

Wolkite University

We Strive for Wisdom!

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF INFORMATION TECHNOLOGY

TITLE: TAXI BOOKING SYSTEM

BY:

NAME OF THE STUDENT	IDNO
1) Solomon Wolde	CIR/226/10
2) Tesfaye Abebe	CIR/228/10
3) Musbahadin Wolde	CIR/216/10

Advisor: Mr. ESKINDIR (MSC)

AUGUST 25, 2021

WOLKITE ETHIOPIA

WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF INFORMATION TECHNOLOGY

Title: Taxi Booking System

A Taxi Booking System Submitted to Department of Information Technology in Partial fulfillment of the requirement for the Degree of Bachelor of Science in Department of Information Technology.

BY:

NAME OF THE STUDENT	IDNO
1. Solomon Wolde	CIR/226/10
2. Tesfaye Abebe	CIR/228/10
3. Musbahadin Wolde	CIR/216/10

PROJECT ADVISOR: Mr. ESKINDIR. T(MSC)

Wolkite University, Wolkite, Ethiopia

MAY, 2021

Intellectual Property Declaration

This is to declare that the work under the supervision of Mr. ESKINDIR TEKLU having title "Taxi Booking System" carried out in partial fulfillment of the requirements of Bachelor of Science in 2021 is the sole property of Wolkite University for department of information technology and the respective supervisor and is protected under the intellectual property right laws and conventions. It can only be used for purposes like extension for further enhancement, product development, adoption for commercial/organizational usage, etc., with the permission of the University and respective supervisor. This above statement applies to all students and faculty members.

Date: _____

Name:

Solomon Wolde Signature: _____

Tesfaye Abebe Signature: _____

Musbahadin Wolde Signature: _____

APPROVAL SHEET

This is certified that the project entitled Taxi Booking System will be the proposal is written by:

1. Solomon Wolde
2. Tesfaye Abebe
3. Musbahadin Wolde

A Project Proposal Submitted to the Department of Information Technology of Wolkite University in Partial Fulfillment of Requirements for the Degree of Bachelor of Science in Information Technology

Advisors' Name	Signature	Date
Mr. Eskindir Teklu (MSC)	_____	_____
Department Head Name	Signature	Date
_____	_____	_____
Examiner Name	Signature	Date
1, _____	_____	_____
2, _____	_____	_____
3, _____	_____	_____

Acknowledgment

First and Foremost, we would like to thank creature of the universe, almighty God and Allah for giving us the wisdom and strength to fulfill our duty. Thank you, God!

We would like to thank our Advisor Mr. Eskindir Teklu for his valuable comments, support, and guidance throughout this project work. Without his encouragement and constant guidance, we could not have finished this project.

It is a great pleasure to get an opportunity to thank our families for supporting and teaching us. Thank you, our Families, Finally, we would like to thank our lectures and friends for their encouragement and support throughout this this work.

Contents

CHAPTER ONE 1
1. INTRODUCTION 1
1.1 BACKGROUND OF THE STUDY 1
1.2 STATEMENT OF THE PROBLEM 2
1.3 OBJECTIVES OF THE STUDY 3
1.3.1 GENERAL OBJECTIVES 3
1.3.2 SPECIFIC OBJECTIVES 3
1.4 FEASIBILITY STUDY 4
1.4.1 TECHNICAL FEASIBILITY 4
1.4.2 BEHAVIORAL FEASIBILITY 4
1.4.3 OPERATIONAL FEASIBILITY 4
1.4.4 LEGAL FEASIBILITY 4
1.4.5 ECONOMIC FEASIBILITY (COST BENEFIT ANALYSIS) 4
1.5 SCOPE OF THE STUDY 5
1.6 LIMITATION OF THE STUDY 5
1.7 SIGNIFICANCE OF THE PROJECT 6
1.8 BENEFICIARIES OF THE SYSTEM 6
1.9 SYSTEM DEVELOPMENT AND METHODOLOGY 7
1.9.1 INVESTIGATION (FACT-FINDING) METHODS 7
1.9.2 SYSTEM ANALYSIS AND DESIGN TECHNIQUES 7
1.9.3 OBJECT ORIENTED ANALYSIS (OOA) 7
1.9.4 OBJECT ORIENTED DESIGN (OOD) 7
1.9.5 SYSTEM DEVELOPMENT TOOLS 8
1.10 TESTING METHODOLOGY 8
1.10.1 UNIT TESTING 9
1.10.2 INTEGRATION TESTING 9
1.10.3 SYSTEM TESTING 9
CHAPTER TWO 10
2. DESCRIPTION OF THE EXISTING SYSTEM 10
2.1. INTRODUCTION OF EXISTING SYSTEM 10
2.2 THE EXISTING SYSTEM 10

2.3 COMPARISON BETWEEN THE EXISTING SYSTEM 11
2.4. PASSENGERS OF EXISTING SYSTEM 12
2.5. MAJOR FUNCTIONS OF THE EXISTING SYSTEM 13
2.6. DRAWBACKS OF THE EXISTING SYSTEM 14
2.7. BUSINESS RULES OF THE EXISTING SYSTEM 14
CHAPTER 3 16
3. PROPOSED SYSTEM 16
3.1. FUNCTIONAL REQUIREMENTS 16
3.1.1 PASSENGER APP MODULE 16
3.1.2 DRIVER APP MODULE 16
3.1.3 ADMINISTRATION MODULE 17
3.2. NON-FUNCTIONAL REQUIREMENTS 18
3.2.1. PASSENGER INTERFACE AND HUMAN FACTORS 19
3.2.2. HARDWARE CONSIDERATION 19
3.2.3. SECURITY ISSUES 19
3.2.4. PERFORMANCE CONSIDERATION 19
3.2.5. ERROR HANDLING AND VALIDATION 20
3.2.6. QUALITY ISSUES 20
3.2.7. BACKUP AND RECOVERY 20
3.2.8. PHYSICAL ENVIRONMENT 20
3.2.9. RESOURCE ISSUES 21
3.2.10 DOCUMENTATION 21
CHAPTER FOUR 22
4. SYSTEM ANALYSIS 22
4.1 INTRODUCTION 22
4.2. SYSTEM MODEL 22
4.2.1. USE CASE MODEL 23
4.3. OBJECT MODEL 43
4.3.1. CLASS DIAGRAM 43
4.3.2 DATA DICTIONARY 44
4.3.3 DYNAMIC MODEL 47
4.3.4 SEQUENCE DIAGRAM 47

4.3.5 ACTIVITY DIAGRAM 53
4.3.6 STATE CHART DIAGRAM 59
CHAPTER FIVE 63
5. SYSTEM DESIGN 63
5.1 INTRODUCTION 63
5.2 PURPOSE AND GOALS OF DESIGN 63
5.3 PROPOSED SOFTWARE ARCHITECTURE 64
5.3.1. SUBSYSTEM DECOMPOSITION AND DESCRIPTION 65
5.3.2 DEPLOYMENT DIAGRAM 69
5.3.3. DETAILED CLASS DIAGRAM 70
5.3.4. PERSISTENT DATA MANAGEMENT 70
5.3.5. ACCESS CONTROL AND SECURITY 74
5.4. PACKAGES 75
5.5. PASSENGER INTERFACE DESIGN 75
CHAPTER SIX 79
6. IMPLEMENTATION AND TESTING 79
6.1. Implementation of the Database 79
6.2. Implementation of the Class Diagram for the backend 81
Implementation of the Class Diagram for the user app 87
Implementation of the Class Diagram for the driver app 94
6.3 Configuration of the Application Server 101
6.4. Configuration of Application Security 102
6.5. Implementation of User Interface 103
6.6 Testing 108
6.6.1 Unit testing 108
6.6.2 Integration testing 108
6.6.3 Acceptance testing 108
6.6.4 System testing 109
CHAPTER SEVEN 110
Conclusions and Recommendations 110
7.1 Conclusions 110
7.2 Recommendations 110

References 112

List of figures

Figure 1: Use Case Diagram 25
Figure 2: Class Diagram 44
Figure 3: Sequence Diagram for Log In. 48
Figure 4: Sequence Diagram for Register 49
Figure 5: Sequence Diagram for Book Request 49
Figure 6: Sequence Diagram for Update 50
Figure 7: Sequence Diagram for Cancel Booking 51
Figure 8: Sequence Diagram for Pickup 52
Figure 9: Sequence Diagram for Map 52
Figure 10: Sequence Diagram for Delete 53
Figure 11: Activity Diagram for Login 54
Figure 12: Activity Diagram for Delete 55
Figure 13: Activity Diagram for Book Request 56
Figure 14: Activity Diagram for Update 57
Figure 15: Activity Diagram for Delete 58
Figure 16: Activity Diagram for Cancel 59
Figure 17:State Chart Diagram for Booking 60
Figure 19: Component Diagram for Admin Functionality 66
Figure 20: component diagram for Passenger functionality 67
Figure 21: Component Diagram for Driver Functionality 68
Figure 22: Deployment Diagram 69
Figure 23: Class Diagram 70
Figure 24: Persistence Diagram for Admin 71
Figure 25: Persistence Diagram For Passenger 72
Figure 26: Persistence Diagram for Dispatcher 72
Figure 27: Persistence Diagram for Driver 73
Figure 28: Persistence Diagram for Booking 74
Figure 29: Package Diagram 75
Figure 30: Passenger Homepage 76
Figure 31: Passenger Sign up Page 77
Figure 32: Passenger Sign in Page 77
Figure 33: Admin Sign in Page 78

List of Tables

Table 1: Comparison between the Existing Systems 11
Table 2: Use Case Documentation for Login 25
Table 3: Use Case Documentation for Dispatcher Registration 27
Table 4: Use Case Documentation for View Scheduled Book Registration 28
Table 5: Use Case Documentation For Manage Fleet Registration. 29
Table 6: Use Case Documentation for Tracking Trips Registration 30
Table 7: Use Case Documentation For Manage Request Registration. 31
Table 8: Use Case Documentation for Assign Driver Registration. 32
Table 9: Use Case Documentation for View Request History Registration 34
Table 10: Use Case Documentation for Account Statement 35
Table 11: Use Case Documentation for View Revenue 36
Table 12: Use Case Documentation For Confirm Booking 36
Table 13: Use Case Documentation For Accept Booking 37
Table 14: Use Case Documentation for Pickup and Drop off Location 38
Table 15: Use Case Documentation for Taxi Tracking 38
Table 16: Use Case Documentation for Book Request 39
Table 17: Use Case Documentation for Payment 40
Table 18: Use Case Documentation for Cancel Booking 41
Table 19: Use Case Documentation for Fare Calculation 42
Table 20: Data Dictionary for Admin 45
Table 21: Data Dictionary for Passenger 45
Table 22: Data Dictionary for Driver 46
Table 23: Data Dictionary for Dispatcher 47

LIST OF ABBREVIATIONS

Acronyms	Description
API	Application Program Interface
BR	Business Rule
CPU	Central Processing Unit
FK	Foreign Key
GPS	Global Positioning System
OOA	Object Oriented Analysis
OOD	Object Oriented Analysis
PHP	Hyper Text Pre Processor
PK	Primary Key
RAM	Random Access Memory
SDK	Software Development Kit
UI	Passenger Interface
UML	Unified Modeling Language

ABSTRACT

Now a day, people are more inclined towards owning a smart phone. In such a scenario, mobile application development is one of most sought-after platforms. Android is one of the largest platforms that run in most smart phones from manufacturers like Samsung, iPhone etc. Android based taxi booking system is an Android phone taxi reservation application.

Unfortunately, the passengers are just tired of waiting at different locations for a taxi that they have just reserved to pick them up and take them home. What we need is a reliable mobile application which reserves a taxi for us from a specific place at a certain time and which tells its passengers the status of the taxi in order to keep them from waiting for long hours. Taxi booking System serves just that purpose. This taxi booking system allows the Passengers to get a taxi from any location, even if they seem to be lost. One does not have to spend hours on phone with the taxi services to tell them where exactly they are located currently. This application allows filling pickup place and destination place to the passengers, and knows which taxi would take to reach to the destination. Once a Passenger reserves a taxi, he is checked booking whether it is accepted or denied by the Admin. The application provides the Passenger the facility to view available taxi or status of taxi and status of driver.

CHAPTER ONE

1. INTRODUCTION

In today's world the numbers of mobile Passengers are increasing and a lot of applications, and services generate a lot of data, and information to mobile Passengers. These mobile Passengers are professional people who can be executive, sales people, service engineers, teachers, etc. In general everyone is becoming a mobile service Passenger. These Passengers are using different applications; among the many applications used by many Passengers are productivity applications. Our team member build a productivity application named Taxi Booking system. The main objective of the project is to provide an easy to use and handy mobile application to the android Passengers which enables them to book a taxi from any location to any location. The Passengers can keep track of their bookings and the taxi's current status. The aim of this project is to produce a software system that matches customers and taxi drivers in real time using existing cell phone and the designed system consists of a client mobile application for customers, a mobile application for taxi drivers. The purpose here is to book the taxi any point in time using the Android application. The customer just books the taxi using this application and his details and location is sent to the web site where the software finds taxi near to the customer. The software or web apps controls all the situations at the top and give orders to the taxi drivers. After finding the taxi, software sends the customer details to the nearest taxi driver then the taxi is booked for the given customer.

1.1 BACKGROUND OF THE STUDY

Nowadays, taxi services are everywhere big cities, medium-sized cities and even small cities around the world have reasonable infrastructure to provide this service. Taxi booking systems try to meet service demands of native population and tourists that visit or work on these places. At most of the cities around the world, it is possible to observe that taxi drivers waste a large portion of time at vacant times. Because of this high rate of wasting time, there are a large number of studies that intend to improve the efficiency of these services, without increasing the costs.

It is possible to conclude that the driver is assign by the administrator, since the driver who is nearest to the costumer in a real time location.

This project involves the study of current system in more detail to handle Taxi Booking system. The existing system is a manual system, which means customers are stand long time on the street waiting for coming taxi or make a call for driver they already know. Call phones to taxi drivers or to the taxi service providers and make physical contact to the taxi drivers or taxi service providers. This project needs to make significant change on the existing system. In this project customers can book taxi from anywhere in the town at any time by using their mobile phone. The drivers are only accepting order of system and serving customer. Then the central system make decision based on customer needs, send booked taxi numbers and arrival time with corresponding amount of the cost for the customer and send notification for selected taxi drivers. This range of features ensures booking a taxi online is a lot faster and easier than booking and dispatching a taxi by making call to taxi service providers or to the taxi drivers and physical contact, so it helps to overcome at least the following problems in the existing system.

1.2 STATEMENT OF THE PROBLEM

There are a lot of problems in the existing system. Due to that passengers as well as taxi drivers are wasting their time and money, passengers are stand long time on the street waiting for coming taxi or make a call for driver they already know, so it is difficult to customers to find a taxi on the go. In other way taxi drivers are also waste their time and resource by staying somewhere until the customer call them or come to them or they find customer by roaming . Although as there is no legally specified pick and drop fee for every place based on km or any distance measurement, customers are paying extra money for single route. Road side booking of passengers is inefficient and causes occasional disputes among bookers, passengers and drivers. Therefore taxi booking project will came up with mobile based system to solve those problems mentioned above by enabling the customer to book taxi and drivers to receive booking order from the system anywhere in the town. This would enables that each customer who made a booking is served within the shortest time, thus increases operational efficiency and enhancing customer satisfaction. But it is better to use location

based response for drivers. Therefore drivers will response to customers request by current location. And our system makes payment options simply, customers can pay online while the have access online payment options (systems).

In ride hailing system, the payment mechanism is in hand to hand payment (or the consumer/passenger pays the cost directly to the driver not to directly to the organization). So this payment method has so many negative impacts on the organization, government and consumers/passengers. This payment mode doesn't reduce man power specifications.

On the new updated software, order history doesn't show the date on the order detail. As you all know the app didn't support corporate option, you guys also blocked and increased the tariff of telephone line we are using for corporate booking service. It doesn't support daily corporate Passengers. Most of the time this application has a problem of opening through 3G networks and most of our countries phone are using this network.

1.3 OBJECTIVES OF THE STUDY

1.3.1 GENERAL OBJECTIVES

The aim of this work is to design and develop Taxi Booking apps for Android mobile phone Passengers.

1.3.2 SPECIFIC OBJECTIVES

- To develop procedures for appropriate recorded passengers.
- To develop interactive and Passenger friendly interface.
- To Designing appropriate database for our system.
- To create report, generate system.
- To develop an application that is easy and convenient way in getting a taxi.
- To develop a system that enables the administrator to standardize the customer bookings.

1.4 FEASIBILITY STUDY

1.4.1 TECHNICAL FEASIBILITY

To develop this application, an internet connection is used. The application is deployed and tested on android mobile phone.

1.4.2 BEHAVIORAL FEASIBILITY

This application requires no Passenger guidance or manual as it is easy to use and understand and also has a Passenger friendly interface. The application works in accordance with the design.

1.4.3 OPERATIONAL FEASIBILITY

Our new system is operationally feasible and Operational Feasibility deals with the degree to which the proposed application solves business problems. The proposed application solve the problems in the following ways.

- By providing a new and reliable means to handling booking taxi.
- By providing current location and appropriate confirmation for booking.
- Better management of taxi.

1.4.4 LEGAL FEASIBILITY

Legal feasibility determines whether the proposed system conflicts with the legal requirement or not. A project may face legal issues after completion if this factor is not considered at the first stage.

As a result our system is legal and doesn't conflict with the law of the country; it works based on the rule and regulation of the Ethiopian constitution.

1.4.5 ECONOMIC FEASIBILITY (COST BENEFIT ANALYSIS)

The application is economically feasible as it only requires an android device with Android SDK 2.3 or higher and Passengers should be able to connect to the internet either through cellular or Wi-Fi and should able to receive messages. This would be the only cost incurred on the project and this project is economically feasible; because the system is developed in low cost.

1.5 SCOPE OF THE STUDY

- **Smart Geo-location**

Once the service seeker accesses the app, it tracks the real-time location and shows the available taxis nearby. After tapping "Book-now," available drivers (nearby) will be notified and the passenger's pickup location is confirmed (once the ride request has been accepted).

- **In-app communication (Inter-app communication b/n driver app and passenger app)**

Driver and passenger/rider app communicate with each other during the following operational scenarios:

- ✓ Live location (of driver) sent to the passengers' app
- ✓ Booking requests to drivers
- ✓ Booking cancellation (Driver and passenger app)
- ✓ Pick-up location (of passengers) to the driver app
- ✓ Drop-off location (of passengers) to the driver app

- **Administration (Web App)**

Web application is used for performing administration and dispatching operations. Web App controls and manages the whole process between taxi drivers and passengers. It is the administration staff.

1.6 LIMITATION OF THE STUDY

Just as any researcher is bonded and restricted to encounter some limitation, this project is faced with the following limitations. This project is limited to contract taxi service Centre and not accommodates for higher officials and is also limited to transport within the town. Therefore, passengers travelling are not possible outside the town and drivers will not be accommodated. We faced some limitations to do this project those are appropriate internet access, money for paying to buy some Google functionalities and shortage of time to fulfill our project.

1.7 SIGNIFICANCE OF THE PROJECT

The significance of the study is to create taxi booking system that will provide services to customers and easy to use. Some of the significant of the proposed system described as follow:

Taxi Booking Application is the ideal taxi booking application for individuals on the go. This Passenger-friendly application offers convenience by allowing Passengers to pre-set their favorite locations and journeys for their taxi bookings. It even allows Passengers to book a taxi at their current location. For ease of convenience, Passengers can access and book from a list of completed trips.

Instead of dialing to the service center, taxi booking system helps Passengers deliver their requests directly to the nearest taxi drivers through internet. It greatly saves the trouble calling to service center and waiting reply from it.

- Reduce waiting time.
- Easily used.
- Customers can send pickup, drop off place and arrival time to the drivers
- It prevents to the Passenger from fear, stress and confusion.

1.8 BENEFICIARIES OF THE SYSTEM

Different parts are going to be benefited from this system such as:

- **Drivers** benefited from the proposed system by tracking the pickup and destination place of the passenger and decrease his tired of finding passengers on the road.
    - Drivers who use the system, they perform their work effectively and efficiently without loss time and resource.
- **Customers:** It totally remove wastage of customers time who are stand long time on the street waiting for coming taxi, gains fast access and save their time.
- **And taxi service center** can manage properly all taxi those are inside the service provider.

- **Government:** The development of this project includes sales report of drivers which is information source for the government and government can collect fair tax.

1.9 SYSTEM DEVELOPMENT AND METHODOLOGY

In order to achieve our aim, we use different methods to bring the system from imagination to realization. These methods include different models, techniques and tools for our work.

1.9.1 INVESTIGATION (FACT-FINDING) METHODS

To get a precise data from the society the team has used the following fact finding techniques.

Those are: -

For the collection of data: Observation, Interview and Document analysis was used.

❖ **Observation**:-Observation is common methods of scientific research to collect the data. We used observation to know how the existing system work, to know exactly how Passengers, Drivers and Administrator are handling the work in the system.

❖ **Document Analysis: -** Document analysis is used to understand how the system is working. We used this method to know all about the Passengers, Drivers, Administrators and overall of their work in short and brief.

1.9.2 SYSTEM ANALYSIS AND DESIGN TECHNIQUES

In this project, we had used object oriented system and Analysis and design development methodology (OOSAD) and it has two phases.

1.9.3 OBJECT ORIENTED ANALYSIS (OOA)

During this phase, we used to model the function of the system (use case modeling), find and identify the business objects, organize the objects and identify the relationship between them and finally model the behavior of the objects.

1.9.4 OBJECT ORIENTED DESIGN (OOD)

During this phase, the model interactions and behaviors that support the use case scenario, and finally update object model to reflect the implementation environment. And also transforms the conceptual model produced in object-oriented analysis to take account of the

constraints imposed to our system format, so that we will use this phase to refine the use case model to reflect the implementation environment.

1.9.5 SYSTEM DEVELOPMENT TOOLS

For this project system development, we uses JAVA, PHP for the front end as means of communication between system & the Passenger. This enables the system to be friendly interactive.

Programming Language Tools

- **Wamp Server or XAMPP**
- **Hardware Tools:** - pc (personal computer) including its internal part and external part (i.e. system, keyboard, mouse, monitor, storage parties and etc.).

  - ✓ Highest processor speed and latest CPU.
  - ✓ 4GB RAM.
  - ✓ Hard disc 700MB.
  - ✓ Flash Disk

Software Tools: - software is the main requirement to develop the project. Software we are going to use in our project are; programming language such as JAVA, JAVASCRIPT and PHP since the project is Android based.

- ✓ Microsoft office word
- ✓ XAMPP
- ✓ EdrawMax
- ✓ Notepad++ (editor)
- ✓ Android Studio

1.10 TESTING METHODOLOGY

Testing is the final phase of our implementation. Testing is checking of the system workability in an attempt to discover errors and avoiding such errors from the system. In this the team members tested the entire system as a whole with all forms, code, modules. In this we tested all the functionalities in the System. All errors in the forms, functions, will be

tested. To simplify the testing process the project team followed the different types of tests that break the testing process up into the distinct levels. These types testing are unit testing, integration testing and system testing.

1.10.1 UNIT TESTING

Unit testing is every module of the System is separately tested. It is often done by the programmer to test that the unit he/she has implemented is producing expected output against given input. Functional Test Specifications: that will be done in our proposed system the tests done as follow:

1.10.2 INTEGRATION TESTING

Integration testing is the activity of software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before acceptance testing. Thus, Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested.

1.10.3 SYSTEM TESTING

It is the final step of testing. In this the team members tests the entire system as a whole with all forms, code, modules. This form of testing is popularly known as Black Box testing or System tests. In this the team members tests all the functionalities in the System.

CHAPTER TWO

2. DESCRIPTION OF THE EXISTING SYSTEM

2.1. INTRODUCTION OF EXISTING SYSTEM

Existing system is totally on book and thus a great amount of manual work has to be done. In existing system, there are various problems like customer/taxi finding, prices of services and fixing bill generation on each bill and also finding out details regarding any information is very difficult. Major problem was lack of security because costumers don't have any guarantee about the driver's identity. With this existing system low priority in terms of exchanging of data within short period of time. And usually there is no published fares and the customers will end up paying what the Driver demand for the service amount and also there is no standard system that will number the distance in the sense that customers are not assured of the distance preference.

2.2 THE EXISTING SYSTEM

I. RIDE Application are made in Addis Ababa, withstanding strict Ethiopian directions to guarantee that their customers have the best quality apparatus in the market. RIDE is Transit Application. The application gives a stage to different stakeholder's in a substantial endeavor to work together and share data for enhanced administration and administration ability. Through cutting edge APIs, the framework is fit for interfacing with outer frameworks like planning framework, resource control framework, and corporate money related/stock framework.

The key elements of RIDE platform is:-

- ✓ Location tracking see the area which the driver and the customer located.
- ✓ Assign driver to customers through nearest location of drivers for customers.
- ✓ Deal with different provider to give services for customers.
- ✓ Multi-gadget support from PC to cell phones, to tablets.
- ✓ Capacity to refresh vehicle data and transfer photographs of the vehicle for documentation reason and care.

- ✓ Capacity to audit travelled course for now, yesterday, this week, and the whole month.
- ✓ See the vehicle type, vehicle data and driver data of the vehicle.
- ✓ Fare calculator for payment options [1].

II. FERES MILES: As one of the prestigious selection of GPS following framework supplier in Ethiopia. FERES offers Driver GPS tracker, vehicle GPS tracker, and costumer GPS tracker. It has an opportunity to make taxi booking sufficiently. It can handle and gives an appropriate ordering and servicing for appropriate costumer requests.

FERES has different functions to perform:

- ✓ Location tracking for services.
- ✓ Driver and customer registration for security issues.
- ✓ Capacity to refresh customer, driver and vehicle data/information for transfer to documentation in future use.
- ✓ Capacity to audit daily, monthly and yearly usages.
- ✓ Capacity for price calculation in the payment system through appropriate distance traveled [2].

