

AUTOLEND

Project Report Submitted by

BIBIN P DANIEL

Reg. No.: AJC20MCA-I025

In Partial fulfillment for the Award of the Degree Of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

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]

2022-2023

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**AUTOLEND**” is the bonafide work of **BIBIN P DANIEL (Regno: AJC20MCA-I025)** 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 2022-23.

Ms. Grace Joseph
Assistant Professor
Internal Guide

Ms. Meera Rose Mathew
Assistant Professor
Coordinator

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

DECLARATION

I hereby declare that the project report “**AUTOLEND**” is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Integrated Master of Computer Applications (INMCA) from APJ Abdul Kalam Technological University, during the academic year 2022-2023.

Date:20/04/2023

BIBIN P DANIEL

KANJIRAPPALLY

Reg: AJC20MCA-I025

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 wish to express my sincere gratitude to our Manager **Rev. Fr. Dr. Mathew Paikatt** and 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 us. I extend my wholehearted thanks to the project coordinator **Ms. Meera Rose Mathew, Assistant Professor** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Ms. Grace Joseph, Assistant Professor** 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.

BIBIN P DANIEL

ABSTRACT

This web-based mini project is designed to streamline the process of renting vehicles for both owners and renters. By providing a platform for clients to list their vehicles for rent, the project aims to connect individuals in need of a vehicle with those who have one to spare.

Clients will have the ability to list their vehicles for rent, including details such as make, model, year, images, rental cost, and availability. They can also set their own rental terms and conditions. Users will be able to search and view available vehicles, filter by make, model, rental cost, location etc., and then proceed to rent the vehicle of their choice. They can also check the reviews and ratings of the vehicle and the owner before booking.

The project will include a user-friendly interface, easy navigation, and search options to make the process of renting a vehicle as simple as possible. It will also include a secure payment system, so users can make payments safely and easily. The booking system will be simple and easy to use, allowing both clients and users to manage their bookings and reservations with ease.

This project aims to provide an alternative for individuals who do not own a vehicle or need a temporary replacement, while also providing a source of income for vehicle owners. Additionally, it will help to reduce the traffic congestion on the roads by providing an alternative mode of transportation.

Overall, this web-based mini project will provide an efficient and convenient solution for both vehicle owners and renters. It will make the process of renting a vehicle more accessible, affordable, and secure for everyone involved. This will be a win-win situation for both parties as vehicle owners can earn extra income by renting out their idle vehicles and renters can rent a vehicle at a fraction of the cost of buying one.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	3
2	SYSTEM STUDY	4
2.1	INTRODUCTION	5
2.2	EXISTING SYSTEM	5
2.3	DRAWBACKS OF EXISTING SYSTEM	6
2.4	PROPOSED SYSTEM	6
2.5	ADVANTAGES OF PROPOSED SYSTEM	6
3	REQUIREMENT ANALYSIS	8
3.1	FEASIBILITY STUDY	9
3.1.1	ECONOMICAL FEASIBILITY	9
3.1.2	TECHNICAL FEASIBILITY	9
3.1.3	BEHAVIORAL FEASIBILITY	10
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	10
3.2	SYSTEM SPECIFICATION	11
3.2.1	HARDWARE SPECIFICATION	11
3.2.2	SOFTWARE SPECIFICATION	11
3.3	SOFTWARE DESCRIPTION	12
3.3.1	PHP	12
3.3.2	MYSQL	12
4	SYSTEM DESIGN	13
4.1	INTRODUCTION	14
4.2	UML DIAGRAM	14
4.2.1	USE CASE DIAGRAM	15
4.2.2	SEQUENCE DIAGRAM	18
4.2.3	STATE CHART DIAGRAM	20
4.2.4	ACTIVITY DIAGRAM	20
4.2.5	CLASS DIAGRAM	23
4.2.6	OBJECT DIAGRAM	25
4.2.7	COMPONENT DIAGRAM	27

4.2.8	DEPLOYMENT DIAGRAM	27
4.2.9	COLLABORATION DIAGRAM	27
4.3	USER INTERFACE DESIGN USING FIGMA	28
4.4	DATA BASE DESIGN	30
5	SYSTEM TESTING	37
5.1	INTRODUCTION	38
5.2	TEST PLAN	38
5.2.1	UNIT TESTING	39
5.2.2	INTEGRATION TESTING	40
5.2.3	VALIDATION TESTING	40
5.2.4	USER ACCEPTANCE TESTING	41
5.2.5	AUTOMATION TESTING	41
5.2.6	SELENIUM TESTING	42
6	IMPLEMENTATION	51
6.1	INTRODUCTION	52
6.2	IMPLEMENTATION PROCEDURE	52
6.2.1	USER TRAINING	53
6.2.2	TRAINING ON APPLICATION SOFTWARE	53
6.2.3	SYSTEM MAINTENANCE	53
7	CONCLUSION & FUTURE SCOPE	57
7.1	CONCLUSION	57
7.2	FUTURE SCOPE	57
8	BIBLIOGRAPHY	58
9	APPENDIX	60
9.1	SAMPLE CODE	61
9.2	SCREEN SHOTS	69

List of Abbreviation

- DE - Integrated Development Environment
- HTML - Hyper Text Markup Language
- CSS - Cascading Style Sheet
- UML – Unified Modeling Language
- SQL – Structured Query Language
- AJAX – Asynchronous JavaScript and XML

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

AutoLend is a vehicle rental system that connects car owners with car lenders and renters. It has different components that allow users to register, verify their identity, search, select, and pay for vehicles. Car owners can add their vehicles, manage availability, view ratings, download reports, and cancel bookings. The admin module provides analytics, tools for managing users, vehicles, bookings, and payments, and the ability to process refunds. AutoLend aims to offer a smooth and user-friendly experience for all users.

