CAR RENTAL SYSTEM

Project Report Submitted By

PRADEEP SOJAN

Reg. No: AJC20MCA-2057

In Partial fulfillment for the Award of the Degree

MASTER OF COMPUTER APPLICATIONS (2 YEAR)
(MCA)
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY

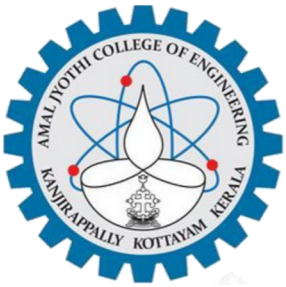
[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2020-2022

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, "**DRIVE-IT CAR RENTAL SYSTEM"** is the bonafide work of **PRADEEP SOJAN (Reg.No:AJC20MCA-2057)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

Ms Ankitha Philip
Internal Guide

Rev. Fr. Dr. RubinThottupurathu Jose
Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

External Examiner

DECLARATION

I hereby declare that the project report **"CAR RENTAL SYSTEM"** is a bonafided work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

Date: 21/07/2022 **PRADEEP SOJAN**

KANJIRAPPALLY **Reg.No:AJC20MCA-2057**

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I would like to express my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and our Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev.Fr.Dr. Rubin Thottupurathu Jose** for helping me. I extend my whole hearted thanks to the project coordinator **Rev.Fr.Dr. Rubin Thottupurathu Jose** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also like to express sincere gratitude to my guide**, Ms**. **Ankitha Philip** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

PRADEEP SOJAN

ABSTRACT

This Car Rental System project is designed to help the small car rental organizations to enable renting cars through an online system. It helps the clients to view the available cars and book the cars for the time period. The client can also view their profile and edit their profile details. It has an easy to understand interface which helps the client to check for cars and rent them for the period determined. They could likewise make payment online. Based on a type of car required by the client, the client will have able to make bookings. The utilization of internet technology has made it simple for the clients to rent a car at anytime. This Car Rental System makes the booking easy. It spares time and work. The apparatus will approach the client for data, for example, the date and time of journey, kind of car and so forth. Likewise, it will require a recognizable identification. Utilizing these details, the tool will help the client with booking a car for the journey. There is also an option to hire a driver if client already has a vehicle and no driver. Anyone with an Indian driving license can apply to be a driver by registering in the drivers registration page and is approved by the admin.

CONTENT

List of Abbreviation

IDE	-	Integrated Development Environment
UML	-	Unified Modeling Language
UI	-	User interface
API	-	Application programming interfaces
SDK	-	Software Development Kit

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The real power of this project is not just rent cars, but in the formation of stronger relationships with customers and delivering of a high level of support and service, which in turn improves establishment sales and its goodwill. Our project's primary goal is to create an application software that will lessen the amount of manual work required to manage bookings, payments, clients, drivers, and cars. The Car Rental System is a software application which allows the user to easily rent cars according to their needs. The user also has the option to request for a driver along with the car. The project have user friendly interface and accurate information and description of the car the customer wants to rent. Finally after renting the user can download the PDF generated containing all the details of their booking. In the future various features like first aid services can be added.

1.2 PROJECT SPECIFICATION

The proposed system is an app in which customer can book car for renting. Also the customer can request for driver along with the car.

The system includes 4 actors. They are:

1.Admin

Admin manages and controls the system . Admin can verify/authenticate drivers and car owners and has the ability to block them. Admin can view all the registered users and also manage them. Admin can also check the price of some used cars to better understand the rates of the used car in the current market. Admin can view the complaints given by the customers and take necessary actions. Admin gets a small set cut of the rent and can view all the car bookings that has completed.

2.Customer/User

Customer/User can register in the car rental system after providing necessary information. They can then view all the available cars and their details and can rent a suitable car of their choice. After renting the car they can pay the overall amount. They can request for a driver separately if they require. They can communicate with the driver and even raise complaints.

3.Driver

Driver first needs to be authenticated/verified by the Admin. Driver then can accept the

requests sent by the users and take up the job. Drivers can call the user to communicate and clarify any needs or requirements.

4.Car Owner

Car Owner can add cars to the system which includes giving the required details about the car and view their current status. The user after completing the car rent can request for bill generation to the car owner. The car owner generates the bill and set percentage is also gone to the admin.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

In technical terms system analysis can be referred to the process of collecting and analysing these collected data to resolve the errors in the data. These data can be used in the system to make necessary changes. During this process the users of the system and system analysts must communicate efficiently for better information gathering. System analysis is the first and foremost step in the software development process. The system is carefully inspected and evaluated. The system analyst adopts the position of an investigator and probes the inner workings of the existing system. The project is observed, and the input to the system is recognized. The different processes can be connected to the organisational outputs. System analysis involves comprehending the errors or faults, recognizing the crucial and relevant factors, analysing, and synthesising the numerous components, and choosing the best and most acceptable methodologies.

The procedure must be carefully investigated utilizing a range of approaches, such as surveys and interviews. The data which is collected from various resources should be thoroughly evaluated to draw a conclusion. knowing how the system functions is the main conclusion that can be made. This system is known as the present one. Now, problem areas have been identified after a detailed examination of the present system. The problems faced in the system are managed and controlled by the system designer. The proposal for solutions are initially detected. By examining the proposal that is put forward, A most suitable solution is adopted. The proposal is presented to the user with the option to accept or reject it. On the basis of user requests, the plan is assessed for any necessary revisions. This is a loop like procedure ends when all the user requirements are met.

Primary research is the process in which gathering and analysing data are takes place which in turn used for upcoming system investigations. Initial research requires strong collaboration between system users and developers since it involves problem-solving. It carries out a number of feasibility studies. An overview of the project activities can be recognized by these studies, which may be utilised to choose the methods to be applied for effective system research and analysis. The information’s gathered during these processes is much more helpful for the system designers and system developers during the development stage of the system. In fact this information helps them to identify what exactly the customer needs.

2.2 EXISTING SYSTEM

Car Rental System will help the users with booking a car for some fee determined. Till now there was no easily accessible app to help the clients to rent cars in a small centre or establishment. Also the task of requesting a driver along with the ride was not properly conducted. It was a difficult task to manage the rental system. Keeping track of all the rental cars was also an issue.

The current system must be altered in order to include new data, increase its efficiency, and make it more adaptable and secure. The system should me made more easier for user to use and must be practical.

2.3 DRAWBACKS OF EXISTING SYSTEM

- No proper online management of small car rental system.
- UI is not user-friendly.
- Poor transparency and customer service
- complex booking and pricing management

2.4 PROPOSED SYSTEM

The suggested project is intended to address every drawback of the current system. For the rental business to flourish, it is essential to have a system that is more user-friendly and appealing to users; based on this idea, a system is offered. The administrator in our suggested system has access to see every consumer. Customers can use it to schedule their bookings and complete purchases using an online payment method. They can also request for a driver separately. The administrator, clients, drivers, and car owners are the system's planned users.

This application maintains the data in a central location that is simultaneously accessible to all users. Managing historical data in a database is extremely simple. To utilize this programme, distributors don't need any special training. The system is very easy to use They may easily utilize the tool that reduces the number of manual hours needed to complete routine tasks, improving performance. The information about internet transactions can be easily recorded in databases.

2.5 ADVANTAGES OF PROPOSED SYSTEM

The system is relatively easy to implement and design. The system works in practically all settings and uses very little system resources. It has the following characteristics:

- **Provides improved security: -**

Users' security and privacy are vastly more safeguarded in the system. Users receive unique login credentials that have undergone extensive verification. The user password is encrypted in the database. Data security requires preventing access to unauthorized. Data protection refers to its protection against various forms of deletion. The four interrelated issues that make up the system security challenge are security, consistency, privacy, and confidentiality. By needing a email and password to log in, security is preserved. Data security will be guaranteed since we retain the documents in secure databases.

- **Guarantees data accuracy: -**

The proposed system helps to avoid mistakes made while inputting user details during the registration process. Additionally, the system's architecture prevents it from storing extraneous information like useless history, etc. The system does not keep irrelevant information regarding the customers.

- **Provides greater service: -**

The proposed system eliminates the need of hard copy keeping. We can also save time and money by carrying out the same task in a different way. It is possible to save the data for a longer period without losing any of it. The status of the sales and payment may be simply comprehended with the aid of a better and more effective report production system.

- **Better Interface: -**

The system's user interface is substantially more straightforward. The user's system can communicate with the system without any specialized knowledge. The reports and status are effectively streamlined and optimized. As a result, the system's users may simply comprehend the information pertaining to system transactions.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

To determine if the project will finally achieve the organization's aims and be worthwhile of the work, effort, and time put in it, a feasibility study is carried out. The developer can forecast the project's usefulness and possible future thanks to a feasibility study. The premise for a feasibility study is the system proposal's viability, includes impacts on the business, the capacity to satisfy consumer demands, and efficient resource usage. As a result, the proposals undergo a feasibility evaluation. The project's technical, economic, and operational viabilities, as well as other elements that were carefully considered throughout this project's feasibility study, are all included in the paper's description of the project's viability.

The features are: -

3.1.1 Economic Feasibility

Analyses of costs and benefits are necessary to support the developing system. It is done to ensure that the system will develop within the economic constraints. The cost associated with the system which is intended to develop must be calculated.

Some important finance related questions raised are during development are:

- The expenses of doing a thorough system examination.
- The price of dependencies needed to develop the system.
- The advantages in terms of lower expenses or less expensive mistakes.

The suggested system is a project-related development, thus there are no additional manual expenses. Additionally, the fact that all of the resources are already at hand indicates that the system may be developed affordably.
The cost of the DRIVE-IT project was broken down into three categories: system costs, development costs, and hosting costs. All estimates indicate that the project was created at a modest cost. As open source software was used to create it entirely.

3.1.2 Technical Feasibility

First, the project has to be gauged technically. For the assessment of this viability, a general design of the system's requirements for input, output, programmers, and processes must be used as the foundation. After identifying an outline system, the inquiry must next recommend the type of equipment, essential steps for building the

system, and methods of operating the system after it has been built.

The investigation ran into the following technological issues:

- I the existing technologies are sufficient for developing the proposed system.?
- Is there any possibility to scale the system as part of upgradation.?

The system has to be developed such that it should be implemented within the available constraints and limitations. Even if the technology could become outdated after a while, the system can still be utilized because newer versions of the same program support previous ones. There are therefore few restrictions associated with this undertaking.The minimum requirements for running the system are a system with processor which is greater than or equal to Intel Core i3 and minimum Ram of size 4GB And also hard disk of storage capacity 256Gb is recommended.

3.1.3 Behavioral Feasibility

The proposed system leads to questions such as:

- Do the users of the system get sufficient support?
- Is there any chance for data loss or harm?

Since the project meets the preset constraints, it is more feasible and advantageous. The project is assumed to be behaviorally viable after calculating all behavioral factors. While examining the factors which leads to challenges in the implementation, the system will prevent it to a great extent.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	-	Intel core i3 or i5
RAM	-	8 or 16 GB
Solid state/Hard disk	-	500 GB

3.2.2 Software Specification

Front End	-	Dart
Backend	-	Dart, Firebase
Client on PC	-	Windows 7 and above.
Technologies used	-	Dart, Flutter, Firebase

3.3 SOFTWARE DESCRIPTION

3.3.1 Firebase

Firebase is a set of tools or a platform that you can use to enhance, create and expand your web or mobile application. It was originally developed by an independent company founded in 2011. Google acquired the platform in 2014. A year later google acquired a web hosting platform named Divshot and merged it with the firebase team and it is now their flagship offering for application development. The resources it offers deals with a substantial amount of the services that software enthusiasts would ordinarily be needed to create on their own but don't really want to do because they would rather focus on the user exposure of the app. This area includes items like analytics, authentication, databases, configuration, file storage, push messaging, and a lengthy list of other things. The cloud-hosted services scale with little to no additional development work.

Firebase introduced a real time document database in October 2017 named Cloud firestore, successor to the original firebase Realtime Database.

Because they are stored on the cloud, Google manages and controls all of the backend components for the goods. There is no need to set up any middleware between your app and the service because Firebase's client SDKs speak straight to these backend services. In order to query the database in your client app while using one of the Firebase database choices, you normally write code.

3.3.2 Flutter

Google built and unveiled Flutter, a free and open-source mobile UI framework, in May 2017. Simply said, it enables you to develop a native mobile application using only one codebase. This implies that you may construct two distinct apps using a single programming language and codebase (for iOS and Android).Flutter is made up of two crucial components:

- An An SDK (Software Development Kit) is a group of tools that will assist you in creating your apps. Tools for converting your code to native machine code are also included (code for Android and iOS).
- A Framework (User Interface Library which is based on widgets) is a set of reusable User Interface components (such as buttons, text input fields, sliders, and other elements) that you may customize to meet your specific requirements.

You will utilize the programming language Dart to create Flutter applications. It is an typed object programming language. Although Google first developed the language in October 2011, it has significantly advanced since then. Dart is a front-end development language that can be used to build both online and mobile apps. The syntax of Dart is comparable to JavaScript.

Flutter executes on the Dart virtual machine, which has a just-in-time execution engine, when developing and debugging an application. This permits quick compilation times as well as "hot reloading," which lets changes to source files be pushed into an application that is already executing. Flutter takes this a step further by supporting stateful hot reloading, in which most source code changes are instantaneously implemented in the current app without the need for a reboot or any information loss.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Design is the first step in the development of any engineering system or product. Design is an inventive and ingenious procedure. A great design is the key to an efficient system. The act of specifying a process or system in adequate detail to allow for its physical implementation utilizing various approaches and concepts is referred to as "design." It may be described as the practice of applying numerous methodologies and concepts to specify a device, method, or system in sufficient depth to allow for its physical implementation. Regardless of the development paradigm chosen, software design forms the technical core of the software engineering process. The architectural characteristic needed to construct a product or a system is developed through the system design. This software has also experienced the highest suitable conceptual design fine optimizing all productivity, performance, and accuracy levels, as in the case of any systematic technique. Throughout the design phase, a user-oriented document is turned into a document for database employees or programmers. A project or system design is created in two stages: physical and logical design.

4.2 UML DIAGRAM

The artefacts of a software system are described, shown, constructed, and recorded using the common language known as UML. A draught of the UML 1.0 specification was provided to the Object Management Group (OMG), which was in charge of creating UML, in January 1997.

UML stands for Unified Modeling Language. UML is distinct from other popular programming languages such as C++, Java, COBOL, and so on. To generate software designs, a visual language known as UML is employed. UML is a general-purpose visual modelling language for software system visualization, specification, construction, and documentation. UML is used for more than only representing software systems, despite the fact that this is its most popular application. It is also used to imitate non-software-based systems. For example, the process flow of a manufacturing plant, etc. Although UML is not a programming language, tools in a variety of languages can be used to generate code from UML diagrams. UML is intimately related to the analysis and design of object-oriented systems. A comprehensive UML diagram that depicts a system is made up of all the parts and relationships. The most crucial aspect of the entire

procedure is the UML diagram's aesthetic impact. It is completed by using all the other components. The UML diagrams consists of the following diagrams.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Activity diagram
- State chart diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

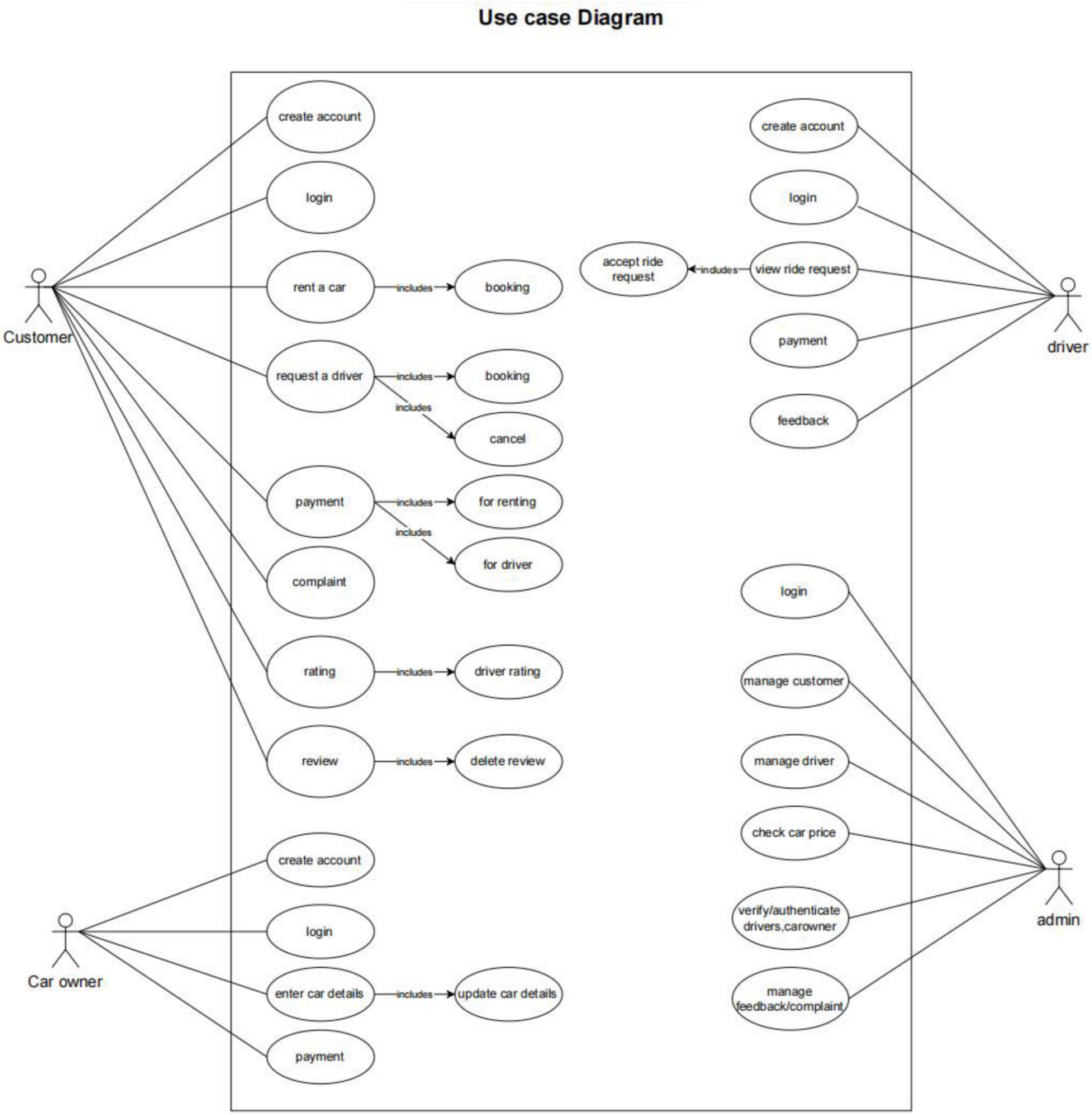
A use case diagram is an illustration of the interactions amongst system components. A approach for identifying, outlining, and organizing system requirements is called a use case. The word "system" here refers to a thing that is being created or run, like a website for mail-order goods sales and services. UML (Unified Modeling Language), a common language for modelling actual objects and systems, uses use case diagrams.

