COURSE PROJECT

OBJECT ORIENTED ANALYSIS AND DESIGN COURSE

Instructor: Dr. Truong Ninh Thuan

Project team:

- Ly Van Tuan Student.No: 14020513 K59CLC
- Bach Van Thuan Student.No: 14020611 K59CLC
- Tran Xuan Dat Student.No: 14020102 K59CLC
- Nguyen Viet Long Student.No: 14020669 K59CLC
- Tran Viet Tiep Student.No: 14020470 K59CLC

Requirement: Give the analysis and design documentation of a startup software

Contents

1. Uber System Requirements

1.1. Problem statement

Recently, people in Vietnam often mention Uber as a new way to travel in the city or travel to other areas. Uber is similar to traveling by taxis but it's much more convenient for both drivers and customers, we'll discuss about this later. Uber company doesn't even have a car and that's a special thing about it. Drivers can use their own cars when they register to work as Uber's drivers. This company became one of the private companies that has the highest income with total income up to 50 million USD. What makes this company so famous and successful? That's because the company has been known to use information technology in its services strategy. Uber software provides a connection between customers and drivers. With this application, customers can quickly get a ride in the shortest time, at the same time drivers can easily get information about the locations of the nearest customers. Uber will get 20% discount for each trip. Uber company provides services for both web and mobile application. Uber website has an own version for Vietnamese people : https://www.uber.com/vi-VN/, everyone can easily registers and uses it. Uber system splits users into 3 parts: drivers, customers (passengers), system administrators.

First, users have to register for accounts to use the system. After successful registration, users can login and use this application. The application allows customers to send requests to the system when there are demands. The system will accept requests from customers and then customer management system will save information about customers (names, locations, destinations,...). Based on the locations where the requests has been sent, the driving management system will find drivers who are nearest to each customer. The application will inform passengers about the driver's name, car number plate, the type of car,.... at the same time inform drivers about passengers' information.
The drivers also received notification from the system about customers, if a driver is in free mode, he/she can come to the customer's location and make the trip. When drivers accept the trip, the system will change the drivers' status.

After the end of the trip, customers can pay by cash or by bank cards. Uber also has a system called Billing system.

Uber also has the trip management system. An important feature of this system is that Uber can provide information of the nearest customers, drivers so that the services can be done faster, easier. All system users are able to login and change personal information. In addition the system also has a two-way assessment

system, the clients can check information about the drivers and vice versa to promote responsibility community.

1.2. Glossary

Introduction

This document is used to define specific terminology to the problem domain, explain terms - which may be unfamiliar to the reader of the use-case description or other documents. Often, this document can be used as an informal data dictionary, capturing data definitions so that use-case description and other project documents can focus on what the system must do with the information.

Definition

The glossary contains the working definitions for the key concepts of the Uber Manager systems.

Manager

A person, who manages systems, manages all activities of drivers, information of passengers, details of trips.

Trip

A journey which means that you go somewhere.

User

An account belongs to the system. It may represent a passenger, driver, or manager.

Passenger

A person who pays for a trip

Cash

Money that people use to pay directly for something

Bill

A record which contain price, history of journeys.

Billing system

The component that can access, query and process on database of bills.

Statistic Data

Data given as a result of querying the database.

Promotions

Activities done in order to increase the sales of a product or service; a set of advertisements for a particular product or service

Rating

Countable a measurement of how good driver, passenger is.

Driver

The persons are drive their cars or their motorbikes.

1.3. Supplementary Specification

Objectives

The purpose of this document is to define requirements of Uber System. This Supplementary Specification lists the requirements that are not readily capture in the use-case model. The Supplementary Specification and the use-case model together capture a complete set of requirement system.

Scopes

This Supplementary Specification was wrote base on Uber system. It was analyzed and designed by OOAD students.

References

Course project of K55 CA given by instructor.
IBM Rational Software Documentation (Version 2004)

Functionally

Driver is easy to find passenger have demand and Passenger call a trip is faster.

Usability

The new user can learn how to use system within 30 minutes. The use interface has nice and clear.

Reliability

The system will be available 24 hours a day 7 days a week, with no more than 2% down time

Performance

The latency of get statistic data must be less than 10 seconds and that one of other operations are less than 2 seconds. The GUI transitions must be smooth. No error in the system.

Supportability

None

Security

Almost changes of the system databases can only be done by the manager. Require confirm password before submit the changes.

Design Constraints

The system will provide both web application interface and Mobile application Interface.

1.4. Use-case model

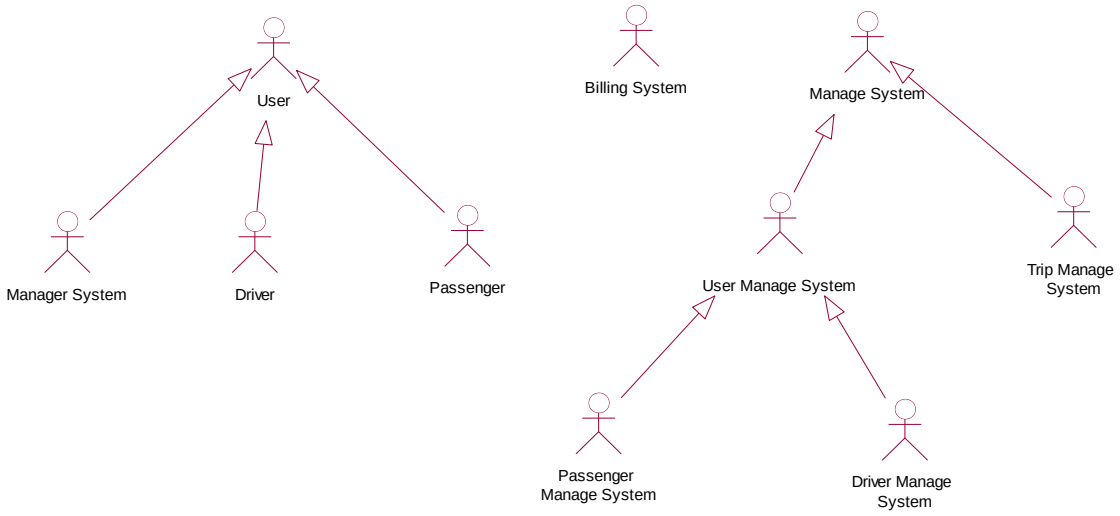


Figure 1.4.0.1 Uber actor and System Model

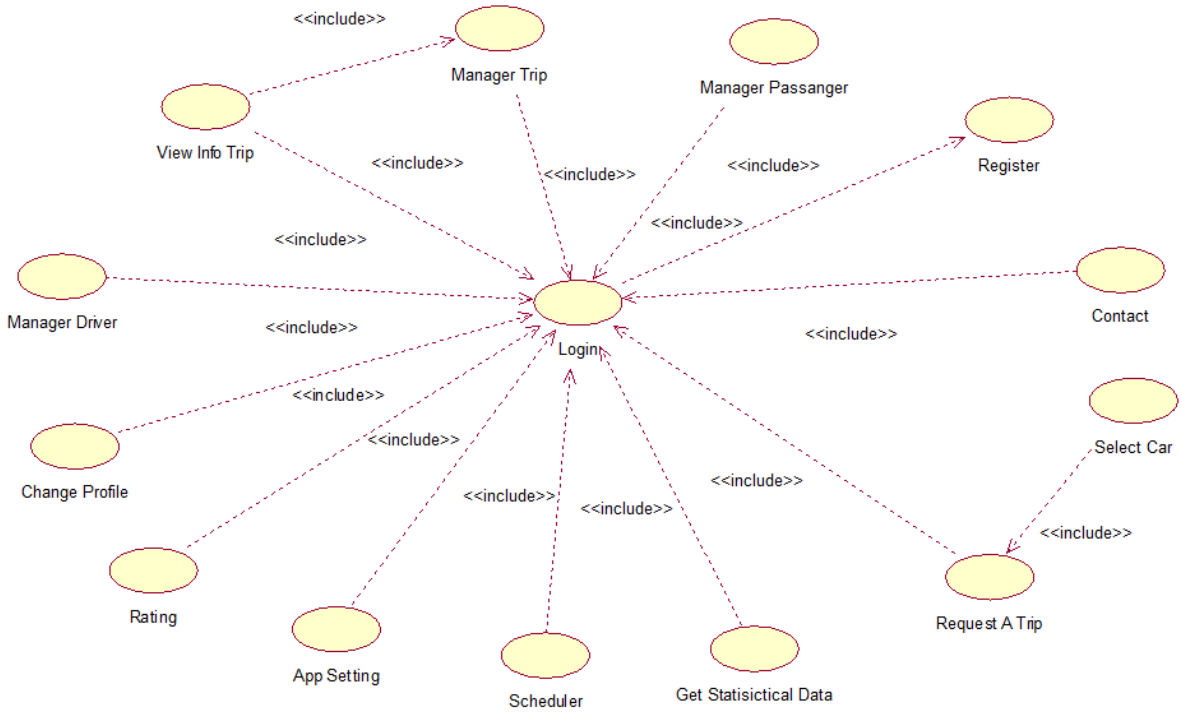


Figure 1.4.0.2: The use case dependencies

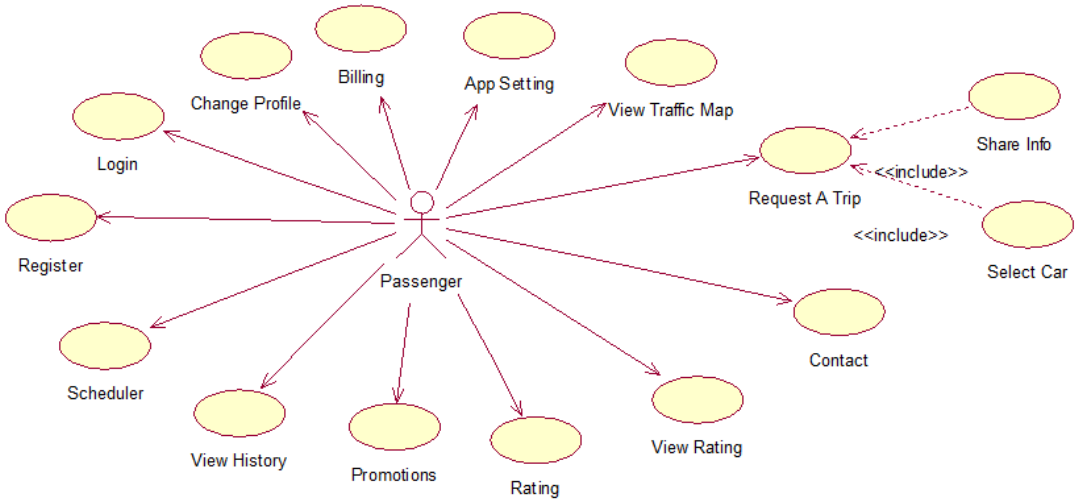


Figure 1.4.0.3: The use-case model of Passenger View

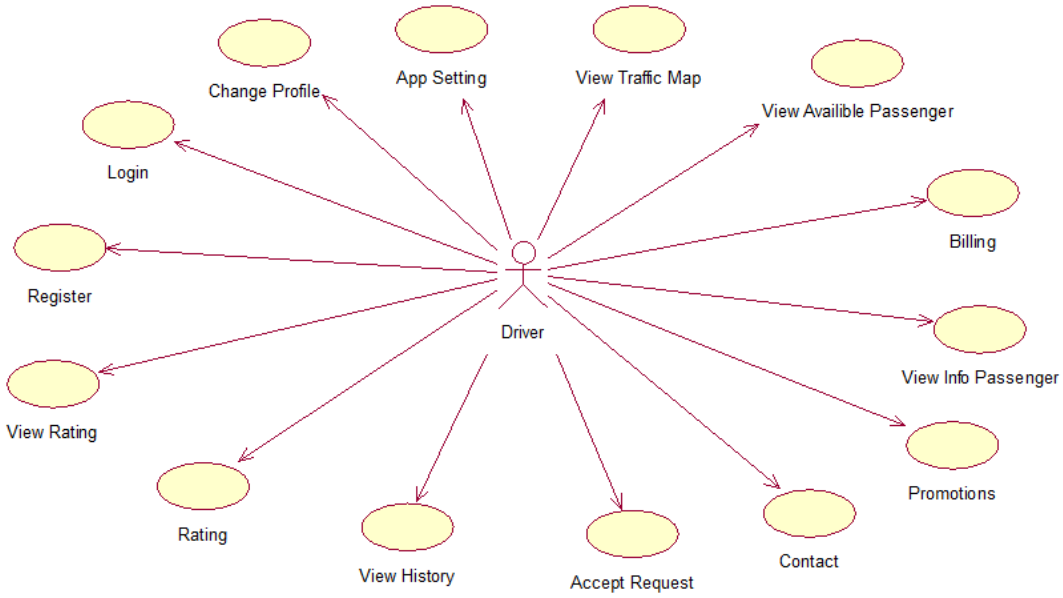


Figure 1.4.0.4: The use-case model in Driver View

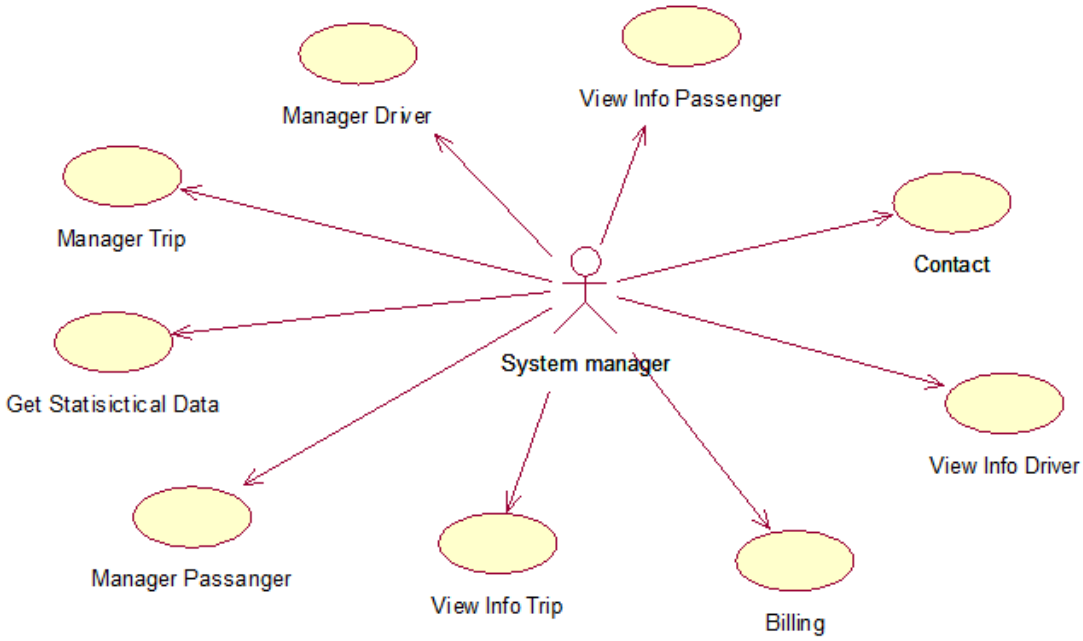


Figure 1.4.0.5: The use-case model in System manager view

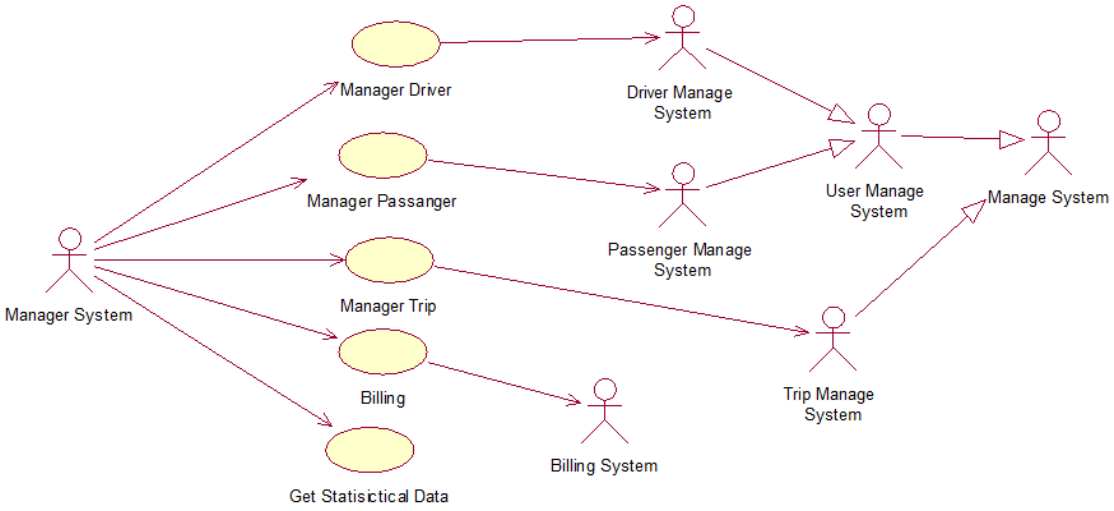


Figure 1.4.0.6: Use-case model in the view of integrated with subsytem

1.4.1. Login

Brief Description

This use case describes how an user logins the Uber system.

Flow of events

Basic flow

1.Actor enters his/her email or phone number, password or login with Facebook, Google account.

2.The system validates the entered email or phone number, password or Facebook, Google account.

3. System informs actor that he/she successfully logins to system and allows actor to use the system.

Alternative flow

Invalid Email/Password or Invalid facebook account.

If, in the *Basic flow***,** the actor entered a wrong email/phone number/password or an unregistered Facebook, Google account, the system displays an error message. The actor can choose to return to the beginning of *Basic flow* or cancel the login step.

Special Requirements

None

Pre-Conditions

Actor had a registered account

Post-Conditions

If use case was successfully done, the actor can use the system. If not, the system is unchanged.

Extension Points

None

1.4.2. Change profile

Brief Descriptions

This use case describes how an actor change his/her information

Flow of events

Basic flow

This use case starts when the actor wants to change something in his/her profile.

1.The system retrieves and displays the information of the actor.
2. The actor makes his/her changes.
3.The system saves the changes and then updates the actor's information in database.

Alternative Flows

Invalid Inputs

If in the *Basic Flow*, the actor entered invalid inputs, the system displays an error message and highlights the positions where the errors occur. The actor can choose to go back to the beginning of the *Basic flow* or cancel the use case.

Special Requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to change his/her profile.

Post-Conditions

If the use case was successfully done, the actor's information is updated. If not, the system is unchanged.

Extension Points

None

1.4.3. View Info Passenger

Brief Description

This use case describes how an actor views a passenger's information

Flow of events

Basic flow

1. Actor chooses to view a passenger's information
2. Actor chooses the passenger he/she wants to view
3. System retrieves information of that passenger and displays it

Alternative flows

None

Special Requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to view a passenger's information

Post-Conditions

If the use case was successfully done, the actor can view the passenger's information. If not, the system is unchanged.

Extension Points

None

1.4.4. View Info Trip

Brief Description

This use case describes how an actor views the information of a trip

Flow of events1

Basic flow

1. Actor chooses to view a trip's information
2. Actor chooses the trip he/she want to view
3. The system retrieves information of that trip and displays it

Alternative flows

None

Special Requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to view a trip's information.

Post-Conditions

If the use case was successfully done, actor can view that trip's information. If not, the system is unchanged.

Extension Points

None

1.4.5. View Notification

Brief Description

This use case describes how actor can see a notification.

Flow of events

Basic flow

1. Actor chooses to view the notifications menu
2. System displays the notifications menu
3. Actor chooses the notification which he/she wants to see
4. System retrieves and displays the notification's details

Alternative flows

None

Special requirement

If the actor is a driver, the system shows notifications about the passengers. If the actor is a passenger, the system shows notifications about the drivers. If the actor is a system manager, it displays notifications of all drivers and passengers.

Pre-Conditions

Actor successfully logins to the system and chooses to view notification

Post-Conditions

If the use case was successfully done, actor can view the notification's details. If not, the system is unchanged.

Extension points

None

1.4.6. Enter Pickup Location

Brief Description

This use case describes how a passenger enters the pickup location

Flow of events

Basic flow

1. System shows a map

2. Passenger can enter his/her location to the input field or he/she can choose a location on the map (If it is a famous location such as a temple, pagoda, hotel,...)

3. System sends notifications to drivers and system managers. The notifications contain information of the location

Alternative flows

Invalid location

If the passenger enters an invalid location to the input field, the system displays an error message. The passenger can start again from the beginning of *Basic flow* or cancel the use case.

No GPS

If GPS was turned off while actor uses this function then the system will display a message. Actor can turn on GPS and start again from the beginning of *Basic flow* or cancel the use case.

Special requirement

GPS on.

Pre-Conditions

Passenger successfully login to the system and chooses to enter a pickup location.

Post-Conditions

If the use case was successfully done, drivers and system managers can get notifications. If not, there ‘ll be no new notification for drivers and system managers to see.

Extension points

None

1.4.7. Billing

Brief Description

This use case describes how the billing steps happen.

Flow of events

Basic flow

1. Actor (A passenger) chooses the billing method

2. System displays the price for the trip

3. Passenger enters information about his/her bank card (if he/she chooses to pay by bank cards)

4. Billing steps' information save to the database

Alternative flows

Invalid card information

If passenger enters an invalid information of the bank card, system displays an error message. He/she can starts again from the beginning of *Basic flow* or cancels the use case.

Not enough money in the bank card

If passenger doesn't have enough money in the bank card, system displays a message and the use case starts again from step 2 of *Basic flow*

Special requirement

System displays specific interfaces for specific actors. For example, if the actor is a driver, the Billing interface don't have to contain the payment method section because he/she doesn't have to pay for anything.

Pre-Conditions

Actor successfully logins to the system

Post-Conditions

If the use case was successfully done, a bill record is updated to the database. If not, there is no new bill record in the database.

Extension points

None

1.4.8. View available passengers

Brief Description

This use case describes how a driver can see available passengers

Flow of events

Basic flow

1. Driver chooses to see available passengers

2. System displays information of the available passengers. If there is no available passenger, it displays an empty menu.

Alternative flows

None

Special requirements

GPS on.

Pre-Conditions

Driver successfully logins to the system

Post-Conditions

If the use case was successfully done, the system can see available passengers. If not, the system remains unchanged.

Extension Points

None

1.4.9. Register

Brief Description

This use case describes how an actor can register an account to use the system

Flow of events

Basic flow

1. Actor fills the register form

2. System validates the information given by the actor

3. System saves information of actor into database

4. System informs actor that he/she successfully registers his/her account

Alternative flows

Invalid input

If actor enters invalid information, system displays an error messages and actor can start again from the beginning of *Basic flow* or cancel the use case.

Special Requirements

None

Pre-Conditions

None

Post-Conditions

New account added to the database

Extension Points

None

1.4.10. Get statistic data

Brief Description

This use case describes how an actor (system manager) can get statistic data such as most favorite streets, average time drivers spend a day to drive passengers,... for further strategy.

Flow of events

Basic flow

1. System displays every kinds of statistic data

2. Actor chooses the kind of statistic data he/she wants to view

3. System collects data from database and display the results to the actor

Alternative flows

Empty database

If the database is empty, system displays a message to inform actor. The actor can choose to get another kinds of statistic data or cancel the use case.

Special requirements

Actor gets the results within 10 seconds

Pre-Conditions

Actor successfully logins to the system

Post-Conditions

If the use case was successfully done, actor can get the results within 10 seconds. If not, the system remains unchanged.

Extension points

None

1.4.11. View info driver

Brief Description

This use case describes how an actor can view information about drivers

Flow of events

Basic flow

1. Actor chooses to view information about drivers
2. Actor chooses the driver he/she wants to view
3. System retrieves information about that driver and displays it

Alternative flows

None

Special requirements

Actor can get the result within 5 seconds

Pre-Conditions

Actor successfully logins to the system and chooses to view information about drivers

Post-Conditions

If the use case was successfully done, actor can get his/her result within 5 seconds. If not, system remains unchanged.

Extension points

None

1.4.12. Accept request

Brief Description

This use case describes how an actor can accept a request from a passenger

Flow of events

Basic flow

This use case starts when a passenger wants a trip and requests a driver, then that driver receives the request on the app.

1. The driver receives a notification about that passenger's request
2. The driver views information about the request (trip, passenger,...)
3. The driver accepts the request

Alternative flow

Cancel request

If the driver can accept the request from passenger because he/she can't give that passenger the ride which the passenger wants, he/she can just view the request and stop at step 2 of the *Basic flow.*

Special requirements

None

Pre-Conditions

Driver successfully logins to the system and there is a request from a passenger sent to that driver.

Post-Conditions

If the use case was successfully done (which means the driver accept that passenger's trip), a new trip will be saved to the database. If not, there is no new trip in the database.

Extension points

None

1.4.13. View history

Brief Description

This use case describes how an actor can see information of the trip he/she went in the past.

Flow of events

Basic flow

1. Actor chooses to view history
2. System displays a list of trips he/she went
3. Actor chooses the trip he/she wants to see information
4. System displays the information of that trip
5. Actor checks the information

Alternative flow

None

Special requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to view the information of a trip he/she went (view history).

Post-Conditions

If the use case was successfully done, actor can see the information of the trip he/she went. If not, the system displays an error message.

Extension points

None

1.4.14. Scheduler

Brief Description

This use case describes how a passenger can schedule a trip

Flow of events

Basic flow

1. Actor chooses a pick-up location
2. Actor chooses to schedule a trip
3. Actor chooses the date, type of car,... for the trip
4. Actor chooses the payment method
5. System displays the scheduled trip's information
5. Actor verifys the information

Alternative flow

No payment method

If actor didn't give information about the payment method or give an invalid payment method, system will display a popup to inform actor. Actor can choose the payment method again or quit the SCHEDULER module.

Change information

If actor wants to change the information of the trip he/she has just scheduled, or cancel that trip, he/she can choose "Change" to change the information or delete the trip.

Special requirements

GPS on.

Pre-Conditions

Actor successfully logins to the system and chooses to schedule a trip.

Post-Conditions

If the use case was successfully done, actor has a scheduled trip in the database. If not, no trip was scheduled.

Extension points

None

1.4.15. Promotions

Brief Description

This use case describes how an actor can get extra benefit from the system.

Flow of events

Basic flow

1. Actor chooses to get/view promotions
2. Actor enters his/her gift code
3. System informs actor about the promotions which he/she has just received

Alternative flow

Invalid or used gift code

If actor enters an invalid or a gift code used by another user, system displays a message to inform actor. Actor can start again from the beginning of *Basic flow* or cancel the use case.

Special requirements

None

Pre-Conditions

Actor successfully logins to the system.

Post-Conditions

If the use case was successfully done, actor will be informed by the system about the extra benefit he/she will received. If not, the system display an error message and actor will received no extra benefit from the system.

