



CAR RENTING SYSTEM

Software Engineering Project Report
Group-02

Report Submitted 1 January, 2023

A project submitted to Dr. Rudra Pratap Deb Nath, Associate Professor, Department of Computer Science and Engineering, Chittagong University (CU) in partial fulfillment of the requirements for the Software Engineering Lab Course. The project is not submitted to any other organization at the same time.

Table 1: Details of Group-16

Roll Id	Name	Sigature	Date	Supervisor Approval
19701074	Md Abu Noman sikdar	Noman	01.01.2023	
19701075	Md.Siam	Siam	01.01.2023	
19701044	Md.Jahangir Alam Jehad	Jehad	01.01.2023	

Abstract

This report will be focusing on Car Renting System. It will discuss the causes of the problem and then suggest a software solution. The report will first discuss the problem with a focus on the causes and possible solutions to these. This general view will then be used to create a problem statement for the software solution. Afterwards the architecture and interaction design of the solution will be formulated. The interaction design defines a design language which is a standard of the solutions interaction design. The architectural and interaction design are then used as a basis for the final solution. The product of the report is a software system. The last part of the report concludes and sets the future improvements for the program.

Preface

This report is written by Abu Noman Shawn, Md.Siam and Jahangir Alam Zihad. The intellectual property rights of this report and all of its contents belong to the members of the project. ll material gathered from third party resources will be referred to according to the IEEE standard.The source code for the software solution of the project, is open source. The code is freely available at GitHub.The colors for this program, were chosen by personal preference.The figures used in the report, have been created using online tool call visual paradigm, unless stated otherwise. A authors would like to thank participants of the usability test for their time and constructive critique, as well as the informants for the interviews and user stories, for their cooperation. The report makers would also like to thank Rudra Pratap Deb Nath for his help and constructive criticism throughout the project process.

Contents

1	Introduction	8
1.1	Project Description	9
1.2	Problem Statement	9
1.3	Aim and Objectives	9
2	Requirement Gathering and Analysis	10
2.1	Requirements Gathering	10
2.1.1	Interviews:	10
2.1.2	Document Analysis:	11
2.2	Types of Requirements	12
2.2.1	User Requirements:	12
2.2.2	System Requirements:	12
2.2.3	Functional Requirements:	12
2.2.4	Non-Functional Requirements:	14
3	Software Process Model	15
3.1	Suitable Models	15
3.2	Not Suitable Models for Our System	17
4	System Choice	18
4.1	Appreciate The Situation:	18
4.1.1	Rich Picture	18
4.2	Cultivate New Ideas	20
4.3	FACTOR	20
4.4	System Definition	21
5	Problem Domain	22
5.1	Classes	23
5.1.1	Physical	23
5.1.2	Person	23
5.1.3	Places	23
5.2	Class Diagram	24
5.3	Events	25
5.3.1	Event Definition:	25
5.3.2	Event Table	27
5.4	Event Traces	28
5.4.1	Hire Car:	28
5.4.2	Trip History	29
5.4.3	Passenger Profile	30

5.4.4	Add Car	31
5.4.5	Add Driver	32
5.4.6	Notification	33
5.4.7	Car Rent Profile	34
5.5	Behavior	35
6	Application Domain	36
6.1	Usage	36
6.1.1	Actors	36
6.1.2	Actor Specification	37
6.1.3	Use Case Diagram	38
6.2	Functions and Complexity	39
7	System Architecture Design	42
7.1	Priority of Criteria	42
7.2	Architectural Design	44
7.2.1	Architecture Component	45
7.2.2	Model Component	46
8	User Interface Design	47
8.1	Low Fidelity Prototype	47
8.2	High Fidelity Prototype	48
9	Implementation	59
10	Testing	75
10.1	Unit Testing	76
10.2	Intregation Testing	76
10.3	System Testing	76
10.4	Acceptence Testing	77
11	Software Deployment	84
11.1	Software Deployment Methodologies	84
11.2	Software Deployment For Our System	84
12	Conclusion and Future Work	85
	Appendices	86
A	Interviews	86
13	Bibliography	88

List of Figures

1	The Waterfall Model	16
2	The Iterative Model	16
3	Rich Picture of Car Renting System	18
4	Rich Picture of Car Renting System	19
5	Class Diagram of Car Renting System	24
6	Statemachine diagram for the Hire Car	28
7	Statemachine diagram for the Trip History.	29
8	Statemachine diagram for the Profile	30
9	Statemachine diagram for the Add Car	31
10	Statemachine diagram for the Add Driver	32
11	Statemachine diagram for the Notification	33
12	Statemachine diagram for the Notification	34
13	Use Case Diagram	38
14	Architecture Component Diagram	45
15	Architecture Component Diagram	46
16	Login Page	48
17	Hire Car Page	49
18	Request Page	50
19	Notification Page	51
20	Trip page	52
21	Profile Page	53
22	Car Owner Home Page	54
23	Add Car page	55
24	Update Car Page	56
25	Car Owner Notification Page	57
26	Car Owner Profile Page	58
27	Testing Sequence	75
28	System Testing	76
29	Search Available Car	78
30	See Notification	79
31	See Trip History	80
32	Profile update	81
33	Owner See Notification	82
34	Owner See Notification	83

List of Tables

1	Details of Group-16	1
---	-------------------------------	---

2	Event table of Car renting system	27
3	Functions	40
4	Priorities for different criteria of Car Renting System	42

Listings

1	main.dart file containing "User" class	59
2	hirecar.dart file containing the "HireCar" class	62

1 Introduction

Our project is about car renting system. Car renting system is developed for customer so that they can book car in online as their need. Also a passenger can see the details about car and the rating of the car renting service center.

Offline car renting system has many problems. For example maintenance of the car rent center, passenger don't see about the details of a car what service the car rent center provide. For this reason we developed car renting system so that a passenger can easily search a car rent center in online and send booking request for available car. On the other hand, a car rent center can easily maintain his services.

Finally, we developed a software for car renting system to use this. A user have to register at first a passenger or car rent center. A passenger can search a car rent center with proper information for example pickup location, drop-off location, pickup time ,drop-off time and so on. After providing information a passenger can search car rent center. Then a list of car rent center shown to the passenger. Also a passenger can search with location of the car rent center from the list. A passenger can choose any car rent center. When a passenger choose a car rent center he directly enter into the car rent center interface. Here he will see a list of cars with proper information. Besides he can send request for car. On the other side a car rent center can add ,delete or update car and driver. Besides this it can see the current trip details. Also it can see all the request send by passengers. Then it can accept or delete request. As a result a notification is send to the passenger who requesting for car. If the it accept the request of passenger ,he will able to see the trip details in trip history .Then the passenger can rate the car rent center.

However our software can solve the problem facing with offline car renting system but a passenger can't choose the location by selecting location from google map. Besides car rent center can't see the location of car that is on a trip.

In most cases, when a passenger try to hire a car he can't choose a car according to his preferences. But our software can provide this benefits. Besides, this in offline base car renting system a passenger don't know about the rating of car. As a result he can't guess the car is good or bad for hire. But in online base car rent center a passenger can see the rating of car rent center. Also a passenger can rate a car after completing the trip. Moreover, our software provide a user friendly system for the passenger to hire a car.

1.1 Project Description

This project is designed so as to be used by Car Rental Company specializing in renting cars to customers. It is an online system through which customers can view available cars, register, view profile and book car. The advancement in Information Technology and Internet penetration has greatly enhanced various business processes and communication between companies (services provider) and their customers of which car rental industry is not left out.

This Online Car Renting System is developed to provide the following services:

- Enhance Business Processes: To be able to use internet technology to project the rental company to the global world instead of limiting their services to their local domain alone, thus increase their return on investment (ROI).
- Online Vehicle Reservation: A tool through which customers can reserve available cars online prior to their expected pick-up date or time.
- Customer's registration: A registration portal to hold customer's details, monitor their transaction and used same to offer better and improve services to them.
- Group bookings: Allows the customer to book space for a group in the case of weddings or corporate meetings (Event management).
- The content management system (CMS) for managing the content of the cars.
- The data security system.
- Reporting of the cars, booking etc.

1.2 Problem Statement

A car rental is a vehicle that can be used temporarily for a fee during a specified period. Getting a rental car helps people get around despite the fact they do not have access to their own personal vehicle or don't own a vehicle at all. The individual who needs a car must contact a rental car company and contract out for a vehicle. This system increases customer retention and simplifies vehicle and staff management.

1.3 Aim and Objectives

- To produce a web-based system that allow customer to register and reserve car online and for the company to effectively manage their car rental business.
- To ease customer's task whenever they need to rent a car.

2 Requirement Gathering and Analysis

A requirement is a condition or a capability needed by a user to solve a problem or achieve an objective. It is a description of the features and functionalities of the target system. A system development project usually starts by defining the requirements of the system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from the client's point of view. The requirements form the basis for development.

2.1 Requirements Gathering

Requirements are some criteria collected from users. It basically reflects what the user wants. There are different kinds of techniques to gather requirements. we describe some effective requirement-gathering processes. In this system, requirements are collected in two ways.

2.1.1 Interviews:

Interview is the most important part to collect requirements from users. We have arranged a meeting with Abdul Kader, a customer who has faced some difficulties in travelling. Different types of questions have been asked on our behalf. Here are some notable questions.

1. We heard you went to the Dhaka recently, How do you go there?

Ans: I went to Dhaka by renting a car.

2. When you reached Dhaka ?what kind of problems did you face there?

Ans: I reached Dhaka in late. But I need to reach in time.

3. Do you have to face any problem to rent the car?

Ans: I have to face huge problem. First day when I go to a renting office for renting there was no available car. For this reason Next day I again go and rent a car. But I can not find my favourite car.

4. What about the renting fees?

Ans: Fees are very high because there was no option for me. 5. What about the driver? Is he good driver?

Ans: The Driver is not Experienced.

2.1.2 Document Analysis:

Requirements elicitation phase of a project. It describes the act of reviewing the existing documentation of comparable systems in order to extract pieces of information that are relevant to the current project and therefore should be considered project requirements. Analyzing some documents online, we find closest car rental company for car renting.

Existing System:

- An existing system can provide manually paper work or excel sheet to track the booking and registered vehicles details.
- The user has to go in the office where the user can get the car on rent and book their car. Most of the time user does not get a sight of the car in which he is planning to travel. This results in compromising the travel comfort.
- In the existing system, you cannot provide feedback of the user to the admin directly. The user gets fluctuation every time he/she travels.
- Maintaining excel sheet or paper book record of reservation is very laborious work. Chances of error are more. No automation involves which means they are a very slow to process.

2.2 Types of Requirements

Various types of requirements related to our project are discussed below.

2.2.1 User Requirements:

User requirements are the abstract statement of the system. It's written for customers. The user demands the requirements our system will provide.

- Textual information in an emergency.
- Nearby repair centers and office locations.
- Automatic call and massaging service.

2.2.2 System Requirements:

After collecting requirements, we have to analyze them. By analyzing user requirements, we get system requirements. System requirements are description which determine the functionality of the system.

2.2.3 Functional Requirements:

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.

Reservation

- The system must allow the customer to register for reservation.
- The system shall allow the customer to view detail description of particular car.
- The system must notify on selection of unavailable cars while reservation.
- The system must notify on selection of unavailable cars while reservation.
- The system shall allow the employee to view reservations made by customers.
- The system shall presents information on protection products and their daily costs, and requests the customer to accept or decline regulation terms during reservation.

Log in

- The system should allow manager to login to the system using their username and password.
- The system should allow employee to login to the system using their username and password.
- The system shall allow the manager to create new user account.
- The system shall allow manager to change account password.
- The system shall allow staff to logout.
- The system shall allow manager to logout.

Car

- The system should allow staff to register new cars.
- The system shall allow staff to select cars in the list.
- The system shall allow customer staff to Search cars by specific record.
- The system shall allow staff to update information of the car in need of modification.
- The system shall allow staff to display all rented car.
- The system shall allow staff to display all off duty car.

Rent

- The system shall allow staff to register customers into rental list.
- The system shall allow staff to update about customer rent record details in the rental list.
- The system shall allow staff to search rent record of customers using specific categories.
- The system shall allow staff to display customers, who rent cars.
- The system shall allow staff to display all customers rent record.
- The system must provide printable summary for successful committed rent

2.2.4 Non-Functional Requirements:

Non-functional requirements, as the name suggests, are requirements that are not directly concerned with the specific services delivered by the system to its users. They may relate to emergent system properties such as reliability, response time, and store occupancy. Alternatively, they may define constraints on the system implementation such as the capabilities of I/O devices or the data representations used in interfaces with other systems. Non-functional requirements, such as performance, security, or availability, usually specify or constrain characteristics of the system as a whole. These are the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to another. They are also called non- behavioral requirements. Our system provides the following non-functional requirements.

- **Simplicity:** The architectural design of our system should be simple to use. System should be user-friendly
- **Usability:** The system provides a help and support menu in all interfaces for the user to interact with the system. The user can use the system by reading help and support
- **Security:** The system secure all confidential data. Must be security checked. The system provides username and password to prevent the system from unauthorized access. The staffs' password must be greater than eight characters. The subsystem should provide a high level of security and integrity of the data held by the system, only authorized Personnel of the company can gain access to the company's secured page on the system; and only users with valid password and username can login to view user's page.
- **Performance:** The system shall perform in iOS, Android, and Windows. Our system provides a quick response to users' actions. Performance should be better. The system response time for every instruction conducted by the user must not exceed more than a minimum of 10 seconds. The system should have high performance rate when executing user's input and should be able to provide response within a short time span usually 50 second for highly complicated task and 20 to 25 seconds for less complicated task.
- **Availability:** Our system provides 24x7 hours of service. Only needs internet to run our system. System must be portable. Also in the occurrence of any major system malfunctioning, the system should be available in 1 to 2 working days, so that business process is not severely affected
- **Flexibility:** Our system is smooth to use.

3 Software Process Model

A software process is a set of related activities that leads to the production of a software product. We develop our software using both waterfall model and iterative model. The concepts of the methods, strengths and weaknesses will be discussed. This information will then be used to support the choice of method. Afterwards the chosen method and its application in the report will be described. For this reasons, we are using those model are discussed below in Subsection 4.1 and why the other models are not suitable for our system are discussed later in the Subsection 4.2

3.1 Suitable Models

The following is a discussion of the reasons why the model are appropriate for our purpose.

The waterfall model: Waterfall model is a plan driven type process and iterative model is agile type process. Waterfall model have well specified steps and help us to structure our project and write detailed documentation about project. One of the strengths of the waterfall method is the consecutive order of progress the structure gives. This gives the project members an overview of how far they are in a specific part or in the overall progress of the project.

The Iterative model: On the other hand Iterative model is flexible with it's activities and for changing the situation and if we further want to add a feature that don't notice to add. Iteration gives a better understanding of the project context, development and requirements.

