WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF INFORMATION TECHNOLOGY

PROJECT TITLE: ONLINE VEHICLE MANAGEMNT SYSTEM

Prepared By:

Name	ID
1. Alehegn Admasu	CIR/209/11
2. Abebe Adam	CIR/206/11
3. Elias Abebaw	CIR/229/11

Advisor: Mr. Nigus A.

Wolkite University, Wolkite, Ethiopia

February 28, 2022

WOLKITE UNIVERSITY

COLLEGE OF COMPUTING AND INFORMATICS

DEPARTMENT OF INFORMATION TECHNOLOGY

PROJECT TITLE: ONLINE VEHICLE MANAGEMNT SYSTEM

SUBMITED TO DEPARTMENT OF INFORMATION TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHLER OF SCIENCE IN INFORMATION TECHNLOLGY

BY:

Student Name	ID Number
1. Alehegn Admasu	CIR/209/11
2. Abebe Adam	CIR/206/11
3. Elias Abebaw	CIR/229/11

Advisor: Mr. Nigus A.

Wolkite University, Wolkite, Ethiopia

Feb 23, 2014 E.C

DECLARATION

This is to declare that this project work which is done under the supervision of Mr. Nigus Asres and having the title Online Vehicle Management System with the sole contribution of:

1. Alehegn Admasu
2. Abebe Adam
3. Elias Abebaw

No part of the project work has been reproduced illegally (copy and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. We will be responsible and liable for any consequence if the violation of this declaration is proven.

Date:-_____

Group Members

Full Name Signature

1._____ _____

2._____ _____

3._____ _____

APPROVAL FORM

This is to confirm that the project report entitled Web Based Vehicle Management System submitted to Wolkite University, College of Computing and Informatics Department of Information Technology by: Alehegn Admasu, Abebe Adam and Elias Abebaw are approved for submission.

Advisor Name	Signature	Date
Department Head Name	Signature	Date
Examiner 1 Name	Signature	Date
Examiner 2 Name	Signature	Date
Examiner 3 Name	Signature	Date

ACKNOWLEDGEMENT

First of all, the project team member would like to thanks to God and after that our sincerely appreciation goes to our advisor MR. Nigus Asres for his care full super vision and advise. After our advisor we would like to thank you for our department for their contribution either by material resource or by other knowledge-based contribution. And also, we want to thank you to the persons who help us in different activities.

Contents

Contents

PROJECT TITLE: ONLINE VEHICLE MANAGEMNT SYSTEM....1

PROJECT TITLE: ONLINE VEHICLE MANAGEMNT SYSTEM 2

DECLARATION....3

Group Members....3

APPROVAL FORM....4

ACKNOWLEDGEMENT....3

Contents....4

LIST OF TABLES 9

LIST OF FIGURES 10

LIST OF ABBREVIATIONS 10

CHAPTER ONE....1

1.1. Background of VMS ***1***

1.2. Statement of Problem....2

1.3. Objectives....2

1.3.1. General objective....2

1.3.2. Specific objectives....2

1.4. Feasibility study....3

1.4.1. Operational Feasibility 3

1.4.2. Technical feasibility 3

1.4.3. Schedule Feasibility 4

1.4.4. Economic feasibility....4

1.4.5. Political feasibility....4

1.5. Scope of the project and Limitation of the project....4

1.5.1. Scope of the project....4

1.5.2. Limitation of the project....4

1.6. Significance of the system....4

1.7. Beneficiary of the Project *5*

1.8. Methodology *5*

1.8.1. Data collection 6

1.8.2. System requirement tools Hardware Requirements 6

Software Requirements 6

1.8.3. System development model 7

1.8.4. System analysis and design 7

1.8.5. Testing Procedure 8

Unit Testing 8

Integration Testing 8

System Testing 8

CHAPTER TWO 9

2.1. Introduction of Existing System *9*

2.2. Users of Existing System *9*

2.3. The major function of the existing system *9*

2.4. Drawbacks of the existing system *10*

2.5. Business Rules of the Existing System *10*

CHAPTER THREE 12

3.1. Introduction *12*

3.2. Functional Requirement *12*

3.3. Non-Functional Requirements *13*

3.3.1. User Interface and Human Factors 14

3.3.2. Hardware Consideration 14

3.3.3. Security Issue 14

3.3.4. Performance Consideration 14

3.3.5. Error Handling and Validation 14

3.3.6. Quality Issues Usability: - 15

3.3.7. Backup and Recovery 15

3.3.8. Physical Environment 15

3.3.9. Resource Issues 15

3.3.10. Documentation 16

CHAPTER FOUR 17

4.1. System Model 17

4.1.1. Use Case Model Actor specification 17

4.1.1.1. Use case Diagram 18

4.1.1.2. Descriptions of use case diagram 19

4.1.1.3. Use case Scenario 27

4.2. Object Model 28

4.2.1. Class Diagram 29

4.2.2. Data Dictionary 29

4.3. Dynamic Model 34

4.3.1. Sequence diagram 34

4.3.2. Activity diagram 39

4.3.3. State chart diagrams 46

CHAPTER FIVE 54

5.1. Introduction 54

5.2. Design Goals 54

User Interface and Human Factor 54

Hardware Consideration 55

Security 55

Performance Consideration 55

Error Handling and Validation 55

Quality Issues 56

Storage capability 56

Backup and Recovery 56

Physical Environment 56

Resource Issues 56

Documentation 56

5.3. Architecture of the proposed System 57

Application layer 57

Logic layer 57
Database layer 57
5.4. Subsystem Decomposition and Description ***57***
5.5. Hardware/Software Mapping ***59***
5.6. Detailed Class Diagram ***60***
5.7. Persistent Data Management ***61***
5.8. Access Control and Security ***62***
5.9. Packages ***64***
5.10. Algorithm Design ***65***
Algorithm for login: 65
Algorithm for registration 65
Algorithms for add user: 65
Algorithms for search: 65
5.11. User Interface Design ***66***
CHAPTER SIX **68**
6.1 Implementation of the Database ***68***
6.2. Implementation of the Class Diagram ***71***
6.3. Configuration of Application Server ***72***
6.4. Configuration of Application Security ***73***
6.5. Implementation of User Interface ***76***
6.6. Testing ***78***
6.6.1. Test case 78
6.6.2. Testing Tools and Environment 78
6.6.3. Unit Testing 79
6.6.4. Integration Testing 80
6.6.5. System Testing 84
CHAPTER – SEVEN **86**
7. CONCLUSION AND RECOMMENDATION ***86***

7.1. Conclusion *86*

7.2. Recommendation *86*

References **87**

APPENDICES **88**

APPENDIX I: Sample Source Code ***88***

LIST OF TABLES

Table 4.1: Use Case Description for Login *19*
Table 4.2: Use Case Description for Create Account *20*
Table 4.3: Use Case Description for Rent Vehicle *21*
Table 4.4: Use Case Description for Add Vehicle *22*
Table 4.5: Use Case Description for Feedback *23*
Table 4.6: Use Case Description for Generate Report *24*
Table 4.7: Use Case Description for Remark Account *25*
Table 4.8: Use Case Description for Update Vehicle *25*
Table 4.9: Use Case Description for Search Vehicle *26*
Table 4.10: Use Case Description for Payment *27*
Table 4.11: Use Case Description for Sell Vehicle *28*
Table 4.12: Data Dictionary for Admin *31*
*Table 4.13**:** Data Dictionary for **C**ustomer* *32*
Table 4.14: Data Dictionary for Vehicle *33*
Table 4.15: Data Dictionary for Manager *34*
Table 4.16: Data Dictionary for Dealer *34*
Table 4.17: Data Dictionary for Report *35*
Table 5.1: Privilege Given to the Actor of the System *65*
Table 6.1: Configuration of Class Diagram *72*

LIST OF FIGURES

Figure 4.1: Use Case Diagram *18*
Figure 4.2: Conceptual Class Diagram *30*
Figure 4.3: Sequence Diagram for Login *36*
Figure 4.4: Sequence Diagram for Vehicle Registration *37*
Figure 4.5: Sequence Diagram Customer Registration *38*
Figure 4.6: Sequence Diagram for Search Vehicle *39*
Figure 4.7: Sequence Diagram for Update Vehicle *40*
Figure 4.8: Activity Diagram for Login *41*
Figure 4.9: Activity Diagram for Create Account Of Users *42*
Figure 4.10: Activity Diagram for Rent Vehicle *43*
Figure 4.11: Activity Diagram for Update Vehicle *44*
Figure 4.12: Activity Diagram for Buy Vehicle *45*
Figure 4.13: Activity Diagram for Search Vehicle *46*
Figure 4.14: Activity Diagram for Remark Vehicle *47*
Figure 4.15: Activity Diagram for Write Report *48*
Figure 4.16: State Chart Diagram for Login *49*
Figure 4.17: State Diagram for Create User Account *50*
Figure 4.18: State Diagram for Add Vehicle *51*
Figure 4.19: State Diagram for Update Vehicle *52*
Figure 4.20: State Diagram for Rent Vehicle *53*
Figure 4.21: State Diagram for Remark Vehicle *54*
Figure 4.22: State Diagram for Search Vehicle *55*
Figure 5.1: System Architecture *59*
Figure 5.2: Subsystem Decomposition *61*
Figure 5.3: Deployment Design *62*
Figure 5.4: Detail Class Diagram *63*
Figure 5.5: Persistent Data Management *64*
Figure 5.6: Package Diagram *67*
Figure, For Homepage *69*
Figure 5.8: GUI for User Login *70*

Figure 6.1: Configuration of Application Server *74*
Figure 6.2: User interface for Home Page *77*
Figure 6.2: User interface for login page *77*
Figure 6.3: User Interface for Admin page *78*
Figure 6.5: User Interface for User Registration form *79*

LIST OF ABBREVIATIONS

CD ..... Compact Disk

CSS..... Cascading Style Sheets

GUI..... Graphical User Interface

HTML ..... Hypertext Markup Language

JS..... Java Script

ODBC..... Object Database Connectivity

PC..... Personal Computer

PHP ..... Hypertext Preprocessor

SQL ..... Structured Query Language

VMS ..... Vehicle Management System

BR ..... Business Rule

ABSTRACT

An online vehicle management system allows a person to reserve a vehicle with/without payment on one end while the company staff handles the transactions, on the other via the internet. Online vehicle management system is a business used the technologies available to expand and provide more facilities to their customers. The basic functions of an online vehicle management system are to keep tracks of vehicles, staff and customers. it provides useful information to the staff such as giving daily reports of vehicles to be delivered/picked up and acts as a vehicle management system by monitoring the use and price of the vehicles.

In the current vehicle management system, information exchange and service control is processed in manual way. There are various problems on Vehicle service providers due to the file manual handling of its daily activity. This project will intend to advocate for the need of Vehicle owner and vehicle users to change the manual system to automated system vehicle management system. Vehicle owner use the system as a work place for searching purchasing and tracking of vehicles for customers. Customers can log in to the system remotely and access the data.

CHAPTER ONE

1. INTRODUCTION

Currently, the vehicle service gives using manual way of information gathering like vehicle‟s brand, vehicles address, vehicles price, and documenting the information so it takes time, cost, energy and no reliable communication between Vehicle Owner and vehicle users.

A web based vehicle management system can offer more advantages. It can make the interaction between the user and vehicle owner. Users can access the system “anytime” and from “anywhere” to make, change or cancel a reservation while management can have a better understanding in knowing stand by vehicle inventory at a specified time. It also makes the system device and platform independent.

This project will intend to advocate for the need of vehicle owner and vehicle users to change the manual system to automated system vehicle management system. Vehicle owner use the system as a work place for searching purchasing and tracking of vehicles for customers. Customers can log in to the system remotely and access the data.

Automated system make it possible to have a better accuracy, to increase the quality of the work, to reduce the time it takes, to minimize cost, to keep the security of data in most advantageous condition, to make data transfer easier and also make it possible to save and back up all transactions in case of vehicle management to keep the data in a centralized way which is available to all the users simultaneously.

1.1. Background of VMS

Vehicle Management Systems (VMS) have become increasingly important over time in space missions, due both to the demands for increased flexibility and capability of these missions, and the supply of increasingly capable computing systems to provide this improved functionality. VMSs include the management of uncertainties in vehicle state. The increasing complexity of the tasks that space systems are asked to accomplish, and the software and operational procedures necessary to accomplish them, have made VMSs a necessity for exploration missions. This paper investigates the underlying needs and functionality of Vehicle Management System, so as to better understand, and ultimately to better design them. To do this, we shall draw upon ideas

from information theory and system health management theory. Since VMSs necessarily use information to manage complex systems, information theory provide important insights.

1.2. Statement of Problem

In the current vehicle management system, information exchange and service control is processed in manual way. There are various problems on Vehicle service providers due to the file manual handling of its daily activity.

As we able to understand from our survey the following are discovered as major problems that vehicle service providers to provide quality services to its customers:-

- Recording user‟s information is very tedious.
- Controlling (changing) user information is difficult.
- Recording vehicle information is laborious.
- Customers have difficulty to know the vehicle owner address.
- Vehicle owner advertise their service manually like speaking literally, it consumes long time.
- The existing system has no automated database, Lack of security to store files.
- There is a lack of data organization in current existing system.
- Data or files loss due to improper handling.
- To advertise the vehicles, vehicle owners post manual in case they spend the paper and spill, cola and etc.
- It takes long time to generate a reports and the report may also inaccurate.
- There are so many phone calls so there is workload on manager and other employees.

1.3. Objectives

1.3.1. General objective

The main objective of this project is to develop a web based Vehicle Management System.

1.3.2. Specific objectives

To achieve the general objective the following specific objectives are drawn:-

- To review and identify the problems of the existing system.

- To formulate solution for the existing problems that had listed from statement of problems.
- To perform requirement analysis.
- To analyses the system that we proposed from gathering information.
- To design and implement the system.
- To test the developed system.

1.4. Feasibility study

Feasibility study is essential to evaluate the cost and benefits of the new system. On the basis of the feasibility study decision is taken on whether to proceed or to cancel the proposed project

Need for feasibility study:

- Determines the potential of the existing system.
- It is used to determine the problem of the existing system.
- To determine all goals of the new system.
- It finds all possible solutions of the problems of the existing system.

1.4.1. Operational Feasibility

The system will perform all operations to achieve the specified objective, User friendly and interactive with the environment and the system will perform all operation that the organization runs, and it will not have any difficulty or procedures to perform the operation of the system, so our system will be operationally feasible.

1.4.2. Technical feasibility

The system can be easily maintained and repaired without high experts or technical assistants, because the system will develop by familiar programming language or framework. The project team members have learned programming languages that required for the successful completion of the project such as java script, CSS, HTML, PHP, and MySQL, so that the project will be technically feasible.

1.4.3. Schedule Feasibility

The system after development gives efficient and effective services in short period of time. And also, the tasks may be scheduled for effective use of the system. The project will be finished at the schedule time. So, the project is feasible

1.4.4. Economic feasibility

The system uses new technology and have centralized database that cannot need more resources. It requires minimum amount of cost.

1.4.5. Political feasibility

The system we will develop does not conflict with any government directives, because it gives services for the passengers. So the government is profitable and the system will be politically feasible.

1.5. Scope of the project and Limitation of the project

1.5.1. Scope of the project

The new system will be expected to implement the following basic operations.

- Vehicle registration
- Buying and Selling vehicles
- Rent Vehicles
- Deal vehicles
- Advertise vehicles
- View vehicles information
- Searching and updating registered vehicles.
- Searching and updating registered user.
- Enabling the users to submit the required information in computerized way.

1.5.2. Limitation of the project

- The proposed system does not use GPS to control vehicles because of lack of enough material and short of time.

1.6. Significance of the system

After development of the new system: it will give the following advantages.

- The system used for vehicle owners as a work place for searching and purchasing of vehicles for customers.
- Users can log in to the system remotely and access the data.
- Reduce customer‟s cost, the work load and time because they can get vehicles full information without being physically present.
- Help the vehicle owner to easily give services to their customers.
- Increase Efficiency, Reduce manual handling.
- Easy to manage vehicles data in secure manner.
- Avoid improper resource consumption and data loss because the data is put in the database.
- Easy to select vehicles those customers want to do.
- Have good authentication and security.

1.7. Beneficiary of the Project

- Vehicle owners will be benefited from this project in the following points
  - ✓ The system will help to manage vehicles in efficient manner.
  - ✓ The system will help to advertising, selling, renting easily.
  - ✓ Data/files related to vehicles will be stored on centralized database and become secured when this system is implemented.
- Customers will be benefited from this project in the following points
  - ✓ Customers will log in to the system remotely and can access the data about vehicle.
  - ✓ The customer will benefit from the system to get on demand service.
- Group beneficiaries
- The project has initiated our team to get knowledge of how to develop the required software system application.
- While struggling with some difficulties, the team will get a lot of experiences of solving problems through the principles of software development.

1.8. Methodology

1.8.1. Data collection

- Interview

We Interview the vehicle owners to know about the existing system and we get some information which is important to do our project.

- Document analysis

To get more secondary source information and ideas about the vehicle management systems, we referred some documents about vehicle management system that are helpful to develop this system.

- Practical observation: -

We observed all the activities that are performing and notice down how they did. It helps us to get real information how the service providers performs its function and this helps to strength the data that gathered through interview and document analysis.

1.8.2. System requirement tools

Hardware Requirements

- Laptop Computer: To develop the documentation as well as the implementation.
- Flash: Used as move data from one to other.
- CD: Used as a backup.
- Printer: For printing the documentation.
- Paper : To store the documents in the form of hard copy

Software Requirements

- Modeling software
  - ✓ Draw.io: Used to draw diagrams like use case diagram and activity diagram.
- Designing Software
  - ✓ SQL server: This software will used for designing Database.
- Implementation Software
  - ✓ Windows 10: To have good speed of operations or execution time of tasks.
  - ✓ MS Word: document editing software
  - ✓ Sublime text 3 to write PHP scripts, html 5

1.8.3. System development model

Software Process

- Our system software follows Iterative development model. The reasons that we follow iterative development model is, to develop a system through building small portions of all the features, across all components. In each increment, a slice of system features is delivered, passing through the requirements till the deployment.

Programming Language

In our proposed system we use the following programming language: -

- PHP: -because it is a popular and widely used programming language which is utilized to build dynamic web applications with SQL database connections. Because it is user Friendly and Easy to Learn, Cost user Friendly, it‟s fast and easy, it accesses everything etc...
- Our system uses SQL software of the data base system and PHP language to develop and managing the back end of the system.
- The user interface of our system develops by using html, java script, CSS since it easily designing the front end and connected in to database easily.

1.8.4. System analysis and design

In our system, we use the object-oriented approach, the reasons that we use the object-oriented approaches are:

- To simplify the design and implementation of complex program
- We can inherit properties of the class that are defined in the super class.
- We can reuse methods for avoiding redundancy.
- To make it easier for teams of designers and programmers to work in a single software project
- The data and functions are encapsulated in the objects that help us for easily debugging purpose.
- It increases consistency among analysis, design and programming activities.
- It improves communication among users, analysis, design and programming
- It enables us to comprehensively model a system before we develop it.
- Modification of the object implementation is easy because objects are loosely coupled.

- Understanding of the structure is easy because Object oriented modeling represents real world entities.
- Direct manipulation of architectural components is possible because several Object oriented programming languages exist.

1.8.5. Testing Procedure

To simplify the testing process, our project team follows 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.

Unit Testing

In this level of testing process, users test the different sub procedures, functions and tested by applying the black and white box testing.

Sample Tests

- Check whether the return type of the functions is correct.
- Check how the sub procedures or functions are called correctly.
- Check if the correct output is produced for different inputs.
- Check the efficiency of the code with respect to the memory and CPU time.

Integration Testing

In this level of testing we will examine how the different procedures work together to achieve the goal of the sub system.

System Testing

In this level of testing process, we will examine how nicely the subsystems of the whole members work together to achieve the desired goal.

CHAPTER TWO

2. DESCRIPTION OF THE EXISTING SYSTEM

2.1. Introduction of Existing System

Here we are going to discuss the existing systems and other drawbacks in the manual earlier system of the vehicles management. The current system we are using i.e. manual system of the management of the vehicles is very tiresome and time-consuming. It takes a lot of time and going there in place, lining up in queues. Although many online portals have come into the picture for providing online vehicle management services. But, now a day, most of the vehicle owners use the traditional way to deal, rent and sell vehicles to the customer. Those are time and labor-consuming. An existing system can provide manual paperwork or an excel sheet to track registered vehicle details.

In the existing system, you cannot provide feedback of the user to the admin easily. Maintaining an excel sheet or paper book record of the reservation is very laborious work and chances of error are more. No automation involves which means they are very slow to process.

In the existing System it is difficult to maintain the vehicle information individually and to supply for the customers who are eager to buy them. Customer has to face difficulty in order to know the information of vehicle like manufacturing year, vehicle model, vehicle type and other valuable information in a single domain.

2.2. Users of Existing System

- The vehicle owner: advertise their service manually like speaking literally and state the rental price indicate their location by posting manually and state selling price deal vehicles and state the wholesale price.
- Manager: get job opportunity by manage vehicle.
- Customers: view vehicles and get services based their needs.

2.3. The major function of the existing system

- The customer makes decision by looking vehicles physically.
- Customer rent, buy vehicle by going directly.

- Vehicle owner sell his/her vehicles.
- Vehicle owner deal his/her vehicles by manual process.
- Customers can get service like rent, buy…by go to the location of the vehicle owner.
- During renting a vehicle the customer personal information, payments status and hire agreements are filled in the vehicle rent agreement form in order to hold legal contract between the customer and the owner for renting the vehicle.
- Posting of news, events, and notice
  - ✓ Vehicle owner post the notice paper to advertise he‟s vehicle services.
- Registration of new customer
  - ✓ First post the notice paper to aware the customers about vehicle services.
  - ✓ Then customers come to the place of vehicle services.
  - ✓ The admin registers the customers detail to manual customers file.
- Registration of new vehicle
  - ✓ Vehicle owner supply vehicles.
  - ✓ Manger registers new vehicles.

2.4. Drawbacks of the existing system

- Recording vehicle information is exhausting.
- Customers have difficulty to get the vehicle owner address.
- Vehicle owner advertise their service manually like speaking literally.
- It‟s a time-consuming process.
- Vulnerable storage of documents.
- May be outdated information gets passed.
- Sometimes it is the spelling error in the manual system that gets saved and after that, while we check it in the final phase it gets rejected.
- The existing system has no automated database, files are stored manually.
- There is a lack of data organization in current existing system.
- Data or files loss due to improper handling

2.5. Business Rules of the Existing System

Vehicle owners have policies to achieve the business objectives, to satisfy customers, to make good or wisely use of resource, and to conform laws or general business conventions. Business

rules become requirements for the services that give the criteria and conditions for making a decision.

In vehicle management system there are business rules that must be considered for this project.

BR1: Vehicle owners describe his/her service and location to the customers.

BR2: - Customers must go to the specific place of the vehicle owner physically in order to rent and buy vehicle.

BR3: - The customers must make agreement with vehicle owners by filling all necessary information to get the service.