1.2 PROJECT SPECIFICATION

Auto Lend is a comprehensive online vehicle rental system that aims to simplify the process of car rental. The system connects car owners with car lenders, providing a platform for easy vehicle bookings. The system is having three main users, including admin, car owner, and car lender, each serving a specific purpose.

Car Lender:

The car rental system provides the primary interface for users to rent vehicles. Users need to register and log in to the website and verify their identity by submitting their vehicle license details. After being verified by the admin, users can proceed to reserve vehicles. The user interface enables users to search, sort, and browse through a list of available vehicles. When users click on a specific vehicle, they can view its full details, including ratings provided by other users. Users can select a pickup location, drop-off location, and dates to book and proceed to the payment page.

Car Owner:

The Interface for car owners enables them to easily manage their vehicles that are available for rent. They can view their vehicles and quickly enable or disable them for booking. Additionally, they can download a report of their vehicle and update its details, such as the ratings given by users. Car owners can also keep track of all their bookings, view their status, cancel them if necessary, and download a report of their bookings.

Admin:

The admin function of the system is responsible for overseeing and managing the entire platform. The admin is provided with a dashboard that displays an overview of key metrics, including the total number of vehicles, bookings, and average income. From there, the admin can access a table that lists all users and take actions such as blocking or activating their accounts. Additionally, the admin can verify the license details of users and view car owner information, enabling them to block or activate accounts, as necessary.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

AutoLend is a web-based vehicle rental system that provides a platform for car owners to rent out their vehicles and for car lenders to rent vehicles for personal or commercial use. The system is designed to simplify the process of car rental, making it easy for users to search for available cars, book them, and make payments online. AutoLend is targeted towards individuals and businesses who want to rent cars for short or long-term periods.

The system study of AutoLend involves an in-depth analysis of the requirements, objectives, and constraints of the project. The primary goal of the system study is to identify the user requirements and design a system that meets their needs. The system study also involves identifying the technical, operational, and economic feasibility of the project.

2.2 EXISTING SYSTEM

2.2.1 NATURAL SYSTEM STUDIED

The natural system for renting and renting out vehicles would involve a more manual and decentralized process. Without this project, the process of renting and renting out vehicles would involve traditional methods such as classified ads or word of mouth. This would be less efficient, as it would require more time and effort to find and rent a vehicle. It would also be less secure as there would be no way of verifying the credibility of the vehicle owner or the renter and there would be no standardized payment methods or booking systems in place. It would not have a centralized platform that connects vehicle owners and renters making it more difficult and time-consuming for both parties to find each other.

2.2.2 DESIGNED SYSTEM STUDIED

AutoLend is a web-based system that allows customers to search and book vehicles from different car owners at their preferred location and time. The system provides a user-friendly interface that makes it easy for customers to search for vehicles based on their preferences, such as vehicle type, location, and rental duration. Customers can also view vehicle details, ratings, and reviews before making a booking.

For car owners, the system provides a platform to rent out their vehicles and earn extra income. They can list their vehicles on the platform and set their own rental prices and availability. Car owners can also view their bookings, update their vehicle details, and receive payments through the system.

AutoLend also provides an administrative module that allows administrators to manage the system, verify users, and monitor bookings. Administrators can view reports on the system's performance and make decisions based on the data.

Overall, AutoLend provides a more convenient and efficient way of renting vehicles, benefiting both customers and car owners.

2.3 DRAWBACKS OF EXISTING SYSTEM

- **Time-consuming process:** In the traditional vehicle rental system, customers must physically visit rental offices or make phone calls to book a vehicle. This process can be time-consuming, especially if the customer must visit multiple rental offices to compare prices and vehicle availability.
- **Limited choices:** Customers have limited choices when it comes to vehicle models and rental locations. This can be inconvenient, especially if the customer has a specific vehicle model or location in mind.
- **Paperwork:** The traditional vehicle rental system involves a lot of paperwork, such as rental agreements and insurance forms. This can be tedious and confusing for customers, especially if they are not familiar with the rental process.
- **Inflexible rental terms:** Rental terms in the traditional vehicle rental system can be inflexible, with fixed rental durations and limited options for pickup and drop-off locations. This can be inconvenient for customers who need more flexibility in their rental arrangements.
- **Limited transparency:** Customers may not have access to sufficient information about the rental vehicle's condition, previous usage, and maintenance history. This lack of transparency can lead to unexpected issues during the rental period.

2.4 PROPOSED SYSTEM

The proposed system is an online vehicle rental platform called AutoLend, which aims to improve the existing vehicle rental system by providing a user-friendly and efficient online platform. It allows car owners to list their vehicles, while car renters can easily search and book a vehicle that suits their needs. The system also includes features such as vehicle verification, booking management, payment processing, and a rating system. Overall, the proposed system provides a convenient and hassle-free experience for both car owners and renters.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- **User-friendly Interface:** AutoLend has a simple and user-friendly interface, making it easy for both car owners and renters to use the system.
- **Online Verification:** The system provides online vehicle verification, ensuring that all

vehicles listed on the platform are safe and legal to use.

- **Easy Booking Management:** AutoLend offers efficient booking management, enabling car renters to easily search, book, and manage their bookings.
- **Secure Payment Processing:** The system provides secure payment processing, ensuring that all transactions are safe and reliable.
- **Rating System:** AutoLend offers a rating system that allows car renters to rate their experience with the vehicle and the car owner, helping other renters to make informed decisions.
- **Efficient Reporting:** The system offers efficient reporting, enabling car owners to view reports on their vehicles and bookings.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

A feasibility study evaluates a project's or system's practicality. As part of a feasibility study, the objective and rational analysis of a potential business or venture is conducted to determine its strengths and weaknesses, potential opportunities and threats, resources required to carry out, and ultimate success prospects. Two criteria should be considered when judging feasibility: the required cost and expected value.

