBLIC OF CAMEROON**

**\*\*\*\*\*\*\*\***

**Peace-Work-Fatherland**

**\*\*\*\*\*\*\*\***

**UNIVERSITY OF BUEA**

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DEPARTMENT OF COMPUTER ENGINEERING**

COURSE CODE: CEF 440

COURSE TITLE: INTERNET AND MOBILE PROGRAMMING

**TASK 4. DESIGN OF THE PASSENGER POSITIONING SYSTEM.**

**Presented By:**

|  |  |
| --- | --- |
| NAME | MATRICULE |
| BALEMBA JUNIOR BALEMBA | FE20A021 |
| BEBONGNCHU YANNICK NKWETTA | FE20A022 |
| TABOT CHARLES BESSONG | FE20A106 |
| TANDONGFOR SHALOM CHANGEH | FE20A022 |
| YIMNAI NERUS ZAUMU | FE20A123 |

**SUPERVISED BY: :**

**Dr NKEMENI VALERY**

**INTRODUCTION**

After carrying out various tasks on mobile applications like finding out the major types of mobile applications and their differences, we went further to look at the programming languages used for mobile application as well as the very large number of frameworks available for mobile application development. More so, we looked on how we can collect and analyze the requirements for mobile applications. Considering we are building a mobile application for municipal commuting, we analyzed our system based on both the perspectives of the drivers and passengers.

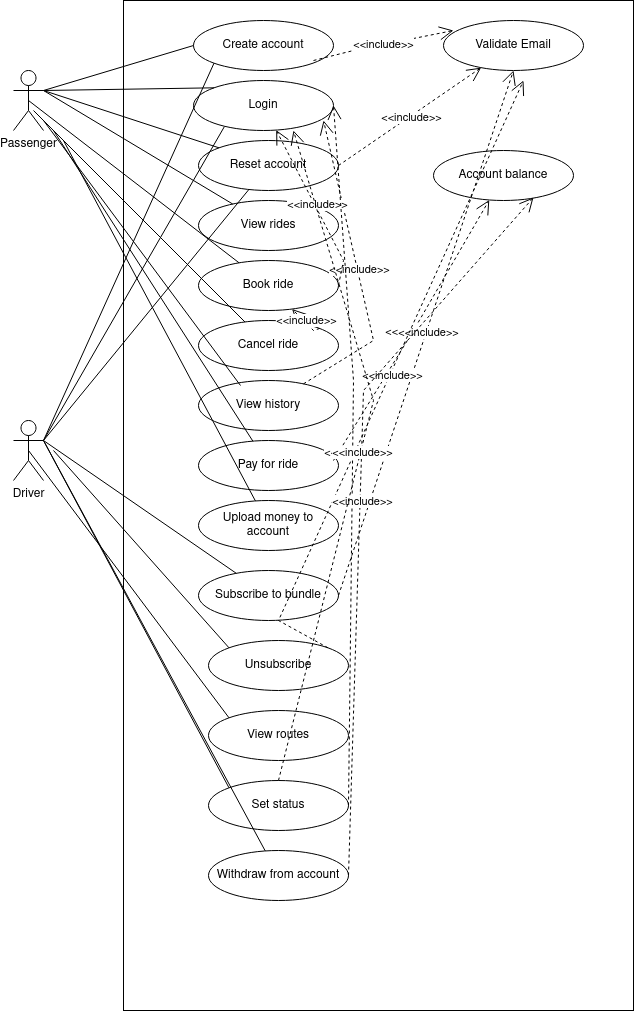
Design here is a drawing which is produced to show the function or workings of a system. On designing our system, we took into consideration all the analysis we gathered which we will elaborate on based on various UML diagrams. They include:

* Use case diagram
* Class diagram
* Sequence diagrams
* Activity diagrams
* Data flow diagram.

In our design, we made use of the waterfall model which has to do with breaking down of project activities into linear sequential phases where each phase depends on the previous one in order to function.

**Use case diagram:**

Our use case diagram here shows a graphical representation on how the various users of the system interact with the system. Below is the use case diagram of our system.