Extension points

None

1.4.16. View traffic map

Brief description

This use case describes how an actor can see the traffic map while travelling

Flow of events

Basic flow

1. Actor chooses to view traffic map
2. System displays the traffic map
3. Actor views traffic map

Alternative flow

No GPS

If GPS was turned off while actor uses this function then the system will display a message. Actor can turn on GPS and start again from the beginning of *Basic flow* or cancel the use case.

Special requirements

None

Pre-Conditions

GPS on and actor are travelling.

Post-Conditions

If the use case was successfully done, system will display a map and actor can see where he/she is while travelling. If not, system will display a message.

Extension points

None

1.4.17. App setting

Brief description

This use case describes how an actor can change the setting of the application.

Flow of events

Basic flow

1. Actor chooses "App setting"
2. System displays list of setting options that actor can change
3. Actor makes his/her change(s)
4. Actor confirms after his/her change(s)
5. System saves change(s) and changes to what actor wants
6. Actor checks the app after change(s)

Alternative flow

None

Special requirements

None

Pre-Conditions

Actor successfully logins to the system

Post-Conditions

If the use case was successfully done, application will change to what actor wants. If not, application remains unchanged.

Extension points

None

1.4.18. Request a trip

Brief description

This use case describes how an actor can request a trip

Flow of events

Basic flow

1. Actor chooses to request a trip

2. System will ask actor to choose car type, pick-up location, destination (The detail of these steps will be described in other use cases)

3. After the actor has finished providing information about the trip, he/she will confirm the information and the requesting steps are done.

4. System will save the information about actor's trip to the database.

Alternative flow

Invalid information

If actor enters an invalid information (such as pick-up location, destination,...), system will display a message to inform actor. He/she can check the wrong information, correct it and continue with the requesting steps or choose to cancel the use case.

Special requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to request a trip

Post-Conditions

If the use case was successfully done, system will have a new trip in the database. If not, there is no new trip in the database.

Extension points

None

1.4.19. View rating

Brief description

This use case describes how an actor can see his/her rating which was rate by other people.

Flow of events

Basic flow

1. Actor chooses to view his/her rating
2. System displays actor's rating
3. Actor views the rating

Alternative flow

None

Special requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to view his/her rating

Post-Conditions

If the use case was successfully done, system will display information about actor's rating. If not, system will remain unchanged.

Extension points

None

1.4.20. Rating

Brief description

This use case describes how an actor can rate other people (such as a passenger rates a driver)

Flow of events

Basic flow

1. Actor chooses “Rating” to rate another person

2. System displays a “Rating” form

3. Actor chooses the person who he/she wants to rate

4. System displays a “Rating” form

5. Actor provides information which was stated in the form (how many stars he/she wants to rate that person, his/her opinions about his/her choice,...)

6. Actor verifies the information and confirms with the system

7. System saves the information provided by actor and the person who was rated by actor can see his/her rating in the use case “View rating”

Alternative flow

None

Special requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to rate a person

Post-Conditions

If the use case was successfully done, system will save the information provided by actor and the person who was rated can see it in the use case “View rating”. If not, system will have no new information about rating of that person.

Extension points

None

1.4.21. Contact

Brief description

This use case describes how an actor can contact another person (like a passenger can contact with a driver when he/she requests a trip)

Flow of events

Basic flow

1. Actor chooses contacts with another person

2. System sets up a connection between actor and the person whom that actor wants to contact with.

3. Actor discusses with that person through that connection

Alternative flow

None

Special requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to contact another person.

Post-Conditions

If the use case was successfully done, a connection will be set up between actor and the person whom he/she want to contact with. If not, no connection will be set up.

Extension points

None

1.4.22. Manage driver

Brief Description

This use case describes how system manager can manage drivers' information and check their locations on the map

Flow of events

Basic flow

* Manage drivers' information:

1. Actor chooses to manage information about a driver (view, search, change, delete,...)

2. System displays a list of drivers and actor can search for the driver whose information he/she wants to manage

3. Actor chooses the driver whose information he/she wants to do the management steps (change, delete,...)

4. System displays information about that driver and actor can chooses what to do with that information. According to actor's choice, there will be one of these sub-flows:

- Change information:

+ System displays information of that driver in a interface that he/she can change that driver's information

+ Actor chooses which information he/she wants to change and changes it to what he/she wants

+ Actor confirms the change(s) with the system

+ System displays new information about that driver

- Delete driver's account:

+ System displays a message box to ask for actor's confirmation about his/her choice

+ Actor confirms with the system

+ System deletes that driver's information from the database

* Manage drivers' location:

1. System displays a list of drivers who successfully logined to the system and their status are "Online".

2. System displays a "Search" interface

3.Actor chooses criterias to search for the rights drivers

4. Actor chooses the driver who he/she wants to check the location

5. System displays information about that driver's location

Alternative flow

Invalid driver

In drivers' information management steps, if system can't find the driver who actor wants, system will display a message and actor can check again or cancel the use case.

No online driver

In drivers' location management steps, if there is no online driver, system will display a message to inform system manager.

Special requirements

1. In drivers' locations management steps, drivers have to turn on GPS in their devices, successfully logins to the system and their status are "Online".

2. In drivers' information management steps, there are some fields that system manager can't change such as driver's names, age, sex, phone numbers,...

Pre-Conditions

Actor (System manager) successfully logins to the system and chooses to manage drivers

Post-Conditions

If the use case was successfully done, actor can manage drivers' information or locations. If not, information about drivers remains unchanged in the database (drivers' information management steps); actor can't manage drivers' locations (drivers' locations management steps)

Extension points

None

1.4.23. Manage passenger

Brief Description

This use case describes how an actor (system manager) can manage information about passengers

Flow of events

Basic flow

1. Actor chooses to manage passengers' information (change, delete,...)

2. System displays a "Search" interface

3. Actor searchs for the passenger he/she wants to do information management steps with

4. According to actor's choice, there will be one of these sub-slows:

- Change information:

+ System displays information of that passenger in a interface that he/she can change that passenger's information

+ Actor chooses which information he/she wants to change and changes it to what he/she wants

+ Actor confirms the change(s) with the system

+ System displays new information about that passenger

- Delete passenger's account:

+ System displays a message box to ask for actor's confirmation about his/her choice

+ Actor confirms with the system

+ System deletes that passenger's information from the database

Alternative flow

Invalid passenger

If system can't find the passenger who actor wants, system will display a message and actor can check again or cancel the use case.

Special requirements

There are some fields that system manager can't change such as passenger's names, age, sex, phone numbers,...

Pre-Conditions

Actor (System manager) successfully logins to the system and chooses to manage passengers

Post-Conditions

If the use case was successfully done, actor can manage passengers' information. If not, information about passengers remains unchanged in the database

Extension points

None

1.4.24. Manage trip

Brief Description

This use case describes how an actor (system manager) can manage trips

Flow of events

Basic flow

1. System displays a list of trips made
2. Actor searches for the trip he/she wants
3. System displays available trips with information entered by actor
4. Actor chooses a specific trip
5. System displays detail about that trip

Alternative flow

Invalid trip

If system can't find the trip with information provided by actor, system will inform actor and actor can choose to do it again or cancel the use case.

Special requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to manage trips

Post-Conditions

None

Extension points

None

1.4.25. Select car

Brief Description

This use case describes how an actor (passenger) can select car for her trip

Flow of events

Basic flow

1. Actor chooses to select car for her trip

2. System displays a list of car

3. Actor chooses a car from the list or he/she can suggest another type of car for his/her trip if actor's car is not in the list

4. System saves actor's choice

Alternative flow

None

Special requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to select car after requesting a trip.

Post-Conditions

If the use case was successfully done, system will save actor's choice to the database, it'll also create a bill for the trip. If not, there is no new information about actor's car choice in the database.

Extension points

None

1.4.26. Share info

Brief Description

This use case describes how an actor (passenger) can share his/her trip information to his/her friends or other people.

Flow of events

Basic flow

1. Actor chooses to share information about his/her trip
2. System displays information about actor's trip and ways to share trip information
3. Actor checks the information and chooses a way to share information
4. Actor confirms with the system

Alternative flow

None

Special requirements

None

Pre-Conditions

Actor successfully logins to the system and chooses to share information about his/her trip

Post-Conditions

If the use case was successfully done, actor successfully shares trip information to his/her friends and other people he/she knows. If not, actor can't share information about his/her trip.

Extension points

None

2. Uber Analysis

2.1. Architectural analysis

2.1.1. Key abstractions

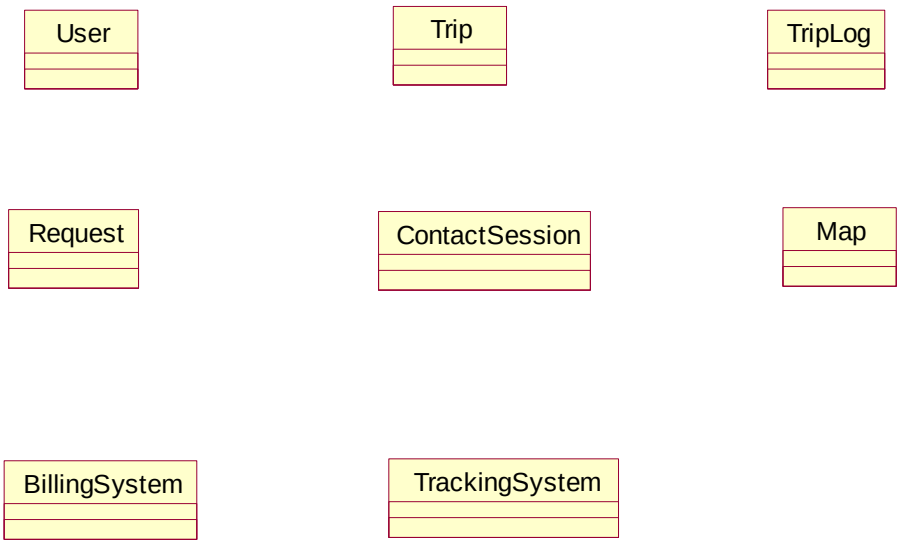


Figure 2.1.1 Architectural analysis

2.1.1.1. Key abstractions definitions

User : The customers of system. It includes driver and passenger

Analysis mechanism: Persistency, Security

Trip: The movement of driver and passenger. It is created after passenger send a request to system

Analysis mechanism: Persistency, Security

TripLog: The detail information of trip written automatically by system

Analysis mechanism: Persistency, Security

Request: Passenger's request to system to create trip. It include time and location, choosen car type

Analysis mechanism: Information exchange, Process control and synchronization

Contact session: Communication between driver and passenger

Analysis mechanism: Message rouing, Distribution, Process control and synchronization

Map: A map for detect location of driver and passenger and describe trip road

Analysis mechanism: Persistency

BillingSystem: A component to estimate trip's fare for passenger

Analysis mechanism: Persistency, Information exchange

TrackingSystem: A component to detect driver location to send passenger's request to them

Analysis mechanism: Persistency, Information exchange

2.1.2. Upper-Level Components and their dependencies

2.1.2.1. Component definitions

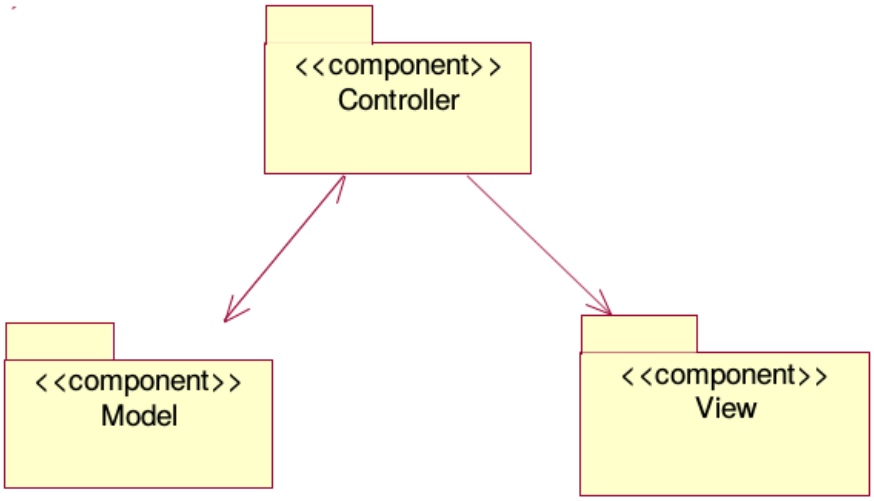


Figure 0: The upper-level layer architecture

2.2. Use Case Analysis

Controller: represents the classes connecting the model and the view, and is used to communicate between classes in the model and view

Model: represents the underlying, logical structure of data in a software application and the high-level class associated with it. This object model does not contain any information about the user interface

View: a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, and so forth)