A feasibility study is a comprehensive evaluation of a proposed project that evaluates all factors critical to its success to assess its likelihood of success. Business success can be defined primarily in terms of ROI, which is the number of profits that will be generated by the project.

Various feasibility studies are:

- a)Economic Feasibility
- b)Technical Feasibility
- c)Behavioral Feasibility

3.1.1 Economic Feasibility

- Increased accessibility: Online platform increase the accessibility of the rental vehicles to a larger audience, leading to an increase in demand.
- Reduced costs: By removing the need for a physical rental location and staff, the cost of operating the system can be reduced, leading to increased profitability.
- Data analytics: The system can collect and analyze data on rental patterns, customer preferences, and vehicle utilization, which can be used to optimize pricing and improve the rental experience.
- Convenience: Online rental systems provide customers with the convenience of browsing, booking, and paying for rentals from their own devices, increasing the overall satisfaction.
- Increased revenue: By providing a platform for more vehicle owners to list their vehicles for rent, the system can increase the overall revenue generated by the rental industry.

3.1.2 Technical Feasibility

With the advancement of technology and widespread use of the internet, creating and implementing a web-based vehicle rental system is a feasible and viable option. The system can be designed using various technologies such as HTML, CSS, JavaScript, and a backend database like MySQL or MongoDB. The system can be hosted on a server, allowing users to access it from any location with an internet connection. Additionally, various security measures can be implemented to ensure that the user's data and information is secure. These technical advancements have made it possible to build a functional, user-friendly, and secure web-based vehicle rental

system.

3.1.3 Behavioral Feasibility

Behavioral feasibility is a crucial aspect to consider when implementing a new system. It involves assessing how well the system will be received and used by the end-users. In the case of the AutoLend system, behavioral feasibility can be evaluated by considering factors such as user experience, ease of use, and user satisfaction.

The AutoLend system has been designed with a user-friendly interface that is easy to navigate. Users can easily search for their preferred vehicles, view the vehicle details, book and pay for their reservations, and manage their bookings. The system also provides a seamless and convenient experience for car owners to manage their vehicles and bookings.

Furthermore, the system offers a rating system that allows users to rate the vehicles they have rented. This provides a platform for feedback, which can be used to improve the system's overall user experience. The system also offers a booking cancellation feature that provides users with the flexibility to cancel their bookings, if necessary, albeit with a 10% loss of the booking amount.

Overall, the AutoLend system's design and features have been developed with user behavior in mind, making it easy and convenient for users to interact with the system.

3.14. Questionnaire

1. Is there any legal documentation to be done before renting out vehicle?

Yes, there are documents required before renting a vehicle which include:

- Vehicle Registration Certificate (RC)
- Insurance Certificate
- Pollution Under Control (PUC) certificate
- Driving License of the person who will be renting the vehicle
- Address Proof of the vehicle owner

And there will agreement between vehicle owner and the renter that outlines the terms and conditions of the rental.

2.What are the ways to assure if a renter is genuine?

It is typically ensured by examining official identity documents and withholding any such documents during the rental duration.

3.What data do you gather from customers before renting out vehicles?

Before renting out vehicles, personal information and information on the renter's driver's license are collected.

4.How do you currently handle vehicle maintenance and upkeep?

The vehicles are serviced for each 5000 kms and are washed and thoroughly checked after each trip.

5.what are the actions you take when an accident or any another similar incident occur?

Usually, when an accident occurs, the repair costs are covered by insurance, and the renter is charged for any excess costs.

6.How do you make payments and records them?

The cost of renting a car is determined by the car and the rental period. Most payments are done in person, either with cash or online, and they are noted in the register.

7.How are your vehicle bookings made?

Vehicles are frequently reserved by phone or in person. A little fee is charged for reservations.

8.What are the most common issues that arise during a rental period?

Several issues can arise throughout the rental time, such as late returns, damaged vehicles, or renters who break traffic laws.

9.What are the key factors that influence a customer's decision to rent a vehicle from your showroom?

The mileage, number of seats, and brand of the car are the key factors.

10. What are the most common customer requests or questions?

Most customers would ask for extra services like child seats and GPS. Additionally, there will be inquiries about the locations of vehicle drop-offs and fuel policy.

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - intel i3 or higher

RAM - 4 GB or higher

Hard disk - 1 5 GB or higher

3.2.2 Software Specification

Front End - HTML, CSS,

Backend - MySQL

Client on PC - Windows 7 and above.

Technologies used - JS, HTML5, AJAX, J Query, PHP, CSS, PHP mailer, Razor pay, Dompdf, Bootstrap

3.3 SOFTWARE DESCRIPTION

3.3.1 PHP

PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language. PHP is now installed on more than 244 million websites and 2.1 million web servers. Originally created by Rasmus Lerdorf in 1995, the reference implementation of PHP is now produced by the PHP group. While PHP originally stood for personal home page, it now stands for PHP: Hypertext Preprocessor, a recursive acronym code is interpreted by a web server with a PHP processor module which generates the resulting web page. PHP commands can be embedded directly into a HTML source document rather than calling an external file to process data. It has also evolved to include a command-line interface. capability and can be used in standalone incompatible with the GNU General Public License (GPL) due to restrictions on the usage of the term PHP. PHP can be deployed on most web servers and as a standalone shell on almost every operating system and platform, free of charge.

3.3.2 MySQL

MySQL is a relational database management system (RDBMS) that is widely used for managing structured data. It is an open-source software, which means that users can download and use it for free. MySQL is known for its high performance, reliability, and ease of use, making it a popular choice for many applications and websites.