So we can make version of our software or if a serious bug is issued than we can immediate change this. Both methods have their strengths and weaknesses and one method might have a strength that is a weakness for the other. When using this method it can be difficult to keep an overview because each iteration adds more information. On the other hand each iteration gives a better understanding of the project context, development and requirements. This could lead to an endless circle of improving and expanding the project, rendering it difficult to say when the project is actually finished. This could prove to be difficult to answer, whereas the waterfall method gives a structure that restricts the multiple iterations and sets an end point. The waterfall model provides an easy overview of the project progress. It is also a good choice because the system requirements and the goals that have been set, can be defined after each part of the documentation. This allows for confirmation of the goals and requirements that have been set.

Waterfall model steps:

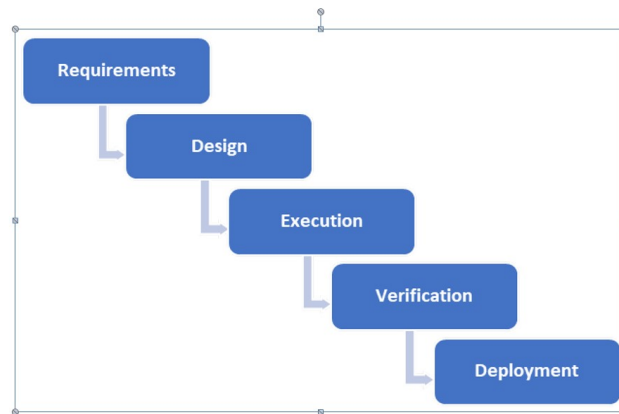


Figure 1: The Waterfall Model

Iterative model steps:

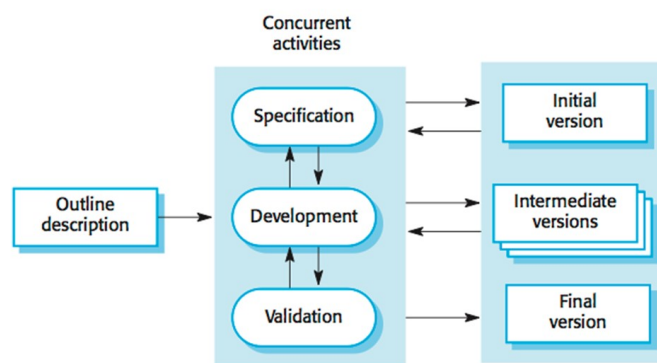


Figure 2: The Iterative Model

3.2 Not Suitable Models for Our System

The following is a discussion of the reasons why the other models are not appropriate for our purpose.

The Reuse-Oriented Model: When people working on the project are aware of designs or codes that are comparable to what is required, this frequently occurs informally. These are sought after, modified as necessary, and added to the system. The majority of the time, developers are not concerned with user needs. This could result in a system that doesn't actually serve the demands of people.

On a website, we looked for projects that were identical to ours or parts that were associated with it. But we failed to locate it. Additionally, we collect requirements through online surveys and interviews to ensure that they reflect the actual needs of users. As a result, this paradigm is inappropriate for our purpose.

The Spiral Process Model: The spiral model places a strong emphasis on risk analysis and combines the iterative development process model with sequential linear development model, or the waterfall model.

Use of this model when there is a budgetary restriction, risk assessment is crucial, and the consumer is unsure of what they need. The requirements from the customer for our project are rather straightforward. For our project, there is no need to place a lot of focus on risk analysis. We make an effort to keep project development expenses to a minimum. As a result, this paradigm is inappropriate for our purpose.

4 System Choice

System choice must be based on 3 sub-activities.

4.1 Appreciate The Situation:

By working with “Rich Pictures”, we can explicate important user views of a situation, facilitate debate and get an overview of the situation quickly.

4.1.1 Rich Picture

A rich picture is an informal drawing that presents the illustrator’s understanding of a situation. It will give us the current scenario of what a person faces difficulty who is in bad situation and how to solve the problem without facing any problem. We will highlight the situation we are facing with the help of this rich picture. The purpose of this picture is to highlight the problems of the existing system and to understand them easily.

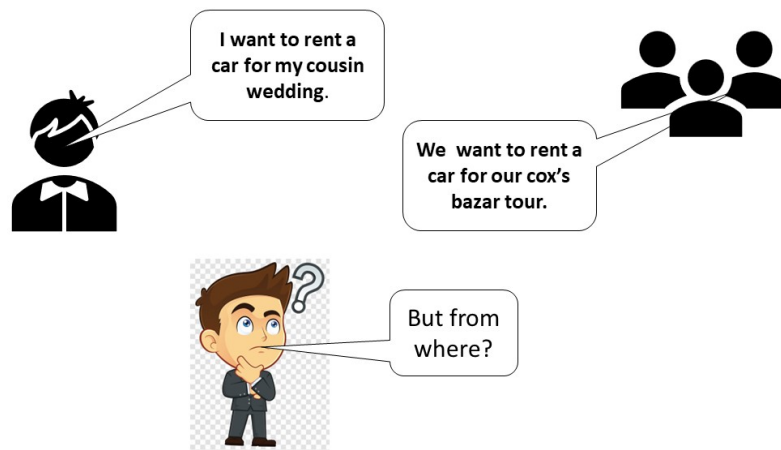


Figure 3: Rich Picture of Car Renting System



Figure 4: Rich Picture of Car Renting System

4.2 Cultivate New Ideas

After analyzing the current situation problem, we decided to develop a system that will solve the arising issue in the rich picture. If it happens any accident We decided to provide first-aid services which can be given by anyone. Besides survival guidelines are also given which can save someone's life. Most of the cases, people don't remember emergency services. Keeping this in mind, we decided to provide emergency services. In case, the emergency services is busy, for that we decided to provide a customized map that will show the nearest repair service or garages. If the customers are in the danger zone, we decided to provide an emergency calling or messaging service to all favorite numbers by artificial intelligence.

4.3 FACTOR

Before defining our system, we need to include some characteristics of the definition of the system which is known as FACTOR.

The FACTOR criteria of our Car Renting System are given below:

1. F-Functionality: Our system can be used for emergency repair service center, locating nearby garage/service center through a customized map.
2. A-Application domain: A good road for driving and nearby good mechanic.
3. C-Conditions: Must have a smartphone.
4. T-Technology: Global mapping and automated calling and messaging service.
5. O-Objects: To help a damage car and an endangered person.
6. R-Responsibility: This system will act as a communication medium for saving times, money and life.

4.4 System Definition

A system definition expresses fundamental properties for system development and use. It describes the system in context, what information it should contain, which functions it should provide, where it is to be used, and which development conditions apply. A system definition should be brief and precise, and contain the most fundamental decisions about the system. Creating a brief and precise formulation provides an overview and makes it easier to compare alternatives. A long, detailed description makes this difficult.

The system definition of our system is:

”A computerized system which is used for seeking help from emergency, locate nearby garage/mechanic/rescue team and give support on problematic situation. The system should be operable in any smartphone with required memory space. Both video and textual description will be available to give services and survival guideline to the person. The system can be used with no prior experience by a person who have the mentality to help a traveller who is struggling. A user account should be created to send and message to multiple person those who are likely to help the person.”

5 Problem Domain

In order to understand a system well and develop it in a good way, a developer has to know the problem domain of the system well. The Problem domain is the part of the system, which gets administrated, monitored, or controlled by the system. The purpose of the Problem domain is trying to identify the parts of the system, and model them in the categories: Classes, Objects, Structures, and Behavior to give an overview of that part of the system. Once an overview has gotten achieved, the developer can start to supply details to different categories. As an example, this can get achievable by structuring relations between Classes and Objects. An experimental attitude is vital in order to make a strong single coherent model, is an iterative process, which in most takes time to perfect. In many cases, even an experienced developer needs to go back and add or delete a new Class in the model. The final result is a coherent model, with good relations between Classes, Structures, and Behaviors. This Section and its subsections create an overview of this project's Classes and Class Relations. The goal is to achieve a clear overview of the system

5.1 Classes

Before being able to provide a full picture of the Problem domain, the classes of the Problem domain must be described in detail first. In this Section, the actors of the system will be introduced and somewhat described. actors would usually be dealt with in the Application domain of the system, but are introduced in this part since they play a crucial role in the system and within the Problem domain. Every class described in this Section is a blueprint for objects that contain data on different parts of the system, even the actors

The chosen classes from the class candidate analysis will be defined in this section.

5.1.1 Physical

Car:This class holds information about the car details.

TripDetails:This class holds information about the running and completed trip.

BookingInfo:This class holds the information about booking request and accepted request.

5.1.2 Person

Passenger:This class holds the information about passenger who are the general user of this system.

Admin:This class holds the information about the main authority of the system who can control the user of the system.

Driver:This class holds the information about the driver of the specific car renting service center.

5.1.3 Places

CarRentCenter:This class holds information about the car service center, its location, car collection, completed trip, contact number and providing service details.

5.2 Class Diagram

A class diagram is used to describe and illustrate, in a visual format, the relation between classes in a given system. Using the visual attributes of the class diagram, it becomes much easier to understand how different parts of a system, is related.

The class diagram is structured with CarRentCenter as central class. Car class and Driver class are containing the details of car and driver associated with the CarRentCenter class. If the CarRentCenter will delete Car and Driver class will be automatically deleted. So Car class and Driver class have composition relationship with CarRentCenter class. CarRentCenter, Passenger and Admin class inherit the User class as at first all member's are User. Car class has association relationship with BookingInfo class. TripDetails class has aggregation relationship with BookingInfo class as if the BookingInfo class is deleted the TripDetails class have to sustain.

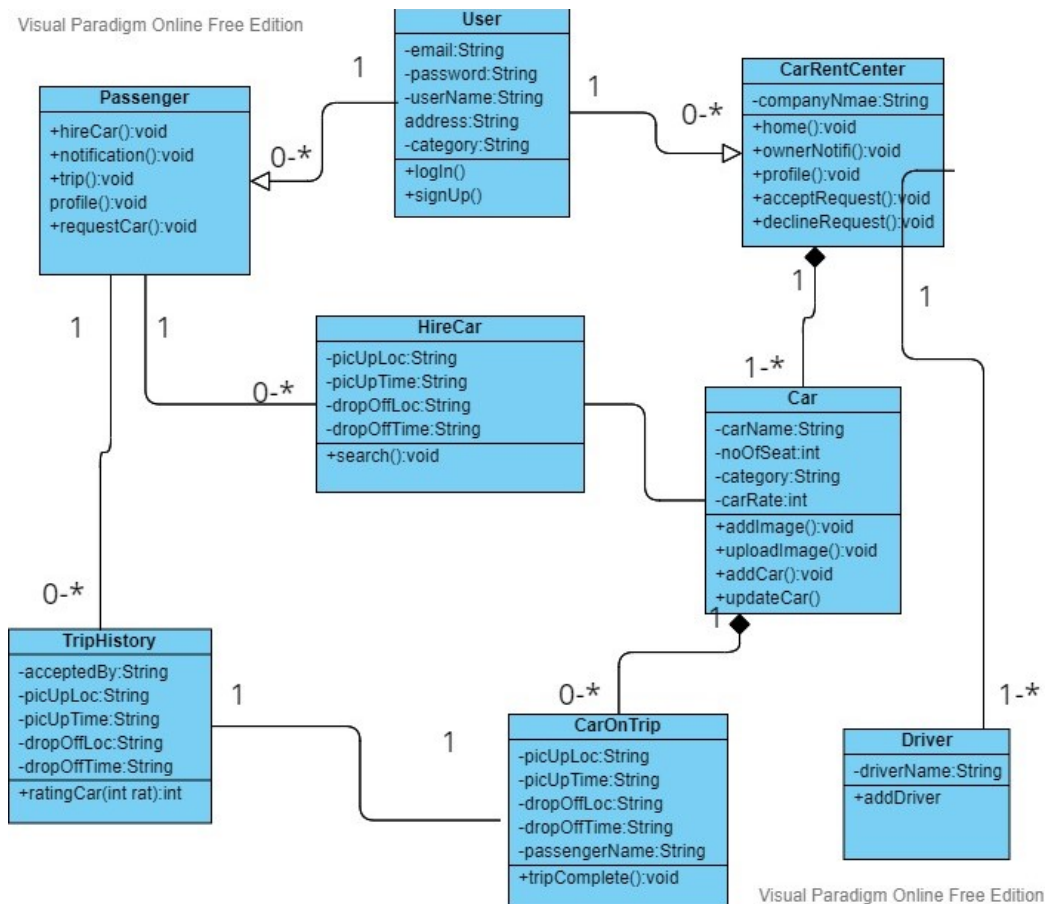


Figure 5: Class Diagram of Car Renting System

5.3 Events

In this section a list of events will be presented. The events described are those which have been chosen through an event candidate analysis. The events and what trigger them are listed below.

5.3.1 Event Definition:

Event selection defines the second set of building blocks for problem domain model. However, there is a fundamental difference between events and classes. Numerous verbs relate to the way users carry out their jobs. Such verbs do not belong to the problem domain, but rather to the application domain. We can eliminate many of these immediately.

The chosen events from the event candidate analysis will be defined in this section.

Car added: A car is added to the Car class when CarRentCenter want to add a new car to his car collection or have an addition from its original volume/quantity.

Car removed: A car should be removed from the car collection, or have an subtraction from its original volume/quantity.

Trip completed: When a trip is in drop-off date, it should be removed from the inventory and thrown out. Booked a passenger: From available request for this car booked a passenger.

Send a request for car: After searching the available car of car rent center a passenger can send a request.

Select Pick-up location: Passenger have to select pickup location for searching.

Select Drop-off location: Passenger have to select drop-off location for sea.

Pickup date: select pickup day , month and year from calendar.

Pickup time: select pickup time from set exact second, minute and hour in time frame.

Drop-off date: select drop-off day , month and year from calendar.

Drop-off time: select drop-off time from set exact second, minute and hour in time frame.

Add a driver: Rent service center owner add a new driver with additional information of driver like driver name, driver age etc.

Remove a driver: Delete a driver from all driver list.

Car on trip: Set car name ,driver name ,pickup and drop-off location.

Add trip type: Add different type of trip from wedding , trip around city and trip outside of the city . Trip type describe the service type that the car renting service center want to give passenger . Also set cost of per according to trip type.

5.3.2 Event Table

Based on the classes in Section 6.1 and the class relations in the class diagram in Section 6.2, the events between the classes will now be described. The figure 6.3.1 shows the most common events, which occurs between the classes in the system. The events on the figure mainly describe the events cause by the actor classes since these are the classes which cause most events to occur. Since the figure describes the classes from the previous Sections relations based on their events, it is possible to attain an even better overview of the full system.