The planning of general requirements, the validation of a hardware design, the testing and debugging of a software product in development, the creation of an internet support guide, or the completion of a job focused on customer support are all examples of system objectives. For instance, use cases in a product sales context can involve customer service, item ordering, catalogue updating, and payment processing. In a use case diagram, there are four elements.

- The line separating the system of interest from the environment around it. The actors, usually individuals involved with the system defined according to their roles.
- The players inside and surrounding the system perform the use cases, which are the specialized roles.
- The connections and inter-dependencies between the actors and use cases.

Use case diagrams are created to depict a system's functional needs. To create an effective use case diagram after identifying the aforementioned things, we must adhere to the following rules. The name of a use case is very important. It should be selected in such a way so that it can identify the functionalities performed.

- Give the actors names that fit them.
- Clearly depict links and dependencies in the diagram.
- As the primary function of the diagram is to establish the needs, avoid attempting to incorporate all kinds of linkages.
- When necessary, take notes to help you remember certain crucial details.



4.2.2 SEQUENCE DIAGRAM

A sequence diagram simply demonstrates the sequential order in which events occur or how they interact with one another. Event diagrams and event scenarios are other names for sequence diagrams. Sequence diagrams display the activities performed by a system's parts in time order. Sequence diagrams are extensively used by business people and software engineers to document and explain the essential requirements for existing and new systems.

Sequence Diagram Notations –

i. **Actors –** An actor in a UML diagram symbolizes a certain sort of role in which it interacts with the system's elements. An actor is never inside the scope of the system that we want to describe using the UML diagram. For a range of roles, including those of human users and other external topics, we use actors. An actor is shown using the stick person notation in a UML diagram. A sequence diagram could have several actors.

ii. **Lifelines –** A lifeline is a named element depicting an individual participant in a sequence diagram. A lifeline represents each event in a sequence diagram. The lifeline elements of a sequence diagram are at the top.

iii. **Messages –** Messages are used to depict the communication between users. The messages are arranged in a sequential order in which they occur. Th messages are depicted in the diagram with the help of arrows. The core sequence of the sequence diagrams is formed by lifelines and the messages.

Massages can be classified into categories such as:

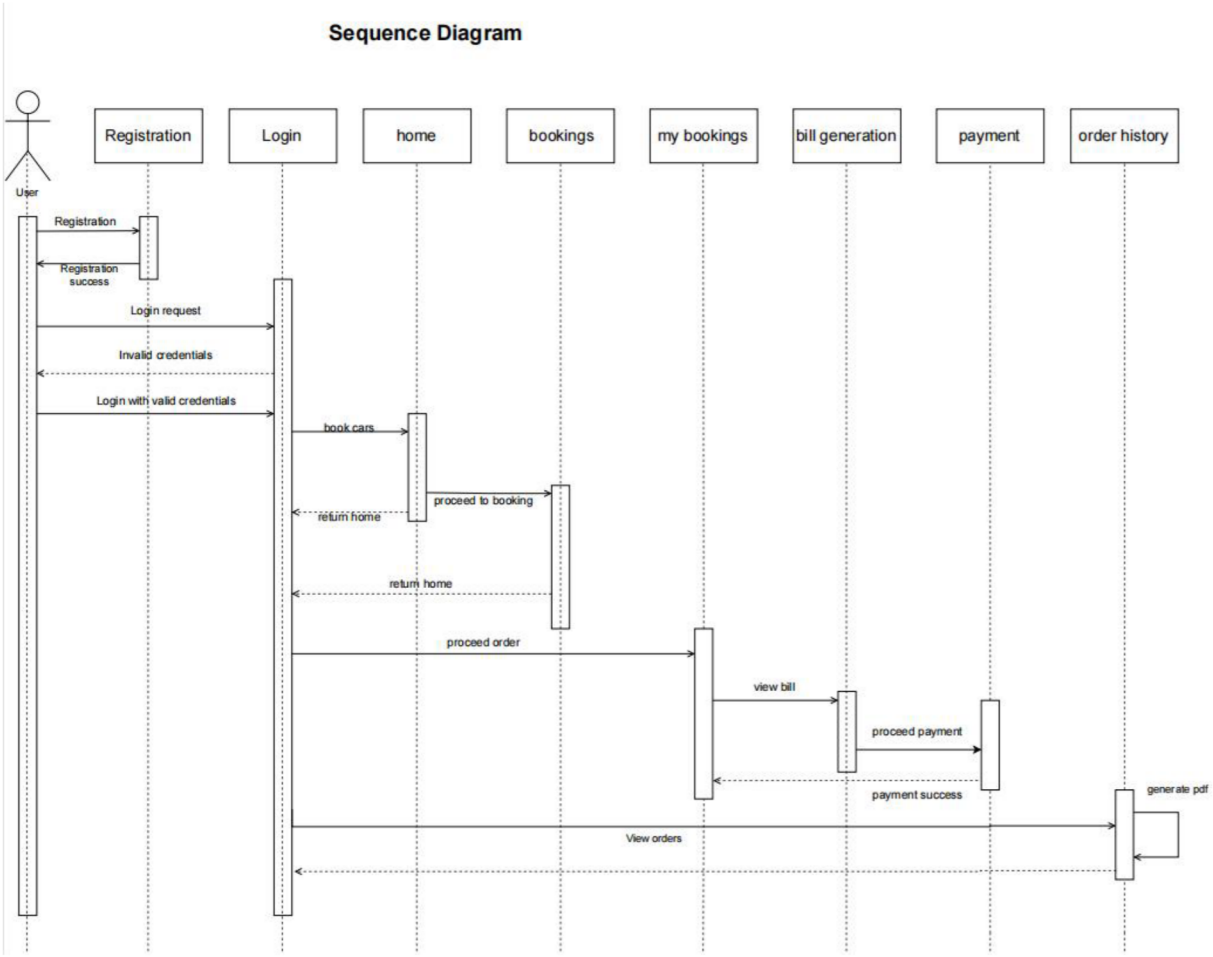
- Synchronous messages
- Asynchronous Messages
- Create message
- Delete Message
- Self-Message
- Reply Message
- Found Message
- Lost Message

iv. **Guards –** The conditions in the UML diagram are depicted using Guards. The

guards are used to block the messages flow when the preset conditions are met. Guards has a crucial role in showing how the constraints attached to a system or a particular process to developers.

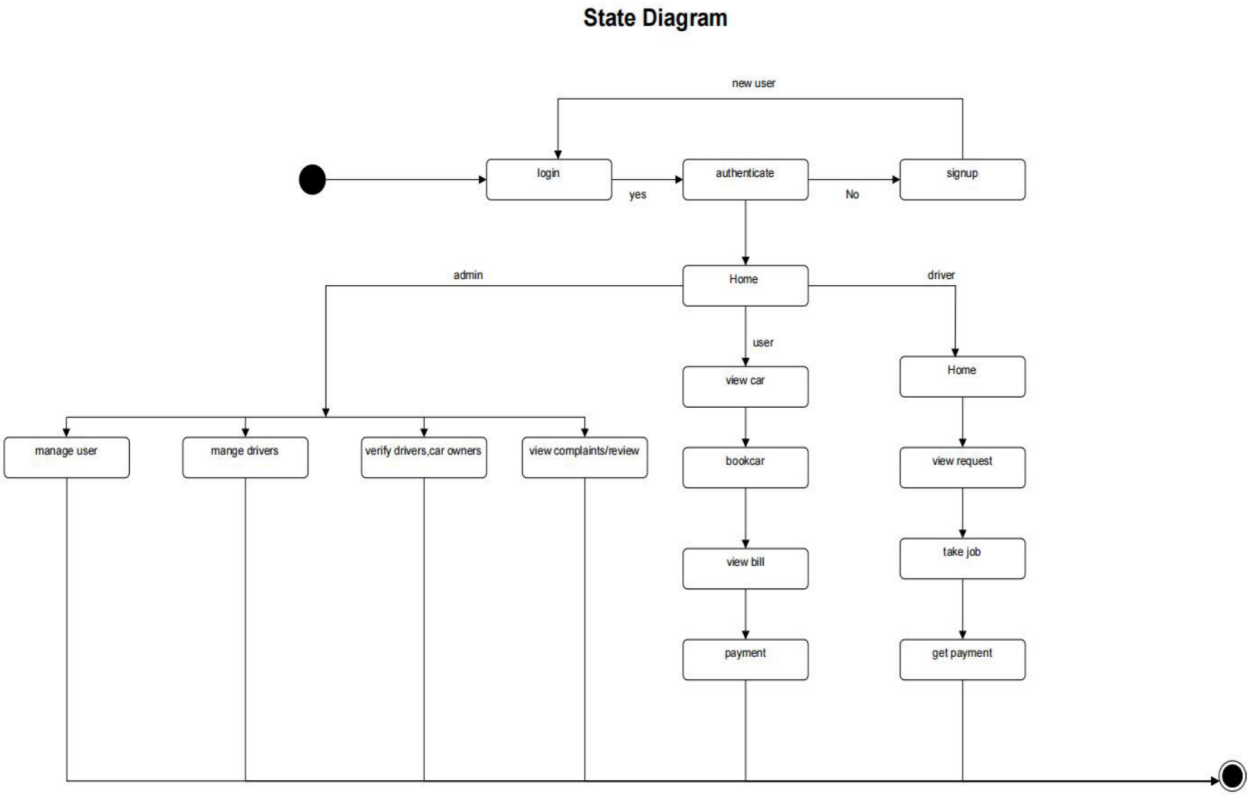
Uses of sequence diagrams –

- The complicated functions, logic and the procedure behind every process is represented using sequence diagram.
- Use case diagrams details are explained I the sequential diagram.
- Sequence diagram helps to understand the detailed functionality of the system.
- It helps to visualize how the messages and controls flow through the system.



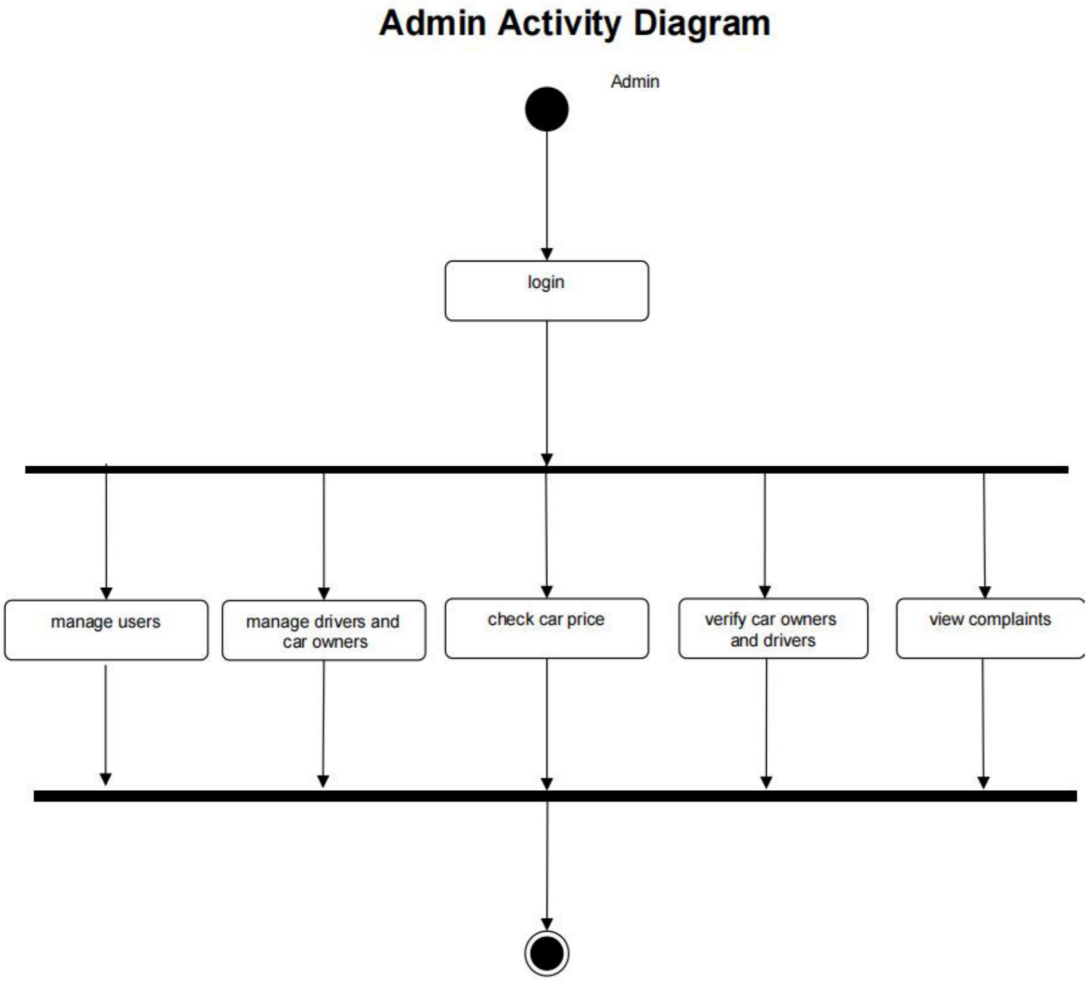
4.2.3 STATE CHART DIAGRAM

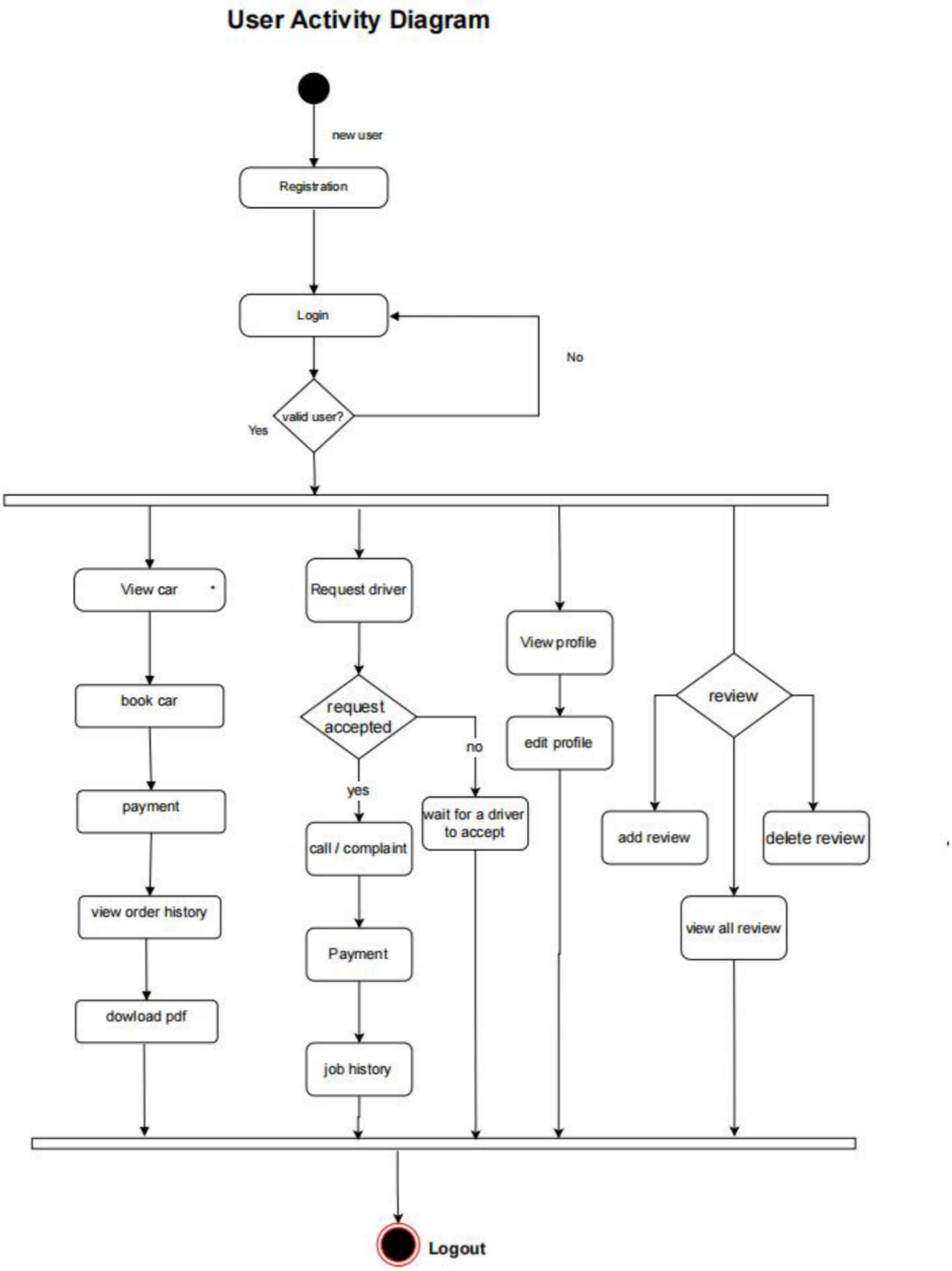
The behavior of the system is represented using the state chart diagrams. It is a kind of UML diagram which represent a class, subsystem, package, or even the behavior of an entire system can be shown using the state chart diagram. State cart diagram is also referred to state transition diagram. State chart diagrams help us to examine the interactions take place between external entities and a system. State chart diagram is used for representation of the event-based systems. An object's state may be controlled with the help of an event. State chart diagrams are used to show the various states of an entity inside an application system.