MySQL is based on the Structured Query Language (SQL), which is a standard language used for managing relational databases. It supports a wide range of SQL commands, including SELECT, INSERT, UPDATE, DELETE, and many more. This makes it easy for users to perform a variety of operations on their data, from simple queries to complex transactions.

MySQL is also highly scalable, which means that it can handle large amounts of data and users without slowing down. It supports multiple concurrent connections and can be used with a variety of programming languages and frameworks.

In addition to its core features, MySQL also offers a range of advanced features, including support for stored procedures, triggers, and views. It also has strong security features, including support for encryption and authentication.

Overall, MySQL is a powerful and reliable RDBMS that is well-suited for a wide range of applications, from small websites to large enterprise systems.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Any designed system or product's development process begins with design. A creative process is design. The secret to an efficient system is a decent design. The process of using different methodologies and concepts to specify a process or a system in enough detail to allow for its physical realization is referred to as "design." It can be characterized as the process of using different methodologies and concepts to specify a device, a process, or a system in enough detail to allow for its physical realization. Regardless of the development paradigm employed, software design forms the technical core of the software engineering process. The architectural detail needed to construct a system or product is developed through the system design. This program has also through the best possible design phase, fine tuning all efficiency, performance, and accuracy levels, as in the case of any systematic technique. A user-oriented document is converted into a document for programmers or database staff throughout the design phase. The two stages of system design development are logical design and physical design.

4.2 UML DIAGRAM

A UML diagram is a graphical representation of a software system that uses the Unified Modeling Language (UML) notation. UML is a standard way of describing the structure, behavior, and interactions of software components using different types of diagrams. UML diagrams can help developers and stakeholders to communicate, document, and analyze the design and requirements of a software system. Some of the most common UML diagrams are:

- Class diagram: shows the classes and relationships among them in a system. Classes are the basic units of object-oriented programming that define the attributes and methods of objects.
- Use case diagram shows the actors and use cases in a system. Actors are the external entities that interact with the system, and use cases are the scenarios or functions that the system provides to the actors.
- Sequence diagram: shows the interactions among objects in a system in a chronological order. Objects are instances of classes that communicate with each other by sending messages.
- Activity diagram: shows the flow of actions or activities in a system. Activities are the steps or tasks that need to be performed to achieve a goal or outcome.
- State diagram: shows the states and transitions of an object or a system. States are the conditions or situations that an object or a system can be in, and transitions are the events or triggers that cause a change from one state to another.
- Component diagram: shows the components and dependencies among them in a system.

Components are the physical or logical parts of a system that provide specific functionality or services.

- Deployment diagram: shows the nodes and artifacts in a system. Nodes are the physical devices or machines that host the components or artifacts, and artifacts are the executable files or documents that are deployed on the nodes.

UML diagrams can be created using various tools, such as Smart Draw, which is an online diagramming software that supports UML notation and allows users to create, edit, and share UML diagrams easily.

4.2.1 USE CASE DIAGRAM

A use case diagram is a graphical representation of the interactions between a system and its external entities, such as users, customers, or other systems. A use case diagram shows the functionality of a system from the perspective of the actors who use it. A use case diagram can help to:

- Specify the context and scope of a system.
- Capture the functional requirements of a system.
- Validate the system architecture and design.
- Drive the implementation and testing of the system.

A use case diagram consists of four main elements:

- Actors: The roles that interact with the system, such as human users, external systems, or devices. Actors are represented by stick figures or icons.
- Use cases: The goals or tasks that the actors want to achieve by using the system. Use cases are represented by ovals with descriptive names.
- Relationships: The connections between actors and use cases, or between use cases themselves. Relationships are represented by different types of lines, such as associations, includes, extends, or generalizations.
- System boundary: The boundary that defines the scope and context of the system under consideration. The system boundary is represented by a rectangle that encloses the use cases.

A use case diagram is a useful tool for modeling and communicating the functional requirements of a system. It can help to clarify the expectations and needs of the stakeholders, as well as to identify any gaps or inconsistencies in the system design. A use case diagram can also serve as a basis for creating other diagrams, such as sequence diagrams, activity diagrams, or state machine diagrams, that describe how the system behaves in more detail.

**Figure 1: Use Case Diagram**

4.2.1 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram that shows how operations are carried out in a system or a process. It illustrates what messages are sent between different objects and when they occur. A sequence diagram is organized according to time, which progresses from top to bottom of the diagram. The objects involved in the operation are listed from left to right according to when they take part in the message sequence.

A sequence diagram consists of several elements, such as:

- Lifelines: vertical dashed lines that represent the existence of an object over time. They are labeled with the name and type of the object.
- Activation boxes: thin rectangles on a lifeline that indicate when an object is active or executing a method.
- Messages: horizontal arrows that show the communication between objects. They can be synchronous (solid line), asynchronous (dashed line), or return (dotted line). They are labeled with the name and parameters of the method being invoked.
- Combined fragments: rectangular frames that enclose a part of the interaction to show conditional or looping behavior. They have an operator (such as alt, opt, loop, etc.) and a guard condition in the top left corner.
- Interaction references: dashed rectangles that refer to another sequence diagram for a sub-sequence of events. They have a name and parameters in the top left corner.

A sequence diagram can help to model the dynamic behavior of a system or a process, to show the interactions and collaborations among objects, and to identify potential problems or errors in the logic or timing of the operation.