2.3 COMPARISON BETWEEN THE EXISTING SYSTEM

Table 1: **Comparison between the Existing Systems**

Features	RIDE	FERES MILES
Payment option	No	No
GPS	Yes	Yes
Book a vehicle feature	Yes	Yes

Real time tracking	Yes	Yes
Help Option	No	Yes
Platform	Yes	Yes
Route history	No	Yes

2.4. PASSENGERS OF EXISTING SYSTEM

- ✓ **Passengers/customers:** Those are consumers who has to be serviced by an organization for their transportation system. These consumers can communicate with the organization through a passenger app module in their mobile phone.
  There is some functionalities of these consumers such as:-
- ✓ Register/log-in
- ✓ Booking requests to Admins
- ✓ Pick-up location (of passengers) to the driver app
- ✓ Drop-off location (of passengers) to the driver app
- ✓ Price calculator
- ✓ Tracking taxi
- ✓ Interactive map
- ✓ **Driver:** These are peoples who has a job in the organization to drive taxis to give services for consumers. And they have accessed the organization and the consumers through a driver app module on their android phone. The have some functionalities such as:-
  - ✓ Registration
  - ✓ Pick-up & drop-off navigation

- ✓ **Admin:** Those are the organizations administrator or manager who controls the whole functionalities inside the organization. And it manages and controls the drivers, cars, cost estimation, and so on inside the organization.
    - ✓ Driver management
    - ✓ Vehicle management
    - ✓ Payment management
    - ✓ Earnings report
    - ✓ Tracking trips
    - ✓ Cost estimation
    - ✓ Receipt
    - ✓ Notification when starting the travel
    - ✓ Manage service requests
    - ✓ Matching drivers & passengers

2.5. MAJOR FUNCTIONS OF THE EXISTING SYSTEM

- **For the Transportation Service**

Upon service request and using GPS technology, customer and the third party service provider will have real time communication and established connection to detect location information to the service provider.
If the Service Provider accepts your request, the Application program notifies you and provides information regarding the Service Provider - including driver's name, vehicle type and vehicle plate number, and the ability to contact the Service Provider by telephone.

The Application also allows you to view the Service Provider's progress towards the pick-up point, in real time.

- **For Better Security**

This application gives a better security for passengers and drivers. It protects everyone's privacy with respect to login systems. Although these applications are legally registered, so the passenger's didn't afraid of any danger or robbery.

2.6. DRAWBACKS OF THE EXISTING SYSTEM

As the fast growing of technologies in the country there are a lot of new software's and applications are developed rapidly.

In ride hailing system, the payment mechanism is in hand to hand payment (or the consumer/passenger pays the cost directly to the driver not to directly to the organization). So this payment method has so many negative impacts on the organization, government and consumers/passengers. This payment mode doesn't reduced man power specifications.

On the new updated software, order history doesn't show the date on the order detail. As you all know the app didn't support corporate option, you guys also blocked and increased the tariff of telephone line we are using for corporate booking service. It doesn't support daily corporate Passengers. It doesn't support all android devices which doesn't access play store like Huawei android phones. Most of the time this application has a problem of opening through 3G networks and most of our countries phone are using this network.

2.7. BUSINESS RULES OF THE EXISTING SYSTEM

Business rule is effectively an operating principle or policies that we try to specify for both the existing system and the new system must satisfy. The business rule is principle or a policy in which the proposed system operates accordingly. There are some business rules and constraint to prevent any violation during the process. This focuses on access control issue.

BR01: Passengers should have Passenger name and password to make booking.

BR02**:** Passengers should fill appropriate personal information (Name, address, starting location, destination location, starting time, arrival time) for getting services.

BR03: Drivers should have a valid driving license to be registered in to the system.

BR04: Drivers fill appropriate personal information (Name, Address, locations, starting time and arrival time for salary and security.

BR05: Cost estimation is calculated through the distance traveled by the passengers and located on the passenger's app and administrators. In this system the initial cost estimation is 40 birr. And the cost is calculated through one birr per one kilometer.

BR06: Location tracking is done by all the Passengers in the system to give and got services efficiently.

CHAPTER 3

3. PROPOSED SYSTEM

3.1. FUNCTIONAL REQUIREMENTS

3.1.1 PASSENGER APP MODULE

- ✓ **Register/log-in:** All the passengers who use this system should register for the first time. And can login through using Passenger name and password for security purposes and privacy.
- ✓ **Booking requests:** passengers requests booking to get services through login in the system and by locating their location to book a service.
- ✓ **Pick-up location (of passengers) to the driver app:** By turning on their GPS passengers can locate their pick up location.
- ✓ **Drop-off location (of passengers) to the driver app:** By locating the destination area they can locate their drop off location.
- ✓ **Fare calculator:** This app module can show the price for passengers to identify how much to pay for the services.
- ✓ **Tracking taxi:** This module can track the taxi status and the driver status.
- ✓ **Interactive map:** The passenger's app module have a Google map which can control and look up for the starting location to the destination location.
- ✓ **Tracking Taxi:** Tracks real time locations and shows the available taxi's nearby.
- ✓ **Payment:** After getting services, the passenger makes payment through the app.

3.1.2 DRIVER APP MODULE

- ✓ **Registration**: when the driver first comes to the system it should be registered all of his status to identify and awareness of the driver to the system.
- ✓ **Pick-up & drop-off navigation:** In the driver app module the driver uses a Google map to locate his pickup location. And also his drop off location.
- ✓ **Accept requests:** accepts Passenger requests and by tracking their real time location provides service to Passengers.

3.1.3 ADMINISTRATION MODULE

- ✓ **Driver management:** This module controls and manages driver's status like pickup and drop-off location, driving licenses, registering new drivers and so on.
- ✓ **Vehicle management:** It controls and manages the vehicles (taxi) like which taxi is given to which driver, Targa number, registering new taxi and so on.
- ✓ **Payment management:** It directly controls the payment mechanism which the passenger's paid successfully or not.
- ✓ **Earnings report:** This module sends a report to passengers when they paid and gives a receipt.
- ✓ **Tracking trips:** In this module the administrators manage the passengers and location and assigning which driver to which passenger by their pickup location. And the driver is assigned to a passenger through its nearby location.
- ✓ **Cost estimation:** In here the price should be calculated through the distance travelling by the customers.
- ✓ **Manage service requests:** When passengers sent booking requests this module manages the requests to assign and give a response to passengers with short period of time.
- ✓ **Matching drivers & passengers:** Here drivers and passengers are introduced each other get services.

3.1.3.1 DISPATCHER

- ✓ **Register/Login:** These are peoples who are employed in the organization and registered by the administrator.
- ✓ **Tracking trips:** track the location of passengers and drivers, when there has to be a service request from customers.
- ✓ **Manage service requests:** control and manage service requests which have to be delivered.
- ✓ **Matching drivers and Passengers:** Assigning drivers to customers through their distance and nearby by location.

3.1.3.2 ACCOUNT MANAGEMENT

- ✓ **Register/Login:** These are peoples who are employed in the organization and registered by the administrator.
- ✓ **Provider statement**: which modulates and consumes the provident account statement that when the provider's daily, monthly and yearly statements.
- ✓ **Account creation and approval:** When new customers and provider/driver are registered to the system and wants to create an account and approved.
- ✓ **Withdrawal request and approval:** when there is a need to budget in the organization to do some works and this manager approves in it.

3.1.3.3 FLEET MANAGEMENT

- ✓ **Register/ Login:** These are peoples who are employed in the organization and registered by the administrator.
- ✓ **Account control**: managing and controlling accounts with in the organization.
- ✓ **View request history:** checking and updating histories that are requested by the customers and the provider's/driver.
- ✓ **View scheduled ride:** checking and managing ride requests comes from customers and transfer requests to dispatcher.

3.2. NON-FUNCTIONAL REQUIREMENTS

- ✓ The load time for customer interface shall A few seconds.
- ✓ The login information shall be verified within few seconds.
- ✓ Response time of the system will not take long time.
- ✓ The system is work 24 hours per day seven days in a week which means the system is always work.

- ➢ Efficiency: - The system will reduce resource consumption for given both customers and drivers. It reduce time consumption for customers and drivers.
- ➢ Usability: - The system can be easily used
- ➢ Passenger Interface: - The android device format and the forms prepared for the information are easy and Passenger friendly to the customer and can be easily

understand. The system shall be design according to standards and automate existing system.

3.2.1. PASSENGER INTERFACE AND HUMAN FACTORS

Our systems applies a simple and easy human friendly interfaces. Anyone can access the system, which doesn't have any complex structure that if someone can access a mobile phone easily, can use it. This system provides security and privacy to keep Passengers free from fear and stress or ambiguity on the system.

3.2.2. HARDWARE CONSIDERATION

This system has integrated with any android devices which API version of 16 and above (Android 4.0 and above devices). And the system works with lower internet connection and devices which can support Google maps (GPS).

3.2.3. SECURITY ISSUES

Our system provides security option through login forms to authenticate the Passengers of our system to make them feel secured. And privacy is the main target for this project our system keeps Passengers secret with authorization and authentication system with login forms.

3.2.4. PERFORMANCE CONSIDERATION

- **Response Time:** when accessing the system to get services it is fast to respond and take actions for Passengers requests. When passengers booking request to the system the administrators look their pickup location and assign a driver and reply a notification their request is accepted.
- **Supplies:** It can support many Passengers at given time until all the drivers are taken by other customers.
- It can load infinite number of requests because the system is available all the time.

3.2.5. ERROR HANDLING AND VALIDATION

- The system display message if the Passenger enters incorrect input into different fields.
- The system display message if the customer enters large quantity that is greater than the existing product in store.
- The passenger cannot book if taxi is not available.
- The system display message if the Passenger tries to open the privilege page using wrong password and wrong Passenger name.

3.2.6. QUALITY ISSUES

- **Reliability:** This system has developed through different programing languages and works effectively and efficiently to perform the given services to Passengers as the customer's needs.
- **Availability:** The system should always be available for access at 24 hours, 7 days a week.
- This system is an automated system, so it has many advantages to Passengers by consuming wastage of time, ease of access, availability and so on. Passengers are highly benefit on the system.

3.2.7. BACKUP AND RECOVERY

Backup and recovery is the necessary part of the system and our system have its own database server to store and it has an online maintenance to fix simple failures of the system. Backup is taken when new features or functionalities added to the existing system.

3.2.8. PHYSICAL ENVIRONMENT

This system is deployed or addressed to a particular organization which has its business conduct. And that company can deploy the different module of the system to Passengers and other aspects of the organization. These modules can distribute to Passengers on different websites.

3.2.9. RESOURCE ISSUES

The system can provide available resource for the community as well as the members of the firm on specified time as needed. Some limitations that should be taken into consideration are related to security and reliability of the system. The system should be designed in such a way that it's impossible for a third party to see or change data in any way. Such an intrusion could allow a third party to act as a taxi driver, and could cause significant loss for the taxi organization. Furthermore, the system should be available at all times, and resistant to database, network and other failures. Such an error could cause a loss of communication between taxi drivers and dispatchers, and should be treated accordingly.

3.2.10 DOCUMENTATION

The new system required full documentation, help contents and tips to allow further maintainability and to support and guide Passengers how to use the new systems.

CHAPTER FOUR

4. SYSTEM ANALYSIS

4.1 INTRODUCTION

After careful observing and analyzing the problem of this type mobile application our team decide to design and develop the android based Taxi Booking system as a solution. The new mobile application is a faster, accurate and a bit more personal application, specially designed to make your taxi reservation application is better and also provide a sustainable way to get taxi easily when you want to go everywhere in the town.

The proposed application will use the major functionality of the prior related work application to advance the prior related work .This means that taxi booking provides information to the Passenger about the current location they exist and track the nearest taxi based on transportation mood.

4.2. SYSTEM MODEL

System Model is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. It is about representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML). Models help the analyst to understand the functionality of the system; they are used to communicate with customers.

The project development in a team used a UML (unified model language) to specify the work of the proposed system diagrammatically through different angles. The UML (unified model language) development approach gives easier and simple, small and manageable components so that it reduces the vague appearance of the big problem. The system model document describes all data, functional and behavioral requirements of the software under production or development.

Models can explain the system from different perspectives: An external perspective, where we model the context or environment of the system, An interaction perspective, where we

model the interactions between a system and its environment, or between the components of a system, A structural perspective, where we model the organization of a system or the structure of the data that is processed by the system, A behavioral perspective, where we model the dynamic behavior of the system and how it responds to events.

Five types of UML diagrams that are the most useful for system modeling: Activity diagrams; which show the activities involved in a process or in data processing, Use case diagrams; which shows the interactions between a system and its environment, Sequence diagrams; which show interactions between actors and the system and between system components, Class diagrams, which shows the object classes in the system and the associations between these classes, State diagrams, which shows how the system reacts to internal and external events.

The following lists of objects are System Model for Taxi Booking System.

- ⇨ Use Case Diagram
- ⇨ Use Case diagram list and their details description
- ⇨ Description of actors
- ⇨ Activity diagram
- ⇨ Class diagram
- ⇨ Sequence diagram
- ⇨ Passenger Interface prototyping diagram

4.2.1. USE CASE MODEL

This Use case Model shows that Administrator, Dispatcher, Fleet Manager, Account Manager, Driver and Passengers are interact with the system. And all those actors have to make their own role to the system with respect to different classes of objects.

Use Cases

- ❖ Login
- ❖ Passenger/Customer registration
- ❖ Make payment
- ❖ Send Request

- Cancel Booking
- Source/Destination
- Fare Estimation
- Accept Booking
- Confirm Booking
- Account Statement
- Account Settlement
- Manage Driver
- Register Taxi
- View Scheduled Ride
- View Request History
- Assign Driver
- Vehicle Management
- GPS tracking
- Manage Account Manager
- Manage Fleet Manager
- Manage Dispatcher
- Register Taxi

4.2.1.1. USE CASE DIAGRAM

Use case diagram is a graphical representation of use case model of our project. In our system there are five actors and twenty three attributes which satisfies and full fills all the system requirements. Each actor has their attributes to identify the functionalities of each actor on the system [3].



Figure 1: Use Case Diagram

4.2.1.2. USE CASE DESCRIPTION

Table 2: Use Case Documentation for Login

Section	Purpose
Use case number	UI01
Name	Log in
Actor	admin, driver, Passenger, dispatcher

Description	This use case describes how admin, driver, Passenger and dispatcher to login into the System.	
Goal	To login	
Precondition	The actor is on the homepage	
Post condition	If the use case was successful, the actor is now logged into the system. If not, the system state is unchanged.	
Basic course of action	Passenger Action	System Response
	1 The admin, Passenger, dispatcher and driver is on the home page to login to the system. 3. The admin, Passenger, dispatcher and driver enters Passengername and password, Click on Login Button.	1. The system promotes the admin, Passenger, dispatcher, and drivers to enter Passenger name, Password. 4. The system verifies that all the filled have been filled out and valid. 5. The system successfully logged in the system. 6 Use case Exit

Alternate course of action	6.1 If all fields are not filled out and not matched to the Passenger name and password the system notifies the actor a message Verify Passenger name or Password and then goes back or returns to step 4 of basic course of Action to enter again.

Table 3: Use Case Documentation for Dispatcher Registration

Section	Purpose	
Use case no	UI02	
Name	Register Dispatcher	
Actor	Admin	
Description	Allow admin to register Dispatcher	
Goal	To register a Dispatcher	
Precondition	Login	
Post condition	Registered	
Basic course of action	Passenger Action	System report
	1 The administrator wants to register Dispatcher 3 The administrator open Registered link and enter or fill necessary information about Dispatcher.	2 The system allows admin to enter Dispatcher's information. 4 The system verifies that all the filled have been filled out and valid. 5 The system notifies Dispatchers that registration

		is successful 6 Use case end
Alternate course of action	A.5 the system indicates the Passengers that he/she enters invalid information or misses something. A.6 the use case continues step3 and the basic course of action ends.	

Table 4: Use Case Documentation for View Scheduled Book Registration

Section	Purpose
Use case no	UI05
Name	View Scheduled Book
Actor	Admin, dispatcher
Description	An admin and dispatcher can see and control Scheduled Book
Goal	To monitor and control Scheduled Book
Precondition	Admin and dispatcher should be logged in.
Post condition	check and manage the scheduled rides

Basic course of action	1. The admin and dispatcher wants to check View Scheduled Book. 2. The system allows admin and dispatcher to enter Passengername and password information. 3. The admin and dispatcher open Registered link about View Scheduled Book. 4 The system responses all the scheduled book and propagates it. 5 Use case end
Alternate course of action	A.5 the system indicates the Passengers that he/she enters invalid information or misses something A.6 the use case continues step1 and the basic course of action ends.

Table 5: Use Case Documentation For Manage Fleet Registration.

Section	Purpose
Use case no	UI06
Name	manage fleet
Actor	Admin, Dispatcher
Description	Allow admin and Dispatcher control and manage over the fleet.
Goal	To manage and control fleet

Precondition	Admin and Dispatcher should login, and fleet manager can give permission to access it.
Post condition	Get full permissions on the fleet to manage and control.
Basic course of action	1 The admin and Dispatcher wants to manage fleet. 2 The system allows admin and Dispatcher to manage fleet's information when they are logged on. 4 The system shows all fleet information to admin and dispatcher. 5 The system notifies that the admin and dispatcher manage and control fleet is successful 6 Use case end
Alternate course of action	A.5 the use case continues step2 , 3,4,5 and use case ends

Table 6: Use Case Documentation for Tracking Trips Registration.

Section	Purpose
Use case no	UI07
Name	tracking trips
Actor	Dispatcher, Driver, passenger
Description	Dispatcher, Driver and Passenger can check and look after tracking trips of drivers and passengers.

Goal	To control and look after the way that the driver and passenger has assigned.	
Precondition	The Driver, Dispatcher and passenger should Login, GPS	
Post condition	Management on the location tracking	
Basic course of action	Passenger Action	System report
	1 The Dispatcher, Driver and passenger wants to look and manage tracking trips 3 Dispatcher, Driver and passenger soul be logged on the system.	2 The system allows Dispatcher, Driver and passenger see and manage tracking trips. 4 The system verifies and shows all tracking trips information. 5 The system notifies tracking trips when logged in is successful 6 Use case end
Alternate course of action	A.5 the system indicates the Passengers that he/she enters invalid information. A.6 the use case continues step3, 4, 5 and use case end.	

Table 7: Use Case Documentation For Manage Request Registration.

Section	Purpose
Use case no	UI08
Name	manage request

Actor	Dispatcher	
Description	Allowing Dispatcher to manage request and give responses for the request history.	
Goal	To manage request	
Precondition	The dispatcher and Passenger should login, passenger should make request and dispatcher should pass request history.	
Post condition	Show quest history and managing all the requests.	
Basic course of action	Passenger Action	System report
	1 The Dispatcher wants to manage request. 3 The dispatcher should logged in first and should pass all information to dispatcher.	2 The system allows Dispatcher to manage request information. 4 The system verifies that all passengers request are passed to dispatcher. 5 The system notifies the Dispatcher can manage request is successful 6 Use case end
Alternate course of action	A.5 the use case continues step2, 3, 4, 5 and use case ends.	

Table 8: Use Case Documentation for Assign Driver Registration.

Section	Purpose

Use case no	UI09	
Name	Assign Driver	
Actor	Dispatcher	
Description	Dispatcher should Assign Driver to give services for passenger's request.	
Goal	To register Assign Driver	
Precondition	Login, booking request, tracking trips	
Post condition	Assign driver to each request by their nearest location.	
Basic course of action	Passenger Action	System report
	1 The Dispatcher wants to Assign Driver to each request respectively. 3 The Dispatcher can control over the driver information's and can assign driver to passengers based on the nearest location.	2 The system shows all driver information's to Dispatcher. 4 The system verifies that the driver is Assigned to the passengers correctly. 5 The system notifies that the Assign Driver is successful 6 Use case end
Alternate course of action	A.5 The system indicates when invalid address get caught, A.6 the use case continues step3 and the basic course of action ends.	

Table 9: Use Case Documentation for View Request History Registration

Section	Purpose	
Use case no	UI10	
Name	View Request History	
Actor	Dispatcher	
Description	Allow Fleet managers to View Request History	
Goal	To View Request History	
Precondition	Login, Tracking trips, book request	
Post condition	Registered	
Basic course of action	Passenger Action	System report
	1 The dispatcher wants to view and see all request history. 3 dispatcher check request and pass to the dispatcher.	2 The system shows all the Passengers request to fleet manager 4 The system verifies that all the request are comes from passengers are stored or not. 5 The system notifies that all requests are appropriately serviced or stored is successful. 6 Use case end
Alternate course of action	A.5 the system indicates the Passengers make wrong booking	

	A.6 the use case continues step3, 4, 5 and use case ends.

Table 10: Use Case Documentation for Account Statement

Section	Purpose
Use case number	UI11
Name	Account statement
Actor	Admin
Goal	Manage Account
Description	This use case describes account statement is a periodic summary of account activity with a beginning date and an ending date
Precondition	UI01
Post condition	View daily, monthly ,yearly, provider, overall ride statement
Basic course of action	1 the Admin wants to view details about Account statement 2 And by clicking "provider statement" button view driver statement 3 By clicking "daily statement" button view today Statement 4 By clicking "monthly statement " button view this month statement 5 By clicking "yearly statement" button view this year's statement
Alternate course of action	A6 the use case continues step 3, 4, 5 and use case ends.

Table 11: Use Case Documentation for View Revenue

Section	Purpose	
Use case number	UI14	
Name	View revenue	
Actor	Admin	
Goal	View revenue	
Description	This use case can it the Admin can view overall revenue	
Precondition	UI01	
Post condition	Know about overall revenue	
Basic course of action	Passenger action	System response
	1 the Admin wants to view revenue 2 the Admin can see the overall revenue	3 the system can provide overall revenue 4 use case ends
Alternate course of action		

Table 12: Use Case Documentation For Confirm Booking

Section	Purpose
Use case number	UI 15
Name	confirm booking
Actor	Drivers

Description	Allows the drivers to reply passengers request
Goal	To send confirmation
Pre-condition	Receive booking
Post condition	Passenger receive confirmation
Basic course of action	1 the drivers want to send confirmations to the customers 2 the system allows drivers to reply
Alternate course of action	3 The system notifies when the drivers send error confirmation

Table 13: Use Case Documentation For Accept Booking

Section	Purpose
Use case number	UI 16
Name	Accept booking
Actor	Drivers
Description	Allows the drivers to receive booking
Goal	To accept booking
Precondition	UI01
Post condition	Send confirmation
Basic course of action	1 the drivers want to receive booking 2 the system allows drivers to receive booking 3 the drivers can receive booking and can send confirmation to the Passengers

Alternate course of action	4 if the driver cannot receive booking, the system notifies booking is rejected.

Table 14: Use Case Documentation for Pickup and Drop off Location

Section	Purpose
Use case number	UI 17
Name	pickup and drop off location
Actor	Drivers and Passenger
Description	Passenger send pickup and drop off location and driver view pickup and drop off location of Passengers
Goal	To meet driver and Passengers
Precondition	UI01
Post condition	The driver gets pickup and drop off location of Passenger
Basic course of action	1 the drivers want to view Passenger's pickup and drop off location 2 the system allows drivers to receive booking 3 the drivers can receive booking and can send confirmation to the passengers
Alternate course of action	4 if the driver cannot receive booking, the system notifies booking is rejected.

Table 15: Use Case Documentation for Taxi Tracking

Section	Purpose

Use case number	UI 18
Name	taxi tracking
Actor	Passenger
Description	Provides maps with taxi and the Passenger easily view and choose the nearest taxi
Goal	To view and select the nearest taxi
Precondition	UI01
Post condition	The Passenger selects the nearest taxi
Basic course of action	1 Passenger track taxi and can view the nearest taxi 2 Passengers can select the nearest taxi 4 Passengers can view driver details that the selected taxi
Alternate course of action	.

Table 16: Use Case Documentation for Book Request

Section	Purpose
Use case number	UI19
Name	book request
Actor	Passenger
Description	The Passenger can send book request online on his mobile

	device.	
Goal	To order taxi online	
Precondition	UI01	
Post condition	Passenger booked taxi	
Basic course of action	Passenger Action	System Response
	1 Passengers want to make booking 3 Passenger fills up booking information. 4 The passenger submits booking.	2 The system allows passengers to enter basic booking information. 5 The system confirms booking. 6 The use case end.
Alternate course of action	A.5 the system indicates that passenger enters invalid information or jumps fields. A.6 the system continues at step three and four. Finally exit form use case.	

Table 17: Use Case Documentation for Payment

Section	Purpose
Use case number	UI20
Name	Payment
Actor	Passenger

Description	The Passenger can pay be using mobile device in online	
Goal	To make payment in online	
Precondition	UI01, view fare calculation	
Post condition	Make payment for the service on online	
Basic course of action	Passenger Action	System Response
	1 Passengers want to pay the service 3.Passenger fill the required form to gate billing system and transfer the money that show at the fare calculation	2 the system leads the Passenger to payment system and shows the payment options 4 The system the system generates report 5 The use case end.
Alternate course of action	A.1 the Passenger inserts invalid information A.2 then continues to step 3, 4 and use case end	

Table 18: Use Case Documentation for Cancel Booking

Section	Purpose
Use case number	UI 21
Name	Cancel booking
Actor	Dispatcher and Passenger

Description	Allows Passengers to cancel booking if there is booked before.
Goal	To cancel booking
Precondition	Make booking
Post condition	Booking canceled
Basic course of action	1 passengers and dispatcher want to cancel booking. 2 The system allows dispatcher and passengers to enter ("cancel booking form"). 3 Dispatcher and passengers enter booking code. 4 The system verifies the booking code. 5 Use case ends.
Alternate course of action	A.3 the system notifies that the Passenger enters invalid booking code. A.4 the system continues to step 4 and use case end.

Table 19: Use Case Documentation for Fare Calculation

Section	Purpose
Use case number	UI 22
Name	Fare calculation
Actor	Drivers, Passenger, Account manager and Admin
Description	Allows the Passenger can show the fare calculation
Goal	To identify fare amount calculation for payment

Precondition	UI01, tracking taxi and after using the service	
Post condition	View the fare calculation or cost estimation	
Basic course of action	Passenger Action	System response
	1 the Passenger, driver, account manager and admin want to view the fare calculation. 2 the Passenger, driver, account manager and admin can show the fare calculation	3 The system displays the cost-estimation/fare-calculation 4 Use case ends.
Alternate course of action	A6 the use case continues step 2, 3, 4 and use case ends.	

4.3. OBJECT MODEL

Object Model is a logical interface, software or system that is modeled through the use of object-oriented techniques. It enables the creation of an architectural software or system model prior to development or programming. In our system we use object model like class diagram and data dictionary.