BR4: Vehicle owner describe vehicles type, price and model to customers.

BR5: - Customers to register in the system they must have Keble identity card.

BR6: Customers Negotiate with service owners to sell, rent and get other services.

BR7: - The customer must pay money for service

BR8: -The customer agrees to pay penalty, if he/she does not return the rented vehicle on time.

BR9:-The vehicle owner must have trade license.

CHAPTER THREE

3. OVERVIEW OF PROPOSED SYSTEM

3.1. Introduction

The newly developed system that we are going to design is Web based vehicle management system. This system has a wide range of benefits for the user and vehicle owners. Especially for customers by saving the time required. To analysis and design the newly developed system we have been used object oriented approach diagrams such as use case diagram, component diagram, sequence diagram, Activity diagram, state diagram, object diagram and class diagram.
This project will intend to advocate for the need of Vehicle owner and vehicle users to change the manual system to automated system vehicle management system. Vehicle owner use the system as a work place for searching purchasing and tracking of vehicles for customers. Customers can log in to the system remotely and get information about vehicles.
Automated systems make it possible to have a better accuracy, to increase the quality of the work, to reduce the time it takes, to minimize cost, to keep the security of data in most advantageous condition, to make data transfer easier and also make it possible to save and back up all transactions in case of vehicle management to keep the data in a centralized way which is available to all the users simultaneously.
Our system is a web based for managing vehicle and provides features like time efficiency to show vehicle details, user profiles and whatever the customer will give the feedback.

3.2. Functional Requirement

These are statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations. It specifies the application functionality that the developers must build into the product to enable users to accomplish their tasks.

The system shall allow the manager to:

- Add new vehicles to the existing list of available vehicles:-Manager accept vehicles from vehicle owner and register them.

- Modify the price of each vehicle:-if new model vehicles are supply, the manager will be modify the price according to the order of vehicle owner.
- Update information of the vehicles if modification is needed.
- Search vehicles by specific record.
- Generate report about vehicle information.
- Display all list of available vehicles, rent and sell vehicles, and off duty vehicles for the purpose of information.

The system shall allow the vehicle owner to:

- View vehicles information
- Sell vehicles
- Deal vehicles
- View vehicle info
- Notify the last dates of the rented vehicle to the customer.
- Rent vehicle to the customer.
- Generate report

The system shall allow the Admin to:

- Manage users:-the System shall allow to the manager to add or register employees by create account, deactivate (remark) and block the user account, update users account, save users information and view their information.
- Generate report

The system shall allow the Customers to:

- Make registration
- View the vehicle information
- Search vehicles by specific record
- Buy vehicle
- Fill required information of customers of agreement to rent or buy vehicle
- Give feedback.

3.3. Non-Functional Requirements

3.3.1. User Interface and Human Factors

The interface of the proposed system is simple to understand; easy to use and user-friendly interface and users of the system easily use and perform their tasks. To design a better user interface we design buttons, checkboxes, menus, and others by using bootstrap framework.

3.3.2. Hardware Consideration

Desktop: almost all tasks of our project are performed on computer.

Flash disk: required for data movement to store & transfer data from one PC to another PC.

3.3.3. Security Issue

In order to make the system safe from an unauthorized access and modification, the system uses a login account to differentiate among the different users of the system on WWSSO to protect the sensitive customer and material information. This enables the system to verify who has logged in using the correct logging account provided and display the right form associated with that user.

3.3.4. Performance Consideration

Response Time: - Upon request for user the system under normal condition should display results as quickly as possible.

Processing Time: - Since the system is developing with efficient programming language and database upon request for user‟s Activities the system under normal condition should `process the request as quickly as possible by using multi-tier architectures.

Concurrent: - Processing the system can support multiple users at a time.

Efficiency: - The system gives appropriate output based on the expected lists of inputs.

The system must ensure allocation and use of services being requested for the users by using minimum memory storage, cost, time and human power.

Accuracy: - Proposed system will be better due to reduction of error. All operation can be done correctly and it ensures that whatever information is coming from the database is accurate.

3.3.5. Error Handling and Validation

Our system will face different interactions from different users. Therefore, each interaction may bring an invalid input to the system. These invalid inputs may crash the system. Our system will

be implemented to capture any invalid data with the error handling mechanism. Therefore; any invalid data will be thrown and notified to the user as soon as possible.

- Enable the user to confirm that details are correct before creation, deletion or modification occurs.
- Respond to error inputs by asking the user to renter data in the correct format.
- The system should display an error message if the user inputs an invalid character. This all will be done by using JavaScript which supports declaration of a function and we will use that function to validate the user input strongly

3.3.6. Quality Issues

Usability: -

- The system shall provide a uniform look and feel between all the web pages.
- The system shall provide the use of icons and toolbars.
- The interface should contain prompts and help to avoid making mistakes
- The product should be used by people with no training

Reliability: - The system should be reliable in retrieving and displaying only the requested data for the user. Users can rely on the information get would be true and dependable.

Availability: - The system should be available 24 hours a day and 7 days a week.

Storage capability: - The system has an ability to store the required information of the customer in the database.

3.3.7. Backup and Recovery

We use aback up method regularly in order to prevent the data from loose. We use some of the Medias (HD, flash).

3.3.8. Physical Environment

In the physical environment factor to protect server from overheat and other natural disaster like rain the server should keep in well-equipped and ventilated rooms for better protection.

3.3.9. Resource Issues

The system requires resources that are high speed processor and memory for both client and server.

3.3.10. Documentation

The new system provides required full documentation that is system and user documentation. The developed system has full documentation if some failure occurred the maintainer could easily maintain the system using documentation.

CHAPTER FOUR

4. SYSTEM ANALYSIS

In this chapter, we describe the system model, object model, and dynamic model. System analysis is the analysis of the role of a proposed system and the identification of a set of requirements that the system should meet, and thus the starting point for system design. The design is passing to the programmers, who are responsible for the actual implementation of the system. Analysis model contains three models: Functional, object and dynamic models. Use case diagrams can describe the functional model. Class diagrams and object diagram describes the object model. The dynamic model can also describe in terms of sequence, activity and state chart diagrams.

4.1. System Model

To produce a model of the system which is correct, complete, and consistent we need to construct the analysis model, which focuses on structuring and formalizing the requirements of online vehicle management system. The analysis model contains three models: functional model this model deals with the functionality of the system, what expected from the online vehicle management system. The use case diagram generally describes the functional model. Object model this model describes the structure of the system. The object model includes attribute, object, and association. Dynamic models deal with the internal behavior of the vehicle management system and how one-use case executes, series of activities that one use case executes as well as the state. The sequence, state chart, and activity diagrams describe the dynamic model.

4.1.1. Use Case Model

Actor specification

In our system, we have the following actors

- Manager
- Vehicle owner
- Customer
- Admin

4.1.1.1. Use case Diagram

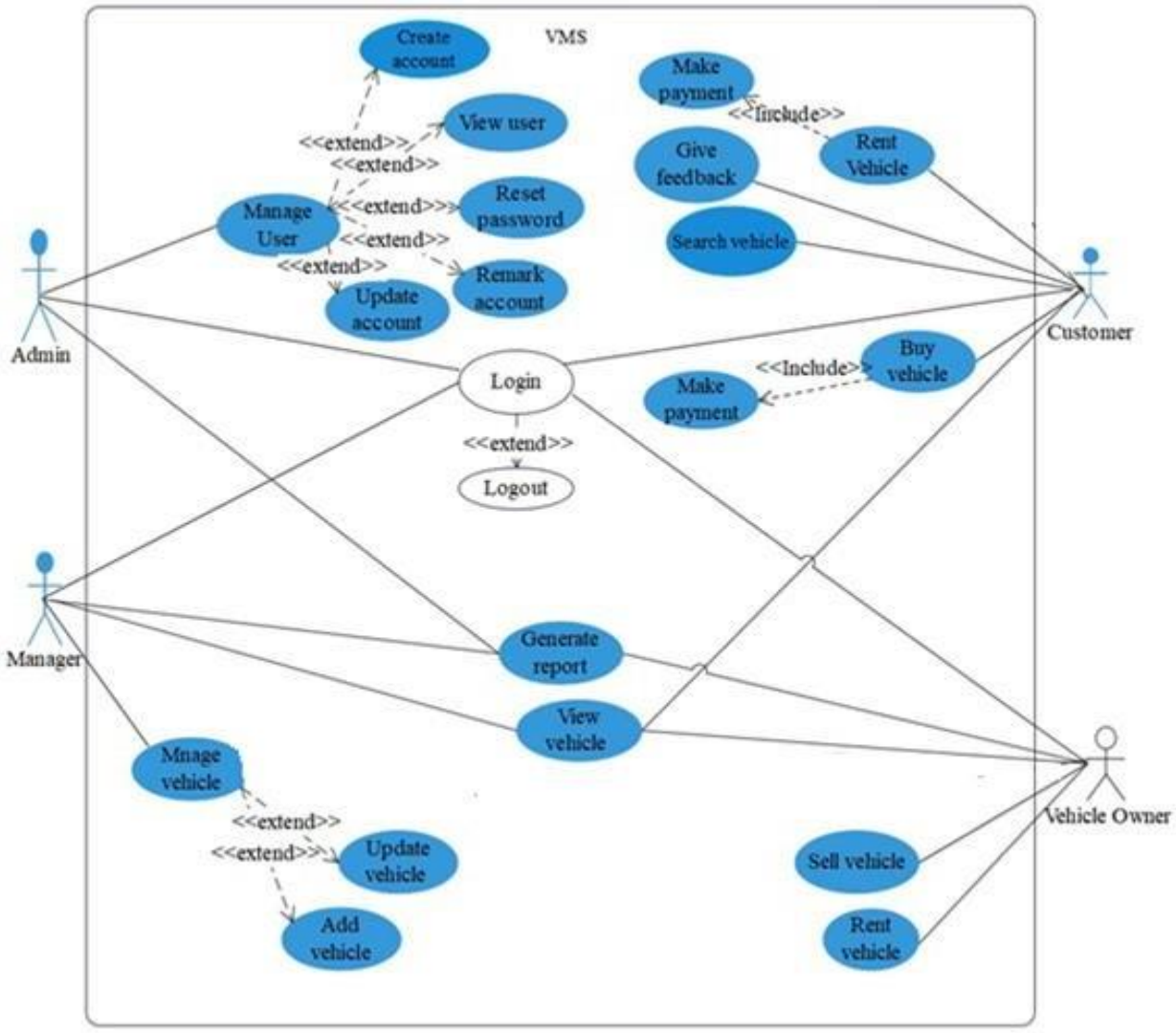


Figure 4.1: Use case diagram

4.1.1.2. Descriptions of use case diagram

Table 4.1: Use case description for login

Use case	Login
Use case number	UC-01
Actor	Vehicle owner, manager, Customer and Admin
Description	This use case describes how the vehicle owner, manager, customer and admin to login in the system
Precondition	The vehicle owner, manager, customer and admin must be registered on the system
Post-condition	The authenticated person gets the appropriate page
Basic course of Action	Flow of action 1.Actor select the account privilege 2.The system displays the login form 3. Fill username and password, click on login button. 4. The system verifies that all the filled have been filled out and valid. 5.the system display the personal page 6. Use case End
Alternate course of action	4.1 if all fields are not filled out and not matched to the username and password the system notifies the actor a message verify username or password and then go back or return to step 3 of basic course of action to enter again.

Table 4.2: Use case description for create account

Use case	Create account
Use case number	UC-05
Actor	Admin
Description	These use case allows Admin to create account for user.
Precondition	Login

Post-condition	Successfully create user account
	Flow of event
	1. Admin enters username and password. 2. The system checks username and password. 3. Admin login and open create account page for user form. 4. The system display form. 5. Admin fills the form correctly. 6. The system checks each field is fulfill. 7. Click create button. 8. Successfully create the account. 9. Use case exit.
Alternate courses of action	2.1. If Admin not enters username and password goes to basic course of action 1. 6.1. If Admin not enter correct information goes to basic course of action 5.

Table 4.3: Use case description for rent vehicle

Use case	Rent vehicle
Use case number	UC-02
Actor	Customer
Description	This use case permits customers to rent and make schedule for renting vehicle based on the availability of the vehicle
Precondition	Customer wants to rent a vehicle and service details about customer have to be entered.
Post-condition	Customers rent successfully
Basic course of Action	Flow of action

	1. The customer wants to rent vehicle go to the page. 2. The customer clicks rent link. 3. The system prompts the customer to fill a rental form. 4.The customer enters the necessary information (First Name , Middle Name, Last Name, Phone Number, Email Address, Location, Drop Address, From Date, To Date) 5. The customer clicks rent button to rent. 6. the system checks all required information had been filled and the date entered dates are valid 7. The system presents information to accept or decline the rental agreement. 8. The customer accepts the reservation and click accept. 9. The systems show the customer that the reservation has been completed. 10. Use case ends.
Alternate course of action	6.1 If the customer fills invalid information, the system goes back to step 4 to enter the invalid again. 6.2 If the customer declines the agreement, the system displays a message that service canceled.

Table 4.4: Use case description for add vehicle

Use case	Add Vehicle
Use case number	UC-03
Actor	Manager
Description	These use case permits manager to register new vehicles to the system with detail descriptions about the vehicle such as model, brand and price per day.
Precondition	New vehicle purchased.
Post-condition	New vehicle information stored successfully
Basic course of	Flow of event

Action	1. The manager wants to add a new vehicle and click add 2.The requests add new vehicle form page 3. The system response or displays a form to be filled out for vehicle registration. 4. The manager enters the following information in the form(Vehicle brand, vehicle type, vehicle model, plate number, price per day, Vehicle Image, Number of seats, vehicle price,) 5. The manager clicks or presses on the save or insert button. 6. The system verifies that the fields have been filled out correctly. 7. The system displays a successfully stored message to the manager. 8. Use case End.
Alternate course of action	6.1 If all fields are not filled out the system goes back or returns to step 4 of basic course of action.

Table 4.5: Use case description for feedback

Use case	Feedback
Use case number	UC-04
Actor	Customer
Description	This use case permits customer to give feedback.
Precondition	None
Post-condition	Send feedback
Basic course of	Flow of action
Action	1. The customers click on feedback link. 2. The system displays form. 3. The customers give feedback with their details (Name, Phone number, Email, Subject, and Message). 4. Then the system checks the field are fulfill. 5. Feedback successfully send. 6. Use case End.

Alternate course of action	4.1. if not fulfill the details customer go back or returns step 3 of basic course of action

Table 4.6: Use case description for generate report

Use case	Generate report
Use case number	UC-09
Actor	Vehicle owner, manager, Admin
Description	These use case allow Vehicle owner, Admin, manager to generate a report about their activity.
Precondition	Login
Post-condition	Generate report information
Basic course of Action	Flow of even 1The vehicle owner, admin and manager wants to generate report 2. The vehicle owner, manager or admin report pages. 3.the system responds the requested page 4. Then report page the staff click on the generate button. 5. use case Ends
Alternate course of action	1.The vehicle owner, admin and manager wants to generate report 2.the vehicle owner, manager or Admin report pages 4.then report page the staff click
	on the generate button

Table 4.7: Use case description for remark account

Use case name	Remark account
Use case number	UC-10

Actor	Admin
Description	The Admin remark user account
Precondition	The Admin must have a full privilege to remark user accounts
Post-condition	The account will be remark
Basic course of Action	Flow of even
	1. Select the account that he want to remark. 2. Click on remark button. 3. The system show confirm dialog box is you want to remark this user account? 4. Admin click „Yes" button the system remark customer account. 5. Admin click „NO" button the system not remark user account. 6. Use case End.
Exit condition	System admin logout from the system

Table 4.8: Use case description for update vehicle

Use case name	Update vehicle information
Use case number	UC-11
Actor	manager
Description	The manager Update vehicle
Precondition	Login
Post condition	manager will be Update Vehicle Information
Basic course of	Flow event

Action	1. Click on „Update“ link 2. Insert the vehicle number to be updated detail. 3. Change the new value that the form must have. 4. Click on update button. 5. The system prompts the manager to Update vehicle Information. 6. Use case End.
Alternative action	2.1 If Incorrect information entry Display data not found error message 2.2 Go to step 2 and try again

Table 4.9: Use case description for search vehicle

Use case name	Search vehicle
Use case number	UC-12
Actor	users
Description	The Customer Search a vehicle
Precondition	Login
Post condition	Customer will be Search a vehicle
Basic course of Action	Flow of action
	1. Enter the vehicle type on the search field. 2. Click on „search“ button 3. The system display all about the vehicle information. 4. Use case end.
Alternative action	2.1 If incorrect information entry display not found or error message
	2.2 Go to step 1 and try again

Table 4.10: Use case description for payment

Use case name	Buy

Use case number	UC-13
Actor	Customer
Description	The Customer buy a vehicle
Precondition	Login
Post condition	Customer makes payment.
Basic course of Action	Flow of event
	1. Select vehicle. 2. The system displays the order form. 3. fill(payment ID ,Name, price, email, Gender ,address , City ,phone , country ,Vehicle type ,Bank account Number)and click button. 4. The system verifies that all the filled Have been filled out and valid. 5. The system display notification automatically. 6. Use case end.
Alternative action	3.1 If incorrect information entry go to step 3 and try again

Table 4.11: Use case description for sell vehicle

Use case name	Buy
Use case number	UC-14
Actor	Vehicle owner
Description	The Vehicle owner sell a vehicle
Precondition	Login
Post condition	Vehicle owner sell vehicle
Basic course of	Flow of action

Action	1. Go to home page and click sell link. 2.The system displays the order form 3. fill(customer name, vehicle name email, Gender ,address , City ,phone , country ,Vehicle type ,vehicle price)and click sell button. 4. The system verifies that all the filled have been filled out and valid. 5.the system display sell successfully message 6. Use case End
Alternative action	3.1 If Incorrect information entry go to step 3 and try again

4.1.1.3. Use case Scenario

A use case scenario is a single path through the use case. A scenario is an instance of a use case describing a concrete set of actions. Scenarios are used as examples for illustrating common cases or describe a real-world example of how one or more people or organizations interact with a system. They describe the steps, events, and/or actions that occur during the interaction. Their focus is on understandability. The following Scenarios describe the interaction between the users.

Scenario name: Login

Participant actor: admin, vehicle owner, manager, customer when actors wants to login click login button the system display login form to enter his/her username and password after that user fill all required field and click login button, the system validates for validity and existence if pass browse his/her personal homepage and access all private information else redirected to reenter username and password.

Scenario name: Rent vehicle

Participant actor: customer, when the customer wants to rent a vehicle, the customer click rent link, the system display rent form to enter the required information after that user fill all required

field and click rent button, the system validates for validity and if all required field are filled the it displays successfully rent message.

Scenario name: Create account
Participant actor: Admin, when the admin wants to create account, the admin click rent link, the system display create account form to enter the required information after that admin fill all required field and click create button, the system validates for validity and if all required field are filled it displays successfully create message.

Scenario name: Add vehicle
Participant Actor: Manager, when the manager wants to add a vehicle, the manager click add vehicle link, the system display add vehicle form to enter the required information after that manager fill all required field and click add button, the system validates for validity and if all required field are filled it displays successfully add message.

Scenario name: Deal vehicle
Participant Actor: Vehicle owner: when the dealer wants to deal a vehicle, the dealer click deal vehicle link, the system display deal vehicle form to enter the required information after that dealer fill all required field and click deal button, the system validates for validity and if all required field are filled it displays successfully deal message.

4.2. Object Model

4.2.1. Class Diagram

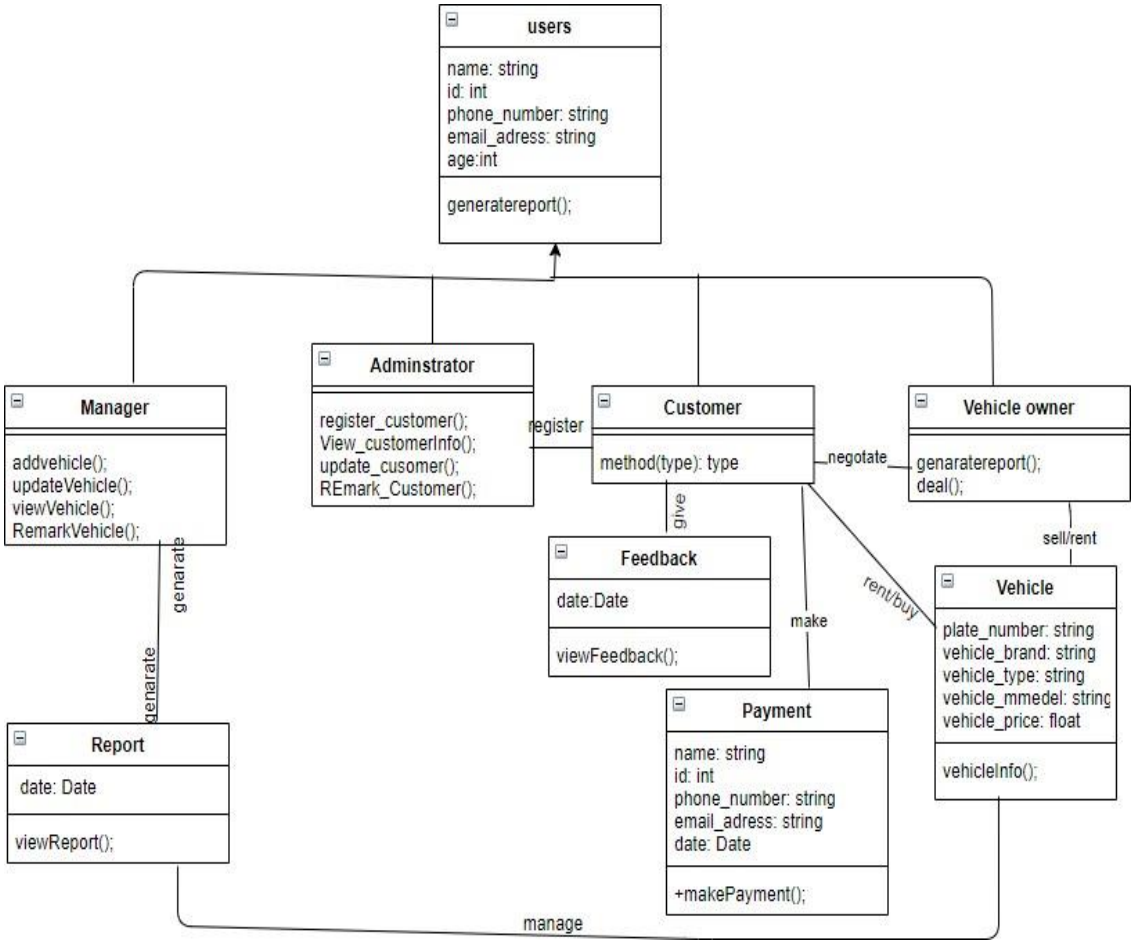


Figure 4.2: Conceptual class diagram

4.2.2. Data Dictionary

A data dictionary is a collection of data definitions used by applications to describe and access a database. These definitions do not contain actual data; rather, they comprise standardized "book keeping" information about the data (metadata). This includes details on the types, names and structures of all data elements, as well as n their interrelationships within the larger context of the database In SQL Server the data dictionary is a set of database tables used to store information about a database‟s definition. The dictionary contains information about database objects such as tables, indexes, columns, data types, and views column Level Details Get column level detail on SQL Server primary keys, data types, and defaults along with descriptions. Easily find missing indexes and incorrect column definitions.