Sequence Diagram

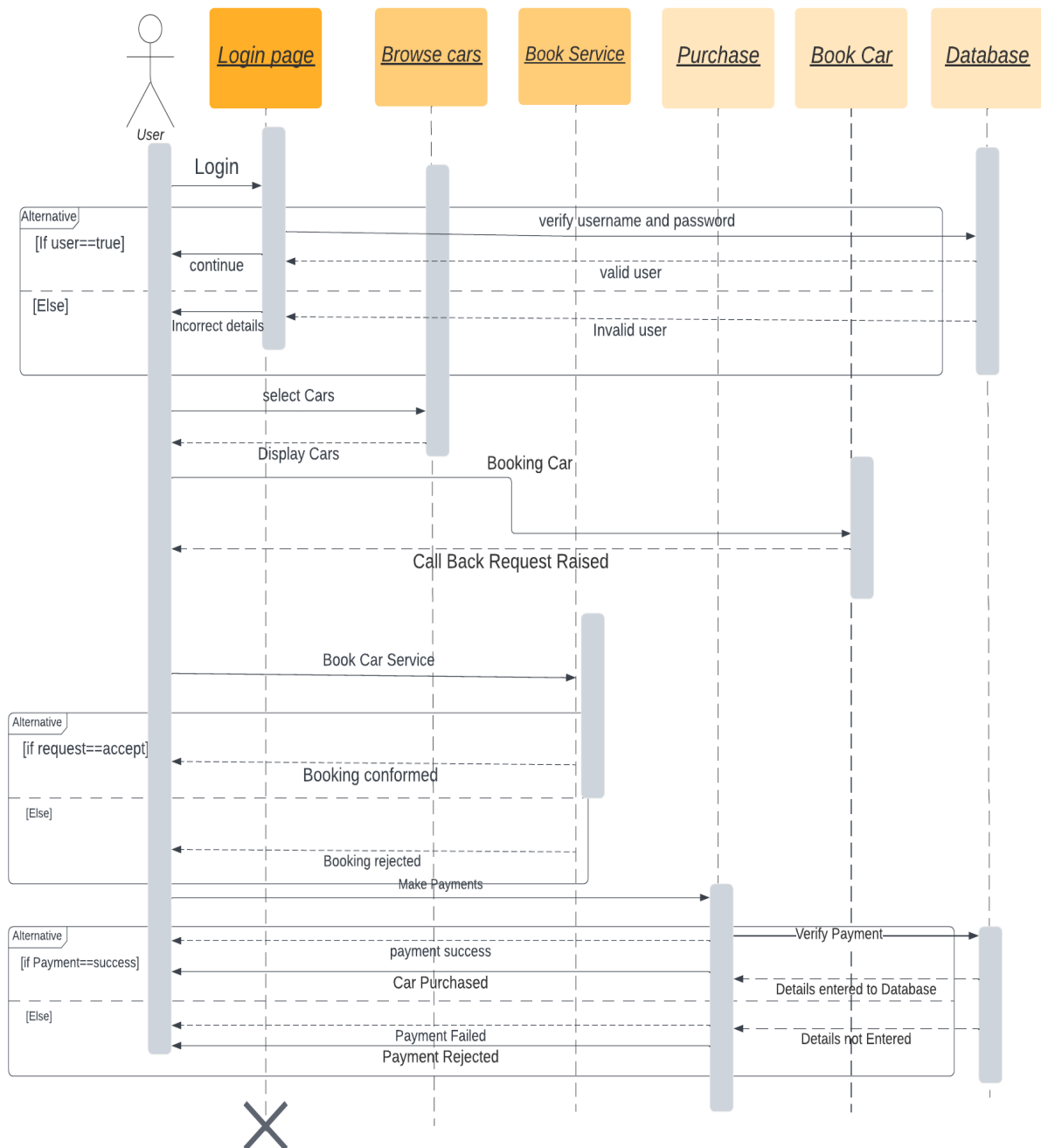


Figure 2: Sequence Diagram

4.2.2 State Chart Diagram

A State chart diagram is a graphical representation of a state machine. A state machine is a system that can have different states depending on some events or conditions. A State chart diagram shows the possible states of an object and the transitions between them.

A State chart diagram consists of the following elements:

- States: A state is a condition or situation in which an object exists. A state can have a name, an entry action, an exit action, and internal activities. A state is represented by a rounded rectangle with the name inside.
- Transitions: A transition is a change from one state to another. A transition can have a trigger, a guard condition, and an effect. A trigger is an event that causes the transition to occur. A guard condition is a Boolean expression that must be true for the transition to take place. An effect is an action that is performed when the transition happens. A transition is represented by a solid arrow with the trigger, guard condition, and effect along the arrow.
- Initial state: An initial state is the state in which an object starts its existence. An initial state is represented by a solid circle.
- Final state: A final state is the state in which an object ends its existence. A final state is represented by a solid circle with another circle around it.
- Choice: A choice is a point where the outcome of a transition depends on a condition. A choice is represented by a diamond with outgoing arrows labeled with conditions.
- Fork: A fork is a point where an object splits into concurrent substates. A fork is represented by a horizontal or vertical bar with one incoming arrow and multiple outgoing arrows.
- Join: A join is a point where concurrent substates merge into one state. A join is represented by a horizontal or vertical bar with multiple incoming arrows and one outgoing arrow.
- History: A history is a point that remembers the last active substate of a composite state. A history is represented by a circle with an H inside.

A State chart diagram can be used to model the behavior of an object, a class, a subsystem, or a system. It can also be used to model complex scenarios, workflows, protocols, or algorithms.

4.2.2 Activity Diagram

An activity diagram is a graphical representation of the dynamic behavior of a system or a process. It shows the flow of control from one activity to another, as well as the conditions and actions that

trigger or result from the transitions. An activity diagram can also depict the roles and responsibilities of the actors involved in the system or process.