Shows an event table that have been constructed in order to get an overview of the relations between classes and events. The event table allows for a better judgement of which classes are relevant in the program. A plus (+) in the table indicates, that an event can occur zero or one time, whereas a star (*) indicates that an event can occur zero or more time.

Event	Car	Passenger	TripDetails	CarRentCenter
Car added	*			
Car removed	*			
Booked a passenger		+		+
Send a request for car		*		*
Select pickup location		*	+	
Driver on trip			*	*
Add trip type			*	+
Select trip type		+		
Add cost per km into city			*	+
Add cost per km outside city			*	+
Select pickup date and time		*	+	
Select drop off date and time		*	+	
Add a driver				*
Remove a driver				*
Available car list	*			+
Available driver list				*
Car on trip	*		*	+
Trip completed	*		+	

Table 2: Event table of Car renting system

5.4 Event Traces

This section will present a statemachine diagram for each class. Each statemachine diagram is the result of examining each class, with a focus on identifying the different states of the class' objects and the events, which effects the objects and changes their state. An event can happen without changing the state of the class and Change scale event will not lead to a new state for the class object. Each class examination first shows a few of the event traces that were formulated for the specific class. The relationship between the classes and events are described in table 5.3.2. The event traces will be used to model the statemachine diagram for the specific class. The event traces and statemachine diagrams are created to get a better understanding of the dynamic in the problem domain. It is possible to use the event definitions for a better understanding of each event, if the event behaviour is not clear from the event name. There are four user pattern diagrams, each representing different areas of the system. The four diagrams showcase the user patterns in the: 1.Hire Car. 2.Trip History. 3.Notification. 4.Passenger profile.

5.4.1 Hire Car:

When a passenger enter into system after login he can see four different option in the navigation bar . After clicking hire car option a form type page will shown. Put all the information for hire and clicking the search button can show all the Car rent center with number of car available for hire.

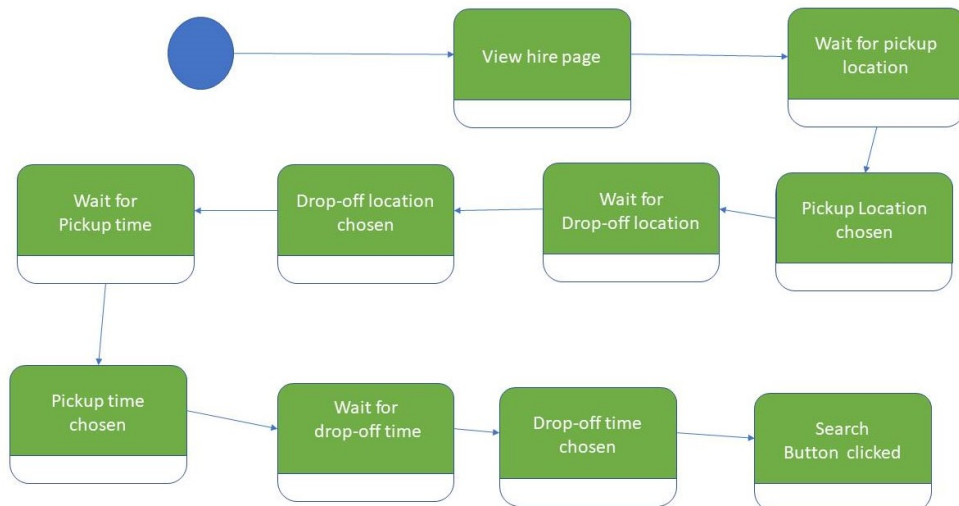


Figure 6: Statemachine diagram for the Hire Car

5.4.2 Trip History

When passenger select trip history from the navigation bar a list of completed Trip of passenger shown to him. Pickup location and time, Drop-off location and time is shown. In the bottom of this card passenger can rate the car.

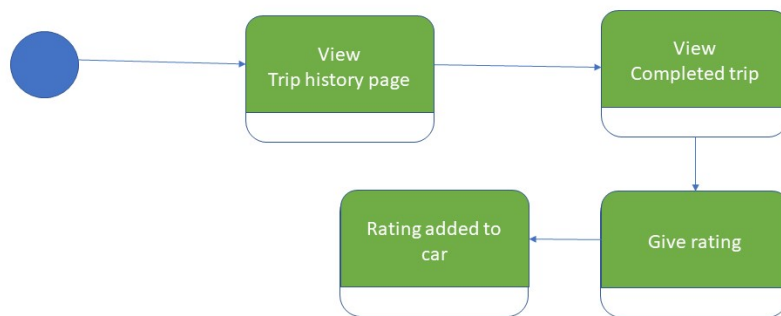


Figure 7: Statemachine diagram for the Trip History.

5.4.3 Passenger Profile

When a passenger clicked on the profile button he can view the profile page. In the profile page a user can update his username, password and email By clicking the update data button.

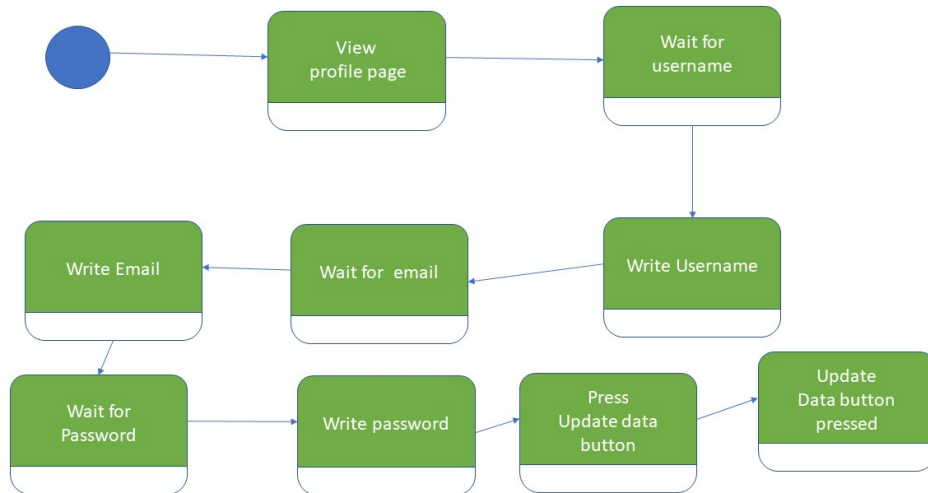


Figure 8: Statemachine diagram for the Profile

Car Rent Center: When a car rent service center owner enter into the system by login page He can see the home page .In the home page he can the three navigation bar: homepage , notification ,profile. Also a owner can see the add car ,add driver , add trip , trip type ,car on trip , driver on trip option.

5.4.4 Add Car

When owner click on the add car button he can see the add car interface. In the add car page he can put the car name, number of seat. He can choose the car category .

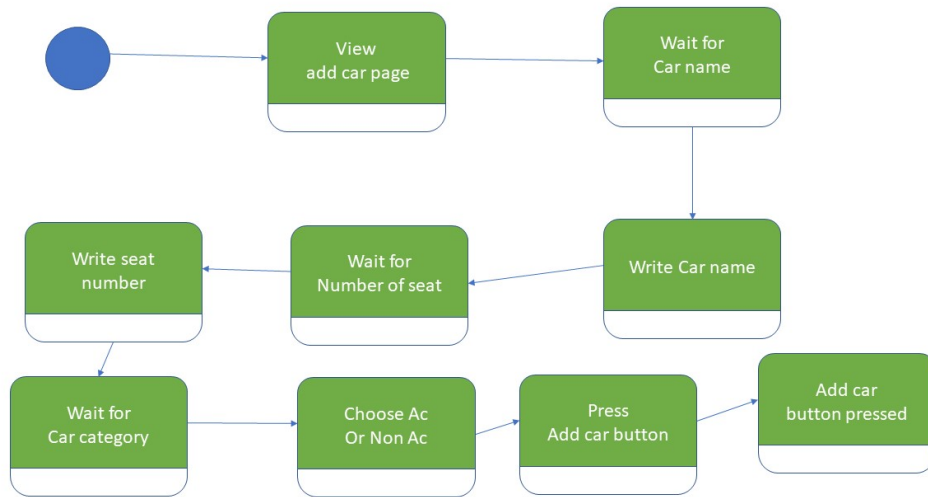


Figure 9: Statemachine diagram for the Add Car

5.4.5 Add Driver

When a car rent center owner want to add a driver ,he has to go to the home page of Car rent center interface where he can see multiple option .He has to select the Add driver button.After clicking on the add driver button add driver interface is shown to him.In this interface there are two TextField.Putting the driver name and driver age a driver is added to the car rent center driver data by clicking the add driver button.

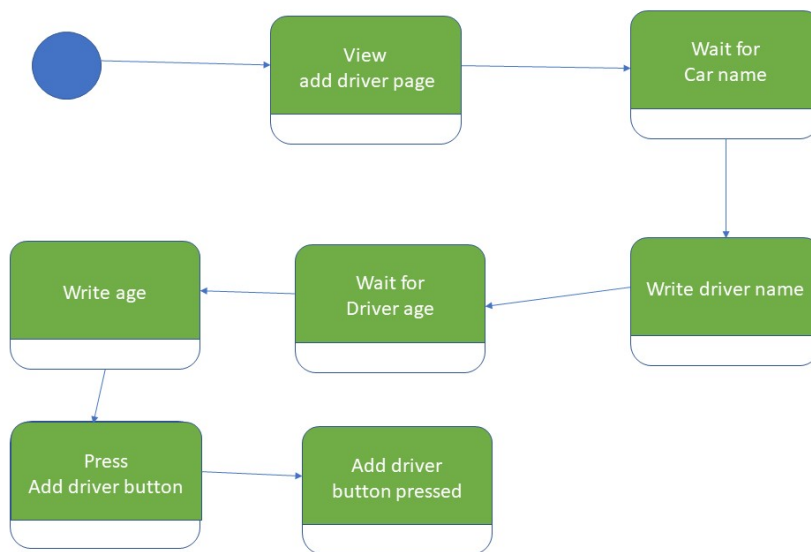


Figure 10: Statemachine diagram for the Add Driver

5.4.6 Notification

When a car rent center click notification from the navigation bar he enter the notification interface. In the notification interface he see all the requests of car send by passenger. He see all the information related to the trip like pickup location and date, drop-off location and date. He all also see the passenger email address. In the below of the card there are two option accept and cancel. If he click the accept an accepted notification is send to corresponding passenger. The accepted request is add on the car on rip .If he click on the cancel button a cancel message is send to the passenger .The canceled message is remove from the notification page.

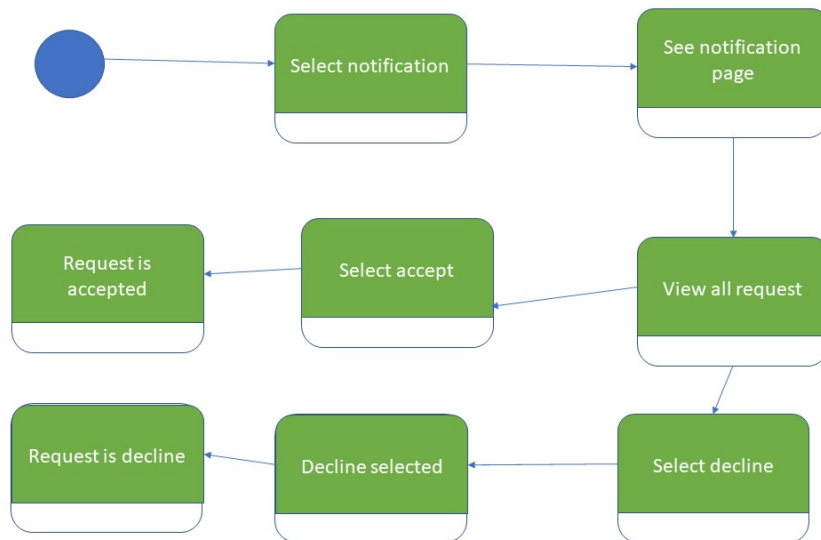


Figure 11: Statemachine diagram for the Notification

5.4.7 Car Rent Profile

When a car rent center select the profile option from the navigation bar he see the profile page/interface.In the profile interface he see three TextField change username,company name,change address.Putting the data on the TextField and click on the update data Updated username,company name and is shown top of the corresponding field.

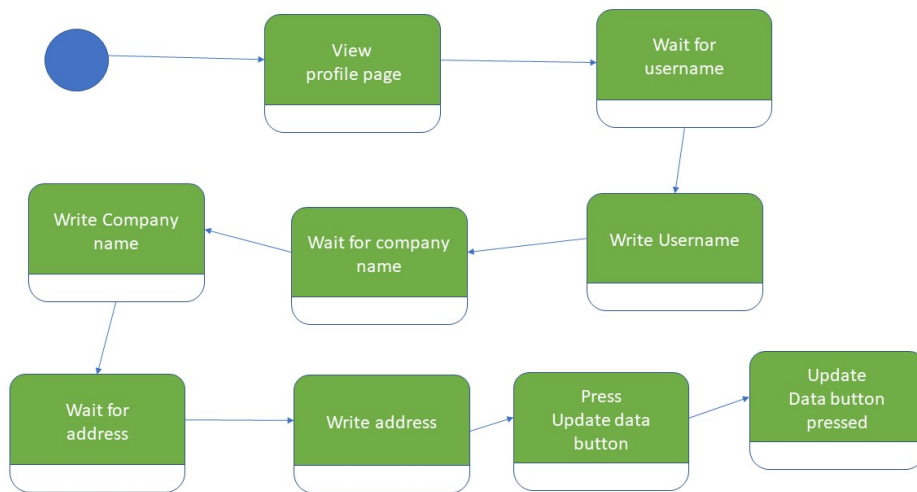


Figure 12: Statemachine diagram for the Notification

5.5 Behavior

A behavioral pattern orders individual events in time using fundamental control structures from structured programming. A behavioral pattern with sequence, selection, and iteration can be described most comprehensibly by a regular expression.

Sequence: Events in a set occur one by one.

Selection: Exactly one out of a set of events occurs.

Iteration: An event occurs zero or more times.

6 Application Domain

As stated in problem domain a developer needs to understand the system full context, and application domain is the other part of domain analysis. Application domain is that part of the system, which is operated by an organization that administrates, monitors and controls a problem domain. The purpose of the application domain is the understanding of the system's usage requirements. This is done by engaging with the users of the future system, in order to get as many information as possible, but even then it might not be enough. The user may not be able to understand technical options and not be able to write them down well enough. This can be overcome by making use case and actors of the future system, to get the proper image of the problem illustrated in pictures to get a better overview of the overall usage requirements. With a better overview of usage requirements, it is possible to start questioning the information processing capabilities, which will make developer able to the create a list of functions and related interfaces in the system. The final result is a complete list of overall usage requirements of the system.