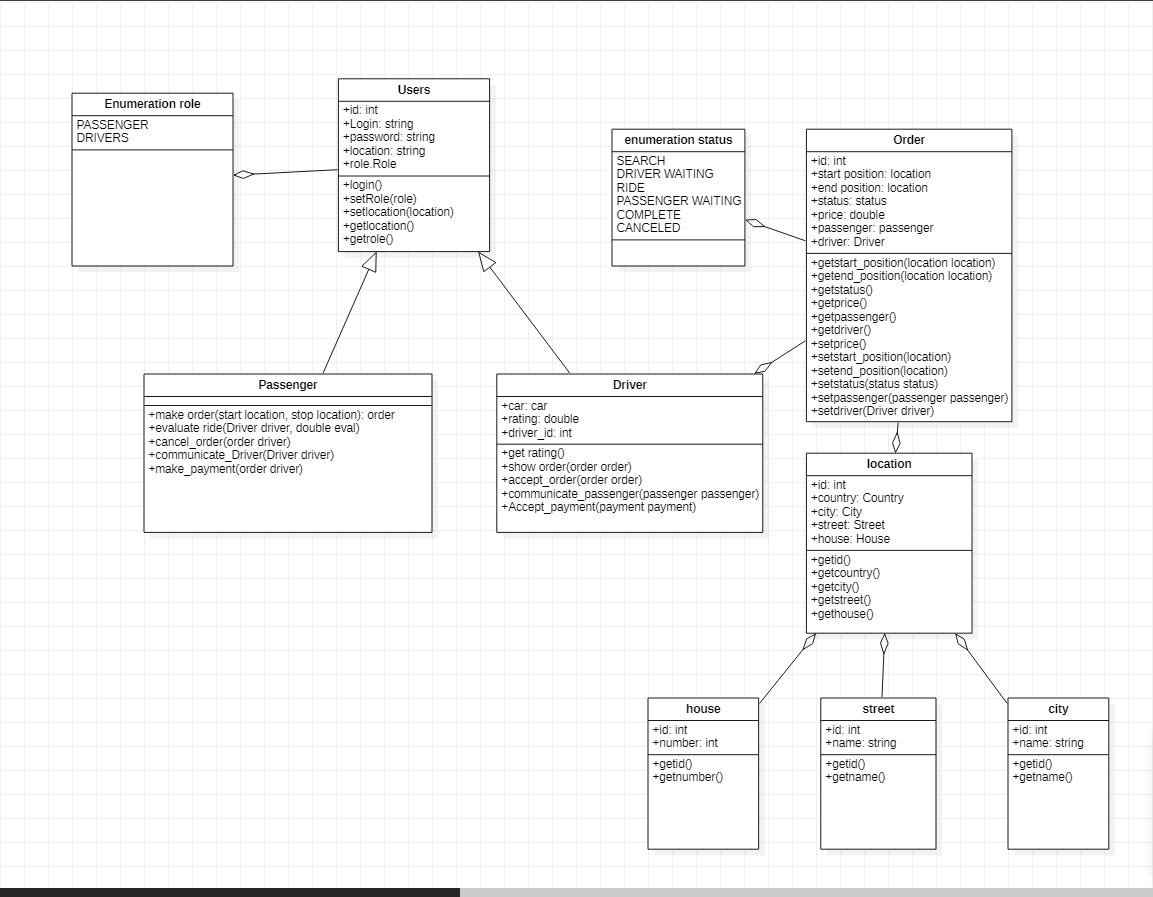


**Explanation:**

Our use case diagram shows the two main actors of the system which are the drivers and the passengers. Both the drivers and passengers have some common actions in which they both perform for example, create account, login, reset account. The drivers and passengers also have actions in which only them perform, for example the driver can subscribe to bundles, unsubscribe, view route etc. as well as the passenger can book ride, cancel ride, view history etc. From the diagram above, the dotted arrows with include shows operations which can not take place unless a different operation has already been done. For example a passenger cannot validate email without creating an account, cannot view history without login in etc. also the driver cannot unsubscribe from a bundle without first subscribing, cannot view his profile without login in etc.

**Class Diagram:**

A class diagram is the main building block of object oriented modeling. We model our system with a class diagram to show the relationship between our various classes and the services that they provide to our system. Below is the class diagram of our system.



From the above diagram, we designed our system to have 8 classes with some of them having relationships with others. Based on our design, we have our class users. The user class has certain attributes like id, login password, location and role. This user class can perform functions which can be seen on the above diagram. Our system is basically designed to support two types of user, which therefore leads us to a generalization which exist between the user class and its sub two classes which are the passenger class and the driver class.

The driver and the passenger automatically inherits all the attributes and functions which the user class has and also has other additional attributes and functions.