2.2.1. Use Case realization interaction diagrams

2.2.1.1. Login

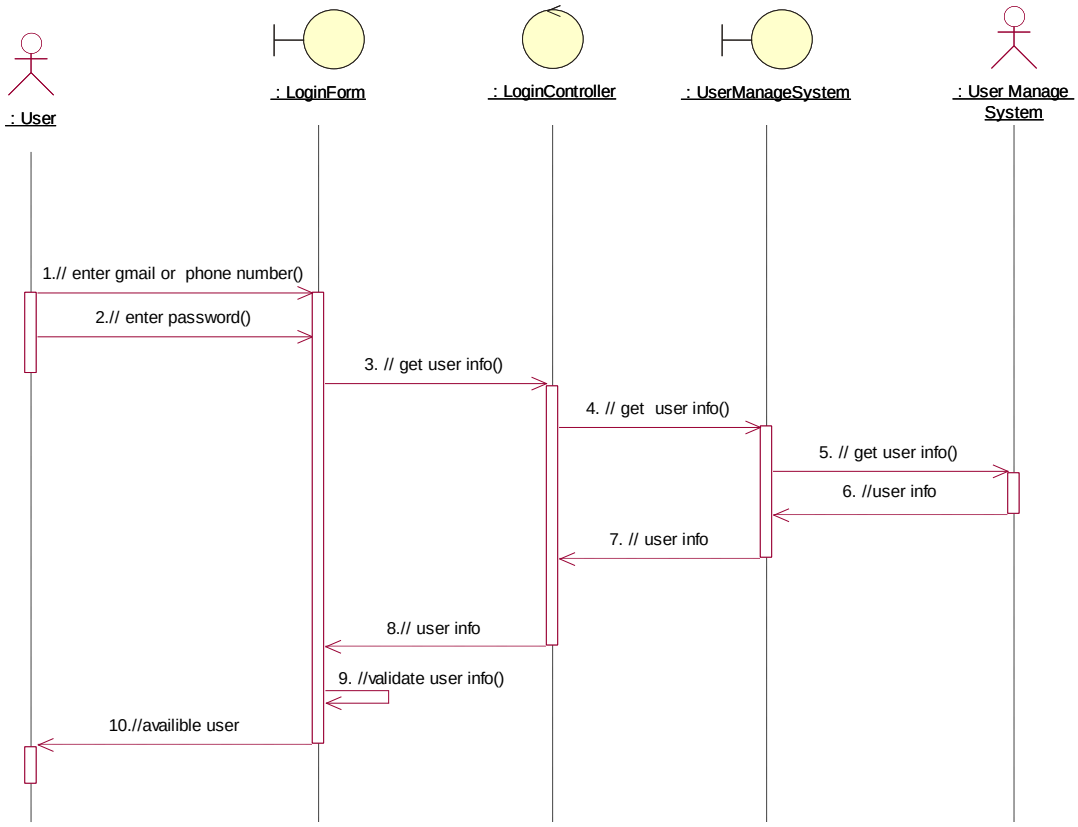


Figure 1: Login Flow

2.2.1.2. Change profile

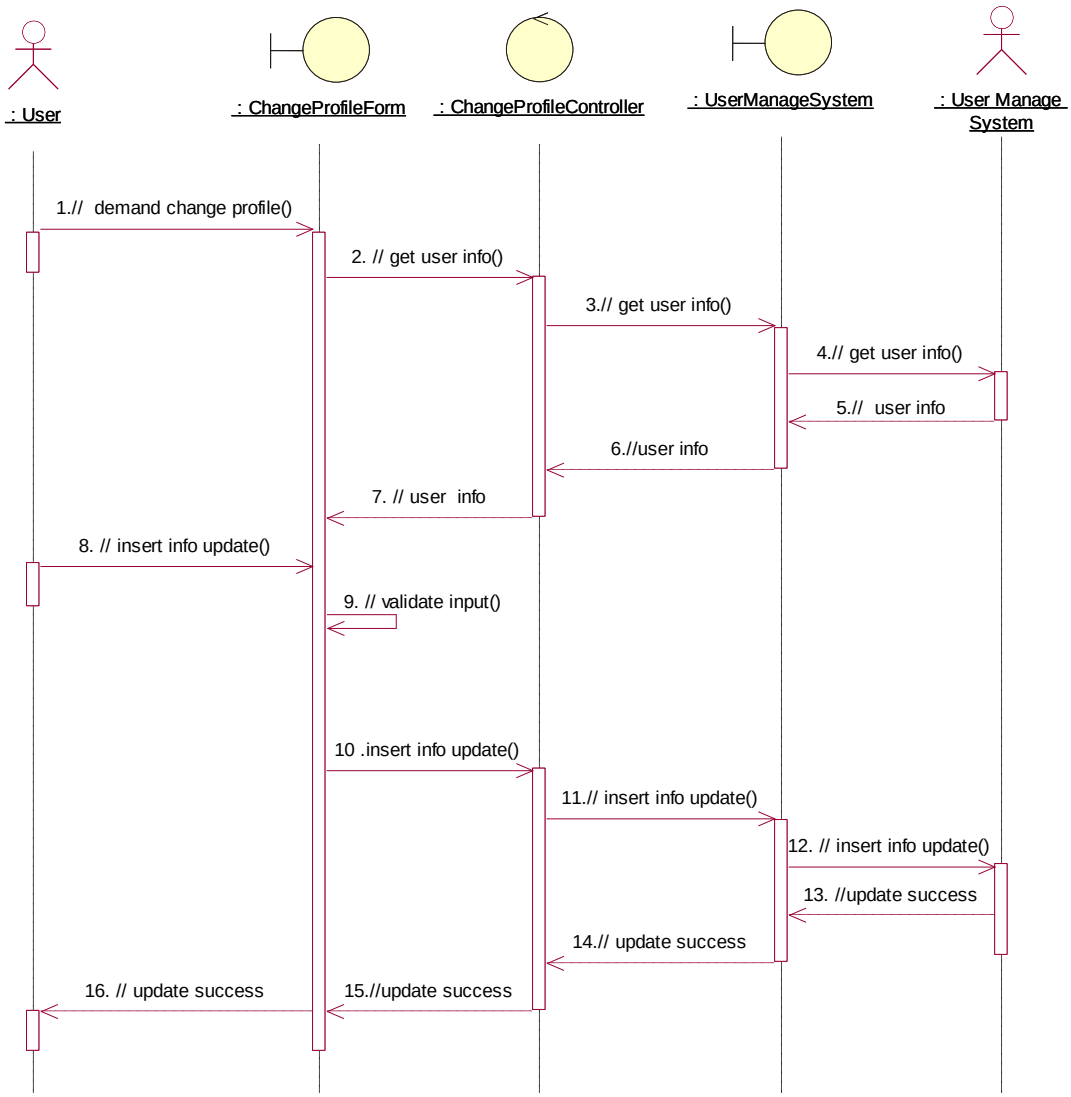


Figure 2: Changeprofile flow

2.2.1.3. View Info Passenger

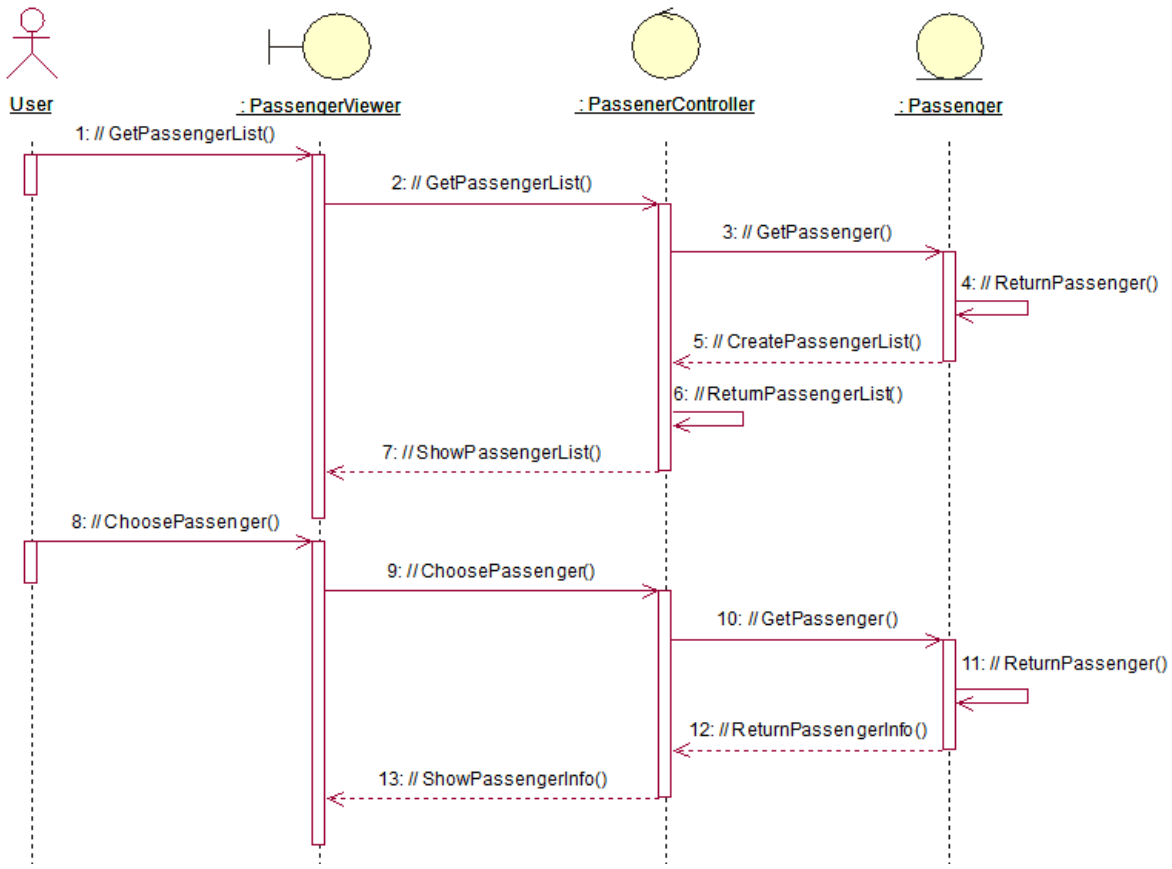


Figure 3 View Information Passenger Flow

2.2.1.4. View Info Trip

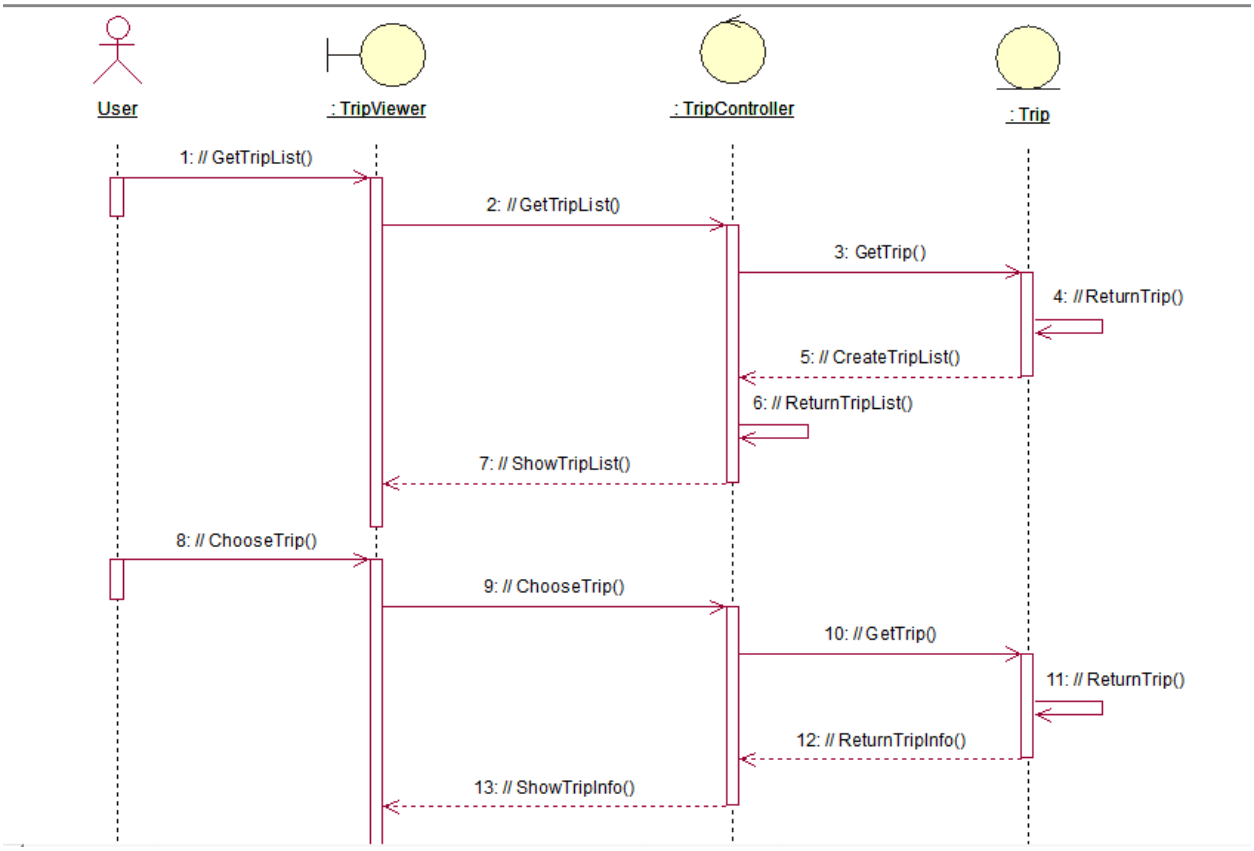


Figure 4: View Information Trip

2.2.1.5. View Notification

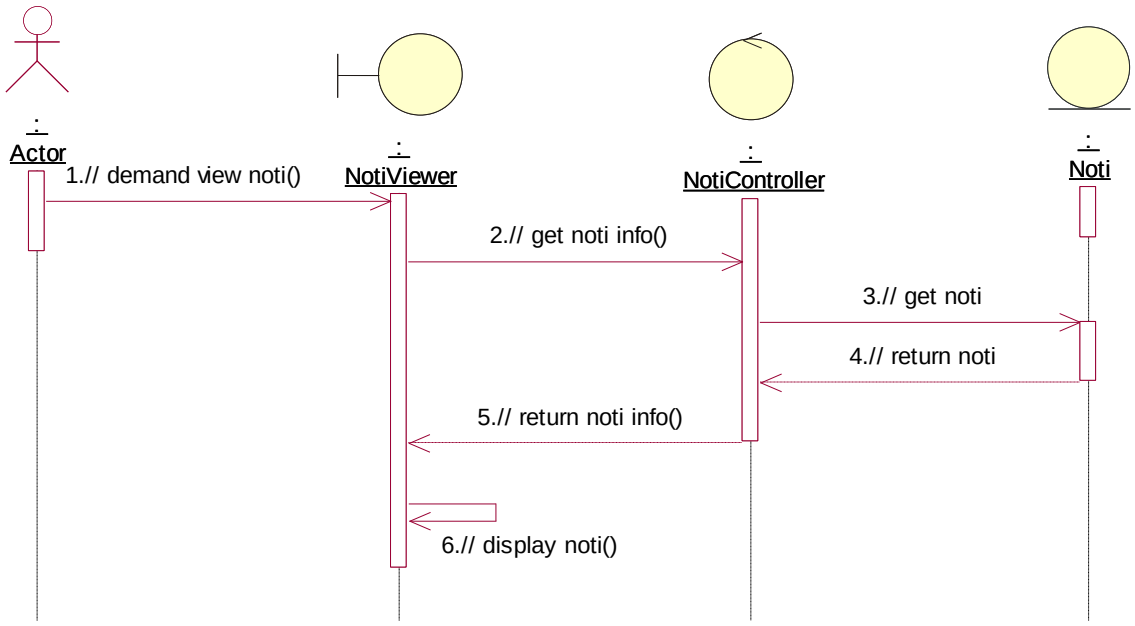


Figure 5: View Notification Flow

2.2.1.6. Enter Pickup Location

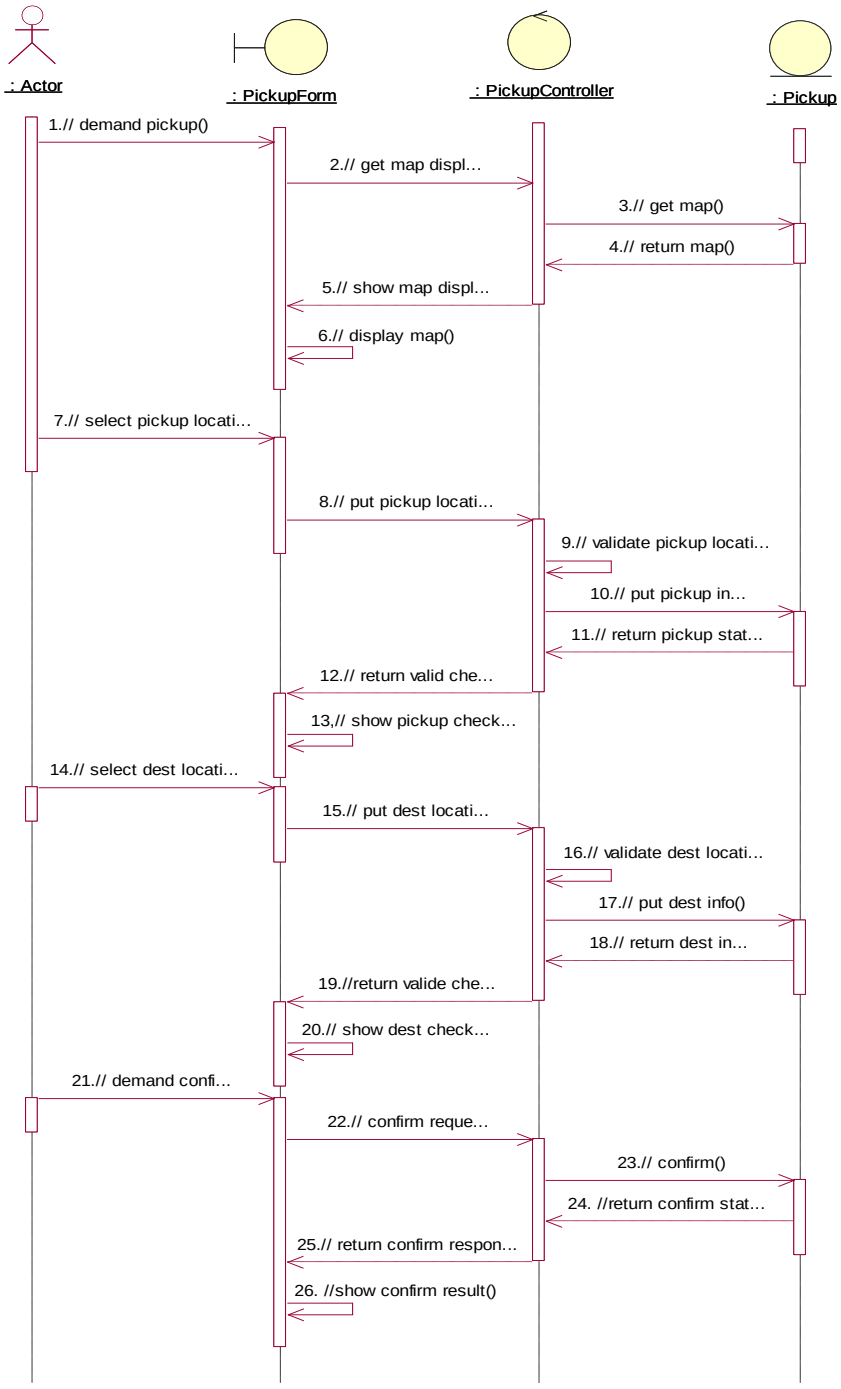


Figure 6: Enter Picup Location Flow

2.2.1.7. Billing

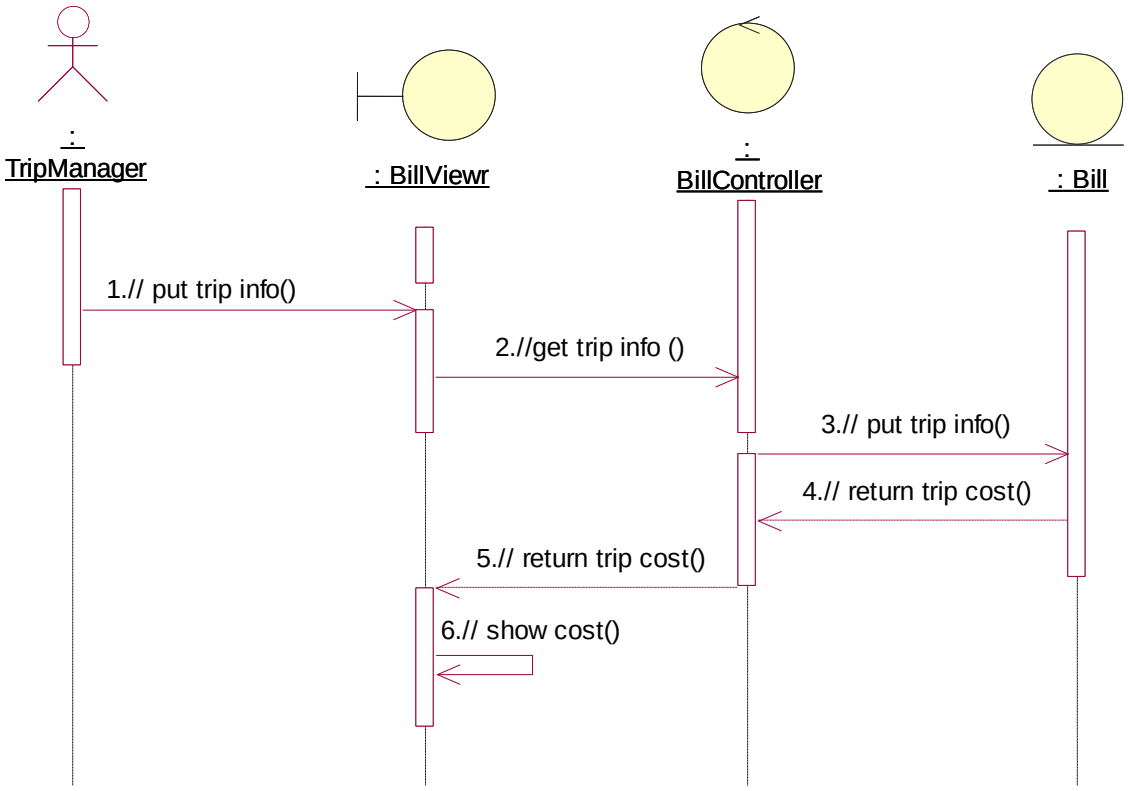


Figure 7: Billing Flow

2.2.1.8. View Available Passenger

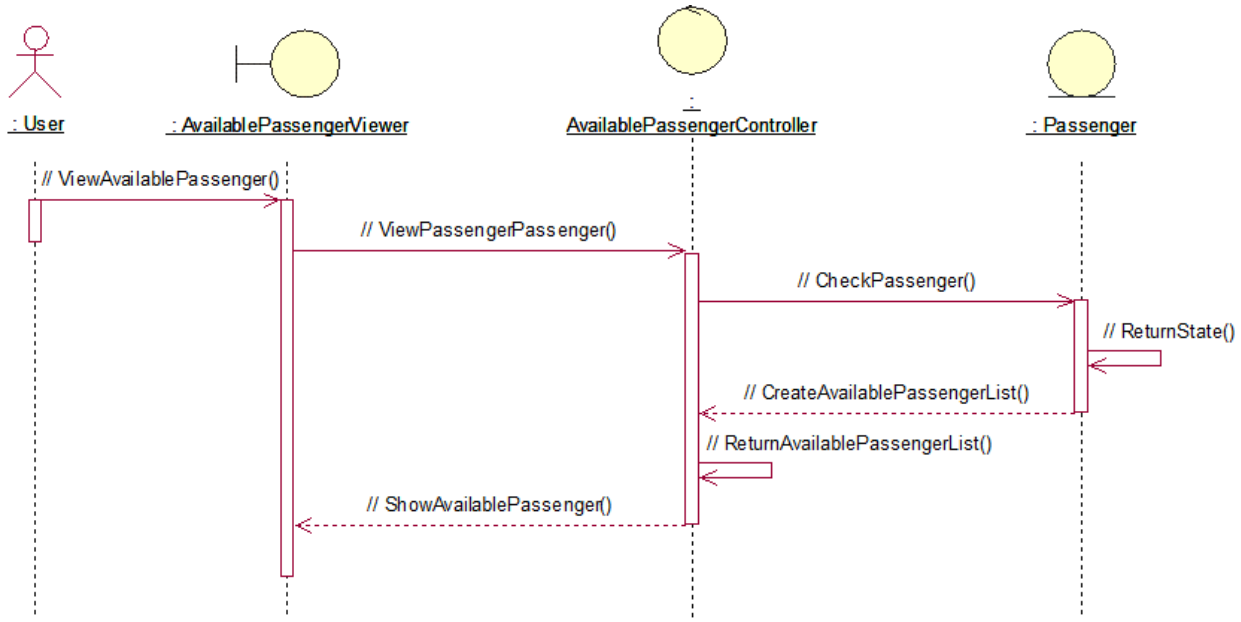


Figure 0.7: View Available Passenger Flow

2.2.1.9. Register

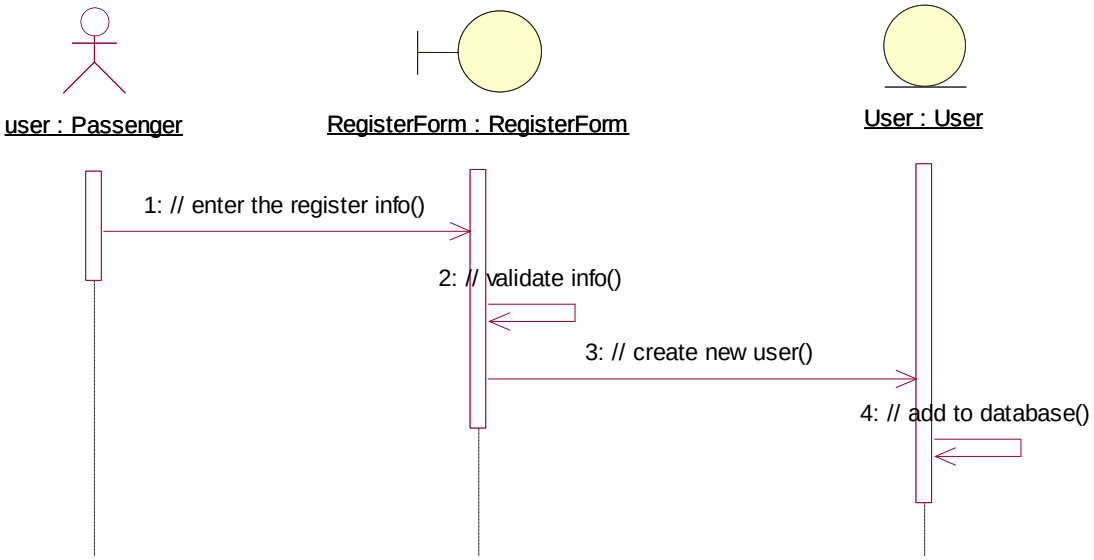


Figure 0.8: Register Flow

2.2.1.10. Get statistic data

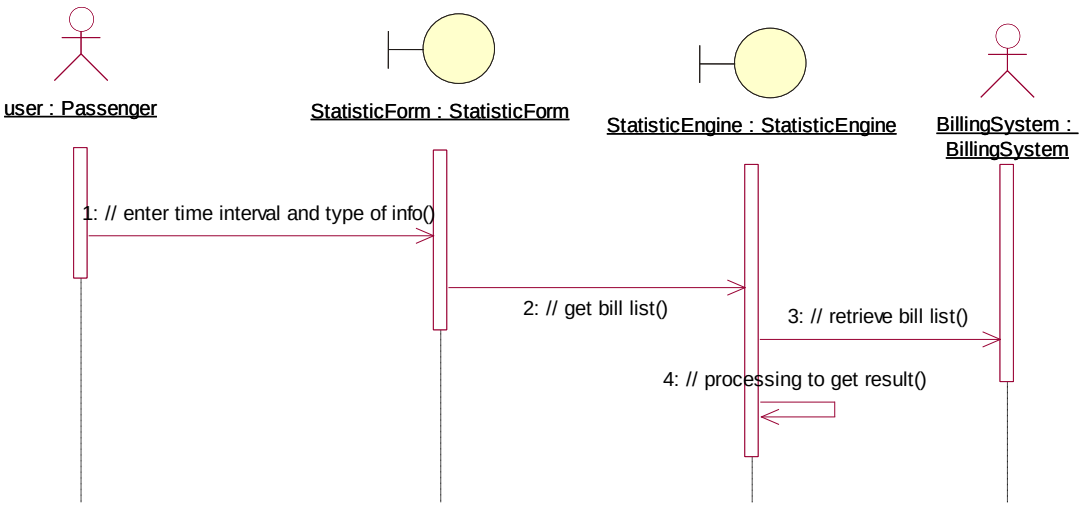


Figure 0.9: Get Statistic Data Flow

2.2.1.11. View Info Driver

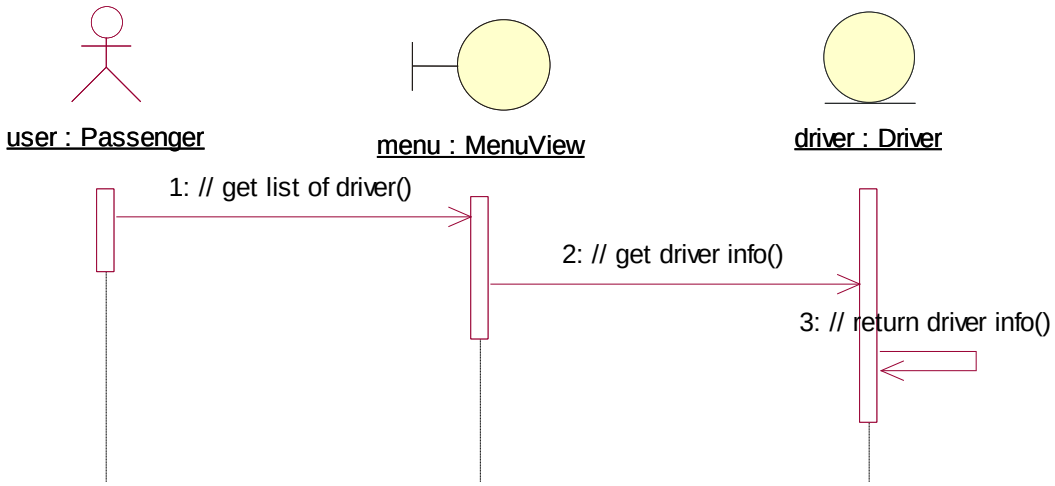


Figure 0.10: View Information Driver Flow

2.2.1.12. Accept request

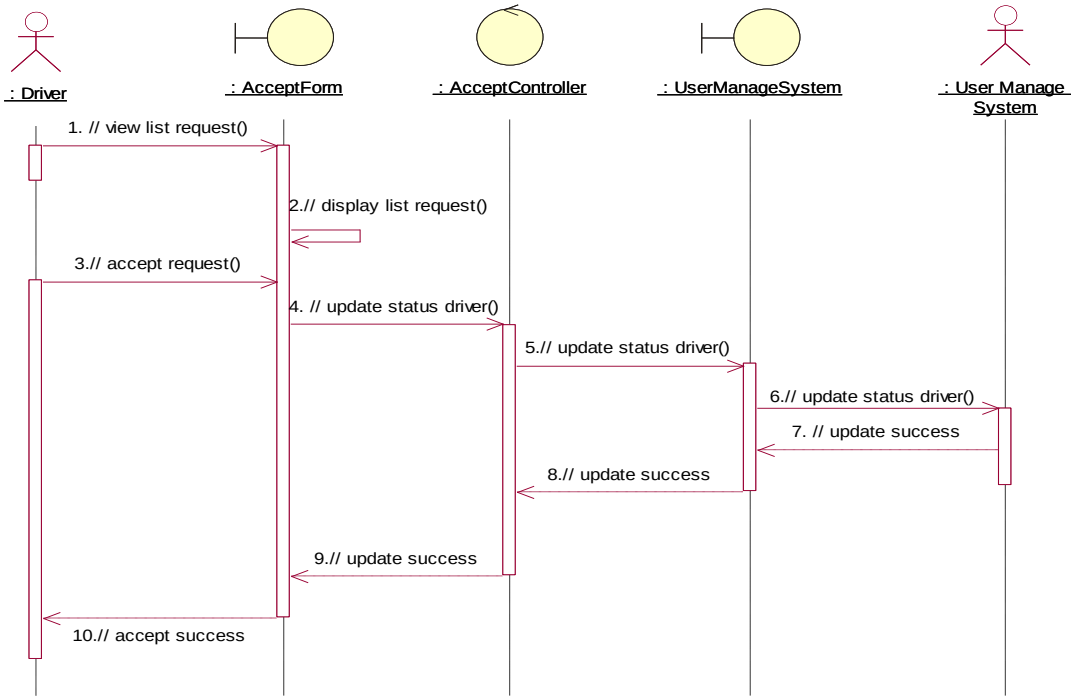


Figure 0.11: Accept Request Flow

2.2.1.13. View history

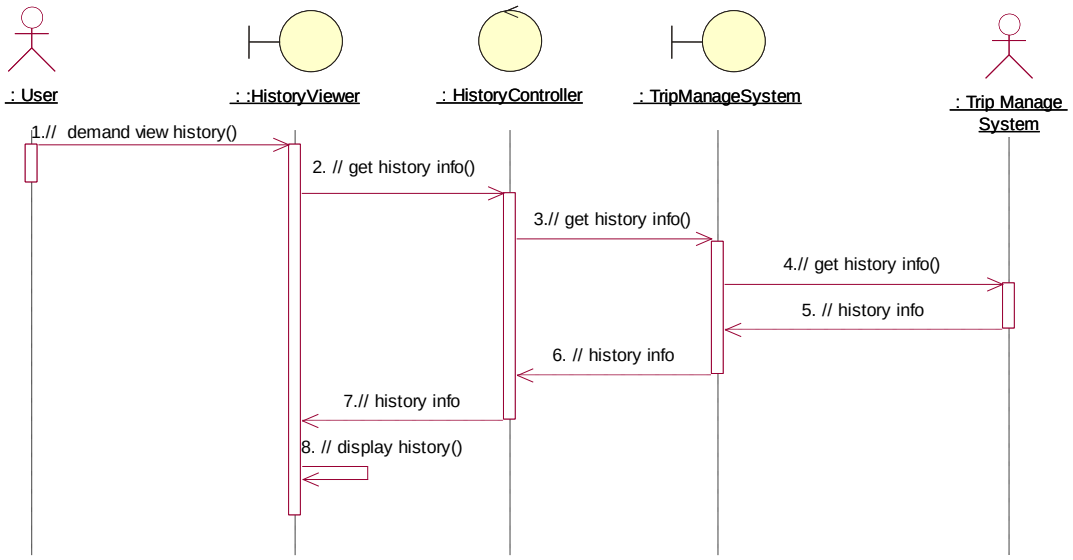


Figure 0.12: View History Flow

2.2.1.14. Scheduler

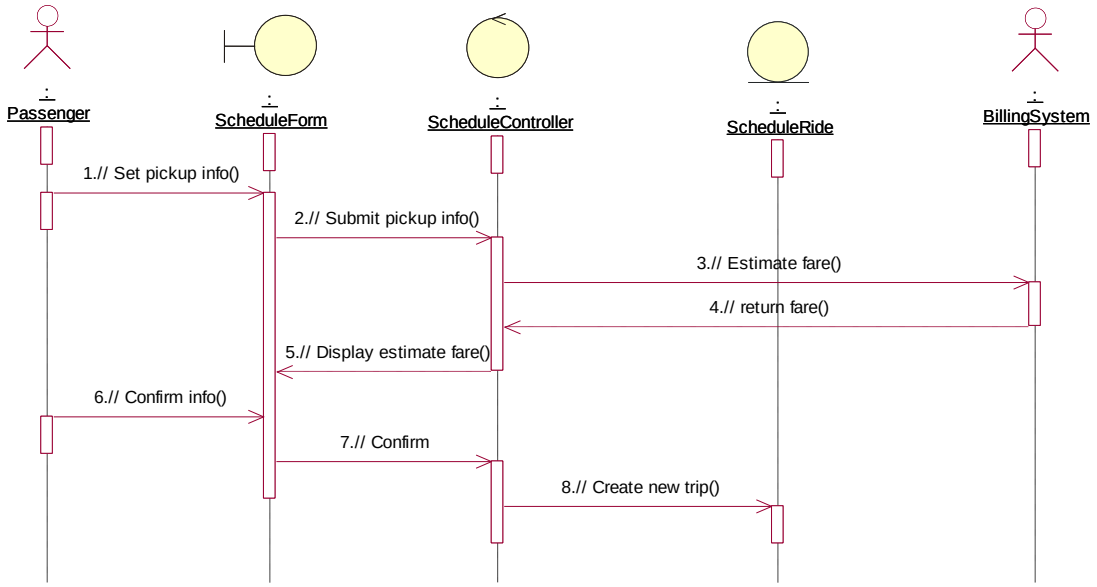


Figure 0.13 Scheduler Flow

2.2.1.15. Promotions

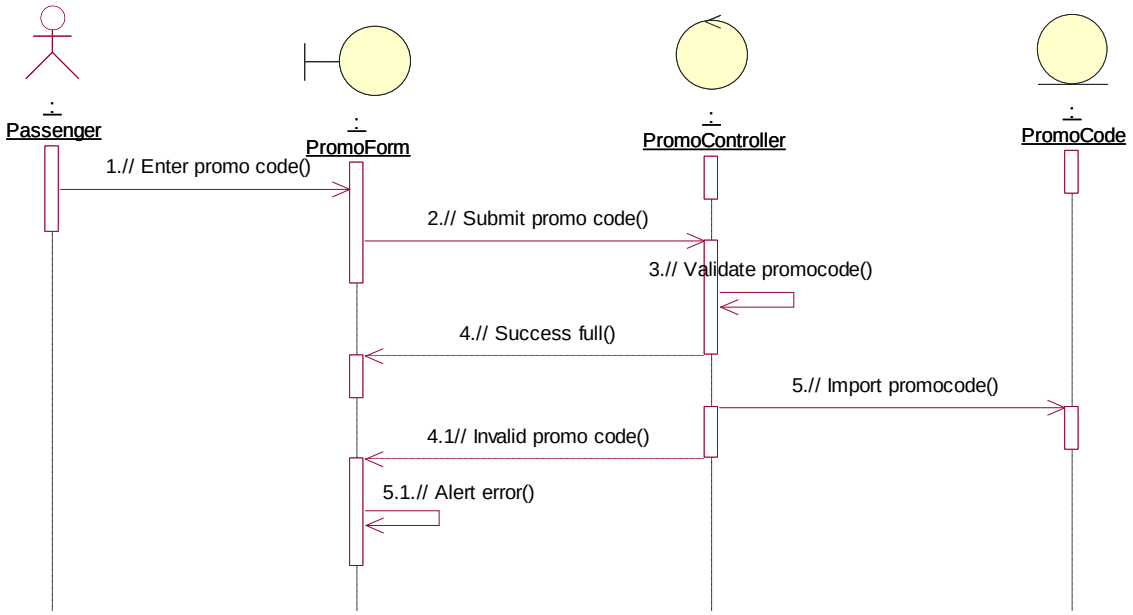


Figure 0.14 Promotions Flow

2.2.1.16. View traffic map

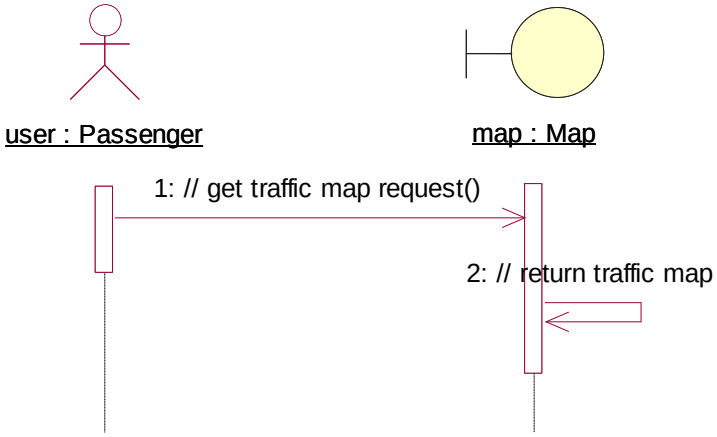


Figure 0.15 View Traffic Map Flow

2.2.1.17. App setting

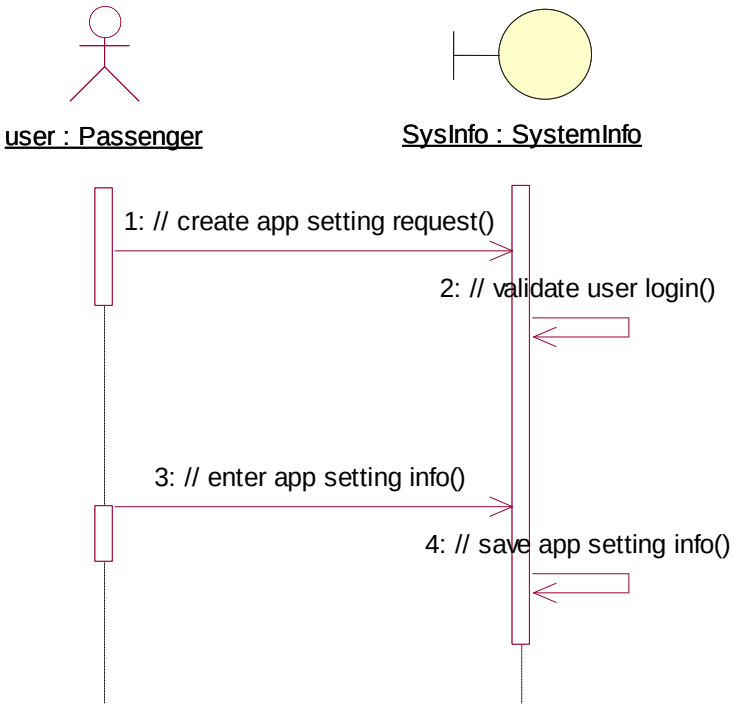


Figure 0.16 App Setting Flow

2.2.1.18. View rating

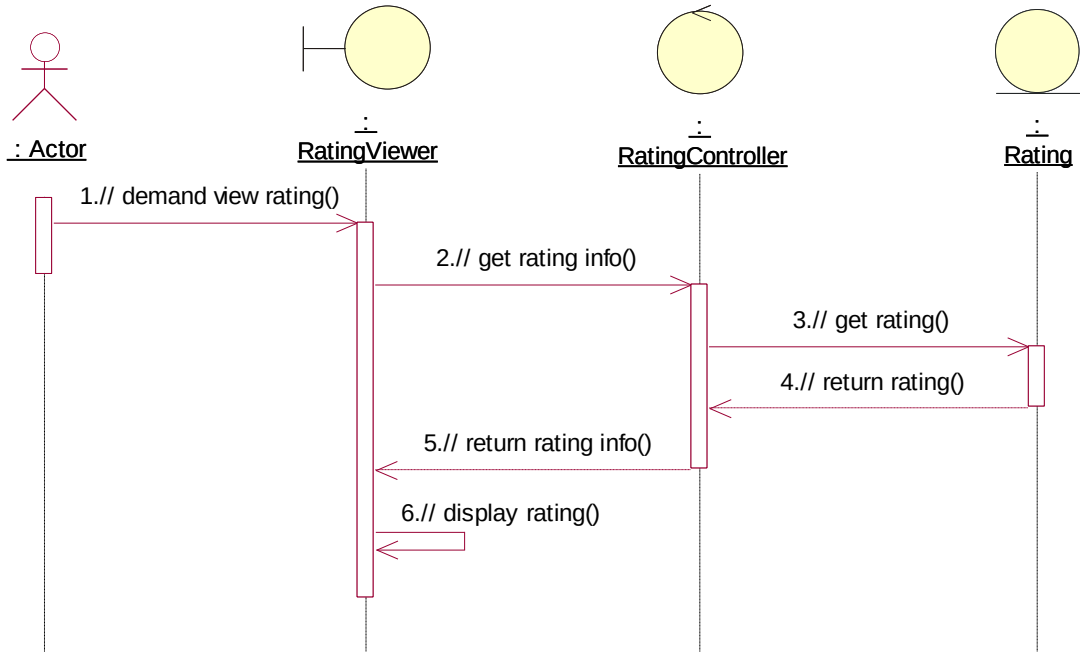


Figure 0.17 View Rating Flow

2.2.1.19. Rating

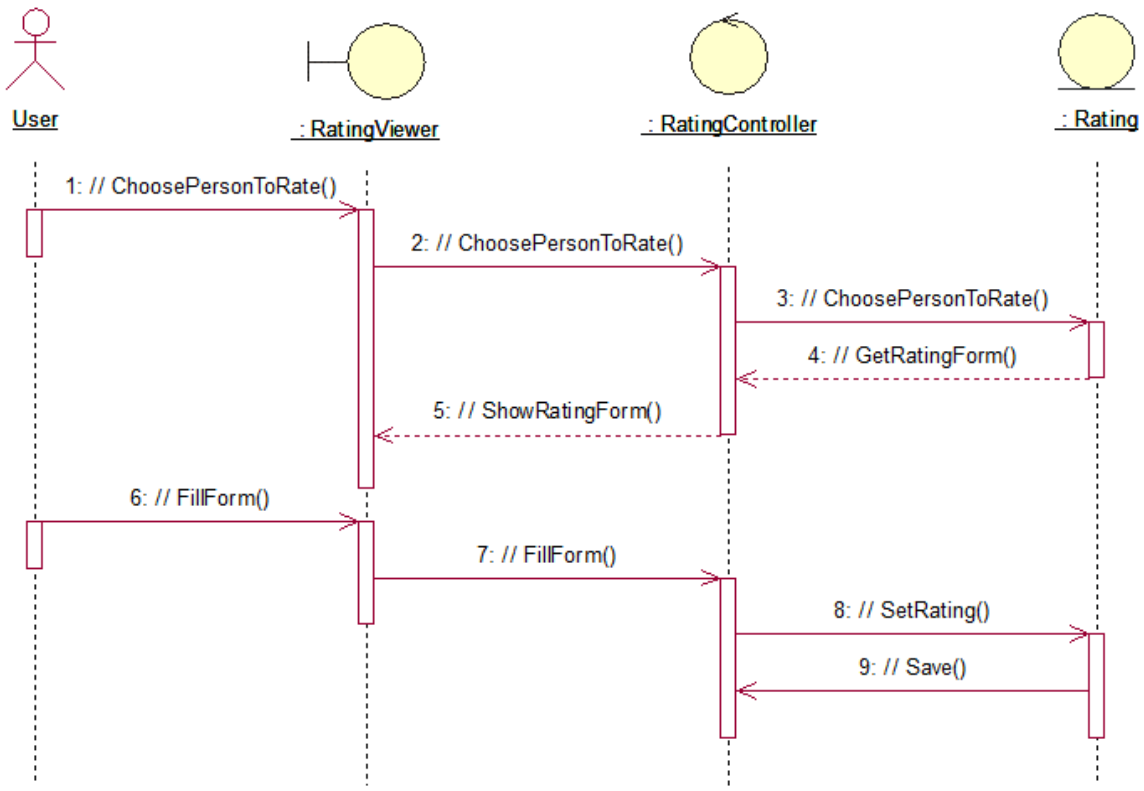


Figure 0.18 Rating Flow