It is a key component in any metadata management or data governance initiative. A data dictionary provides the basis for standardization and harmonization of your data element definitions throughout the enterprise across databases, files, web services, reports, screens and big data systems.

- Objects – contains a row for each object, such as a foreign key or primary key constraint defined within the database.
- Columns – contain a row for each column of an object such as view or tables.
- Tables – return a row for each table object.

Table 4.12: Data dictionary for admin

No	Field Name	Data Type	Constraint	Description
1	ID	Int	Primary Key	It is store Admin id
2	First Name	Varchar (50)	Not null	It is store Admin Name
3	Last name	Varchar (50)	Not null	It is store Admin Name
4	Phone Number	Int (50)	Not null	It is store Admin Phone Number
5	Password	Varchar (50)	Not null	It is store Admin Password
6	E-mail address	Varchar (50)	Not null	It is store Admin E-mail
7	Age	Int	Not null	It sore age

Table 4.13**:** Data dictionary for **c**ustomer

No	Field Name	Data Type	constraint	Description
1	ID	Int	Primary Key	It is store id
2	First-Name	Varchar(50)	Not null	It is store First Name
3	Middle-Name	Varchar(50)	Not null	It is store Middle Name
4	Last-Name	Varchar(50)	Not null	It is store Last Name
5	password	Varchar(50)	Not null	It is store Password
6	E-mail	Varchar(50)	Not null	It is store Email
7	Phone	Int(13)	Not null	It is store Phone Number
8	Address	Varchar(50)	Not null	It is store Address
9	Country	Varchar(50)	Not null	It is store Country
10	birth date	Varchar(50)	Not null	It is store Birth Date

Table 4.14: Data dictionary for vehicle

No	Field Name	Data Type	Constraint	Description
1	Plate Number	Int	Primary Key	It is store Plate Number
2	Vehicle Brand	Varchar(50)	Not null	It is store Vehicle Brand

3	Vehicle Type	Varchar(50)	Not null	It is store Vehicle Type
4	Vehicle Model	Varchar(50)	Not null	It is store Vehicle Model
5	Price	Int(50)	Not null	It is store Vehicle Price
6	Vehicle Image	BLOB(50)	Not null	It is store Vehicle Image
7	Number Of Seats	Int(50)	Not null	It is store Vehicle Seats
8	Vehicle Company	Varchar(50)	Not null	It is store Vehicle Company
9	Vehicle Categories	Varchar(50)	Not null	It is store Vehicle Categories
12	Color	Varchar(50)	Not null	It is store Vehicle Color
13	Description	Varchar(50)	Not null	It is store Vehicle Description

Table 4.15: Data dictionary for manager

No	Field Name	Data Type	constraint	Description
1	First-Name	.Varchar(50)	Not null	It is store First Name
2	Middle-Name	Varchar(50)	Not null	It is store Middle Name

3	Last-Name	Varchar(50)	Not null	It is store Last Name
4	password	Varchar(50)	Not null	It is store Password
5	E-mail	Varchar(50)	Not null	It is store Email
6	Phone	Int(13)	Primary Key	It is store Phone Number
7	Address	Varchar(50)	Not null	It is store address

Table 4.16: Data dictionary for vehicle owner

No	Field Name	Data Type	constraint	Description
1	ID	Int	Primary Key	It is store id
2	First-Name	Varchar(50)	Not null	It is store First Name
3	Middle-Name	Varchar(50)	Not null	It is store Middle Name
4	Last-Name	Varchar(50)	Not null	It is store Last Name
5	Password	Varchar(50)	Not null	It is store Password
6	E-mail	Varchar(50)	Not null	It is store Email
7	Phone	Int(13)	Not null	It is store Phone Number

8	Address	Varchar(50)	Not null	It is store Address
9	Country	Varchar(50)	Not null	It is store Country
10	birth date	Varchar(50)	Not null	It is store Birth Date

Table 4.17: Data dictionary for report

No	Field Name	Data Type	Constraint	Description
1	Report ID	Int	Primary key	It is store id
2	Report date	Date (50)	Not Null	It is store report date
3	Report type	Varchar (50)	Not Null	It is store report type
4	Reporter name	Varchar (50)	Not Null	It is store report name

4.3. Dynamic Model

4.3.1. Sequence diagram

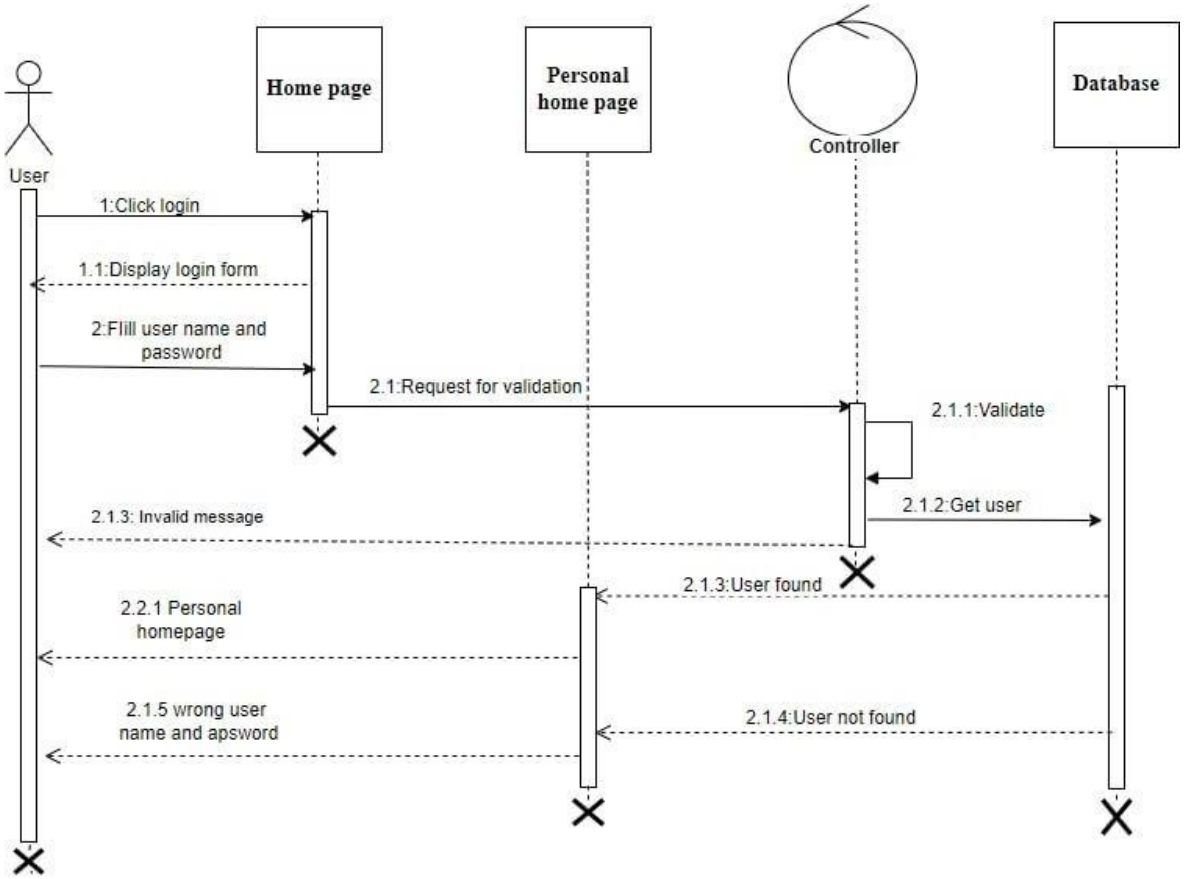


Figure 4.3: Sequence diagram for login

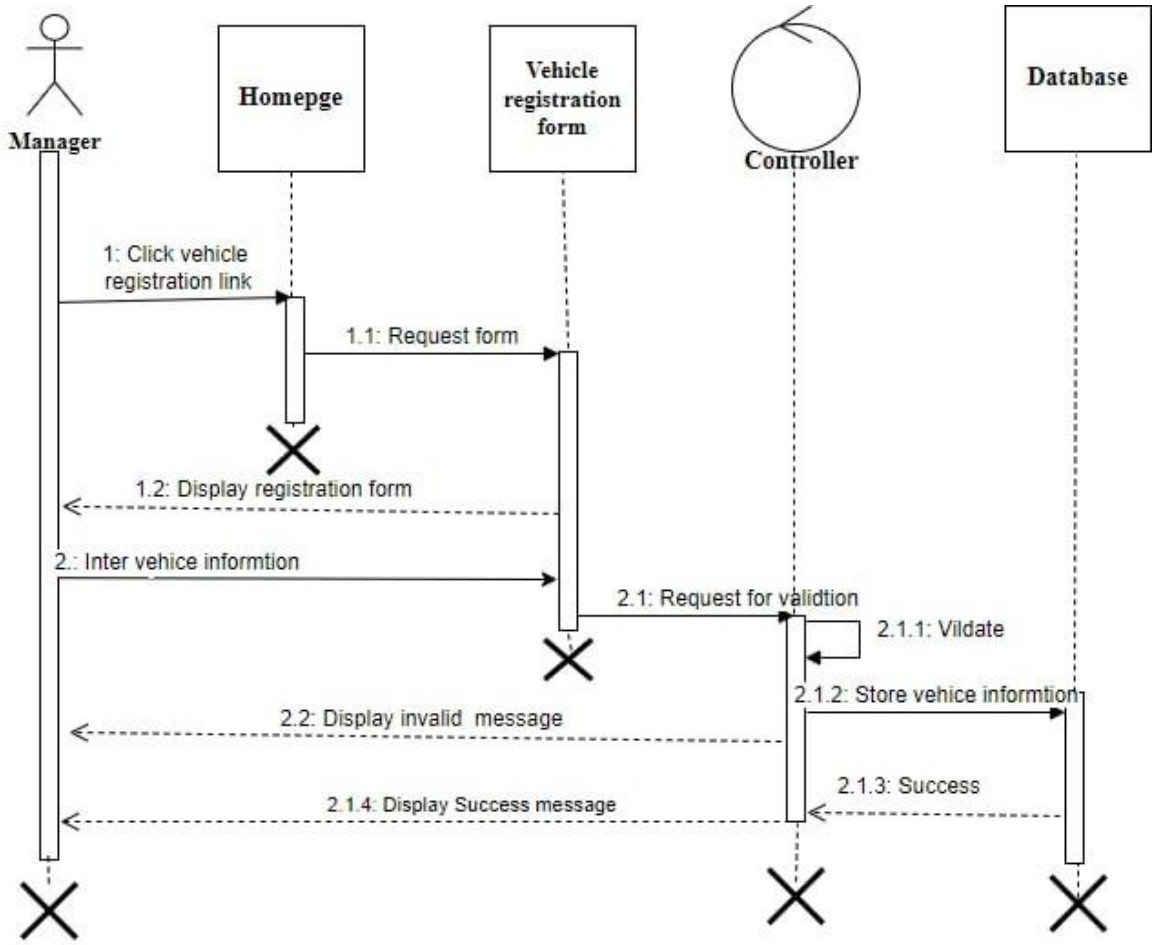


Figure 4.4: Sequence diagram for vehicle registration

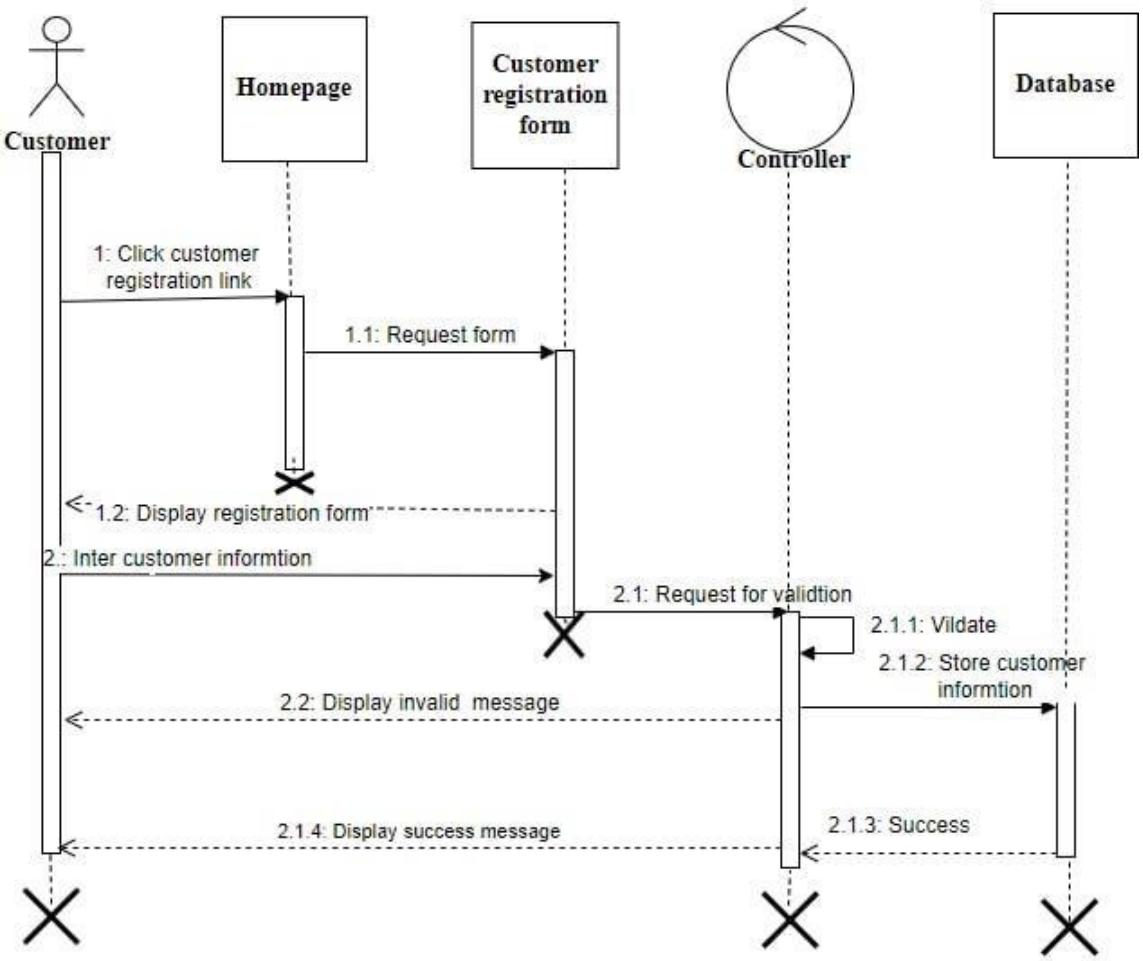


Figure 4.5: Sequence diagram customer registration

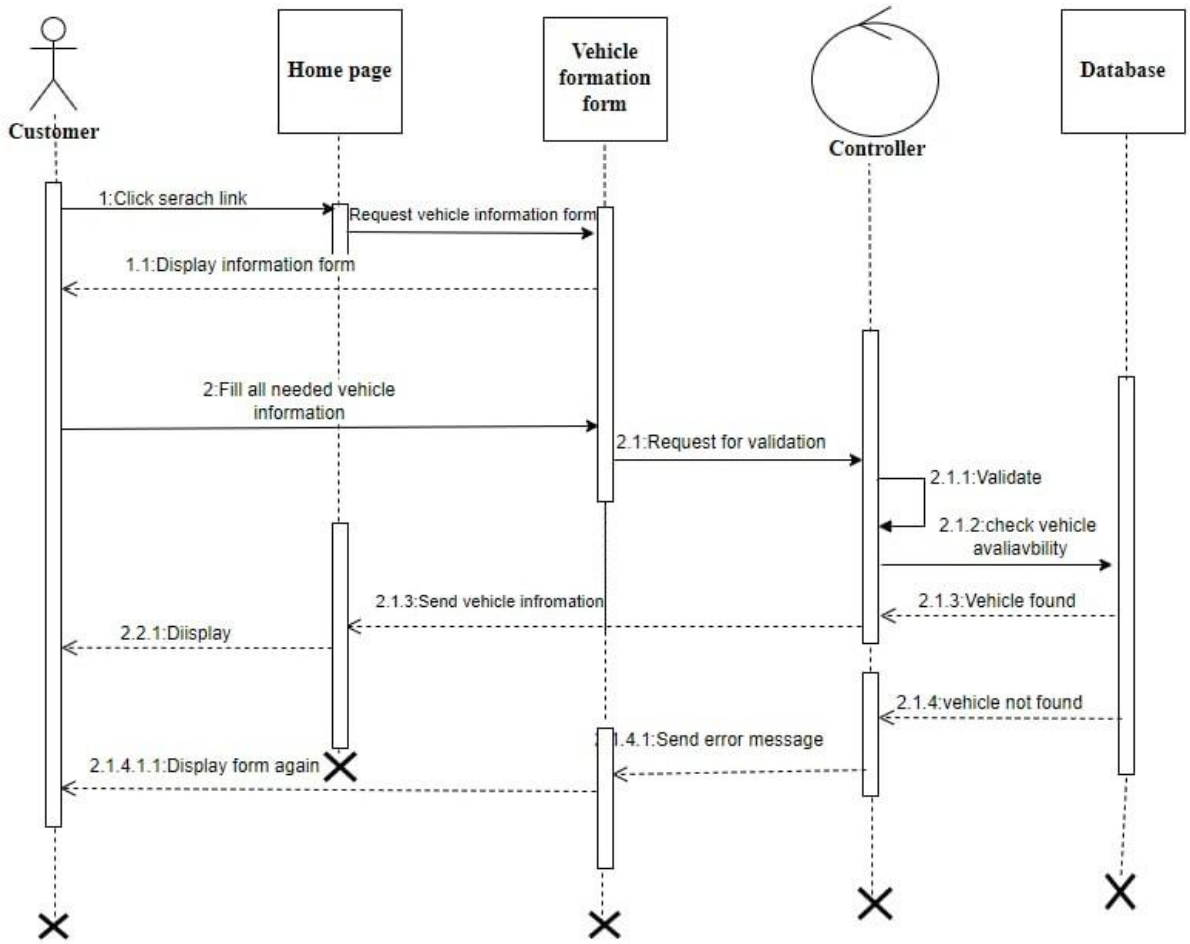


Figure 4.6: Sequence diagram for search vehicle

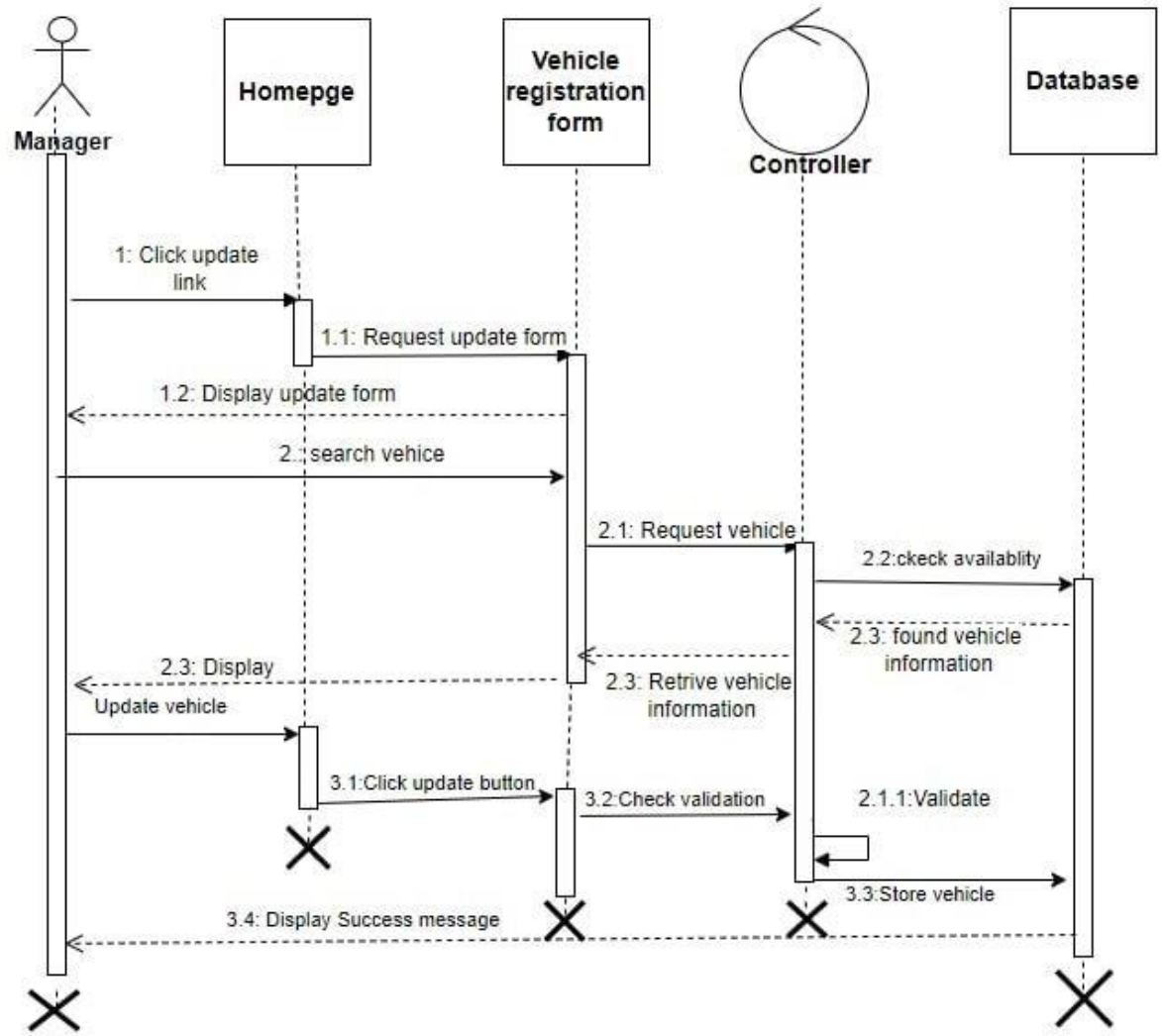


Figure 4.7: Sequence diagram for update vehicle

4.3.2. Activity diagram

Activity diagram is another important diagram to describe dynamic behavior. Activity diagram consists of activities, links, relationships etc. It models all types of flows like parallel, single, concurrent etc. Activity diagram describes the flow control from one activity to another without any messages. These diagrams are used to model high level view of business requirements.

So the purposes can be described as:

- Draw the activity flow of a system.

- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

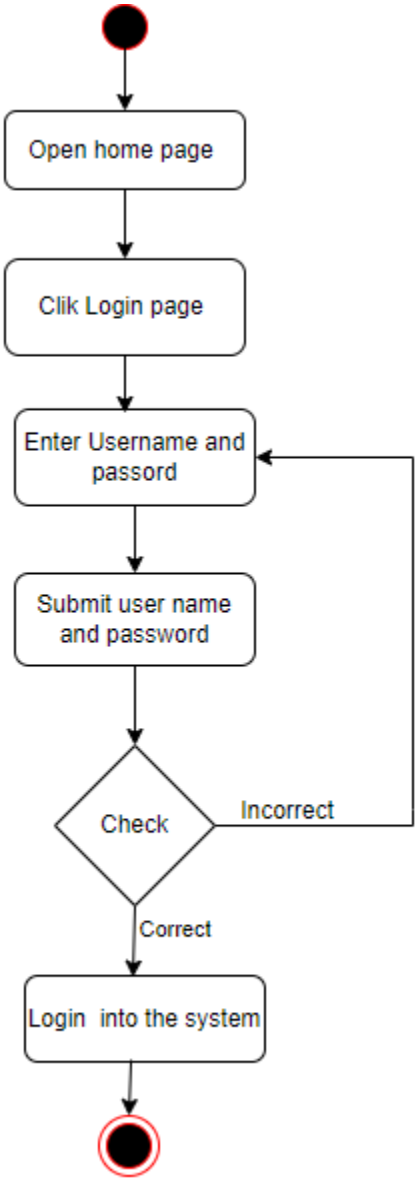


Figure 4.8: Activity diagram for login

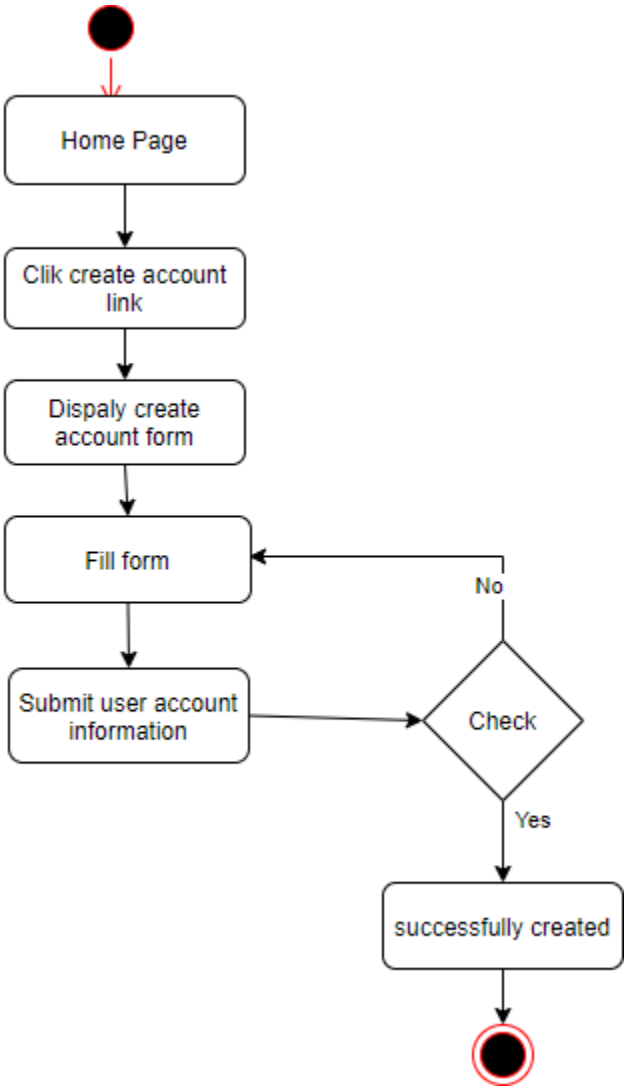


Figure 4.9: Activity diagram for create account of users

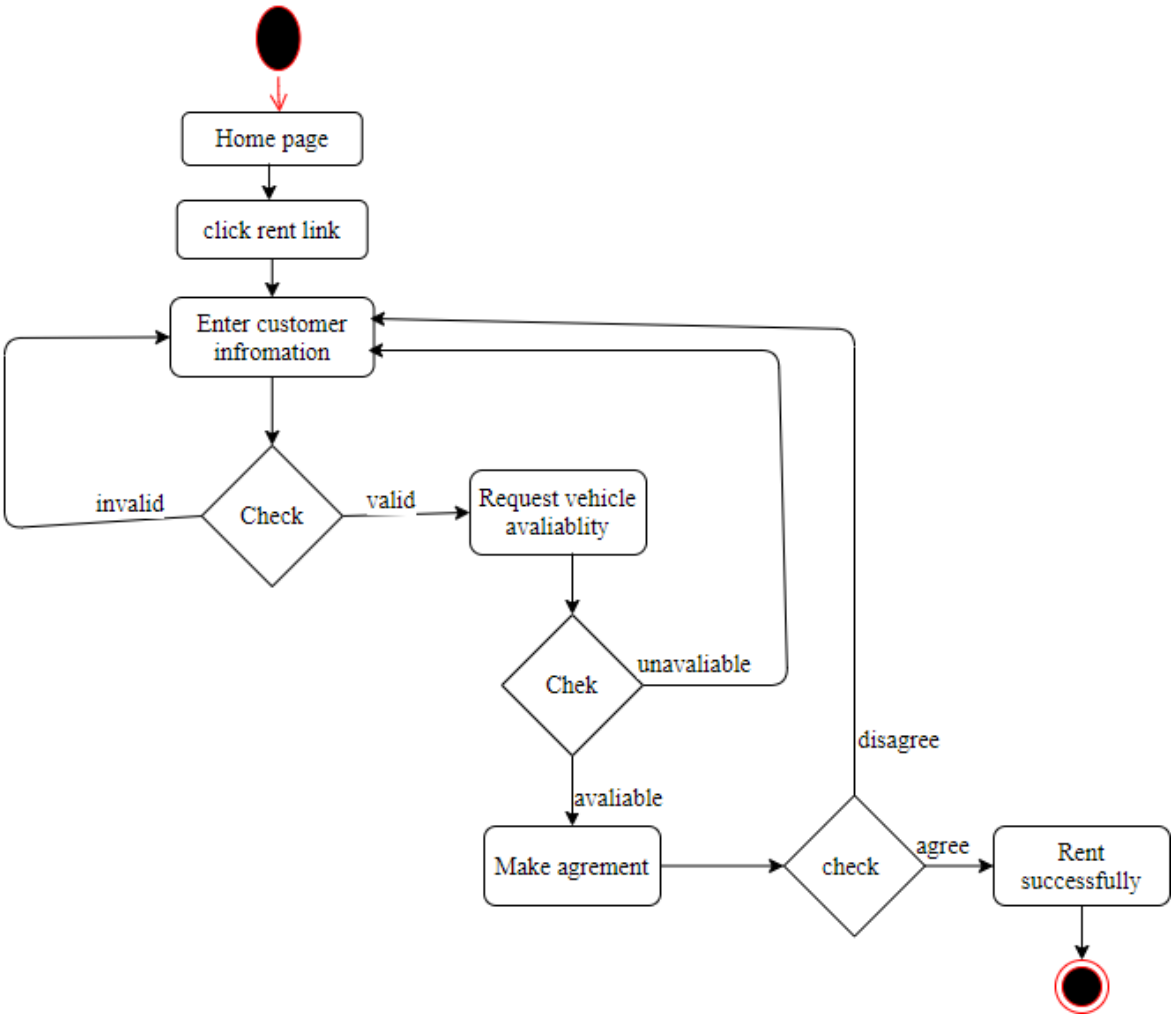


Figure 4.10: Activity diagram for rent vehicle

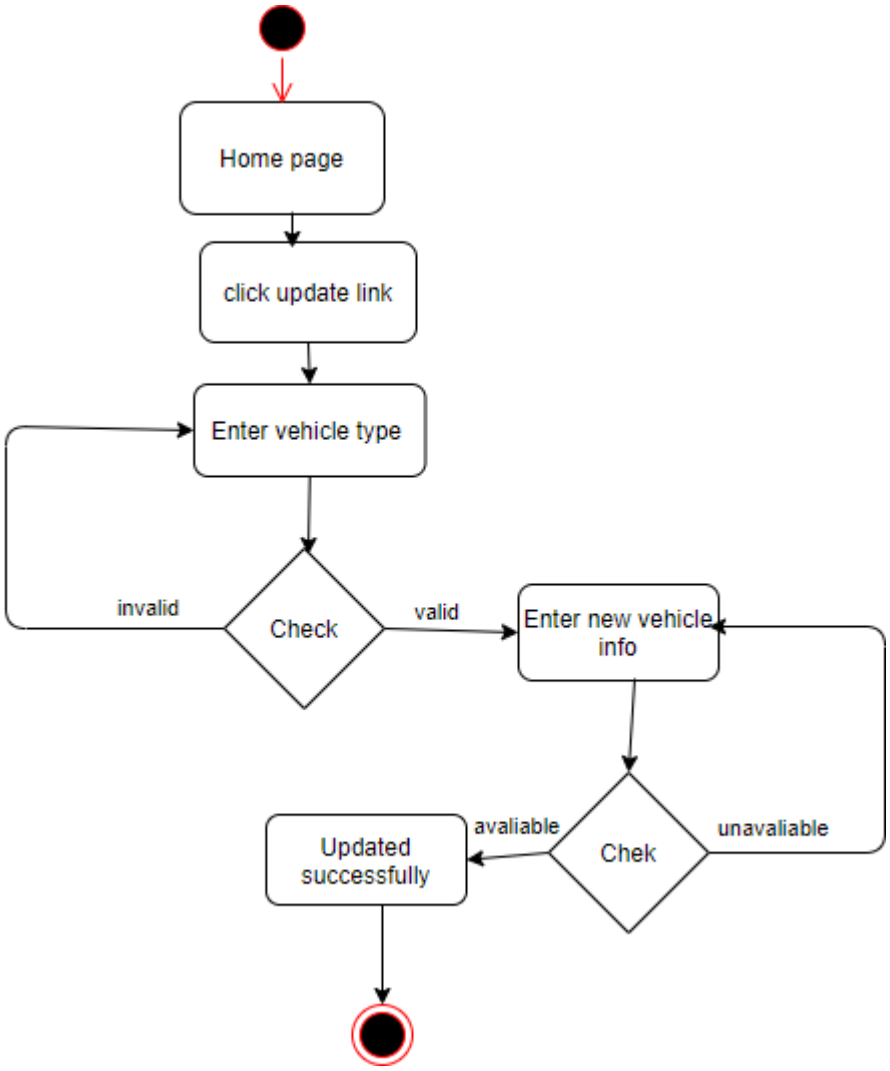


Figure 4.11: Activity diagram for update vehicle

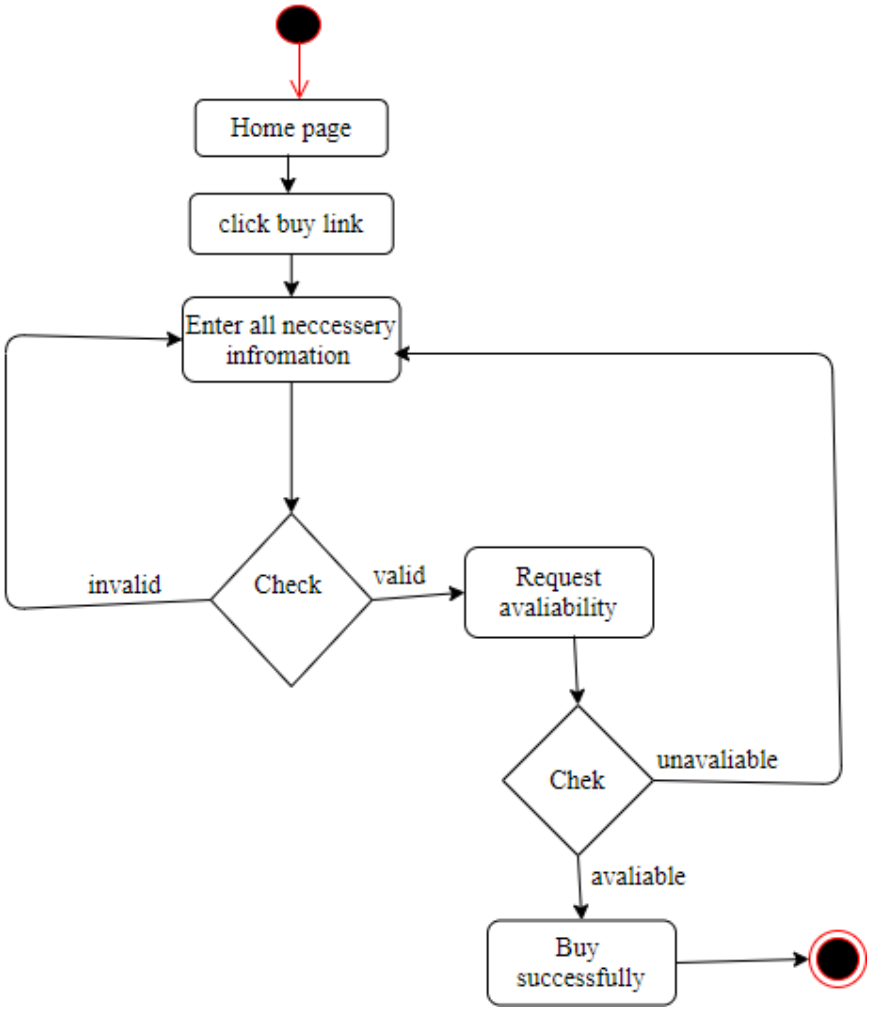


Figure 4.12: Activity diagram for buy vehicle

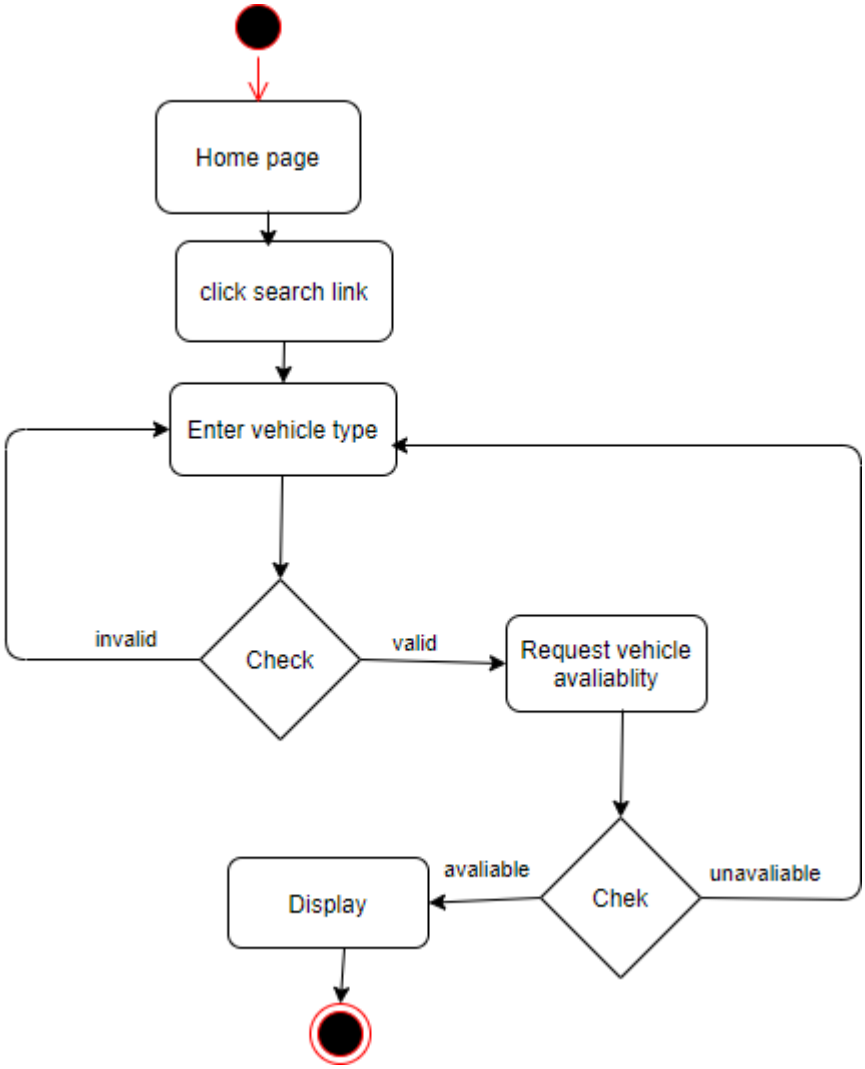


Figure 4.13: Activity diagram for search vehicle

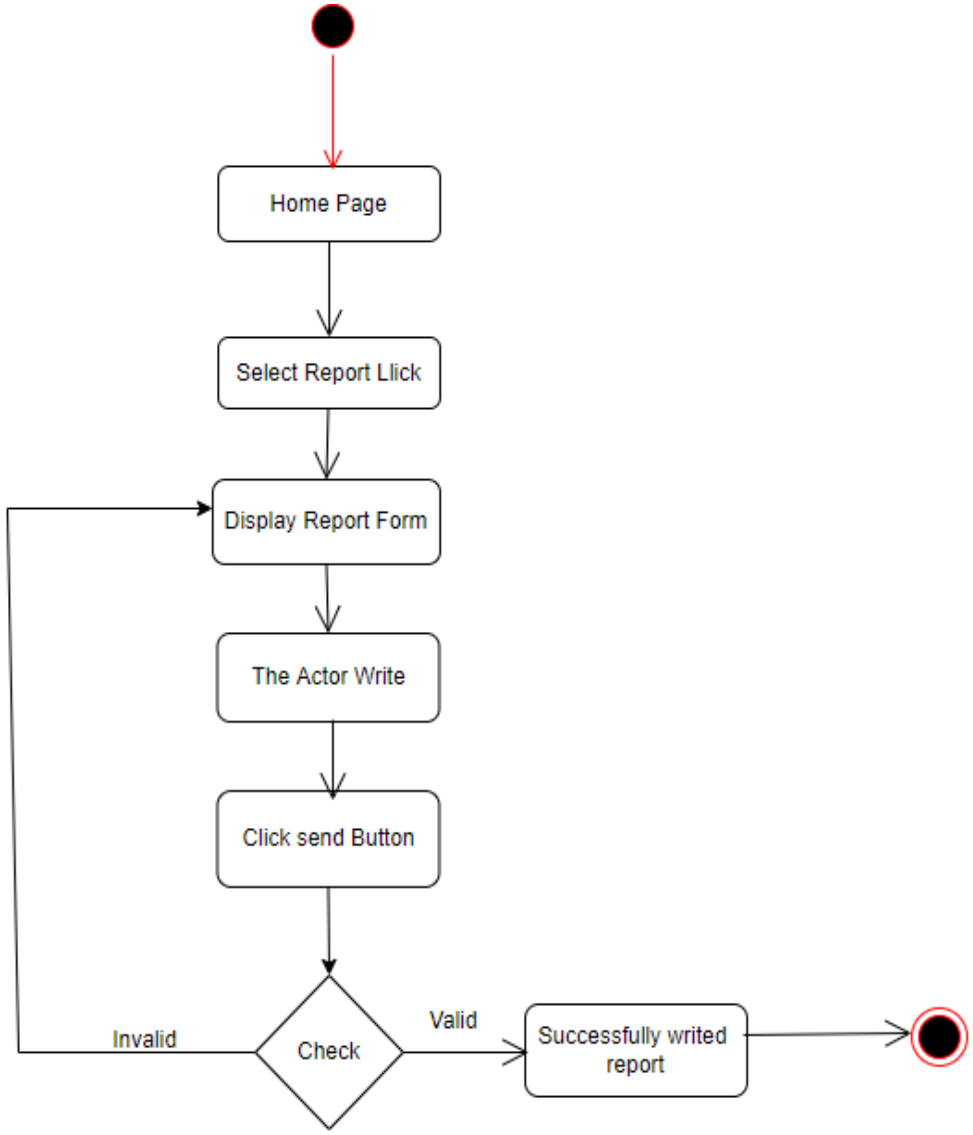


Figure 4.15: Activity diagram for write comment

4.3.3. State chart diagrams

State chart diagram defines the states of a component and these state changes are dynamic in nature. So its specific purpose is to define state changes triggered by events. Events are internal or external factors influencing the system.

The main purpose of State Chart Diagram

- To model dynamic aspect of a system.
- To model life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model states of an object.