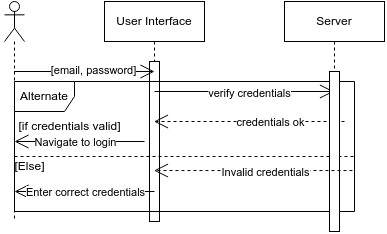
We also have a class order with its attributes as well as its operations. It forms an aggregation relationship with the class location which means the class order is a subordinate to the class location as well as house, street and city and subordinates to the class location.

**Sequence Diagrams:**

A sequence diagram is a**diagram that shows how objects interact with each other and what messages they exchange over time. For our system, we made sure that each use case has its own sequence diagram to show the interactions between its objects. This can be seen in both the passenger and the drivers perspectives.**

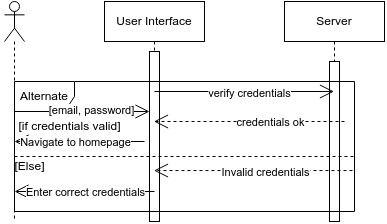
**For the passengers use cases, we have the following:**

* ***Signup Sequence***



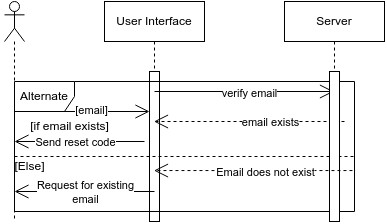
**Here, the system ask the user to enter certain credentials which are his email and password, now the system checks if the information he entered was valid that is in the right way, if valid, the user navigates to login and if not valid, the systems pops up a message for the user to enter correct credentials.**

* ***Login sequence***

******

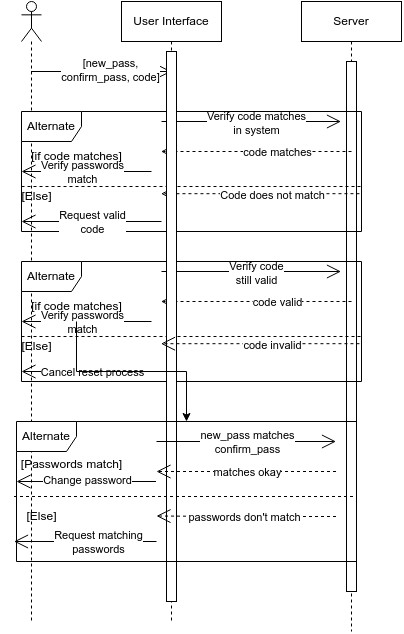
**From the diagram above, we can see the interaction between the user and the server. It’s seen that the user enters their credentials and if credentials are correct, moves to the homepage of the app while if not correct, an error message of invalid credentials pops up. The user then tries again.**

* ***Forgot sequence***



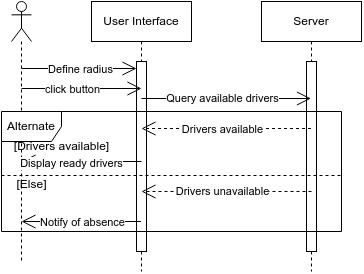
**If for example a user forgets his/her credentials such as password, this user will be sent a reset code if the email is valid and if not valid, the system will request for a valid email.**

***Reset password:***



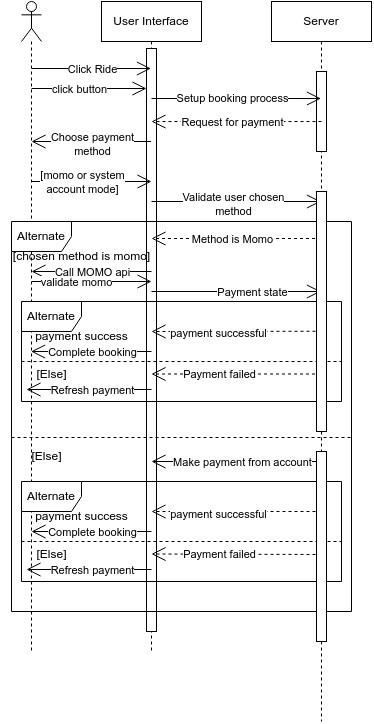
**After the system sends a confirmation code to the user, he then enters the code to very, if the code does not matches, the system tells the user that the code he entered isn’t valid, and ask him to request for a valid code and if the code doesn’t still match, the system cancels the reset process. On the other hand if the code matches, then the system provides a password to the user which he then uses.**