6.1 Usage

6.1.1 Actors

To understand the usage of the product an understanding of how actors and the targeted system, an explanation will be given. An actor is an abstraction of users who can or will have an interaction with the system, which the system is the product. An example of an actor would be the instructor or the students who participate to get drivers licenses. To better understand the viewpoint of the access level for the system, a Table overview will present itself.

6.1.2 Actor Specification

This system contains two types of actor, which is the user of the system. The actor will be described by an actor specification.

Passenger Objective : A person who have needed car for some purpose is called passenger in this system.

Characteristic: The system includes a passenger base of which the passenger have different needs and preferences.

Car Rent Service center:

Objective: A person who have a car renting business can maintain his business easily with this system.

Characteristics: The system includes a car service center base of different type of trip type he offered.

6.1.3 Use Case Diagram

To understand the usage of the product an understanding of how actors and the targeted system, an explanation will be given. An actor is an abstraction of users who can or will have an interaction with the system, which the system is the product. An example of an actor would be the instructor or the students who participate to get drivers licenses. To better understand the viewpoint of the access level for the system, a Table overview will present itself.

This section will describe the systems interaction with its surroundings. This will be done by making an actor specification and then presenting user pattern diagrams, which will be showcased in state machine diagrams.

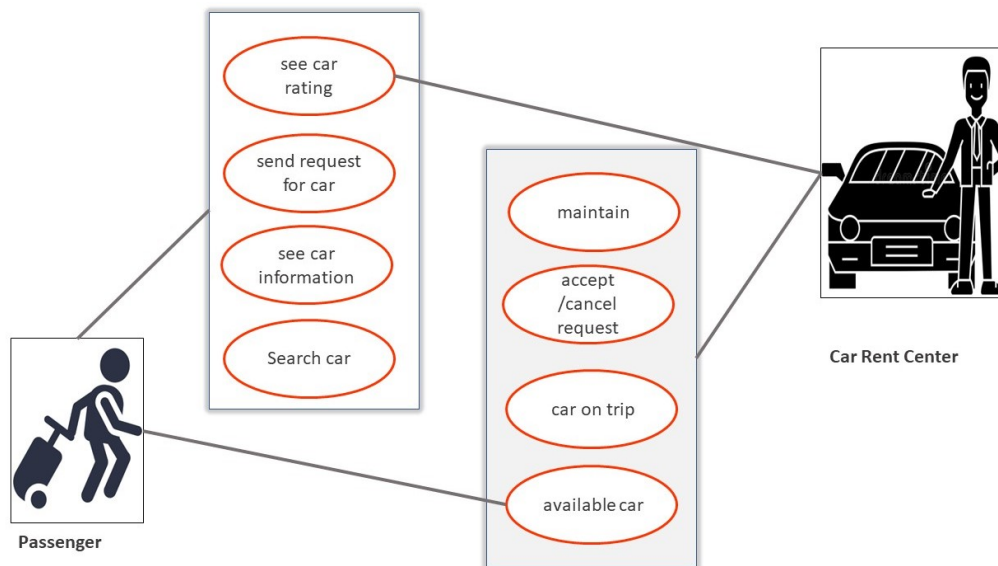


Figure 13: Use Case Diagram

6.2 Functions and Complexity

The product needs functions to be understandable so a function complexity will showcase where the Table will describe and analyze different functions. The primary goal is to identify the different functions needed for the product and give an initial assessment of their complexity. Complexity assesses from the number of tasks the function is expected to perform and expected technical difficulty.

Function	complexity	type
User Sign UP	Simple	Update
User log In	Medium	Read
Car Owner	Medium	Update
User Form	Medium	Update
Select Image	Complex	Read
Upload Image	Medium	Update
Send Driver Data	Complex	Update
Hire Car	Very Complex	Update
Select Data From Picker	Simple	Read
Select Time From Picker	Simple	Read
Trip History	Very Complex	Read
Send Notification	Very Complex	Read
Update Owner Profile	Complex	Update
Set Car Owner Data	Simple	Update
Update Passenger Data	Very Complex	Update
Set Passenger Data	Medium	Update
Screen Start time Duration	Simple	Signal
Send Car Request Data	Medium	Update
Search	Medium	Signal
Search By Address	Complex	Signal
Search Selected	Complex	Signal
Rating Algorithm	Very Complex	Compute
Update Car	Complex	Update
Delete Car	Simple	Update
Delete Passenger Notification	Simple	Update

Table 3: Functions

User Sign Up:An user need to be added to the database.

User LogIn: Sends log in request to a server, and checks for matching Username and password in a database. As this requires sorting through the database for matching credentials, the task is assessed to be of medium complexity.

Car Owner: send car owner information.

User Form:User form collects all information off the user. Also categorized the user.

Select Image: Select image from the local device.

Upload Image:Upload the selected image.

Send Car Data:Send car name ,category ,image ,number of seat to the database.

Send Driver Data:Send driver name , age to the database.

Hire car:Send pickup location,pickup time,drop-off location,drop-off time to the database.

Select Date From Picker:Select pickup and drop-off date from calendar.

Select Time From Picker:select pickup and drop-off time from time picker.

Trip History:Accepted request for trip is in the trip history.

Send Notification:Request is accepted or decline by the owner is shown to the passenger.

Update Owner Profile:Owner username and password is updated on the database.

Set Car Owner Data:Updated data are shown form the database to the profile page.

Update Passenger Data:Update the username and password of the passenger in the database.

Set Passenger Data:updated data is shown to the passenger on the profile page .

Screen Start Time Duration:In the time duration a loading progress is shown to the user.

Send Car Request Data:Request is send to the car rent center.

Search:Search for hire a car.

Search By Address:Search car rent center according to the address.

Search Selected:select a result from the search result.

Rating Algorithm:Rating algorithm is used compute the rating of a car.It follow the average rating technique .where total rating is divided by total number of rating.

Update Car :Car information is updated on the database.

Delete Car :Delete car if the request is decline by the owner.

Delete Passenger Notification:Delete all notification from the passenger database.

7 System Architecture Design

This chapter describes the priority of system criteria, and the system structure which is divided into component architecture and process architecture, with descriptions of the models that has been used.

7.1 Priority of Criteria

When designing a software based system, it is important to consider which criteria are needed for the system, and if some of the criteria are more important than others. The criteria will be organised in to the bellow figure, the importance of them range from very important to trivially fulfilled.

Criteria	Very important	Important	Less important	Irrelevant	Trivially fulfilled
Useful		X			
Secure			X		
Effective			X		
Correct	X				
Reliable			X		
Maintenance		X			
Testable		X			
Flexible		X			
Understandable			X		
Reusable				X	
Movable				X	
Integrable			X		

Table 4: Priorities for different criteria of Car Renting System

Useful indicates the adaptation of the organisational, work related, and technical surroundings. This is important for the Car Renting system to fulfil, since the organisational surroundings indicates that the user has to be able to rent and trip correctly.

Secure indicates if the system is secure against unauthorized accessibility. This is less important in the Car renting system, since the system does not contain sensitive personal information.

Effective indicates the economical usage of the technical platform facilities. This is marked less important, as today's smartphones are capable of supporting the requirements needed for this system.

Correct indicates the level of fulfilment of the formulated requirements. The correctness of the Car renting system is very important, as it is needed to know about correct location for the system.

Reliable indicates the fulfilment of the required precision when functions are executed. The reliability is less important for the Car renting system, as functions can work a little different than stated, as long as the user gets a usable result.

Maintenance indicates the cost of finding and correcting errors in the system. The maintenance of the Car renting system is important, as errors will be made throughout the project. Therefore it is important that errors can be corrected, for the system to work properly.

Testable indicates the cost of securing that the system fulfils the requirements. It is important for the Car renting system to be testable, as testing is part of the waterfall method, and also a good way for the system to be maintainable which is important to the system.

Flexible indicates the cost of changing the system once it has been taken into use. Flexibility is important for the Car renting system, since changes and updates could be needed, in order to maintain the program in the future either because new functionality, or updates.

Understandable indicates the effort to make sure that the system can be understood in the context. Understand ability is less important to the Car renting system, as the system is not planned for other developers to work on, and will not be used in a larger context.

Reusable indicates the possibility to reuse parts of the system in similar systems. Reuse is irrelevant as the system is not planned to be used in relation with other systems.

Movable indicates the cost of moving the system to another technical platform. For the Car renting system it is irrelevant to consider the move ability, as the system is not planned to be moved to other technical platforms than smartphones and tablets.

Integrable indicates the cost of connecting the system to other systems. Integrability is less important for the Car renting system, because the system will probably not be connected to other systems, unless API are going to be included.

7.2 Architectural Design

Architecture is general basic structure, which is extended later. Architectural design is based on three fundamental principles.

1. Define and prioritize criteria. Architectural design start from the system requirement .They must be prioritized.

2. Bridge criteria and technical platform. The architecture must ensure efficient use of the technical platform's facilities. We must therefore analyze them. we have to select the technical platform carefully so that by using it we can fulfill our criteria.

3. Evaluate designs early.

Evaluation helps you determine whether a design can be implemented. It should ensure that the criteria will be met and prevent resource bottlenecks. Evaluation should take place as early as possible, since ambiguity in an architecture can dramatically reduce productivity.

Architectural patterns can help us make consistent design decision. The generic architecture reflects the division of the problem domain and application domain. At the top layer of this architecture is interface layer. Our flutter app is the interface layer. Interface layer is divided into user interface layer. There are two user in our system. After the Interface layer Function component is placed. A part of a system that implements functional requirements. A function describes externally observable behavior that relates directly and meaningfully to the users' work. Functions are designed and implemented using operations in the system's classes. An operation is a description of behavior that can be activated through an object. Conceptually, an operation can be understood as a procedure. It is activated by a message, it carries out prescribed data processing, and it returns to the calling point— with or without a result. In our system different type dart predefined function and some custom made function is used to complete the task.

7.2.1 Architechture Component

The architecture component modeling is a structural way of seeing things and divides the concerns of the system. In the bellow of the model component our problem domain is implemented .Our problem domain is developing the car renting system where we design a sytem for user interface of passenger and car rent center. Model component is connected with technical platform. The technical platform server of our system is Firebase..Firebase database is set of collections. Firebase is the best back-end technical platform for the Flutter app. The Firebase is a real-time database, ready-made authentication options along with the Flutterin-built widgets and a single code for Android and iOS make the whole software development process faster while maintaining the solution's safety and performance.

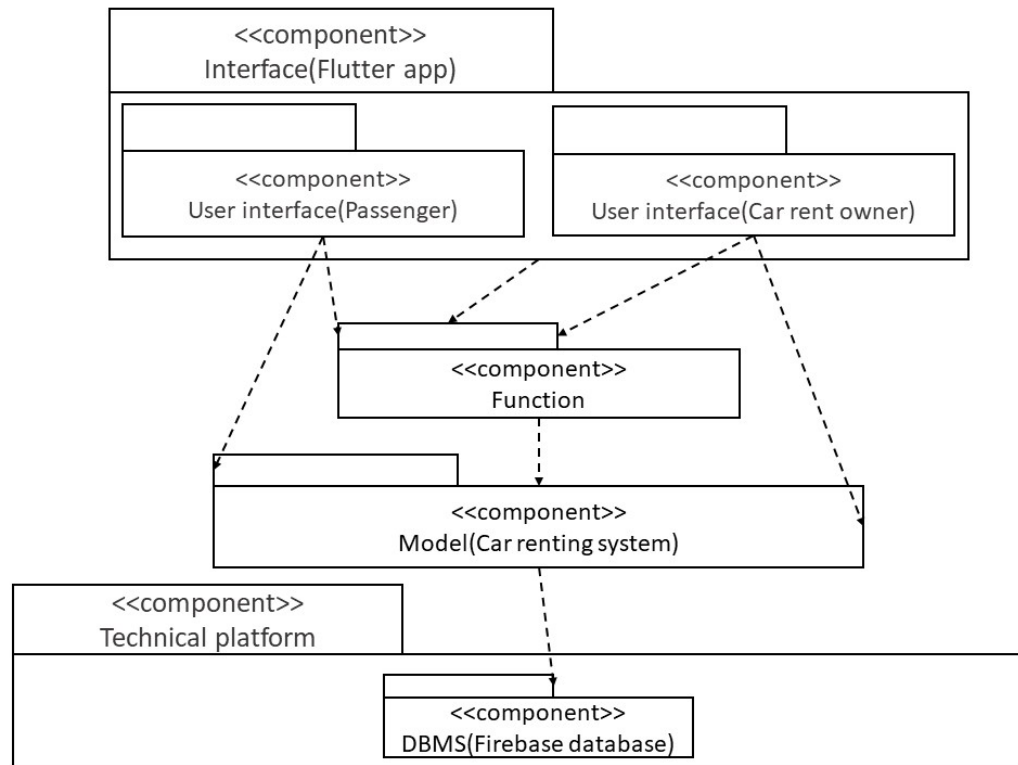


Figure 14: Architecture Component Diagram

7.2.2 Model Component

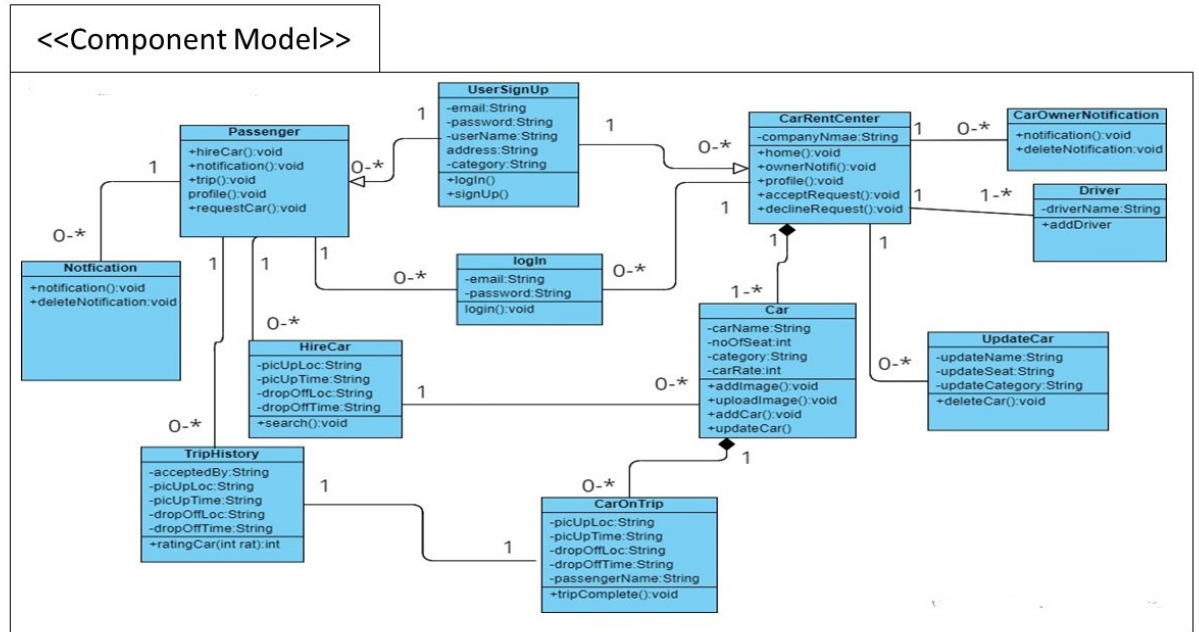


Figure 15: Architecture Component Diagram

8 User Interface Design

To ensure that the end product is operable, both Low and High fidelity prototype gets created before the final implementation. It is done to ensure that an interface made by the group, does fit the needs from the driving schools. It is time-consuming, so prototyping is a critical step before implementation. In this Section 8, we are analyzed the low fidelity design in Section 8.1 and the high fidelity design in Section 8.2.