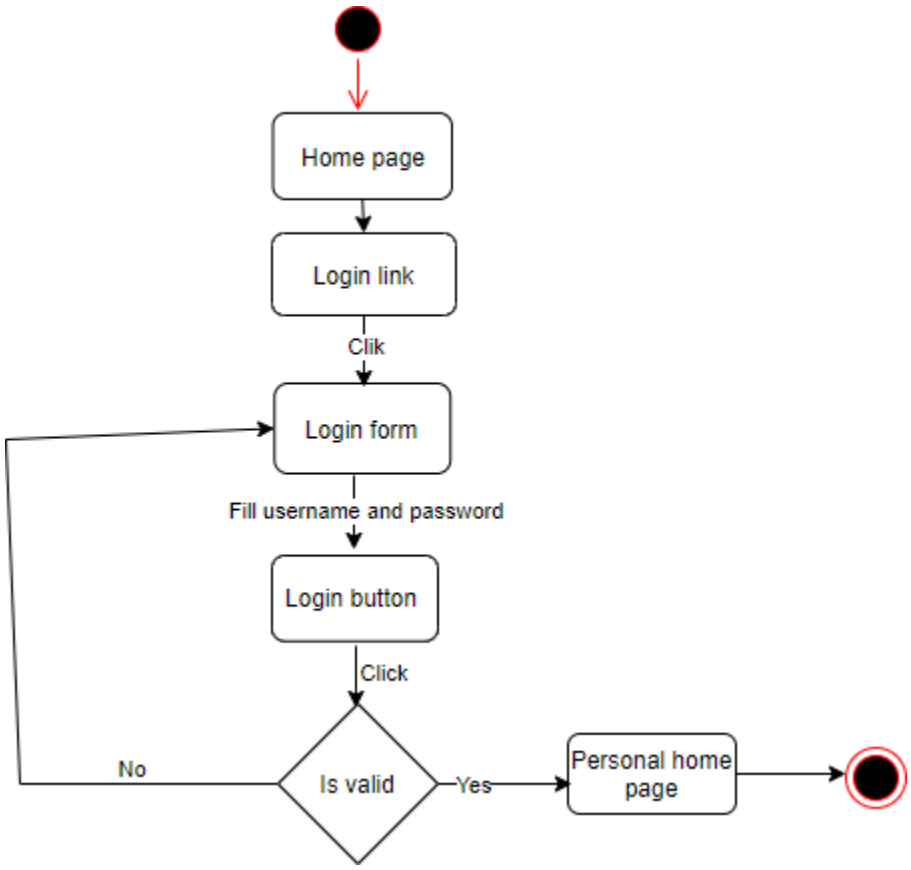


Figure 4.16: State chart diagram for login

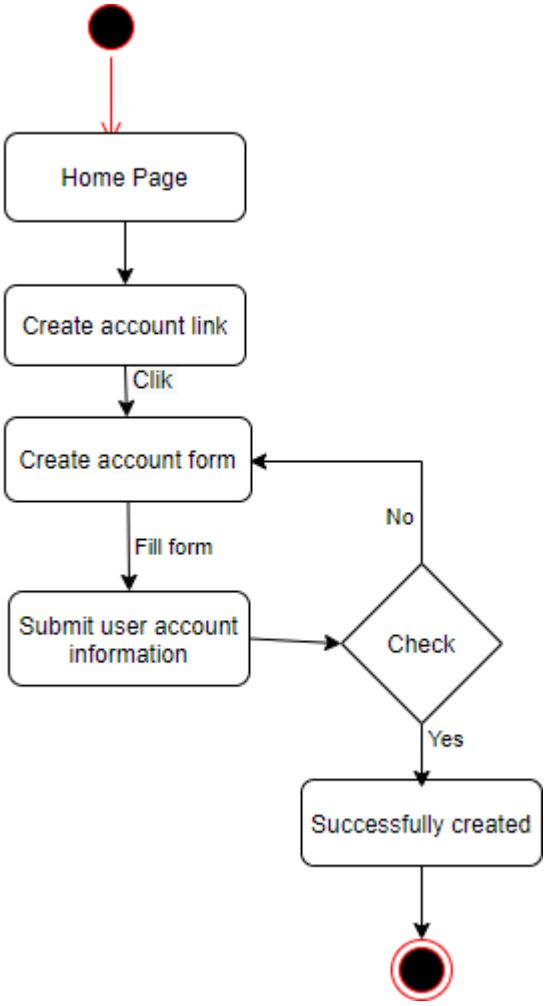


Figure 4.17: State diagram for create user account

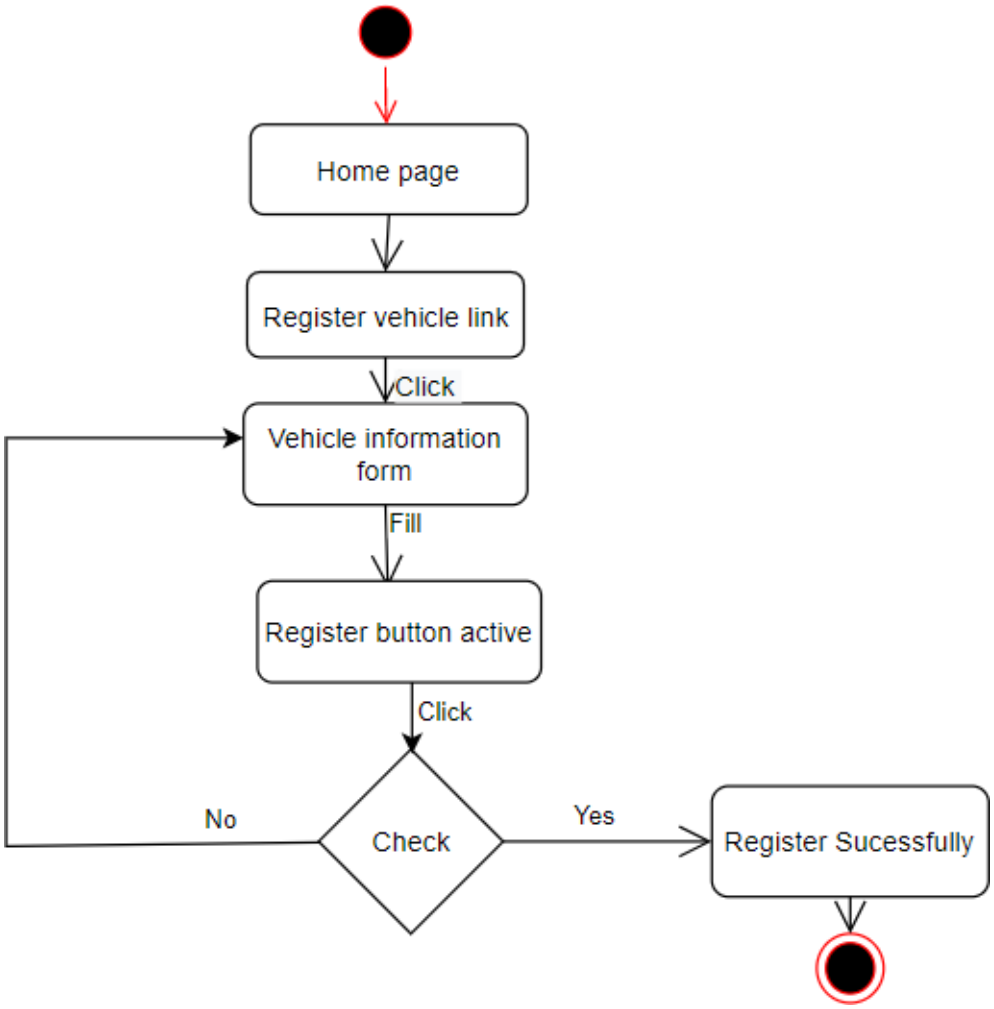


Figure 4.18: State diagram for add vehicle

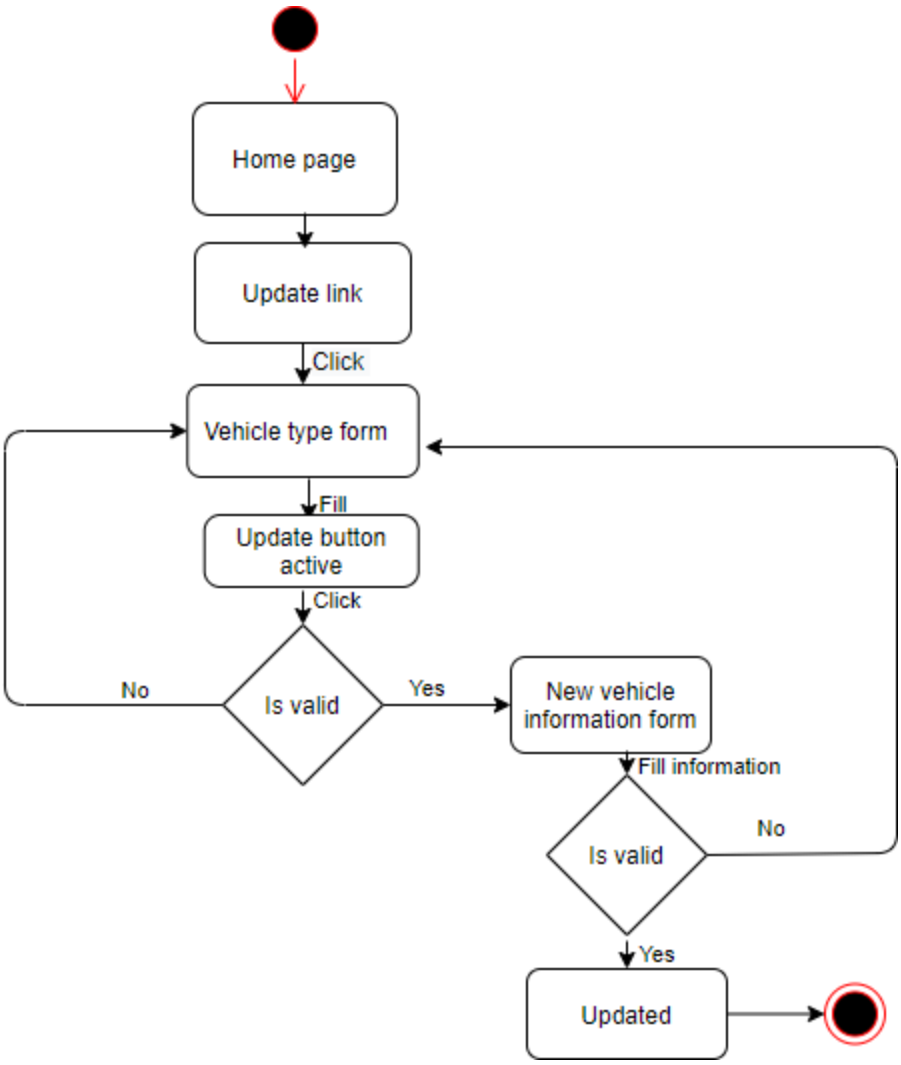


Figure 4.19: State diagram for update vehicle

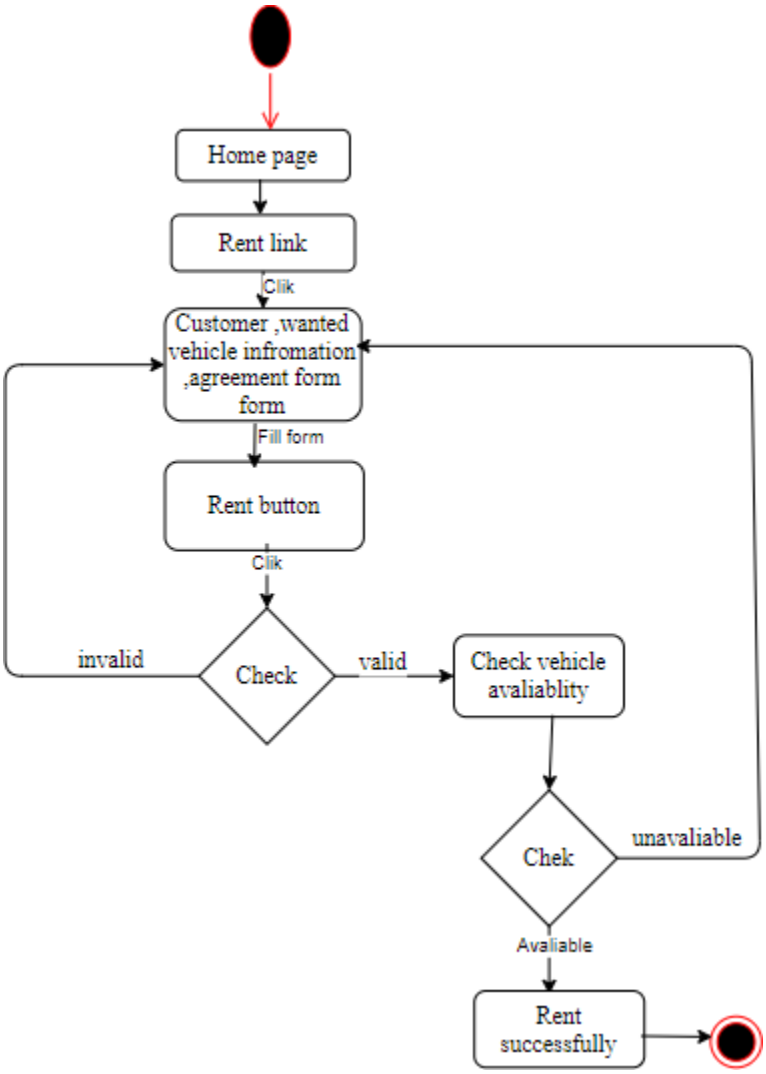


Figure 4.20: State diagram for rent vehicle

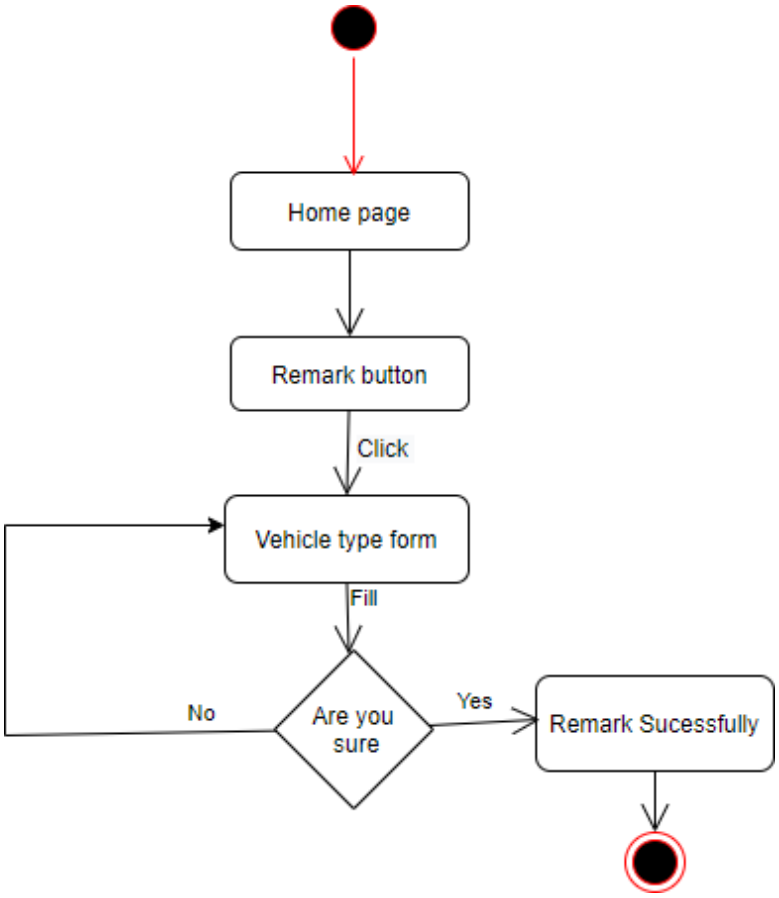


Figure 4.21: State diagram for remark vehicle

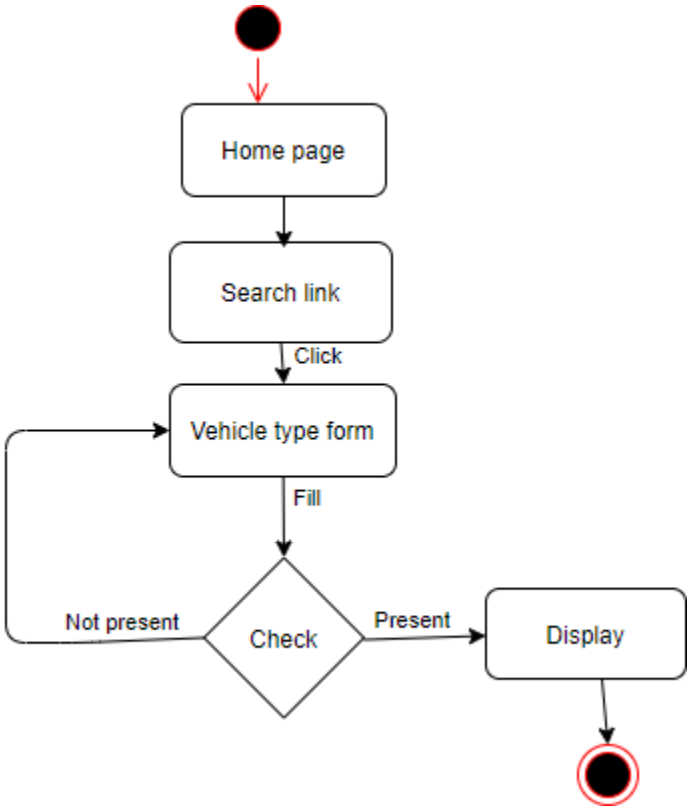


Figure 4.22: State diagram for search vehicle

CHAPTER FIVE

5. SYSTEM DESIGN

5.1. Introduction

In this stage, the complete architecture of the desired system is designed. The system is conceived as a set of interacting subsystems that in turn is composed of a hierarchy of interacting objects, grouped into classes. System design is done according to both the system analysis model and the proposed system architecture. Here, the emphasis is on the objects comprising the system rather than the processes in the system. Object-oriented system design involves defining the context of a system followed by designing the architecture of the system. Context: The context of a system has a static and a dynamic part. The static context of the system is designed using a simple block diagram of the whole system which is expanded into a hierarchy of subsystems. The subsystem model is represented by UML packages. The dynamic context describes how the system interacts with its environment. It is modeled using use case diagrams. The system architecture is designed on the basis of the context of the system in accordance with the principles of architectural design as well as domain knowledge. Typically, a system is partitioned into layers and each layer is decomposed to form the subsystems.

5.2. Design Goals

The design goals describe the qualities of the system that developers should improve. Design goals are normally derived from the non-functional requirements of the system. So the followings section describes the design goals of the system:

There are many aspects to consider in the design of a piece of software. The importance of each consideration should reflect the goals and expectations that the software is being created to meet. Some of these aspects are:

User Interface and Human Factor

The interface of the proposed system is simple to understand; easy to use and user-friendly interface and users of the system easily use and perform their tasks. To design a better user interface we design buttons, checkboxes, menus, and others by using bootstrap framework.

Hardware Consideration

- Desktop: almost all tasks of our project are performed on computer.
- Flash disk: required for data movement to store & transfer data from one PC to another PC.

Security

In order to make the system safe from an unauthorized access and modification, the system uses a login account to differentiate among the different users of the system to protect the sensitive customer and material information. This enables the system to verify who has logged in using the correct logging account provided and display the right form associated with that user.

Performance Consideration

- Response Time**:** - Upon request for user the system under normal condition should display results as quickly as possible.
- Processing Time**: -** Since the system is developing with efficient programming language and database upon request for user‟s Activities the system under normal condition should `process the request as quickly as possible by using multi-tier architectures.
- Concurrent: - Processing the system can support multiple users at a time.
- Efficiency: - The system gives appropriate output based on the expected lists of inputs.
- The system must ensure allocation and use of services being requested for the users by using minimum memory storage, cost, time and human power.
- Accuracy: - Proposed system will be better due to reduction of error. All operation can be done correctly and it ensures that whatever information is coming from the database is accurate.

Error Handling and Validation

Our system will face different interactions from different users. Therefore, each interaction may bring an invalid input to the system. These invalid inputs may crash the system. Our system will be implemented to capture any invalid data with the error handling mechanism. Therefore; any invalid data will be thrown and notified to the user as soon as possible.

- Enable the user to confirm that details are correct before creation, deletion or modification occurs.
- Respond to error inputs by asking the user to renter data in the correct format.

- The system should display an error message if the user inputs an invalid character. This all will be done by using JavaScript which supports declaration of a function and we will use that function to validate the user input strongly

Quality Issues

- Usability**:** -
  - ✓ The system shall provide a uniform look and feel between all the web pages.
  - ✓ The system shall provide the use of icons and toolbars.
  - ✓ The interface should contain prompts and help to avoid making mistakes
  - ✓ The product should be used by people with no training
- Availability**:** - The system should be available 24 hours a day and 7 days a week.
- Reliability**:** - The system should be reliable in retrieving and displaying only the requested data for the user. Users can rely on the information get would be true and dependable.

Storage capability

The system has an ability to store the required information of the customer in the database.

Backup and Recovery

We use aback up method regularly in order to prevent the data from loose. We use some of the Medias (HD, flash).

Physical Environment

In the physical environment factor to protect server from overheat and other natural disaster like rain the server should keep in well-equipped and ventilated rooms for better protection**.**

Resource Issues

The system requires resources that are high speed processor and memory for both client and server

Documentation

The new system provides required full documentation that is system and user documentation. The developed system has full documentation if some failure occurred the maintainer could easily maintain the system using documentation.

5.3. Architecture of the proposed System

The proposed software has three layers of architecture. The layers are the following:-

Application layer

In this project application layer provides graphical user interface that browse application and specific entry forms to the user of the system. Application layer interact with logical layer through http protocol. User (Manager, Dealer, Administrator, Landlord, and customer) interact different interfaces such as login interface, manage user interface, sale interface, add user interface, feedback interface, manage vehicle interface, delete account interface, customer signup etc.

Logic layer

Layer that used to communicate application to database which keep the data structure consistent. Web based vehicle management system the logic layer intermediate database layer with application layer by http protocol and ODBC (object database connectivity)

Database layer

In this architecture the database stores the user information and if it is necessary retrieved from it. It provides the response to application layer request through logical layer. The database administrator keeps the data in database in a secured manner.

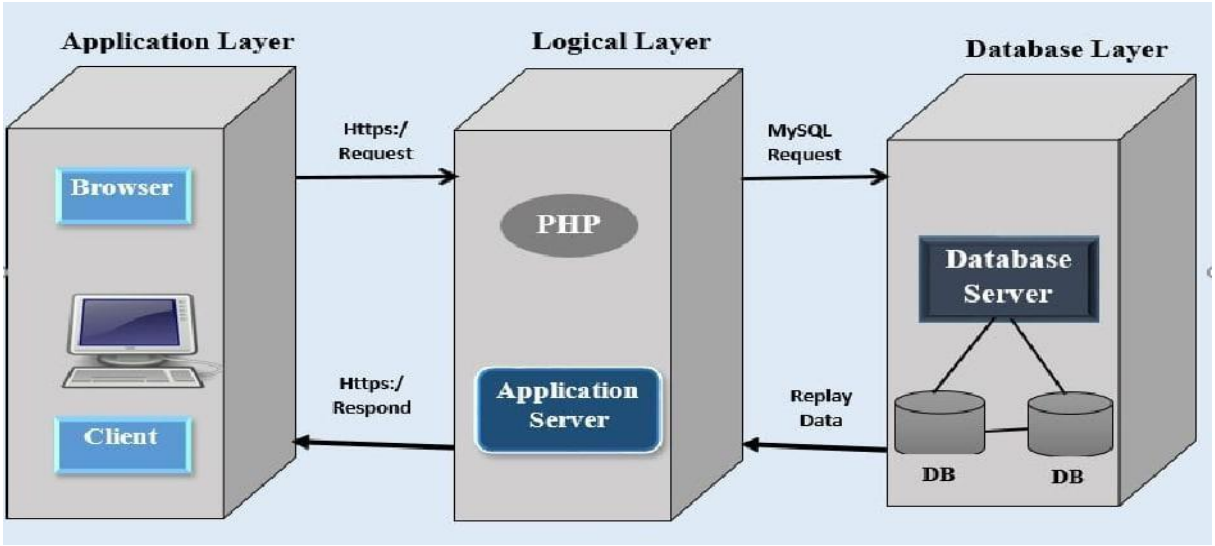


Figure 5.1: System architecture

5.4. Subsystem Decomposition and Description

Sub-system decomposition is the processes of breaking down the system into sub-systems, and each sub-systems analysis separately. The main aim is to minimize the complexity of the system. We decompose a system into simpler parts, called subsystems, which are made of a number of solution domain classes. This subsystem performs the following tasks.