An activity diagram consists of several elements, such as:

- Initial node: A small solid circle that indicates the start of the activity.
- Final node: A small solid circle inside a larger circle that indicates the end of the activity.
- Action: A rounded rectangle that represents a single unit of work or a step in the activity.
- Control flow: A solid arrow that connects two actions or nodes and shows the direction of execution.
- Decision node: A diamond-shaped symbol that represents a point where a choice must be made among two or more alternatives. It has one incoming control flow and two or more outgoing control flows.
- Merge node: A diamond-shaped symbol that represents a point where two or more alternative paths converge into one. It has two or more incoming control flows and one outgoing control flow.
- Fork node: A horizontal or vertical bar that splits a control flow into two or more parallel paths. It has one incoming control flow and two or more outgoing control flows.
- Join node: A horizontal or vertical bar that synchronizes two or more parallel paths into one. It has two or more incoming control flows and one outgoing control flow.
- Object node: A rectangle with dashed borders that represents an object or data used or produced by an action. It can have incoming and/or outgoing control flows and object flows.
- Object flow: A dashed arrow that connects an object node to an action or another object node and shows the movement of objects or data.
- Swimlane: A vertical or horizontal partition that groups related actions or nodes by actor or responsibility. It can have a label at the top or left side.

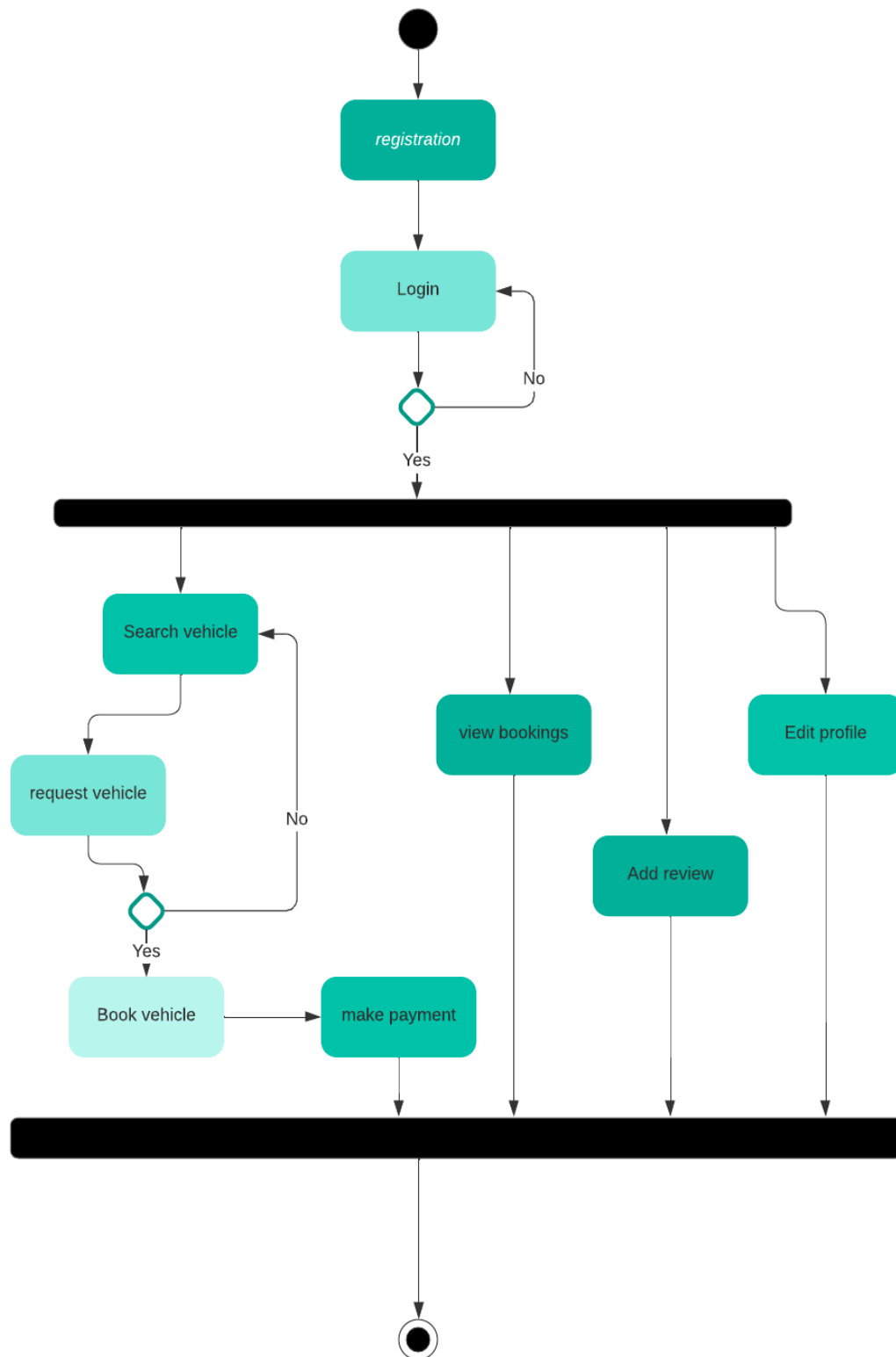


Figure 3: Activity Diagram

4.2.3 Class Diagram

A class diagram is a type of diagram that shows the structure of a system by representing its classes, attributes, operations, and relationships. A class diagram is one of the most common diagrams in the Unified Modeling Language (UML), which is a standard notation for modeling software systems. A class diagram can be used for various purposes, such as:

- Describing the static view of a system at a high level of abstraction
- Analyzing and designing the system's functionality and behavior
- Documenting the system's specifications and implementation details
- Communicating and collaborating with other developers and stakeholders

A class diagram consists of the following elements:

- **Class:** A class is a blueprint or template that defines the properties and behaviors of a group of objects. A class is represented by a rectangle with the class name on top, followed by the attributes and operations in separate compartments.
- **Attribute:** An attribute is a characteristic or feature of a class that describes its state or data. An attribute is represented by a name and an optional type, visibility, and default value.
- **Operation:** An operation is a function or method that defines the behavior or action of a class. An operation is represented by a name and an optional list of parameters, return type and visibility.
- **Relationship:** A relationship is a connection or association between two or more classes that indicates how they interact or depend on each other. There are different types of relationships, such as inheritance, association, aggregation, composition, and dependency. A relationship is represented by a line with an optional name, multiplicity, and direction.