4.2.4 ACTIVITY DIAGRAM

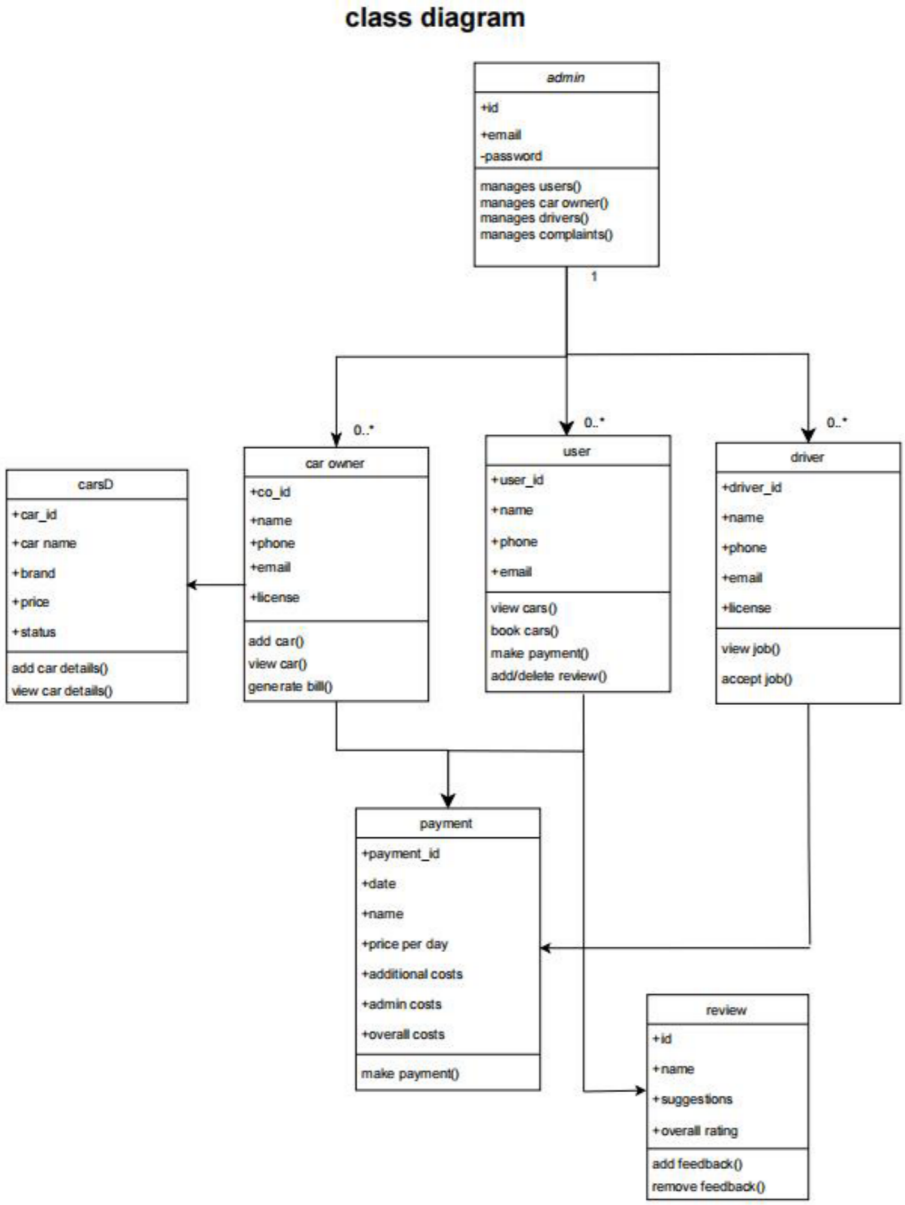
Activity diagrams depict how activities at various levels of abstraction are linked to provide a service. When events in a single use case are connected to one another and activities may overlap and require coordination, an event is usually done by a few operations. It may also be used to mimic the coordination of a collection of use cases used to describe business processes. Activity diagram help us to examine the flow of multiple activities in the system.




User Activity Diagram
new user
Registration
Login
No
valid user?
Yes
View car
book car
payment
view order history
dowload pdf
Request driver
request accepted
no
yes
wait for a driver to accept
call / complaint
Payment
job history
View profile
edit profile
review
add review
delete review
view all review
Logout

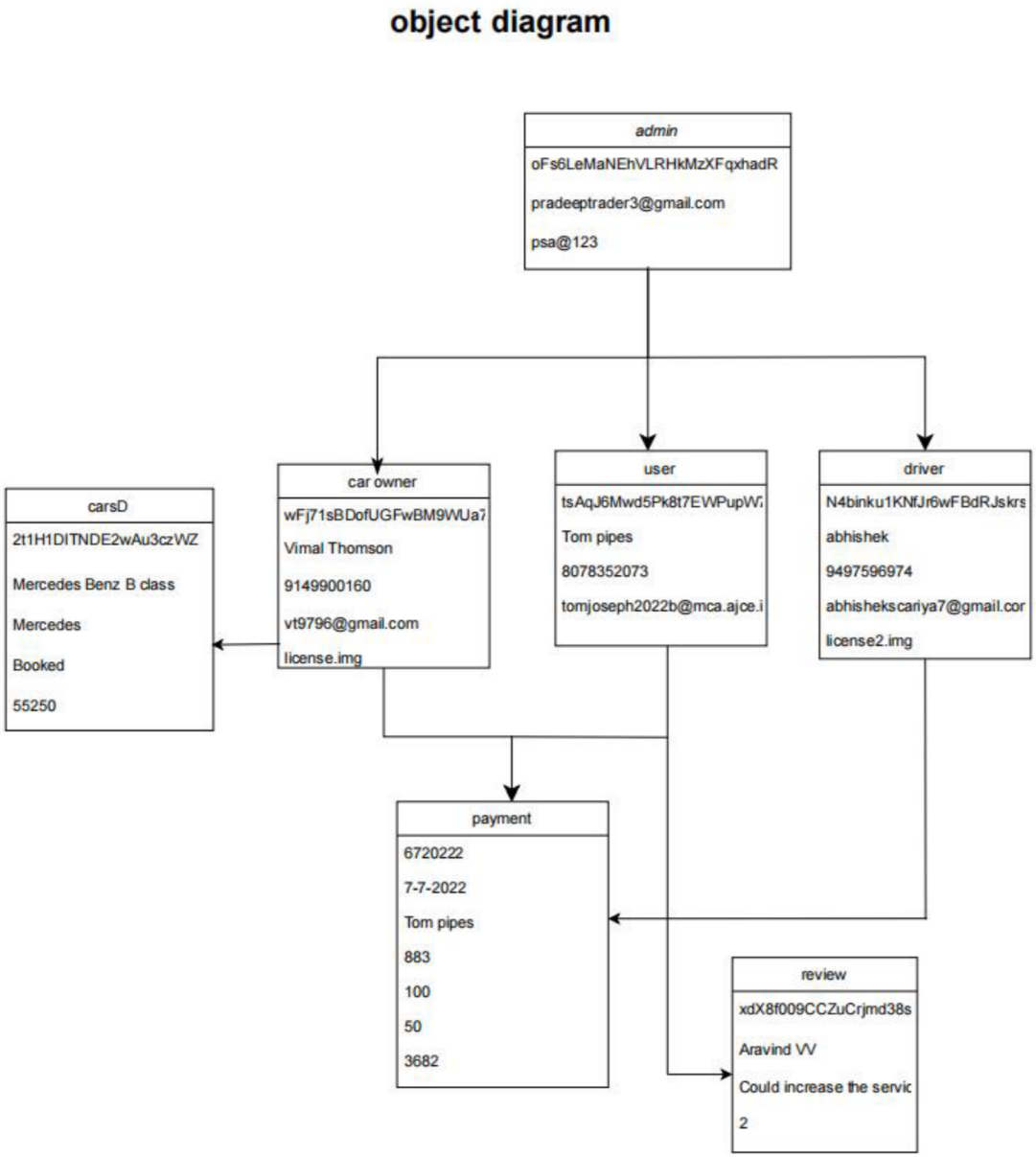
4.2.5 CLASS DIAGRAM

A class diagram is a type of static diagram. The static nature or view of the system is represented using class diagrams. The executable code for software programs is created with the help of class diagrams. Class diagrams can be also utilized for visualizing, explaining many elements of the system as well as for documentation purposes. All the features and the restrictions that are imposed on a class is depicted using class diagrams. Class diagrams are mostly used for designing object-oriented systems, because it is the only UML diagram which can be linked to object-oriented languages. A class diagram depicts a group of classes, interfaces, associations, collaborations, and limits. Class diagram is also known as Structural diagram.



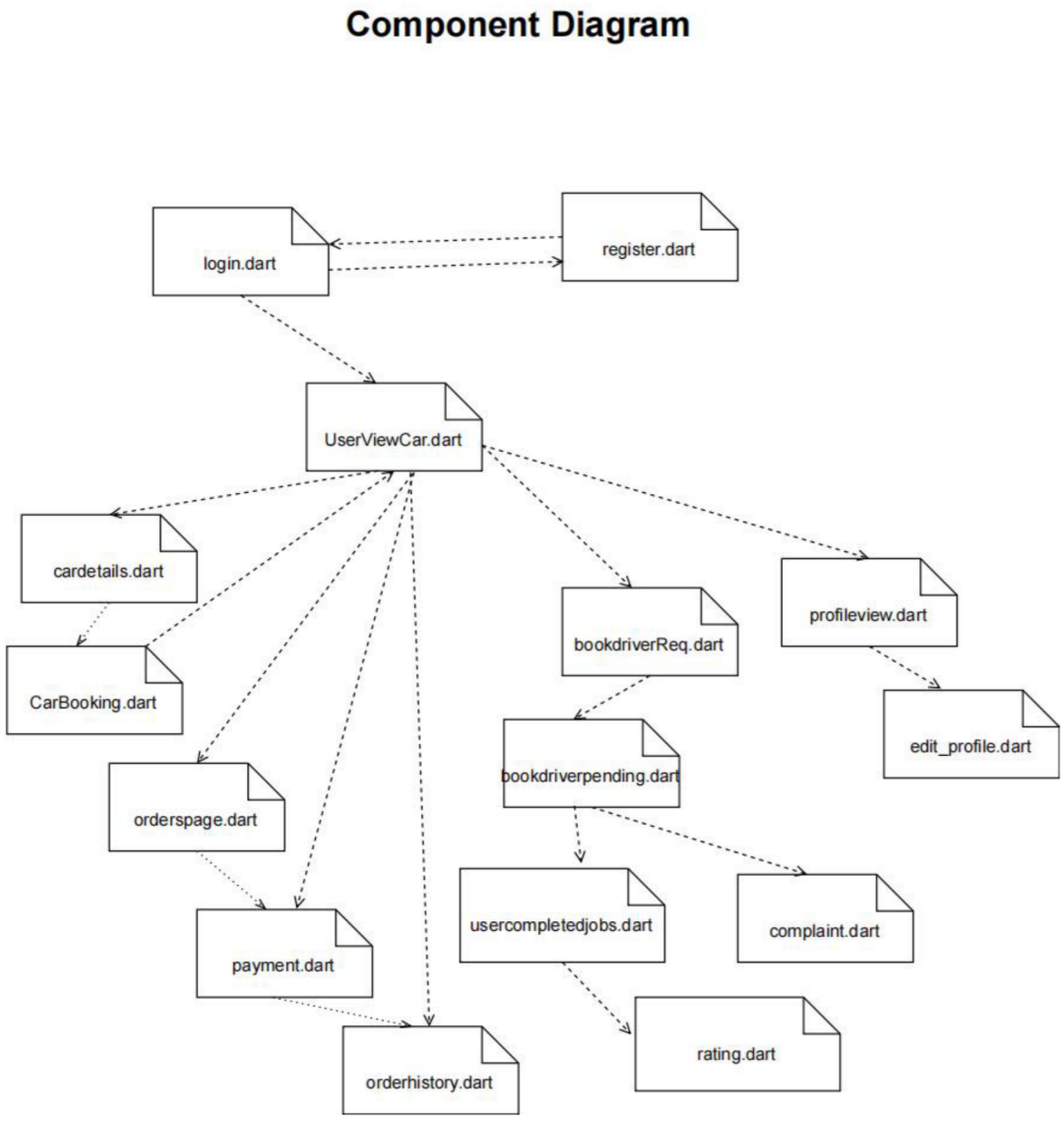
4.2.6 OBJECT DIAGRAM

These are kind of diagrams which can be derived from the class diagram. Object diagrams completely depends on the class diagrams since they are the instances of the classes. The basic concepts are methodologies are almost same for class diagram and object diagram. Object diagrams also reflect the static view of a system; however, this static view is a snapshot of the system at a certain point in time. Object diagrams are frequently used to represent a set of items and their relationships.



4.2.7 COMPONENT DIAGRAM

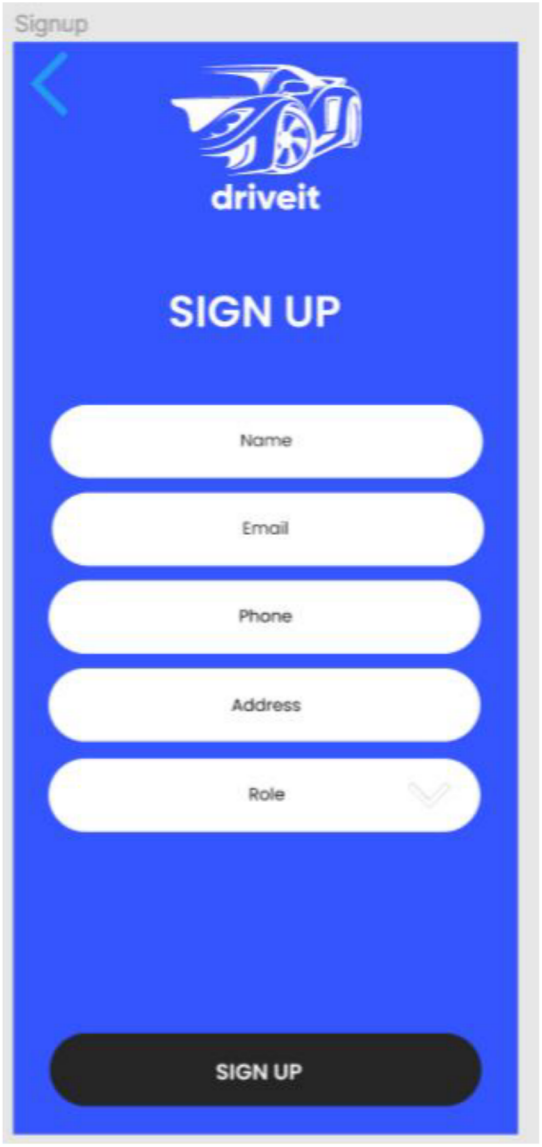
Component diagrams differ in their appearance and function. Component diagrams are being used to depict the physical components of a system. Types of applications, libraries, files, documents, and so on are all physically existent in a node. Component diagrams describe the organization and relationships between the elements of a system. Systems which can be performed are also designed utilizing these designs.