- Administrator subsystem
  - ✓ Manage users
  - ✓ Generate report
- Customer subsystem
  - ✓ search vehicle
  - ✓ View vehicle information
  - ✓ Rent or buy vehicle
  - ✓ Make payment
- Manager subsystem
  - ✓ Mange vehicles
  - ✓ Generate report
- Vehicle owner subsystem
  - ✓ Sell vehicle
  - ✓ Deal vehicle
  - ✓ Rent vehicle
  - ✓ Generate report
  - ✓ View vehicle info

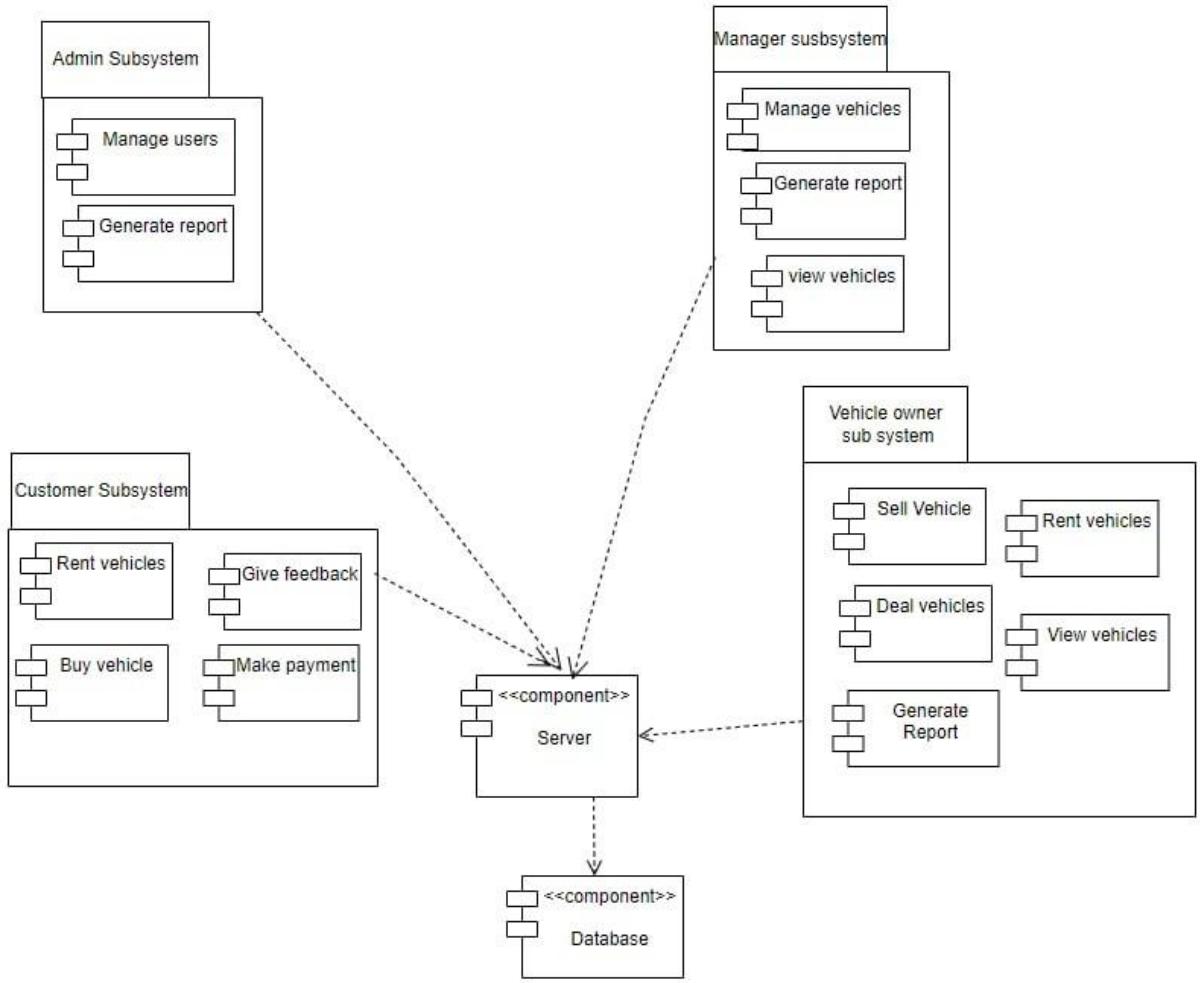


Figure 5.2: Subsystem decomposition

5.5. Hardware/Software Mapping

Deployment diagrams are used for describing the hardware components where software components are deployed. Hardware and software mapping shows us how the hardware and software components are connected together to perform user tasks. The one and the main diagram, which shows this hardware and software mapping, is deployment diagram.

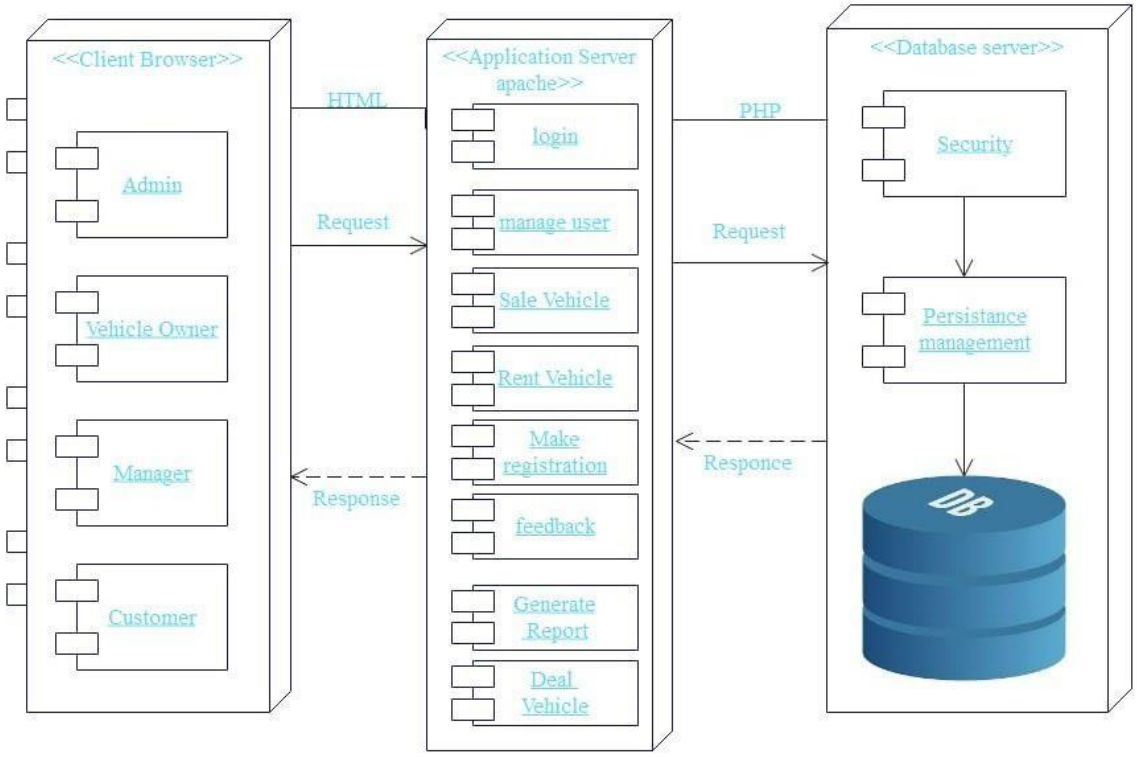


Figure 5.3 Deployment design

5.6. Detailed Class Diagram

A class diagram in the Unified Modeling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations, and the relationships among objects. Detail class diagram is a class diagram that includes attributes, methods, attribute data types, visibility of attributes and methods, inheritance, association, aggregation, composition, dependency, and municipality (cardinality and optimality). The following figure uses the UML class diagram to specify attributes and operations with their visibility information.

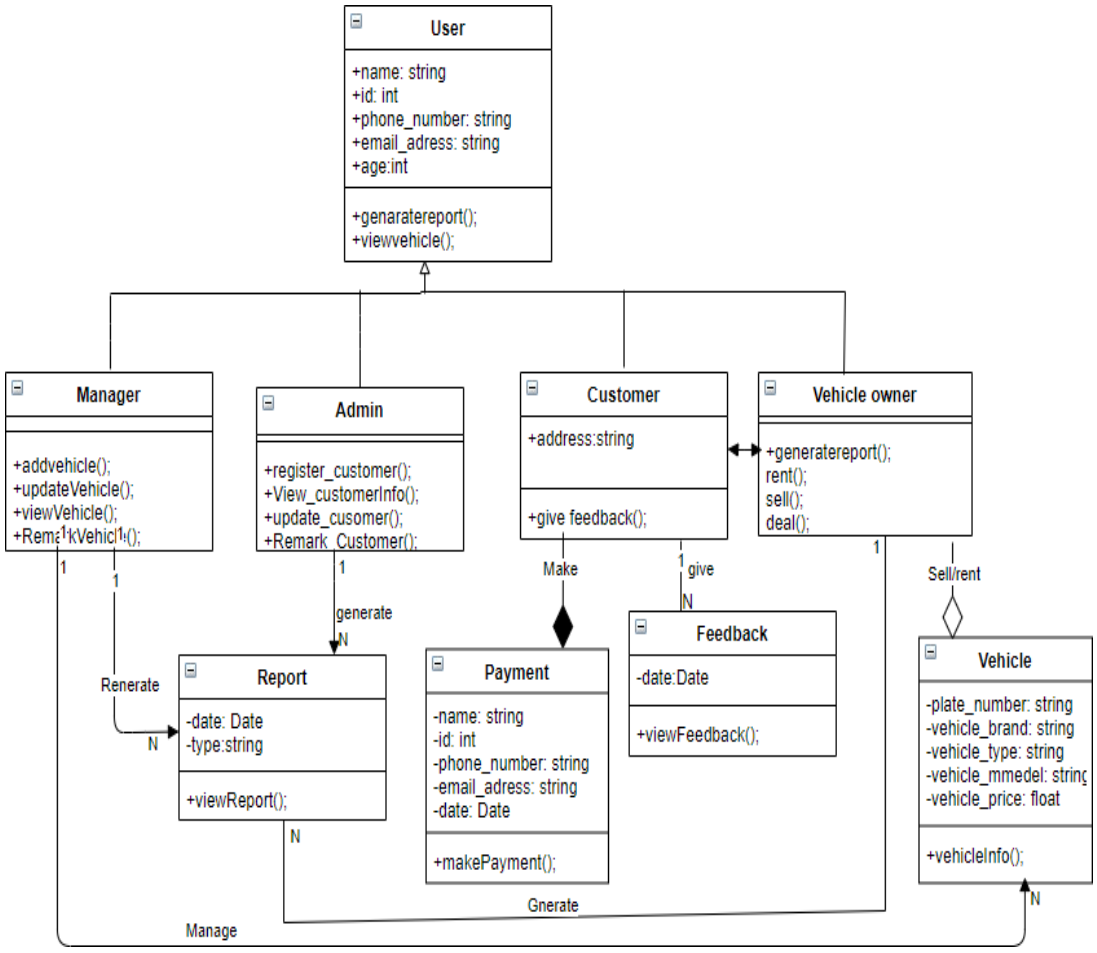


Figure 5.4: Detail class diagram

5.7. Persistent Data Management

Persistence is about storing data permanently. Persistence modeling is used to communicate the design of the database, usually the database to both the users and the developers. It is also used to describe the persistence data aspect of the system. Persistence data encapsulate the capability to store, retrieve, and delete objects/data permanently without revealing details of the underlying storage technology. In the current database system, we have used different tables as object and each object is related to each other and enforced by referential integrity by the use of the foreign and primary key. This schema enables data manipulation activity such as select, search, delete, update on the database. We use MySQL database the following tables indicate the persistence data management of the proposed system.

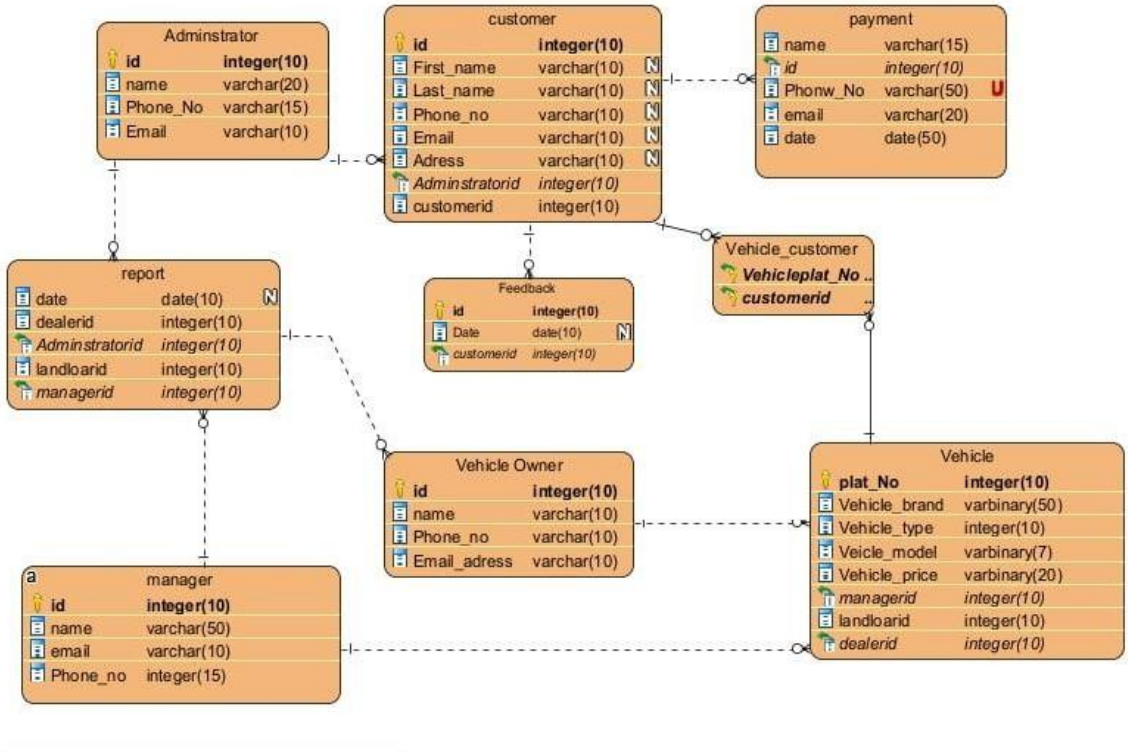


Figure 5.5: Persistent data management

5.8. Access Control and Security

Access control is a security technique that regulates who or what can view or use resources in a computing environment. Its objective is to ensure that only authorized individuals gain access to information or systems services necessary to undertake their duties.

In the proposed system, different actors have access to different information and data. Access control and security specify what the user can access or what cannot perform by some users. This access control is verified by a username and password. The proposed system follows a multi-user system. In a multi-user system, different actors have access to different functions and data.

Table 5.1: Privilege given to the actor of the system

Actor	**Privilege**
Administrator	✓ Login ✓ Create account ✓ Reset password ✓ Remark account ✓ View users ✓ Update account ✓ Generate report ✓ Logout
Manager	✓ Login ✓ Add vehicle ✓ Remark vehicle ✓ Update vehicle ✓ View vehicle ✓ Generate report ✓ Logout
Vehicle owner	✓ Login ✓ Deal vehicle ✓ View vehicle ✓ Rent vehicle ✓ Sell vehicle ✓ Generate report ✓ Logout

Customer	✓ Login ✓ Search vehicle ✓ View vehicle ✓ Get service ✓ Make payment ✓ Give feedback ✓ Logout

5.9. Packages

This section describes the decomposition of subsystems into packages and the file organization of the code. This includes an overview of each package, its dependencies with other packages, and its expected usage.

Package is an organized and functionality-based set of related interfaces and classes. Packages organize classes that belong to the same category or provide similar functionality.

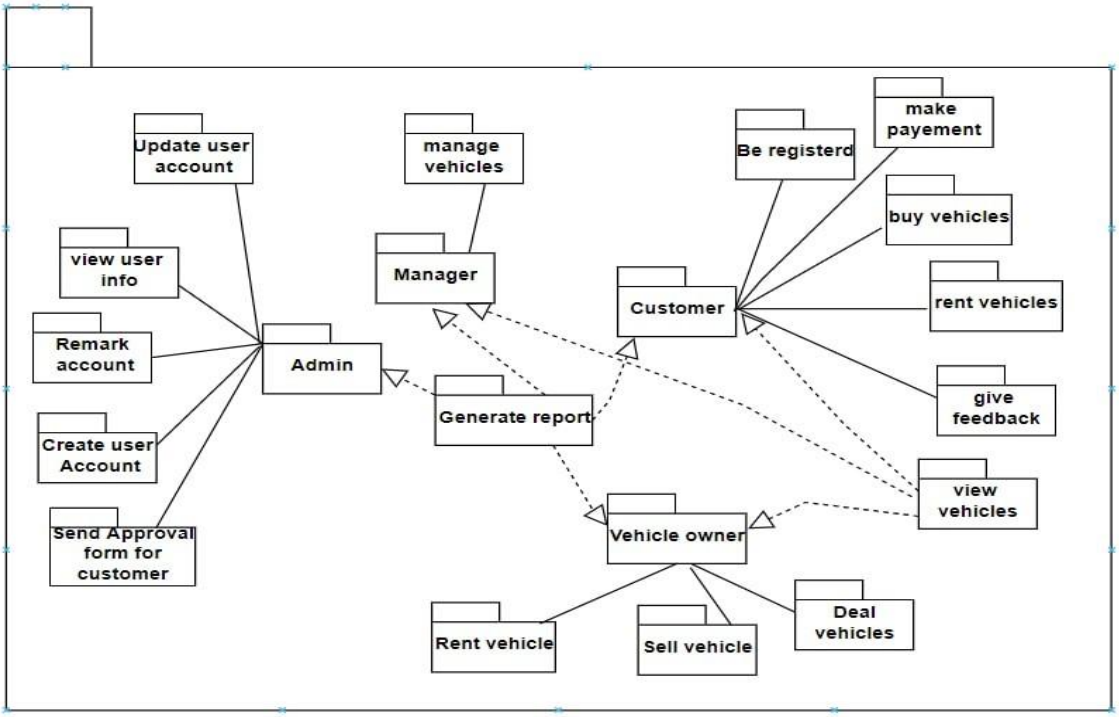


Figure 5.6: Package diagram

5.10. Algorithm Design

Algorithms are designed to show the flow of programs in the system. They are semantic driven rather than syntax driven. That means, the rule of syntax is not respected as other programming language but it has a complete meaning as that of syntax-based programming language. Also, Algorithms show the flow and steps of logic in each function. This design part is important in the coding part of the implementation. Some of the algorithms are listed below. Example:

Algorithm for login:

Step 1: Users open home page and click login link.

Step 2: Enter email and password.

Step 3: system validate Email and Password.

Step 4: if it‟s valid

Step 5: logged into the system.

Step 6: else, error message is displayed.

Algorithm for registration

Step 1: browse the home page and click register link.

Step 2: fill the registration form.

Step 3: if input information is valid.

Step 4: the user is added to database and display successfully added message.

Step 5: else invalid input

Step 6: display error message.

Algorithms for add user:

Step 1: browse the home page and click add link.

Step 2: enter user information.

Step 3: if input information is valid.

Step 4: the user is added to database and home page is displayed.

Step 5: else, display error message.

Algorithms for search:

Step 1: browse the home page and search link.

Step2: enter vehicle type.

Step 3: the system validates the input from database

Step 4: if users enter valid vehicle number.

Step 5: display search information

Step 6: else vehicle type is invalid.

Step 7: display error message

5.11. User Interface Design

Figure 5.7: GUI for homepage

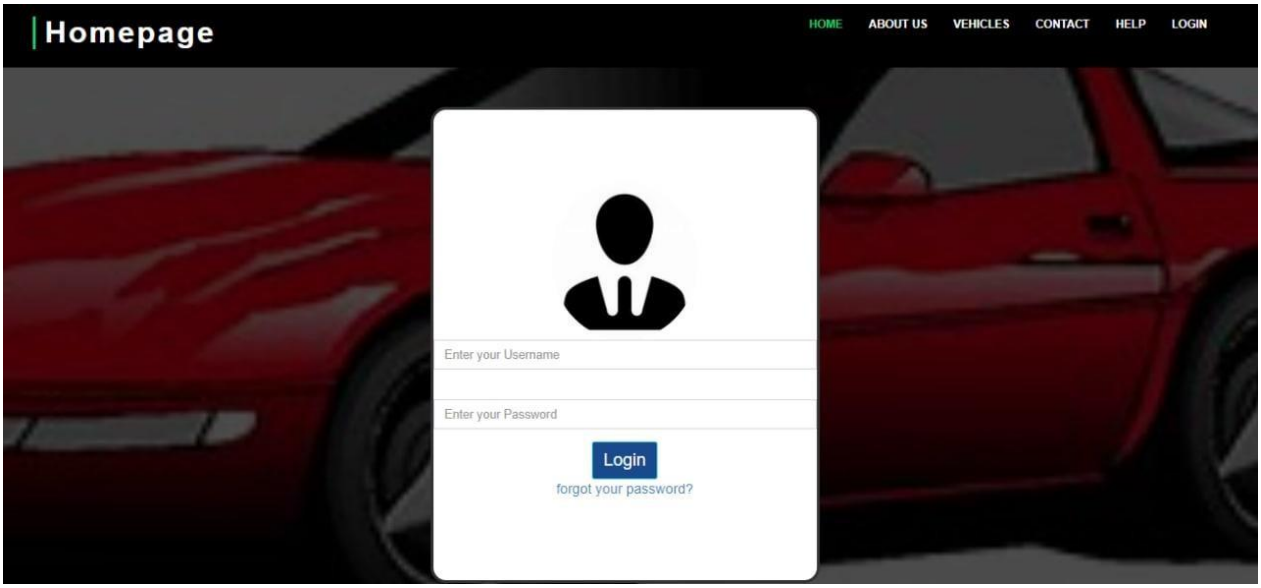


Figure 5.8: GUI for user login

CHAPTER SIX

6. IMPLEMENTATION AND TESTING

In this chapter, we discuss all about implementation of database, detailed class diagram, application server, application security. The implementation phase in the software life-cycle is where the actual software is implemented. The result of this phase consists of source code, together with documentation to make the code more readable. This is what we call software implementation. The purpose of these activities is to convert the final physical system specification into working model with reliable software and hardware, document the work that has been done and provide help for current and future users and take care of the system.