***View ride sequence:***



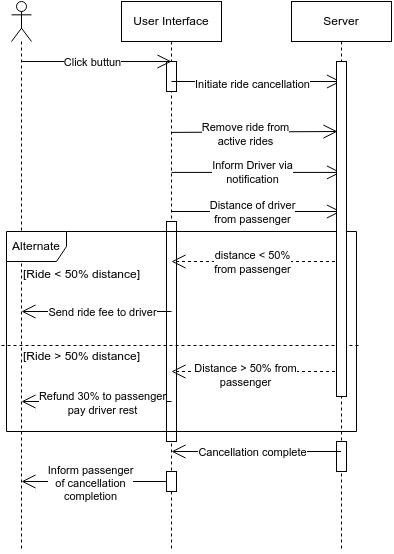
**The user has to select the estimated distance he/she want to go then click the view ride button to see if there are any drivers for that direction. The system then tells the user that there are available drivers but if not, it still sends a message saying no available drivers.**

***Book ride sequence:***



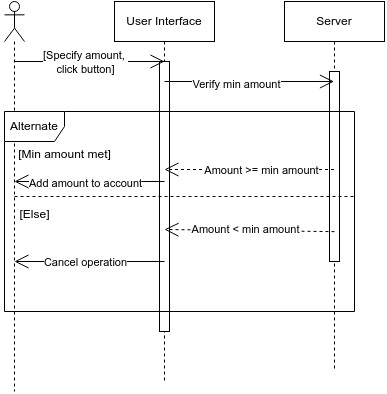
**The system is designed in a way that it has a button on which the user can click to book ride, after which he makes payment either through the system or MoMo. If payment was successful, the system pops a payment validated message and if on failure, a payment failed message which the user can refresh and try to pay again.**

* ***Cancel Ride:***



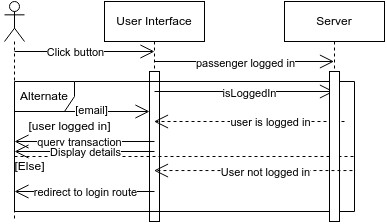
Our system has a cancel ride button which when click, the passengers ride is canceled based on few conditions. If the driver has already covered more than 50% to where the user is, the ride fee is sent to the driver but if the driver has rather covered less than 50%, 30% of the fee is sent to the user and the rest to the driver. The system then informs the user that his cancellation was successful.

* ***Upload money sequence:***

******

When the user want to upload money to his/her account, the user specifies the amount to upload and if the amount is greater or equal to minimum amount, the user adds the amount to account but if the money isn’t greater than minimum amount, the operation is cancelled.

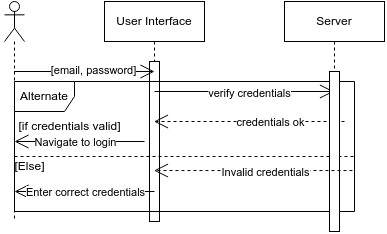
***Passenger history sequence:***



For the passenger or user to view their history, the user clicks the view history button and if the user has already logged in, he can view his transactions details and if user not logged in, the system redirects him to the logging page.

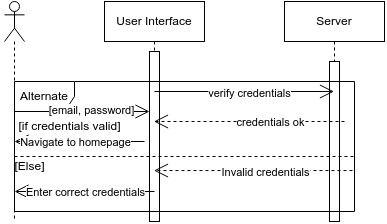
For the drivers use cases, we have the following sequence diagrams:

***Creates Account sequence:***

******

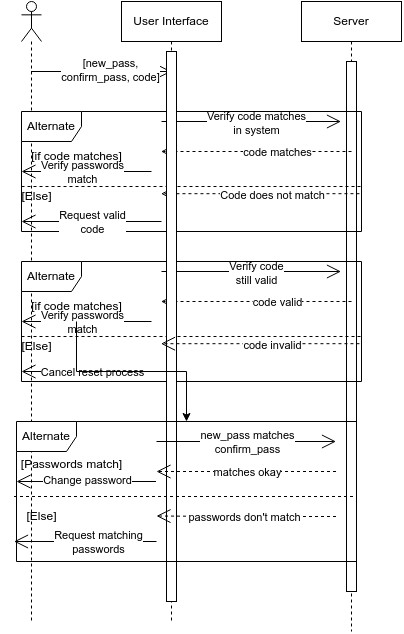
**The system ask the user to enter certain credentials which are his email and password, now the system checks if the information he entered was valid that is in the right way, if valid, the user navigates to login and if not valid, the systems pops up a message for the user to enter correct credentials.**