4.3.1. CLASS DIAGRAM

Class diagram will be the building block system you will develop. Class diagrams should show the objects the system is comprised of and how they are interrelated. Class models contain a wealth of information; it can be used for both the analysis and design of systems [3]. To create and evolve a class model:

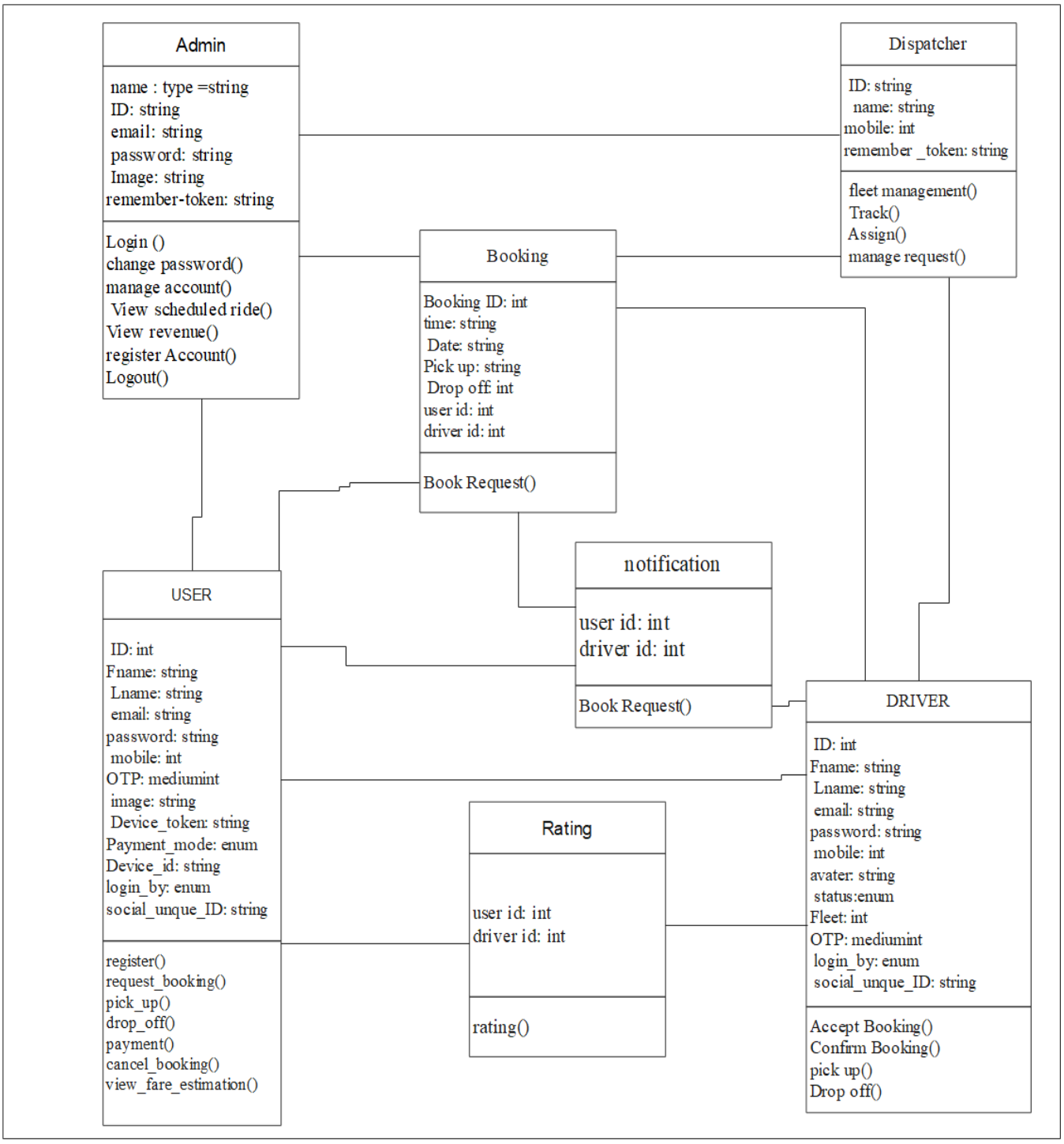


Figure 2: Class Diagram

4.3.2 DATA DICTIONARY

A data dictionary is a file or a set of files that includes a database's metadata. The data dictionary holds records about other objects in the database, such as data ownership, data

relationships to other objects, and other data. In our system the following table describe data dictionary.

Table 20: Data Dictionary for Admin

Attribute	Data type	Data size	Constraint
ID	Int	30	NOT NULL
Name	String	50	
Email	String	50	NOT NULL
Password	String	50	NOT NULL
Image	Long	100	NOT NULL
Remember token	String	40	

Table 21: Data Dictionary for Passenger

Attribute	Data type	Data size	Constraint
ID	Int	50	NOT NULL
Fname	String	50	NOT NULL
Lname	String	50	NOT NULL
Email	String	50	NOT NULL
Password	String	50	NOT NULL
Mobile	Int	50	NOT NULL
Rating	Decimal	50	
Latitude	Double	50	NOT NULL
Longitude	Double	50	NOT NULL

OTP	Medium int	50	NOT NULL
Image	Long	50	NOT NULL
Device token	String	50	
Payment mode	Enum	50	
Device ID	String	50	
Login by	Enum	50	
Social unique ID	String	50	

Table 22: Data Dictionary for Driver

Attribute	Data type	Data size	Constraint
ID	Int	50	NOT NULL
Fname	String	50	
Lname	String	50	
Email	String	50	NOT NULL
Password	String	50	NOT NULL
Mobile	Int	50	NOT NULL
Rating	Decimal	50	
Latitude	Double	50	NOT NULL
Longitude	Double	50	NOT NULL
OTP	Medium int	50	NOT NULL
Image	Long	50	NOT NULL

Device token	String	50	
Payment mode	Enum	50	
Device ID	String	50	

Table 23: Data Dictionary for Dispatcher

Attribute	Data type	Data size	Constraint
ID	Int	50	NOT NULL
Name	String	50	NOT NULL
Mobile	Int	50	NOT NULL
Remember token	String	40	

4.3.3 DYNAMIC MODEL

A dynamic model represents the behavior of an object over time. It is used where the object's behavior is best described as a set of states that occur in a defined sequence.

4.3.4 SEQUENCE DIAGRAM

Sequence diagrams model the interactions between objects in a single use case. They illustrate how the different parts of a system interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario [3]. In our project there will be different process to do specific actions and we include some sequence diagrams to handle these interactions.

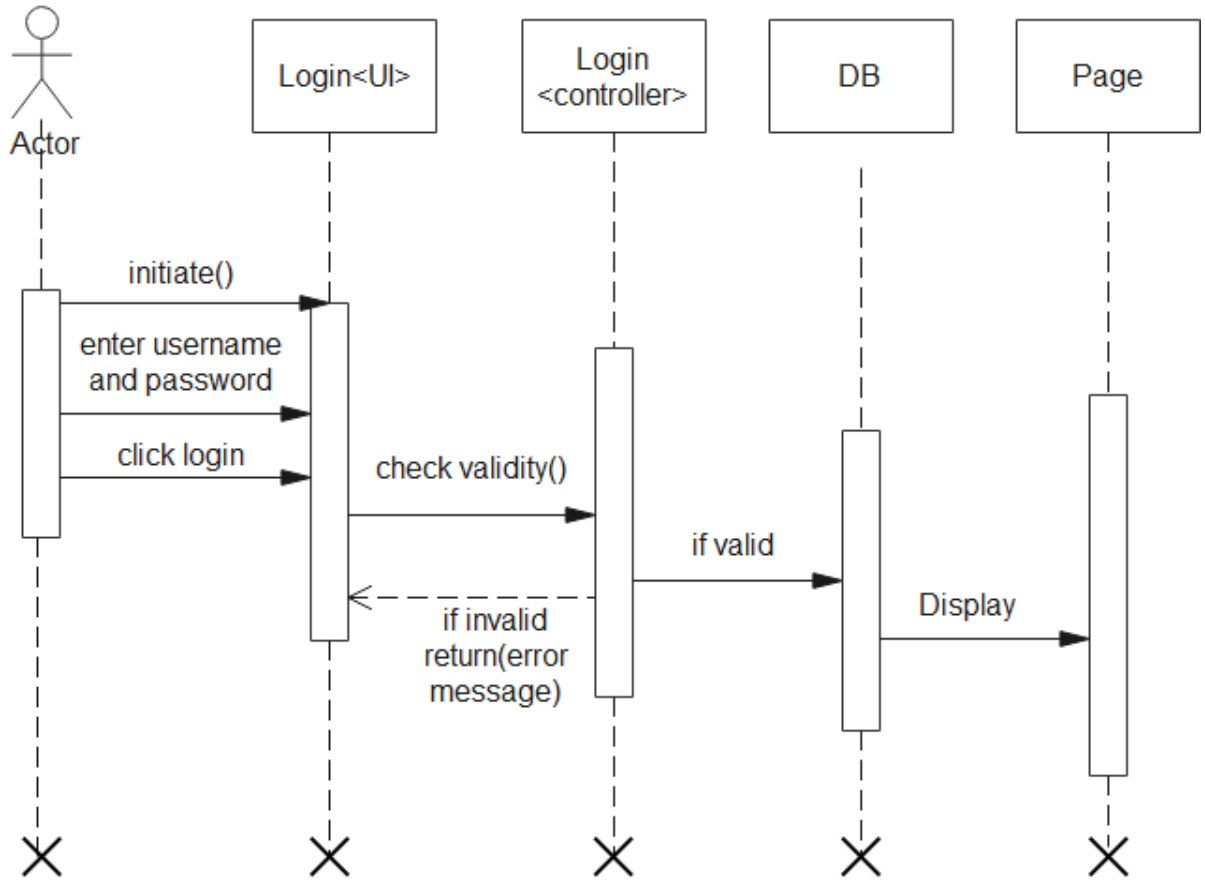


Figure 3: Sequence Diagram for Log In.