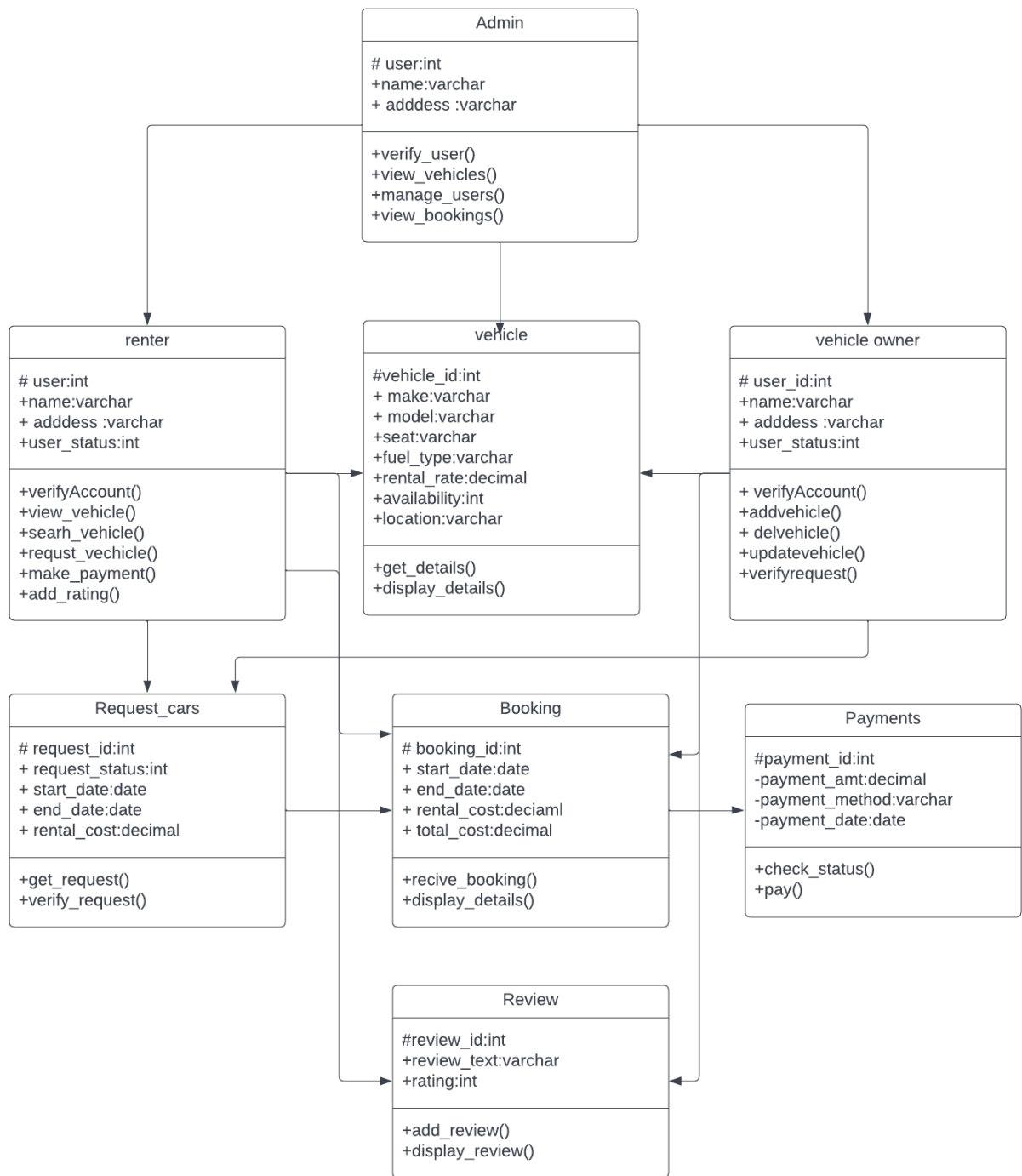


Figure 4: Class diagram

4.2.4 Object Diagram

An object diagram is a graphical representation of the instances of classes and their relationships in a specific scenario or situation. It shows the state of the objects at a given point in time. An object diagram is derived from a class diagram and can be used to illustrate the dynamic aspects of a system. An object diagram consists of the following elements:

- Objects: The instances of classes that are involved in the scenario. They are represented by rectangles with the name and attributes of the object.
- Links: The connections between objects that indicate their association or dependency. They are represented by lines with optional labels and multiplicity.
- Link attributes: The values of the attributes that belong to the links. They are represented by text labels attached to the links.
- Generalization: The inheritance relationship between classes that indicates that one class is a subtype of another. It is represented by a solid line with an empty arrowhead pointing to the superclass.
- Realization: The implementation relationship between classes that indicates that one class realizes the behavior specified by another. It is represented by a dashed line with an empty arrowhead pointing to the interface or abstract class.