***Login sequence:***

******

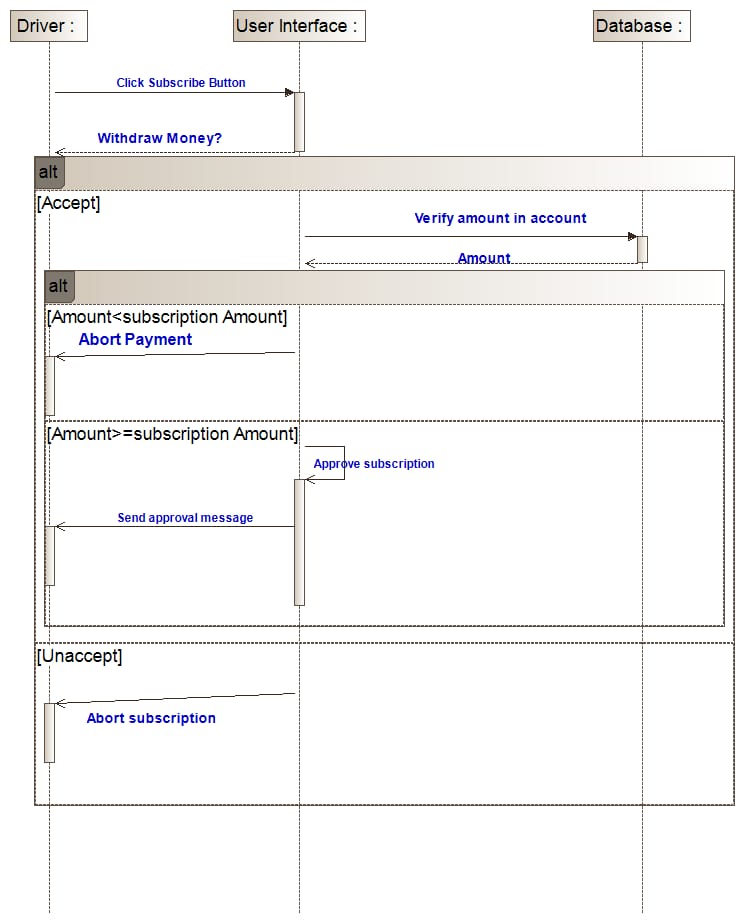
**The user enters their credentials and if credentials are correct, moves to the homepage of the app while if not correct, an error message of invalid credentials pops up. The user then tries again.**

***Reset password sequence:***

******

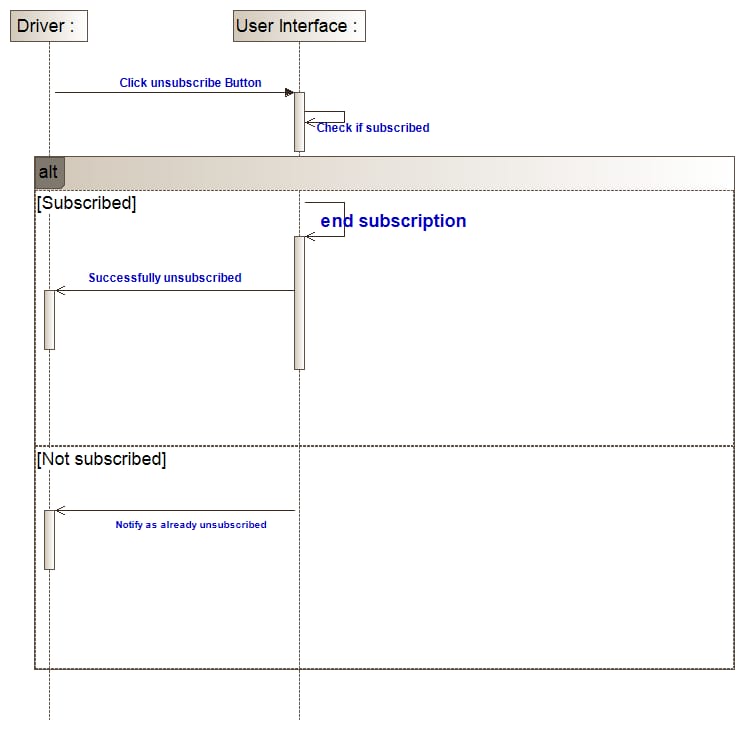
**After the system sends a confirmation code to the driver, he then enters the code to very, if the code does not matches, the system tells the user that the code he entered isn’t valid, and ask him to request for a valid code and if the code doesn’t still match, the system cancels the reset process. On the other hand if the code matches, then the system provides a password to the user which he then uses.**