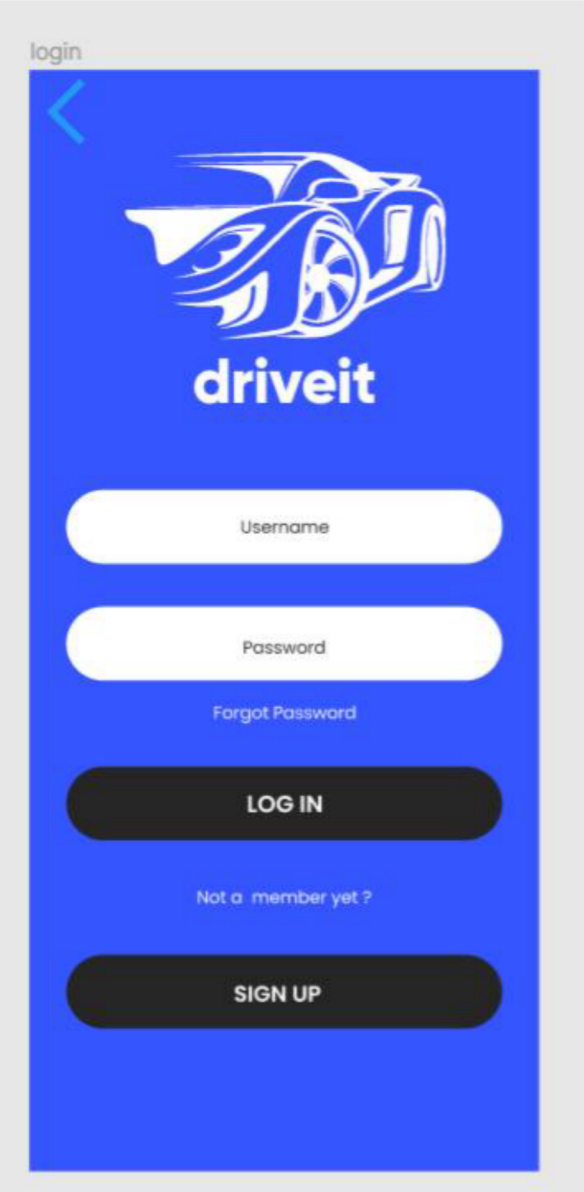
4.5 USER INTERFACE DESIGN

4.5.1-INPUT DESIGN

User Registration

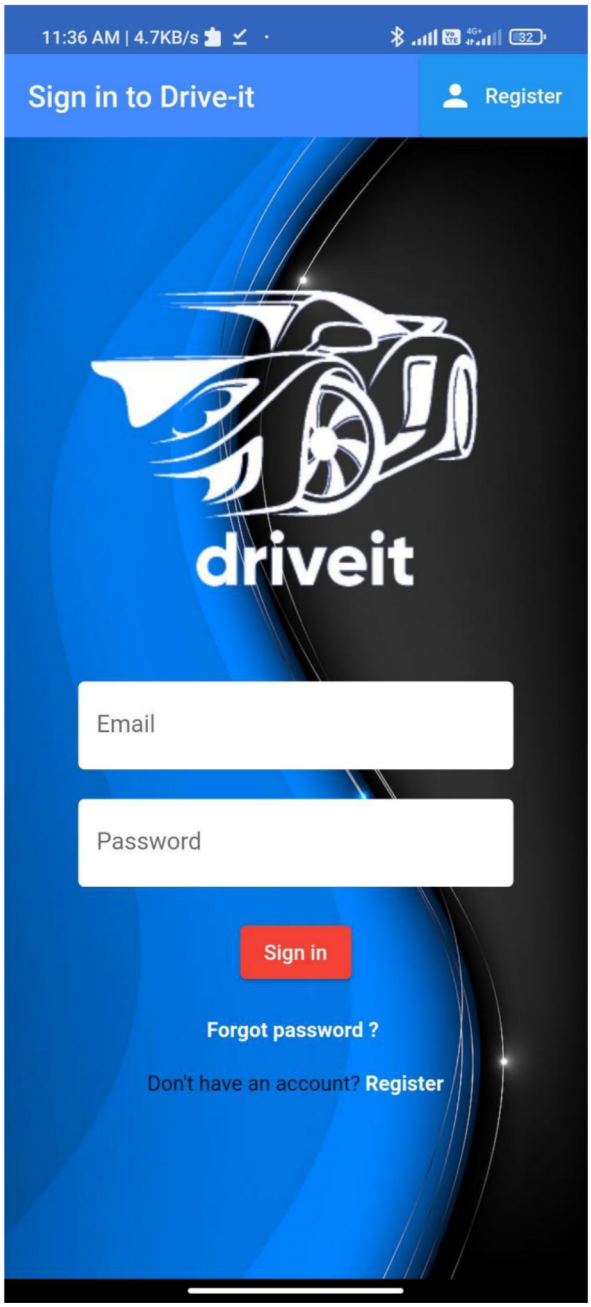


User Login

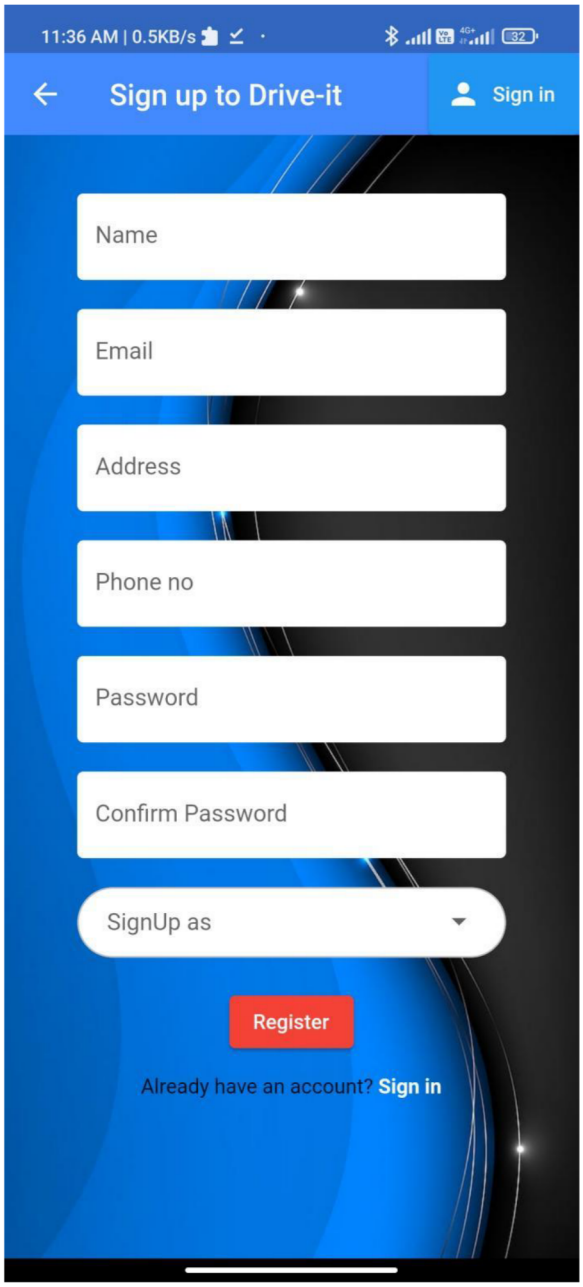

login
driveit
Username
Password
Forgot Password
LOG IN
Not a member yet ?
SIGN UP

4.5.2 OUTPUT DESIGN

User Login


11:36 AM | 4.7KB/s
Sign in to Drive-it
Register
driveit
Email
Password
Sign in
Forgot password ?
Don't have an account? Register

User Registration



4.6. DATABASE DESIGN

A database is a well-organized system that stores data and let users to access it quickly and efficiently. Every database's aim is to protect its data, which must be secured.

Database design process include two stages. In the first phase, user must be identified, and a database is built to suit these needs as directly as possible. This is known as Information Level Design, and it is done independently of any DBMS.

In the second stage, the design for the DBMS is used to develop the system and have to transfer from information level to a design. The Physical Level Design stage is when the DBMS features that will be employed are addressed. A database design operates in tandem with a system design. The database's data structure attempts to accomplish the two key goals outlined below.

- Integrity of the stored data
- Data independence in the data base schema

4.6.1 NoSQL Database

Instead of the columns and rows that relational databases employ to store data, NoSQL database technology stores data in JSON documents. To be precise, NoSQL does not mean "no SQL," but rather "not only SQL." This indicates that a NoSQL JSON database can essentially "use no SQL" to store and retrieve data. Or, for the best of both worlds, you can combine the adaptability of JSON with the strength of SQL. Because of this, NoSQL databases are designed to be adaptable, scalable, and quick to meet the data management needs of contemporary enterprises.

NoSQL database features

Each NoSQL database has special characteristics of its own.Many NoSQL databases, at a high level, include the following characteristics:

- Adaptable schemas
- Horizontal scaling
- Developers' ease of usage
- Quick queries as a result of the data model

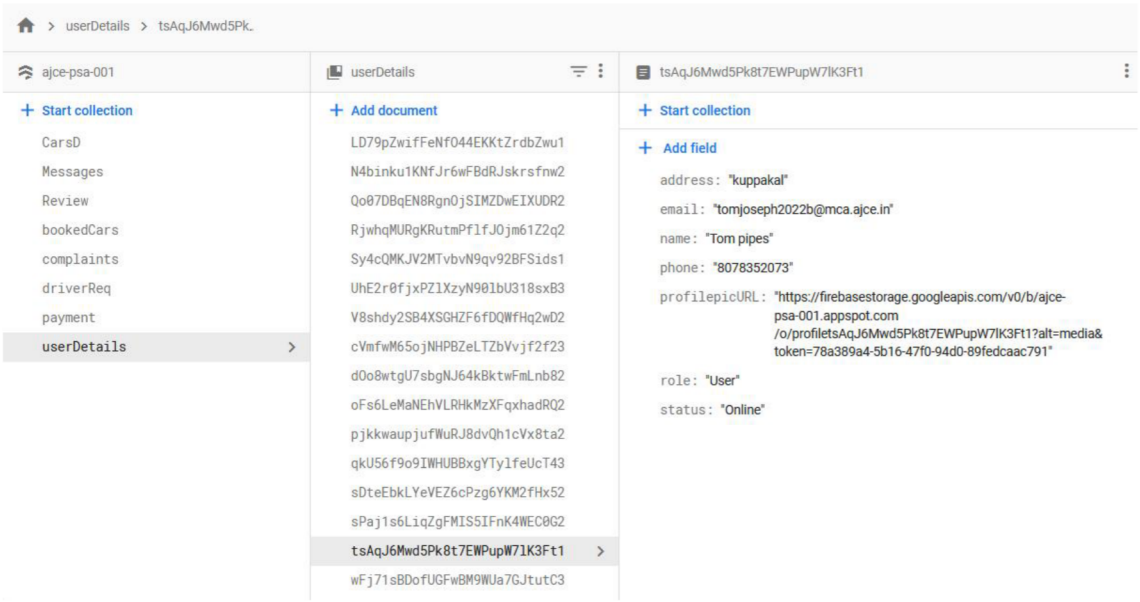
TABLE DESIGN

Collection 1

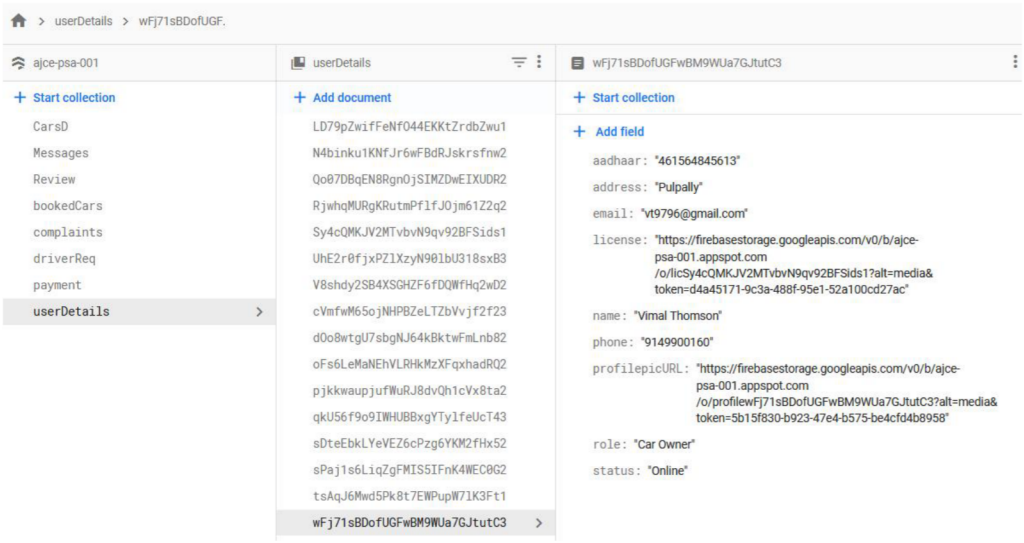
Collection name: userDetails

S. No	Fields	Data type	Description
1	name	String	Name of the user
2	email	String	Email
3	address	String	Address of the user
4	phone	String	Phone number of the user
5	profilepicURL	String	Profile image of the user
6	role	String	The type/role of user
7	status	String	User's Status
8	rating	Number	Rating of the driver
9	aadhaar	String	Aadhaar uploaded by driver/car owner
10	license	String	License uploaded by driver/car owner

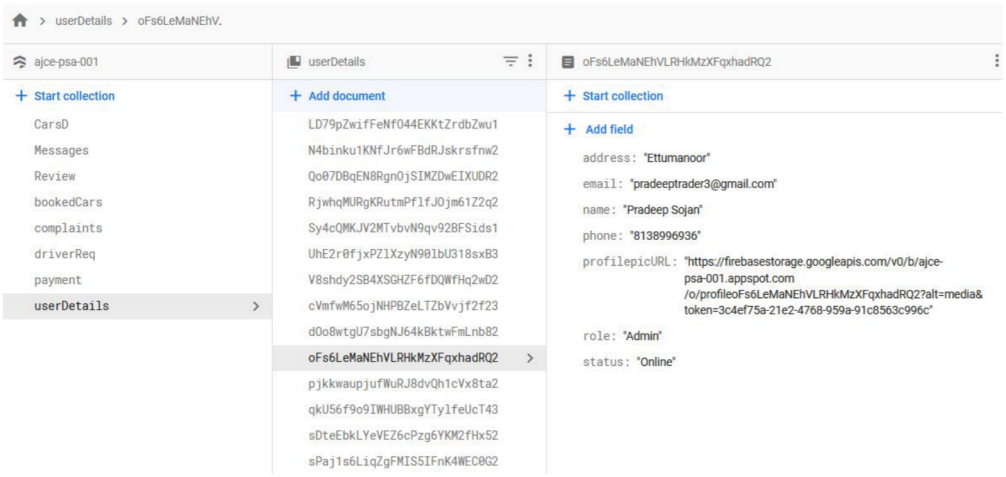
user



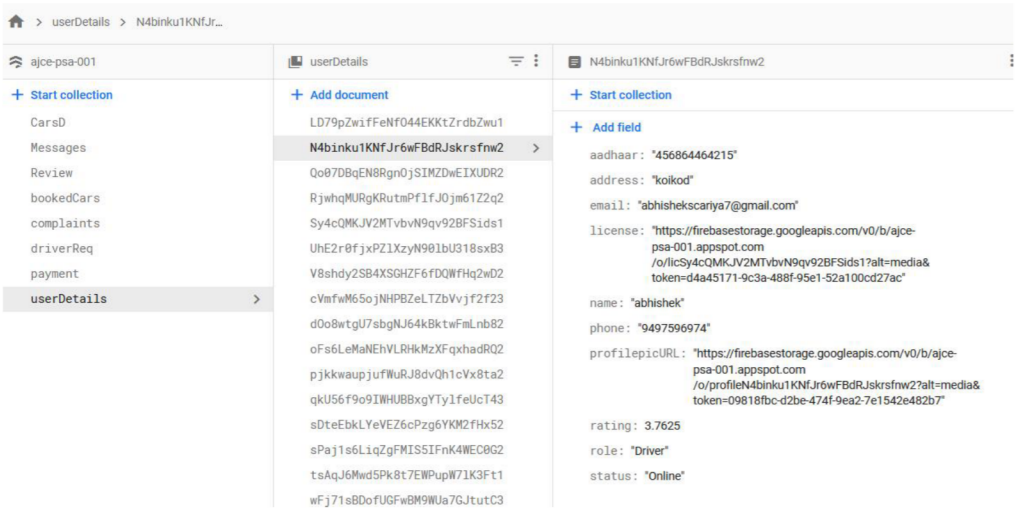
Car owner



Admin



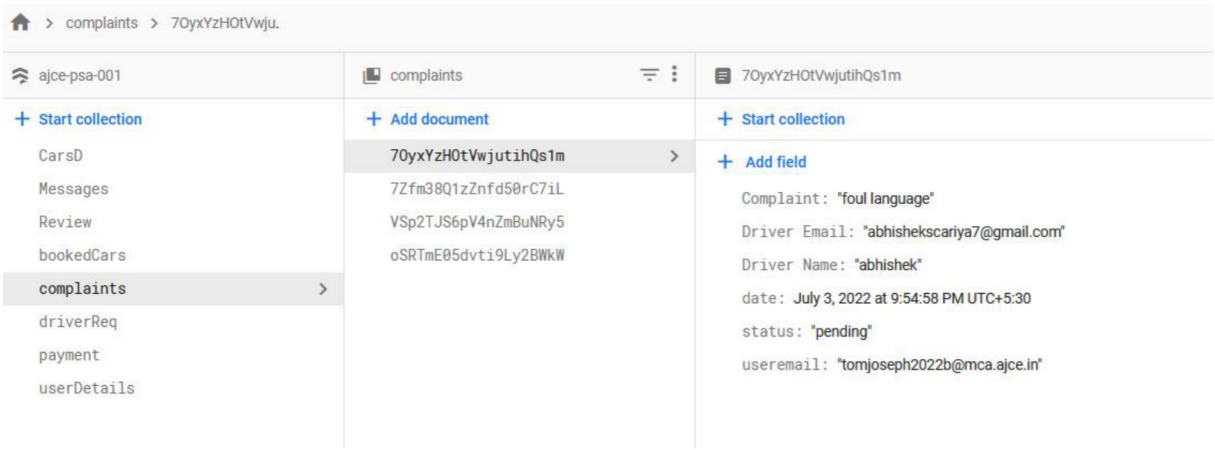
Driver



Collection 2

Collection name: complaints

S.No	Fields	Data type	Description
1	Complaint	String	Complaints written by the users
2	useremail	String	Name of the user who has given the complaint
3	Driver Name	String	Name of the driver who the user has complaint about.
4	Driver Email	String	Email of the said driver
5	status	String	Status of the complaint
6	date	timestamp	Time the complaint was made.