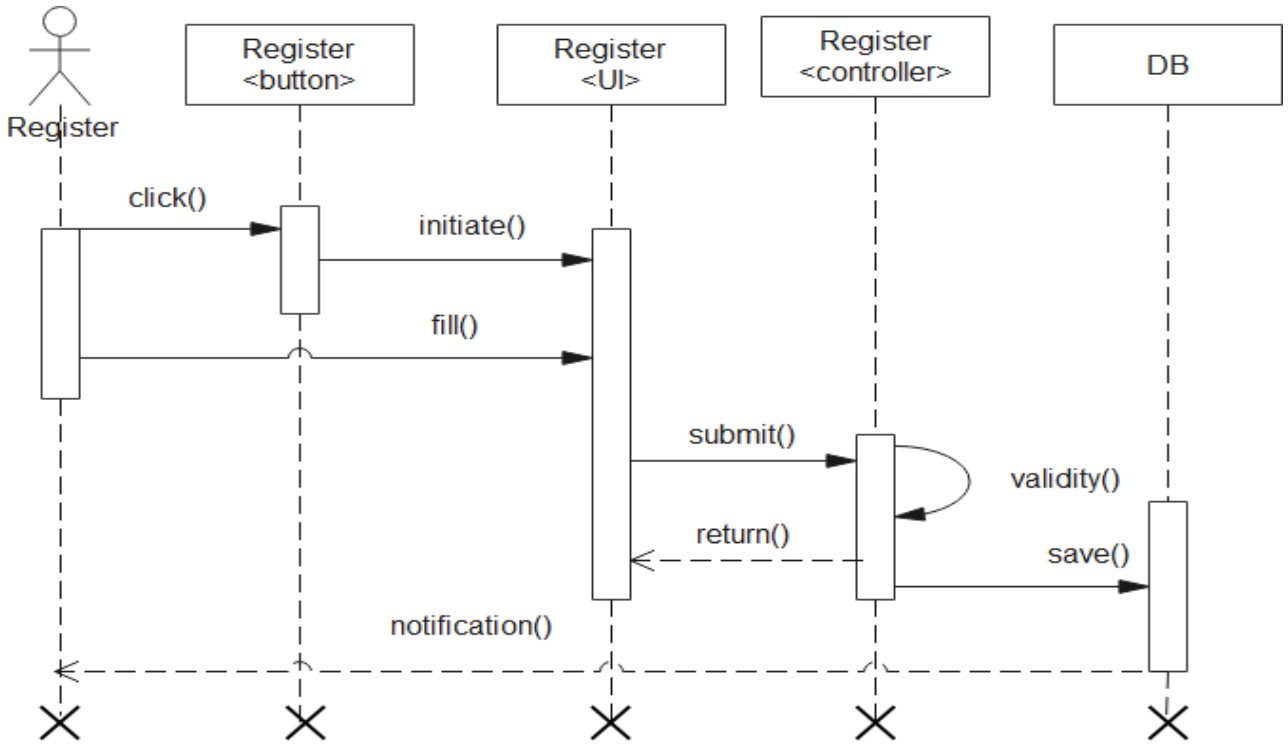


Figure 4: Sequence Diagram for Register

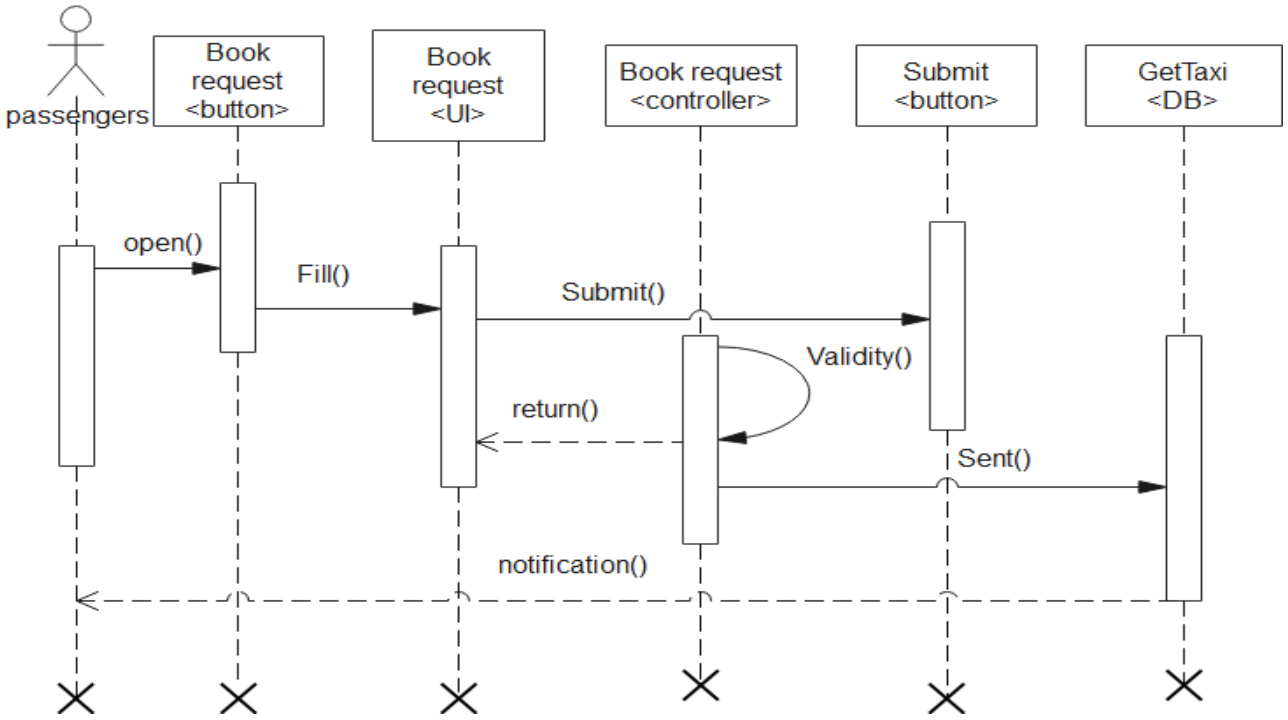


Figure 5: Sequence Diagram for Book Request

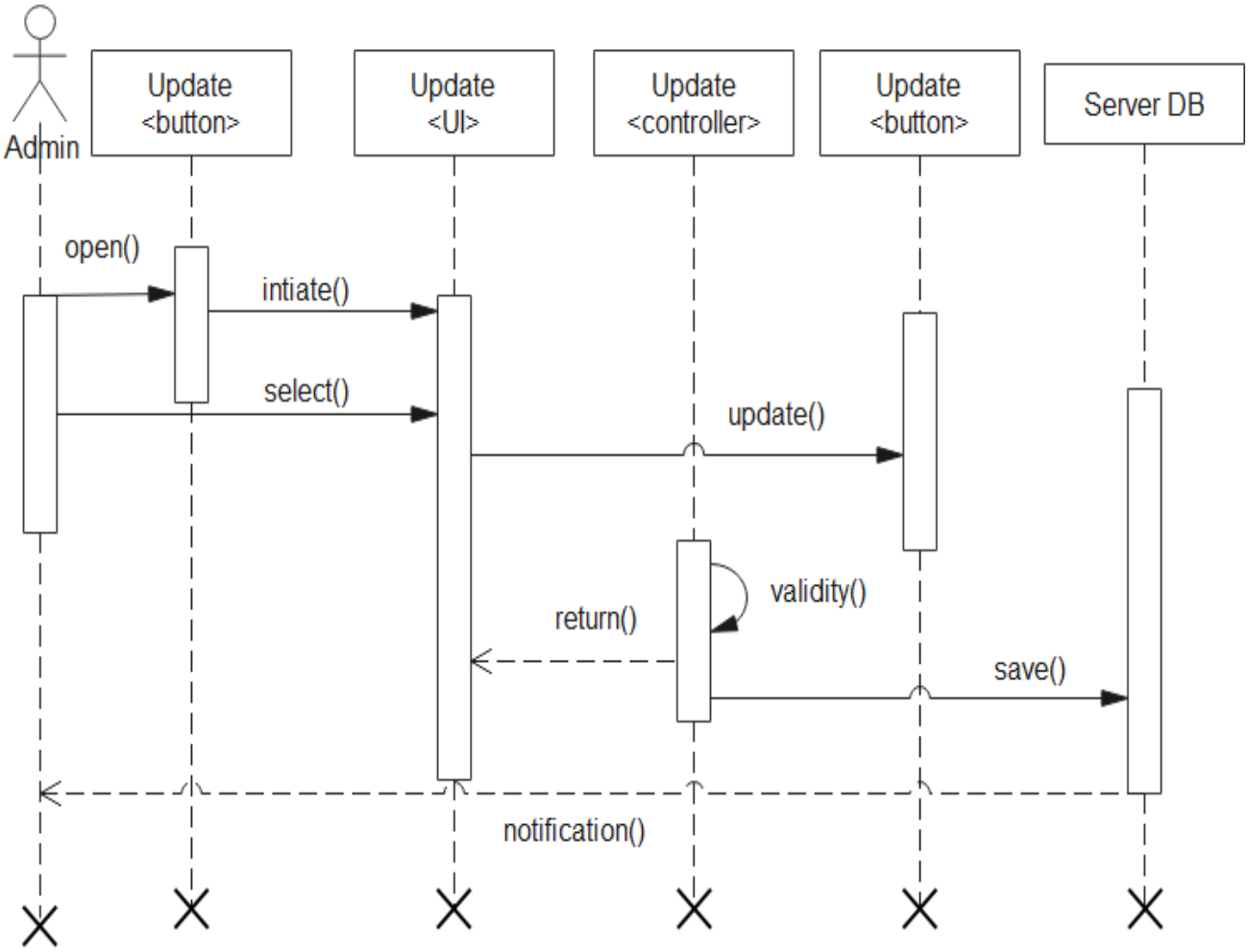


Figure 6: Sequence Diagram for Update

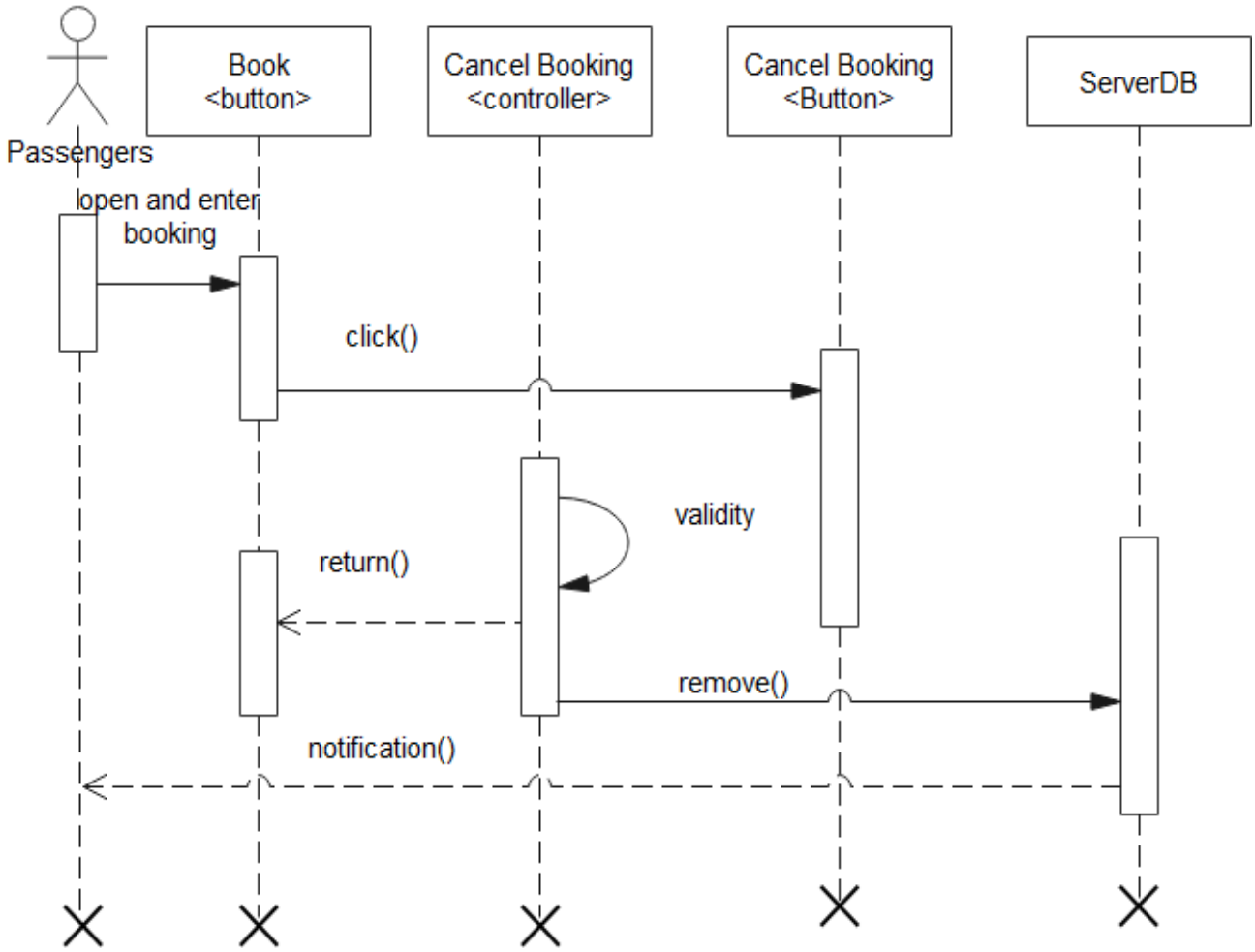


Figure 7: Sequence Diagram for Cancel Booking

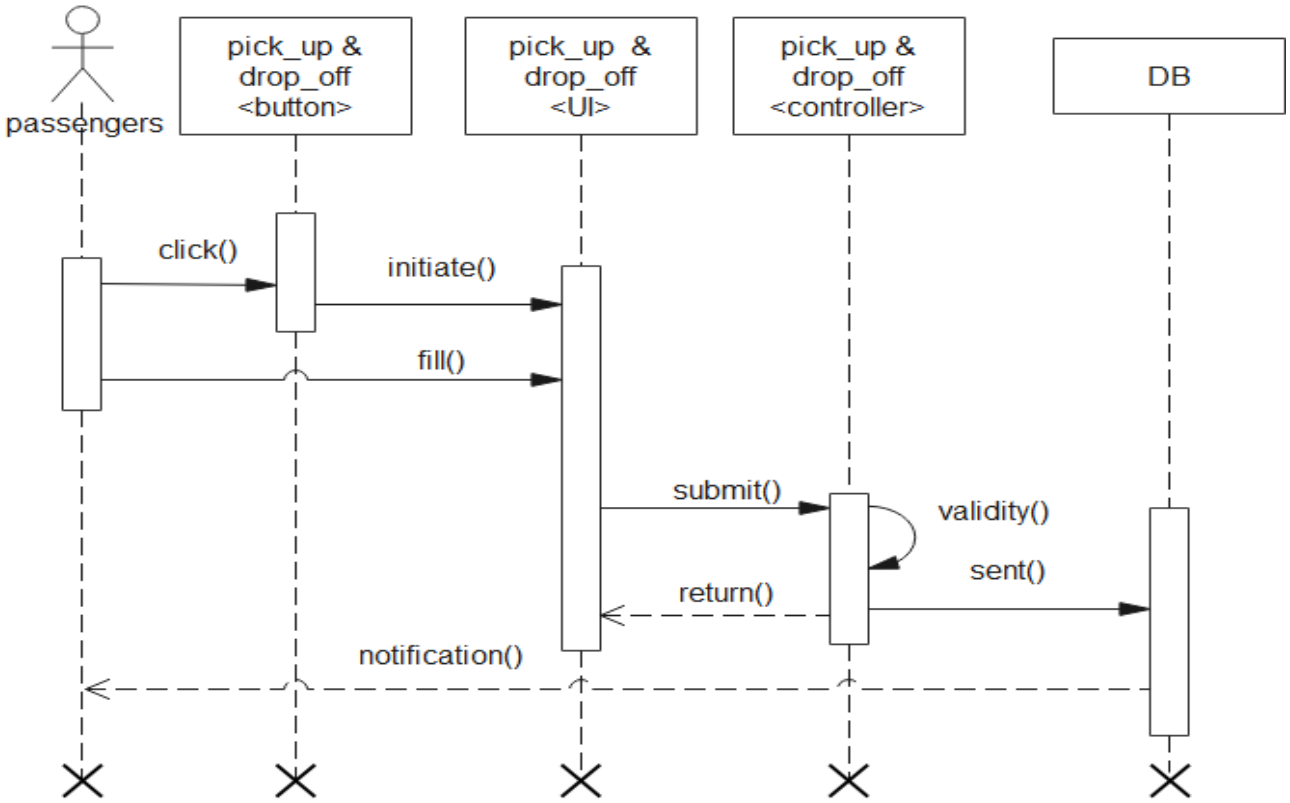


Figure 8: Sequence Diagram for Pickup

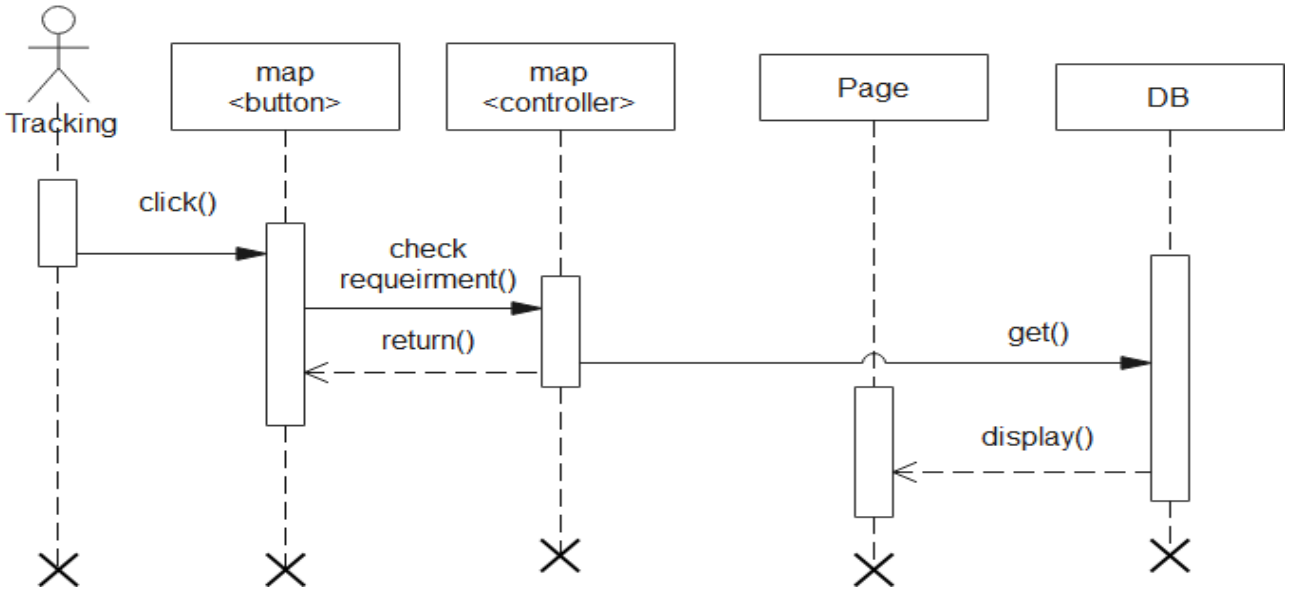


Figure 9: Sequence Diagram for Map

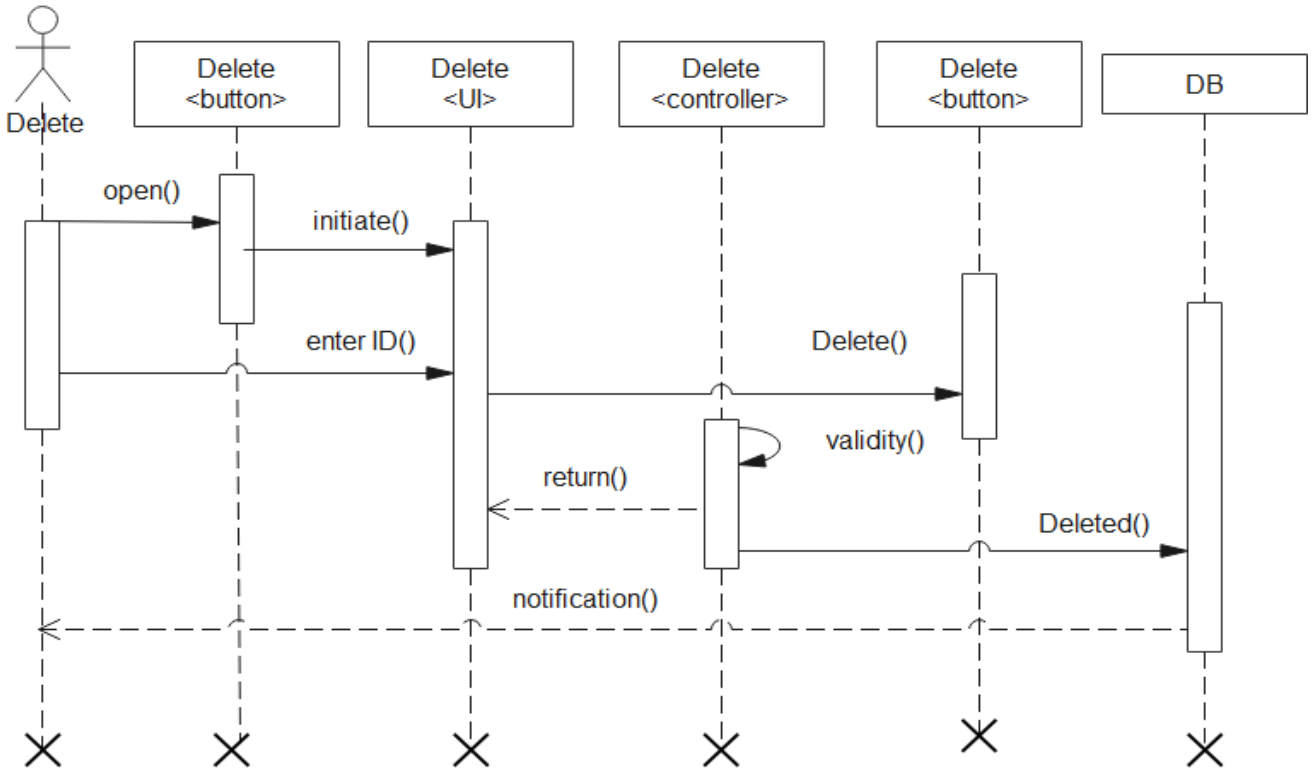


Figure 10: Sequence Diagram for Delete

4.3.5 ACTIVITY DIAGRAM

An Activity diagram is similar to a flowchart to represent the flow from one activity to another activity. Activity diagrams and State chart diagrams are related. While a State chart diagram focuses attention on an object undergoing a process (or on a process as an object), an Activity diagram focuses on the flow of activities involved in a single process. The Activity diagram shows how these single-process activities depend on one another [3].

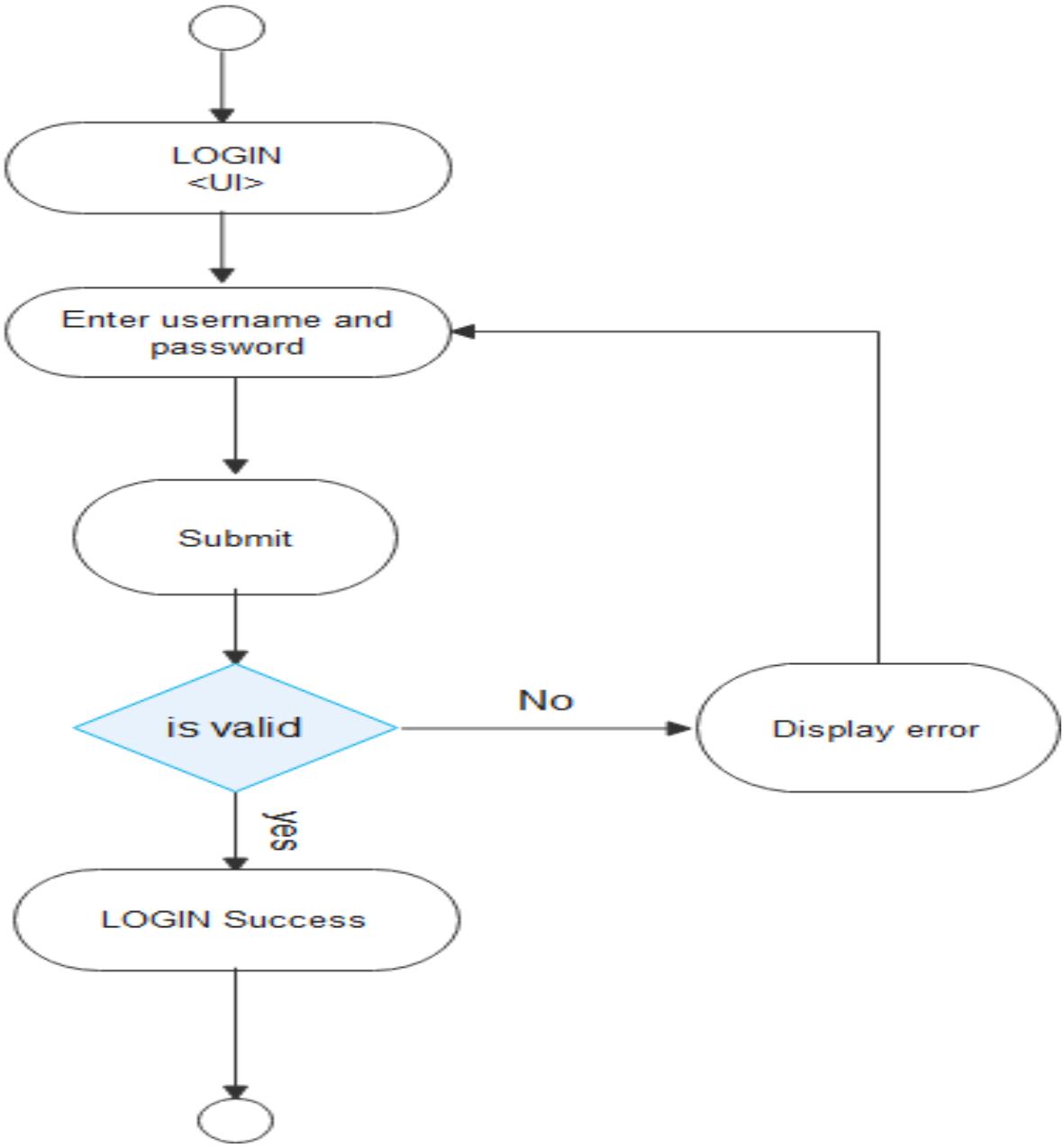


Figure 11: Activity Diagram for Login

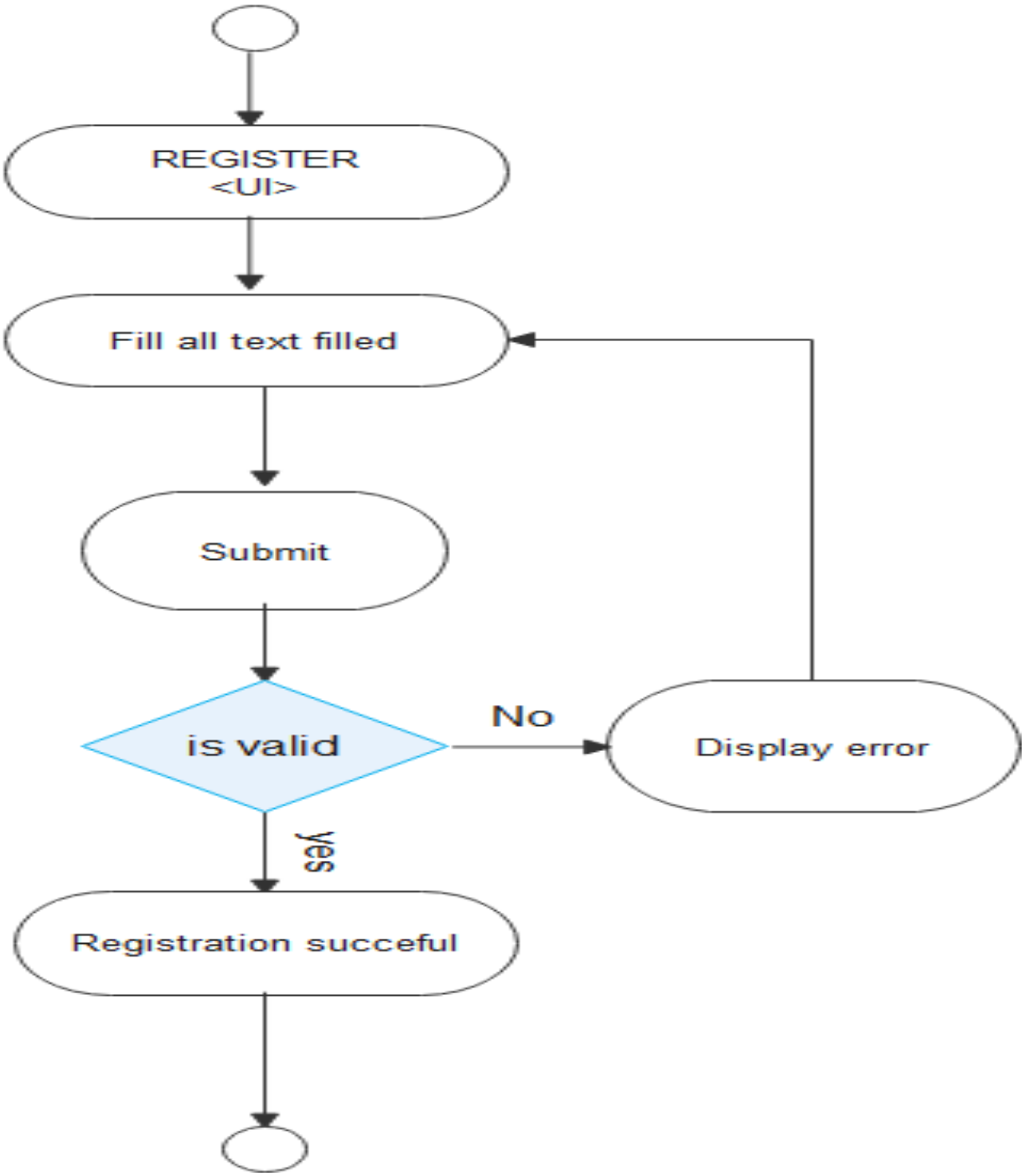


Figure 12: Activity Diagram for Delete

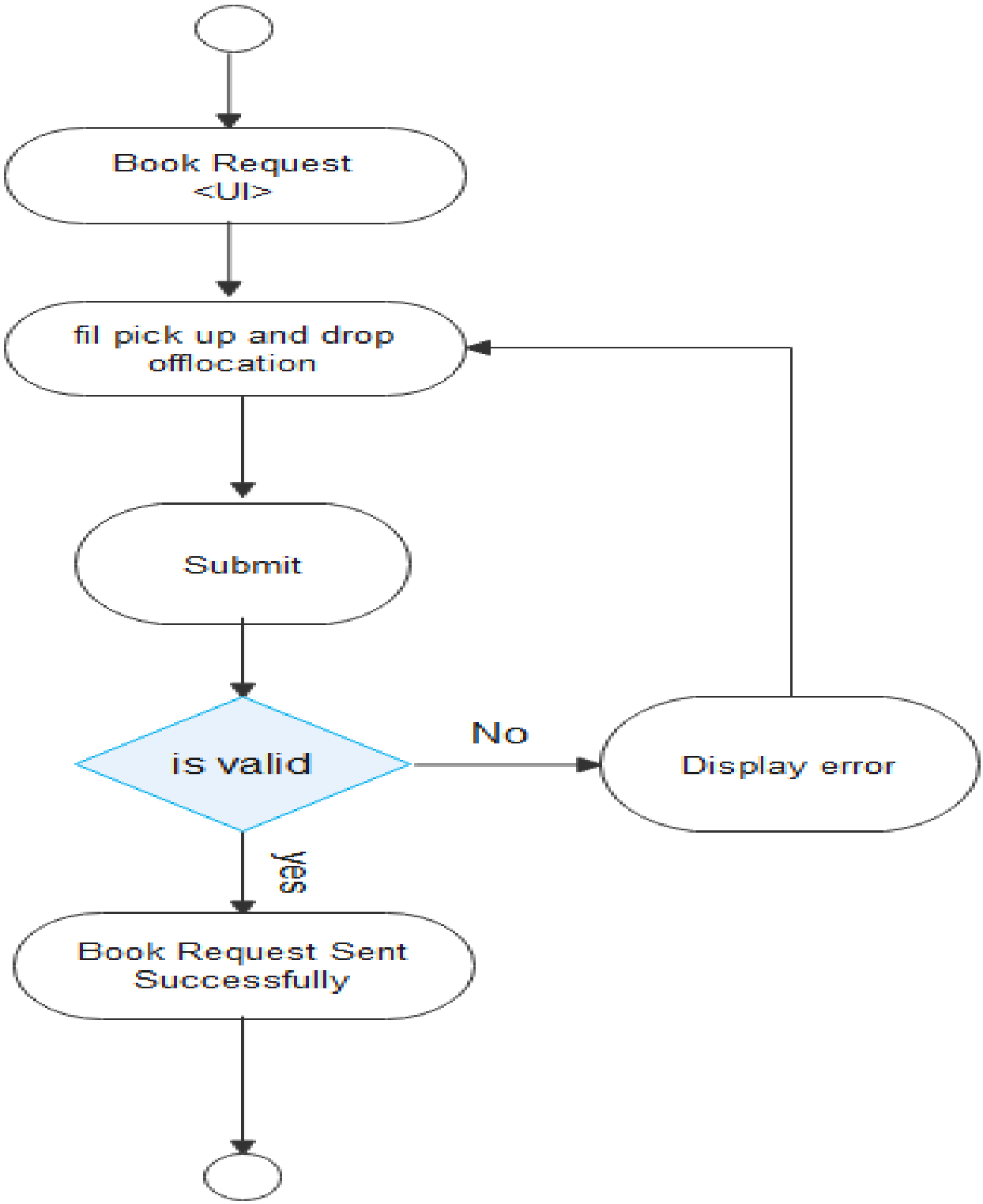


Figure 13: Activity Diagram for Book Request

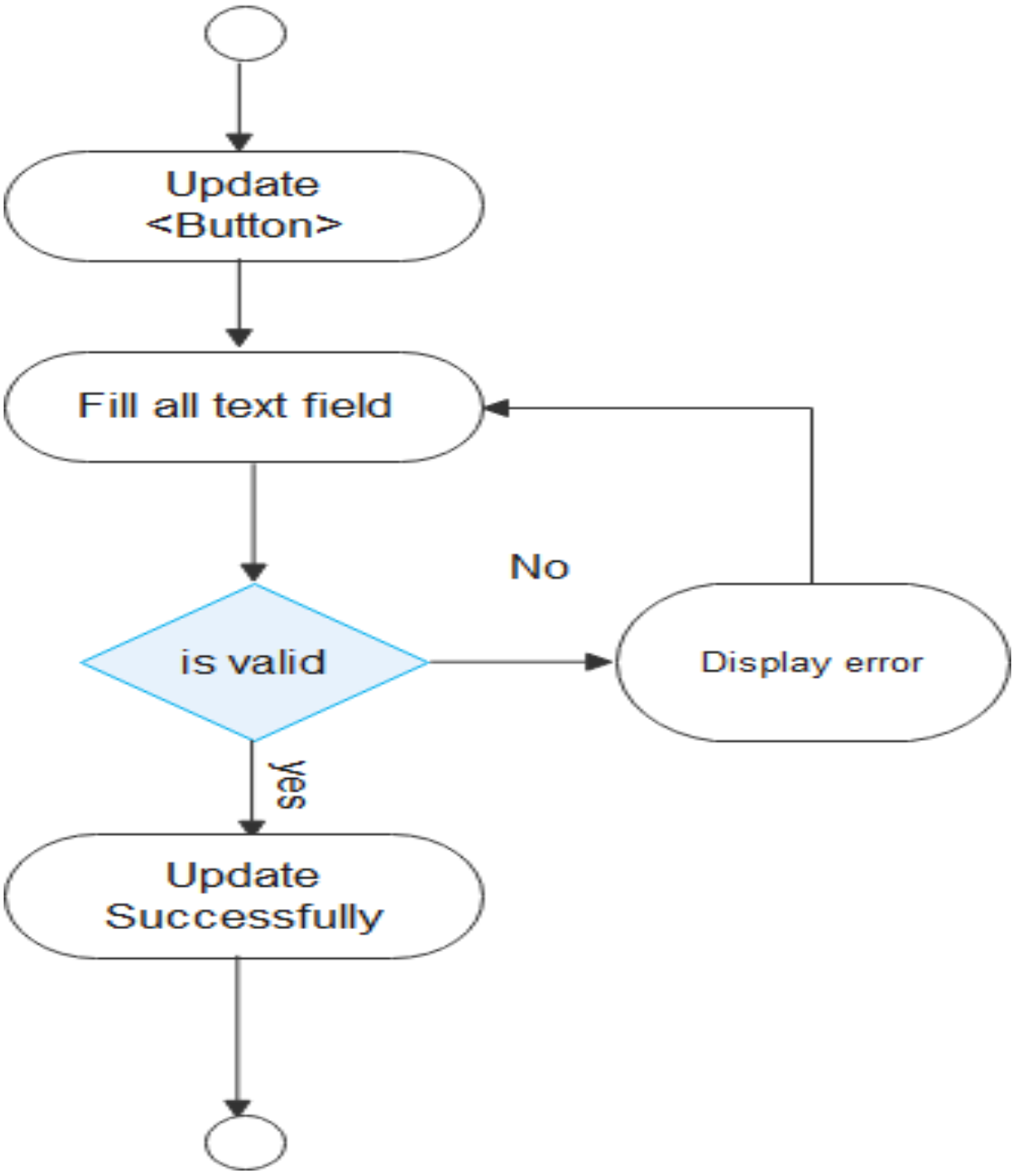


Figure 14: Activity Diagram for Update

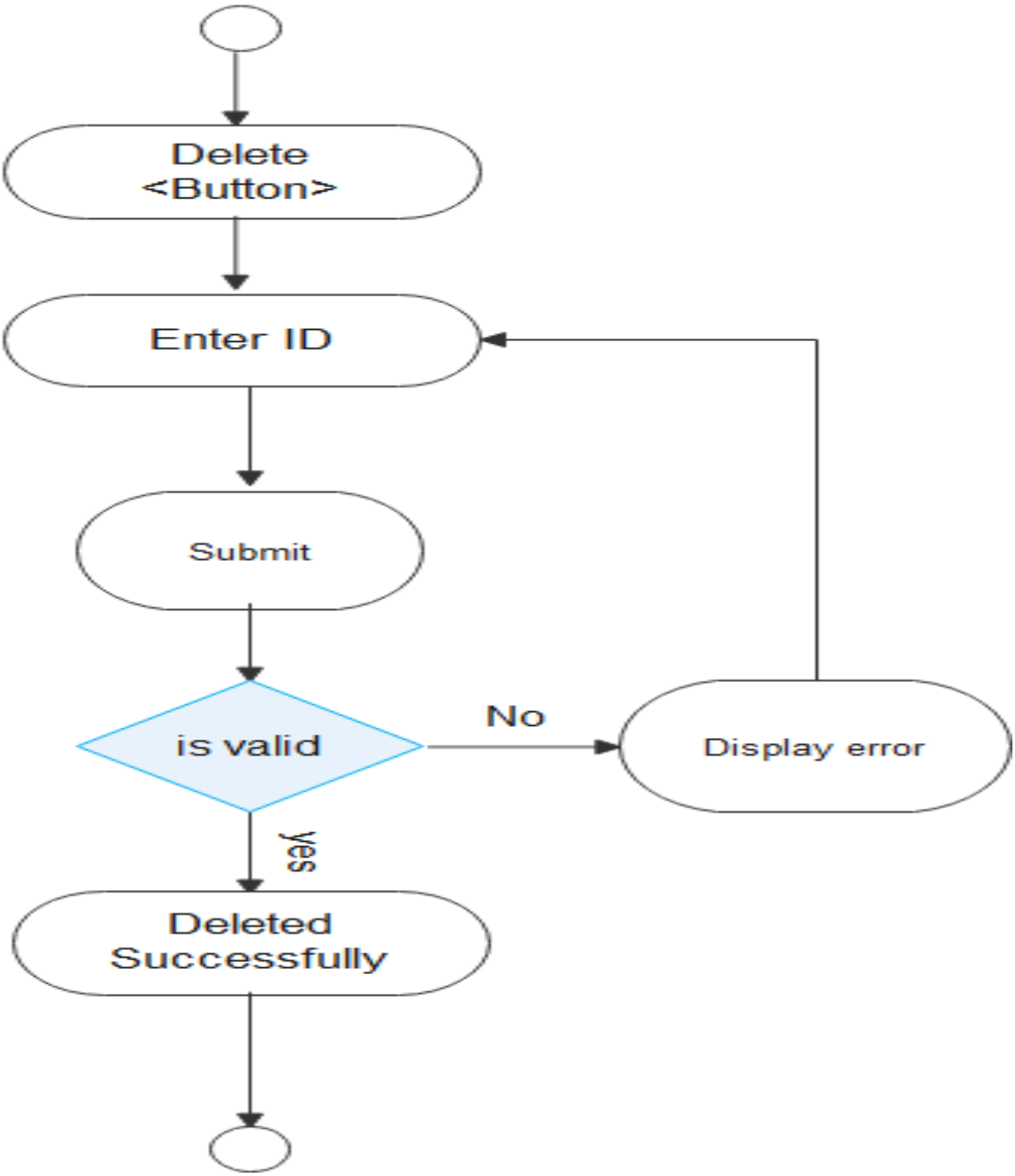


Figure 15: Activity Diagram for Delete

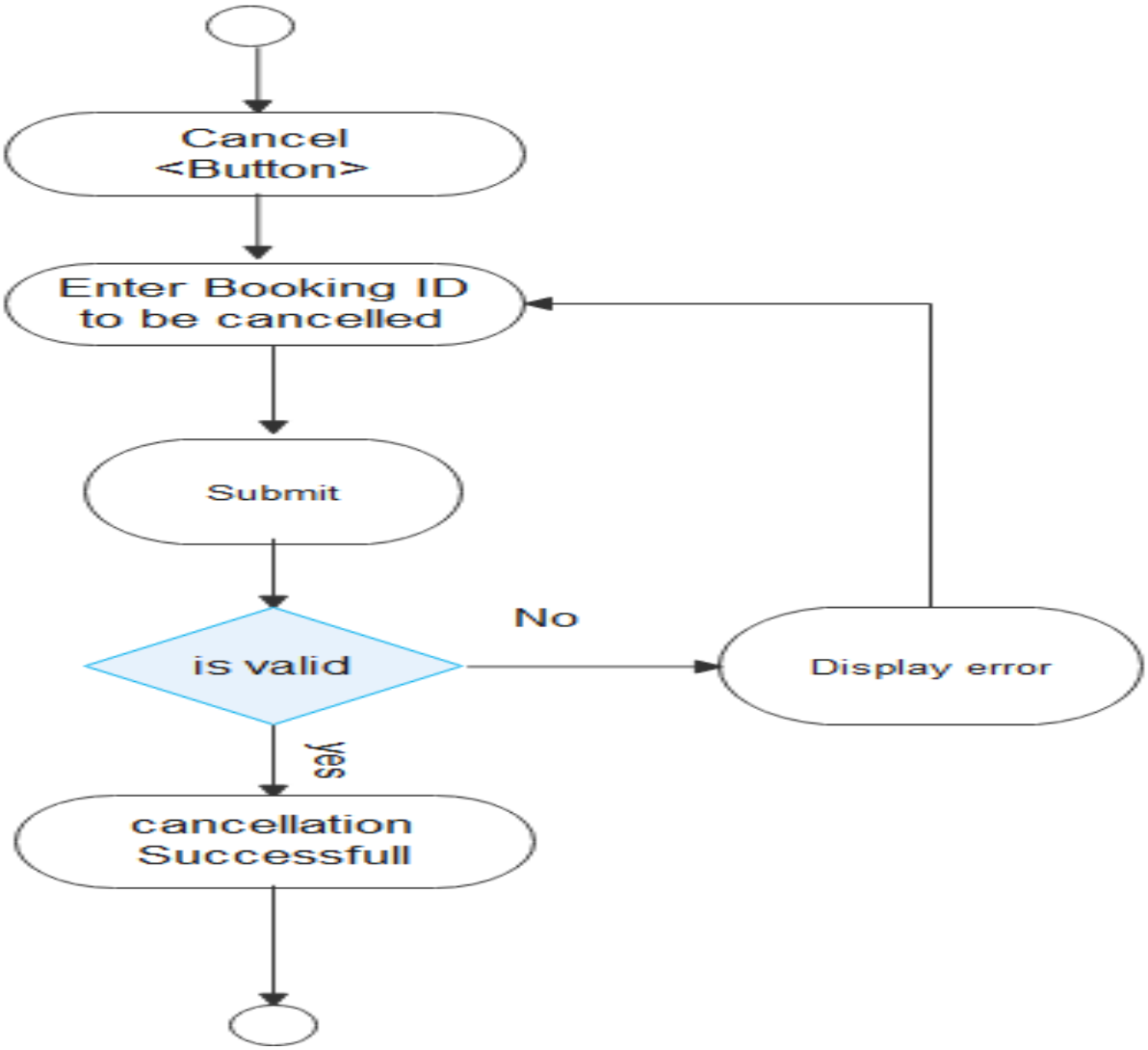


Figure 16: Activity Diagram for Cancel

4.3.6 STATE CHART DIAGRAM

State Chart Diagram is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML). State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. In our project we have listed below some states that can be covered during the Passenger interacts with the system.