2.2.1.20. Contact

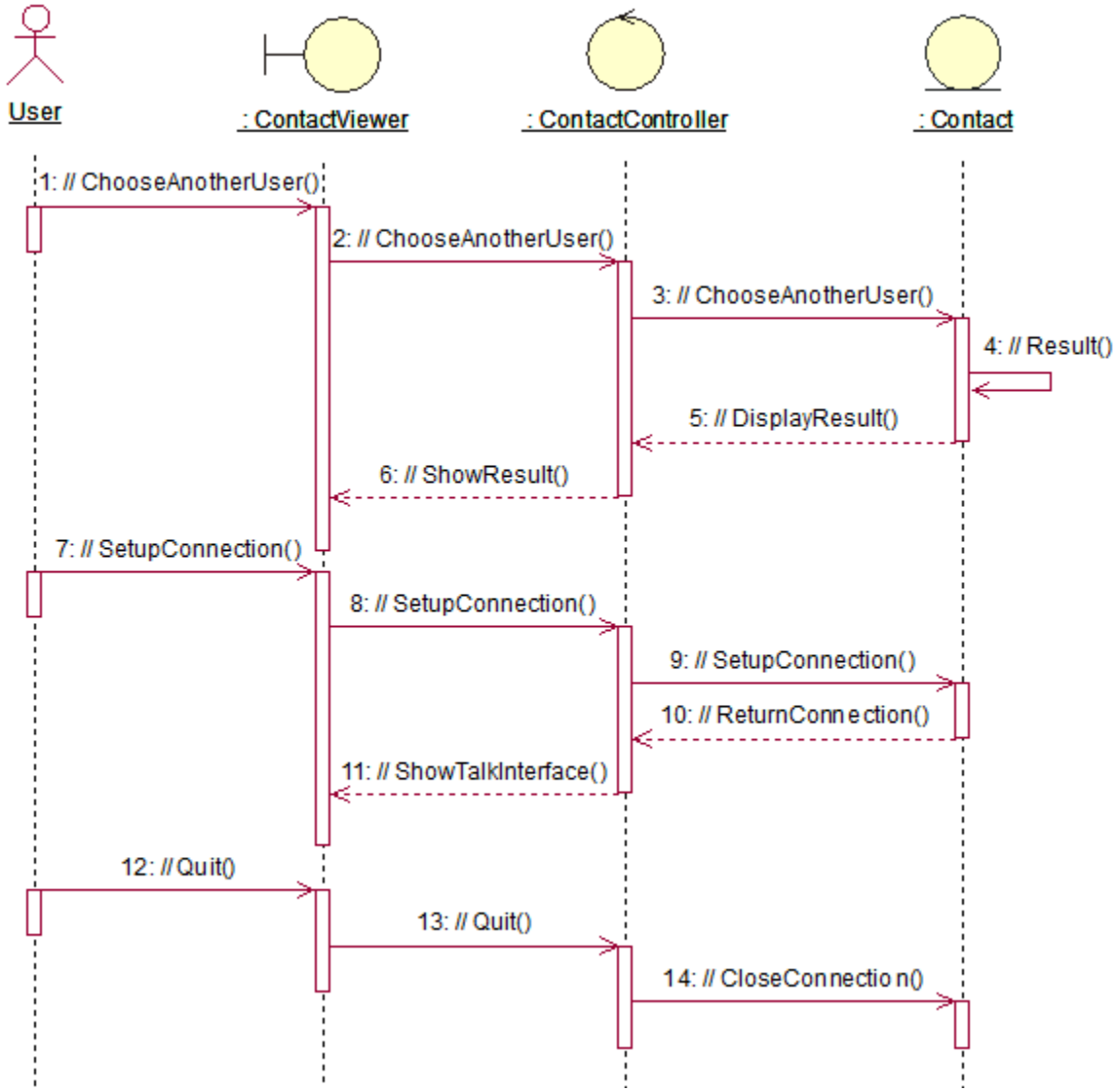


Figure 0.19 Contact Flow

2.2.1.21. Manage driver location

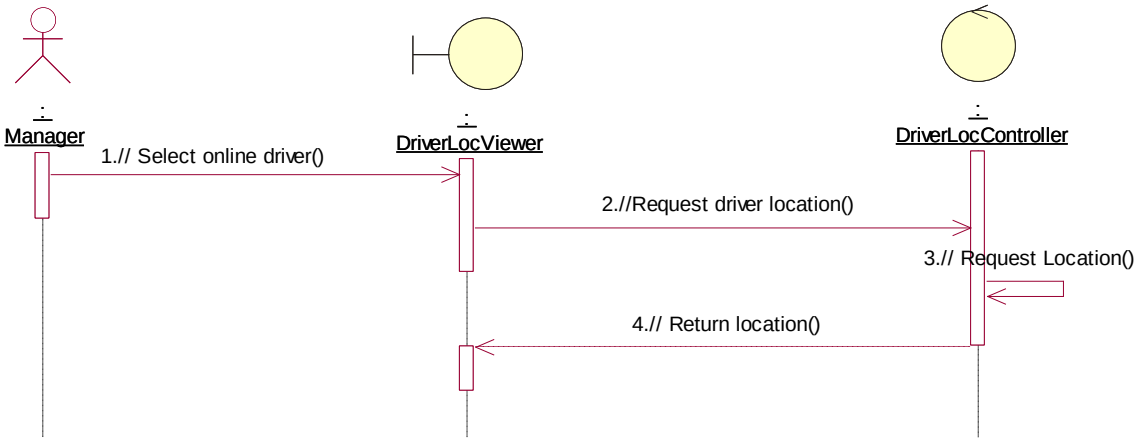


Figure 0.20 Manage Driver Location Flow

2.2.1.22. Manage user information

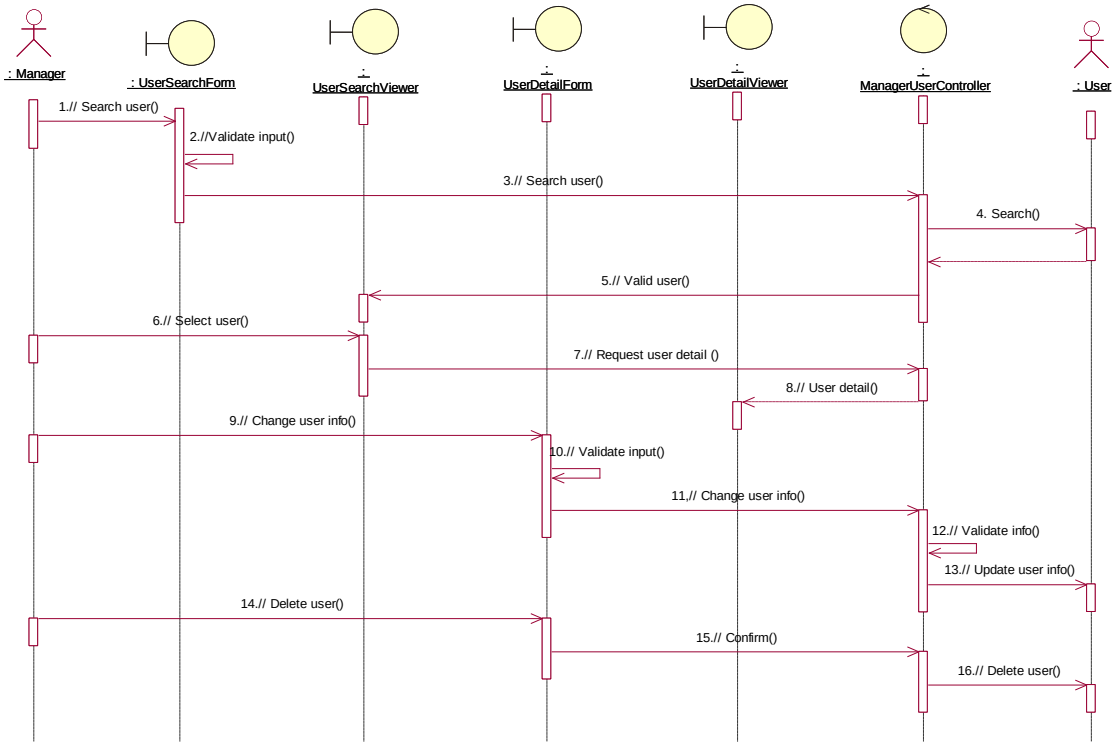


Figure 0.21 Manage User Information Flow

2.2.1.23. Manage trip

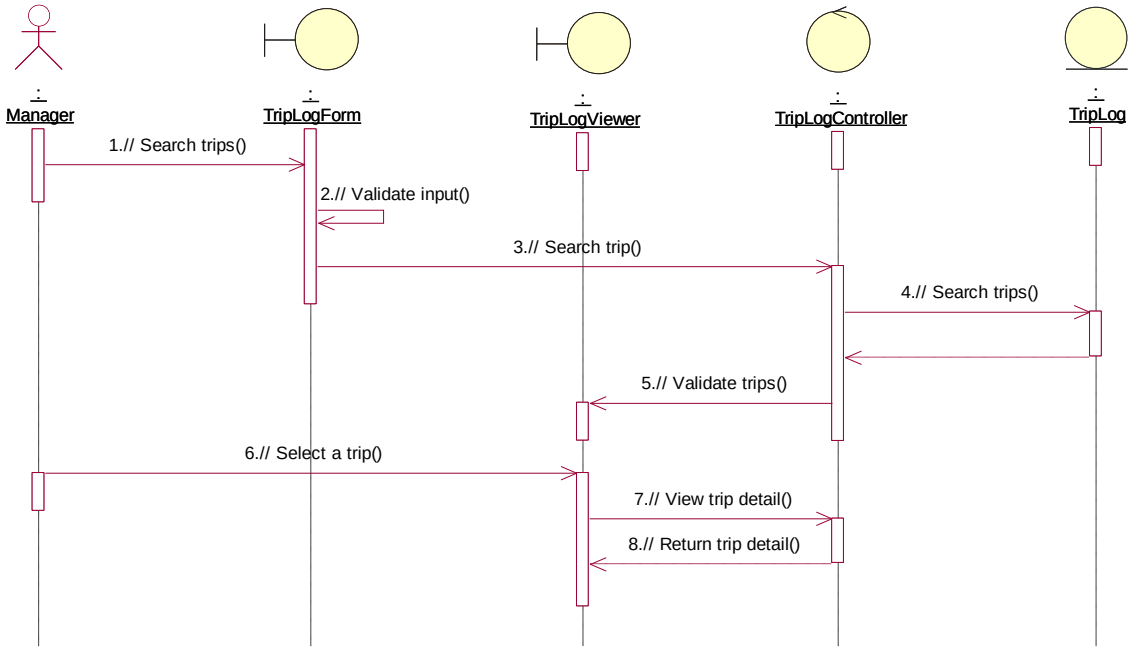


Figure 0.22 Manage Trip Flow

2.2.1.24. Select car

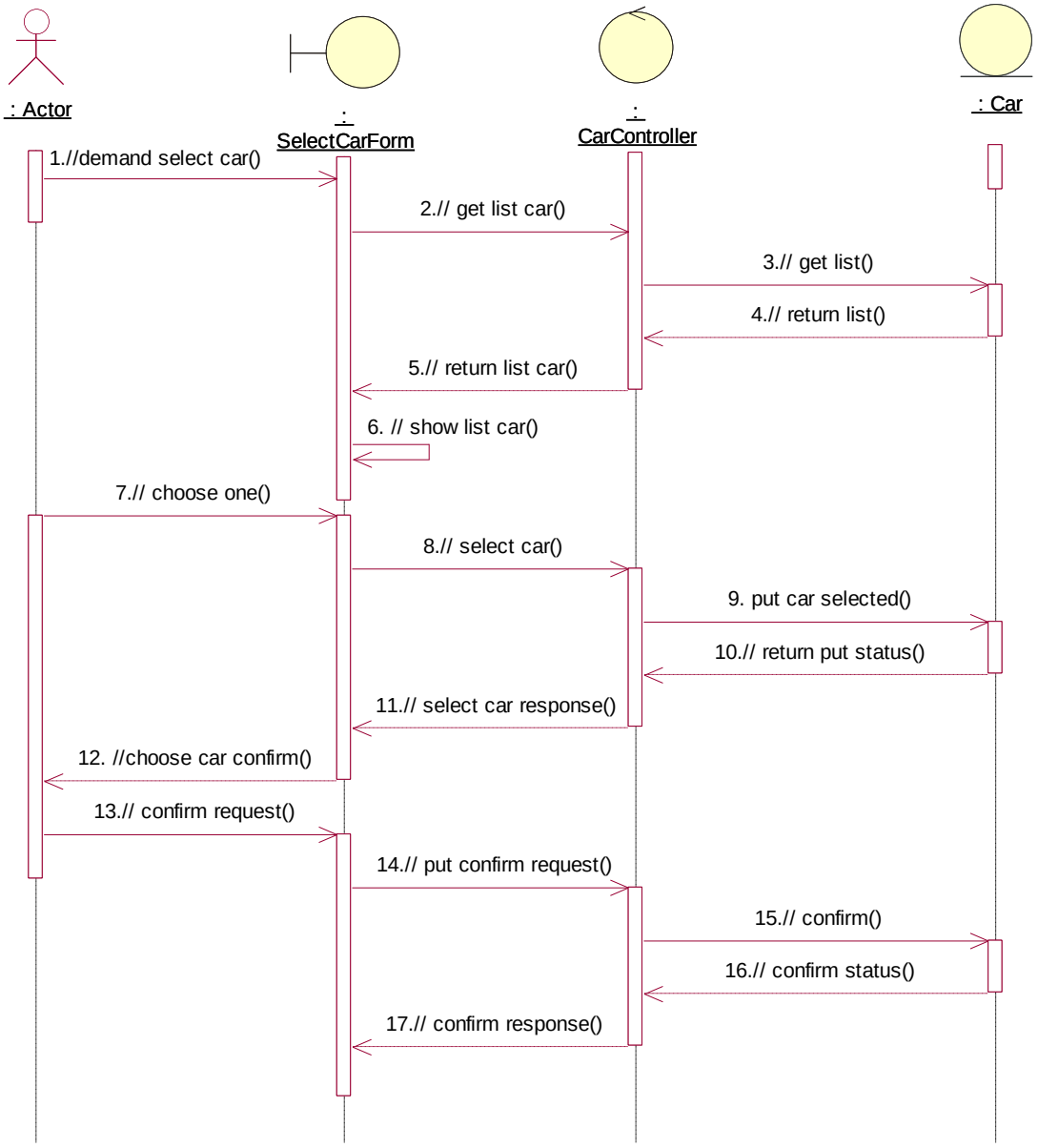


Figure 0.23 Select Car Flow

2.2.1.25. Share info

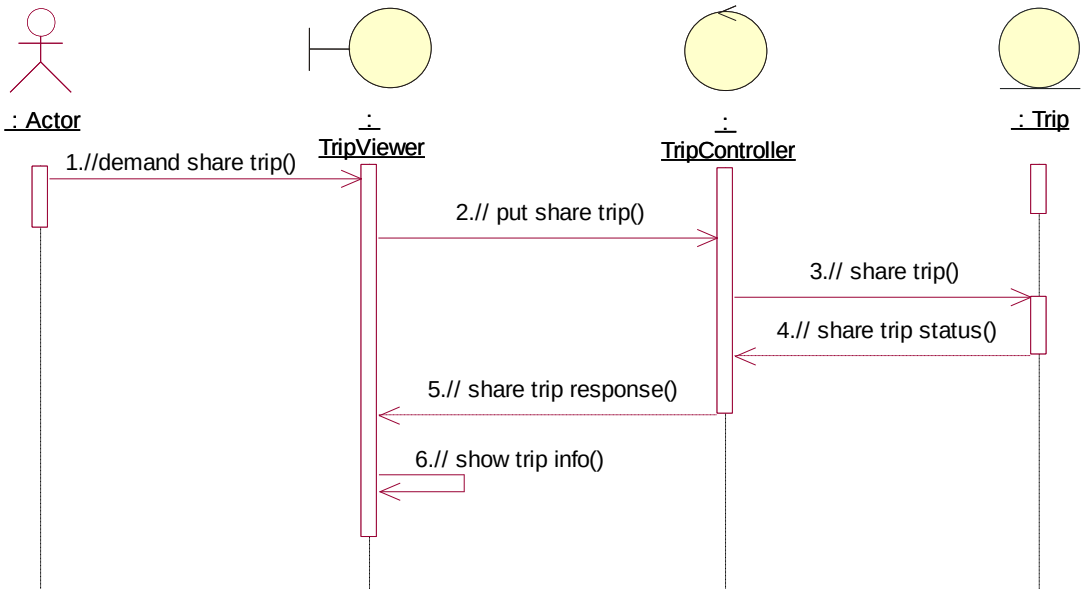


Figure 0.24 Share Info Flow

2.2.2. Use case Realization view of Participating Class (VOPCs)

2.2.2.1. Login

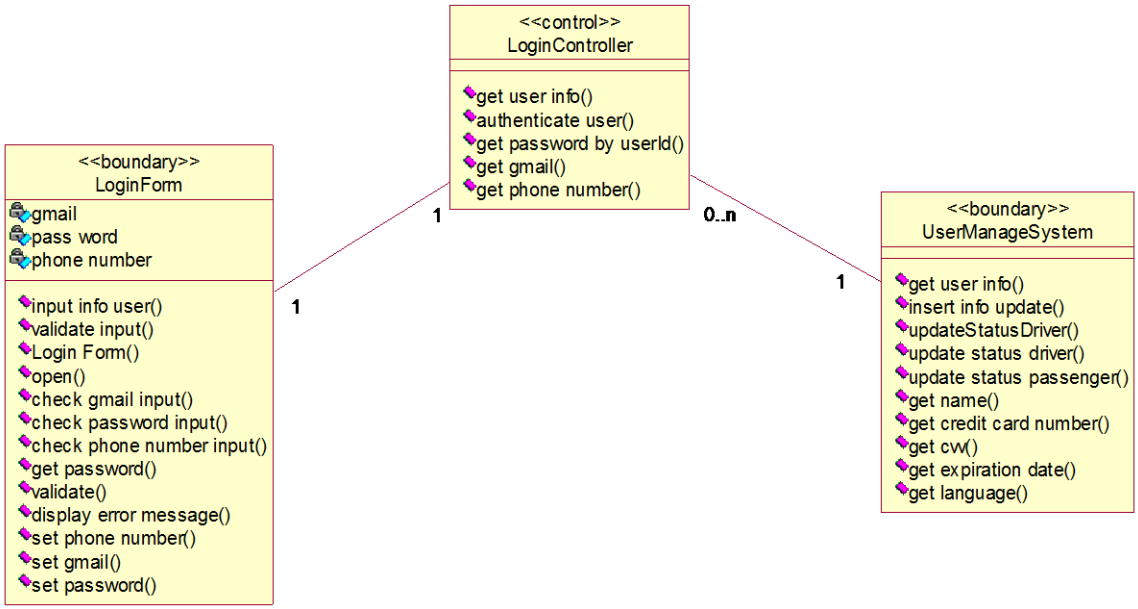


Figure 0.25 Login VOPC

2.2.2.2. Change profile

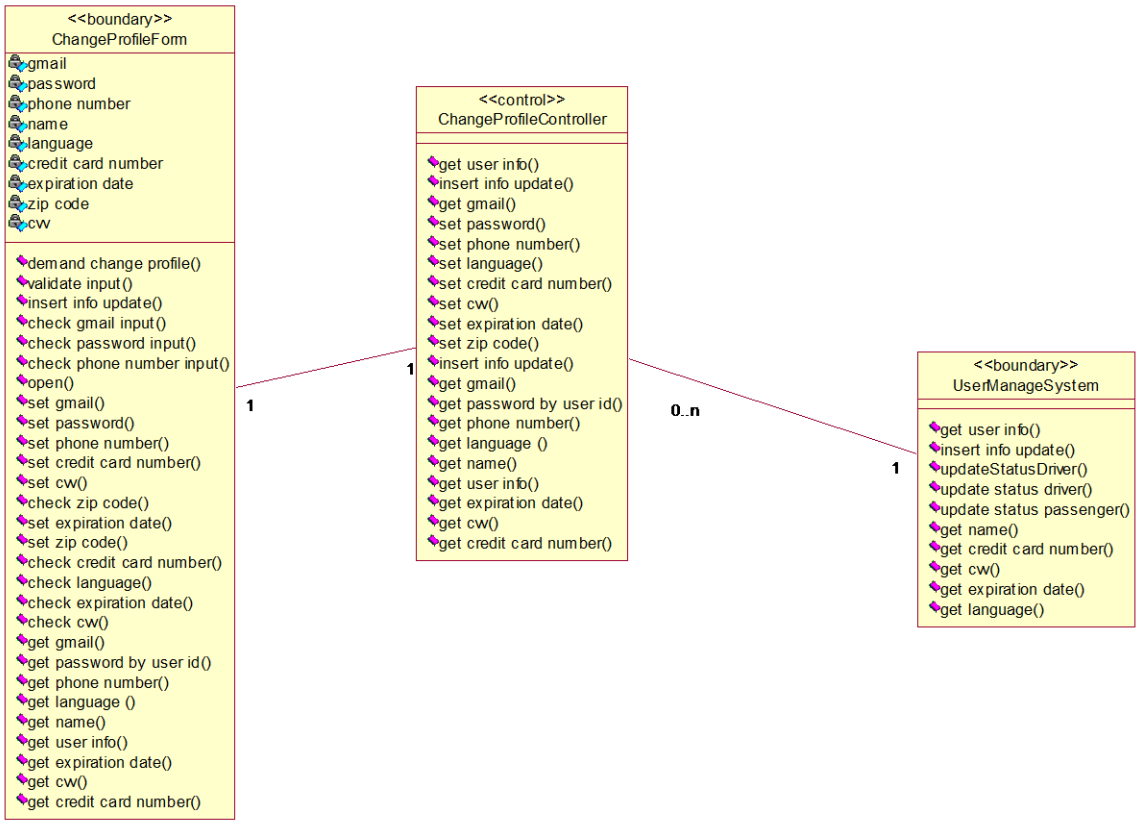


Figure 0.26 Change Profile VOPC

2.2.2.3. View Info Passenger

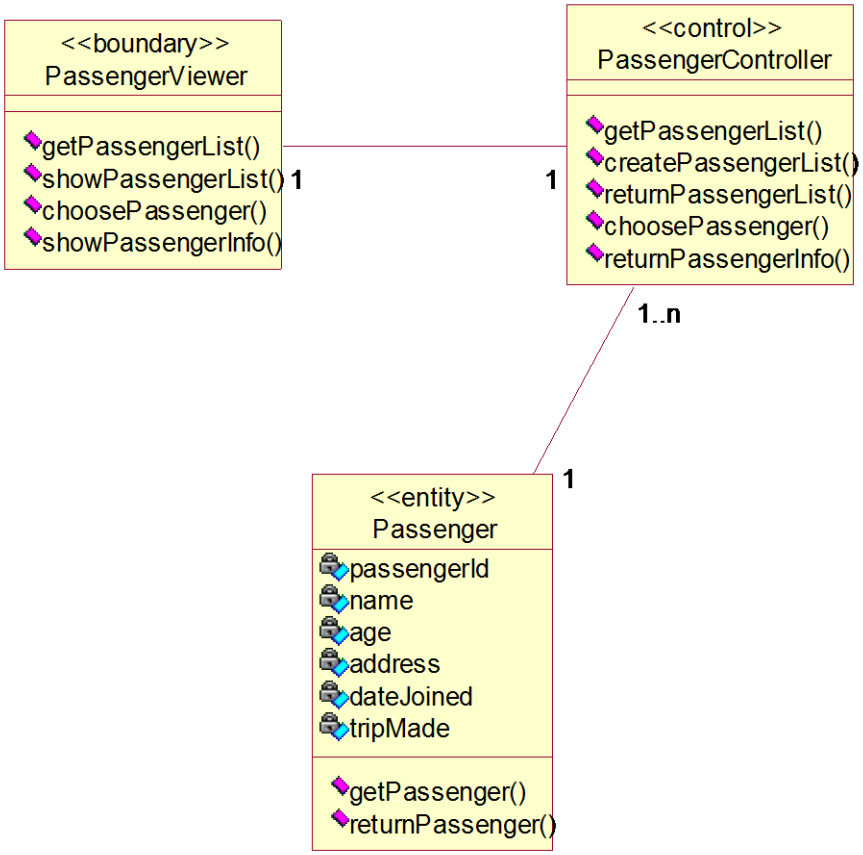


Figure 0.27 View Information Passenger VOPC

2.2.2.4. View Info Trip

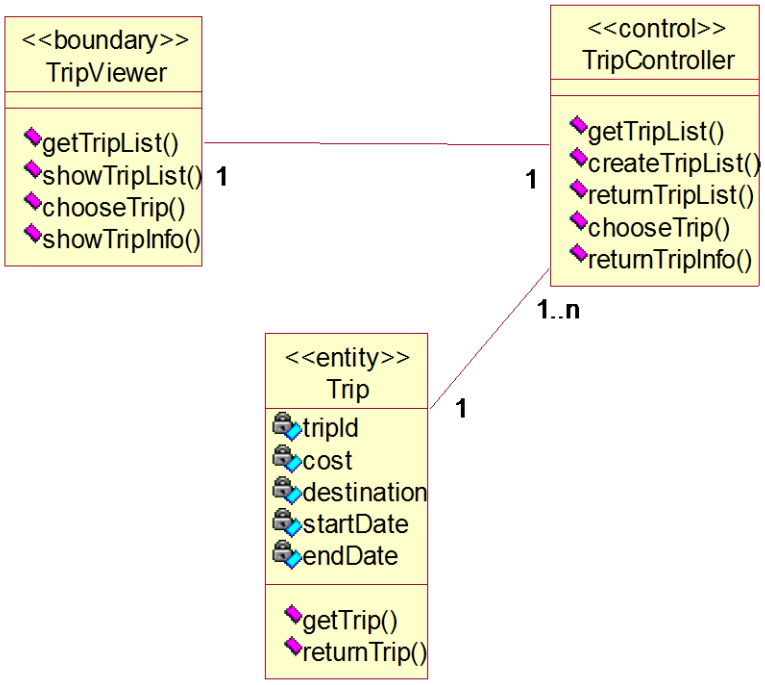


Figure 0.28 View Info Trip VOPC

2.2.2.5. View Notification

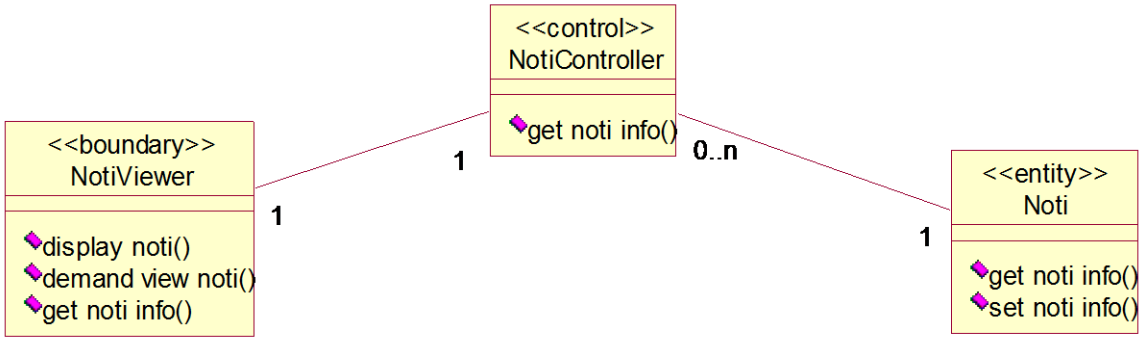


Figure 30 View Notification VOPC

2.2.2.6. Enter Pickup Location

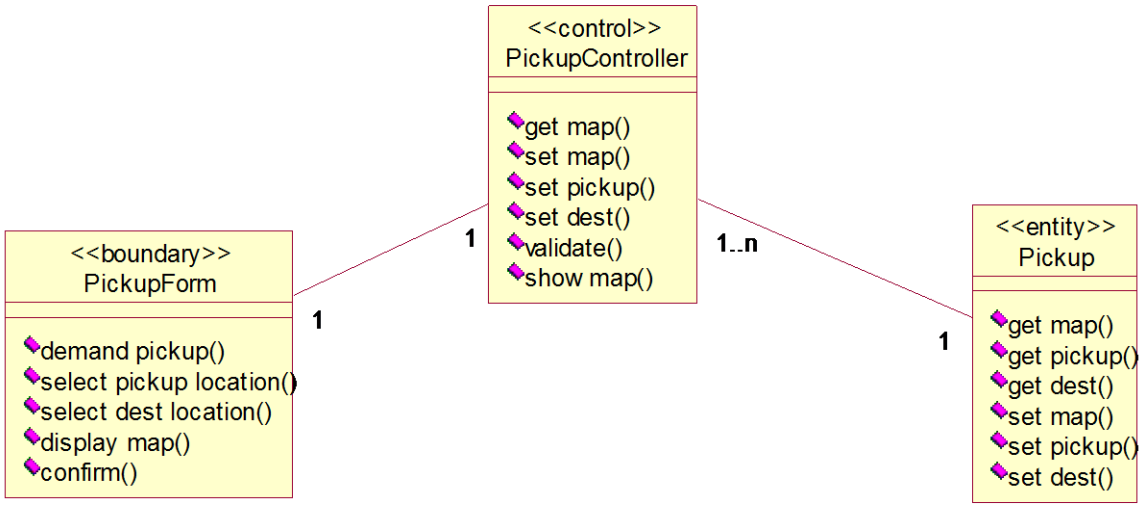


Figure 31 Enter Pickup Location VOPC

2.2.2.7. Billing

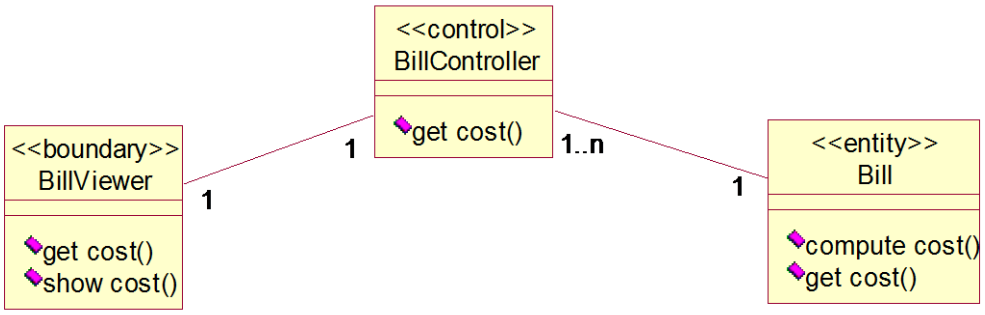


Figure 32 Billing VOPC

2.2.2.8. View Available Passenger

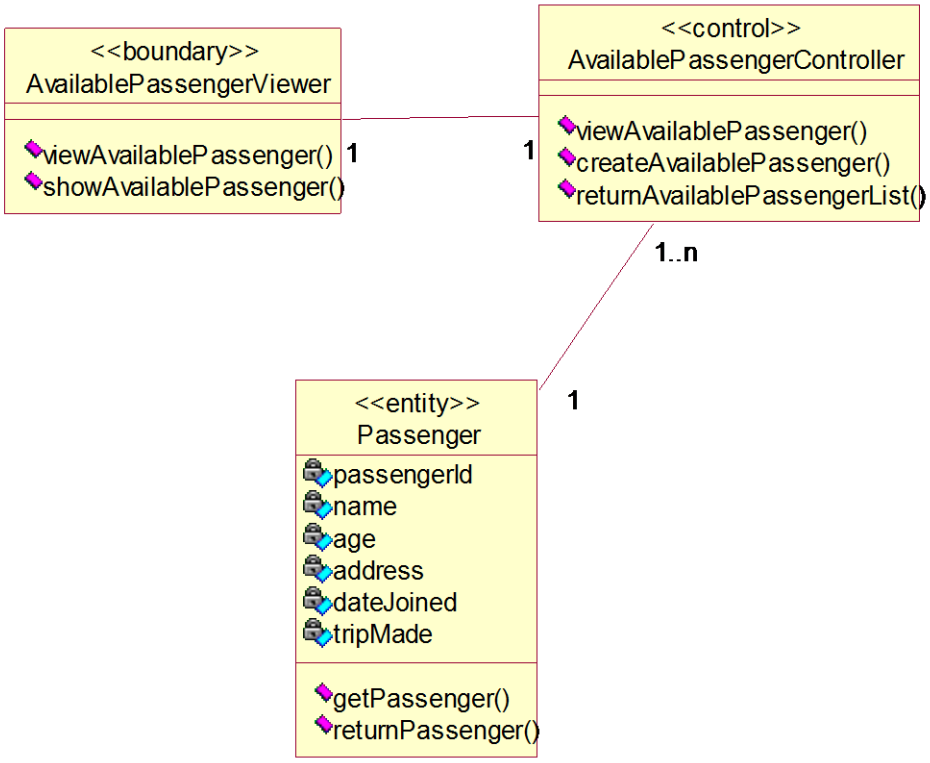


Figure 0.29 View Available Passenger VOPC

2.2.2.9. Register

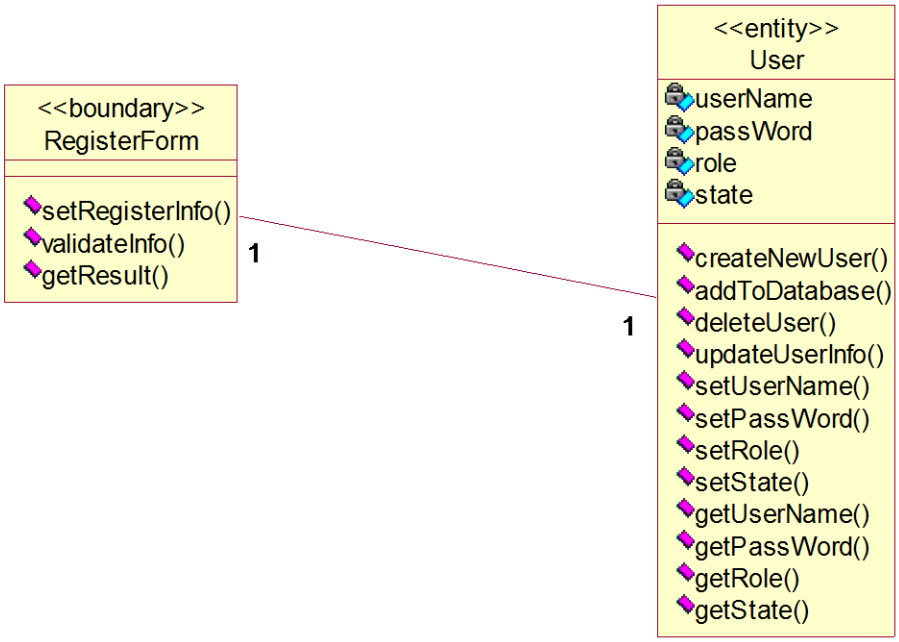


Figure 0.30 Register VOPC

2.2.2.10. Get statistic data

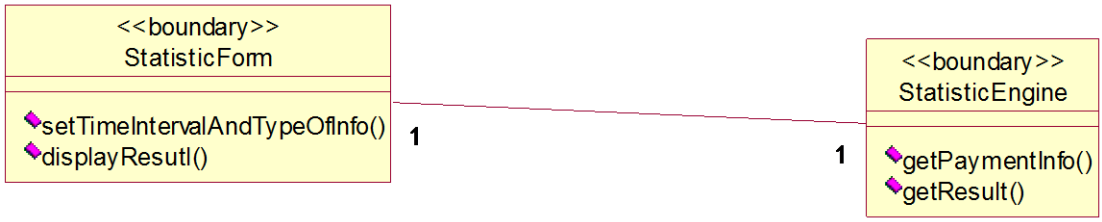


Figure 0.31 Get statistic data VOPC

2.2.2.11. View Info Driver

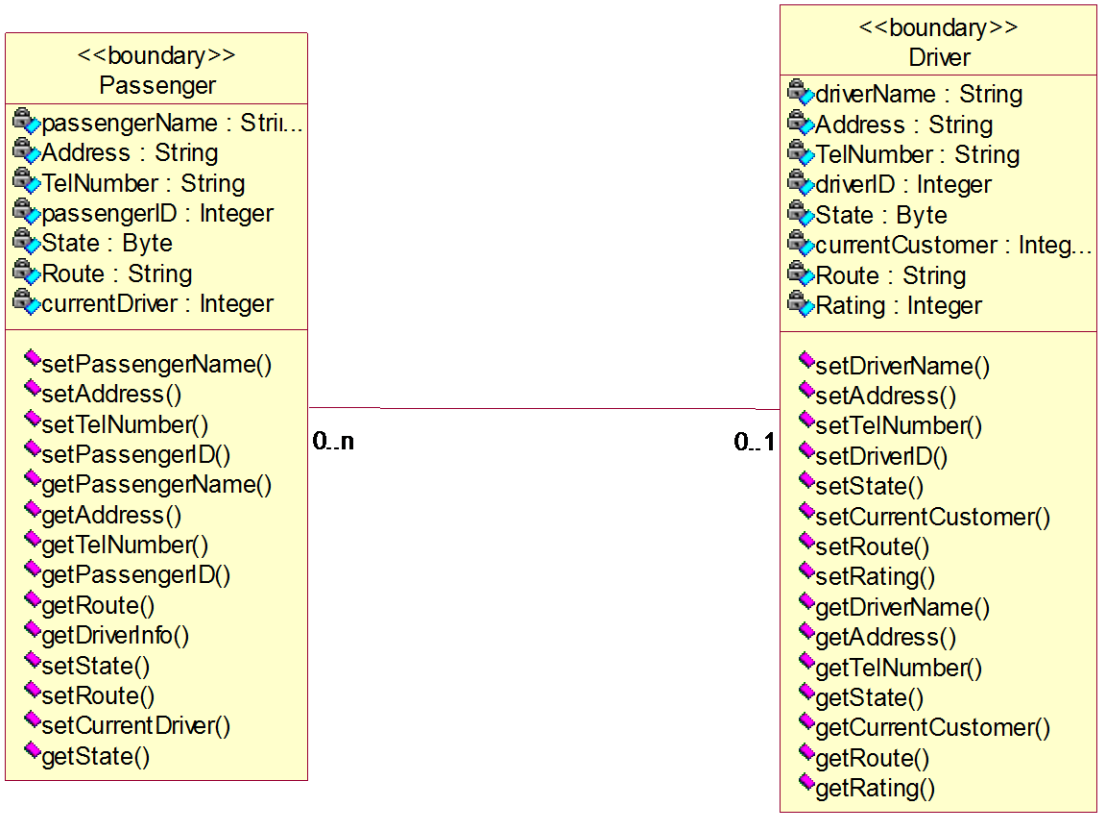


Figure 0.32 View Info Driver VOPC

2.2.2.12. Accept request

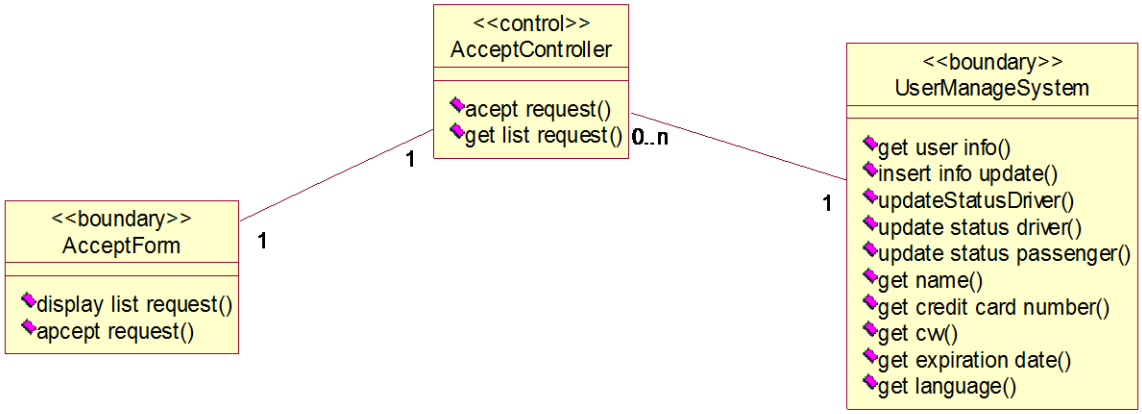


Figure 0.33 Accept Request VOPC

2.2.2.13. View history