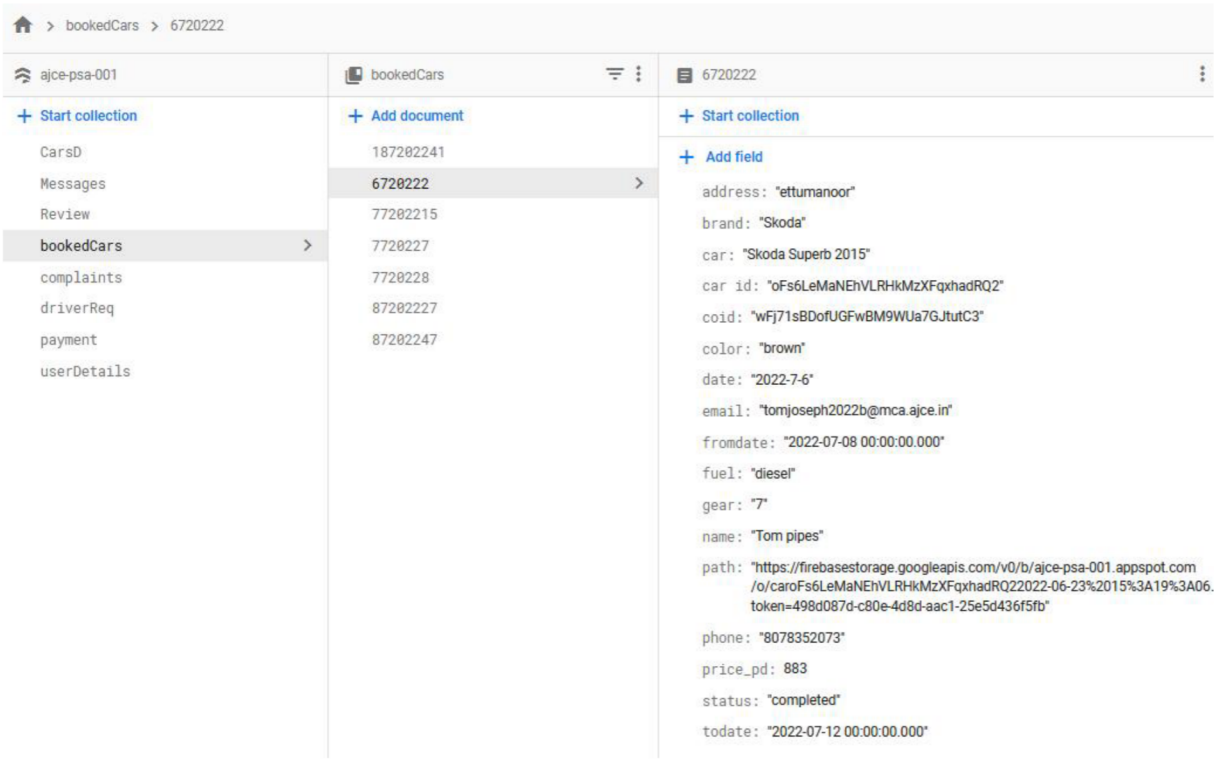


Collection 3

Collection name : bookedCars

S.No	Fields	Data type	Description
1	car	String	Name of the booked car
2	car id	String	Id of the booked car
3	coid	String	Id of the carowner whose car is booked
4	color	String	Color of the car
5	brand	String	Brand of the booked car
6	fuel	String	Fuel type of the car
7	gear	String	No of gears in the car

8	path	String	Image of the booked car
9	name	String	Name of the user who booked the car
10	email	String	Email of the user who booked the car
11	phone	String	Phone number of the user
12	price_pd	Number	Price of the car per day
13	fromdate	String	Date from which the car is booked
14	todate	String	Date to which the car is booked
15	date	String	Date of the booking
16	address	String	Address of the user
17	status	String	Status of the booking

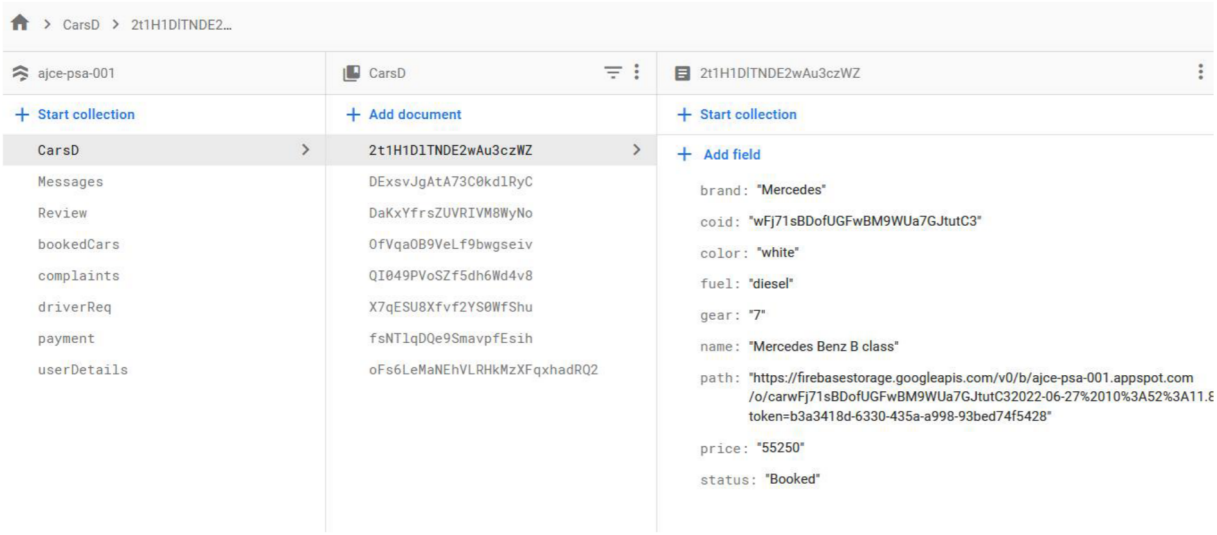


Collection 4

Collection name : CarsD

S.No	Fields	Data type	description
1	name	String	Name of the car
2	brand	String	Brand of the car
3	color	String	Color of the car
4	coid	String	Id of the car owner
5	fuel	String	Fuel type of the car

6	gear	String	No of gears in car
7	path	String	Image of the car
8	price	String	Monthly price of the car
9	status	String	Status of the car

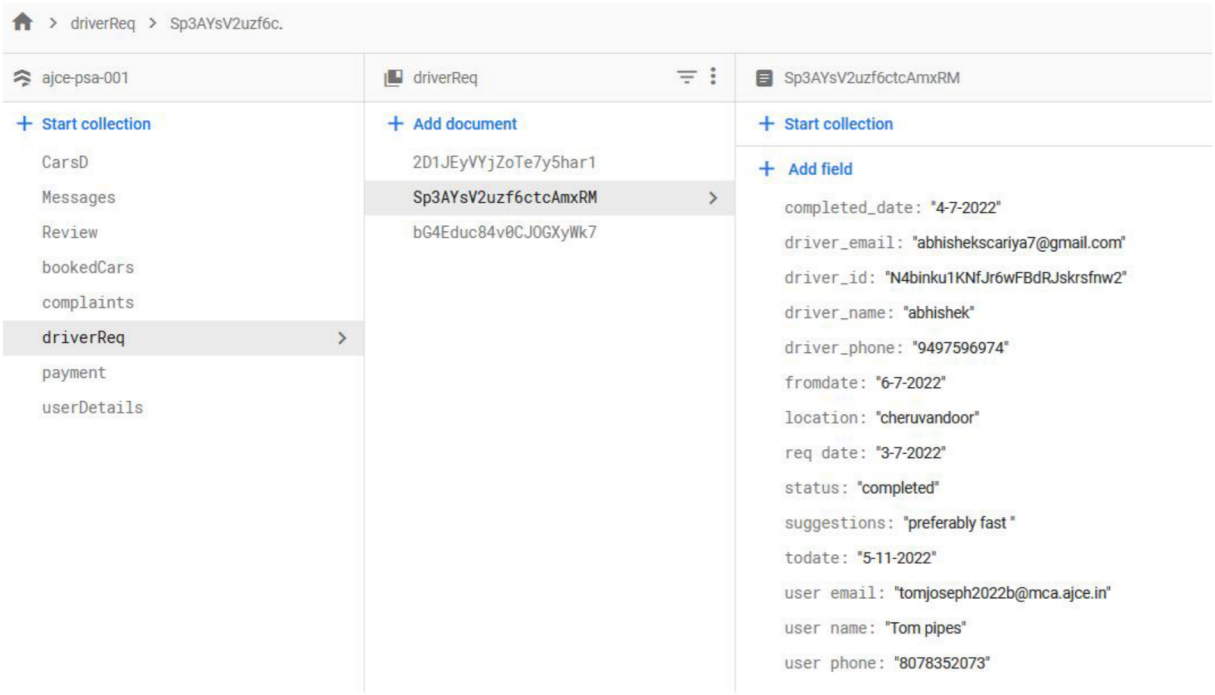


Collection 5

Collection name : driverReq

S.No	Fields	Data type	Description
1	user email	String	Email of the user
2	user name	String	Name of the user who requested for driver
3	user phone	String	Phone no of the user
4	suggestions	String	Suggestions of the user
5	location	String	Address of the user
6	fromdate	String	Date from which driver is required
7	todate	String	Date to which driver is required
8	Req date	String	Date at which request is made
9	driver_name	String	Name of the driver who took the job
10	driver_id	String	Id of the driver
11	driver_email	String	Email address of the driver
12	driver_phone	String	Phone no of the driver

13	completed_date	String	Date at which request is completed
14	status	String	Status of the request

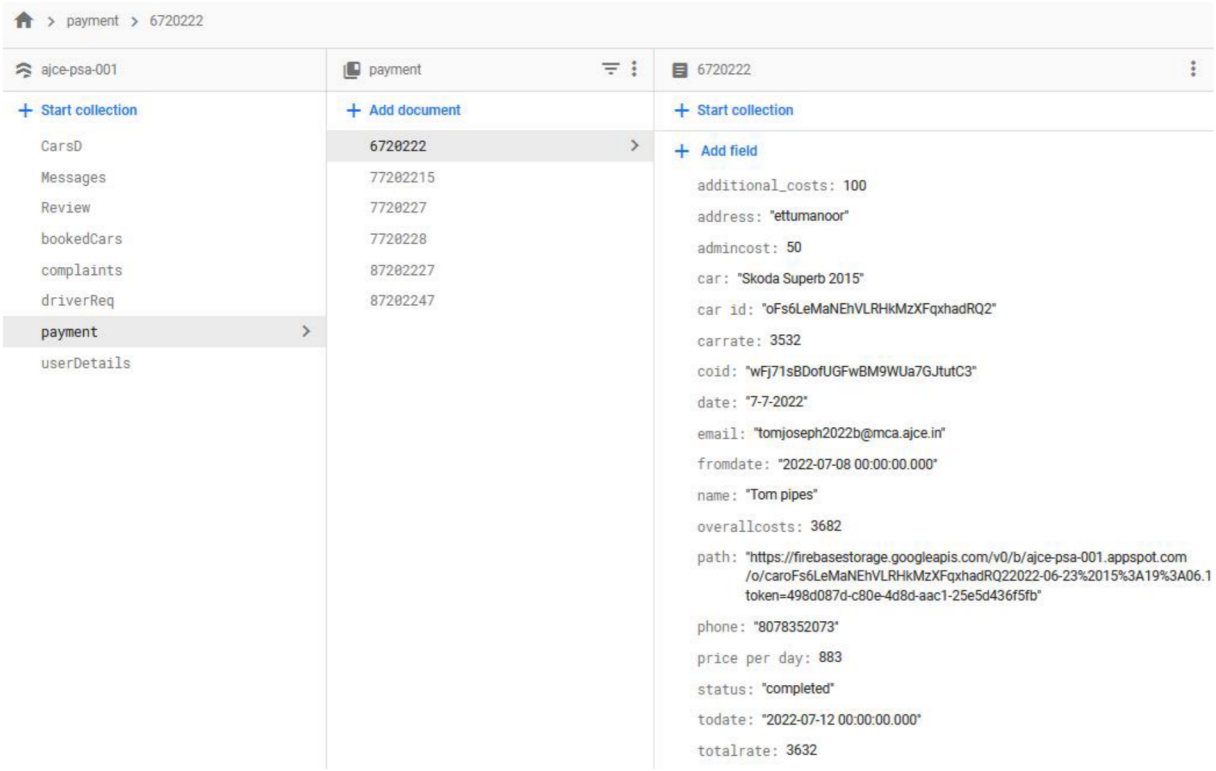


Collection 6

Collection name : payment

S.No	Fields	Data type	Description
1	email	String	Email of the user
2	name	String	Name of the user who requested for driver
3	phone	String	Phone no of the user
4	car	String	Name of the car
5	address	String	Address of the user
6	fromdate	String	Date from which car is booked
7	todate	String	Date to which car is booked
8	date	String	Date at which car is booked
9	price per day	number	Price of the car per day
10	car_id	String	Id of the car
11	coid	String	Id of the car owner
12	carrate	number	Rate of the car depending on the no of days car is booked
13	additional_costs	number	Additional costs

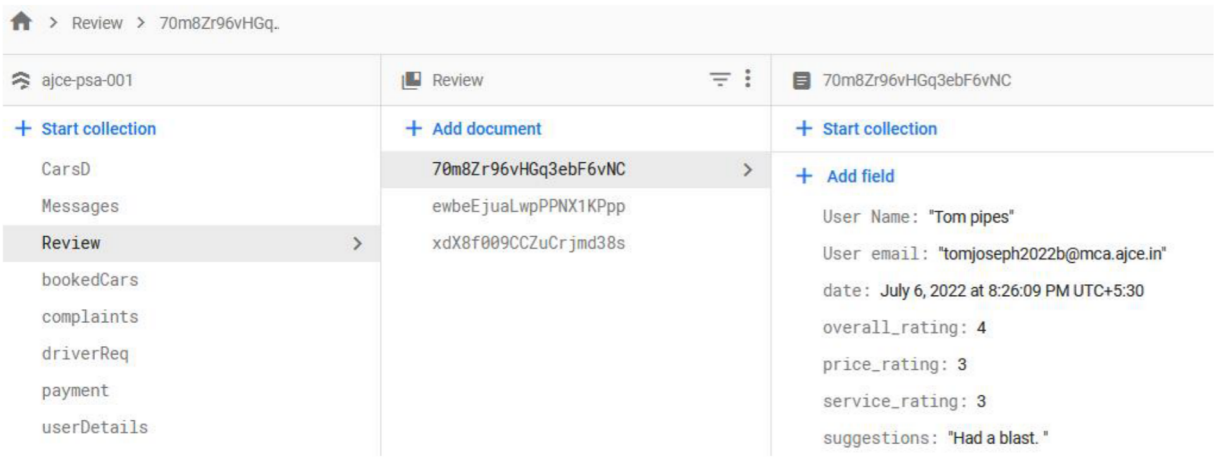
14	totalrate	number	Total rate of car
15	admincost	number	Set percentage that is given to the admin
16	overallcosts	number	Overall cost of the booking
17	path	String	Image of the car
18	status	String	Status of the payment