6.1 Implementation of the Database

```
-- phpMyAdmin SQL Dump
-- version 5.0.4
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Jun 11, 2022 at 09:10 PM
-- Server version: 10.4.17-MariaDB
-- PHP Version: 8.0.2

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
*/;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `vms`

-- --------------------------------------
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
START TRANSACTION;
SET time_zone = "+00:00";

CREATE TABLE `comment` (
```

```
  `cid` int(11) NOT NULL,
  `name` varchar(50) NOT NULL,
  `message` varchar(200) NOT NULL,
  `status` varchar(50) NOT NULL,
  UNIQUE KEY `cid` (`cid`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `sellinfo` (
  `sid` int(11) NOT NULL,
  `uid` int(11) NOT NULL,
  `customername` varchar(50) NOT NULL,
  `phonenumber` varchar(15) NOT NULL,
  `bankaccountno` varchar(50) NOT NULL,
  `address` varchar(50) NOT NULL,
  `username` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `profile` mediumblob NOT NULL,
  `vehicle_id` int(11) NOT NULL,
  `vehiclename` varchar(50) NOT NULL,
  `vehiclemodel` varchar(50) NOT NULL,
  `vehicleimage` mediumblob NOT NULL,
  `date` date NOT NULL,
  `sellprice` double NOT NULL,
  `message` varchar(50) NOT NULL,
  `status` varchar(50) NOT NULL,
  UNIQUE KEY `sid` (`sid`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `rentinfo` (
  `rid` int(11) NOT NULL,
  `uid` int(11) NOT NULL,
  `customername` varchar(50) NOT NULL,
  `phonenumber` varchar(15) NOT NULL,
  `bankaccountno` varchar(15) NOT NULL,
  `address` varchar(50) NOT NULL,
  `username` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `profile` varchar(50) NOT NULL,
  `vehicle_id` int(11) NOT NULL,
  `vehiclename` varchar(50) NOT NULL,
  `vehiclemodel` varchar(50) NOT NULL,
  `vehicleimage` varchar(50) NOT NULL,
  `startdate` date NOT NULL,
  `enddate` date NOT NULL,
  `rentalpriceperday` double NOT NULL,
  `totalrentalprice` double NOT NULL,
```

```
  `message` varchar(50) NOT NULL,
  `status` varchar(10) NOT NULL,
  `action` varchar(50) NOT NULL,
  UNIQUE KEY `rid` (`rid`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `report` (
  `rpid` int(11) NOT NULL,
  `Report_date` date NOT NULL,
  `sender` varchar(50) NOT NULL,
  `reciever` varchar(50) NOT NULL,
  `reporttype` varchar(50) NOT NULL,
  `status` varchar(10) NOT NULL,
  UNIQUE KEY `rpid` (`rpid`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `employee` (
  `id` int(11) NOT NULL,
  `cid` int(11) NOT NULL,
  `sid` int(11) NOT NULL,
  `rid` int(11) NOT NULL,
  `rpid` int(11) NOT NULL,
  `FirstName` varchar(50) NOT NULL,
  `LastName` varchar(50) NOT NULL,
  `Sex` varchar(6) NOT NULL,
  `age` int(11) NOT NULL,
  `PhoneNumber` varchar(13) NOT NULL,
  `regstrationdate` date NOT NULL,
  `address` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  `role` varchar(32) NOT NULL,
  `bankaccountno` varchar(15) NOT NULL,
  `accountbalance` double NOT NULL,
  `photo` varchar(50) NOT NULL,
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `confirmpassword` varchar(50) NOT NULL,
  `status` int(11) NOT NULL,
  UNIQUE KEY `id` (`id`) USING BTREE,
  CONSTRAINT `employee_ibfk_1` FOREIGN KEY (`cid`) REFERENCES `comment` (`cid`)
ON DELETE CASCADE,
   CONSTRAINT `employee_ibfk_2` FOREIGN KEY (`sid`) REFERENCES `sellinfo` (`sid`)
ON DELETE CASCADE,
   CONSTRAINT `employee_ibfk_3` FOREIGN KEY (`rid`) REFERENCES `rentinfo` (`rid`)
ON DELETE CASCADE,
```

```
  CONSTRAINT `employee_ibfk_4` FOREIGN KEY (`rpid`) REFERENCES `report` (`rpid`)
ON DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;


CREATE TABLE `uplodedvehicle` (
  `date` date NOT NULL,
  `uid` int(11) NOT NULL,
  `vname` varchar(50) NOT NULL,
  `vbrand` varchar(50) NOT NULL,
  `vmodel` varchar(40) NOT NULL,
  `vcompany` varchar(50) NOT NULL,
  `rentalprice` double NOT NULL,
  `saleprice` double NOT NULL,
  `noseats` int(11) NOT NULL,
  `mandate` date NOT NULL,
  `image` varchar(40) NOT NULL,
  `availablity` varchar(20) NOT NULL,
   UNIQUE KEY `uid` (`uid`) USING BTREE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE `vehicle` (
  `id` int(11) NOT NULL,
  `rid` int(11) NOT NULL,
  `sid` int(11) NOT NULL,
  `uid` int(11) NOT NULL,
  `vehicle_name` varchar(50) NOT NULL,
  `vehicle_brand` varchar(32) NOT NULL,
  `vehicle_model` varchar(50) NOT NULL,
  `vehicle_company` varchar(50) NOT NULL,
  `rentalprice` varchar(13) NOT NULL,
  `saleprice` varchar(50) NOT NULL,
  `Number_of_seats` int(11) NOT NULL,
  `manfactured_date` date NOT NULL,
  `image` varchar(100) NOT NULL,
  `action` varchar(10) NOT NULL,
   UNIQUE KEY `id` (`id`) USING BTREE,
  CONSTRAINT `vehicle_ibfk_1` FOREIGN KEY (`uid`) REFERENCES `uplodedvehicle`
(`uid`) ON DELETE CASCADE,
   CONSTRAINT `vehicle_ibfk_2` FOREIGN KEY (`sid`) REFERENCES `sellinfo` (`sid`) ON
DELETE CASCADE,
   CONSTRAINT `vehicle_ibfk_3` FOREIGN KEY (`rid`) REFERENCES `rentinfo` (`rid`) ON
DELETE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

6.2. Implementation of the Class Diagram

Table 6.1 Implementation of the Class Diagram

Identifier type	Online Vehicle Management System
Method	+Create Account (); +Update Account (); +Remark Account (); +Register vehicle (); +Update vehicle (); +Delete vehicle (); +View vehicle (); -Write Comment (); -Read Comment (); +Sell Vehicle(); +Buy Vehicle(); -Make Payment(); +Search Vehicle(); #Manage vehicle ();
Variables	+Int ID +Int Age +Varchar Name +Email Email +Varchar message +Varchar vehicle +Date Date +Varchar Price +Blob Image

6.3. Configuration of Application Server

We use XAMPP server since we are familiar with it and it bundle (MySQL, PHP) by installing these two components locally, we can build and test a web system easily.

First, download XAMPPServer from https://xampp-windows.en.softonic.com/download
Once the download process is complete, follow the following simple steps:

- Go to the Downloads folder and locate the XAMPP server installer file.
- Double-click the XAMPP server installer file.
- Click “Run” to initiate the installation process.
- A XAMPP Server wizard screen will pop up. Select “Next” to continue with the installation process
- On the next screen, check “I accept the agreement” at the bottom of the screen”. Now Click “Next.”
- Choose a folder where you would like to install the XAMPP server. In most cases, it‟s Just good to leave it as it is. Now click “Next” to continue.
- On the box adjacent to “Launch XAMPP Server” now and click Finish.

The next step is to create a MySQL database. Now launch XAMPP and you‟ll see start.

- Click on Apache and MySQL server and start it.
- Now click on MySQL „Admin‟ and open phpMyAdmin Server to create your database name.

Fig 6.1. Configuration of Application Server

- . Write your database and click „create‟ button.

6.4. Configuration of Application Security

Validate of the inputs sides allow the user to fill correct information in our system we validate all the input to enter the correct format.

Code for Validation of Registration form

$name = "/^[a-zA-Z]+$/";

```
$emailValidation   =   "/^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9]+(\.[a-z]{2,4})$/";
```

```
$number = "/^[0-9]+$/";
if(empty($c_name) || empty($c_email) || empty($c_pass) || empty($c_repass) ||
empty($c_image) || empty($c_city) || empty($c_kebele) || empty($c_contact) ||
empty($c_address)){
echo "
<div class='alert alert-warning'>
<a href='createaccount.php' class='close' data-dismiss='alert' aria
label='close'>×</a><b>Please Fill all fields..!</b>
</div>
";
exit();
} else {
if(!preg_match($name,$c_name)){
echo "
<div class='alert alert-warning'>
<a href='index.php' class='close' data-dismiss='alert' aria-label='close'>×</a>
<b>this $c_name is not valid..!</b>
</div>
";
exit();
}
if(!preg_match($emailValidation,$c_email)){
echo "
<div class='alert alert-warning'>
<a href='index.php' class='close' data-dismiss='alert' aria-label='close'>×</a>
<b>this $c_email is not valid..!</b>
</div>
";
exit();
}
```

```
if(strlen($c_pass) < 9 ){
echo "
<div class='alert alert-warning'>
<a href='index.php' class='close' data-dismiss='alert' aria-label='close'>×</a>
<b>Password is weak</b>
</div>
";
exit();
}
if(strlen($c_repass) < 9 ){
echo "
<div class='alert alert-warning'>
<a href='hom.php' class='close' data-dismiss='alert' aria-label='close'>×</a>
<b>Password is weak</b>
</div>
exit();
}
if($c_pass != $c_repass){
echo "
<div class='alert alert-warning'>
<a href='index.php' class='close' data-dismiss='alert' aria-label='close'>×</a>
<b>password is not same</b>
</div>
";
}
if(!preg_match($number,$c_contact)){
echo "
<div class='alert alert-warning'>
<a href='index.php' class='close' data-dismiss='alert' aria-label='close'>×</a>
<b>Mobile number $c_contact is not valid</b>
</div>
```