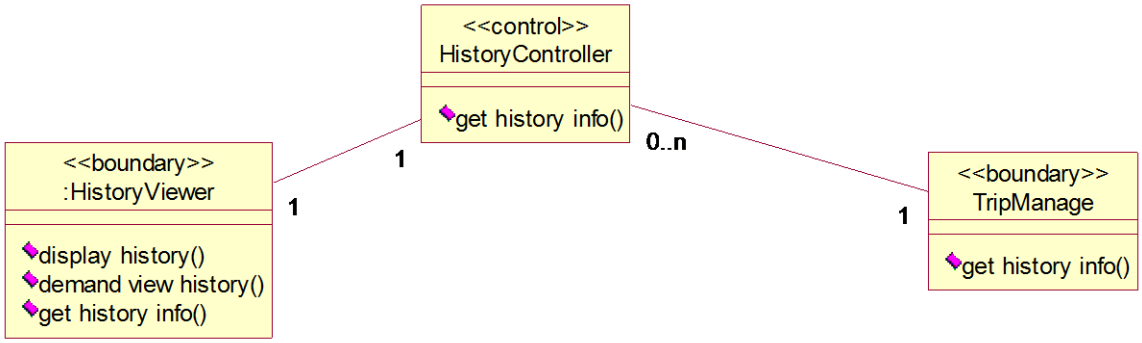


Figure 0.34 View History VOPC

2.2.2.14. Scheduler

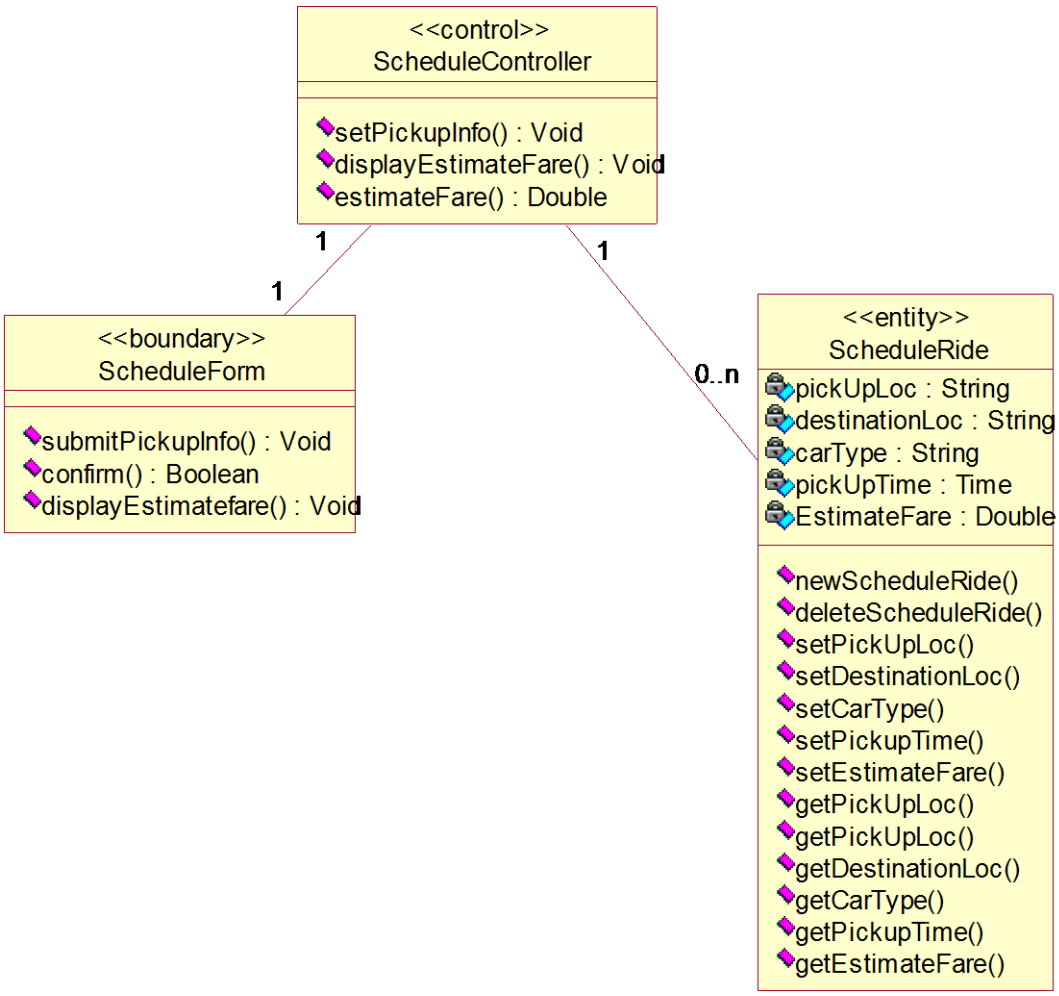


Figure 0.35 Scheduler VOPC

2.2.2.15. Promotions

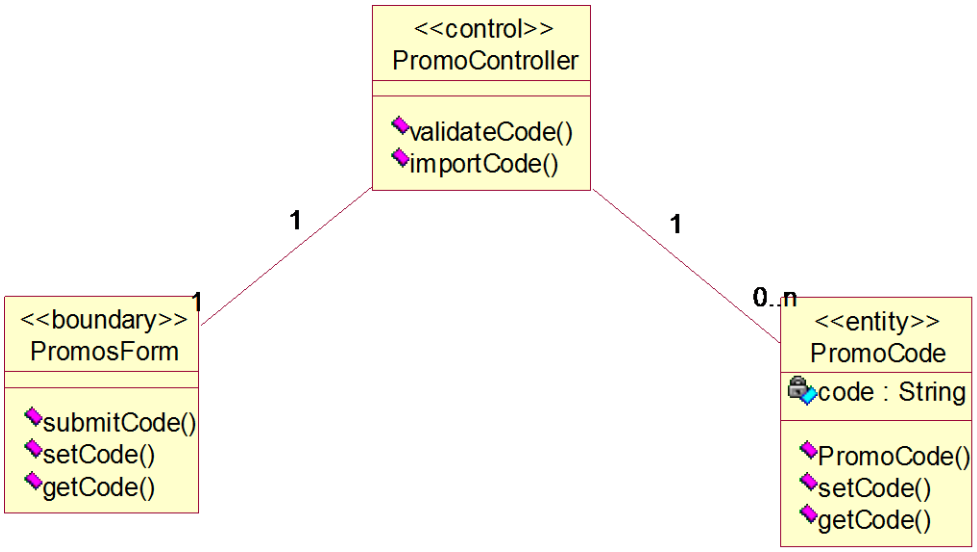


Figure 0.36 Promotion VOPC

2.2.2.16. View traffic map

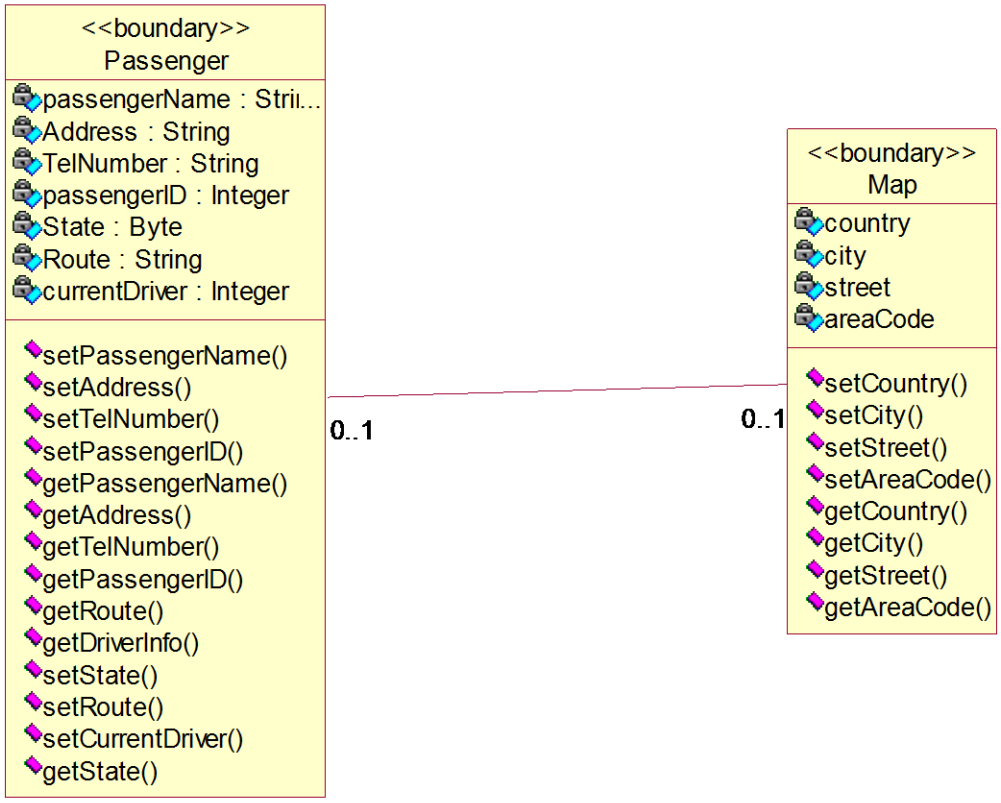


Figure 0.37 View Traffic Map VOPC

2.2.2.17. App setting

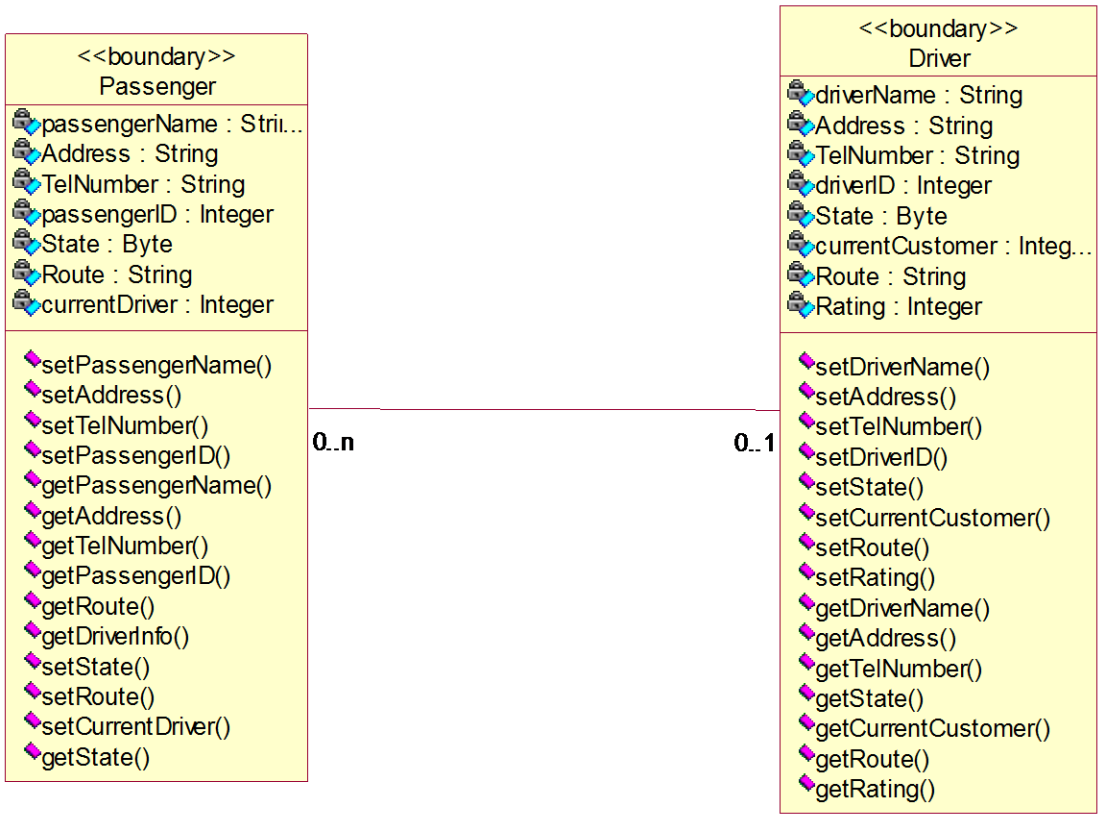


Figure 0.38 App Setting VOPC

2.2.2.18. View rating

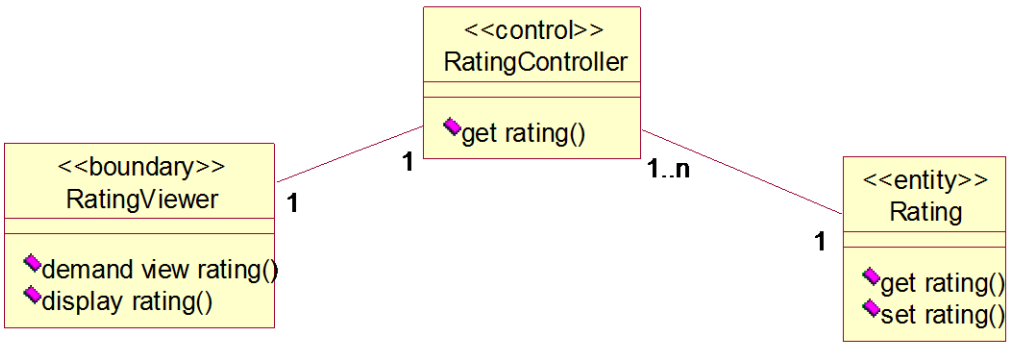


Figure 0.39 View Rating VOPC

2.2.2.19. Rating

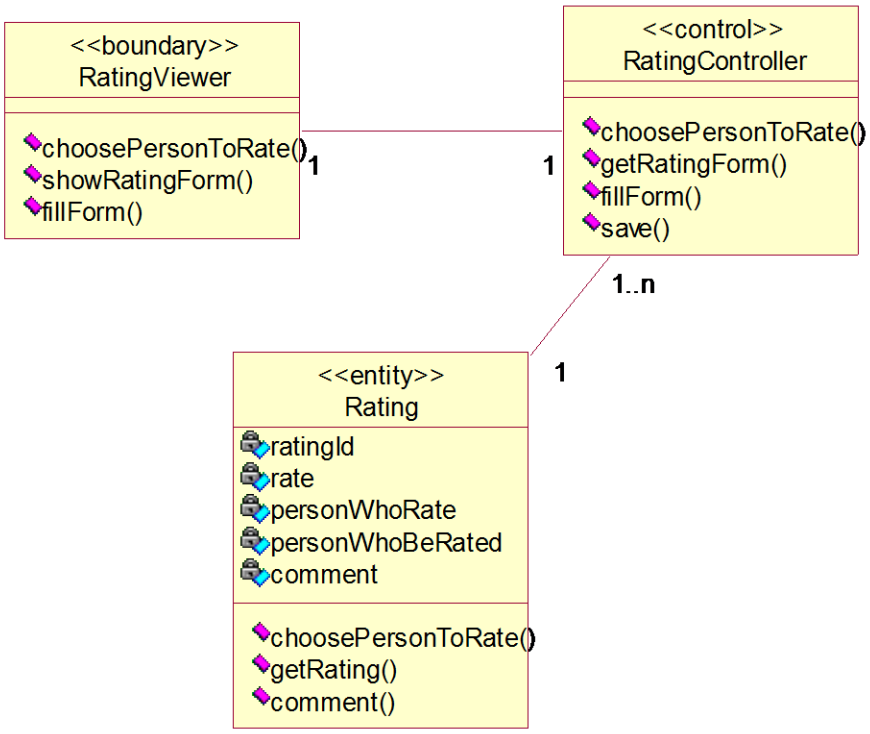


Figure 0.40 Rating VOPC

2.2.2.20. Contact

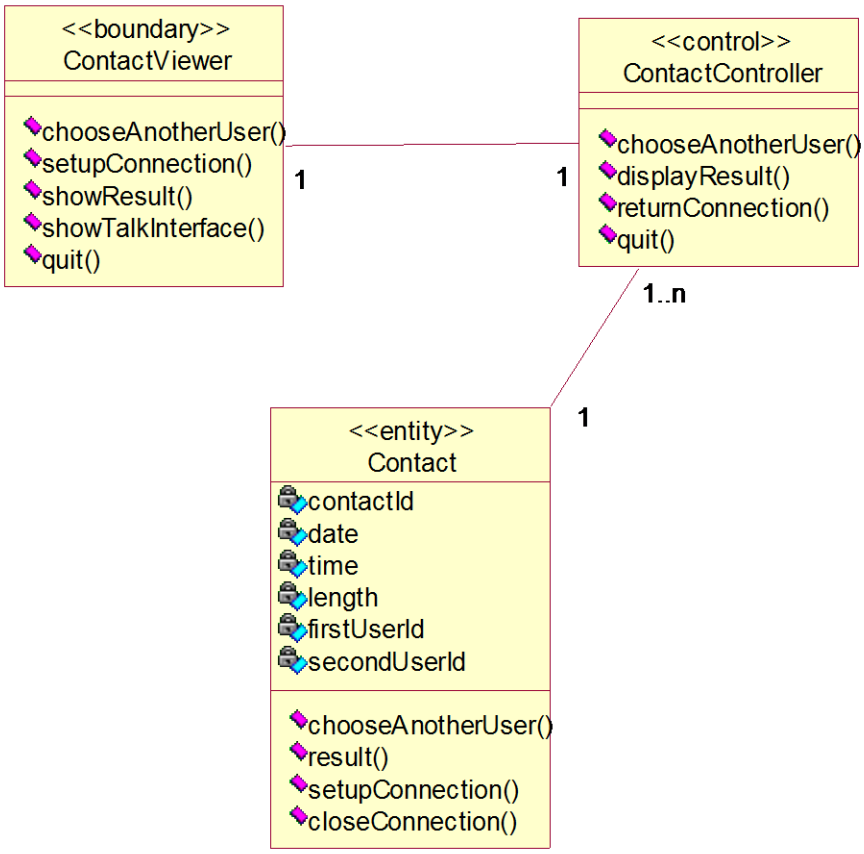


Figure 0.41 Contact VOPC

2.2.2.21. Manage driver

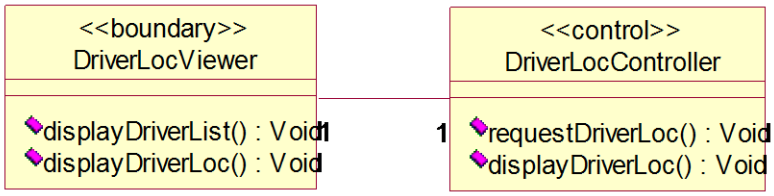


Figure 0.42 Manage Driver

2.2.2.22. Manage User info

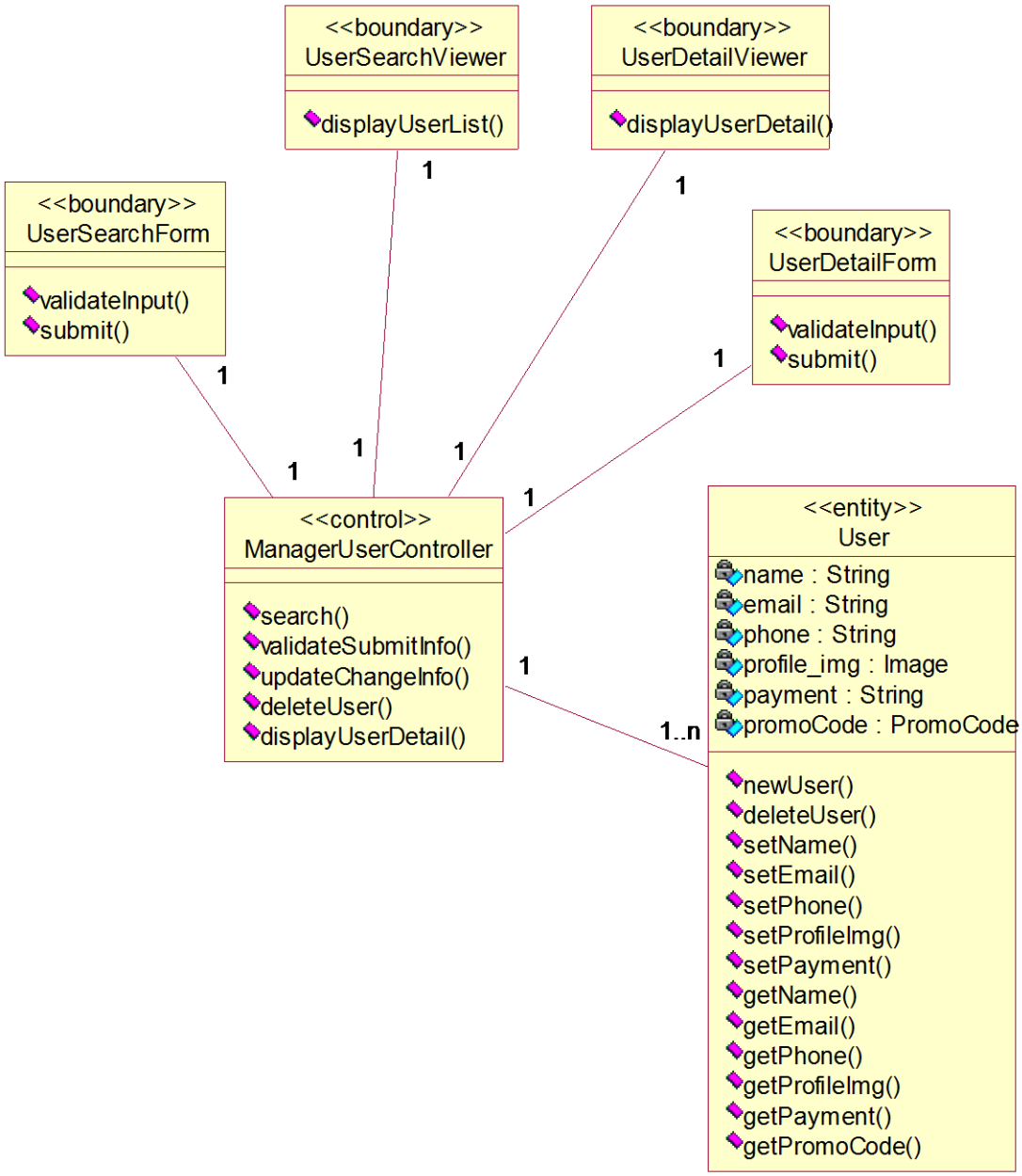


Figure 0.43 Manage User Info VOPC

2.2.2.23. Manage trip

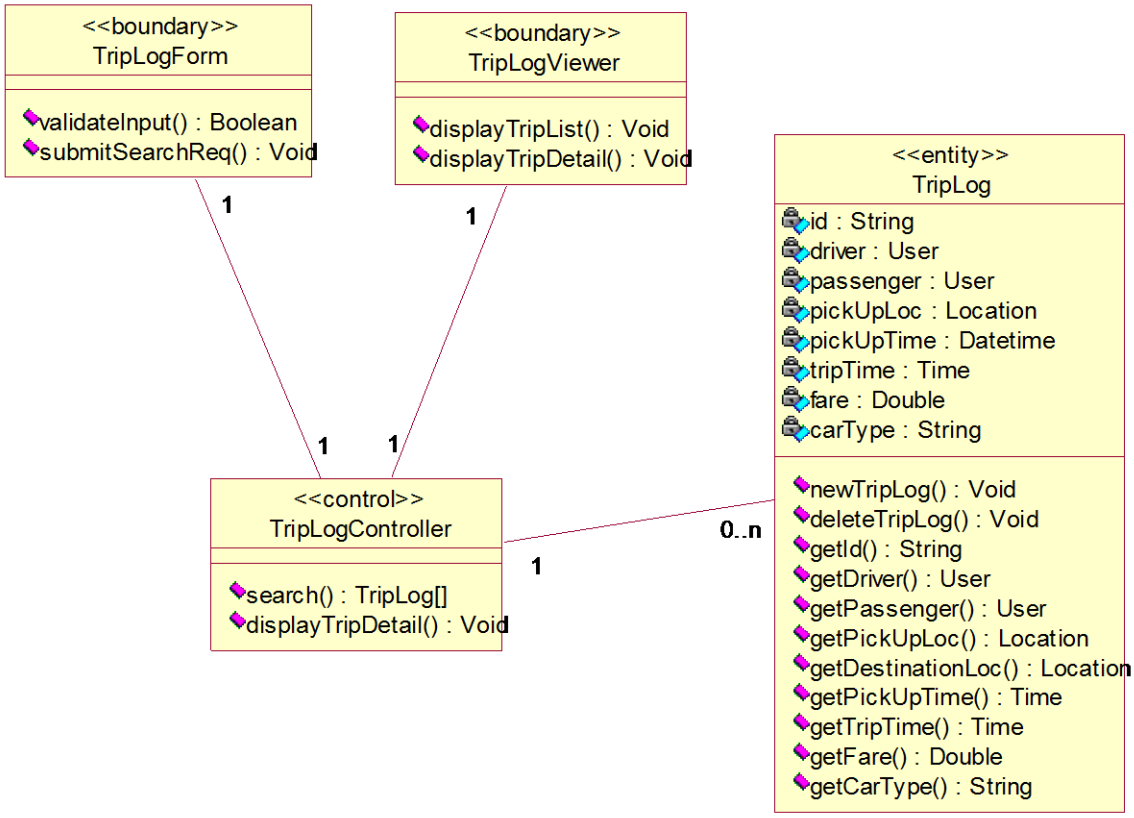


Figure 0.44 Manage Trip VOPC

2.2.2.24. Select car

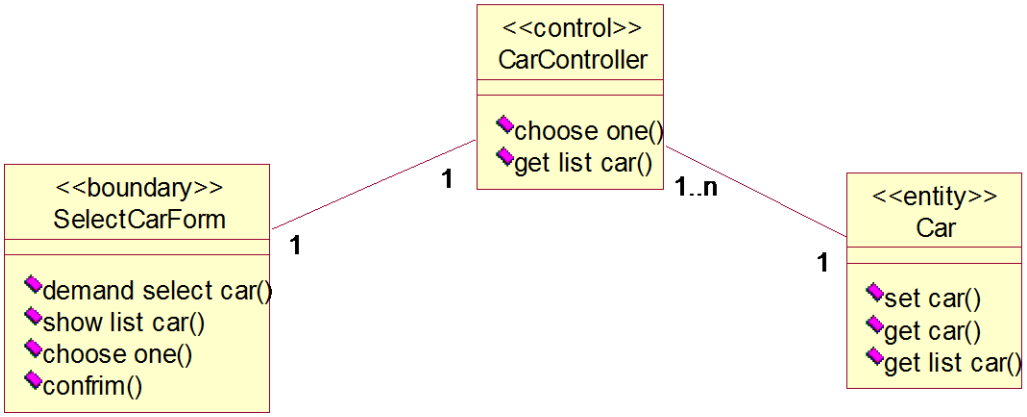


Figure 0.45 Select Car VOPC

2.2.2.25. Share info

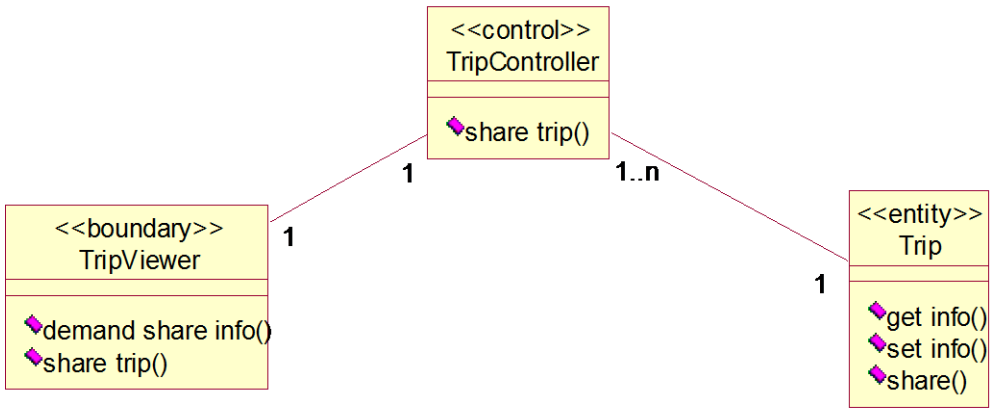


Figure 0.46 Share Info VOPC

2.2.3. Analysis-Class-To-Analysis-Mechanism Map

Analysis Class	**Analysis Mechanism**
LoginForm	Error detection / handling / reporting
LoginController	Communication
UserManagerSystem	Persistency, Security
ChangeProfileForm	Error detection / handling / reporting
ChangeProfileController	Communication
PassengerViewer	None
PassengerController	Communication
Passenger	Persistency, Security
TripViewer	None
TripController	Persistency, Security
TripManager	
Trip	
NotiViewer	None
NotiController	Communication

Noti	Persistency
PickupForm	None
PickupController	Communication
Pickup	None
BillViewer	None
BillController	Communication
Bill	Persistency, Security, Information exchange
AvailablePassengerViewer	None
AvailablePassengerController	Communication