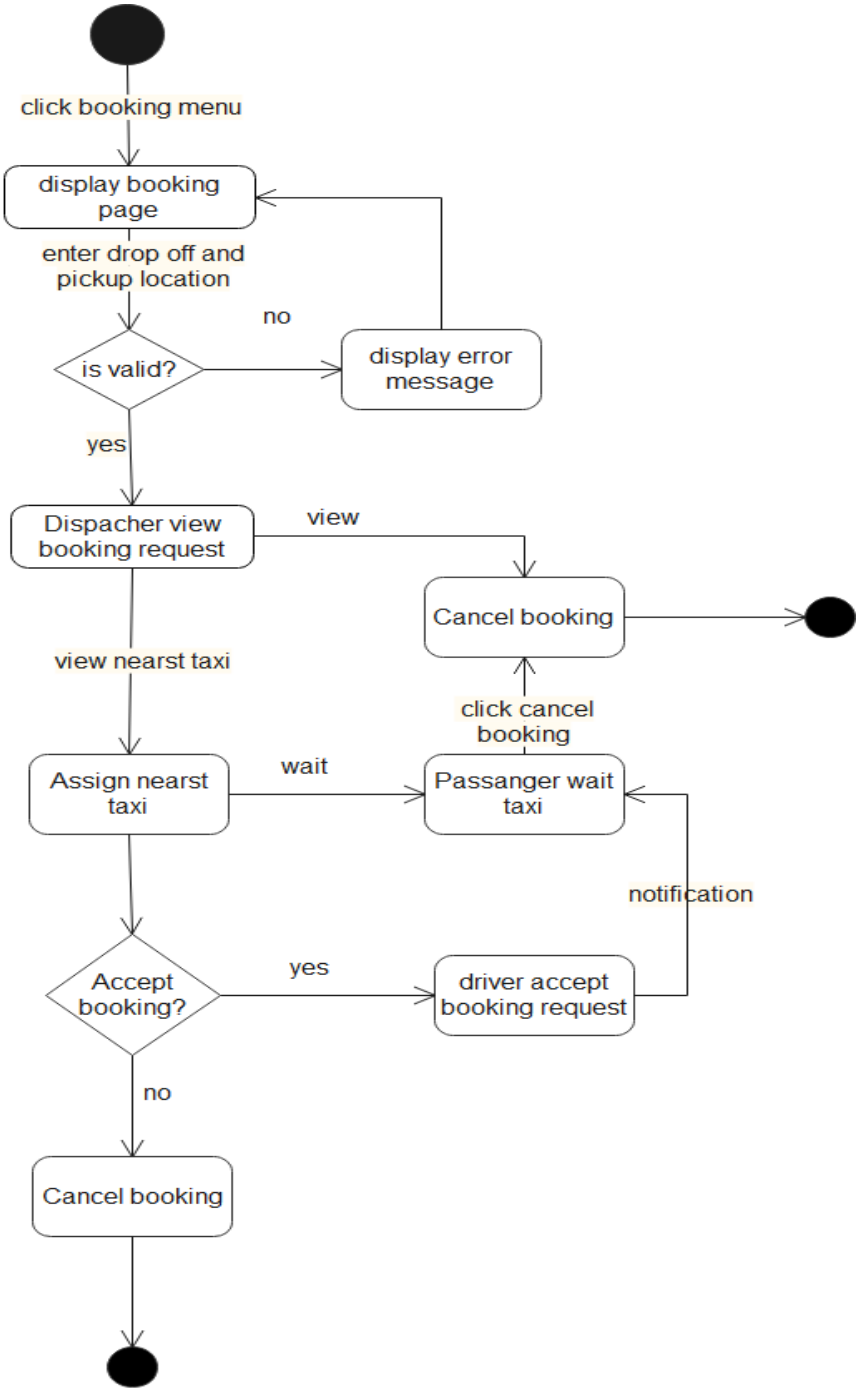


Figure 17:State Chart Diagram for Booking

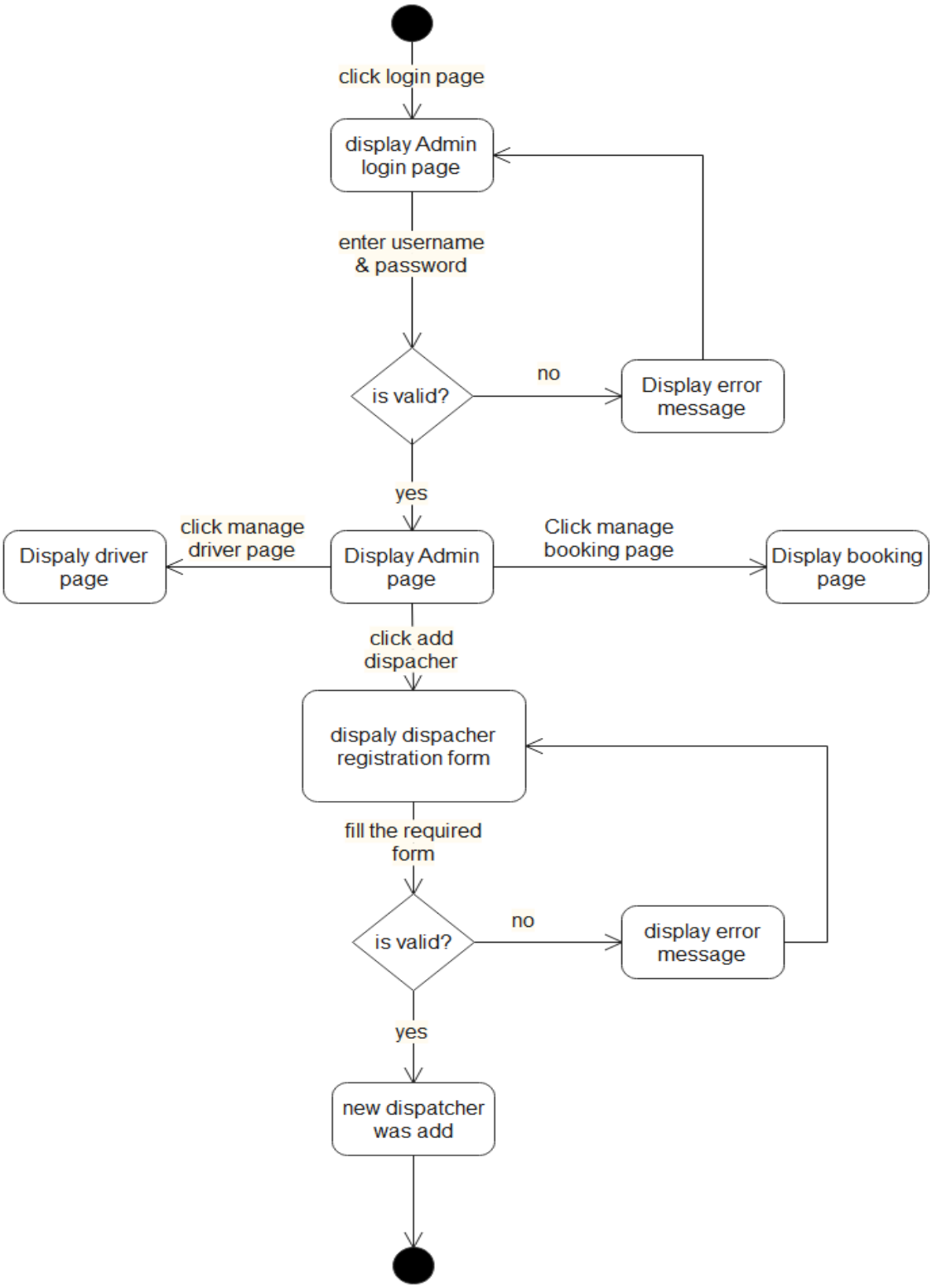


Figure 18:State Chart Diagram for Admin

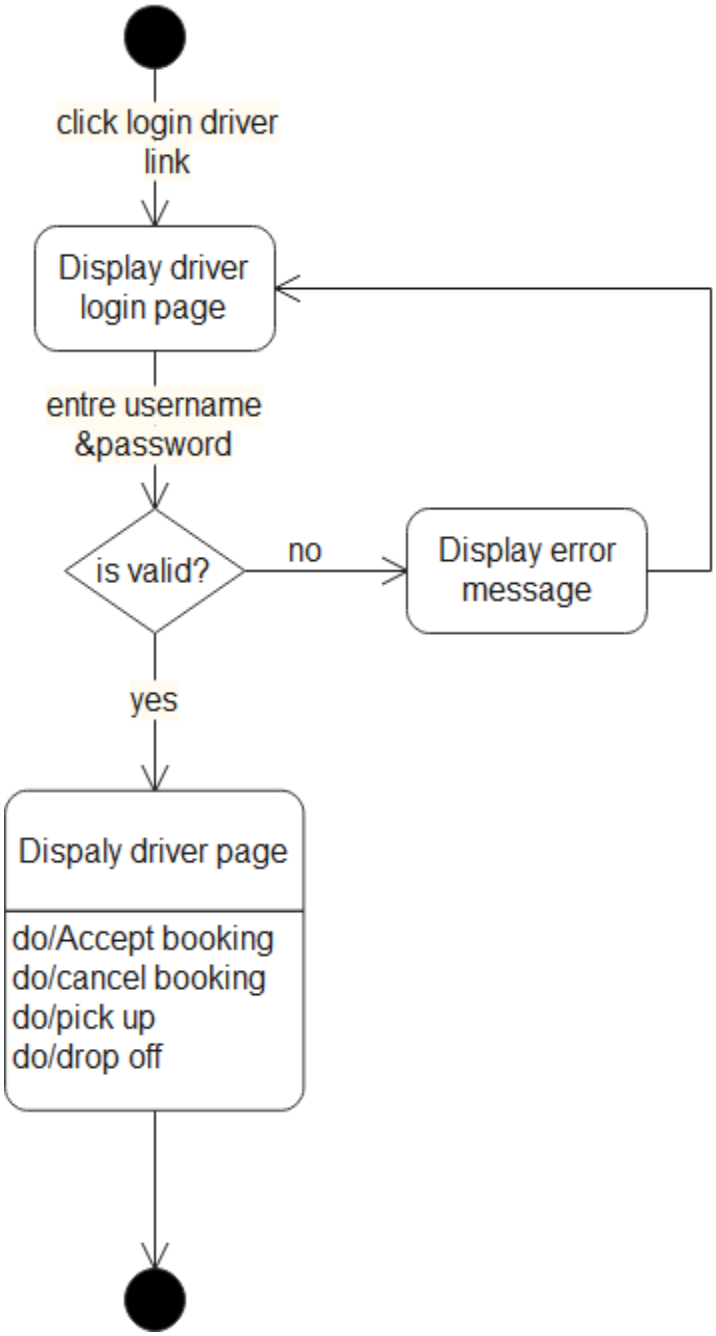


Figure 19:State Chart Diagram for Driver

CHAPTER FIVE

5. SYSTEM DESIGN

5.1 INTRODUCTION

The purpose of designing is to show the direction how the system is built and to obtain clear and enough information needed to drive the actual implementation of the system. It is based on understanding of the model the software built on. The objectives of design are to model the system with high quality. Implementing of high quality system depend on the nature of design created by the designer. If one wants to changes to the system after it has been put in to operation depends on the quality of the system design. So, if the system is design effectively, it will be easy to make changes to it.

5.2 PURPOSE AND GOALS OF DESIGN

The design goals represent the desired qualities the system should have and provide a consistent set of criteria that would be taken into consideration when making design decisions. The following are mentioned as the design goals of "android based Taxi Booking system ".

Security: The system should authenticate it's Passengers by prompting them to enter Passenger name and password in order to get access to the system.

Extensibility: the system should allow any additional services easily if needed, in other words it should not be difficult to extend the system if additions are necessary.

Availability: the system should be available every time the Passenger needs to access it.

Usability: the system should have Passenger friendly Passenger interface to allow the Passenger to interact with the system easily.

Portability: the system should be able to run on any mobile that supports j2me mobile applications.

Performance: the main performance measure for a project is time, so the system should give fast responses for Passenger requests.

5.3 PROPOSED SOFTWARE ARCHITECTURE

The architecture which will be used for the proposed system is a two tier architecture where the client or the Passenger side is a mobile phone containing Passenger interfaces like data entry interfaces, it is used to display information to the Passenger. Passenger directly interacts with the system through the interfaces on this layer.

The data layer or the database is responsible for storing all information needed for the system to function correctly

.

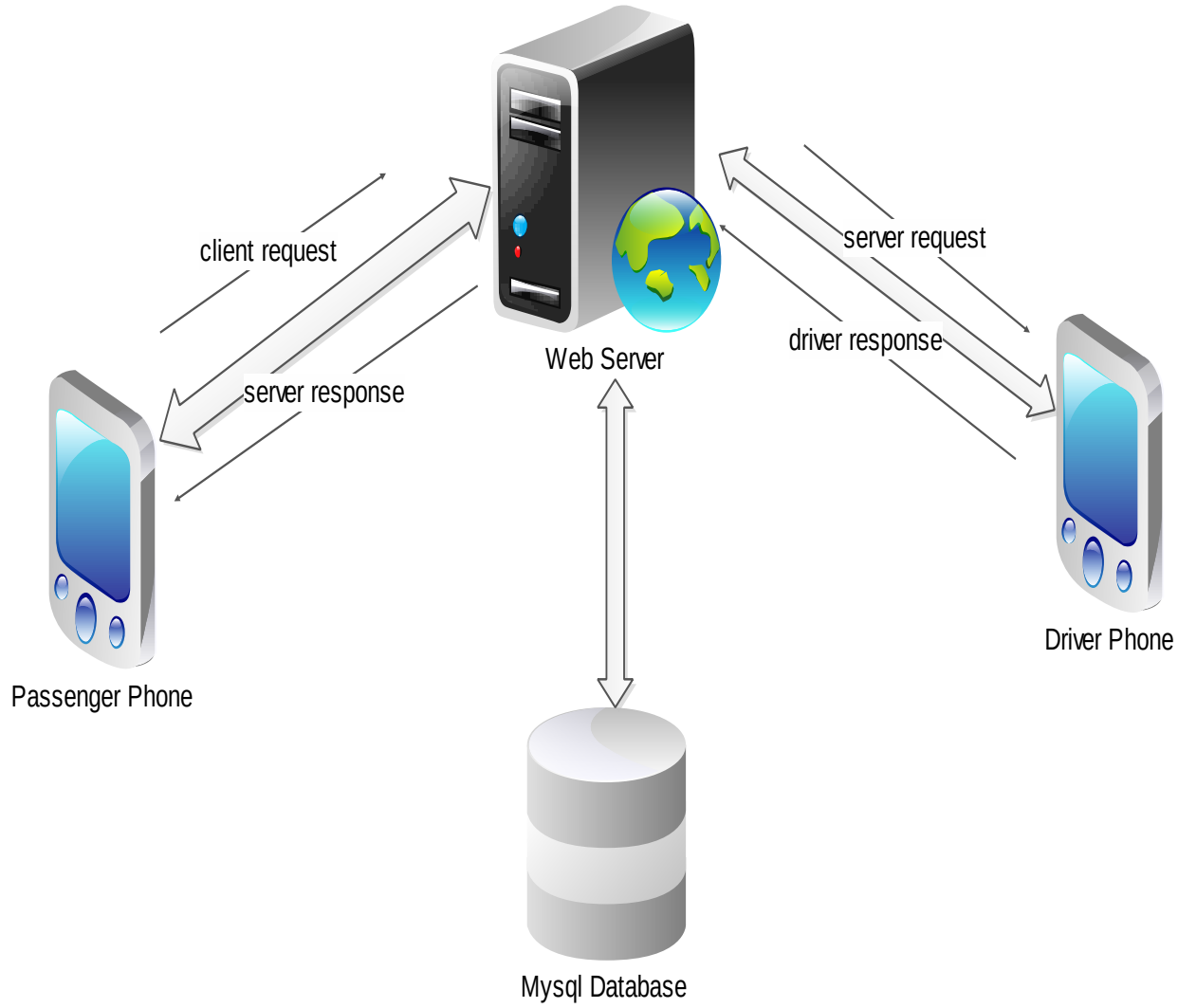


Figure 20: Proposed Software Architecture

5.3.1. SUBSYSTEM DECOMPOSITION AND DESCRIPTION

Subsystem decompositions will help reduce the complexity of the system. The subsystems can be considered as packages holding related classes/objects.