```
";
exit();
}
if(!(strlen($c_contact) == 10)){
echo "
<div class='alert alert-warning'>
<a href=index.php' class='close' data-dismiss='alert' aria-label='close'>×</a>
<b>Mobile number must be 10 digit</b>
</div>
";
exit();
    }
```

6.5. Implementation of User Interface

We implement the system we have developed has good and friendly user interface that user can easily navigate the system. The user interface is portion of an interactive computer system that user can communicate the system. Design of the interface includes any aspect of the system that is visible to user. The system has consistence user interface when we navigating through page.

Figure 6.2: User Interface for Login

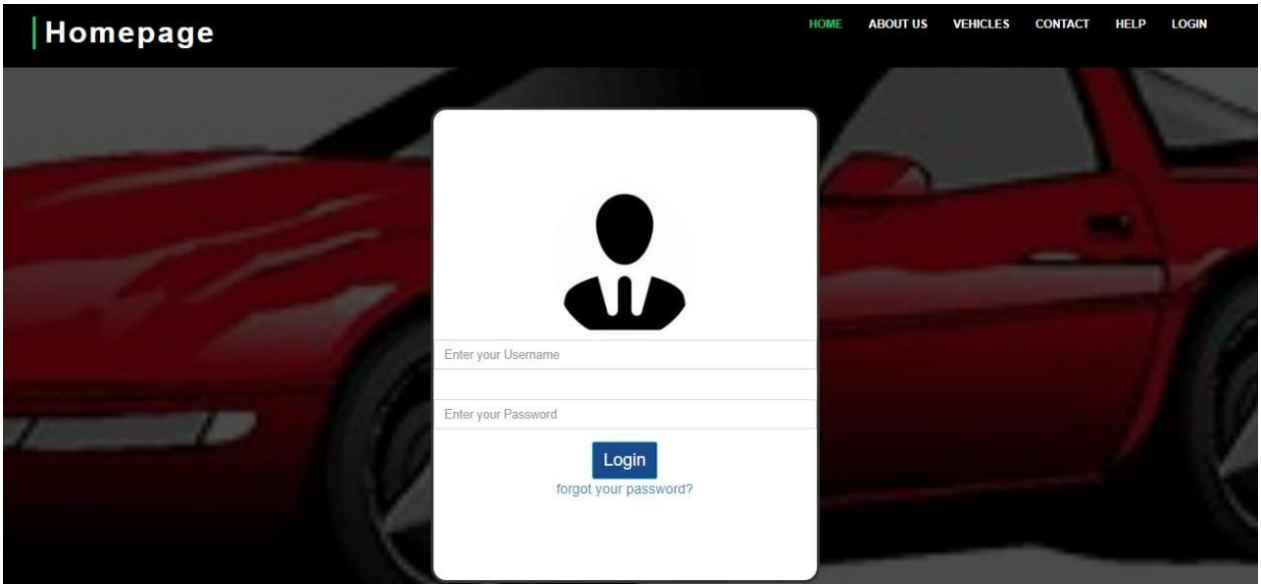


Fig 6.3.User Interface for Login

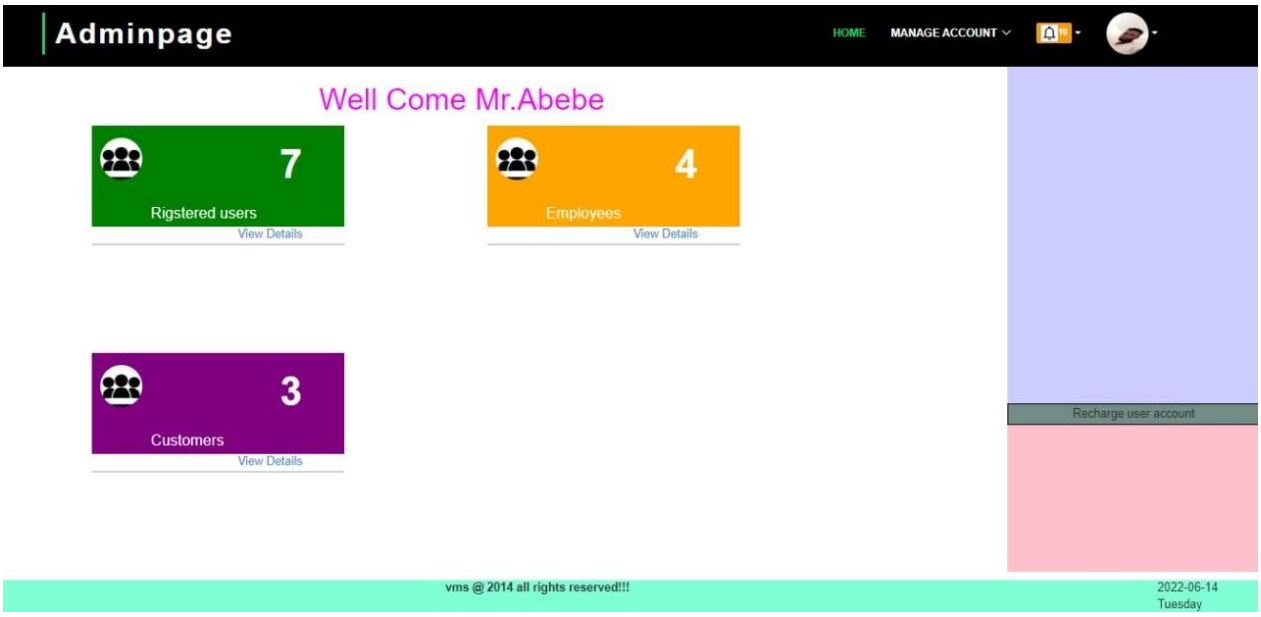


Figure 6.4: User Interface for Admin page

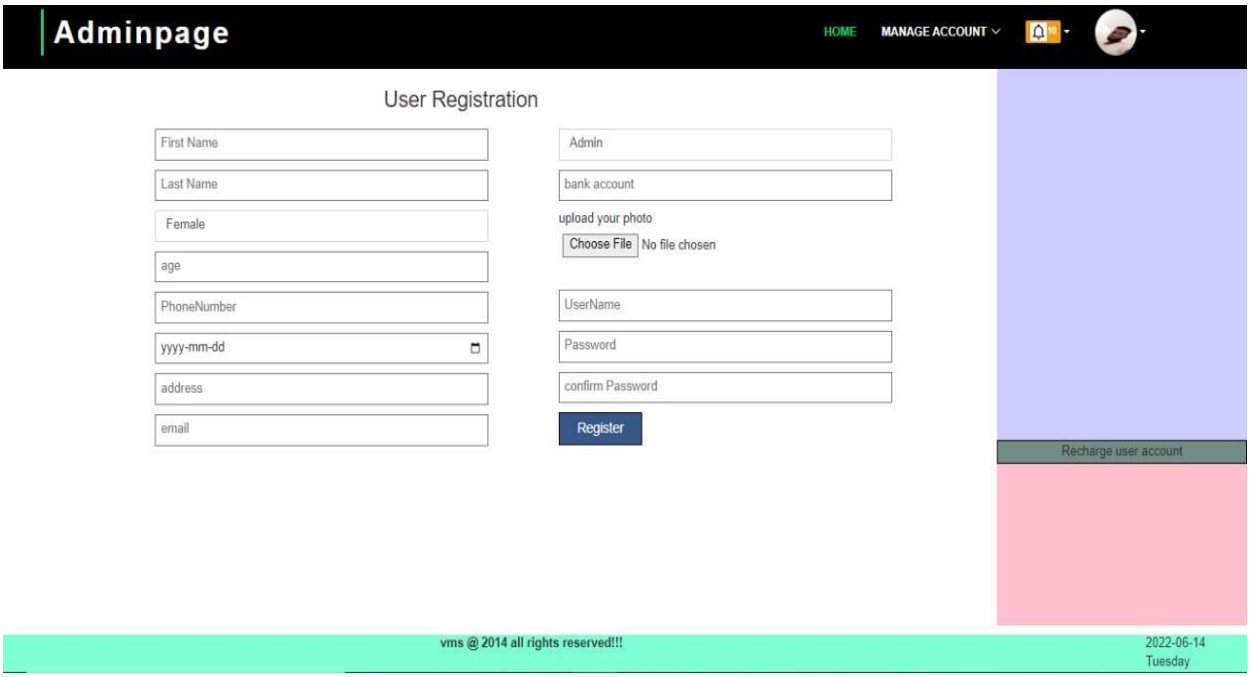


Figure 6.5: User Interface for User Registration form

6.6. Testing

Testing is finding out how well something works. In terms of human beings, testing tells what level of knowledge or skill has been acquired. In computer hardware and software development, testing is used at key checkpoints in the overall process to determine whether objectives are being met.

6.6.1. Test case

Test case is a specification of the inputs, execution conditions, testing procedure, and expected results that define a single test to be executed to achieve a particular software testing objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

6.6.2. Testing Tools and Environment

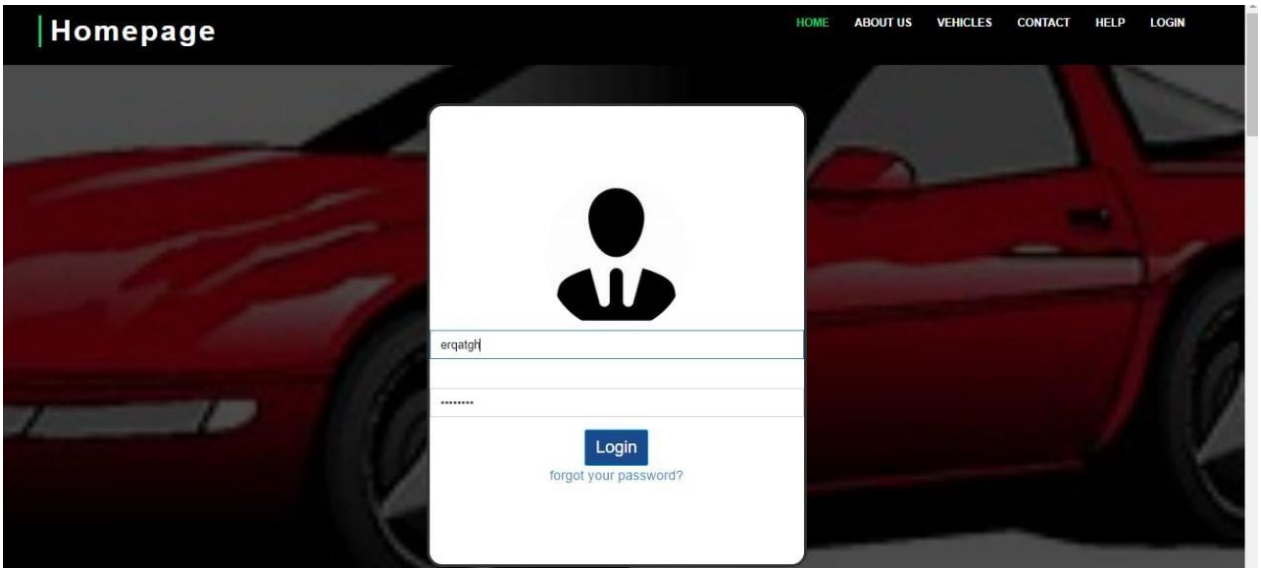
A testing environment is a setup of software and hardware for the testing teams to execute test cases.

- ✓ Editor (Sublime text3): - is used to edit PHP, HTML, JavaScript and CSS code.
- ✓ PHP: - It is used for sever side scripting. In contrast to other scripting language Php can run on various types of platforms, easy database connection, easy to use, open source.

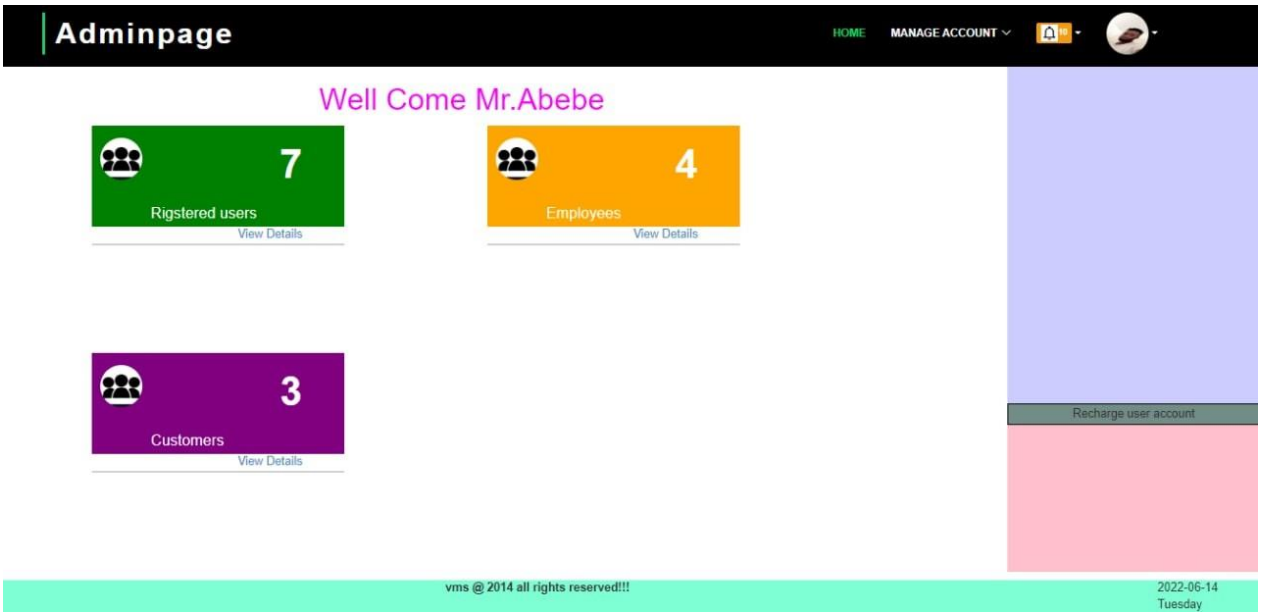
- ✓ XAMPP: - XAMPP allows to work on a local server and test a local copies of websites using PHP code and MySQL databases on window platform.
- ✓ Browser: - like Google chrome used to runs the system on the screen.

6.6.3. Unit Testing

In these types of testing our team tries to test by taking one functionality of the system when we do source code implementations. From those unit testing it includes language-specific programming errors such as bad syntax, logic errors or to test particular functions or code modules. From those tests our team try to test is that check that is employee is login based on their role or not. During these time users do not login based on their role. During these our team gets logical error.



After some try our team solves this problem and tests it that is working correctly.



6.6.4. Integration Testing

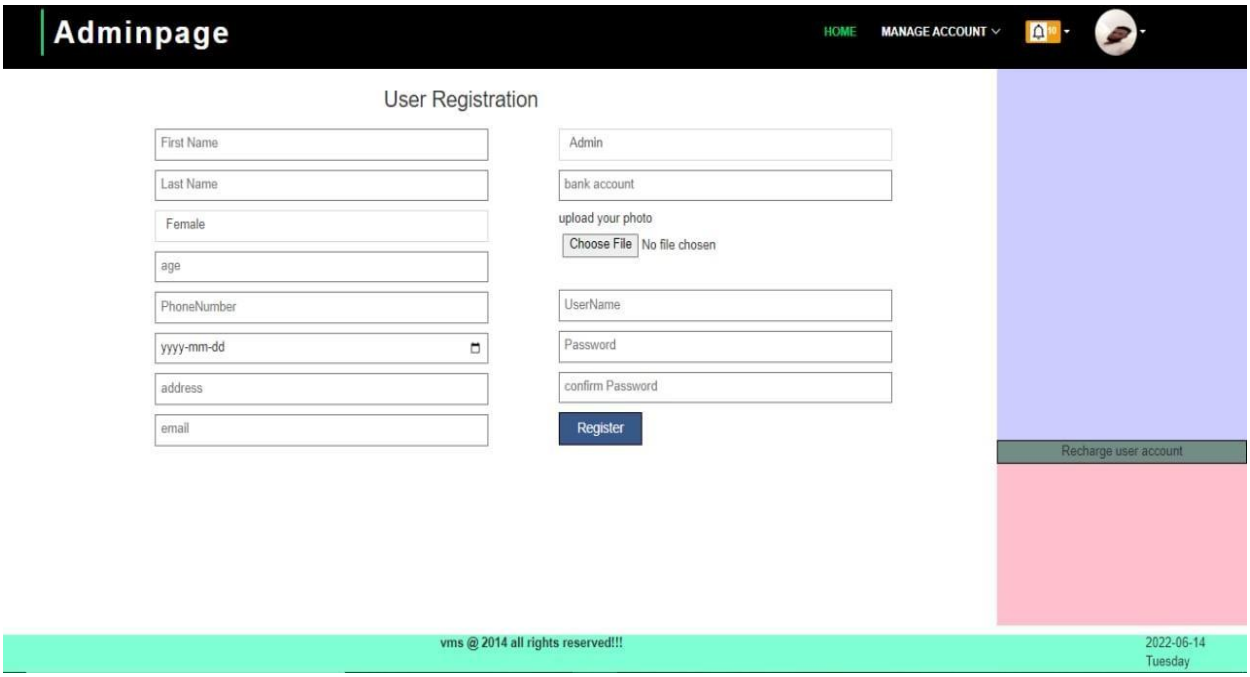
In this level of testing, our team examined how the different procedures work together to achieve the goal of the subsystem. The type of integration testing that we have follow bottom up. So, we integrate each component from single function to the main function incrementally.

Sample Tests:

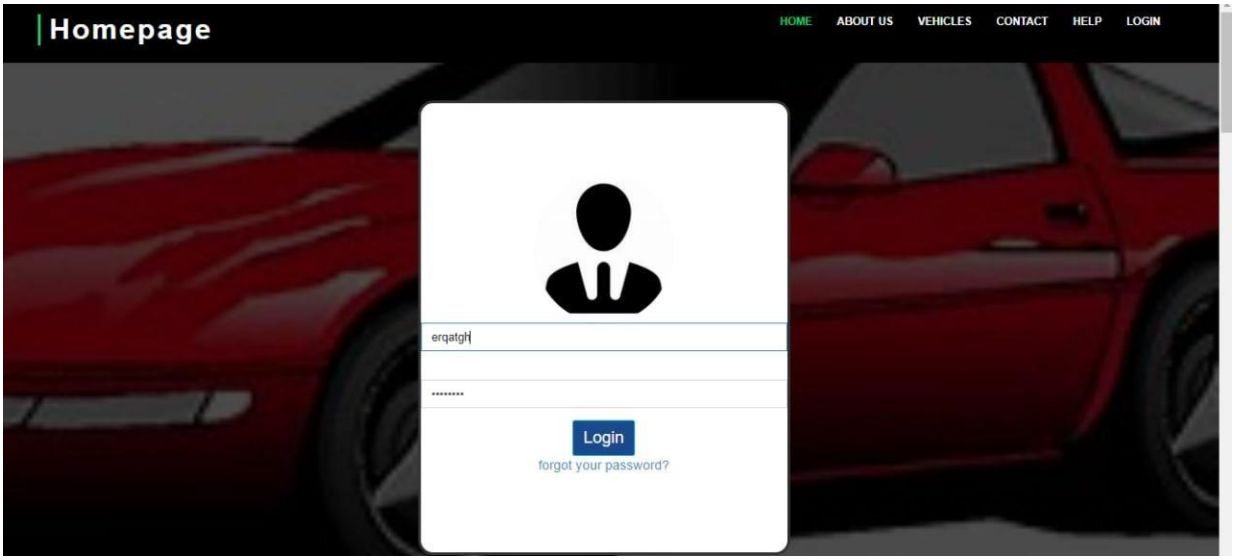
- ✓ Check the interaction between individual functionality which performs the specific tasks.
- ✓ Evaluate the functionality of the subsystem after combination all individual functionality.
- ✓ Identify the Independence of each subsystem with another subsystem.

These can be verified by the following trials for customer from registration to rent vehicle.

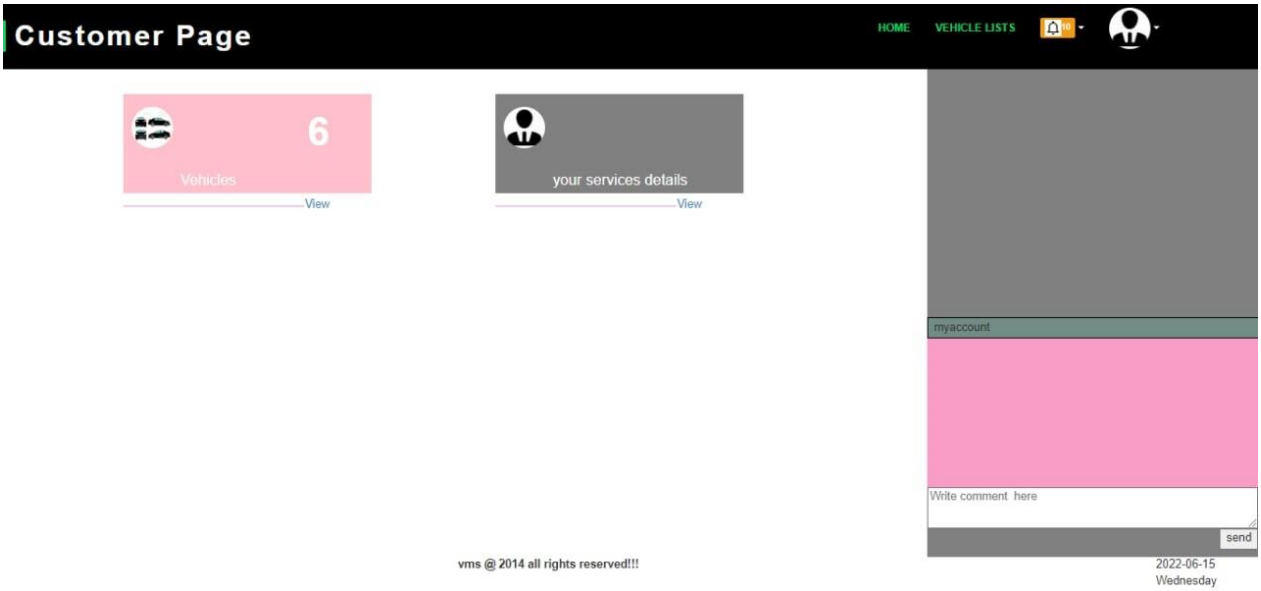
Admin login to his page and register customer



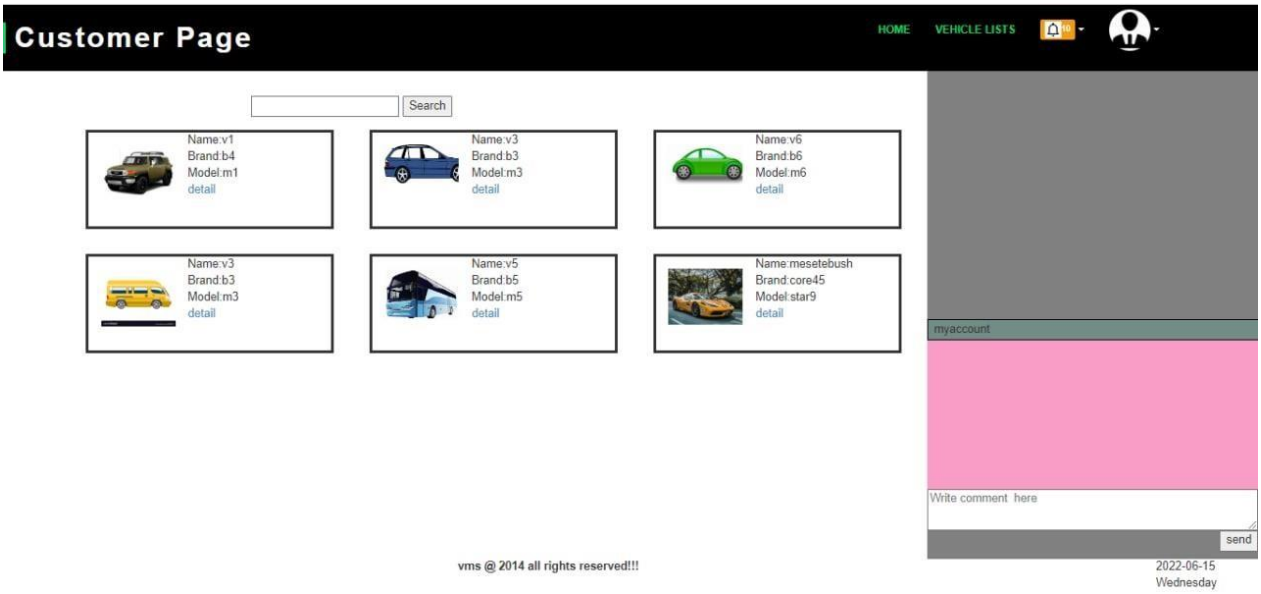
Manager login to his page and register vehicles. Then registered customer login to his/her page based on their username and password



If the username and password is correct it goes to its dashboard

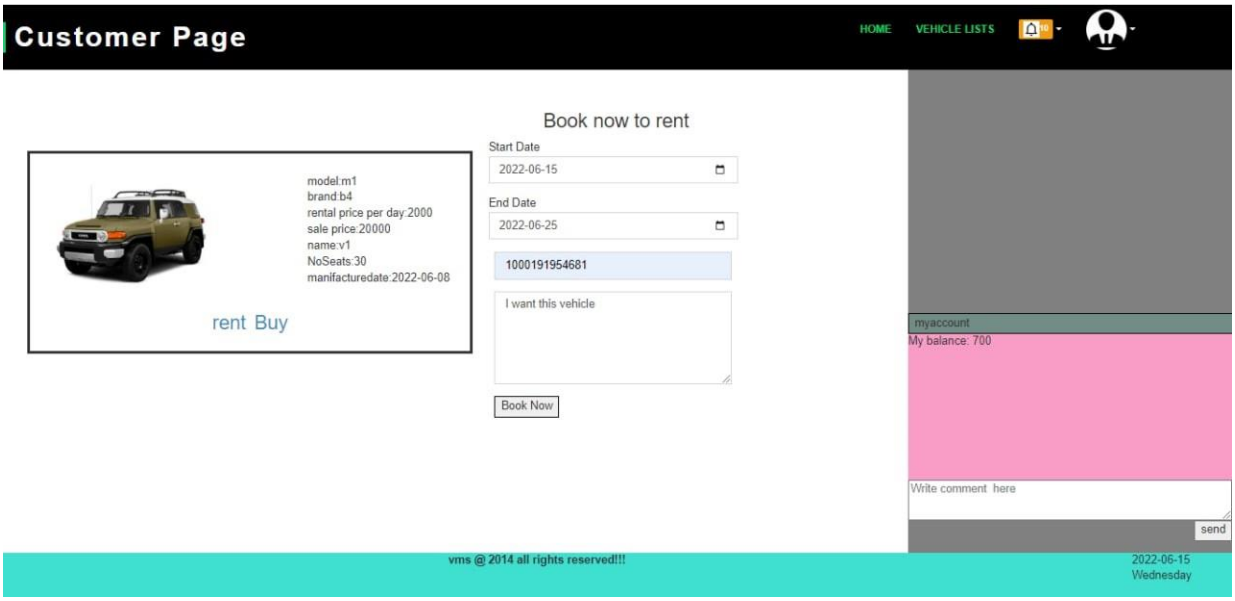


Next customer view vehicles

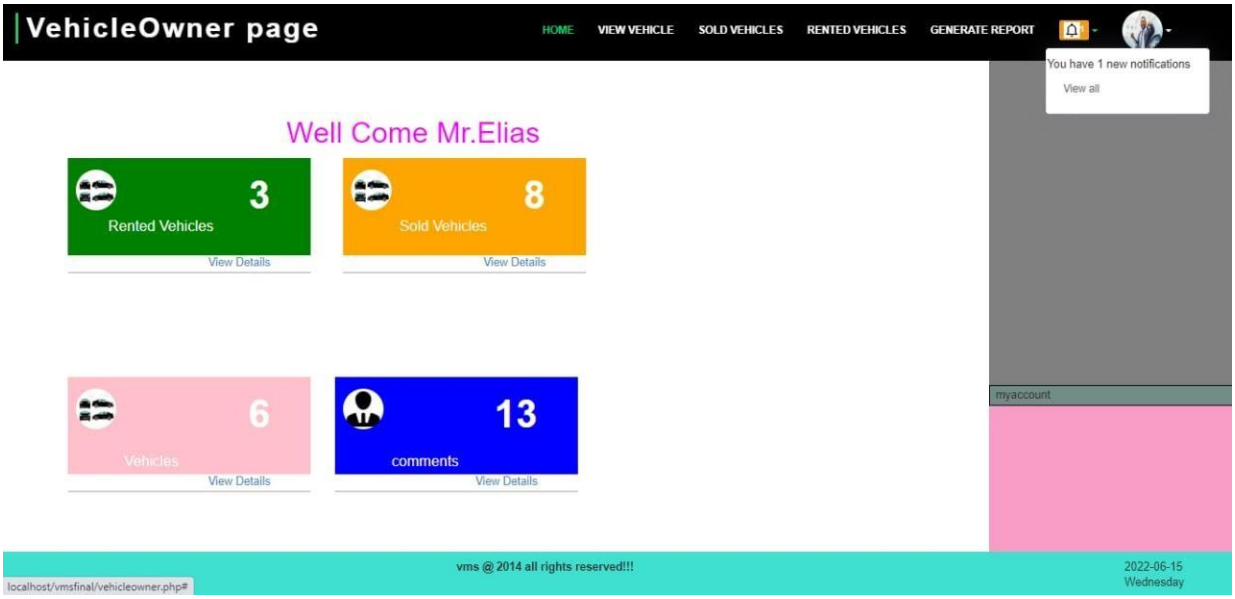


And next customer choose vehicle based on their need

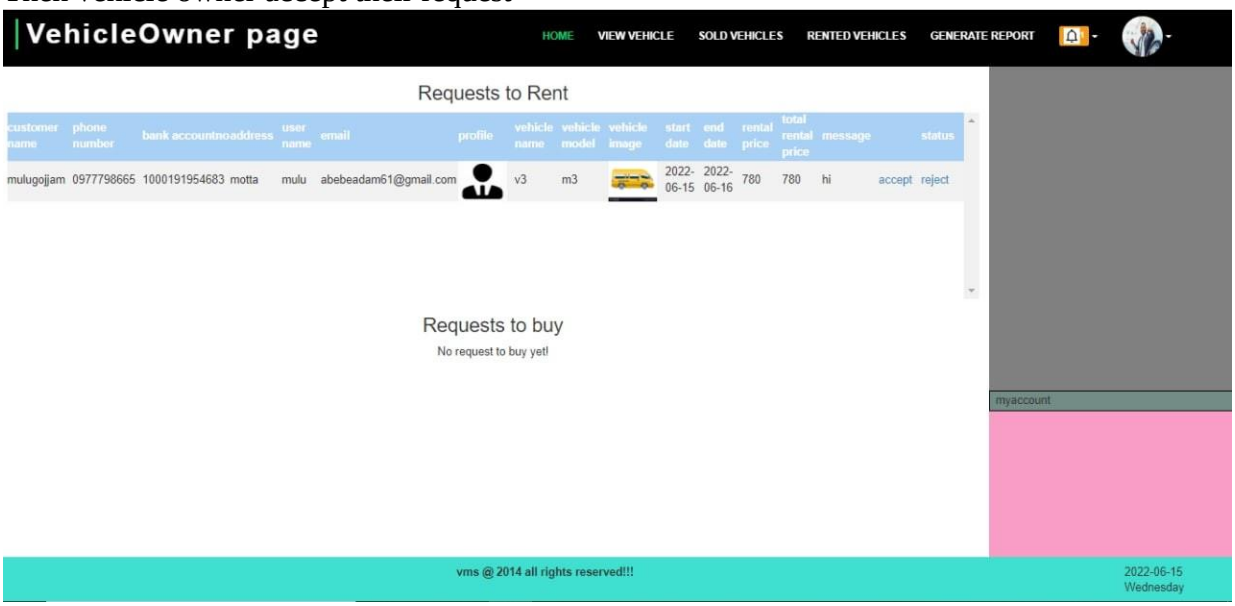
Then customer checks their account and fills the form and the last book their needed vehicle.



Customer request to vehicle owner vehicle owner get notification on his page



Then vehicle owner accept their request



These is all about how customer rent vehicles

6.6.5. System Testing

In this level of testing process, we have examined how the whole subsystems work together to achieve the desired goal (user‟s requirements of the system). The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, under this testing is mainly concerned with areas such as performance, security, validation. But we will more focus only on function validation and performance.

Sample Tests:

- ✓ Evaluate the functionality of the subsystem after a combination of individual subsystem whether it works correctly or not.
- ✓ Check the coherence and coupling of each subsystem.
- ✓ Check the overall functionality that achieves the user‟s requirement

CHAPTER – SEVEN

7. CONCLUSION AND RECOMMENDATION

7.1. Conclusion

The main objective of this project is to develop Online Vehicle management System. In order to solve different problems of the existing system we tried to develop this system. For instance, some problem related with time, labour and Security.

The developed system has also been aimed to improve those problems. Different tools and techniques have helped us a lot in assessing real user requirements and model the right system for the users to perform their activities. The proposed system enables to register users and accept their requesting and give service efficiently. Finally the group expects that the developed system will change the general customer service system and make it more reliable and efficient than the previous manual system.

7.2. Recommendation

According to the scope of our project the team develops web-based application. due to time limitation, we can‟t do all the tasks that are needed in the system so to improve the performance and functionality of the system our team believes that this system should be fully operationally by adding some functionality that are not included in the proposed system. We recommended also the next developer can include the following tasks: -

- Add IO concept on the system to control vehicles remotely.
- To integrate new online payment system like Yenepay and other.

References

[1] S. Kharanshu Bhavsar, "VEHICLE MANAGEMENT SYSTEM project is a web application which is developed in PHP platform.," 3 march 2020. [Online]. Available: https://www.projects.com. [Accessed december 2022].

[2] K. kumar, "project/php/3463/vehicle-management-system," 24 April 2020. [Online]. Available: http://lib.buet.ac.bd:8080. [Accessed 12 october 2022].

[3] A. A. Nain, "project/php/3463/vehicle-management-system," 4 may 2019. [Online]. Available: http://www.onlinesource.org. [Accessed 23 3 2022].

[4] A. Mercurio, "D.mercurio," [Online]. Available: https://www.itsourcecode.com/free_projects. [Accessed 26 February 2021].

[5] S. pavan, "A.PAVA," 5 Septmeber 2019. [Online]. Available: https://www.ijrte.org. [Accessed 21 Janaury 2022].

APPENDICES

APPENDIX I: Sample Source Code

Sample code for login

```
<?php
session_start();
include'connect.php';
if(isset($_POST['submit'])){
$username=$_POST['username'];
$password=md5($_POST['password']);
$result = mysqli_query($con,"SELECT * FROM user WHERE username='$username' and password='$password'") or die(mysqli_error());
$row=mysqli_fetch_array($result);
$status=$row['status'];
if($row>0){
$_SESSION['id']=$row['id'];
$_SESSION['username']=$row['username'];
$_SESSION['passwordd']=$row['password'];
if($status==0)
{
header("location:loginform.php?error=1");
}else
{
 if($row['role']=="customer" and $status==1)
{
        header("location:customer.php" );

}
else if($row['role']=="manager" and $status==1)
{
header("location:manager.php" );
```

```
}
else if($row['role']=="vehicleowner" and $status==1)
{
  header("location:vehicleowner.php");
}
else if($row['role']=="Admin" and $status==1)
{
header("location:admin.php");
 // echo $_SESSION['username'];
}
}
}
else{
        // header("location:loginform.php");
header("location:loginform.php?error=1");
}
// break;}}
}
?>
```

Sample code for search vehicle

```
<html>
<head>
<title> search data</title>
</head>

<body>
<form  action="" method="post" >
    <input type="text" name="id" paceholder="enter id to search">
     <input type="submit" value="Search" name="search">
```

```
</form>

<?php
include'connect.php';
if(isset($_POST['search']))
   {
$sr = $_POST['id'];
  $query = "SELECT * FROM vehicle WHERE vehicle_name= '$sr' || vehicle_brand='$sr' ||
vehicle_model='$sr' || vehicle_company='$sr' '';
$query_run = mysqli_query($con, $query);
$num=mysqli_num_rows($query_run);
if($num)
{
while($row=mysqli_fetch_array($query_run))
{
      $img=$row['image'];
      $vname=$row['vehicle_name'];
      $vmod=$row['vehicle_model'];
     $vbra=$row['vehicle_brand'];
      $vcom=$row['vehicle_company'];
     $renprice=$row['rentalprice'];
      $saleprice=$row['saleprice'];
     $noseat=$row['Number_of_seats'];
      $mandate=$row['manfactured_date'];
      ?>
      <div style=" margin-left: 70px; ">
      <div class="col-lg-4 col-md-6 portfolio-item filter-app">
      <div class="portfolio-wrap" style="float: left; padding: 15px;">
      <div style="border: solid; width: 600px; height: 250px; ">
      <figure style="float: left;">
```

```
        <a href=""><img src="image/<?php echo $img; ?>" class="img-fluid" alt="" width="300"
height="200"></a>
      </figure>
           <div >
            <?php echo "Name:".$vname ; ?><br>
            <?php echo "Brand:".$vbra ; ?><br>
            <?php echo "Model:".$vmod ; ?><br>
        <!--   <a href="detail.php ?id=<?php echo $row[3]?>" >detail</a> -->
        <a href="detail.php ?id=<?php echo $row[3]?>" >detail</a>
           </div>
       </div>
        </div>
       </div>
     </div>
     <?php
}
}
else
{
            echo "<script>
  alert('not found')
</script>";
}
}
?>
     </body>
</html>
```