RegisterForm	Error detection / handling / reporting
StatisticForm	None
StatisticEngine	Persistency, Security, Information exchange
Driver	Persistency, Security
AcceptForm	None
AcceptController	Communication
HistoryViewer	None
HistoryController	Communication
ScheduleForm	Error detection / handling / reporting
ScheduleController	None
ScheduleRide	Persistency, Security
PromosForm	None
PromosController	Communication
Map	Persistency, Process control and synchronization
RatingViewer	None

RatingController	Communication
Rating	Persistency
ContactViewer	None
ContactController	Communication
Contact	Persistency, Security, Message routing
DriverLocViewer	None
DriverLocController	Communication
UserSearchForm	None
UserSearchViewer	Error detection / handling / reporting
UserDetailViewer	None
UserDetailForm	None
ManageUserController	Persistency, Security, Communication
User	
UserManagerSubsystem	
TripLogForm	None
TripLogViewer	None
TripLogController	Communication
TripLog	Persistency, Security
SelectCarForm	None
CarController	Communication
Car	Persistency

3. Uber Design

3.1. Identify Design Elements

3.1.1. Subsystem Context Diagram

3.1.1.1 Billing System

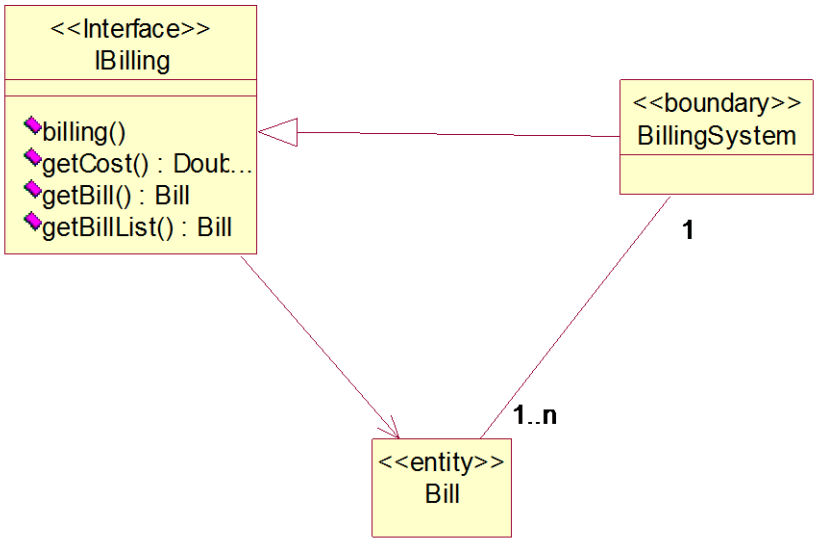


Figure 0.47 Billing System

3.1.1.2 User Manage System

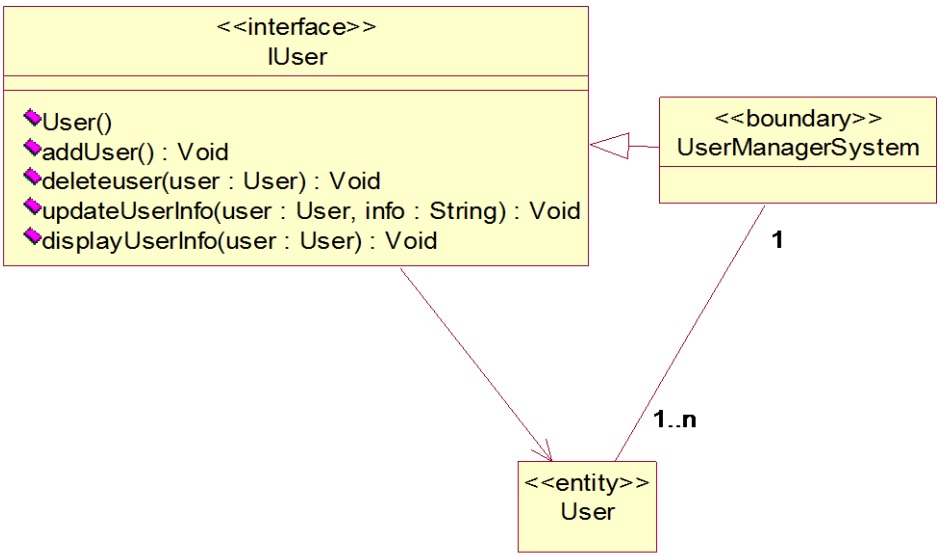


Figure 0.48 User Manage System

3.1.1.3 Trip Manage System

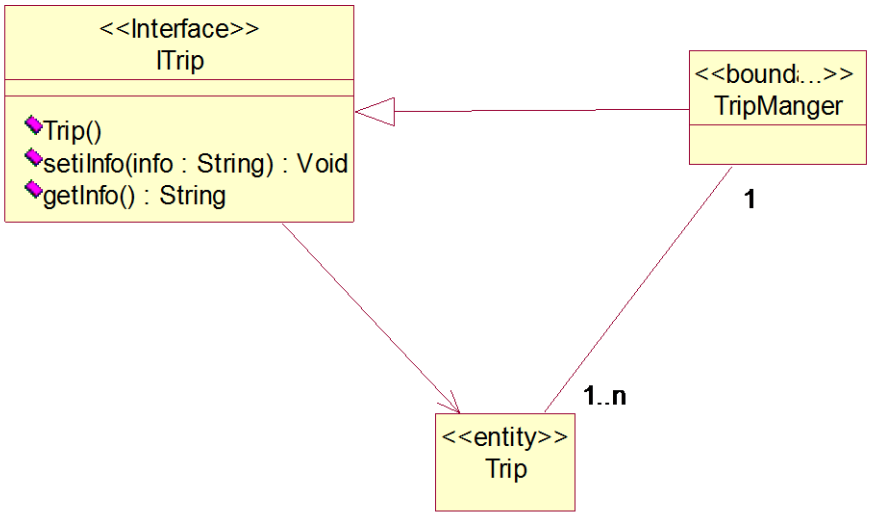


Figure 0.49 Trip Manage System

3.1.2. Analysis-Class-To-Design-Element Map

Analysis Class	**Design Element**
LoginForm	LoginForm
LoginController	LoginController
UserManagerSystem	UserManagerSystem
ChangeProfileForm	ChangeProfileForm
ChangeProfileController	ChangeProfileController
UserManagerSystem	UserManagerSubSystem
PassengerViewer	PassengerViewer
PassengerController	PassengerController
Passenger	Passenger
TripViewer	TripViewer
TripController	TripSubSystem
TripManager	
Trip	
NotiViewer	NotiViewer
NotiController	NotiController
Noti	Noti