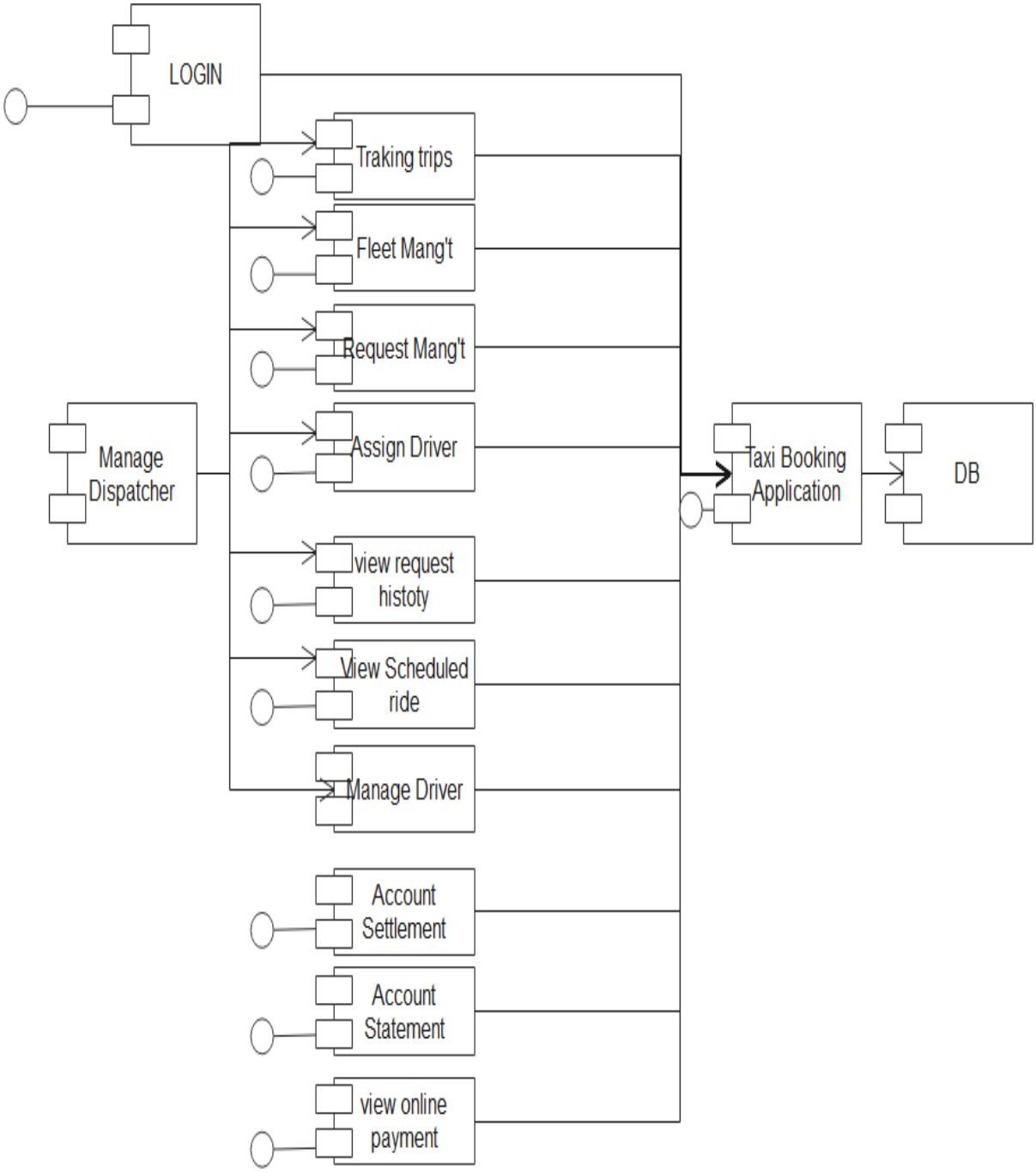


Figure 18: Component Diagram for Admin Functionality

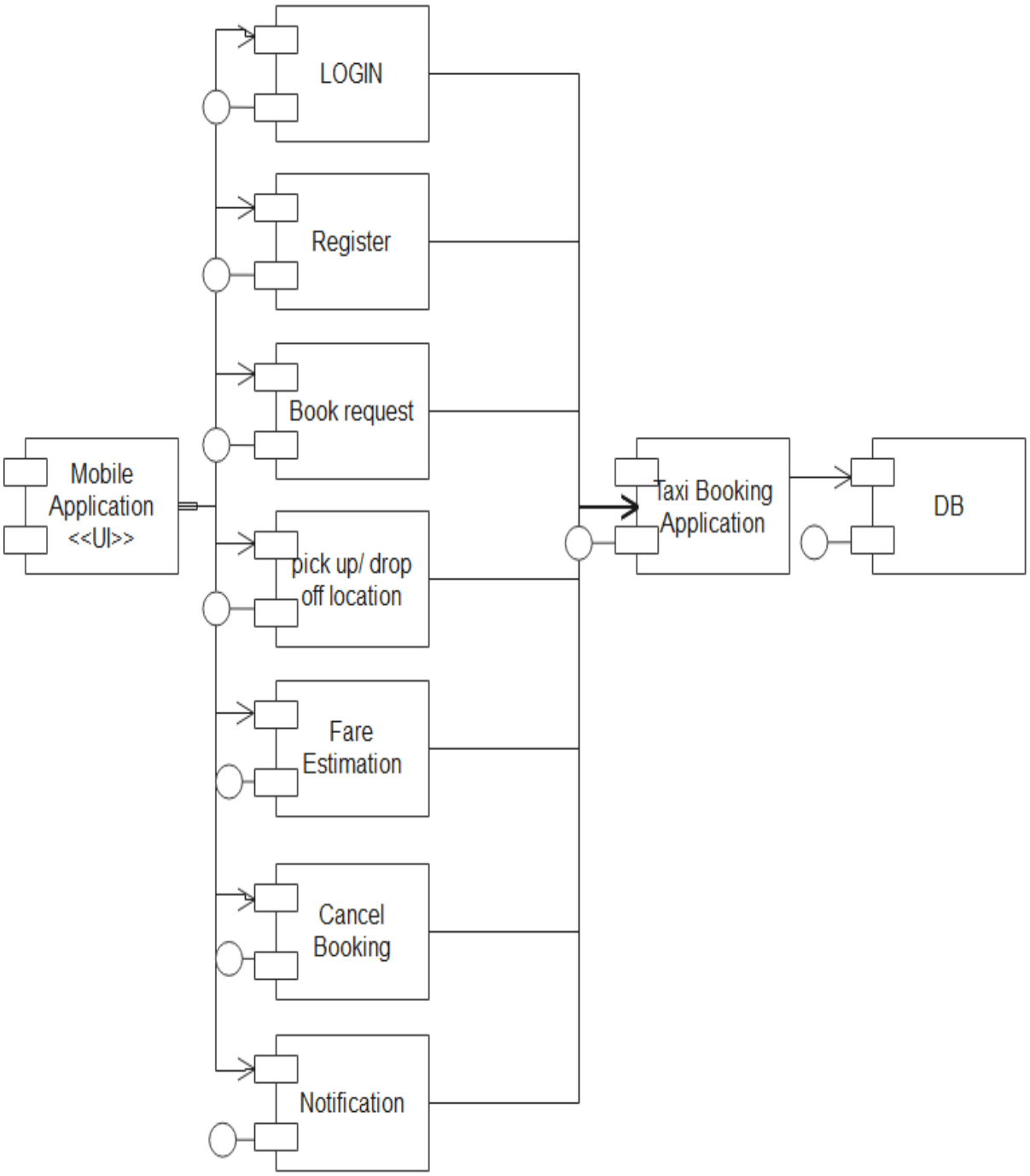


Figure 19: component diagram for Passenger functionality

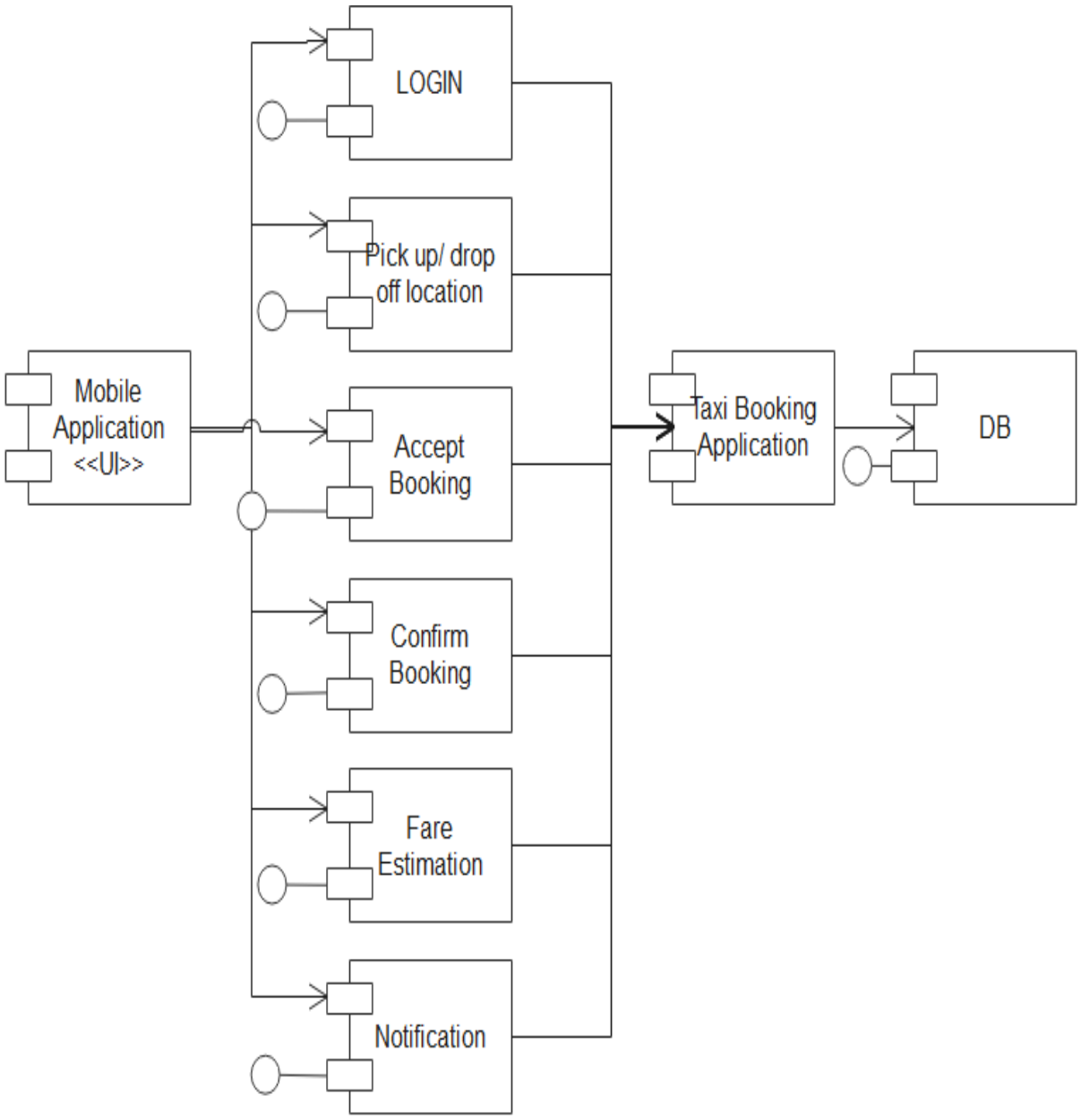


Figure 20: Component Diagram for Driver Functionality

5.3.2 DEPLOYMENT DIAGRAM

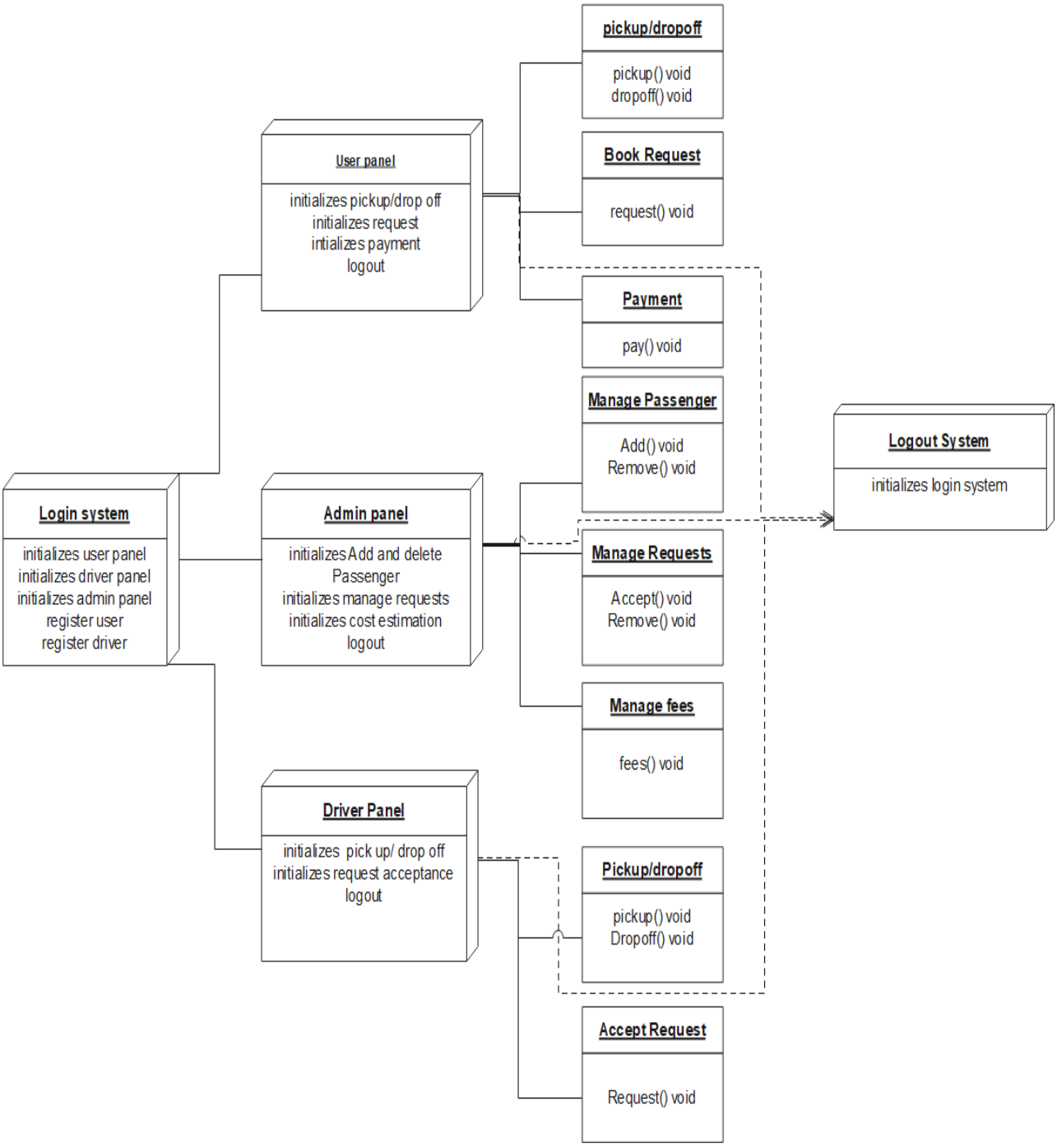


Figure 21: Deployment Diagram

5.3.3. DETAILED CLASS DIAGRAM

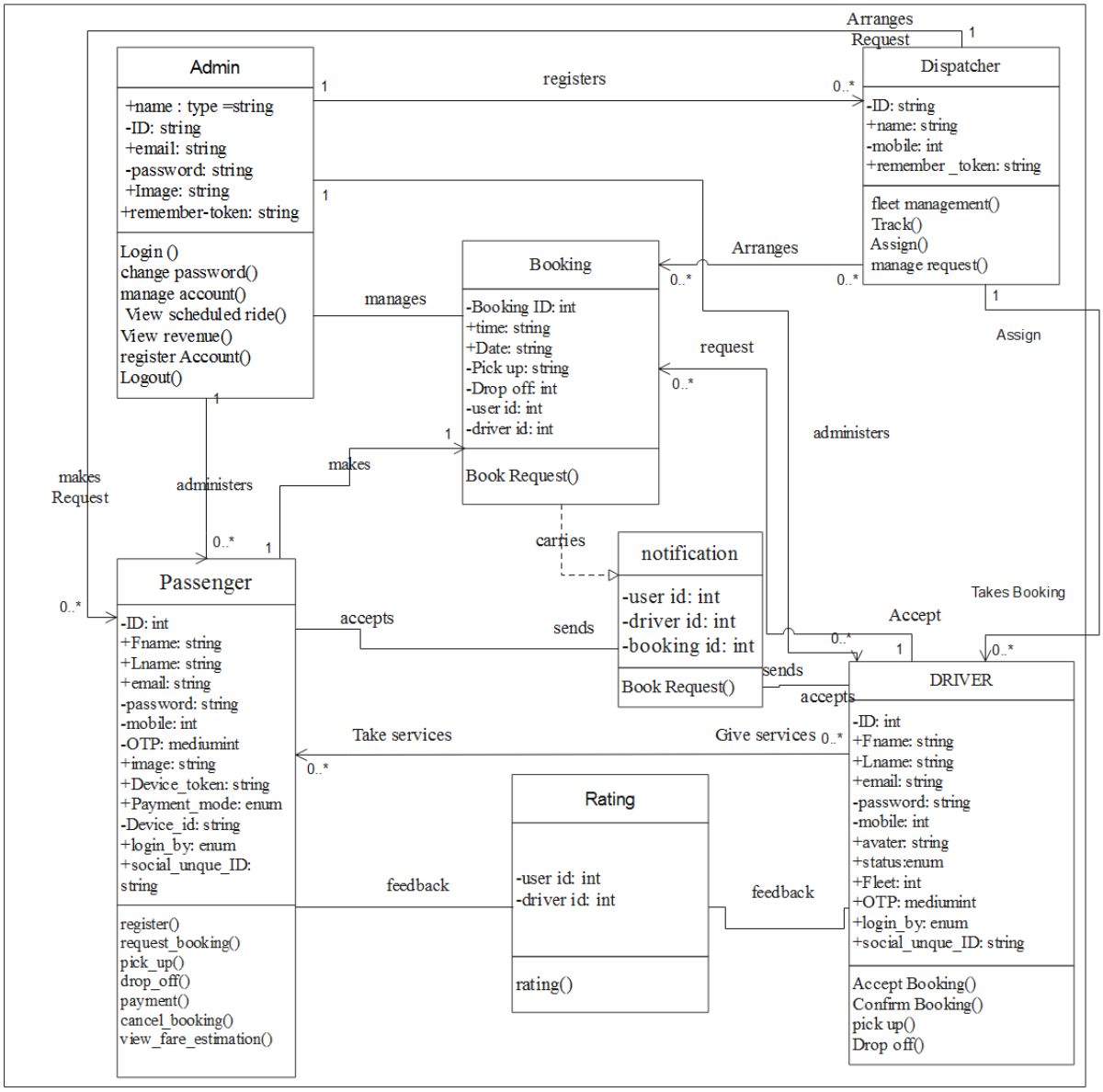


Figure 22: Class Diagram

5.3.4. PERSISTENT DATA MANAGEMENT

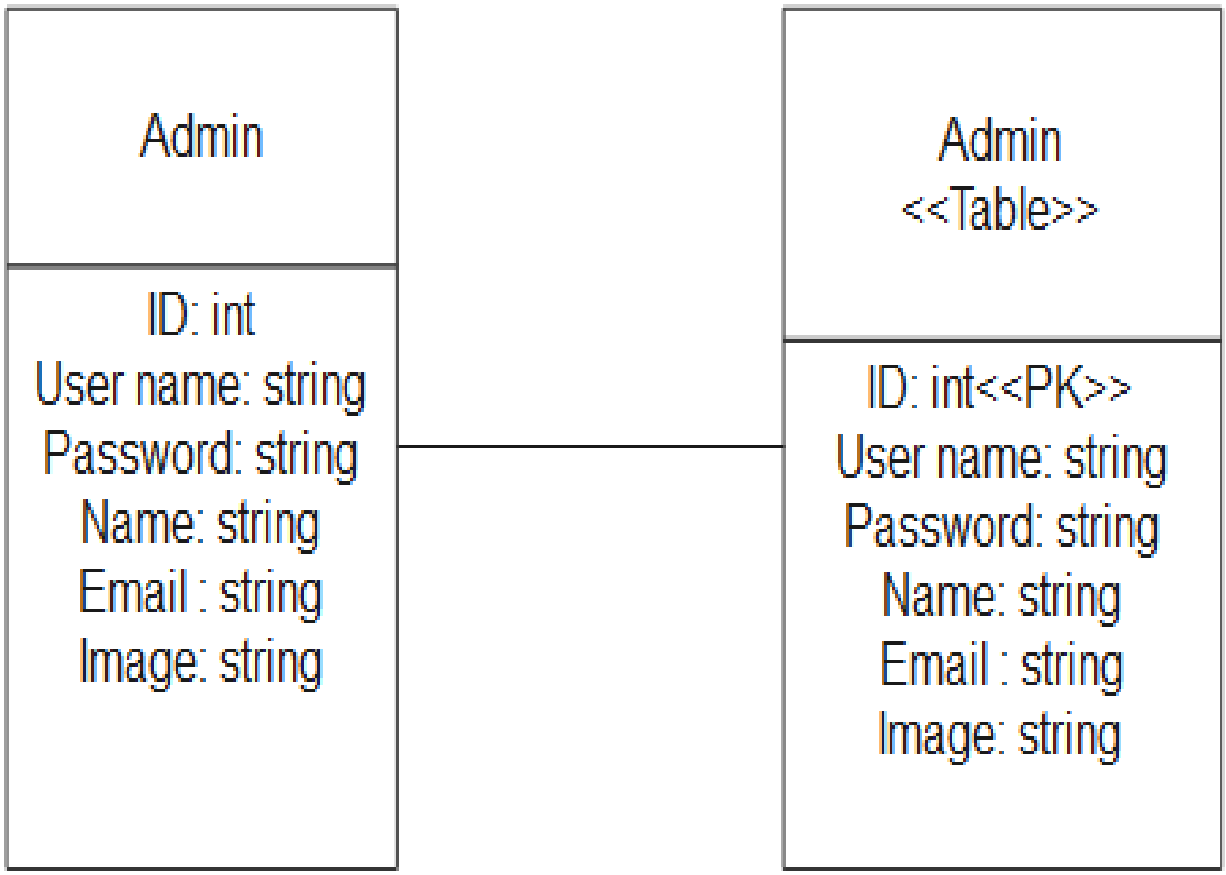


Figure 23: Persistence Diagram for Admin

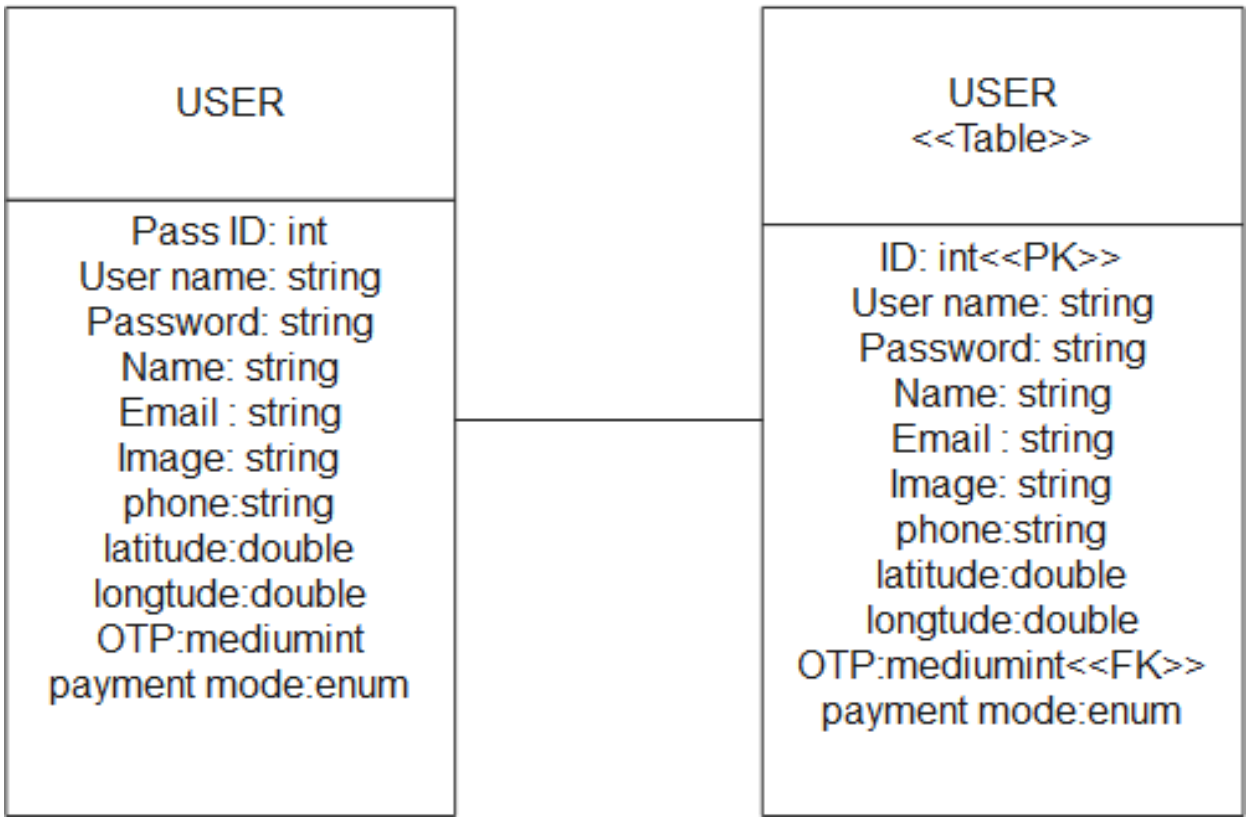


Figure 24: Persistence Diagram For Passenger

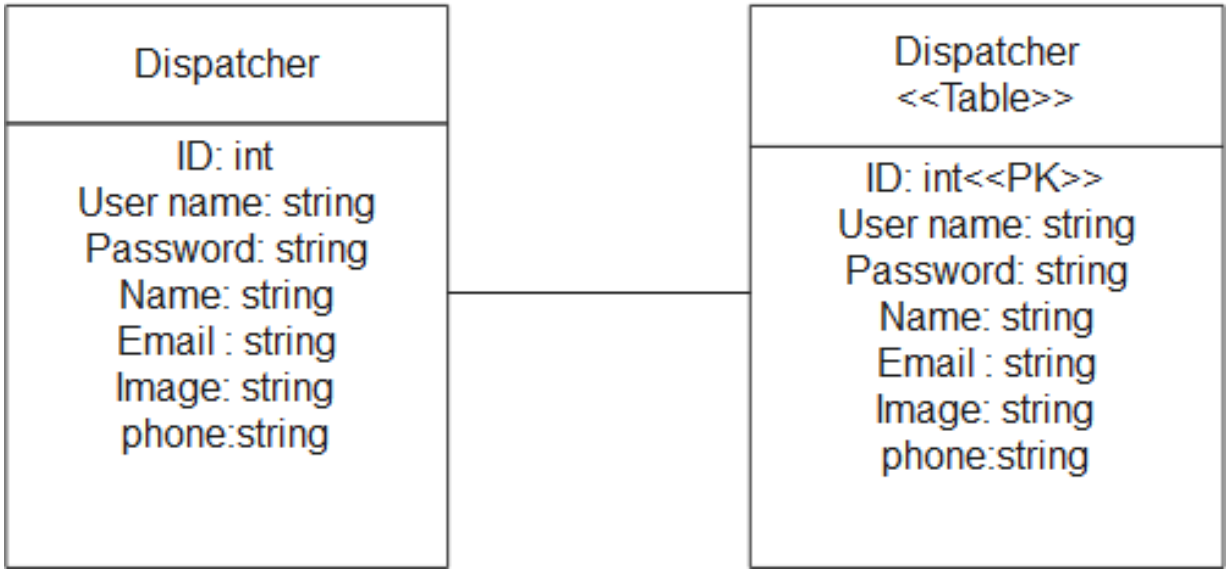


Figure 25: Persistence Diagram for Dispatcher

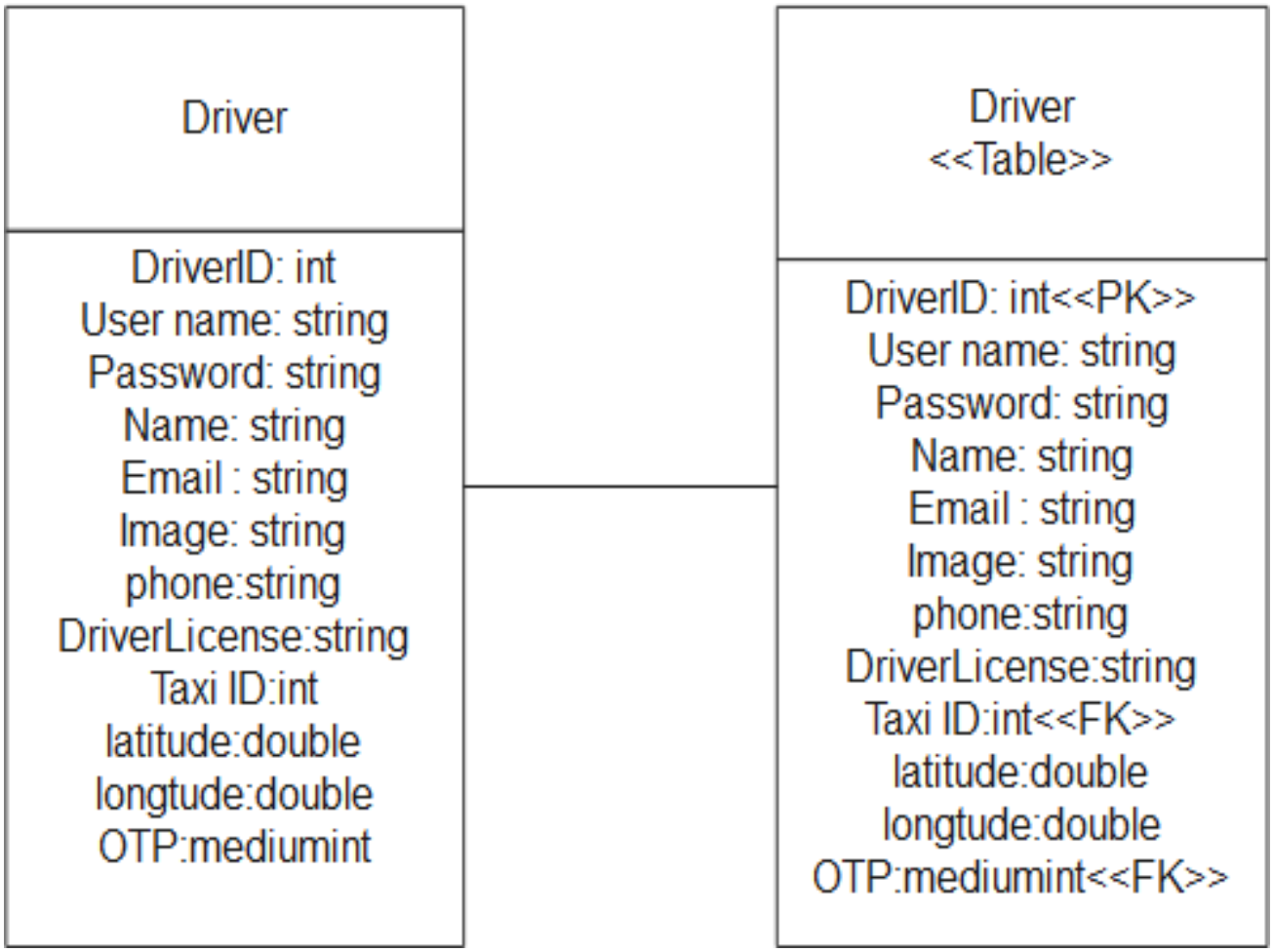


Figure 26: Persistence Diagram for Driver

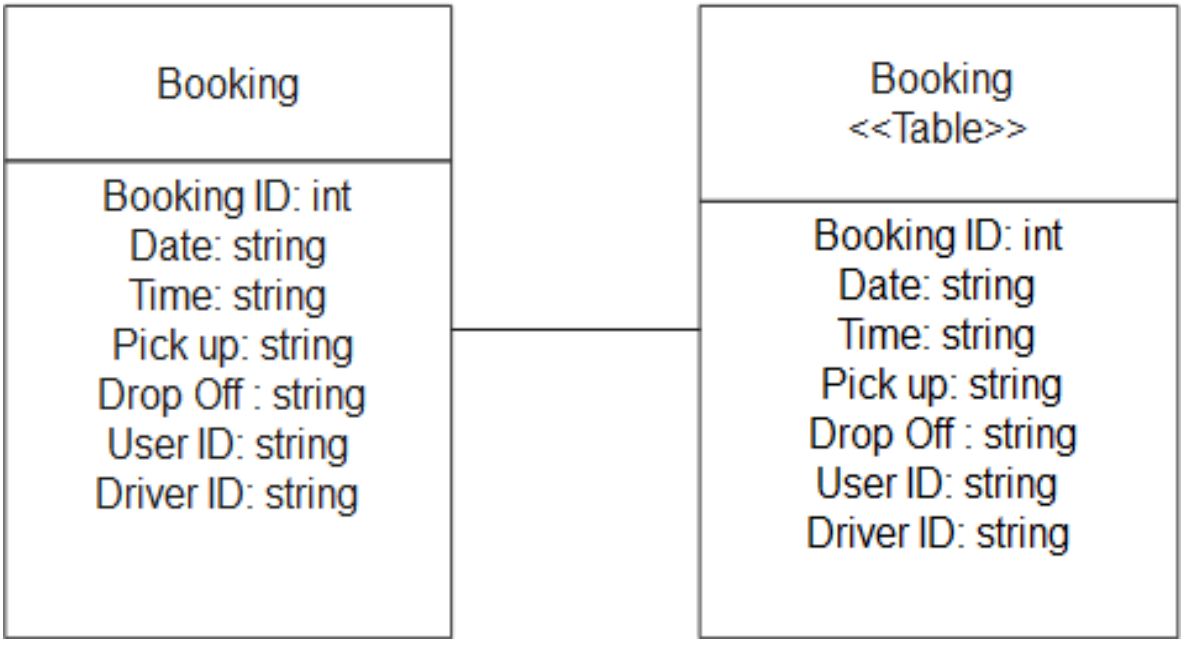


Figure 27: Persistence Diagram for Booking

5.3.5. ACCESS CONTROL AND SECURITY

In the systems, different actors have access to different functionality and data. The system has different Passengers such as passengers, administrator and driver.

The passengers: Represents unauthenticated Passenger. It is used by without Login subsystems to book taxi. And also they responsible to fill booking form, select, search booking.

Driver: Is an authenticate Passenger who is responsible to accept and reject booking, change taxi current status and cancel booking.

Administrator: Is an authenticate Passenger. It is used by login subsystem. He or she is responsible for managing drivers, taxi and control booking.