Collection 7

Collection name : Review

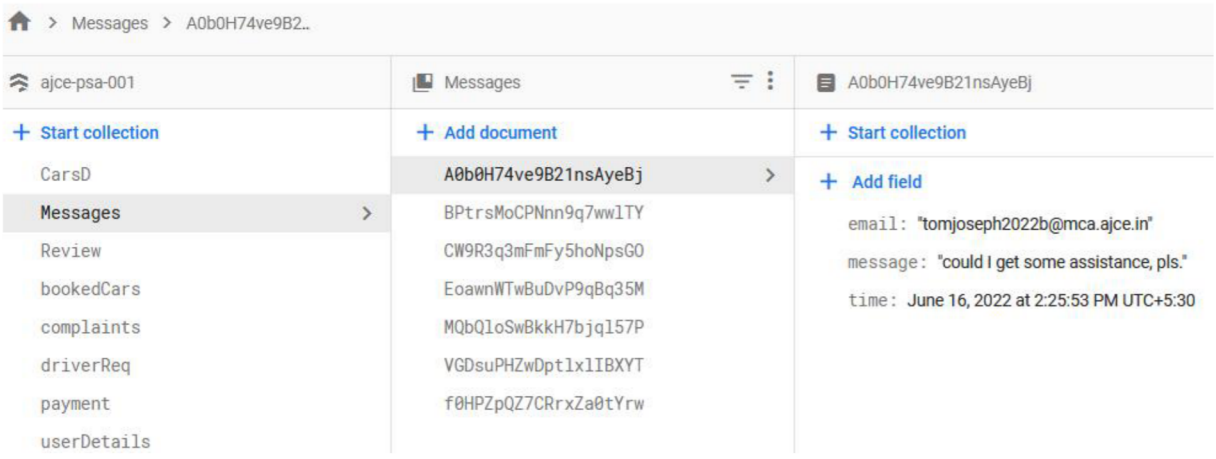
S.No	Fields	Data type	Description
1	User Name	String	Name of the user
2	User email	String	Email of the user
3	suggestions	String	Suggestions of the user
4	date	timestamp	Date of the review
5	price_rating	number	Price rating of the user
6	service_rating	number	Service rating of the user
7	overall_rating	number	Overall rating of the user



Collection 8

Collection name : messages

S.No	Fields	Data type	Description
1	message	String	Messages written by the user
2	email	String	Email of the user
3	time	timestamp	Time of the message



CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

The supervised deployment of software to assess whether or not it behaves as expected is known as software testing. The terms software testing, verification, and validation are occasionally used interchangeably. The examination or testing of objects, including software, for conformance and compatibility with an associated standard is known as validation. Other methods of verification besides software testing include reviews, analysis, exams, and walkthroughs. Validation is the process of ensuring that what is specified adheres to what the user wanted.

Static or dynamic analysis is another process that is commonly connected with software testing. Static analysis examines software source code for flaws and metrics without running it. Dynamic analysis investigates the behaviors of software while it is in use and provides information such as development traces, timing metrics, and test suite statistics..

Testing is a series of tasks that can be planned ahead of time and executed methodically. Module testing is followed by the integration of the entire computer-based system. Nothing is complete till testing is completed, as testing objectives are important to the system's performance. There are numerous criteria that could serve as testing objectives. These are their names:

The process of running software to find faults is known as testing.

- A good test case is one that has a high chance of discovering an unknown error.
- A successful test is one that discovers an unknown error.

If the testing is done appropriately in accordance with the above aims, it will expose flaws in the program. Furthermore, testing shows that the software function appears to be operational, and that the performance condition appears to have been reached.

There are three methods for testing a program.

- For accuracy.
- To improve implementation efficiency.
- For computational difficulty.

A correctness test is used to check that a software does exactly what it is supposed to do. This is significantly more difficult than it appears, particularly for large software.

5.2 TEST PLAN

A test plan is a set of tasks that will be followed in order to complete various testing procedures. The Test Plan acts as a blueprint for the action that will be taken. Software engineers are responsible for the creation of computer programs, documentation, and data structures. The software developers are always responsible for analyzing the individual components of the programs to ensure that they perform the job for which they were designed. An independent test group (ITG) exists to eliminate the inherent problems involved with enabling the builder to test the finished product. Measurable testing objectives should be provided. As a result, the test plan should include the mean time to failure, the cost of locating and correcting flaws, the residual defect density or frequency of occurrence, and the number of test work hours for each regression test.

The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing
- Output Testing

5.2.1 Unit Testing

Unit testing is performed on the lowest unit of software design, the software component or module. Using the component level design description as a guide, important control paths are examined within the module's border. The relative difficulty and breadth of unit testing are determined. Unit testing is white box in nature, and multiple components can be tested at the same time. The modular interface is tested to ensure that data flows effectively into and out of the software unit under test. The local data structure is reviewed to ensure that data saved temporarily preserves its integrity across all phases of an algorithm's execution. Boundary conditions are checked to guarantee that all statements in a module have been performed at least once. Finally, all error handling paths are tested.

Data flow through a module interface must be tested before proceeding with any additional tests. If data does not enter and exit properly, all other tests are rendered ineffective. It is necessary to do selective testing of execution pathways during the unit test. When an issue happens, it should be expected, and error handling methods should be put in place to reroute or cleanly cease operations. The unit testing process concludes with boundary testing. Software typically fails when it reaches its boundaries.

Unit testing was carried out in the Sell-Soft System by treating each module as a separate entity and testing each one with a diverse set of test inputs. Some flaws in the modules' underlying logic were uncovered and corrected. Each module is tested and run independently after coding. All unnecessary code was removed, and all modules were checked to ensure that they worked well and gave the desired results.

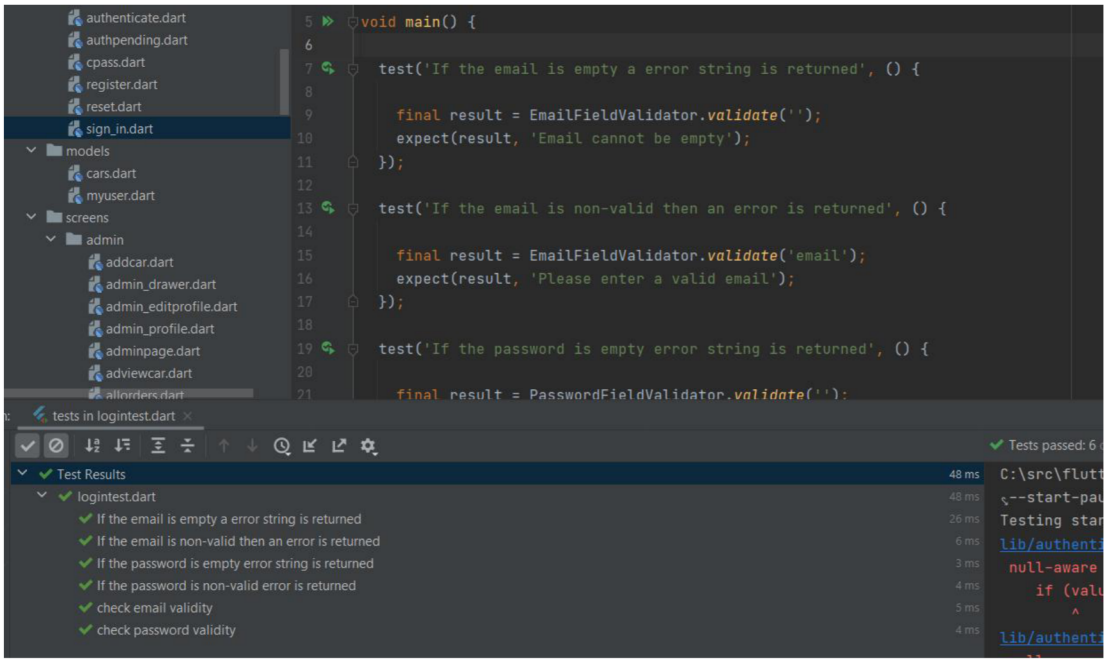
Test Case

Project Name: Drive_it Car rental					
login Test Case					
Test Case ID: Test1			**Test Designed By: Pradeep Sojan**		
Test Priority (Low/Medium/High): High			**Test Designed Date:** 19-07-2022		
Module Name: User			**Test Executed By : Ankitha philip**		
Test Title : Register the user with email and password			**Test Execution Date:** 20-07-2022		
Description: Test the login Page					
Pre-Condition : User has valid email id and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/Fail)**
1	Navigation to login Page		Login Page should be displayed	Login page is displayed	Pass

2	Provide Valid Email Id	Email: pradeeptrader3@gmail.com	User should be able to to login	User logins and went to home page	Pass
5 7	Provide Invalid Email Id or password Click on login	Email Id: pradeep@gmail.com Password: 123psa	User shouldn't login And redirect back to login page.	User didn't login. Invalid email,number etc can also cause error messages	Pass
Post-Condition: User is verified and can now login.					

```dart
import 'package:flutter_test/flutter_test.dart';
import 'package:p1/authenticate/sign_in.dart';

void main() {

  test('If the email is empty a error string is returned', () {

    final result = EmailFieldValidator.validate('');
    expect(result, 'Email cannot be empty');
  });

  test('If the email is non-valid then an error is returned', () {

    final result = EmailFieldValidator.validate('email');
    expect(result, 'Please enter a valid email');
  });

  test('If the password is empty error string is returned', () {

    final result = PasswordFieldValidator.validate('');
    expect(result, 'Password cannot be empty');
  });
```

```
  test('If the password is non-valid error is returned', () {

    final result = PasswordFieldValidator.validate('pass');
    expect(result, 'please enter valid password min. 6 character');
  });
  test('check email validity', () {

    final result = EmailFieldValidator.validate('pradeeptrader3@gmail.com');
    expect(result, null);
  });
  test('check password validity', () {

    final result = PasswordFieldValidator.validate('psa@123');
    expect(result, null);
  });
}
```



5.2.2 Integration Testing

Integration testing is a rigorous approach to establishing the program structure while simultaneously checking for interface difficulties. The goal is to use unit-tested components and create a design-governed program structure. The entire program is thoroughly tested. Correction is tough since the program's vast reach makes distinguishing causes difficult. Once these defects are fixed, new one's surface, and the process appears to continue endlessly. Following system unit testing, all modules were merged to check for interface incompatibilities. Furthermore, differences in program designs were eliminated, resulting in a single program structure.

5.2.3 Validation Testing or System Testing

This is the final phase of testing. This includes putting the entire system through its paces, including all forms, code, modules, and class modules. This form of testing is also known as Black Box testing or System testing.

The Black Box testing method focuses on the functional requirements of the software. In other words, Black Box testing enables a software engineer to create sets of input conditions that fully exercise all of a program's functional requirements.

Erroneous or missing functions, interface flaws, data structure or external data access errors, performance errors, initialization failures, and termination faults are all examples of defects found during black box testing.

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

A project's implementation stage is when the theoretical design is transformed into a working system. The most crucial stage in developing a successful new system is gaining users' confidence that the new system will work, be effective, and accurate. Its primary goal is to provide user training and documentation. Conversion usually occurs at the same time as or after the user is informed. The process of changing a new improved system design into an operational one is referred to as implementation.

Currently, the user department is responsible for most of the workload, causing the most disturbance and having the greatest influence on the existing system. Unplanned or unregulated implementation may result in turmoil and confusion.

All activities undertaken to convert from the current system to the new system are referred to as implementation. The new system could be entirely new, replacing an existing human or automated system, or it could be a modification to an existing system. A dependable system that meets the needs of the organization requires proper implementation. The process of bringing the finished system into use is referred to as system implementation. This includes all activities required to move from the old to the new system. Only after comprehensive testing and confirmation that the system fulfils the specifications can it be deployed. The system personnel assess the system's feasibility. The more complicated the system being deployed, the more effort is required for system analysis and design to complete the three major aspects: education and training, system testing, and changeover.

The implementation state involves the following tasks:

- Planning is essential.
- System and constraint investigation.
- Methods for achieving the transition.

6.2 IMPLEMENTATION PROCEDURES

Software implementation refers to the final installation of the package in its actual surroundings, to the satisfaction of the intended purposes and the operation of the system. Many organizations will have someone commission the software development project who will not be operating it. People are initially skeptical of the software, but we must ensure that opposition does not develop, just as one must ensure that:

- The active user must be informed of the advantages of the new system
- Their trust in the software is growing.
- The user is given proper instruction so that he is comfortable using the application.

Before going to view the system, the user should be informed that the server software must be running on the server in order to view the findings. If the server object is not up and running on the server, the actual process will not take place.

6.2.1 User Training

The purpose of user training is to prepare the user for system testing and conversion. Individuals who will be engaging in the new system must be confident in their role in order to achieve the goal and gain the benefits of a computer-based system. As the system becomes more sophisticated, so does the necessity for training. User training teaches the user how to enter data, respond to error signals, query the database, and activate procedures that generate reports and do other critical tasks.

6.2.2 Training on the Application Software

The user will need to be trained on the new application software after receiving the necessary fundamental computer awareness training. This will give the essential concept of using the new system, such as the screen flow, screen design, kind of help on the screen, type of errors while entering data, the associated validation check at each entry, and methods to correct the date entered. The information necessary by the individual user/group to utilize the system or a section of the system should therefore be covered while providing program training on the application. This training may differ between user groups and at different organizational levels.

6.2.3 System Maintenance

Maintenance is the enigma of system development. When a software product is in the maintenance phase of its life cycle, it is performing valuable work. After an effective system has been created, it must be properly maintained. System upkeep is an essential part of the software development life cycle. System maintenance's goal is to make the system more adaptable to changes in the system environment. All in all, software maintenance requires far more than "Finding Mistakes.".

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

Transportation is essential and a necessity for human beings. As long as we exist, the need for transportation will never cease to exist. The current working system is very old fashioned and there is no accurate system for small rental establishments. Our car rental is a great solution for the troubles of owning and maintaining a car. The proposed system introduces facility for customer to book service online and view all information and is best suited for small rental establishment.