8.1 Low Fidelity Prototype

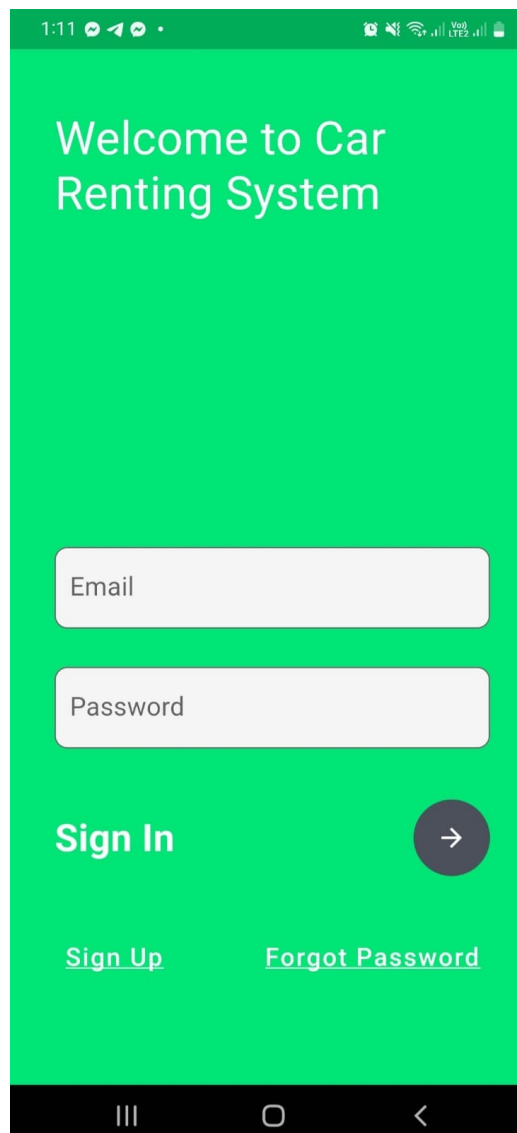
Low-fidelity prototypes sketches provide a cheap way to construct a functional design that can be displayed and receive feedback. After showcasing the initial design, there will be a discussion of what weaknesses got found that should get addressed in the following high fidelity prototype. Initially, the plan was that the solution would include building a website around it. It would mean basic website functionality also would have to be designed.

8.2 High Fidelity Prototype

Log in

The login is centered in the middle of the screen, forcing the user to log in the middle of the page. The user will have to enter their email and password. If a person has an account then he will be login by clicking Sign in button.

If the person trying to log in does not have an account, they will be able to sign up below the "sign in" button.



The image shows a mobile app login screen with a solid green background. At the top, there is a white status bar with the time 1:11 and various icons. Below the status bar, the text "Welcome to Car Renting System" is displayed in white, centered. In the middle of the screen, there are two white input fields with rounded corners. The first field is labeled "Email" and the second field is labeled "Password". Below these fields, the text "Sign In" is displayed in white, followed by a white right-pointing arrow inside a dark green circle. At the bottom of the screen, there are two links: "Sign Up" and "Forgot Password", both in white and underlined. The bottom of the screen features a black Android navigation bar with three icons: a square, a circle, and a triangle.

Figure 16: Login Page

Hire Car

In this page, a passenger can add pick up location, Drop off location, Pick up Date and time, drop up date time then search for available car center and their car.

The screenshot displays a mobile application interface for car rental. At the top, a blue header bar contains a back arrow and the text "Rent a Car and Drive Happy". Below this, a white box titled "Start a Reservation" includes a note: "*indicate required field". The form consists of four input fields: "pick-up-Location *" with a location pin icon, "Drop-Off-Location *" with a location pin icon, "pick-up-..." with a calendar icon, and "return-..." with a clock icon. A large yellow "Search" button is positioned below the form. The background of the form area shows a map of Chattogram, Bangladesh, with labels for "PAHARTALI", "SARAI PARA", and "Pahartali". At the bottom, a blue navigation bar features four icons: a magnifying glass for "Hire Car", a bell for "Notification", a clock for "Trip", and a person icon for "Person".

Figure 17: Hire Car Page

Request

In this page a passenger can see the available car, car name, number of seat and category. He can also see the rating of car. And then request a car for renting. He also remove the notification.

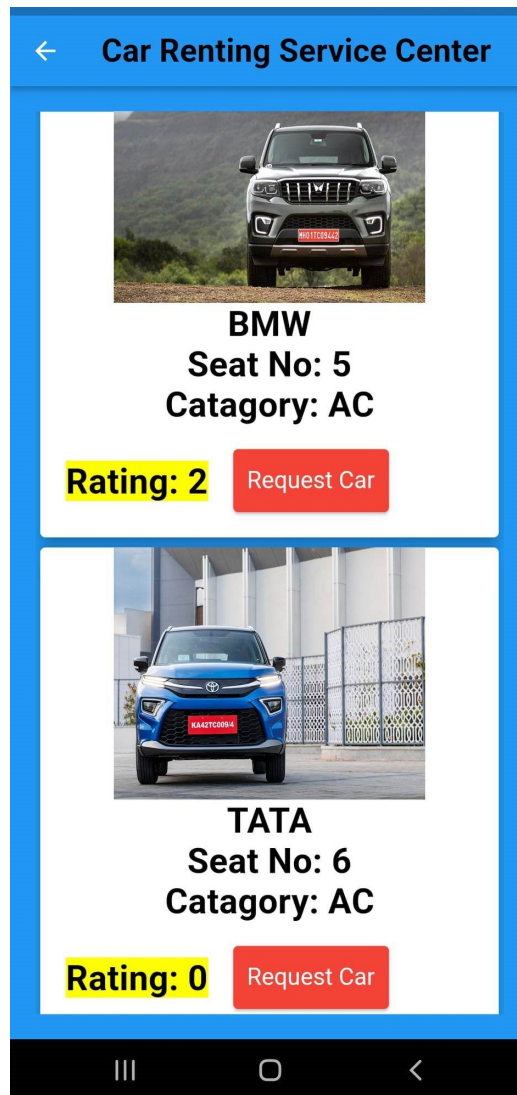


Figure 18: Request Page

Notification

In this page a passenger can see the notification who accept his request and who cancel his request. He also remove the notification.



Figure 19: Notification Page

Trip

In this page a passenger can see the trip history. He can also rating the service.

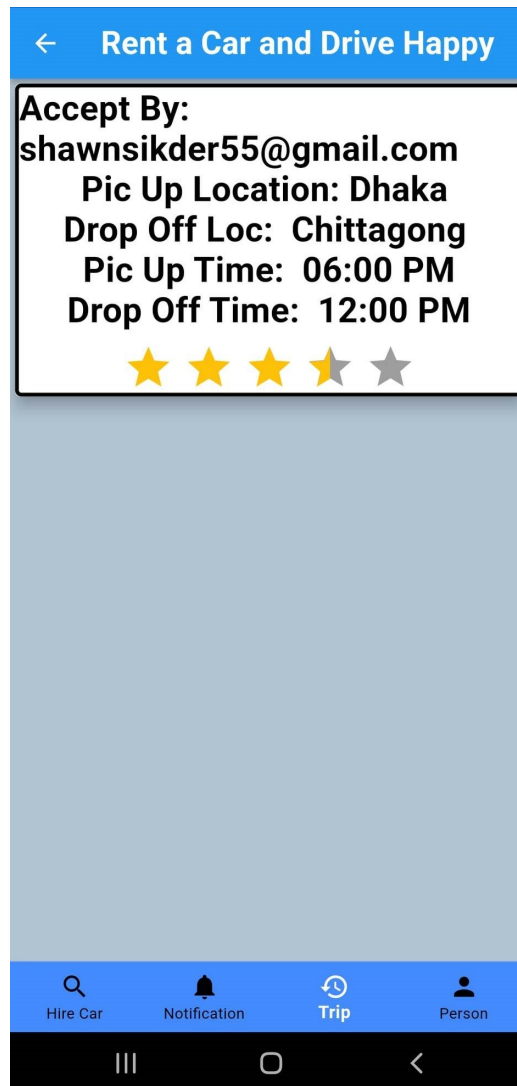
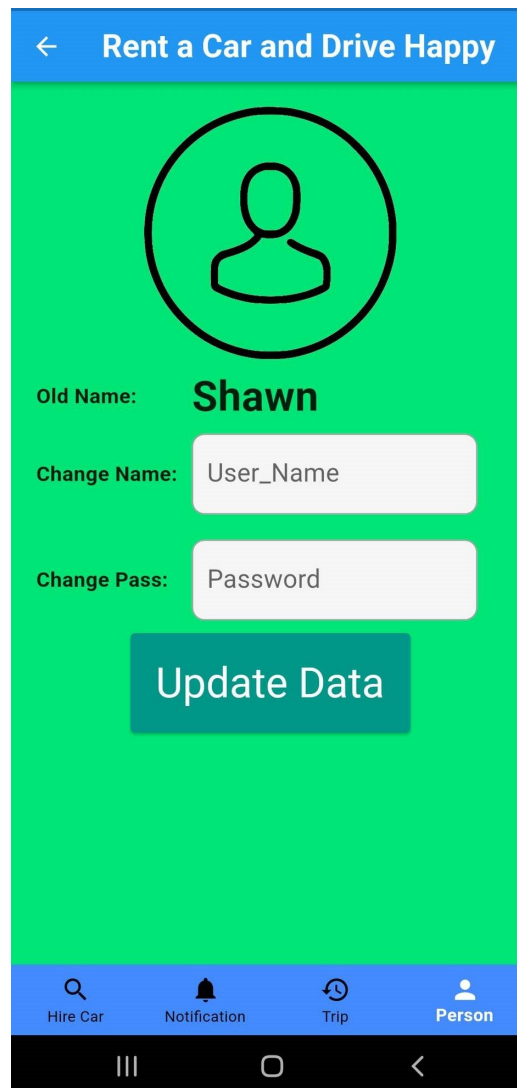


Figure 20: Trip page

Person

In this page a passenger can see his profile. He can add his profile pic. He can update his user name and password.



The image shows a mobile application interface for a car rental service. At the top, a blue header bar contains a back arrow and the text "Rent a Car and Drive Happy". Below this, on a green background, is a large circular placeholder for a profile picture. Underneath the placeholder, the text "Old Name:" is followed by the name "Shawn". Below this, there are two input fields: one labeled "Change Name:" with the placeholder text "User_Name", and another labeled "Change Pass:" with the placeholder text "Password". A large teal button with the text "Update Data" is positioned below the input fields. At the bottom of the screen, a blue navigation bar contains four icons with labels: a magnifying glass for "Hire Car", a bell for "Notification", a clock for "Trip", and a person icon for "Person". The "Person" option is currently selected. Below the navigation bar is a black Android-style status bar with three icons: a hamburger menu, a circle, and a back arrow.

Figure 21: Profile Page

Service Center Side

our system can use able for two type of customer.one is passenger and the other is Service center.Now we see the service center pages.

Home

In this page a car center can see his home page.He can add a new car and add driver.they also know which car on trip and which car are available.The car center can can Update the car and car information.

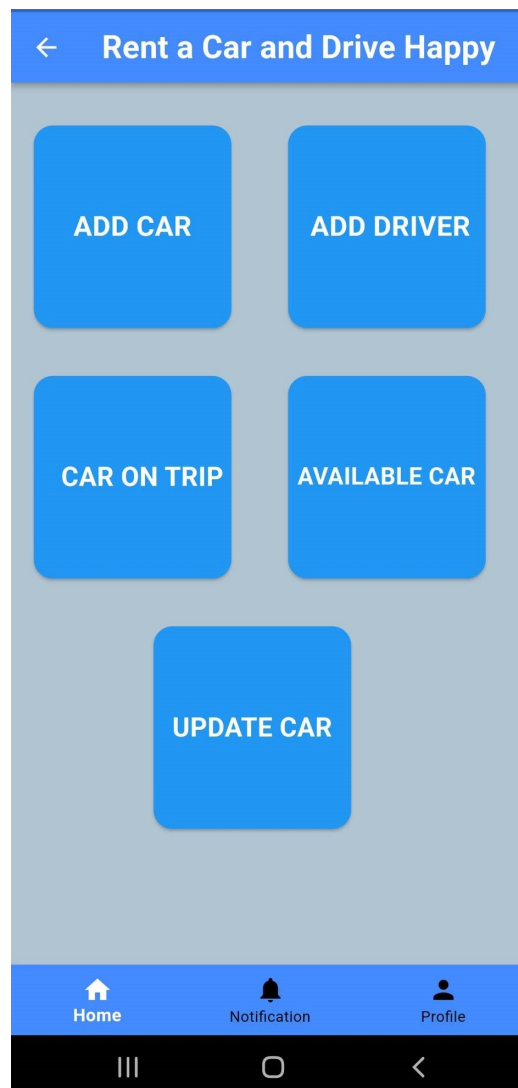


Figure 22: Car Owner Home Page

Add Car

In this page a car center can add a new car .The car owner can add car name, number of seat and set the car category Ac or Non-Ac.