5.4. PACKAGES

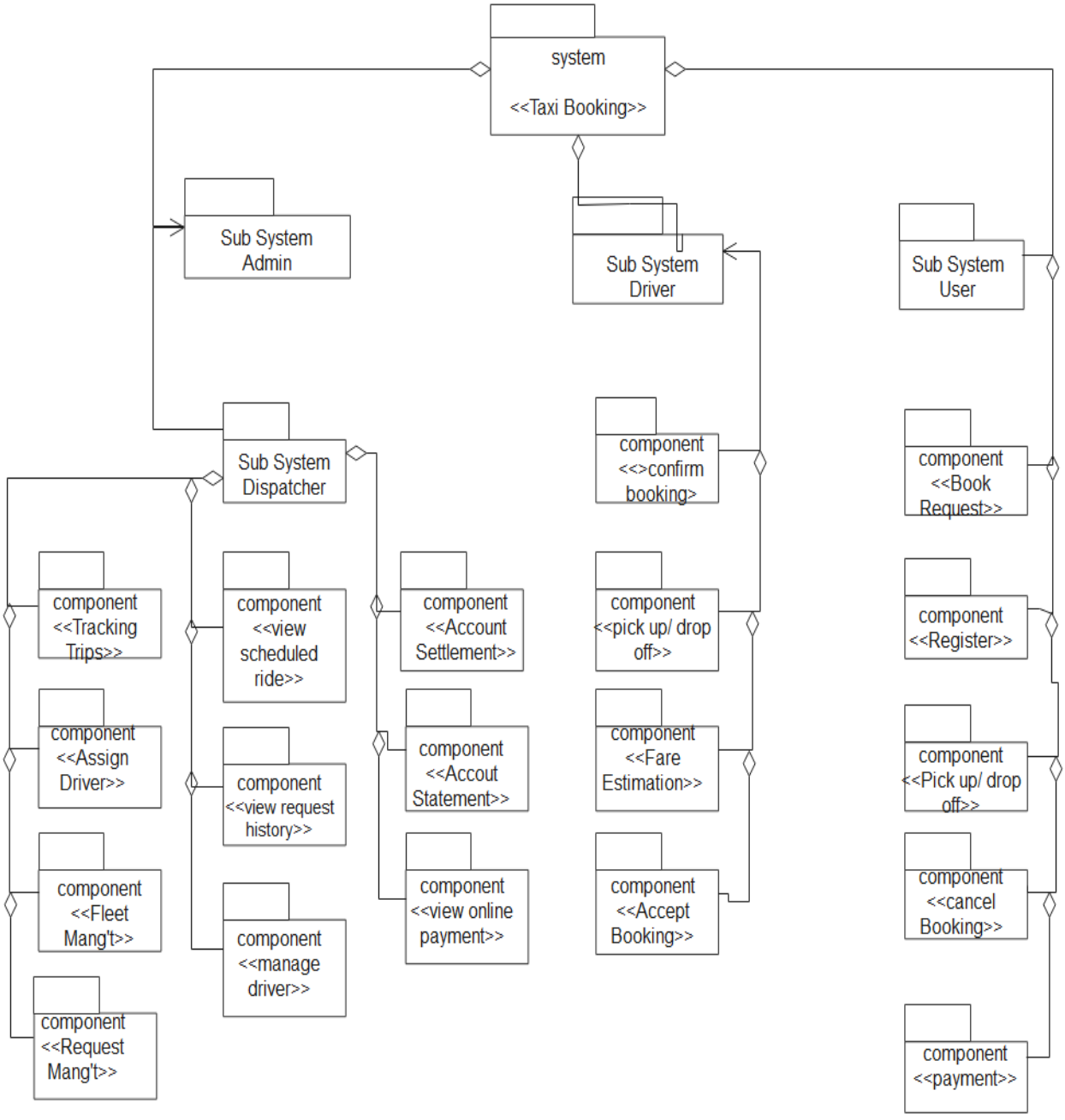


Figure 28: Package Diagram

5.5. PASSENGER INTERFACE DESIGN

Passenger interface design (UI) or Passenger interface engineering is the design of Passenger interfaces for machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing the Passenger experience. The

goal of Passenger interface design is to make the Passenger's interaction as simple and efficient as possible, in terms of accomplishing Passenger goals (Passenger-centered design) so in our project we have designed Passenger interfaces that increase the Passenger experience.

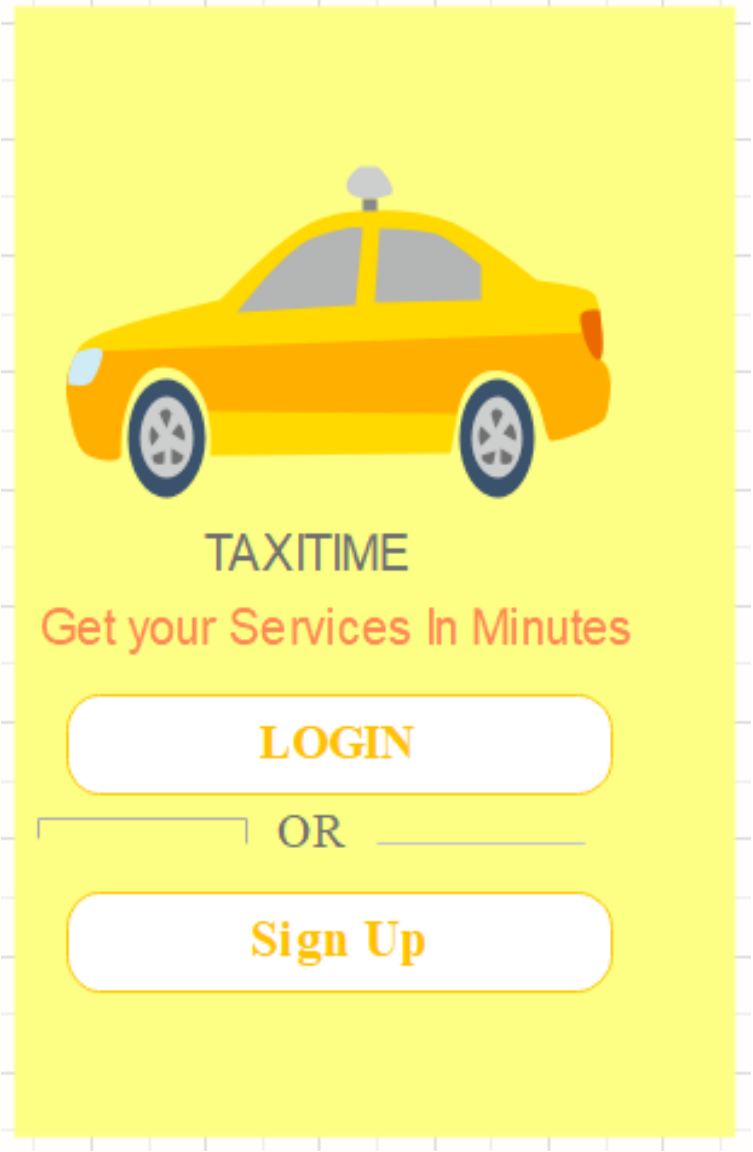


Figure 29: Passenger Homepage

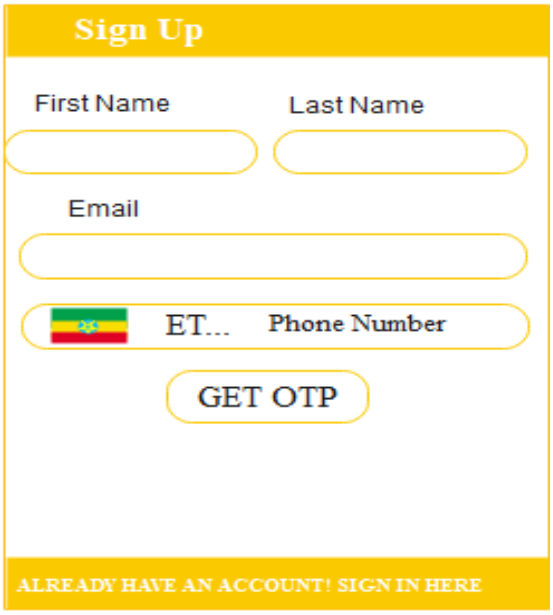


Figure 30: Passenger Sign up Page

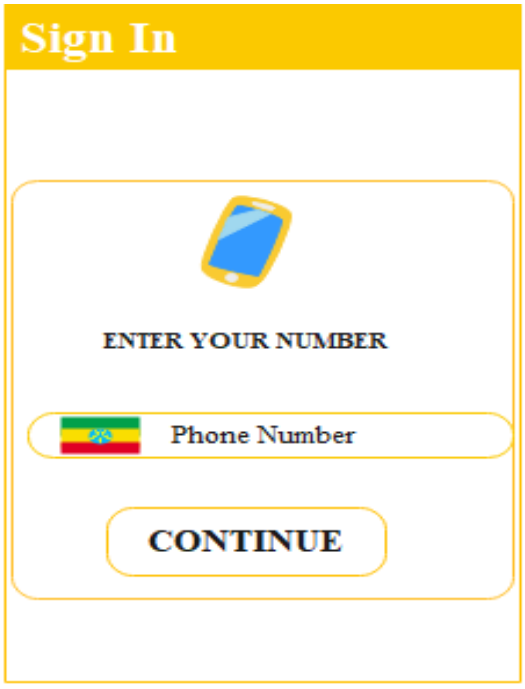


Figure 31: Passenger Sign in Page

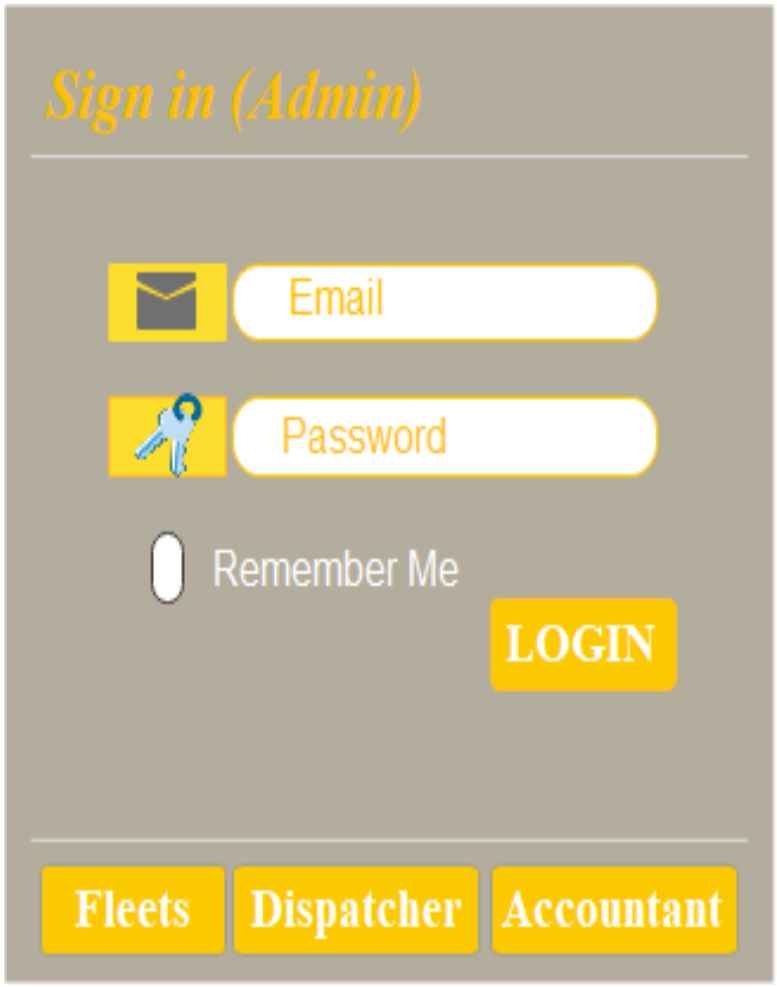


Figure 32: Admin Sign in Page

CHAPTER SIX

6. IMPLEMENTATION AND TESTING

During this phase physical design specification must be turned into working computer code, and provide help for current and future users and take care of the system. And then the code is tested until most of the errors have been detected and corrected. The purpose of this activity is to convert the final physical system specification into working model with reliable software and hardware.

6.1. Implementation of the Database

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time_zone = "+00:00";CREATE TABLE `accounts` (

  `id` int(10) UNSIGNED NOT NULL,

  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,

  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,

  `mobile` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,

  `password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,

  `remember_token` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,

  `created_at` timestamp NULL DEFAULT NULL,

  `updated_at` timestamp NULL DEFAULT NULL

) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO `accounts` (`id`, `name`, `email`, `mobile`, `password`, `remember_token`,
`created_at`, `updated_at`) VALUES

CREATE TABLE `admins` (
```

```
  `id` int(10) UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `picture` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `remember_token` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO `admins` (`id`, `name`, `email`, `password`, `picture`, `remember_token`,
`created_at`, `updated_at`) VALUES

CREATE TABLE `dispatchers` (
  `id` int(10) UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `mobile` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `remember_token` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE `favourite_locations` (
  `id` int(10) UNSIGNED NOT NULL,
  `user_id` int(11) NOT NULL,
```

```
  `address` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `latitude` double(15,8) DEFAULT NULL,
  `longitude` double(15,8) DEFAULT NULL,
  `type` enum('home','work','recent','others') COLLATE utf8_unicode_ci NOT NULL DEFAULT
'others',
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
CREATE TABLE `fleets` (
  `id` int(10) UNSIGNED NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `email` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `password` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `company` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `mobile` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `logo` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `remember_token` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT NULL,
  `updated_at` timestamp NULL DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

6.2. Implementation of the Class Diagram for the backend

```
@extends('admin.layout.base')
@section('title', 'Dashboard ')
@section('styles')
```

```
<link rel="stylesheet" href="plugins/fontawesome-free/css/all.min.css">

<!-- Theme style -->

<link rel="stylesheet" href="dist/css/adminlte.min.css">

<!-- Google Font: Source Sans Pro -->

<link href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700"
rel="stylesheet">

<link href="../mainindex/css/fontawesome-all.min.css" rel="stylesheet">

@endsection

@section('content')

<div class="content-area py-1">

<div class="container-fluid">

   <div class="row row-md">

                  <div class="col-lg-3 col-md-6 col-xs-12">

                        <div class="small-box bg-info">

                              <div class="inner">

                                    <h3>{{$rides->count()}}</h3>

                                    <p>Total Rides</p>                </div>

                              <div class="icon">

                                    <i class="ion ion-bag"></i>                </div>

                              <a href="/admin/requests" class="small-box-footer">History <i
class="fas fa-arrow-circle-right"></i></a>          </div></div>

                  <div class="col-lg-3 col-md-6 col-xs-12">

                  <div class="small-box bg-success">

         <div class="inner">
```

```
<h3>{{currency($revenue)}}</h3>

<p>Total Revenue</p></div>

<div class="icon">

<i class="ion ion-stats-bars"></i>       </div>

<a href="/admin/statement" class="small-box-footer">More info <i class="fas fa-arrow-
circle-right"></i></a>       </div></div>

                <div class="col-lg-3 col-md-6 col-xs-12">

                        <div class="small-box bg-warning">

<div class="inner">

<h3>{{$service}}</h3>

<p>All Services</p>       </div>

<div class="icon">

<i class="ion ion-person-add"></i>  </div>

<a href="/admin/service" class="small-box-footer">More info <i class="fas fa-arrow-circle-
right"></i></a>

</div>    </div>

                <div class="col-lg-3 col-md-6 col-xs-12">

                        <div class="small-box bg-danger">

<div class="inner">

<h3>{{$user_cancelled+$provider_cancelled}}</h3>

<p>Cancelled Ride</p>     </div>

<div class="icon">

<i class="ion ion-pie-graph"></i>       </div>
```

```
<a href="#" class="small-box-footer">More info <i class="fas fa-arrow-circle-
right"></i></a>
</div>              <div class="col-lg-3 col-md-6 col-xs-12">
                    <div class="small-box bg-danger">
 <div class="inner">
  <h3>{{$user_cancelled}}</h3>
  <p>User Cancelled Rides</p>            </div>
 <div class="icon">
                              <i class="ion ion-stats-bars"></i>     </div>
 <a href="#" class="small-box-footer">More info <i class="fas fa-arrow-circle-
right"></i></a>
</div>   </div>
          <div class="col-lg-3 col-md-6 col-xs-12">
                    <div class="small-box bg-primary">
 <div class="inner">
  <h3>{{$provider_cancelled}}</h3>
  <p>Provider Cancelled Ride</p>     </div>
 <div class="icon">
                              <i class="ion ion-stats-bars"></i>    </div>
 <a href="#" class="small-box-footer">More info <i class="fas fa-arrow-circle-
right"></i></a>
</div>   </div>
          <div class="col-lg-3 col-md-6 col-xs-12">
                    <div class="small-box bg-success">
```

```
<div class="inner">
<h3>{{$fleet}}</h3>
<p>Online Payment</p>    </div>
<div class="icon">
<i class="ion ion-pie-graph"></i>      </div>
<a href="#" class="small-box-footer">More info <i class="fas fa-arrow-circle-right"></i></a>
</div>   </div>
<div class="col-lg-3 col-md-6 col-xs-12">
<div class="small-box bg-info">
<div class="inner">
<h3>{{$scheduled_rides}}</h3>
<p>Scheduled Ride</p>   </div>
<div class="icon">
<i class="ion ion-pie-graph"></i>     </div>
<a href="admin/scheduled" class="small-box-footer">More info <i class="fas fa-arrow-circle-right"></i></a>
</div>   </div></div>
<div class="row row-md mb-2">
<div class="col-md-12">
<div class=" bg-white" style="box-shadow:none;">
<div class="box-block clearfix">
<h5 class="float-xs-left">Recent Rides</h5>   </div>
<table class="table mb-md-0 table " id="table-2"style="width:100%">
```

```
<thead>    <tr>
    <th>ID</th>
    <th>User </th>
    <th>Driver</th>
    <th>Ride Details</th>
    <th>Date &amp;amp; Time</th>
    <th>Total</th>
    <th>Status</th>   </tr>
</thead>  <tbody>
  <?php $diff = ['-success','-info','-warning','-danger']; ?>
  @foreach($rides as $index => $ride)    <tr>
    <th scope="row">{{$index + 1}}</th>
                                        <td>{{$ride->user['first_name']}} {{$ride-
>user['last_name']}}</td>   <td>
     {{@$ride->provider->first_name}}  {{@$ride->provider->last_name}}
    </td>   <td>
     @if($ride->status != "CANCELLED")
     <a class="text-primary" href="{{route('admin.requests.show',$ride->id)}}"><span
class="underline">Ride Details</span></a>
     @else
     <span>No Details Found </span>
     @endif                </td>   <td>
     <span class="text-muted">{{$ride->created_at->diffForHumans()}}</span>
    </td>   <td>
```

```
{{ currency($ride->payment['total'])}}   </td>      <td>
@if($ride->status == "COMPLETED")
<span class="tag tag-success">{{$ride->status}}</span>
@elseif($ride->status == "CANCELLED")
<span class="tag tag-danger">{{$ride->status}}</span>
@else
<span class="tag tag-info">{{$ride->status}}</span>
@endif      </td></tr>
<?php if($index==10) break; ?>
@endforeach
</tbody>
</table>  </div></div></div></div>
```

Implementation of the Class Diagram for the user app

```
package com.taxi.Taxi.Activities;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
```

```
import android.widget.ImageView;

import android.widget.LinearLayout;

import android.widget.TextView;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import com.taxi.Taxi.API.RetrofitClient;

import com.taxi.Taxi.Models.Chat;

import com.taxi.Taxi.Models.ChatResponse;

import com.taxi.Taxi.Models.Errorresponse;

import com.taxi.Taxi.R;

import java.util.List;

import retrofit2.Call;

import retrofit2.Callback;

import retrofit2.Response;

public class newlivechat extends AppCompatActivity {

    private RecyclerView recyclerView;

    private List<Chat> chatList;

    private ProgressDialog mProg;

    private Intent mainIntent;

    private String name, img, uid, pid ,booking_id;

    private int lastsize ;
```

```
private EditText chatedittext;

private ImageView sendbtn;

private ImageView btnchatsend;

private  ImageView backArrow;

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_newlivechat);

    mainIntent = getIntent();

    chatedittext = findViewById(R.id.chataddtext);

    btnchatsend = findViewById(R.id.btnchatsend);

    recyclerView = findViewById(R.id.chatrecycleview);

    recyclerView.setLayoutManager(new LinearLayoutManager(this));

    mProg = new ProgressDialog(this);

    mProg.setTitle(R.string.app_name);

    mProg.setMessage("Loading...");

    mProg.show();

    name = mainIntent.getStringExtra("name");

    uid = mainIntent.getStringExtra("uid");

    pid = mainIntent.getStringExtra("pid");

    booking_id = mainIntent.getStringExtra("booking_id");

    //img = mainIntent.getStringExtra("img");

    backArrow = findViewById(R.id.backArrow);
```

```
backArrow.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        onBackPressed();
    }     });
Call<ChatResponse> call = RetrofitClient.getInstance().getApi().getchat(booking_id);
call.enqueue(new Callback<ChatResponse>() {
    @Override
    public void onResponse(Call<ChatResponse> call, Response<ChatResponse> response) {
        chatList = response.body().getChat();
        chatadapter adaper = new chatadapter(newlivechat.this, chatList);
        recyclerView.setAdapter(adaper);
        mProg.dismiss();           }
    @Override
    public void onFailure(Call<ChatResponse> call, Throwable t) {
        mProg.dismiss();
        finish();          }       });
btnchatsend.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

if(chatedittext.getText().toString().trim().equals("")||chatedittext.getText().toString().equals(" "))
}else{
```

```
            Call<Errorresponse> call = RetrofitClient.getInstance().getApi().addchat(booking_id,
uid, pid, chatedittext.getText().toString().trim(), "up");
            call.enqueue(new Callback<Errorresponse>() {
                @Override
                public void onResponse(Call<Errorresponse> call, Response<Errorresponse>
response) {
                    if(!response.body().isError()){
                        chatedittext.setText("");
                        recyclerView.scrollToPosition(lastsize - 1);
                        refreshchat();                    }                }
                @Override
                public void onFailure(Call<Errorresponse> call, Throwable t) {                }
});      }       });
      chatedittext.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            recyclerView.scrollToPosition(lastsize - 1);         }       });
      Handler handleCheckStatus = new Handler();
      //check status every 5 sec
      handleCheckStatus.postDelayed(new Runnable() {
         @Override
         public void run() {
            refreshchat();
            handleCheckStatus.postDelayed(this, 5000);       }
```

```
    }, 5000);}
private void refreshchat() {
    Call<ChatResponse> call = RetrofitClient.getInstance().getApi().getchat(booking_id);
    call.enqueue(new Callback<ChatResponse>() {
        @Override
        public void onResponse(Call<ChatResponse> call, Response<ChatResponse> response) {
            chatList = response.body().getChat();
            if(lastsize != chatList.size()){
                chatadapter adaper = new chatadapter(newlivechat.this, chatList);
                recyclerView.setAdapter(adaper);}        }
        @Override
        public void onFailure(Call<ChatResponse> call, Throwable t) {
            mProg.dismiss();
            finish();        }        });    }
private class chatadapter extends RecyclerView.Adapter<chatadapter.chatholder>{
    private Context mCtx;
    private List<Chat> chats;
    public chatadapter(Context mCtx, List<Chat> chats) {
      this.mCtx = mCtx;
        this.chats = chats;}
    @NonNull
    @Override
    public chatholder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {
```

```
    View view = LayoutInflater.from(mCtx).inflate(R.layout.recycle_chat_live,viewGroup,false);
    return new chatadapter.chatholder(view); }
@Override
public void onBindViewHolder(@NonNull chatholder holder, int i) {
    Chat chat = chats.get(i);
    String type = chat.getType();
    if(type.equals("up")){
        holder.receiverlayout.setVisibility(View.VISIBLE);
        holder.chatmessage.setText(chat.getMessage());
    }else{
        holder.senderlayout.setVisibility(View.VISIBLE);
        holder.chatusermessage.setText(chat.getMessage());
        holder.chatusername.setText(name);          } }
@Override
public int getItemCount() {
    lastsize= chats.size();
    return  chats.size();       }
class chatholder extends RecyclerView.ViewHolder{
    ImageView chatuserimg;
    TextView chatusername, chatusermessage, chatmessage;
    LinearLayout senderlayout, receiverlayout;
    public chatholder(@NonNull View itemView) {
        super(itemView);
```

```
chatuserimg = itemView.findViewById(R.id.chatuserimg);

chatusername = itemView.findViewById(R.id.chatusername);

chatusermessage = itemView.findViewById(R.id.chatusermessage);

chatmessage= itemView.findViewById(R.id.chatmessage);

senderlayout = itemView.findViewById(R.id.senderlayout);

receiverlayout = itemView.findViewById(R.id.receiverlayout);          }}}}
```

Implementation of the Class Diagram for the driver app

```
package com.taxi.driver.Activity;

import android.app.Activity;

import android.content.Context;

import android.content.Intent;

import android.os.Bundle;

import com.google.android.material.snackbar.Snackbar;

import androidx.appcompat.app.AppCompatActivity;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.Toast;

import com.taxi.driver.DriverApplication;

import com.android.volley.AuthFailureError;

import com.android.volley.NetworkError;

import com.android.volley.NetworkResponse;
```

```
import com.android.volley.NoConnectionError;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.TimeoutError;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonObjectRequest;
import com.taxi.driver.Helper.ConnectionHelper;
import com.taxi.driver.Helper.CustomDialog;
import com.taxi.driver.Helper.SharedHelper;
import com.taxi.driver.Helper.URLHelper;
import com.taxi.driver.R;
import com.taxi.driver.Utilities.Utilities;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.HashMap;
import java.util.Map;
public class ChangePassword extends AppCompatActivity {
    String TAG = "ChangePasswordActivity";
    public Context context = ChangePassword.this;
    public Activity activity = ChangePassword.this;
    CustomDialog customDialog;
    ConnectionHelper helper;
    Boolean isInternet;
```

```
Button changePasswordBtn;
ImageView backArrow;
EditText current_password, new_password, confirm_new_password;
Utilities utils = new Utilities();
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_change_password);
    findViewByIdandInitialization();
    backArrow.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            onBackPressed();          }      });
    changePasswordBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String current_password_value = current_password.getText().toString();
            String new_password_value = new_password.getText().toString();
            String confirm_password_value = confirm_new_password.getText().toString();
            if(current_password_value == null || current_password_value.equalsIgnoreCase("")){
                displayMessage(getString(R.string.please_enter_current_pass));
            }else if(new_password_value == null || new_password_value.equalsIgnoreCase("")){
                displayMessage(getString(R.string.please_enter_new_pass));
```

```
            }else if(confirm_password_value == null ||
confirm_password_value.equalsIgnoreCase("")){
                displayMessage(getString(R.string.please_enter_confirm_pass));
            }else if(new_password_value.length() < 6 || new_password_value.length() > 16){
                displayMessage(getString(R.string.password_validation1));
            }else if(current_password_value.equalsIgnoreCase(new_password_value)){
                displayMessage(getString(R.string.new_password_validation));
            }else if(!new_password_value.equals(confirm_password_value)){
                displayMessage(getString(R.string.different_passwords));
            }else{
                if (helper.isConnectingToInternet()) {
                    changePassword();  } else {
                    displayMessage(getString(R.string.something_went_wrong_net));       }       }      }
}); }
    public void findViewByIdandInitialization(){
        current_password = (EditText)findViewById(R.id.current_password);
        new_password = (EditText)findViewById(R.id.new_password);
        confirm_new_password = (EditText) findViewById(R.id.confirm_password);
        changePasswordBtn = (Button) findViewById(R.id.changePasswordBtn);
        backArrow = (ImageView) findViewById(R.id.backArrow);
        helper = new ConnectionHelper(context);
        isInternet = helper.isConnectingToInternet();}
    private void changePassword() {
        customDialog = new CustomDialog(context);
```

```
customDialog.setCancelable(false);
customDialog.show();
JSONObject object = new JSONObject();
try {
    object.put("password", new_password.getText().toString());
    object.put("password_confirmation", confirm_new_password.getText().toString());
    object.put("password_old", current_password.getText().toString());
    utils.print("ChangePasswordAPI",""+object);
} catch (JSONException e) {
    e.printStackTrace();        }
JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST,
URLHelper.CHANGE_PASSWORD_API, object, new Response.Listener<JSONObject>() {
    @Override
    public void onResponse(JSONObject response) {
        customDialog.dismiss();
        utils.print("SignInResponse", response.toString());
        displayMessage(response.optString("message"));
        GoToBeginActivity();        }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        customDialog.dismiss();
        String json = null;
        String Message;
```

```
NetworkResponse response = error.networkResponse;
utils.print("MyTest",""+error);
utils.print("MyTestError",""+error.networkResponse);
utils.print("MyTestError1",""+response.statusCode);
if(response != null && response.data != null){
   try {
      JSONObject errorObj = new JSONObject(new String(response.data));
      utils.print("ErrorChangePasswordAPI",""+errorObj.toString());

      if(response.statusCode == 400 || response.statusCode == 405 || response.statusCode ==
500){
         try{
            displayMessage(errorObj.optString("error"));
         }catch (Exception e){
            displayMessage(getString(R.string.something_went_wrong));
         }
      }else if(response.statusCode == 401){
            GoToBeginActivity();
      }else if(response.statusCode == 422){
         json = DriverApplication.trimMessage(new String(response.data));
         if(json !="" && json != null) {
            displayMessage(json);
         }else{  displayMessage(getString(R.string.please_try_again));              }
      }else{ displayMessage(getString(R.string.please_try_again));  }
```

```
            }catch (Exception e){
                displayMessage(getString(R.string.something_went_wrong));}    } else {
            if (error instanceof NoConnectionError) {
                displayMessage(getString(R.string.oops_connect_your_internet));
            } else if (error instanceof NetworkError) {
                displayMessage(getString(R.string.oops_connect_your_internet));
            } else if (error instanceof TimeoutError) {
                changePassword();  }}  }}){
        @Override
        public Map<String, String> getHeaders() throws AuthFailureError {
            HashMap<String, String> headers = new HashMap<String, String>();
            headers.put("X-Requested-With", "XMLHttpRequest");
            headers.put("Authorization", "Bearer " + SharedHelper.getKey(context, "access_token"));
            return headers;  }};
    DriverApplication.getInstance().addToRequestQueue(jsonObjectRequest);}
  @Override
  public void onBackPressed() {
    super.onBackPressed();  }
  public void GoToBeginActivity(){
    SharedHelper.putKey(activity,"loggedIn",getString(R.string.False));
    Intent mainIntent = new Intent(activity, WelcomeScreenActivity.class);
    mainIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(mainIntent);
```

```
    activity.finish();   }
public void displayMessage(String toastString){
    utils.print("displayMessage",""+toastString);
    try {
        Snackbar.make(getCurrentFocus(), toastString, Snackbar.LENGTH_SHORT)
                .setAction("Action", null).show();
    }catch (Exception e){
        try{
            Toast.makeText(context,""+toastString,Toast.LENGTH_SHORT).show();
        }catch (Exception ee){
            ee.printStackTrace();         }       }    }}
```

6.3 Configuration of the Application Server

We use XAMPPSERVER application server because is simple and Lightweight. Apache distribution its extremely easy to create a local webserver for testing and deployment purposes. Everything we needed is to setup a web server-server application (Apache), MYSQL and scripting language (PHP). The configuration server follows the version and loaded extension etc.