7.2 FUTURE SCOPE

The system can be updated in the future according to the user's needs. New features could be added the system to enhance user experience. First aid services, mechanic services and various other services can be added. All automobile rentals with innovative strategies and cutting-edge technologies to differentiate themselves from the competition may look forward to a bright future.Software for managing automobile rentals is available that may be used to streamline operations and improve customer service.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, "*System Analysis and Design*", 2009.
- Madhuram , Ashu Kumar , Pandyamanian, "*Cross Platform Development using Flutter*", 2019.
- PankajJalote, "So*ftware engineering*: a precise approach", 2006.
- Aakanksha Tashildar, Nisha Shah, Rushabh Gala, Trishul Giri, Pranali Chavhan, "APPLICATION DEVELOPMENT USING FLUTTER", 2020.
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

- https://medium.com/flutter
- https://www.youtube.com/c/flutterdev
- https://flutter.dev/showcase
- https://pub.dev/

CHAPTER 9

APPENDIX

9.1 Sample Code

Signin.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/gestures.dart';
import 'package:flutter/material.dart';
import 'package:fluttertoast/fluttertoast.dart';
import 'package:p1/authenticate/authenticate.dart';
import 'package:p1/authenticate/authpending.dart';
import 'package:p1/screens/admin/adviewcar.dart';
import 'package:p1/screens/cars_overview.dart';
import 'package:p1/screens/driverscreen/driverhome.dart';
import 'package:p1/screens/UserViewCar.dart';
import 'package:p1/services/auth.dart';
import 'package:p1/utils/loading.dart';
import 'package:p1/utils/utils.dart';
import '../screens/admin/topdeals.dart';
import '../screens/car_owner/car_owner_home.dart';
import 'register.dart';
import 'package:p1/authenticate/reset.dart';
class EmailFieldValidator {
  static String? validate(String value) {
    if (value!.length == 0) {
      return "Email cannot be empty";
    }
    if (!RegExp(
        r"^([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-\.])+\.([A-Za-z]{2,4})$")
        .hasMatch(value)) {
      return ("Please enter a valid email");
    }else {
      return null;
    }
  }
}

class PasswordFieldValidator {
  static String? validate(String value) {
    RegExp regex = new RegExp(r'^.{6,}$');
    if (value!.isEmpty) {
      return "Password cannot be empty";
    }
    if (!regex.hasMatch(value)) {
      return ("please enter valid password min. 6 character");
    }else {
      return null;
    }
  }
}

class SignIn extends StatefulWidget {
```

```
  // final Function toggleView;
  // SignIn({ required this.toggleView });

  @override
  _SignInState createState() => _SignInState();
}

class _SignInState extends State<SignIn> {

  final AuthService _auth = AuthService();
  final _formkey = GlobalKey<FormState>();
  bool loading = false;

  final fire = FirebaseFirestore.instance.collection('userDetails');
  String role = '';
  String status = '';
  String email = '';
  String password = '';
  String error = '';

  @override
  Widget build(BuildContext context) {
    return loading ? Loading() : Scaffold(
      backgroundColor: Colors.blueAccent,
      appBar: AppBar(
        backgroundColor: Colors.blueAccent,
        elevation: 0.0,
        title: Text('Sign in to Drive-it'),
        actions: <Widget>[
          ElevatedButton.icon(
            icon: Icon(Icons.person), label: Text('Register'),
            onPressed: () {
              //widget.toggleView();
              Navigator.of(context).push(new MaterialPageRoute(
                  builder: (BuildContext context) => Register()));
            },
          )
        ],
      ),
      body: SingleChildScrollView(
          child: Container(
              height:MediaQuery.of(context).size.height*0.897,
              width: MediaQuery.of(context).size.width,
              decoration: BoxDecoration(
                image: DecorationImage(
                  image: AssetImage("assets/images/wp.jpg"),
                  fit: BoxFit.cover,
                ),
              ),
              padding: EdgeInsets.symmetric(vertical: 20.0, horizontal: 50.0),
              child: Form(
                key: _formkey,
                autovalidateMode: AutovalidateMode.always,
                child: Column(
                  children: <Widget>[
```

```
SizedBox(height: 50),
Image.asset("assets/images/splash.png"),
SizedBox(height: 50.0),
TextFormField(
  decoration: textInputDecoration.copyWith(
      hintText: "Email"),
  // validator: (val) =>
  // val!.isEmpty
  //     ? 'Enter an Email'
  //     : null,
  validator:(value)=> EmailFieldValidator.validate(value!),
  onChanged: (val) {
    setState(() => email = val);
  },
),
SizedBox(height: 20.0),
TextFormField(
  decoration: textInputDecoration.copyWith(
      hintText: "Password"),
  obscureText: true,
  validator:(value)=> PasswordFieldValidator.validate(value!),
  onChanged: (val) {
    setState(() => password = val);
  },
),
SizedBox(height: 20.0),
ElevatedButton(
  style: ButtonStyle(
    backgroundColor: MaterialStateProperty.all(Colors.red),
  ),
  child: Text(
    'Sign in',
    style: TextStyle(color: Colors.white),
  ),
  onPressed: () async {

    if (_formkey.currentState!.validate()) {
      setState(() => loading = true);
      try {
        UserCredential userCredential = await FirebaseAuth.instance
            .signInWithEmailAndPassword(email: email, password: password);
        _checkRole();
      } on FirebaseAuthException catch (e) {
        setState(() {
          loading = false;
        });
        Fluttertoast.showToast(msg: e.toString());
        if (e.code == 'user-not-found') {
          print("No user found with this email");

          Fluttertoast.showToast(
            msg: "No user found with this email",
          );
        } else if (e.code == 'wrong-password') {
          print("You have entered the Wrong Password");
```

```
                    Fluttertoast.showToast(
                      msg: "You have entered the Wrong Password",
                    );
                  }
                }
              }
              else
                loading=false;
            },
          ),
          SizedBox(height: 20.0),
          Text.rich(
            TextSpan(
              children: [
                TextSpan(
                  text: "Forgot password ? ",
                  recognizer: TapGestureRecognizer()
                    ..onTap = () {
                      Navigator.push(context, MaterialPageRoute(
                          builder: (context) => ResetScreen()));
                    },
                  style: TextStyle(
                      fontWeight: FontWeight.bold, color: Colors.white),
                ),
              ]
            )
          ),
          SizedBox(height: 20.0),
          Text.rich(
            TextSpan(
              children: [
                TextSpan(text: "Don\'t have an account? "),
                TextSpan(
                  text: 'Register',
                  recognizer: TapGestureRecognizer()
                    ..onTap = () {
                      Navigator.push(context, MaterialPageRoute(
                          builder: (context) => Register()));
                    },
                  style: TextStyle(
                      fontWeight: FontWeight.bold, color: Colors.white),
                ),
              ]
            )
          ),
        ],
      ),
    )
  ),
 );
}
```

```
 void _checkRole() async {
  User? user = FirebaseAuth.instance.currentUser;
  final DocumentSnapshot snap = await FirebaseFirestore.instance.collection(
     'userDetails').doc(user?.uid).get();

  setState(() {
   role = snap['role'];
   status = snap['status'];
  });
  if (status == 'Online') {
   if (role == 'User') {
    Navigator.of(context).push(new MaterialPageRoute(
       builder: (BuildContext context) => UserViewCar()));
   } else if (role == 'Admin') {
    Navigator.of(context).push(new MaterialPageRoute(
       builder: (BuildContext context) => Adviewcar()));
   } else if (role == 'Driver') {
    Navigator.of(context).push(new MaterialPageRoute(
       builder: (BuildContext context) => DriverHome()));
   } else if (role == 'Car Owner') {
    Navigator.of(context).push(new MaterialPageRoute(
       builder: (BuildContext context) => CarOwnerHome()));
   }
   }else if (
     status=='online'
    ){
   if (role == 'User') {
    Navigator.of(context).push(new MaterialPageRoute(
       builder: (BuildContext context) => UserViewCar()));

   }  else if (role == 'Driver') {
    Navigator.of(context).pushAndRemoveUntil(new MaterialPageRoute(builder: (BuildContext context)
=> auth()), (Route<dynamic> route) => false);
    Fluttertoast.showToast(msg: "verification pending");

   } else if (role == 'Car Owner') {
    Navigator.of(context).pushAndRemoveUntil(new MaterialPageRoute(builder: (BuildContext context)
=> auth()), (Route<dynamic> route) => false);
    Fluttertoast.showToast(msg: "verification pending");
   }}
  else if (
  status=='pending'
  ){
   if (role == 'User') {
    Navigator.of(context).push(new MaterialPageRoute(
       builder: (BuildContext context) => UserViewCar()));

   }  else if (role == 'Driver') {
    Navigator.of(context).pushAndRemoveUntil(new MaterialPageRoute(builder: (BuildContext context)
=> authPending()), (Route<dynamic> route) => false);
    Fluttertoast.showToast(msg: "verification pending");

   } else if (role == 'Car Owner') {
    Navigator.of(context).pushAndRemoveUntil(new MaterialPageRoute(builder: (BuildContext context)
=> authPending()), (Route<dynamic> route) => false);
```

```
      Fluttertoast.showToast(msg: "verification pending");
    }}
   else{
    loading = false;
    Fluttertoast.showToast(
      msg: "You are not authorized to login, please contact admin",
    );
   }
  }
}
```

UserViewCar.dart

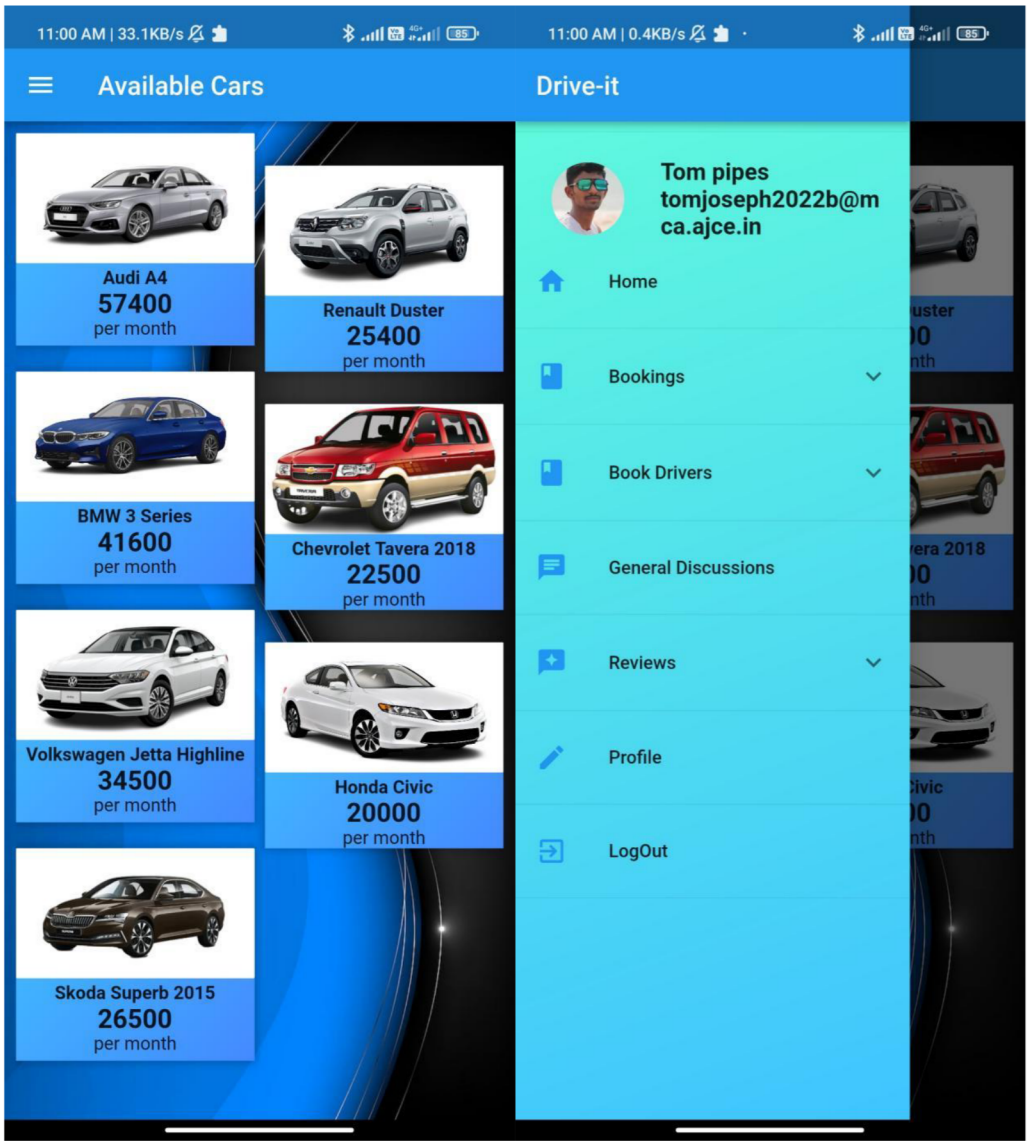
```
import 'package:animate_do/animate_do.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:animated_search_bar/animated_search_bar.dart';
import 'package:flutter/services.dart';
import 'package:p1/screens/car_detail.dart';
import 'package:p1/screens/car_owner/car_info.dart';
import 'package:p1/screens/drawer.dart';
import 'package:p1/utils/utils.dart';
import 'package:sliding_up_panel/sliding_up_panel.dart';


class UserViewCar extends StatefulWidget {
  const UserViewCar({Key? key}) : super(key: key);

  @override
  _UserViewCarState createState() => _UserViewCarState();
}

class _UserViewCarState extends State<UserViewCar> {


  List searchResult = [];
  void searchFromFirebase(String query) async {
    final result = await FirebaseFirestore.instance
        .collection('carsD')
        .where('name', arrayContains: query)
        .get();

    setState(() {
      searchResult = result.docs.map((e) => e.data()).toList();
    });
  }

  @override
  Widget build(BuildContext context) {
    BorderRadiusGeometry radius = BorderRadius.only(
      topLeft: Radius.circular(24.0),
      topRight: Radius.circular(24.0),
```

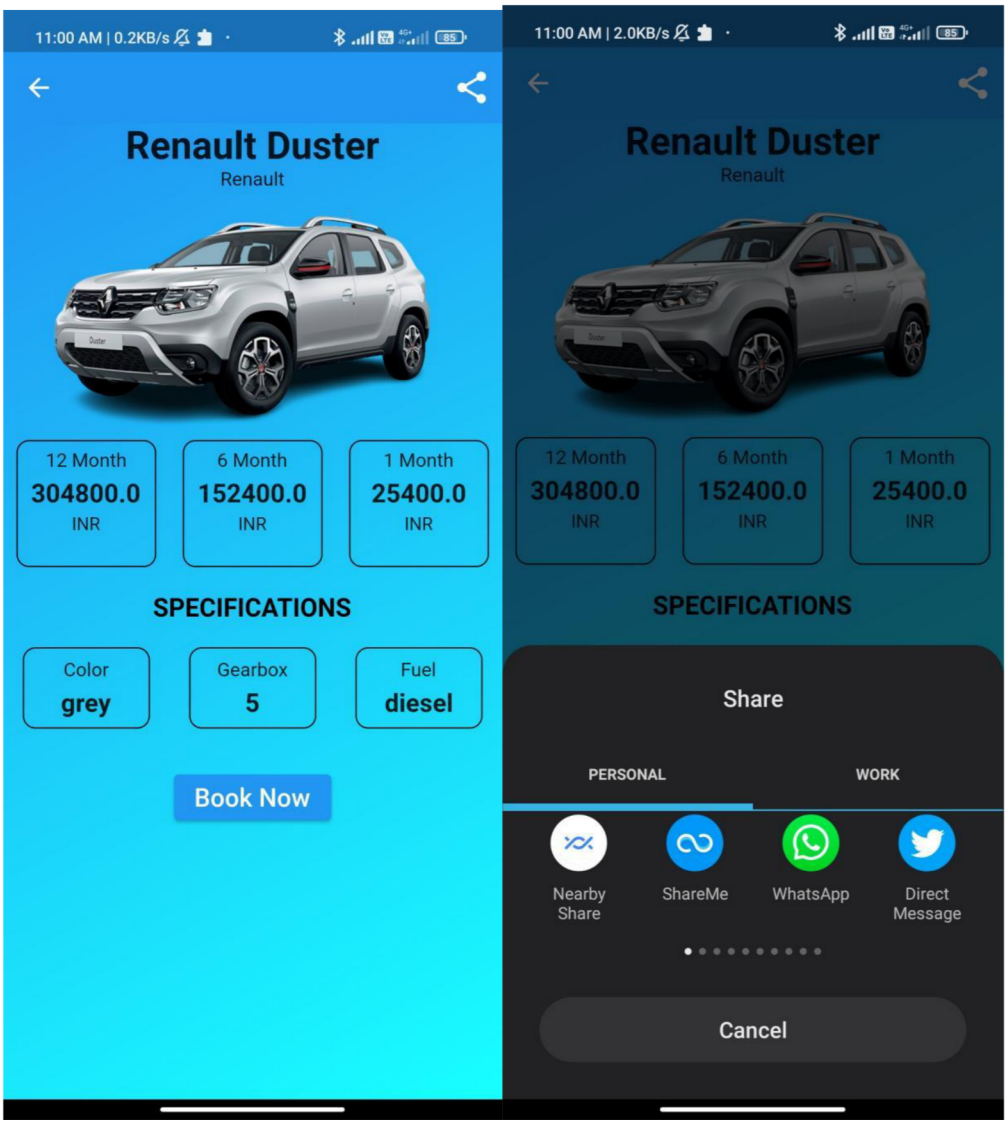
```
  );
  return new Scaffold(
   appBar: AppBar(
    title: Text("Available Cars"), ),

   body: Container(
    decoration: BoxDecoration(
     image: DecorationImage(
      image: AssetImage("assets/images/wp.jpg"),
      fit: BoxFit.cover,
     ),
    ),
    child: Padding(padding: const EdgeInsets.all(9.0),
     child: StreamBuilder(
      stream:  FirebaseFirestore.instance.collection('CarsD').where('status',isEqualTo:
'Available').snapshots(),
      builder: (context, AsyncSnapshot<QuerySnapshot> streamSnapshot) {
       if (streamSnapshot.hasData) {

        return GridView.builder(
         //shrinkWrap: true,
         physics: ScrollPhysics(),
         itemCount: streamSnapshot.data!.docs.length,
         itemBuilder: (BuildContext context,int index) {
          DocumentSnapshot d = streamSnapshot.data!.docs[index];
          final path = d['path'];
          final name = d['name'];
          final blah = d['price'];
          final brand = d['brand'];
          final gear = d['gear'];
          final color = d['color'];
          final fuel = d['fuel'];
          final coid = d['coid'];
          final id = d.id;
          int price = int.parse(blah);