***Subscribe to bundles sequence:***

******

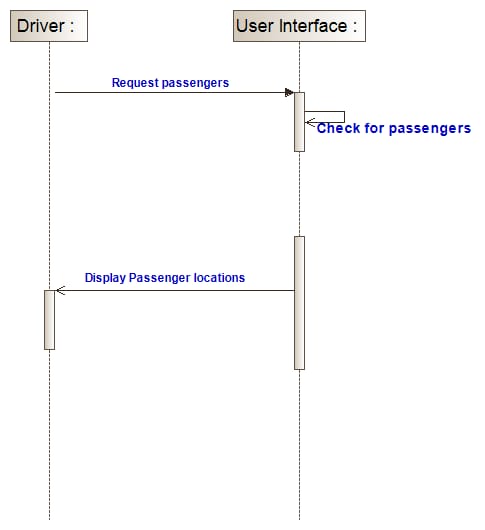
The driver clicks on a subscribe button if he’s interested on a particular bundle. After clicking the system take system ask for permission to withdraw his subscription fee from his account. If he accepts then the system withdraws and displays a success message and if transaction not successful due to insufficient fund, the system displays a failure message.

***Unsubscribe sequence:***

******

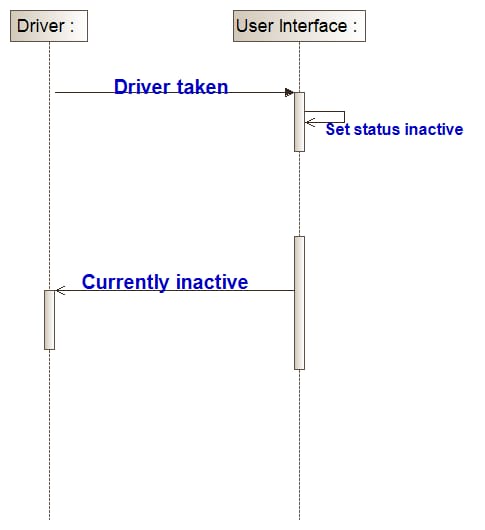
When the driver clicks the unsubscribe button, the system checks if he’s subscribed to that bundle and if true, it grants his request and displays unsubscribed.

***View routes sequence:***

******

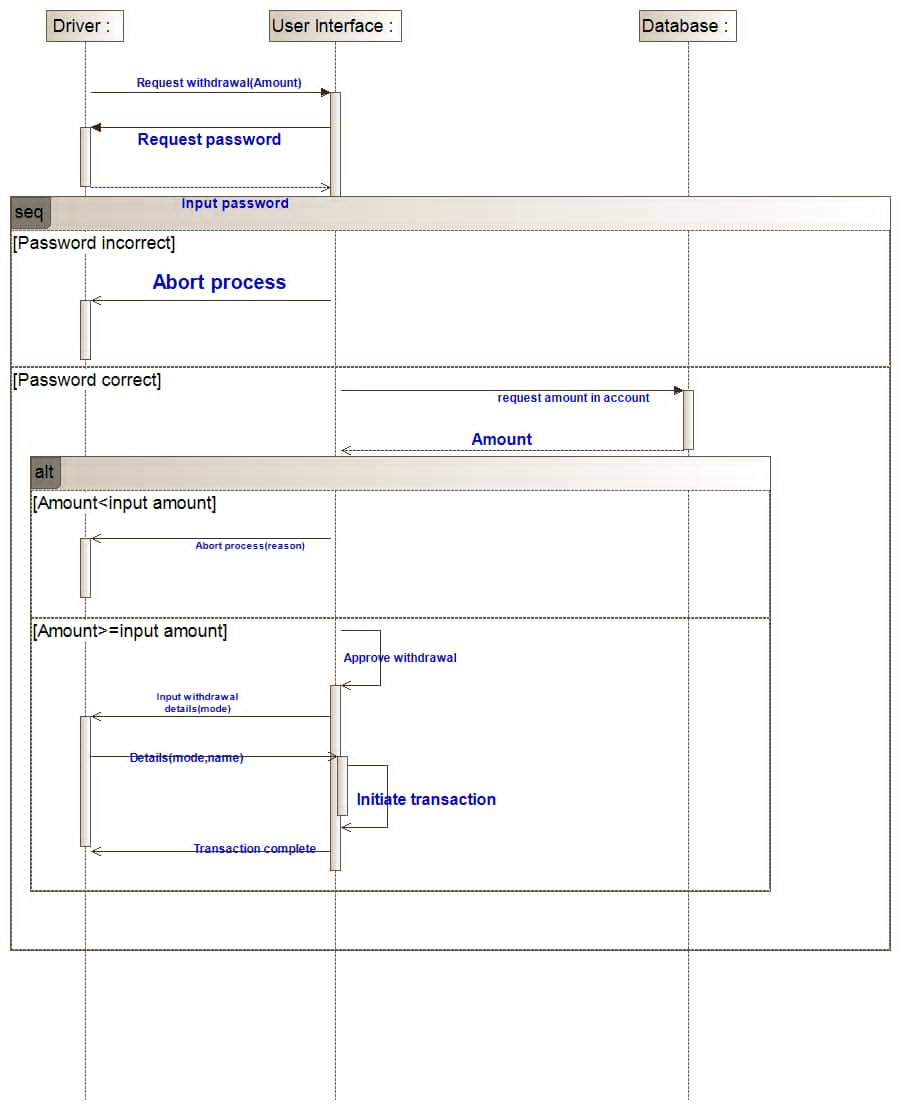
The driver checks the position in which there is a high concentration of passengers then the system simple displays the location.