PickupForm	PickupForm
PickupController	PickupController
Pickup	Pickup
BillViewer	BillViewer
BillController	BillSubSystem
Bill	
AvailablePassengerViewer	AvailablePassengerViewer
AvailablePassengerController	AvailablePassengerController
Passenger	Passenger
RegisterForm	RegisterForm
User	User
StatisticForm	StatisticForm
StatisticEngine	StatisticEngine
Customer	Customer
Driver	Driver
AcceptForm	AcceptForm
AcceptController	AcceptController
HistoryViewer	HistoryViewer
HistoryController	HistoryController
ScheduleForm	ScheduleForm
ScheduleController	ScheduleController
ScheduleRide	ScheduleRide
PromosForm	PromosForm
PromosController	PromosController
Customer	Customer
Map	Map
RatingViewer	RatingViewer
RatingController	RatingController
Rating	Rating
ContactViewer	ContactViewer
ContactController	ContactController
Contact	Contact
DriverLocViewer	DriverLocViewer
DriverLocController	DriverLocController
UserSearchForm	UserSearchForm
UserSearchViewer	UserSearchViewer
UserDetailViewer	UserDetailViewer

UserDetailForm	UserDetailForm
ManageUserController	UserManagerSubsystem
User	
UserManagerSubsystem	
TripLogForm	TripLogForm
TripLogViewer	TripLogViewer
TripLogController	TripLogController
TripLog	TripLog
SelectCarForm	SelectCarForm
CarController	CarController
Car	Car

3.1.3. Design-Element-To-Owning-Package Map

Design Element	**"Owning" Packet**
LoginForm	**Controller::GUI Controller**
ChangeProfileForm	
PickupForm	
RegisterForm	
StatisticForm	
AcceptForm	
ScheduleForm	
PromosForm	
UserSearchForm	
UserDetailForm	
TripLogForm	
SelectCarForm	
PassengerViewer	
NotiViewer	
BillViewer	
AvailablePassengerViewer	
HistoryViewer	
RatingViewer	
ContactViewer	

DriverLocViewer	
UserSearchViewer	
UserDetailViewer	
TripLogViewer	
TripViewer	
LoginController	**Controller::Subsystem**
ChangeProfileController	
PassengerController	
TripController	
UserManagerSystem	
TripManager	
ScheduleRide	**Controller::Activity**
Passenger	**Controller::Uber Elements**
Trip	
Noti	
Pickup	
Bill	
User	
StatisticEngine	
Customer	
Driver	
PromoCode	
Map	
Rating	
Contact	
User	
Car	
Trip	

3.1.4. Architectural Components and Their Dependencies

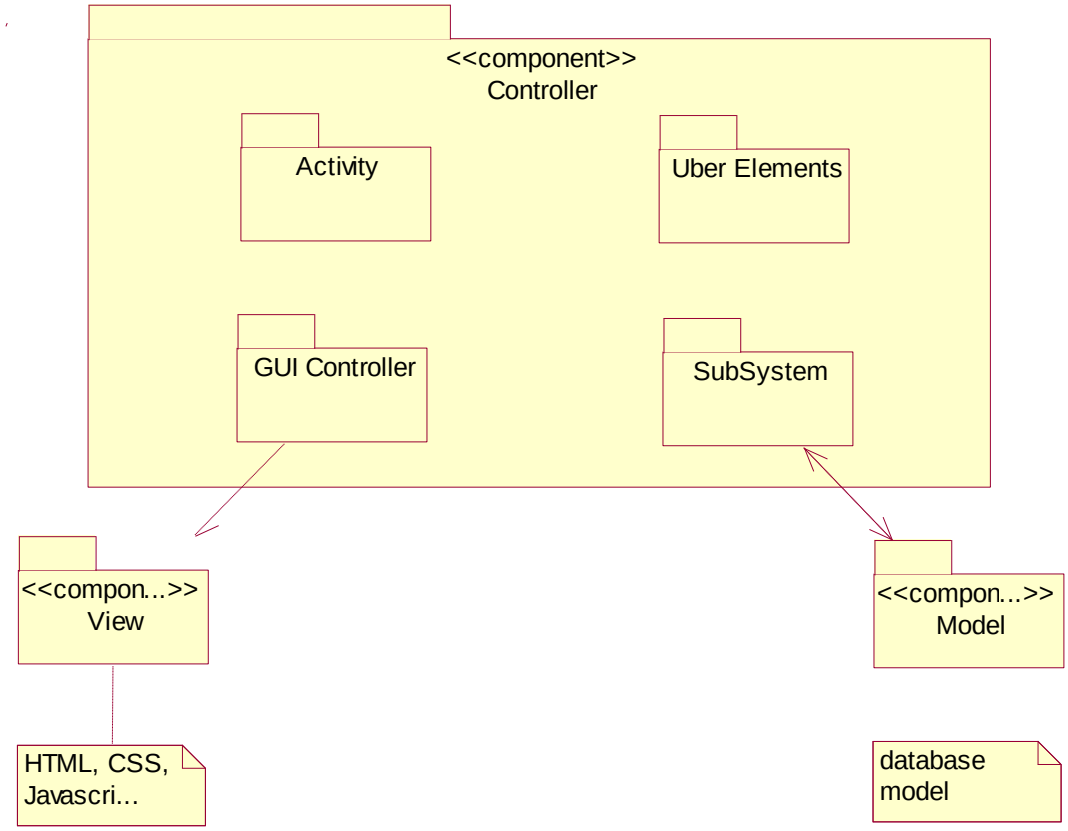


Figure 0.50 Architectural Components and Their Dependencies

3.1.5. Package and Their Dependencies

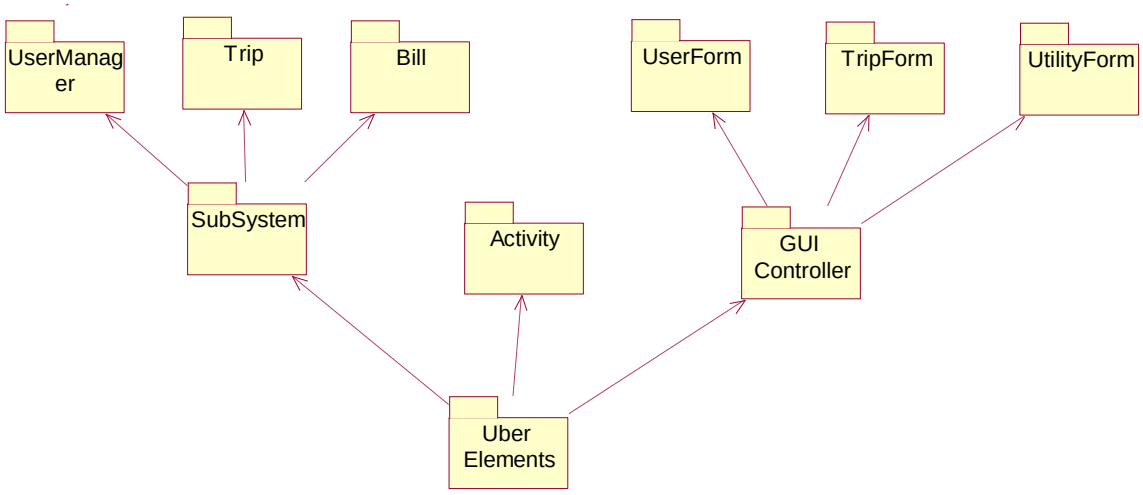


Figure 0.51 Pakage and Their Dependencies

3.2. Describe the Run-time Architectural

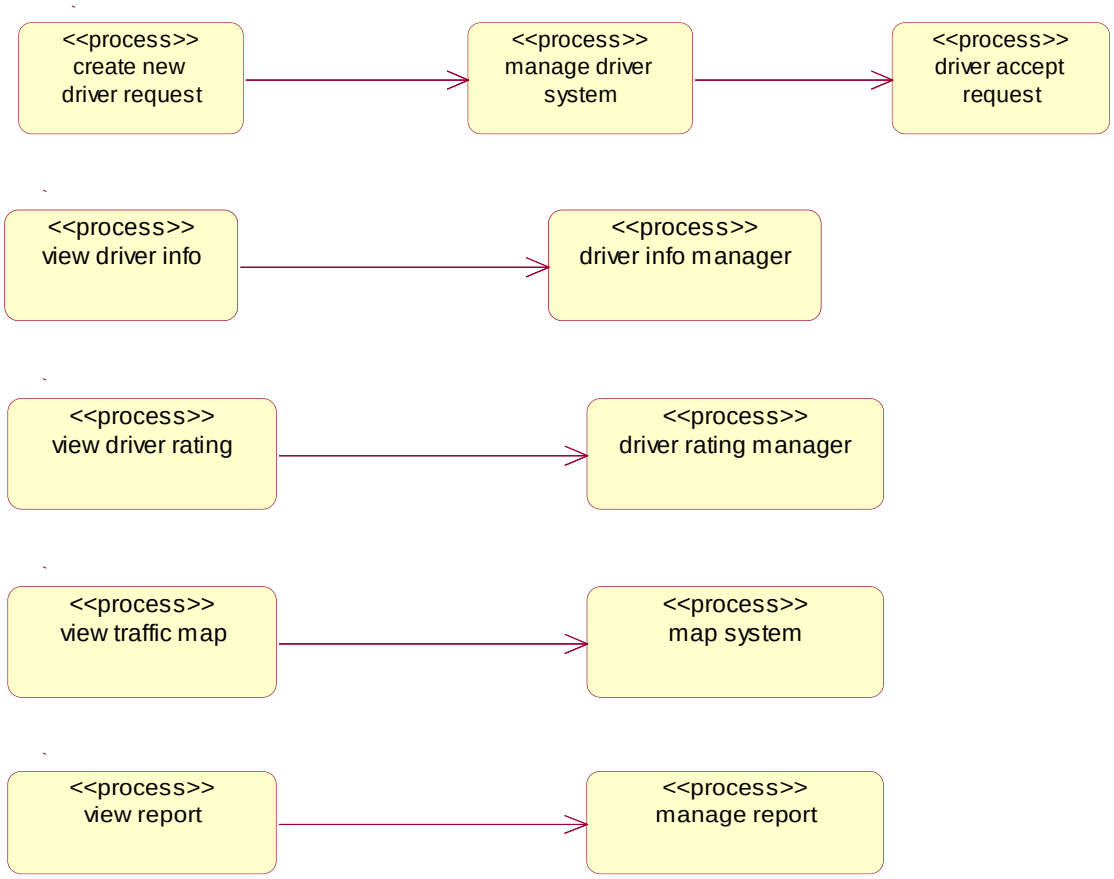


Figure 0.52 Describe the Run-time Architectural

3.3. Describe Distribution

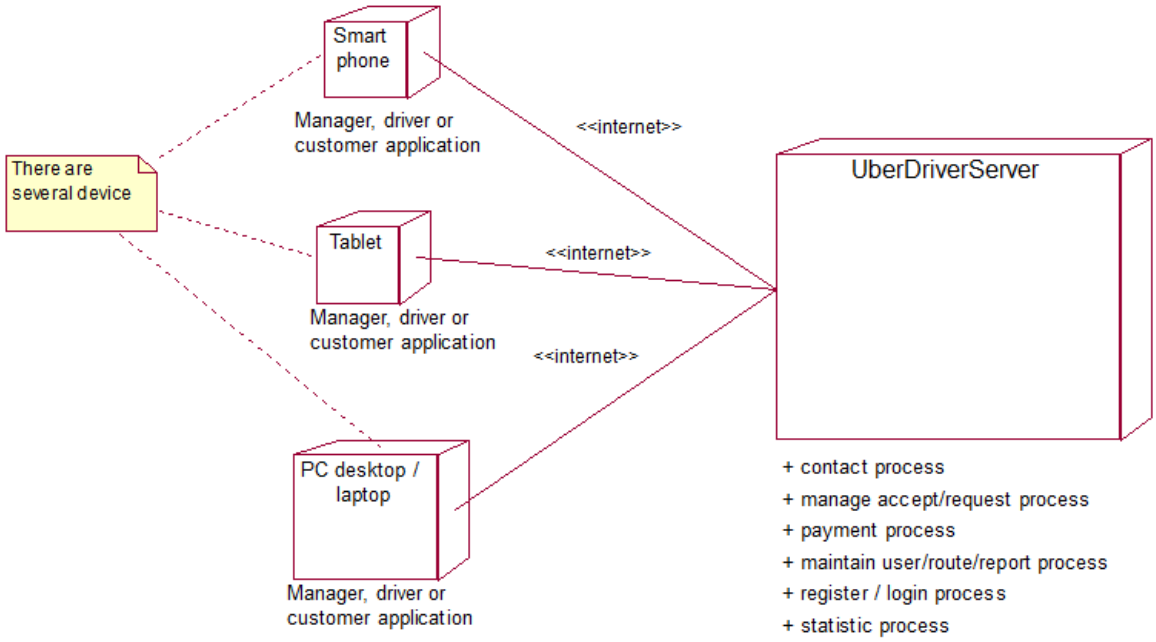


Figure 0.53 Describe Distribution

3.4. Use case Design

This part describes the use case in the consideration of every analysis mechanism that were defined before.