← Rent a Car and Drive Happy

Car Name

No of seat

Add Image

No File Selected

Upload Image

Catagory:

☐ AC

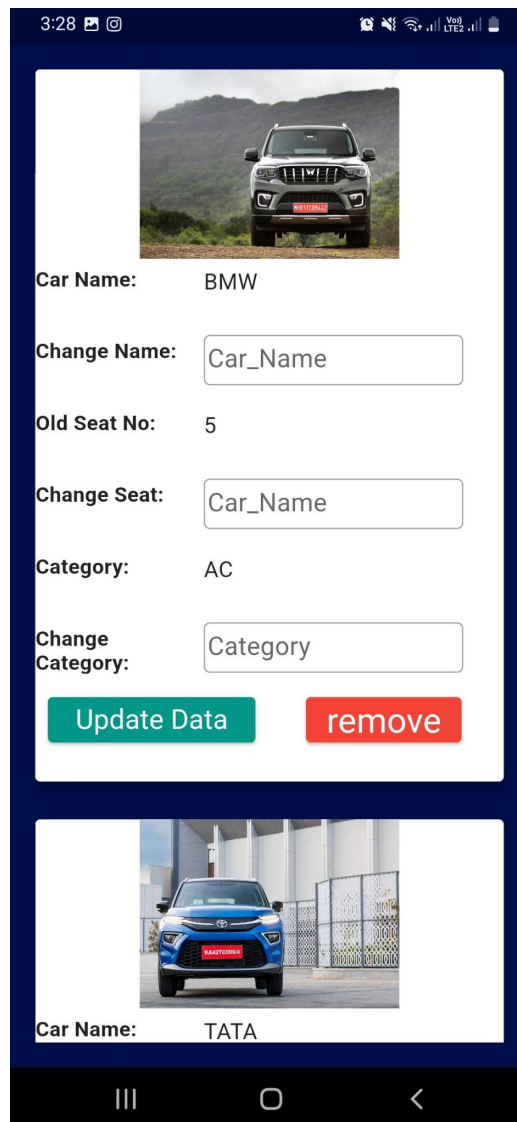
☐ Non-AC

Add Car

Figure 23: Add Car page

Update Car

In this page a car center can update car. The car owner can change car name, number of seat and set the car category Ac or Non-Ac. Then he also remove the car from car center.



3:28

Car Name: BMW

Change Name: Car_Name

Old Seat No: 5

Change Seat: Car_Name

Category: AC

Change Category: Category

Update Data remove

Car Name: TATA

Figure 24: Update Car Page

Notification

In this page a car center can see the request for rent. The car center can accept and cancel the request.

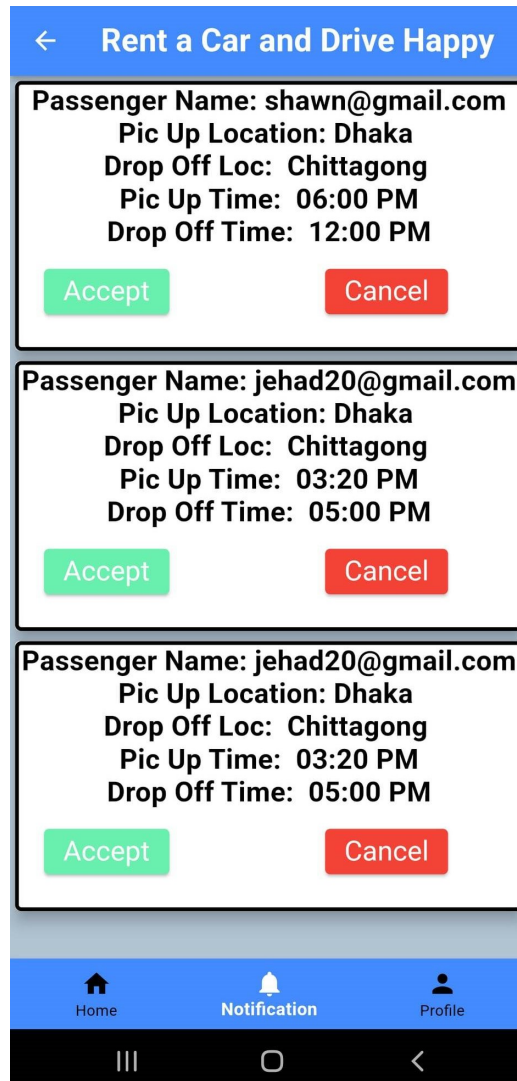
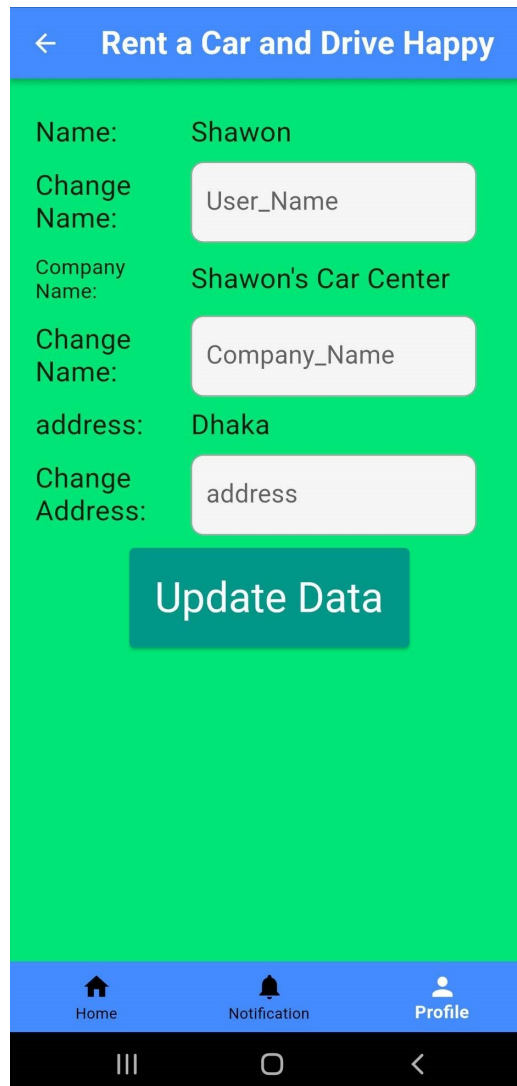


Figure 25: Car Owner Notification Page

Person

In this page a passenger can see his profile. He can update his user name, company name and address.



The image shows a mobile application interface for a car rental service. The top header is blue with a back arrow and the text "Rent a Car and Drive Happy". The main content area has a green background. It displays the user's profile information: Name: Shawon, Change Name: User_Name, Company Name: Shawon's Car Center, Change Name: Company_Name, address: Dhaka, and Change Address: address. Below this information is a large teal button labeled "Update Data". At the bottom, there is a blue navigation bar with three icons: a house for "Home", a bell for "Notification", and a person for "Profile". The very bottom of the screen shows a black Android-style navigation bar with three icons: a square, a circle, and a triangle.

← Rent a Car and Drive Happy

Name: Shawon

Change Name: User_Name

Company Name: Shawon's Car Center

Change Name: Company_Name

address: Dhaka

Change Address: address

Update Data

Home Notification Profile

Figure 26: Car Owner Profile Page

9 Implementation

To implement the project we use flutter which is Google's portable UI toolkit for crafting beautiful,natively compiled applications for mobile, web, and desktop from a single codebase. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source.

According to the class diagram in Figure 5, we have eight classes in our project. We can describe our implementation according to the class diagram.

Listing 1 shows the main.dart file that contains the "User" class of our project. Line number 1 to 23 are for importing the required packages to implement this class. We have shown in the class diagram that the "User" class is connected with classes "CarRentCenter" and "passenger" in aggregation structure.we import corresponding packages for these classes in this file.Here we have four items each corresponding to the four classes that are aggregated into this class. So we can simply say that the class "User" has a "passenger" class, a "carRentCenter" class, and so on.

```
1 import 'dart:async';
2 import 'package:car_renting_system/Login.dart';
3 import 'package:car_renting_system/Screen.dart';
4 import 'package:car_renting_system/addDriver.dart';
5 import 'package:car_renting_system/addcar.dart';
6 import 'package:car_renting_system/carOwner.dart';
7 import 'package:car_renting_system/searchselected.
  dart';
8 import 'package:car_renting_system/update-car.dart';
9 import 'package:firebase_core/firebase_core.dart';
10 import 'package:flutter/cupertino.dart';
11 import 'package:flutter/material.dart';
12 import 'package:flutter/services.dart';
13 import 'package:flutter_screenutil/
  flutter_screenutil.dart';
14 import 'package:car_renting_system/Myregister.dart';
15 import 'package:car_renting_system/Login.dart';
16 import 'package:car_renting_system/bottomnavbar.dart
  ';
17 import 'package:car_renting_system/search.dart';
18
```

```

19 import 'dart:developer';
20 import 'package:fluttertoast/fluttertoast.dart';
21 import 'package:provider/provider.dart';
22
23 import 'mapping.dart';
24
25 void main() async {
26   WidgetsFlutterBinding.ensureInitialized();
27   await SystemChrome.setPreferredOrientations([
28     DeviceOrientation.portraitUp,
29     DeviceOrientation.portraitDown,
30   ]);
31   WidgetsFlutterBinding.ensureInitialized();
32   await Firebase.initializeApp();
33   runApp(const MyApp());
34 }
35
36 class Appcolors {
37   static const Color blue = Color(0xEA0505FF);
38 }
39
40 class MyApp extends StatelessWidget {
41   const MyApp({Key? key}) : super(key: key);
42
43   // This widget is the root of your application.
44   @override
45   Widget build(BuildContext context) {
46     //Set the fit size (Find your UI design, look at
47       the dimensions of the device screen and fill
48       it in,unit in dp)
49     return ScreenUtilInit(
50       designSize: const Size(360, 690),
51       minTextAdapt: true,
52       splitScreenMode: true,
53       builder: (context, child) {
54         return MaterialApp(
55           debugShowCheckedModeBanner: false,
56           routes: {
57             'register': (context) => const SignUp(),
58             'login': (context) => const MyLogin(),
59             'bottomNav': (context) => const

```

```

58         BottomNavBar(),
59         'search': (context) => const Search(),
60         'carOwner': (context) => const CarOwner
61         (),
62         'add-car': (context) => const AddCar(),
63         'add-driver': (context) => const
64         AddDriver(),
65         'update-car': (context) => const
66         UpdateCar(),
67     },
68     title: 'First□Method',
69     theme: ThemeData(
70         primarySwatch: Colors.blue,
71         textTheme: Typography.englishLike2018.
72         apply(fontSizeFactor: 1.sp),
73     ),
74     home: child,
75 );
76 },
77 child: const SplashScreen(),
78 );
79 }
80 }

```

Listing 1: main.dart file containing "User" class

Listing 2 shows the hirecar.dart file that contains the "HireCar" class of our pojejr. Line number 1 to 22 are for importing the required packages to implement this class.. According to the class diagram in Figure 5 The "HireCar" class is related to the "Subcategory" class in the aggregation structure.

```
1 import 'dart:async';
2
3 import 'package:car_renting_system/mapping.dart';
4 import 'package:cloud_firestore/cloud_firestore.dart';
5
6 import 'package:firebase_auth/firebase_auth.dart';
7 import 'package:flutter/material.dart';
8 import 'package:google_maps_flutter/
9     google_maps_flutter.dart';
10 import 'package:google_place/google_place.dart';
11 import 'package:intl/intl.dart';
12 import 'package:date_format/date_format.dart';
13 import 'package:provider/provider.dart';
14 import 'package:uuid/uuid.dart';
15 import 'package:http/http.dart' as http;
16 import 'dart:convert';
17 import 'dart:async';
18 import 'package:location/location.dart';
19
20 import 'package:flutter/cupertino.dart';
21 import 'package:flutter/material.dart';
22 import 'package:google_maps_flutter/
23     google_maps_flutter.dart';
24
25 import 'package:provider/provider.dart';
26
27 class HireCar extends StatefulWidget {
28   const HireCar({Key? key}) : super(key: key);
29
30   @override
31   State<HireCar> createState() => _HireCarState();
32 }
33
34 class _HireCarState extends State<HireCar> {
35   Completer<GoogleMapController> _controller =
```

```

33         Completer();
34
35     @override
36     String? _setTime;
37     String? _hour, _minute, _time;
38     String? dateTime;
39     TimeOfDay selectedTime = TimeOfDay(hour: 00,
40         minute: 00);
41     TextEditingController pickUpDate =
42         TextEditingController();
43     TextEditingController returnDate =
44         TextEditingController();
45     TextEditingController pickUpTime =
46         TextEditingController();
47     TextEditingController returnTime =
48         TextEditingController();
49     TextEditingController picupLoc =
50         TextEditingController();
51     TextEditingController returnLoc =
52         TextEditingController();
53
54     DateTime selectedDate = DateTime.now();
55
56     Future addtrip() async {
57         final FirebaseAuth _auth = FirebaseAuth.instance
58             ;
59         var currentUser = _auth.currentUser;
60         CollectionReference collectionRef =
61             FirebaseFirestore.instance.collection("add-
62                 trip");
63         return collectionRef.doc(currentUser!.email).set
64             ({
65                 "pic-up-loc": picupLoc.text,
66                 "return-loc": returnLoc.text,
67                 "pic-up-date": pickUpDate.text,
68                 "return-date": returnDate.text,
69                 "pic-uo-time": pickUpTime.text,
70                 "return-time": returnTime.text,
71             }).catchError((error) => print("something is
72                 wrong. $error"));
73     }

```



```

62
63 Future<void> _selectDateFromPicker(BuildContext
    context) async {
64     final DateTime? picked = await showDatePicker(
65         context: context,
66         initialDate: selectedDate,
67         initialDatePickerMode: DatePickerMode.day,
68         firstDate: DateTime(2015),
69         lastDate: DateTime(2101));
70     if (picked != null) {
71         setState(() {
72             pickupDate.text = "${picked.day}/${picked.
                month}/${picked.year}";
73         });
74     }
75 }
76
77 Future<void> _selectDateFromPicker2(BuildContext
    context) async {
78     final DateTime? picked = await showDatePicker(
79         context: context,
80         initialDate: selectedDate,
81         initialDatePickerMode: DatePickerMode.day,
82         firstDate: DateTime(2015),
83         lastDate: DateTime(2101));
84     if (picked != null) {
85         setState(() {
86             returnDate.text = "${picked.day}/${picked.
                month}/${picked.year}";
87         });
88     }
89 }
90
91 Future<Null> _selectTime(BuildContext context)
    async {
92     final TimeOfDay? picked = await showTimePicker(
93         context: context,
94         initialTime: selectedTime,
95     );
96     if (picked != null)
97         setState(() {

```

```

98         selectedTime = picked;
99         _hour = selectedTime.hour.toString();
100        _minute = selectedTime.minute.toString();
101        _time = _hour! + '␣:␣' + _minute!;
102        pickUpTime.text = _time!;
103        pickUpTime.text = formatDate(
104            DateTime(2019, 08, 1, selectedTime.hour,
105                selectedTime.minute),
106            [hh, ':', nn, "␣", am]).toString();
107    });
108    }
109    Future<Null> _selectTime2(BuildContext context)
110        async {
111        final TimeOfDay? picked = await showTimePicker(
112            context: context,
113            initialTime: selectedTime,
114        );
115        if (picked != null)
116            setState(() {
117                selectedTime = picked;
118                _hour = selectedTime.hour.toString();
119                _minute = selectedTime.minute.toString();
120                _time = _hour! + '␣:␣' + _minute!;
121                returnTime.text = _time!;
122                returnTime.text = formatDate(
123                    DateTime(2019, 08, 1, selectedTime.hour,
124                        selectedTime.minute),
125                    [hh, ':', nn, "␣", am]).toString();
126            });
127    }
128    @override
129    Widget build(BuildContext context) {
130        //final applicationBlock = Provider.of<
131        ApplicationBlock>(context);
132        return SafeArea(
133            child: Center(
134                child: SingleChildScrollView(
135                    child: Center(
136                        child: Container(

```

```

135 width: (MediaQuery.of(context).size.
      width),
136 height: 2000,
137 padding: new EdgeInsets.all(10.0),
138 child: Card(
139   shape: RoundedRectangleBorder(
140     borderRadius: BorderRadius.
        circular(15.0),
141   ),
142   color: Colors.white,
143   elevation: 10,
144   child: Column(
145     mainAxisAlignment: MainAxisAlignment.min,
146     children: <Widget>[
147       Image.network(
148         'https://c7.alamy.com/comp/
          T8P424/vector-logo-for-car-
          rental-and-sales-T8P424.jpg
          ',
149         height: 200,
150         width: 500,
151       ),
152       const ListTile(
153         title: Text('Start a
          Reservation',
154           style: TextStyle(fontSize:
            30.0)),
155         subtitle: Text('*indicate
          required field',
156           style: TextStyle(fontSize:
            18.0)),
157       ),
158       TextField(
159         controller: pickupLoc,
160         textAlign: TextAlign.center,
161         decoration: InputDecoration(
162           suffixIcon: IconButton(
163             color: Colors.white,
164             onPressed: () {},
165             icon: Icon(
166               Icons.