***Set status sequence:***

******

If a has already been booked, he sets his status and inactive until after he’s done with that ride, then he can change his status to active.

***Withdraw from account sequence:***

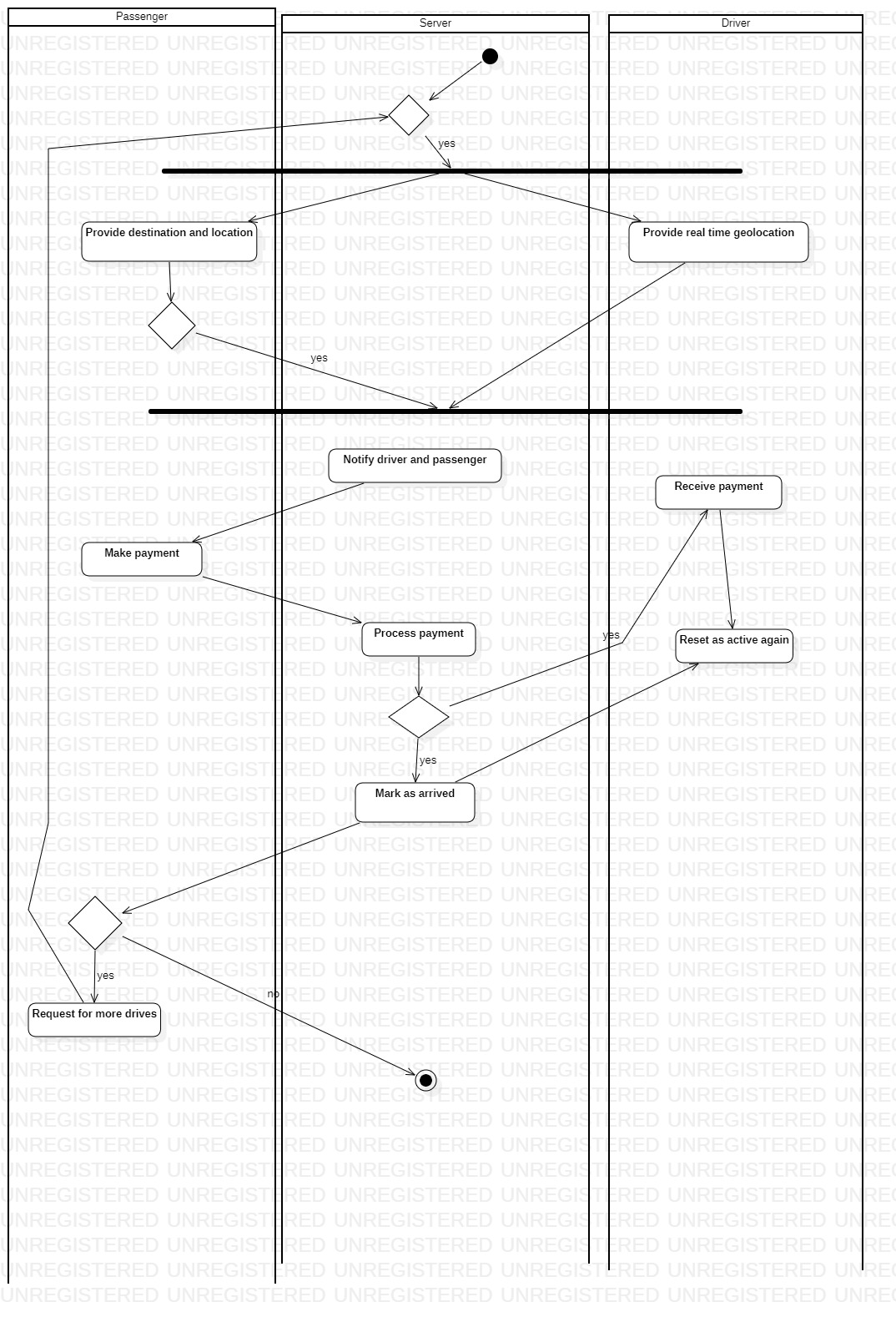
******

The driver request for withdrawal and the system sends him a password and ask him to input the password, if password incorrect, the system aborts the process but if correct, the driver inputs the amount to be withdrawn. If the amount he entered is not up to what is in his account, the system aborts the process but if it’s enough, the system approves withdrawal and after it’s done, sends a transaction complete message.

**Activity Diagram:**

An activity diagram provides a view of the behavior of the system by describing the sequence of actions in a system.

Based on our municipal commuting system, there are verities of actions taking place performed by the use cases in the system. Below is the activity diagram for our system.



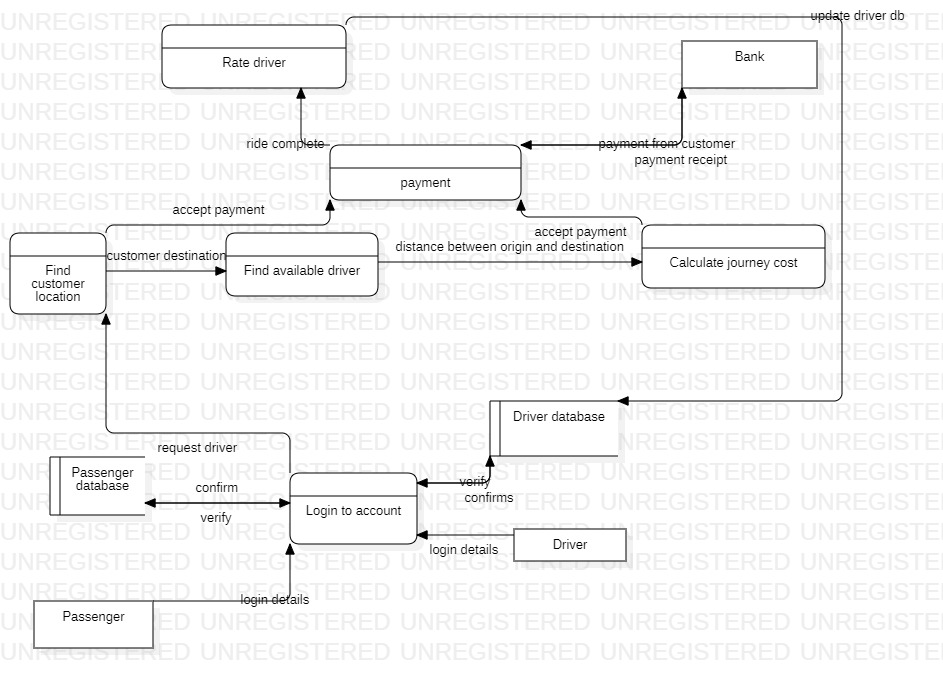
Our system has mainly three actors which are passenger, server and the driver.

No action is done until when a user comes and performs an action. When a passenger gets into the system and books a ride, the server gets the location of the passenger and also get the drivers that are available at that particular location. If there are passengers available at a particular area, it sends a signal to the driver that there are passengers at this location. When the passenger initiates the payment, the server checks if the amount you’ve entered is greater than the minimum amount. If its greater, the system withdraws from the users account and sends a success message and if not, terminates process.

Now if the user reaches his destination and wishes to take book another ride to somewhere else, the system restarts all the entire process again but if no, process terminates.

**Data Flow Diagram:**

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. For our system, we have



When the passenger logs in, his information goes into the database and its checked it this user already exist in our system, if true, then login but if the user doesn’t exist, then error message. When the passenger request for a driver, the system checks for available drivers for that direction and if available, the system calculates the cost to that destination and sends to the passenger to make payments and after the passenger makes the payment, a receipt is sent to both the passenger and the driver that this person has made payments.

For the driver, he logs in too and his credentials are checked and if valid, access is granted and if, error signal. Receives his money on any payment and then services the ride.