          return SlideInDown(
           child: Container(
            //height: MediaQuery.of(context).size.height * 0.25,
            //width: MediaQuery.of(context).size.width * 0.5,
            child: InkWell(
             onTap: () {
              Navigator.push(context, MaterialPageRoute(builder: (context)=>
CarDetail(path,name,price,brand,gear,color,fuel,coid,id)));
             },
             child: Container(
              height: 180,
              decoration: const BoxDecoration(
                boxShadow: [
                 BoxShadow(
                   color: Colors.black26, blurRadius: 5, spreadRadius: 1)
                ],
                gradient: LinearGradient(
                  begin: Alignment.topLeft,
```

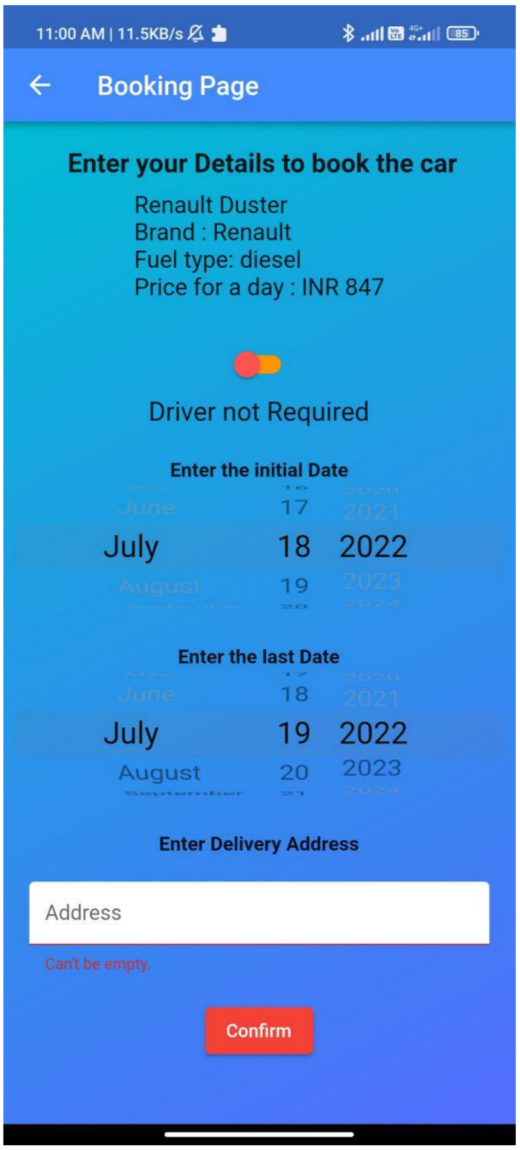
```
                        end: Alignment.bottomRight,
                        colors: [Colors.lightBlueAccent, Colors.blueAccent])
                  ),
                  margin: EdgeInsets.only(
                     top: index.isEven ? 0 : 25, bottom: index.isEven ? 20 : 0),
                  child: Column(
                   children: [
                     Container(
                       height: 100,
                       decoration: BoxDecoration(
                         color: Colors.white,
                         image: DecorationImage(
                           image: NetworkImage(
                             // d['path'],
                             path,
                           ),
                           fit: BoxFit.contain,
                         ),
                         borderRadius: BorderRadius.circular(1),
                       ),
                     ),
                     SizedBox(height: 3),
                     Text(
                       name,
                       style: TextStyle(
                          fontWeight: FontWeight.bold
                       ),
                     ),
                     Text((price).toString(), style: SubHeading),
                     Text('per month'),
                   ],
                  ),
                 ),
                ),
               ),);
             },
             gridDelegate:
             SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 2,crossAxisSpacing: 8),
           );
         }
         return const Center(
           child: CircularProgressIndicator(),
         );
        },
       ),
      ),
     ),
     drawer: AppDrawer(),
    );
   }
}
```

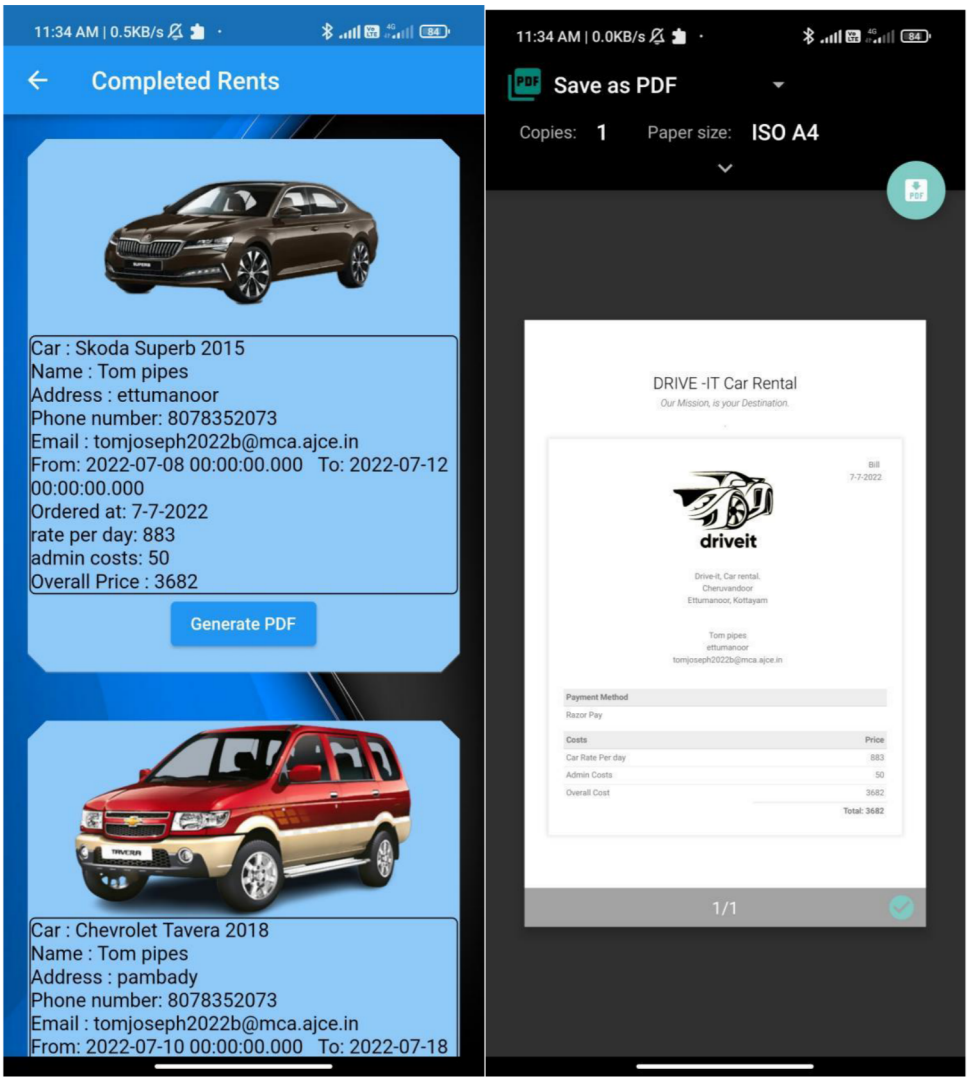
9.2 Screen Shots

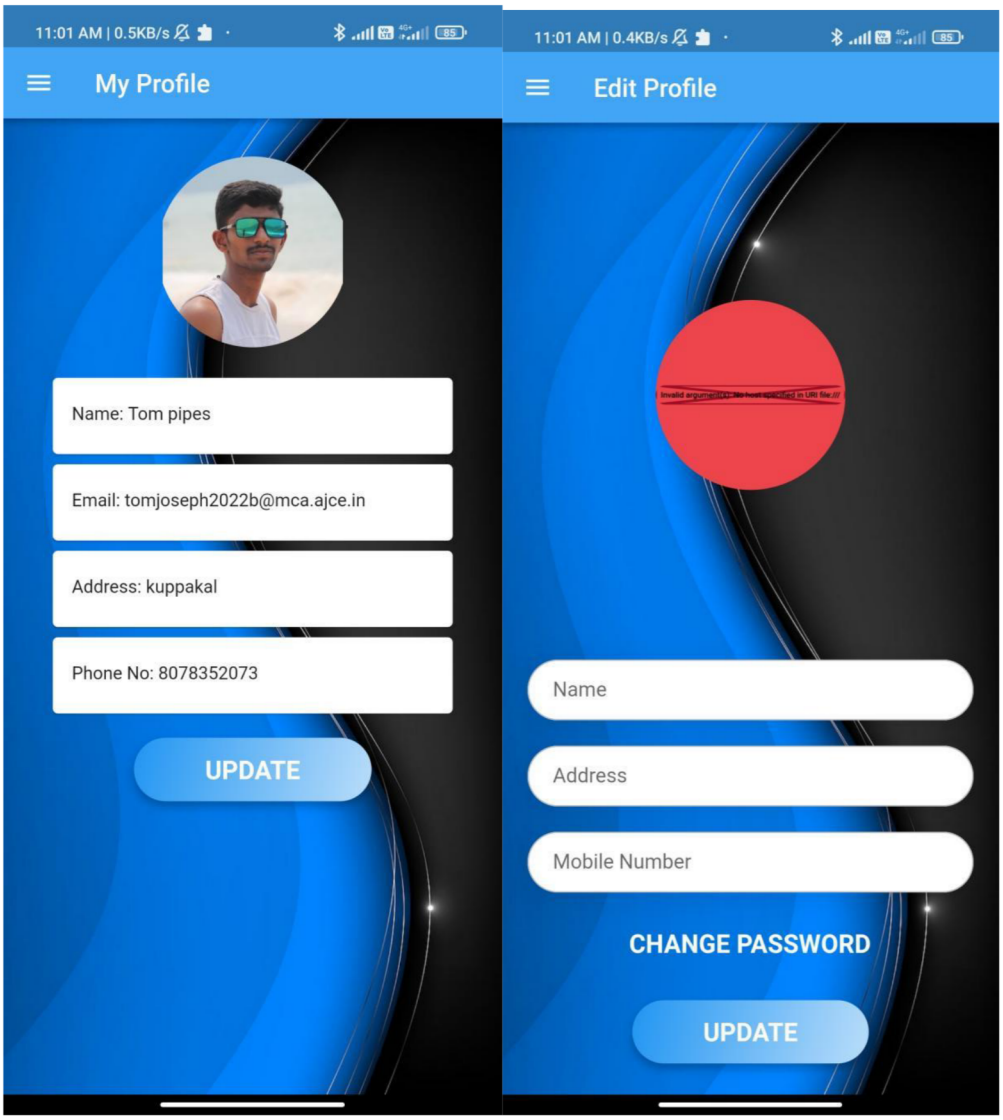
Home page

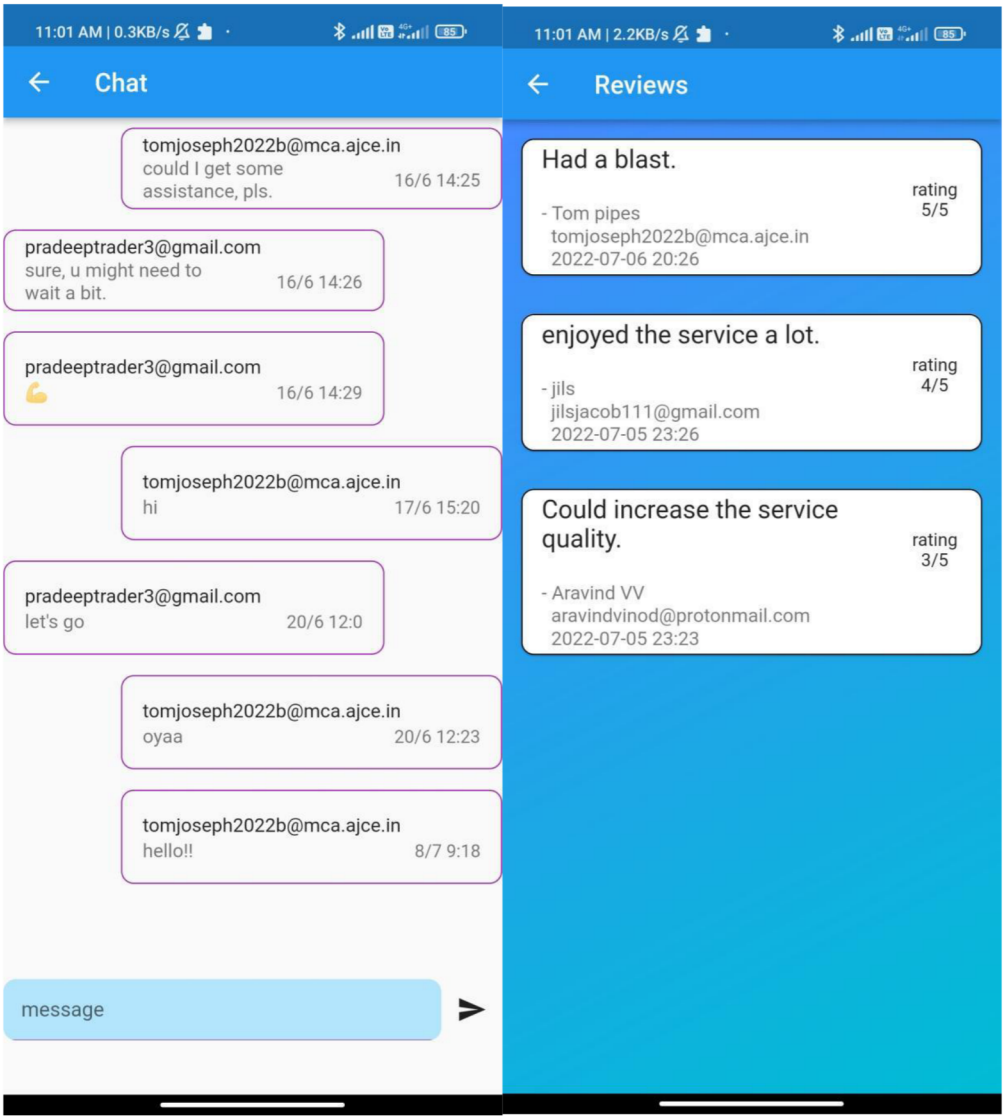



11:00 AM | 0.2KB/s
Renault Duster
Renault
12 Month
304800.0
INR
6 Month
152400.0
INR
1 Month
25400.0
INR
SPECIFICATIONS
Color
grey
Gearbox
5
Fuel
diesel
Book Now
11:00 AM | 2.0KB/s
Renault Duster
Renault
12 Month
304800.0
INR
6 Month
152400.0
INR
1 Month
25400.0
INR
SPECIFICATIONS
Share
PERSONAL
WORK
Nearby Share
ShareMe
WhatsApp
Direct Message
Cancel


11:00 AM | 11.5KB/s
Booking Page
Enter your Details to book the car
Renault Duster
Brand : Renault
Fuel type: diesel
Price for a day : INR 847
Driver not Required
Enter the initial Date
June 17 2021
July 18 2022
August 19 2023
Enter the last Date
June 18 2021
July 19 2022
August 20 2023
Enter Delivery Address
Address
Can't be empty.
Confirm


11:34 AM | 0.5KB/s
Completed Rents
Car : Skoda Superb 2015
Name : Tom pipes
Address : ettumanoor
Phone number: 8078352073
Email : tomjoseph2022b@mca.ajce.in
From: 2022-07-08 00:00:00.000 To: 2022-07-12 00:00:00.000
Ordered at: 7-7-2022
rate per day: 883
admin costs: 50
Overall Price : 3682
Generate PDF
Car : Chevrolet Tavera 2018
Name : Tom pipes
Address : pambady
Phone number: 8078352073
Email : tomjoseph2022b@mca.ajce.in
From: 2022-07-10 00:00:00.000 To: 2022-07-18
11:34 AM | 0.0KB/s
Save as PDF
Copies: 1
Paper size: ISO A4
DRIVE -IT Car Rental
Our Mission, is your Destination.
driveit
Drive-it, Car rental.
Cheruvandoor
Ettumanoor, Kottayam
Tom pipes
ettumanoor
tomjoseph2022b@mca.ajce.in
Payment Method
Razor Pay
Costs
Price
Car Rate Per day
883
Admin Costs
50
Overall Cost
3682
Total: 3682
1/1


11:01 AM | 0.5KB/s
My Profile
Name: Tom pipes
Email: tomjoseph2022b@mca.ajce.in
Address: kuppakal
Phone No: 8078352073
UPDATE
11:01 AM | 0.4KB/s
Edit Profile
Name
Address
Mobile Number
CHANGE PASSWORD
UPDATE


11:01 AM | 0.3KB/s
Chat
tomjoseph2022b@mca.ajce.in
could I get some assistance, pls.
16/6 14:25
pradeeptrader3@gmail.com
sure, u might need to wait a bit.
16/6 14:26
pradeeptrader3@gmail.com
16/6 14:29
tomjoseph2022b@mca.ajce.in
hi
17/6 15:20
pradeeptrader3@gmail.com
let's go
20/6 12:0
tomjoseph2022b@mca.ajce.in
oyaa
20/6 12:23
tomjoseph2022b@mca.ajce.in
hello!!
8/7 9:18
message
11:01 AM | 2.2KB/s
Reviews
Had a blast.
rating 5/5
- Tom pipes
tomjoseph2022b@mca.ajce.in
2022-07-06 20:26
enjoyed the service a lot.
rating 4/5
- jils
jilsjacob111@gmail.com
2022-07-05 23:26
Could increase the service quality.
rating 3/5
- Aravind VV
aravindvinod@protonmail.com
2022-07-05 23:23