```

```

        location_searching_sharp
        ,
        color: Colors.black,
    ),
),
//prefixIcon: Icon(Icons.
    timer),
filled: true,
fillColor: Colors.grey.
    shade100,
labelText: 'pick-up-Location
    □*',
contentPadding: EdgeInsets.
    all(8),
border: OutlineInputBorder()
    ,
),
style: TextStyle(fontSize: 30)
    ,
maxLines: 1,
minLines: 1,
),
SizedBox(height: 10),
TextField(
    controller: returnLoc,
    textAlign: TextAlign.center,
    decoration: InputDecoration(
        suffixIcon: IconButton(
            color: Colors.white,
            onPressed: () {},
            icon: Icon(
                Icons.location_on,
                color: Colors.black,
            ),
        ),
    ),
//prefixIcon: Icon(Icons.
    timer),
filled: true,
fillColor: Colors.grey.
    shade100,
labelText: 'Drop-Off-

```

```

198         Location_*',
        contentPadding: EdgeInsets.
199           all(8),
        border: OutlineInputBorder()
200       ),
201       style: TextStyle(fontSize: 30)
202     ),
203     maxLines: 1,
204     minLines: 1,
205   ),
206   const SizedBox(
207     height: 10,
208   ),
209   Container(
210     height: 300,
211     child: GoogleMap(
212       initialCameraPosition:
213         CameraPosition(
214           target: LatLng(22.3569,
215             91.7832), zoom:
216             14.245),
217       mapType: MapType.normal,
218       onMapCreated: (
219         GoogleMapController
220         controller) {
221         _controller.complete(
222           controller);
223       },
224       myLocationEnabled: true,
225       compassEnabled: true,
226     ),
227   ),
228   const SizedBox(
    height: 10,
  ),
  Column(
    children: [
      Row(
        children: [
          const SizedBox(

```

```

229         width: 5,
230     ),
231     SizedBox(
232         width: (MediaQuery.of(
233             context).size.width
234         ) * .40,
235         child: TextField(
236             controller:
237                 pickUpDate,
238             textAlign: TextAlign
239                 .center,
240             decoration:
241                 InputDecoration(
242                     suffixIcon:
243                         IconButton(
244                             color: Colors.
245                                 white,
246                             onPressed: () {
247                                 _selectDateFromPicker
248                                     (context);
249                             },
250                             icon: Icon(
251                                 Icons.
252                                     calendar_today_outlined
253                                     ,
254                                     color: Colors.
255                                         black,
256                                 ),
257                             ),
258                     //prefixIcon: Icon
259                         (Icons.timer),
260                     filled: true,
261                     fillColor: Colors.
262                         grey.shade100,
263                     labelText: 'pick-
264                         up-date',
265                     contentPadding:
266                         EdgeInsets.all
267                             (8),
268                     border:
269                         OutlineInputBorder

```

```

253         ),
254         style: TextStyle(
255             fontSize: 20),
256         maxLines: 1,
257         minLines: 1,
258     ),
259     const SizedBox(
260         width: 5,
261     ),
262     SizedBox(
263         width: (MediaQuery.of(
264             context).size.width
265         ) * .40,
266         child: TextField(
267             controller:
268                 pickUpTime,
269             textAlign: TextAlign
270                 .center,
271             decoration:
272                 InputDecoration(
273                     suffixIcon:
274                         IconButton(
275                             color: Colors.
276                                 white,
277                             onPressed: () {
278                                 _selectTime(
279                                     context);
280                             },
281                             icon: Icon(
282                                 Icons.timer,
283                                 color: Colors.
284                                     black,
285                             ),
286                         ),
287                     //prefixIcon: Icon
288                         (Icons.timer),
289                     filled: true,
290                     fillColor: Colors.
291                         grey.shade100,

```

```

281         labelText: 'pick-
282             up-time',
                contentPadding:
                    EdgeInsets.all
                    (8),
283         border:
            OutlineInputBorder
            (),
284     ),
285     style: TextStyle(
        fontSize: 20),
286     maxLines: 1,
287     minLines: 1,
288 ),
289 ),
290 ],
291 ),
292 const SizedBox(
293     height: 15,
294 ),
295 Row(
296     children: [
297         const SizedBox(
298             width: 5,
299         ),
300         //Text("pic-Up_Date"),
301         SizedBox(
302             width: (MediaQuery.of(
                context).size.width
                ) * .40,
303         child: TextField(
304             controller:
                returnDate,
305             decoration:
                InputDecoration(
306                 suffixIcon:
                    IconButton(
307                         color: Colors.
                            white,
308                         onPressed: () {
309                             _selectDateFromPicker2

```



```

310         (context);
311     },
312     icon: Icon(
313         Icons.
314             calendar_today_outlined
315             ,
316             color: Colors.
317                 black,
318             ),
319     ),
320     //prefixIcon: Icon
321     (Icons.timer),
322     filled: true,
323     fillColor: Colors.
324         grey.shade100,
325     labelText: 'return
326         -date',
327     contentPadding:
328         EdgeInsets.all
329         (8),
330     border:
331         OutlineInputBorder
332         (),
333     ),
334     style: TextStyle(
335         fontSize: 20),
336     maxLines: 1,
337     minLines: 1,
338     ),
339 ),
340 const SizedBox(
341     width: 5,
342 ),
343 SizedBox(
344     width: (MediaQuery.of(
345         context).size.width
346     ) * .40,
347     child: TextField(
348         controller:
349             returnTime,
350         decoration:

```

```

336         InputDecoration(
suffixIcon:
337             IconButton(
color: Colors.
338                 white,
onPressed: () {
339                 _selectTime2(
context);
340             },
341             icon: Icon(
342                 Icons.timer,
343                 color: Colors.
black,
344             ),
345         ),
346         //prefixIcon: Icon
(Icons.timer),
347         filled: true,
348         fillColor: Colors.
grey.shade100,
349         labelText: 'return
-time',
350         contentPadding:
EdgeInsets.all
(8),
351         border:
OutlineInputBorder
(),
352     ),
353     style: TextStyle(
fontSize: 20),
354     maxLines: 1,
355     minLines: 1,
356 ),
357 ),
358 ],
359 ),
360 ],
361 ),
362 const SizedBox(
363     height: 25,

```

```

364         ),
365         Container(
366             margin: EdgeInsets.all(10),
367             child: ElevatedButton(
368                 onPressed: () {
369                     addtrip();
370                     Navigator.pushNamed(
371                         context, 'search');
372                 },
373                 child: Text("Search"),
374                 style: ElevatedButton.
375                     styleFrom(
376                         primary: Colors.
377                             yellowAccent,
378                         onPrimary: Colors.black,
379                         textStyle: TextStyle(
380                             color: Colors.black,
381                             fontSize: 40,
382                             fontStyle: FontStyle.
383                                 normal),
384                     ),
385             ),
386         ),
387     ),
388 ),
389 );
390 }
391 }
392 }

```

Listing 2: hirecar.dart file containing the "HireCar" class

For the simplicity of the report, we have not listed all the implementation files related to other classes of our project. Other implementations can be found in the GitHub link [here](#).

10 Testing

Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use. Testing cannot demonstrate that the software is free of defects or that it will behave as specified in every circumstance .

As Edsger Dijkstra, an early contributor to the development of software engineering, eloquently stated:

””Testing can only show the presence of errors, not their absence.””

Testing is part of a broader process of software verification and validation. Verification and validation are not the same things, although they are often confused. Barry Boehm, a pioneer of software engineering, succinctly expressed the difference between them.

Validation: Are we building the right product?

Verification: Are we building the product right?

Three types of testing are discussed in this section. The unit testing is discussed in Section 10.1. Integration testing is discussed in Section 10.2. System testing is discussed in Section 10.3 Finally, we have discussed the acceptance test in Section 10.4

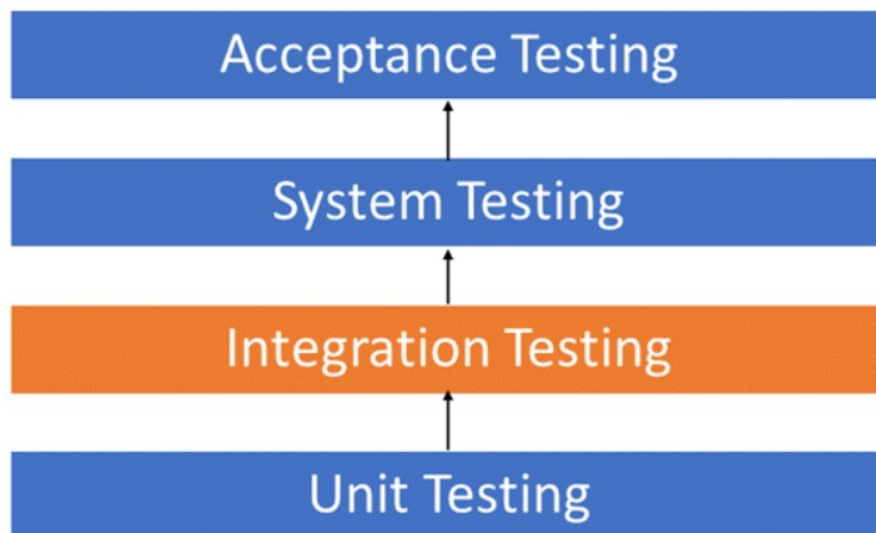


Figure 27: Testing Sequence

10.1 Unit Testing

Unit testing is the process of testing program components, such as methods or object classes. Individual functions or methods are the simplest types of components. We test these routines with different input parameters [3]. This testing method is also the first level of software testing, which is performed before other testing methods such as integration testing which is discussed in Section 10.2

10.2 Intregation Testing

Integration testing – also known as integration and testing (IT) – is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers. Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as ‘IT’ (Integration and Testing), ‘String Testing’ and sometimes ‘Thread Testing’.

10.3 System Testing

System testing is defined as testing of a complete and fully integrated software product. This testing falls in black-box testing wherein knowledge of the inner design of the code is not a pre-requisite and is done by the testing team.

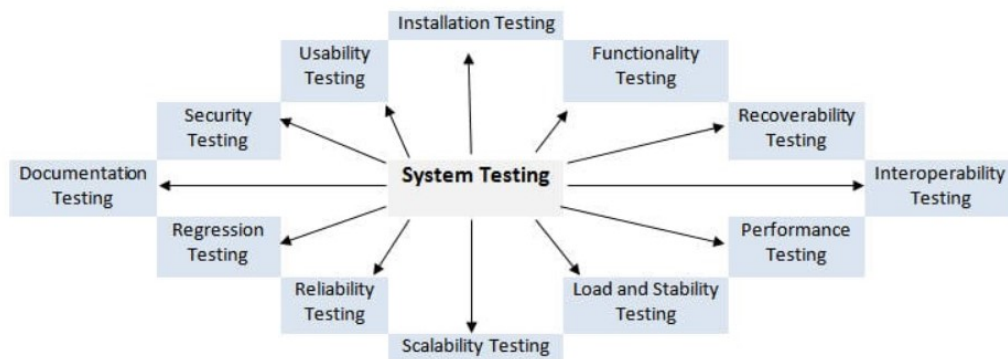


Figure 28: System Testing

10.4 Acceptance Testing

Acceptance testing is where customers test a system to decide whether or not it is ready to be accepted by the system developers and deployed in the customer environment. The purpose of this testing is whether the program works accordingly to the requirements or not. Some requirements for acceptance testing are discussed below:

Requirement-1: A user must be able to search Available car.

In Figure 29 There are a pickup location,date and time;also there are drop up location,date and time.Then he can search This requirement must be needed.In our system, any user can do this task.So, this requirement is fulfilled in our system.

The screenshot shows a mobile application interface for car rental. At the top, a blue header bar contains a back arrow and the text "Rent a Car and Drive Happy". Below this, a white box titled "Start a Reservation" includes a note "*indicate required field". There are two input fields: "pick-up-Location *" with a location pin icon and "Drop-Off-Location *" with a location pin icon. Below these is a map of Chattogram, showing areas like Pahartali and Saraipara. Under the map, there are four input fields for dates and times: "pick-up-..." with a calendar icon, "pick-up-..." with a clock icon, "return-..." with a calendar icon, and "return-..." with a clock icon. A large yellow "Search" button is positioned below these fields. At the bottom, a blue navigation bar features four icons: a magnifying glass labeled "Hire Car", a bell labeled "Notification", a circular arrow labeled "Trip", and a person icon labeled "Person". The very bottom of the screen shows the standard Android navigation bar with three icons: a square, a circle, and a triangle.

Figure 29: Search Available Car

Requirement-2: A user must be able to see and delete notification.

In Figure 31 There are delete option button.If he he want to delete notification,he can. This requirement must be needed.In our system, any user can do this task.So, this requirement is fulfilled in our system.



Figure 30: See Notification

Requirement-3: A user must be able to see trip history and rated them.

In Figure 31 There are rating filled button.If he he want to rate trip ,he can do it.. This requirement must be needed.In our system, any user can do this task.So, this requirement is fulfilled in our system.