None

3.5. Class Design

3.5.1. Describe each class or interface

3.5.1.1. NotiViewer

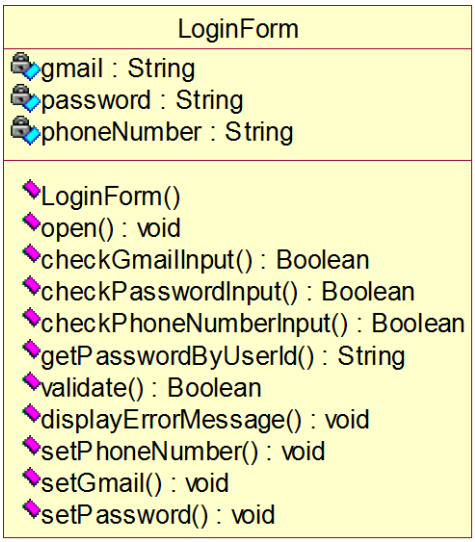


Figure 0.54 Class LoginForm

3.5.1.2. LoginController

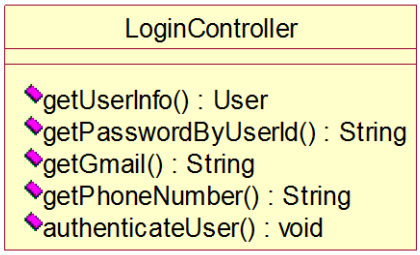


Figure 0.55 Class Login Controller

3.5.1.3. Noti

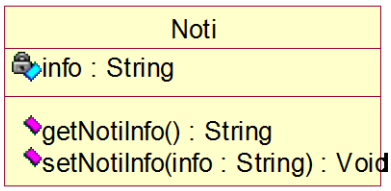


Figure 0.56 Class Noti

3.5.1.4. BillViewer

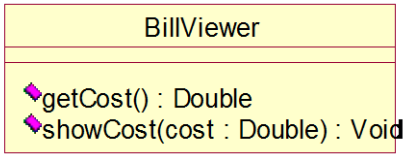


Figure 0.57 Class BillViewer

3.5.1.5. BillController

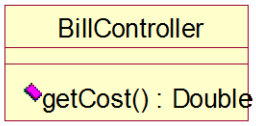


Figure 0.58 Class BillController

3.5.1.6. PickupForm

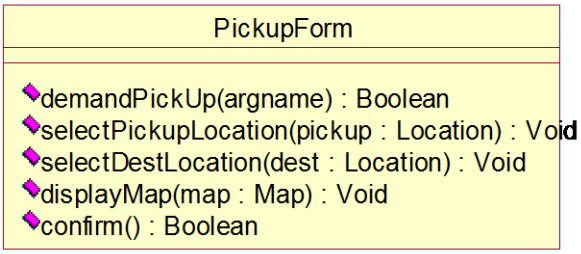


Figure 0.59 Class PickupForm

3.5.1.7. PickupController

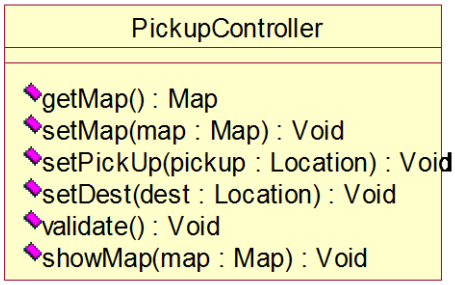


Figure 0.60 Class PickupController

3.5.1.8. Pickup

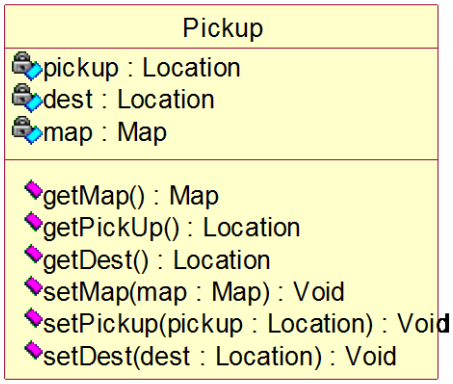


Figure 0.61 Class Pickup

3.5.1.9. MapCatalog

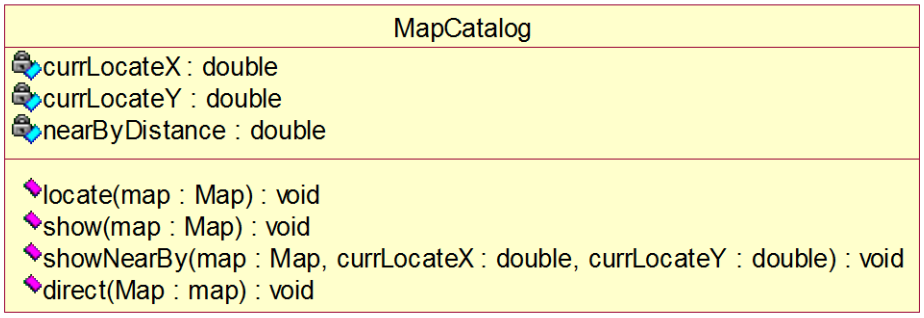


Figure 0.62 Class MapCatalog

3.5.1.10. PassengerViewer

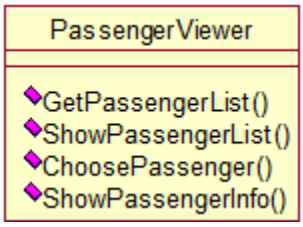


Figure 0.63 Class PassengerViewer

3.5.1.11. PassengerController

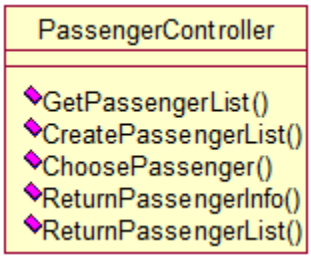


Figure 0.64 Class PassengerController

3.5.1.12. Passenger

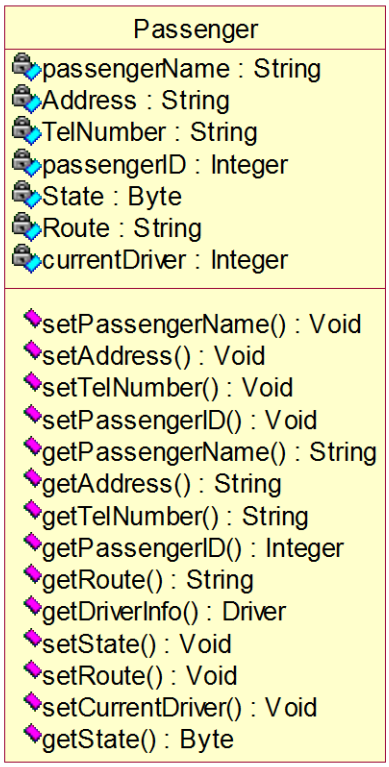


Figure 0.65 Class Passenger

3.5.1.13. TripController

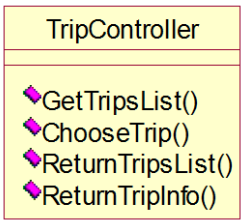


Figure 0.66 Class TripController

3.5.1.14. RatingViewer

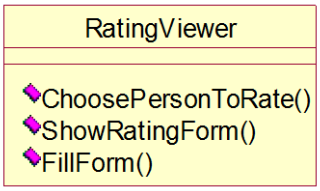


Figure 0.67 Class RatingViewer

3.5.1.15. RatingController

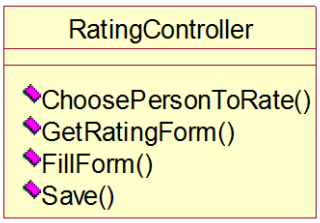


Figure 0.68 Class RatingController

3.5.1.16. Rating

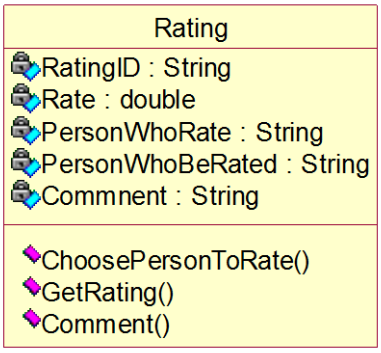


Figure 0.69 Class Rating

3.5.1.17. RatingForm

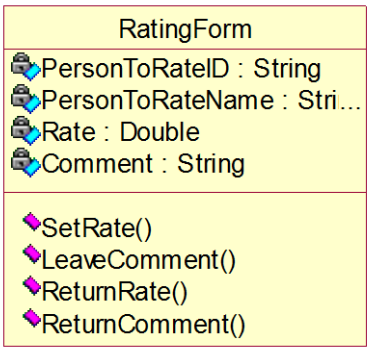


Figure 0.70 Class RatingForm

3.5.1.18. ContactViewer

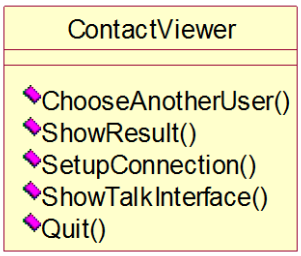


Figure 0.71 Class ContactViewer

3.5.1.19. ContactController

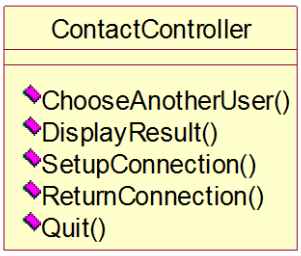


Figure 0.72 Class ContactController

3.5.1.20. Contact

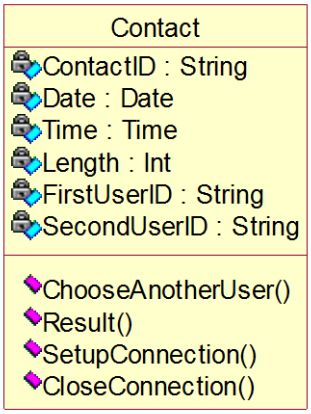


Figure 0.73 Class Contact

3.5.1.21. RegisterForm

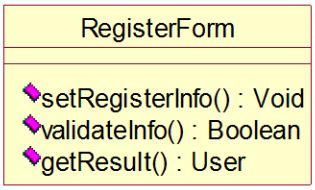


Figure 0.74 Class RegisterForm

3.5.1.22.User

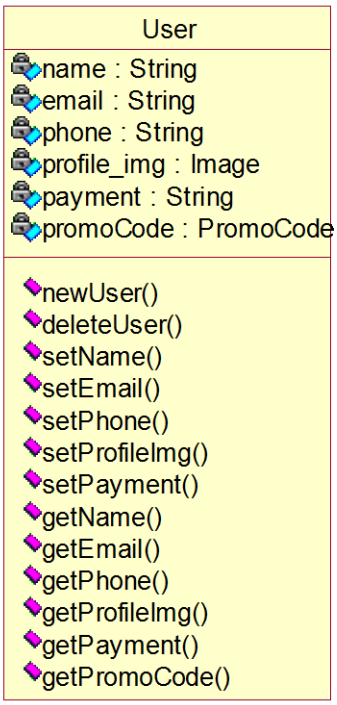


Figure 0.75 Class User

3.5.1.23. StatisticForm

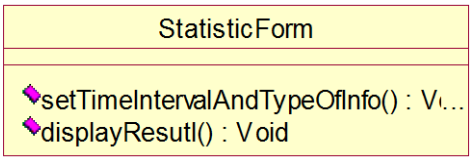


Figure 0.76 Class StatisticForm

3.5.1.24. StatisticEngine

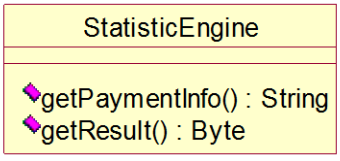


Figure 0.77 Class StatisticEngine

3.5.1.25.Customer

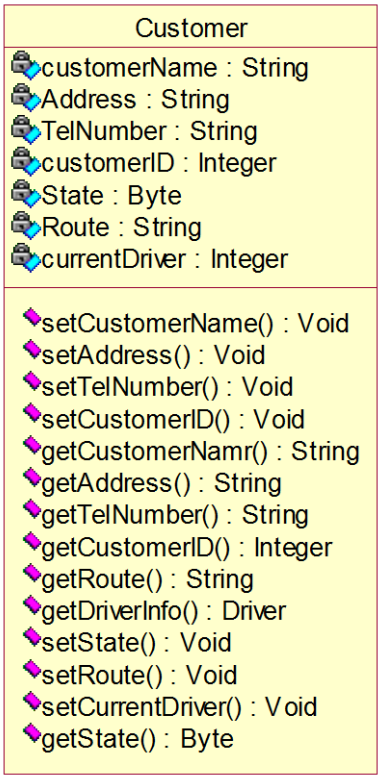


Figure 0.78 Class Customer

3.5.1.26. Driver

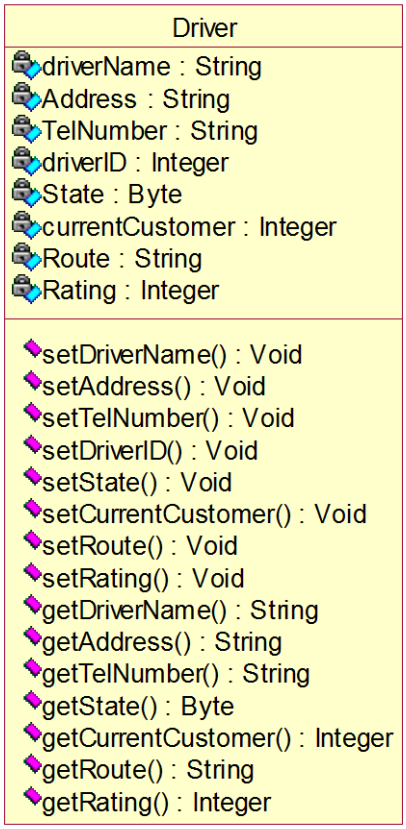


Figure 0.79 Class Driver

3.5.1.27. SystemInfo

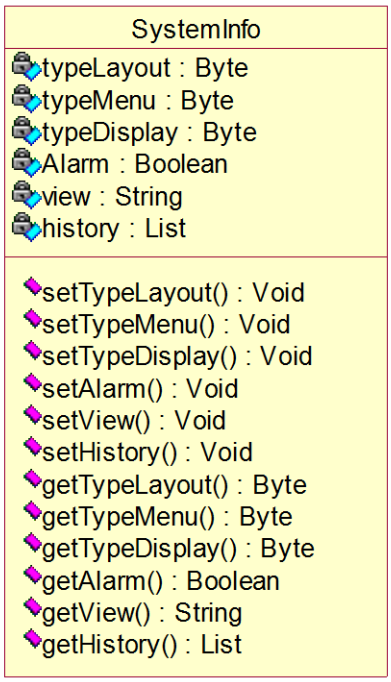


Figure 0.80 Class SystemInfo

3.5.1.28. Map

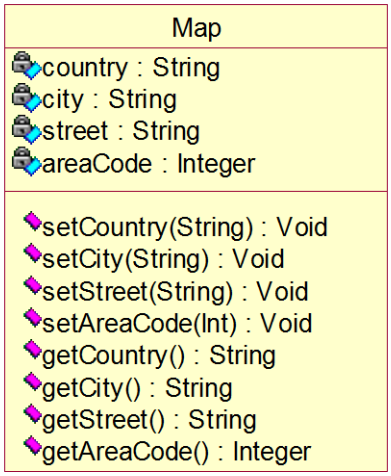


Figure 0.81 Class Map

3.5.1.29. SelectCarForm

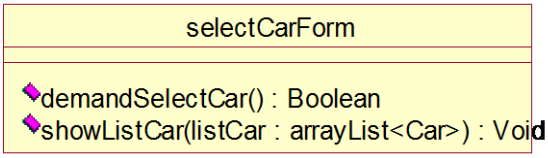


Figure 0.82 Class selectCarForm

3.5.1.30. CarController

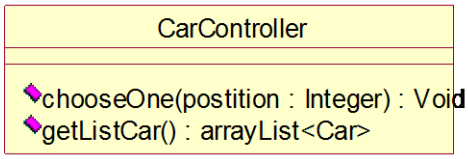


Figure 0.83 Class CarController

3.5.1.31. Car

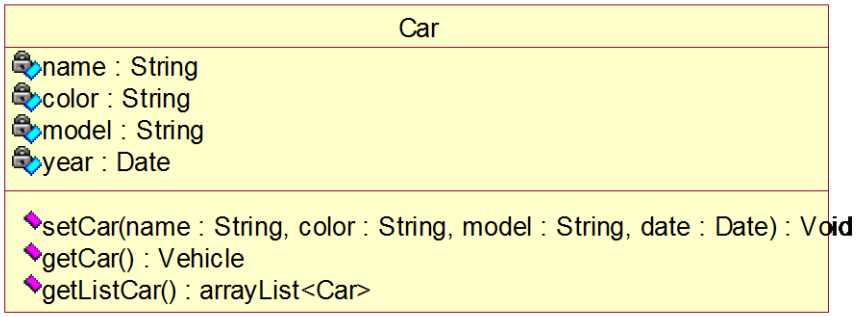


Figure 0.84 Class Car

3.5.1.32. Viewer

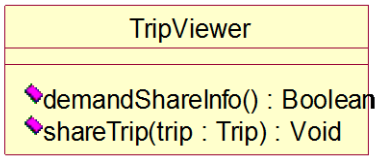


Figure 0.85 Class Viewer

3.5.1.33. Trip

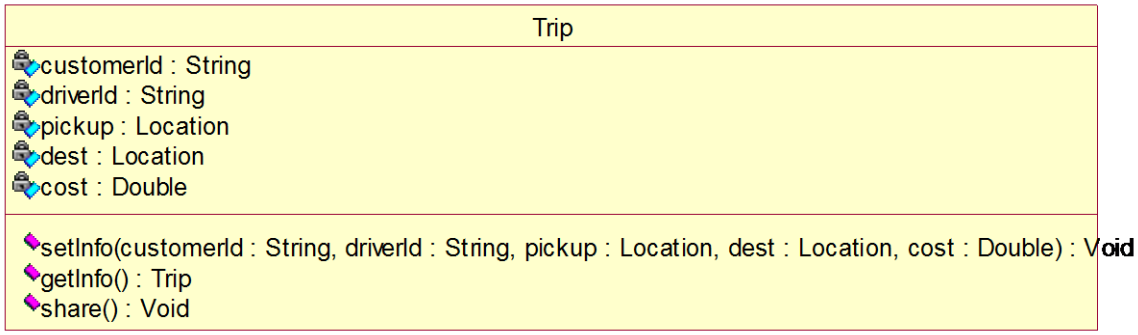


Figure 0.86 Class Trip

3.5.1.34. Bill

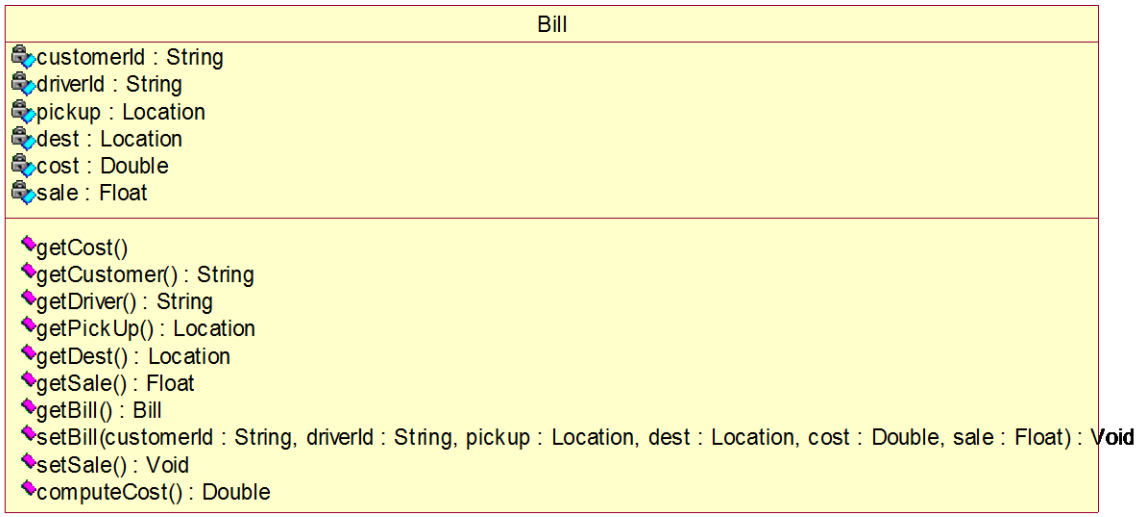


Figure 0.87 Class Bill

3.5.1.35. ScheduleController

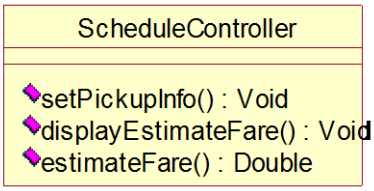


Figure 0.88 Class ScheduleController

3.5.1.36. ScheduleForm

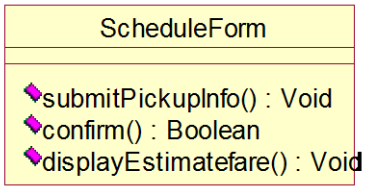


Figure 0.89 Class ScheduleForm

3.5.1.37. ScheduledRide

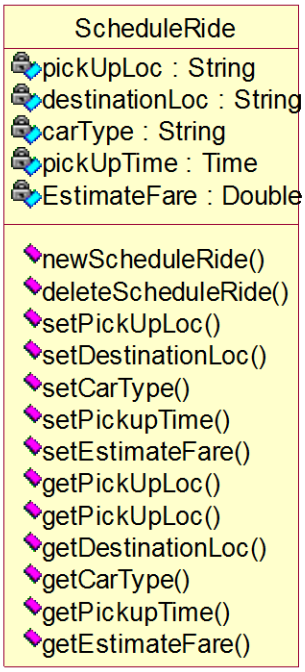


Figure 0.90 Class ScheduleRide

3.5.1.38. PromosForm

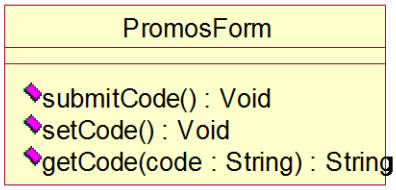


Figure 0.91 Class PromosForm

3.5.1.39. PromosController

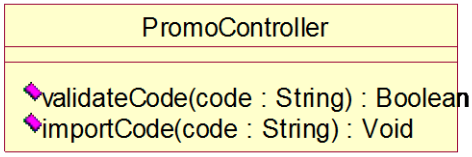


Figure 0.92 Class PromosController

3.5.1.40. PromosCode

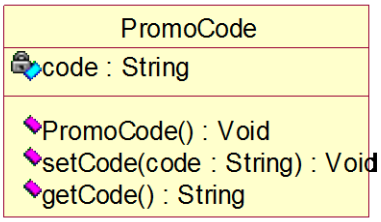


Figure 0.93 Class PromoCode

3.5.1.41. UserSearchForm

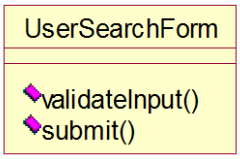


Figure 0.94 Class UserSearForm

3.5.1.42. UserSearchViewer

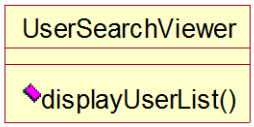


Figure 0.95 Class UserSearchViewer

3.5.1.43. UserDetailForm

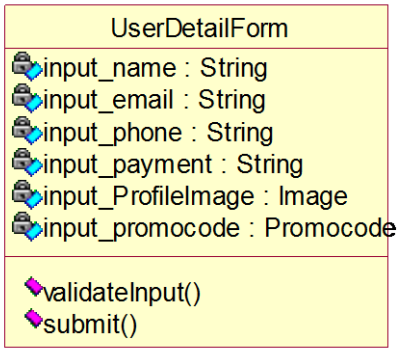


Figure 0.96 Class UserDetailForm

3.5.1.44. UserDetailViewer

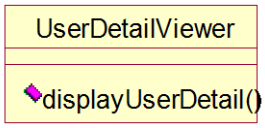


Figure 0.97 Class UserDetailViewer

3.5.1.45. ManageUserController

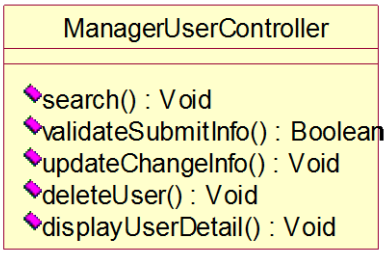


Figure 0.98 Class ManagerUserController

3.5.1.46. TripLog

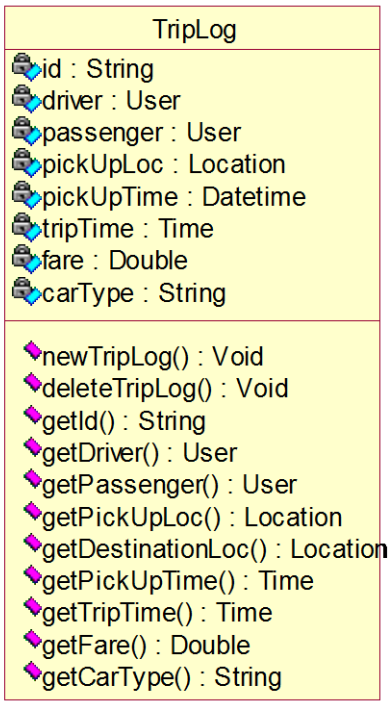


Figure 0.99 Class TripLog

3.5.1.47. DriverLocViewer

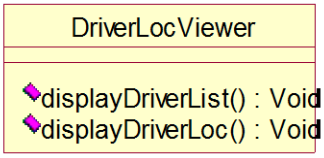


Figure 0.100 Class DriverLocViewer

3.5.1.48. DriverLocController

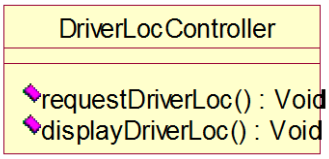


Figure 0.101 Class DriverLocController

3.5.1.49. TripLogForm

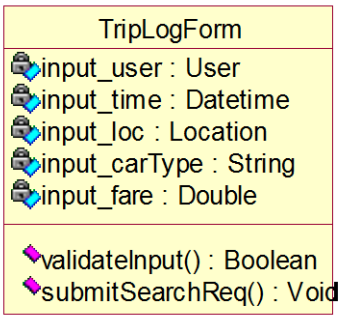


Figure 0.102 Class TripLogForm

3.5.1.50. TripLogViewer

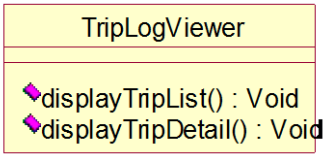


Figure 0.103 Class TripLogViewer

3.5.1.51. TripLogController

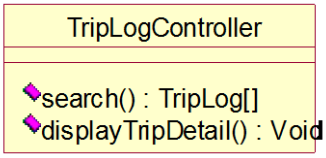


Figure 0.104 Class TripLogController

3.5.1.52. NotiViewer

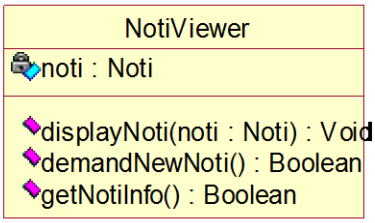


Figure 0.105 Class NotiViewer

3.5.1.53. NotiController

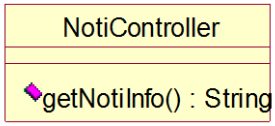


Figure 0.106 Class NotiController

3.5.1.54. UserManageSystem

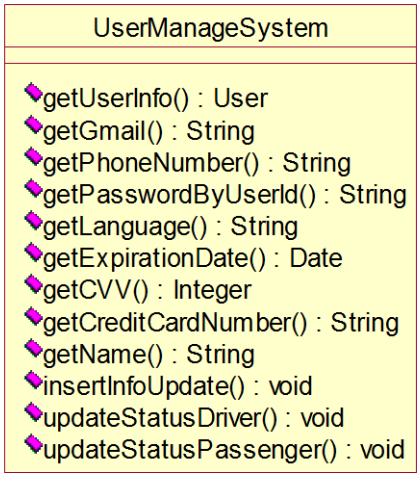


Figure 0.107 Class UserManagerSystem

3.5.1.55. ChangeProfileForm

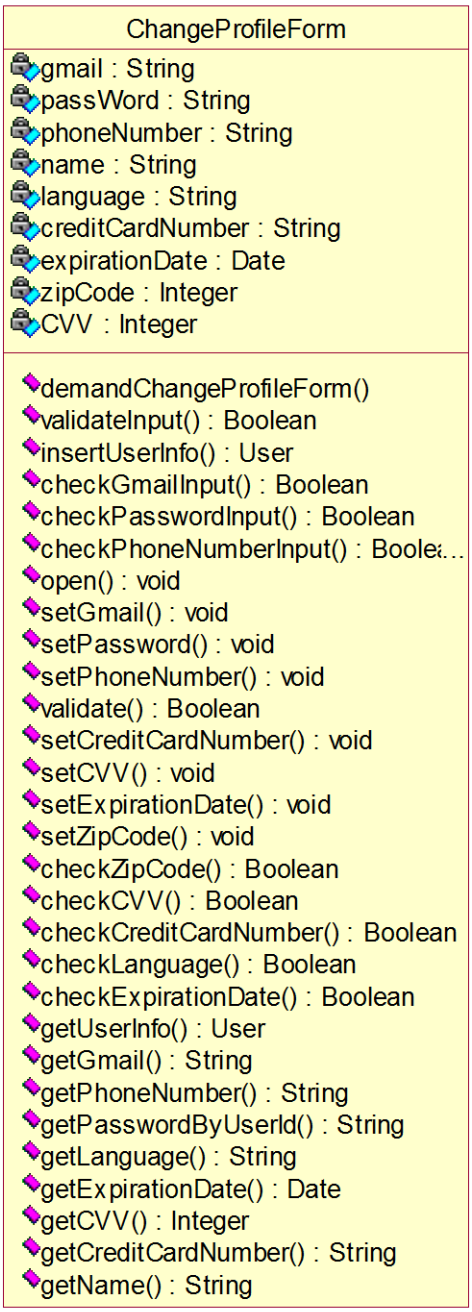


Figure 0.108 Class ChangeProfileForm

3.5.1.56. ChangeProfileController

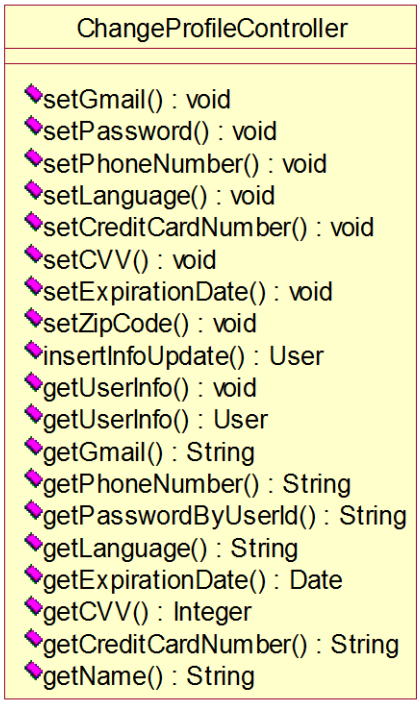


Figure 0.109 Class ChangeProfileController

3.5.1.57. HistoryViewer

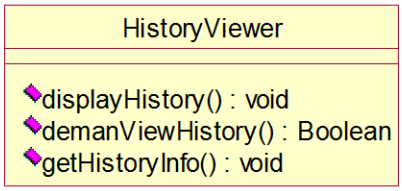


Figure 0.110 Class HistoryViewer

3.5.1.58. HistoryController

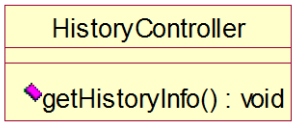


Figure 0.111 Class HistoryController

3.5.1.59. TripManage

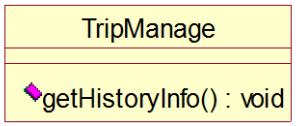


Figure 0.112 Class TripManage

3.5.1.60. AcceptForm

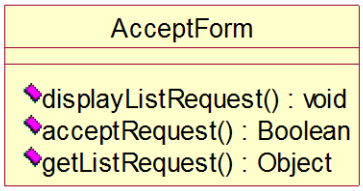


Figure 0.113 Class AcceptForm

3.5.1.61. AcceptController

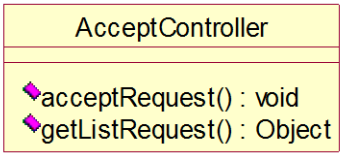


Figure 0.114 Class AcceptController

3.5.1.62. AvailablePassengerViewer

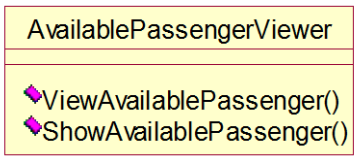


Figure 0.115 Class AvailablePassengerViewer

3.5.1.63. AvailablePassengerController

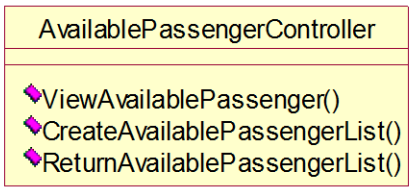
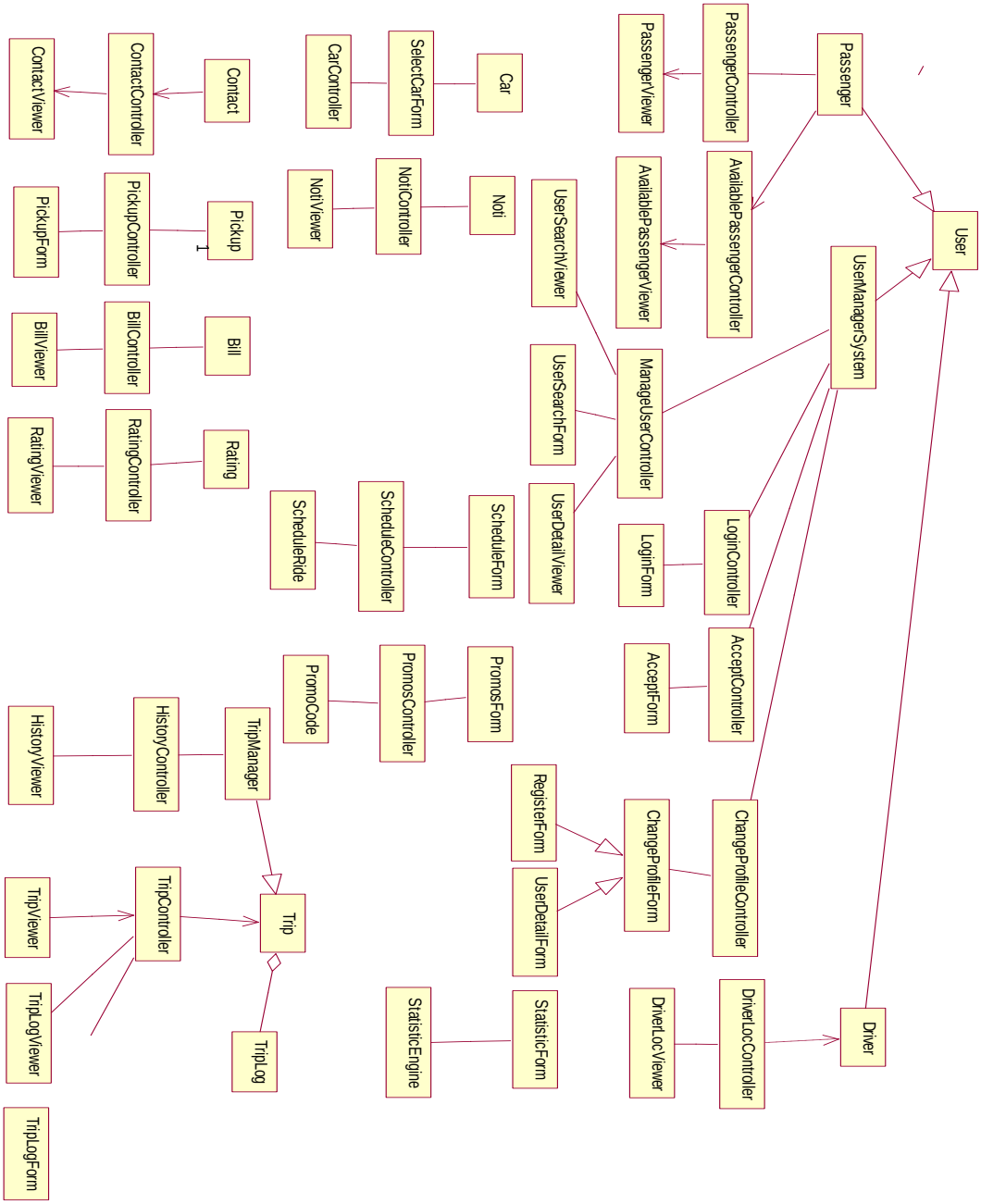


Figure 0.116 Class AvailablePassengerController

3.5.2. Class diagram in total


ContactViewer
ContactController
Contact
CarController
SelectCarForm
Car
PassengerViewer
PassengerController
Passenger
PickupForm
PickupController
Pickup
1
NotiViewer
NotiController
Noti
AvailablePassengerViewer
AvailablePassengerController
User
BillViewer
BillController
Bill
UserSearchViewer
UserSearchForm
UserDetailViewer
ManageUserController
UseManagerSystem
RatingViewer
RatingController
Rating
ScheduleRide
ScheduleController
ScheduleForm
LoginForm
LoginController
AcceptForm
AcceptController
HistoryViewer
HistoryController
TripManager
PromoCode
PromosController
PromosForm
RegisterForm
ChangeProfileForm
ChangeProfileController
TripViewer
TripController
Trip
UseDetailForm
TripLogViewer
TripLog
StatisticEngine
StatisticForm
DriverLocViewer
DriverLocController
Driver
TripLogForm