Welcome to XAMPP for Windows 7.1.11

You have successfully installed XAMPP on this system! Now you can start using Apache, MariaDB, PHP and other components. You can find more info in the FAQs section or check the HOW-TO Guides for getting started with PHP applications.

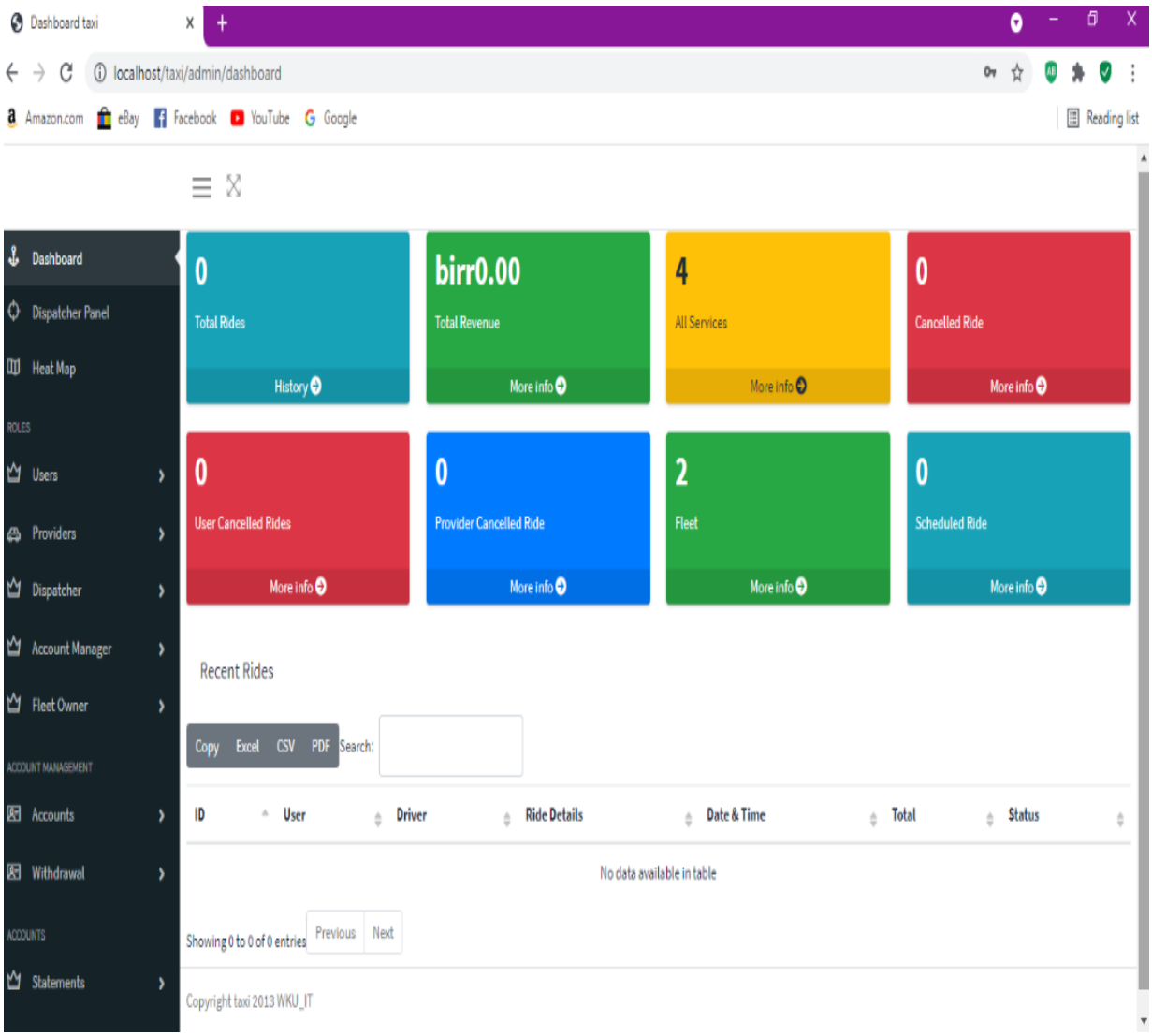
XAMPP is meant only for development purposes. It has certain configuration settings that make it easy to develop locally but that are insecure if you want to have your installation accessible to others. If you want have your XAMPP accessible from the internet, make sure you understand the implications and you checked the FAQs to learn how to protect your site. Alternatively you can use WAMP, MAMP or LAMP which are similar packages which are more suitable for production.

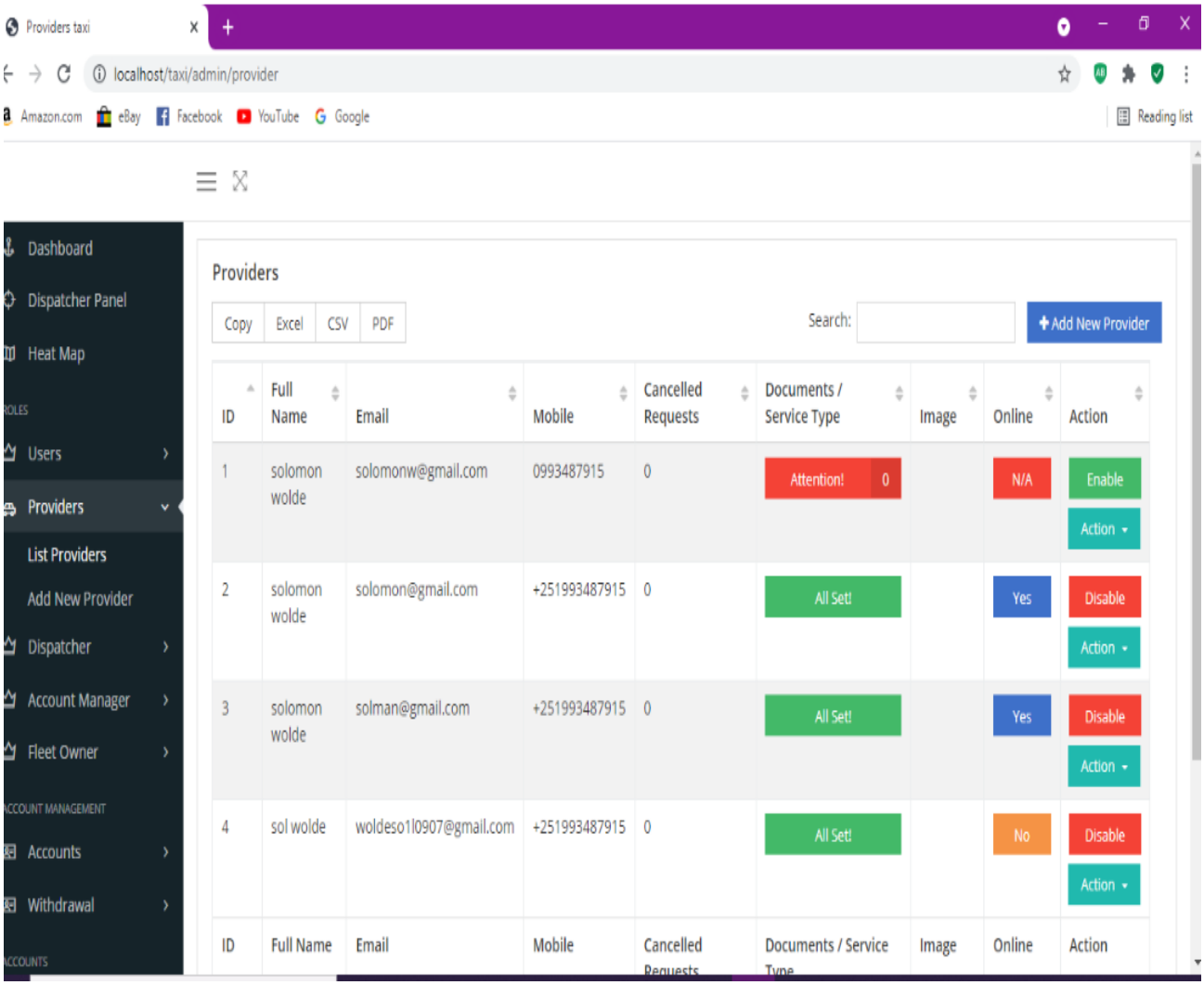
Start the XAMPP Control Panel to check the server status.

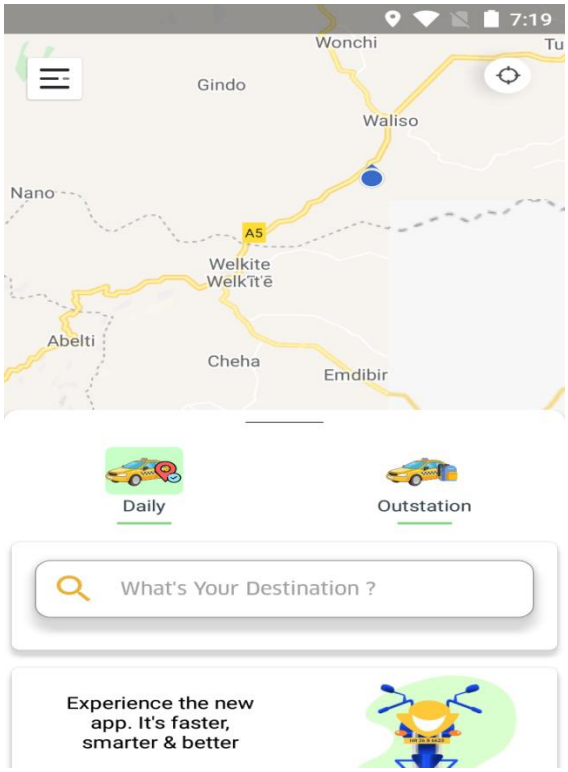
6.4. Configuration of Application Security

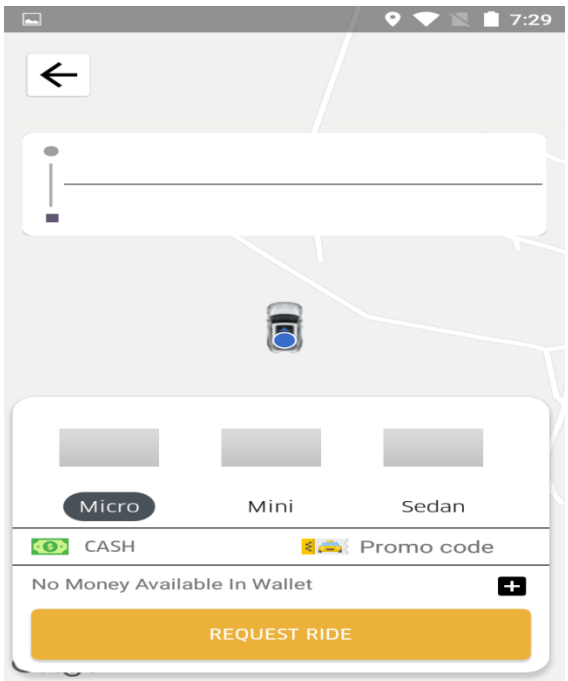
Our system validates all input by returning error message and suggesting to try again message when invalid message occur. We implement hash function for user password when the system admin creates a user accounts. The system has 13 roles and the roles are defined clearly and the user login to their specified page. In the system we implement session to store temporarily username and password of the users to login to the system to redirect to their specified page.

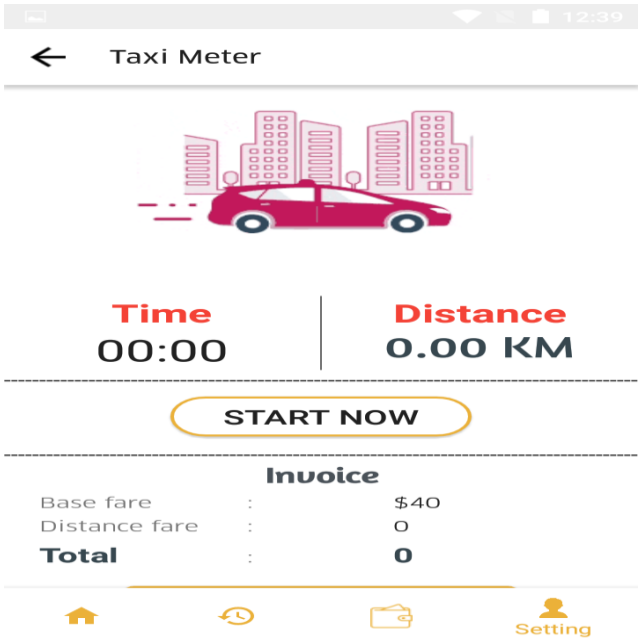
6.5. Implementation of User Interface

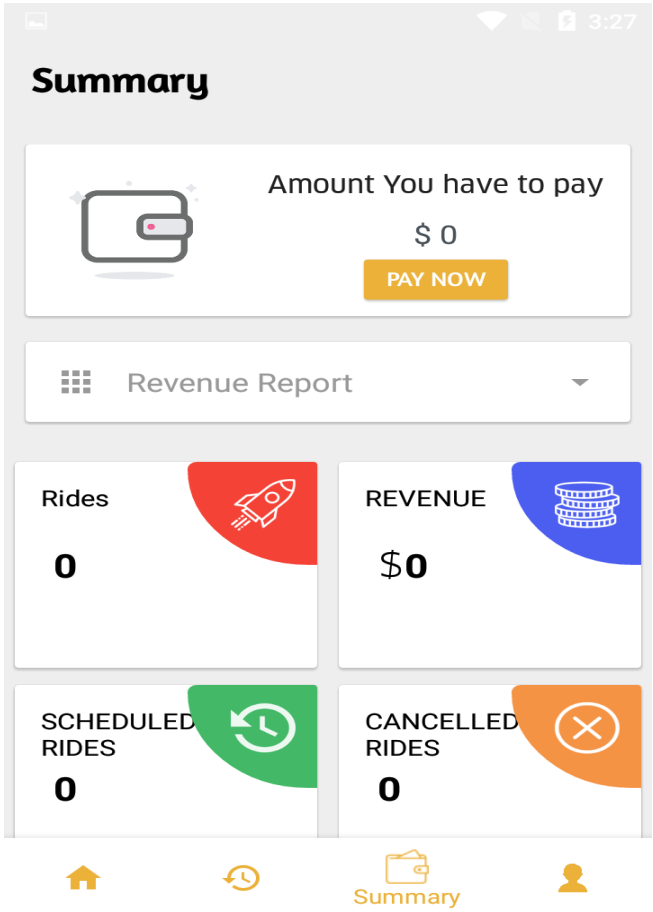


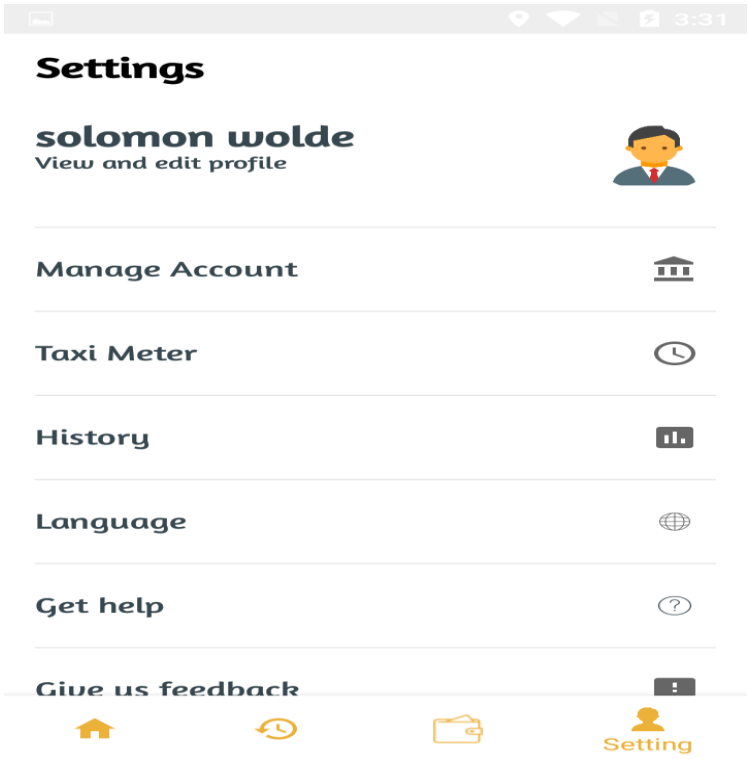

Providers taxi
localhost/taxi/admin/provider
Amazon.com
eBay
Facebook
YouTube
Google
Reading list
Dashboard
Dispatcher Panel
Heat Map
ROLES
Users
Providers
List Providers
Add New Provider
Dispatcher
Account Manager
Fleet Owner
ACCOUNT MANAGEMENT
Accounts
Withdrawal
ACCOUNTS
Providers
Copy
Excel
CSV
PDF
Search:
+ Add New Provider
ID
Full Name
Email
Mobile
Cancelled Requests
Documents / Service Type
Image
Online
Action
1
solomon wolde
solomonw@gmail.com
0993487915
0
Attention!
0
N/A
Enable
Action
2
solomon wolde
solomon@gmail.com
+251993487915
0
All Set!
Yes
Disable
Action
3
solomon wolde
solman@gmail.com
+251993487915
0
All Set!
Yes
Disable
Action
4
sol wolde
woldeso1l0907@gmail.com
+251993487915
0
All Set!
No
Disable
Action
ID
Full Name
Email
Mobile
Cancelled Requests
Documents / Service Type
Image
Online
Action

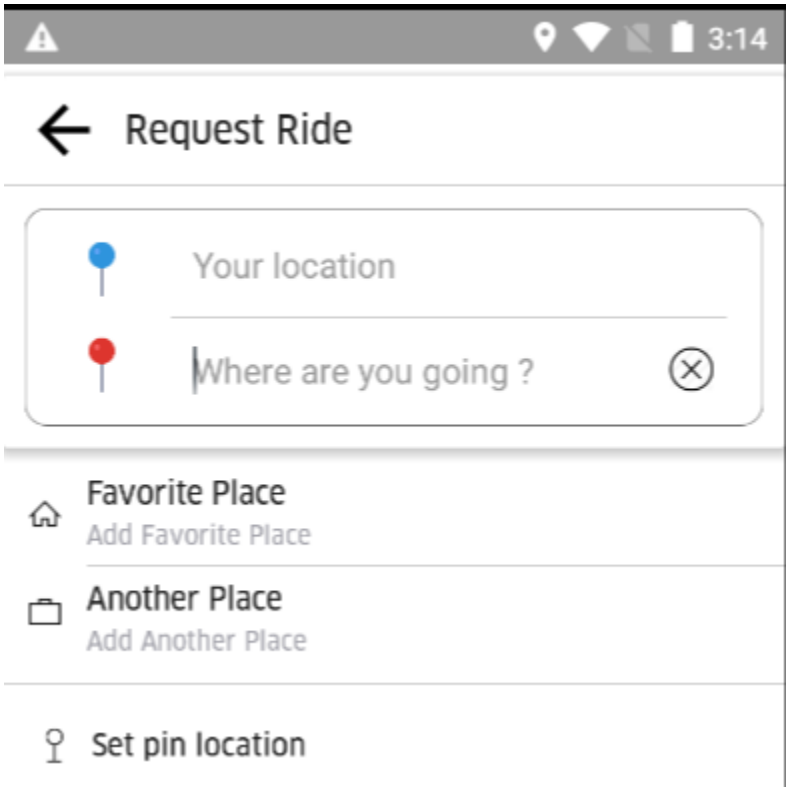

7:19
Wonchi
Tu
Gindo
Waliso
Nano
A5
Welkite
Welkīt'ē
Abelti
Cheha
Emdibir
Daily
Outstation
What's Your Destination ?
Experience the new app. It's faster, smarter & better


7:29
Micro
Mini
Sedan
CASH
Promo code
No Money Available In Wallet
REQUEST RIDE


12:39
Taxi Meter
Time
00:00
Distance
0.00 KM
START NOW
Invoice
Base fare : $40
Distance fare : 0
Total : 0
Setting


3:27
Summary
Amount You have to pay
$ 0
PAY NOW
Revenue Report
Rides
0
REVENUE
$0
SCHEDULED RIDES
0
CANCELLED RIDES
0
Summary


3:31
Settings
solomon wolde
View and edit profile
Manage Account
Taxi Meter
History
Language
Get help
Give us feedback
Setting


3:14
Request Ride
Your location
Where are you going ?
Favorite Place
Add Favorite Place
Another Place
Add Another Place
Set pin location

6.6 Testing

Testing is a process to show the correctness of the program and designed to analyze the logic used in the implementation of the System. In the case of our project we use unit, user acceptance, and integration testing method, which is briefly stated in proposal.

6.6.1 Unit testing

Each module is tested alone in an attempt to discover any errors in its code. In unit testing, each module (roughly a section of code that performs a single function) is tested alone in an attempt to discover any errors that may exist in the module's code. We tested the system as shown below in the appendix.

6.6.2 Integration testing

The process of bringing together for testing purposes all of the modules that a program comprises. Modules are typically integrated in a top-down, incremental fashion. If an error occurs, the process stops, the error is identified and corrected, and the test is redone. The process repeats until the entire program—all modules at all level is successfully integrated and tested with no errors. After the team tested using unit testing the next step is integrating testing. In integrated testing the team tested the system all modules that a program contains.

6.6.3 Acceptance testing

It is the process whereby actual users test a completed information system, the end result of which is the users' acceptance of it once they are satisfied with it. Acceptance refers to the fact that users typically sign off on the system and "accept" it once they are satisfied with it. Testing the system in the environment where it will be used. The purpose of acceptance testing is for users to determine whether the system meets their requirements. In acceptance test the teams test the system by different user that satisfies the requirements. The user tested by inserting real data.

6.6.4 System testing

System testing is simply expanded integration testing, where you are testing the interfaces between programs in a system rather than testing the interfaces between modules in a program. System testing is also intended to demonstrate whether a system meets its objectives. It is the final step of testing**.** In this step the team members tests the entire system as a whole with all forms, code, modules and tests all the functionalities in the System. This form of testing is popularly known as Black Box testing or System tests. All errors in the forms, functions, modules are tested.

CHAPTER SEVEN

Conclusions and Recommendations

7.1 Conclusions

In this project, we have developed a mobile based Taxi Booking application that facilitates various booking activities of people. The system developed in the project consists of mobile and web applications. These are two different applications on the same database. The mobile application enable user's activities like making reservation, edit booking and preview booking histories. The web application facilitates to handle managing driver information, control booking details, taxi information, service properties and so on. Our model contains analysis model which contains the Functional and Non-functional requirements, use case, sequence, state chart, activity diagrams, conceptual modeling of classes, and identifying user interface prototypes. And also contains designs model which consists subsystem structuring, component structuring, deployment model and the persistent modeling of database. The problems, which existed in the earlier banking approach, will be removed to a large extent. And it is expected that this project will go a long way in satisfying user's requirements.

7.2 Recommendations

According to scope of our project the team develops both web application and mobile application. Because of the time constraint we may have some limitations which should be taken in considerations, but in the future the team believes that this system can be fully operational by having some functionality that are not included in the proposed system like online payment for the service we gate.

During the development of this project the group members faced many challenges. like loss of internet access, limited time for doing the complete project and billing system of API providers. However, by the cooperation of all group members and our advisors the team is now able to reach to the final result through this way we try to construct the road and hence this project has a big role in Ethiopia. And for the future if we get enough solutions we can conduct all the functionalities of the project properly

According to this there are some sorts that can be done in the future to locate current technology features in our project. Like online payment of Ethiopian yene pay method, geocoding and some new features to make easy and simple.

References

[1] [Online]. Available: www.ride8294.com.

[2] [Online]. Available: www.feres.et.com.

[3] W. S.Saba, Object oriented analysise and design, 200.