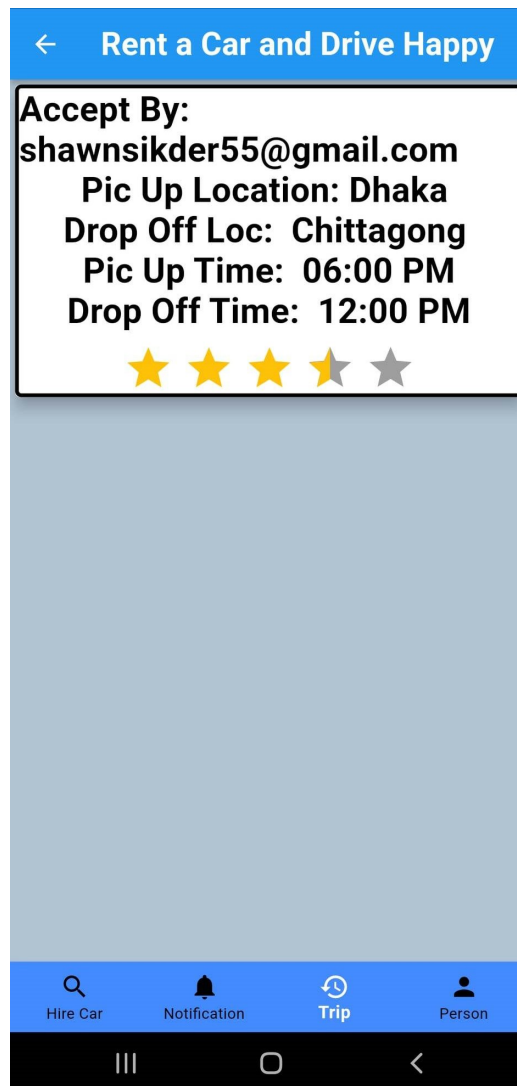
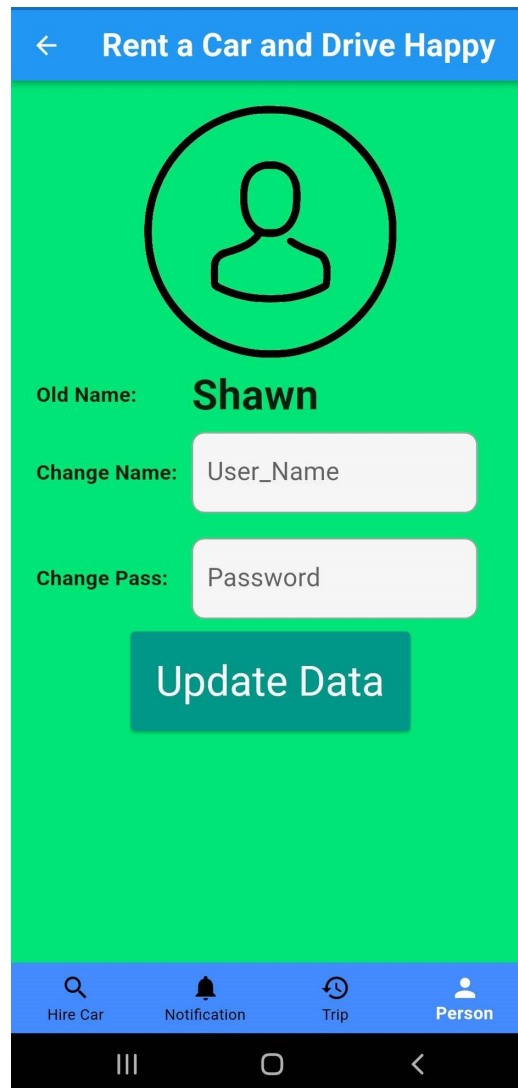


Figure 31: See Trip History

Requirement-4: A user must be able to update his/her Profile. In Figure 32 There are a change name and password option This requirement must be needed. In our system, any user can do this task. So, this requirement is fulfilled in our system.



The image shows a mobile application interface for updating a user profile. At the top, a blue header bar contains a back arrow and the text "Rent a Car and Drive Happy". Below this is a green background area. In the center of the green area is a circular placeholder for a profile picture, represented by a black outline of a person. Below the profile picture, the text "Old Name:" is followed by the name "Shawn". Underneath, there are two input fields: "Change Name:" with the placeholder text "User_Name" and "Change Pass:" with the placeholder text "Password". Below these input fields is a large, teal-colored button with the text "Update Data". At the bottom of the screen is a blue navigation bar with four icons and labels: a magnifying glass for "Hire Car", a bell for "Notification", a circular arrow for "Trip", and a person icon for "Person". The "Person" icon is highlighted. Below the navigation bar is a black Android-style status bar with three icons: a hamburger menu, a circle, and a back arrow.

Figure 32: Profile update

Requirement-5: A car owner must be able to accept and cancel request.

In Figure 33 There are accept and cancel option button.If he he want to accept request request then he click accept button,else he click cancel button. This requirement must be needed.In our system, any user can do this task.So, this requirement is fulfilled in our system.

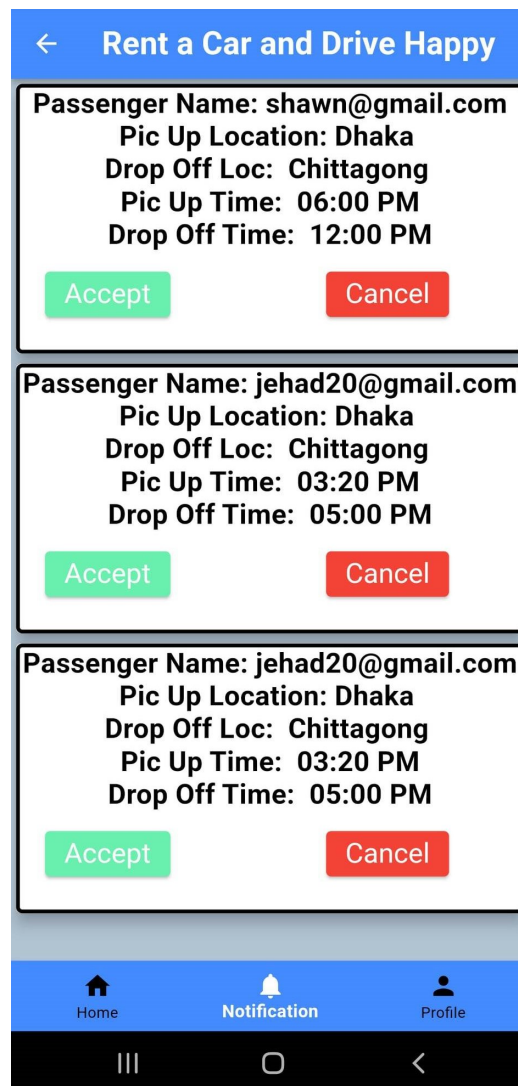
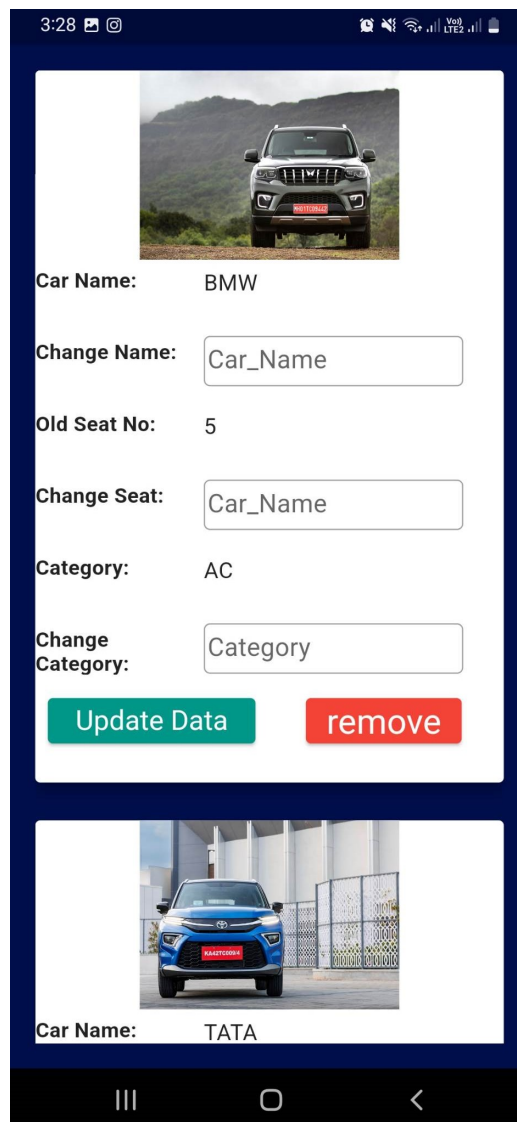


Figure 33: Owner See Notification

Requirement-6: A car owner must be able to update and remove car.

In Figure 34 There are update data and remove optibutton.If he he want to update car information then he click update data,,else he click remove button. This requirement must be needed.In our system, any user can do this task.So, this requirement is fulfilled in our system.



The screenshot displays a mobile application interface with a dark blue header and footer. The main content area is white and contains two car listings. The top listing features a white SUV on a road, with the text 'Car Name: BMW' below it. Further down, there are input fields for 'Change Name:' (containing 'Car_Name'), 'Old Seat No:' (containing '5'), 'Change Seat:' (containing 'Car_Name'), and 'Category:' (containing 'AC'). Below these fields are two buttons: a green 'Update Data' button and a red 'remove' button. The bottom listing features a blue car in a parking lot, with the text 'Car Name: TATA' below it. The status bar at the top shows the time as 3:28 and various icons. The bottom navigation bar has three icons: a hamburger menu, a circle, and a back arrow.

Figure 34: Owner See Notification

11 Software Deployment

Software deployment includes all of the steps, processes, and activities that are required to make a software system or update available to its intended users. Today, most IT organizations and software developers deploy software updates, patches and new applications with a combination of manual and automated processes. Some of the most common activities of software deployment include software release, installation, testing, deployment, and performance monitoring.

11.1 Software Deployment Methodologies

DevOps is a methodology and a set of best practices for software development whose primary goals are to shorten delivery times for new software updates while maintaining high quality. In the DevOps framework, there are seven steps in the software development process:

1. Coding
2. Building
3. Testing
4. Packaging
5. Releasing
6. Configuring
7. Monitoring

11.2 Software Deployment For Our System

Entering our system, a user see a login page.If one log on the system as passenger,he can see a navigation bar.The Navigation ber is four division.They are Hirecar,Notification,TripHistory,Profile.If a man want to hire a car he can search by giving pickup location,drop up location,pic up date and time,drop up date and time.Then see the available car and he can request.If the car owner accept his request ,he will go to trip and rating them in TripHistory.Here is also a profile bar where he can change his picture,name and password.

Second one user means car services center can login into the system and go the home navigation bar.They can add car,add driver ,Update car,add car on trip,add available car.In notification bar,service center can see the request and cencel the request.Finallay there is a profile funtion where center can update their information.

12 Conclusion and Future Work

Car rental business has emerged with a new goodies compared to the past experience where every activity concerning car rental business is limited to a physical location only. Even though the physical location has not been totally eradicated; the nature of functions and how these functions are achieved has been reshaped by the power of internet. Nowadays, customers can reserve cars online, rent car online, and have the car brought to their door step once the customer is a registered member or go to the office to pick the car.

The web based car rental system has offered an advantage to both customers as well as Car Rental Company to efficiently and effectively manage the business and satisfies customers' need at the click of a button.

Appendices

A Interviews

Interview: 01

This interview is conducted between the team and a passenger Md.Jahangir.

The Team: What to do if a person have to rent a car ?

Interviewer: He should go to the Car center.

The Team: What are you do if there is no available car?

Interviewer: Then have go to another car center.

The Team: What to do in case of using our System?

Interviewer: First ensure that which the car center have available car if he search for hire car.

The Team: What's about notification?

Interviewer: If he request for hire a car,then car center response by notification.car Center can accept or cancel request.

The Team: What about the car driver?

Interviewer: can see the driver profile.Can be known driver skills,age,name...

The Team: What can you see the car ?

Interview: i can see the car pictures and car's seat number,car category ac/non-ac.

The Team: Thank you, sir.

Interview: 02

This interview is conducted between the team and a computer Scientist.

The Team: We want to develop software that can help in case of car renting. As a computer scientists, do you think your software will be user effective/friendly?

Interviewer: Yes, such type of software will be helpful.

The Team: How our system can decrease anyone money?

Interviewer: Ensure that the car can rent any person in any time save money if it is not necessary to go car rent center .Then time and money both are save

The Team: What to do in our system?

Interviewer:your system must be updated day by day according to time.

The Team: Do you want to say something?

Interviewer: Couldn't say anything.

The Team: Do you install our system ?

Interviewer: yes i was install your system for rent a car..

The Team: what is your trip.??

Interviewer:The trip was fantastic.And the driver aslo very good person.

The Team: Do you wants service again and again for our system ?

Interviewer: I can not say.if necessary i use your stsyem.

The Team: Any suggestions can you give us for our system?

Interviewer: No, I wanna say good luck, Thank you.

The Team: Thank you.

13 Bibliography

- 1.System, Online. 'Online Car Rental System'. Academia.edu. N.p., 2015. Web. 9 June 2015.
- 2.Scribd.com, Online. '49930505 Car Rental System Project Report'. N.p., 2015. Web. 9 June 2015.
- 3.Scribd.com, Online. 'Car Rental System Documentation'. N.p., 2015. Web. 9 June 2015.
- 4.Freelancer, Online. 'Project Documentation Car Rental Company Software Development Freelancers and Jobs - Freelancer'. N.p., 2015. Web. 9 June 2015.
- 5.Slideshare.net, Online. 'Zook Car Rental System Project'. N.p., 2015. Web. 9 June 2015.
- 6.Kaewman, Sasitorn. 'Online Decision Support System of Used Car Selection using K-Nearest Neighbor Technique'. IJFCC (2012): 164-166. Web.
- 7.Wikipedia, Online. 'Use Case Diagram'. N.p., 2015. Web. 9 June 2015.
- 8.Wikipedia, Online. 'Activity Diagram'. N.p., 2015. Web. 9 June 2015.
- 9.Tutorialspoint.com,Online'UML-Activity Diagrams'N.p,2015.Web.9 June 2015
- 10.Laudon, Kenneth C, and Jane Price Laudon. Management Information Systems. Upper Saddle River, NJ: Prentice Hall, 2000. Print.
- 11.Menkus, Belden. 'Car Rental Chain Former Owners Charged With Computer Frauds'. Computer Fraud Security Bulletin 1993.3 (1993): 3-4. Web.
- 12.Li, Zhang. 'Design And Realization Of Car Rental Management System Based On AJAX+SSH'. Information Technology J. 12.14 (2013): 2756-2761. Web.