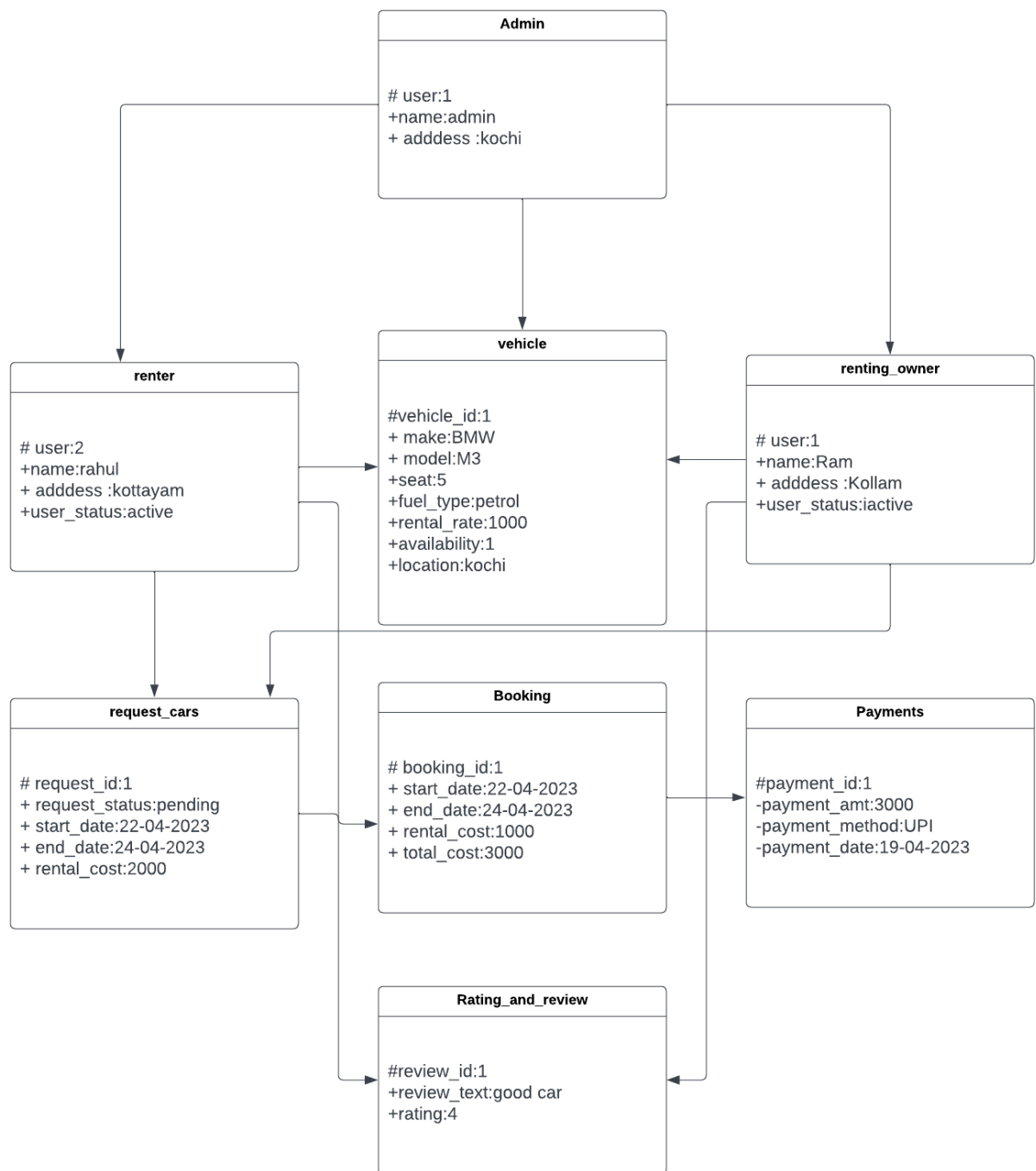


Figure 5: Object diagram

4.2.5 Component Diagram

A component diagram is a type of diagram that shows the physical components of a system and how they are organized and connected. A component is a modular unit that encapsulates some functionality or behavior of a system. Components can be software modules, libraries, frameworks, executables, or hardware devices. A component diagram helps to visualize the structure and dependencies of the components in a system, as well as the interfaces they provide and require. A component diagram can also show the deployment of the components on different nodes or platforms. A component diagram is useful for designing, documenting, and understanding complex systems that involve multiple components with different responsibilities and interactions.

4.2.8 Deployment Diagram

A deployment diagram is a type of diagram that shows the physical arrangement of the components of a software system. It depicts how the software artifacts, such as executable files, libraries, and databases, are deployed on the hardware nodes, such as servers, workstations, and devices. A deployment diagram also shows the communication links between the nodes, such as network connections, protocols, and bandwidth.

A deployment diagram is useful for modeling the physical aspects of a software system, such as the distribution, scalability, performance, and security. It can also show the configuration and dependencies of the components and nodes. A deployment diagram can be used to document the existing system architecture or to design a new one.

4.2.9 Collaboration Diagram

A collaboration diagram is a type of diagram that shows the interactions between objects in a system. It focuses on the roles and responsibilities of each object and how they collaborate to achieve a common goal. A collaboration diagram is also known as a communication diagram or an interaction diagram.

A collaboration diagram consists of objects, links, and messages. Objects are represented by rectangles with the object name and optionally the class name. Links are represented by solid lines that connect objects. They indicate the relationships or associations between objects. Messages are represented by arrows that show the direction and sequence of communication between objects. They can have labels that indicate the name, parameters, and return value of the message.

A collaboration diagram can be used to model different scenarios or use cases in a system. It can help to visualize the dynamic behavior and interactions of objects, as well as their roles and responsibilities. A collaboration diagram can also show the concurrency and synchronization of

messages, as well as conditional and iterative logic.

4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Login Page

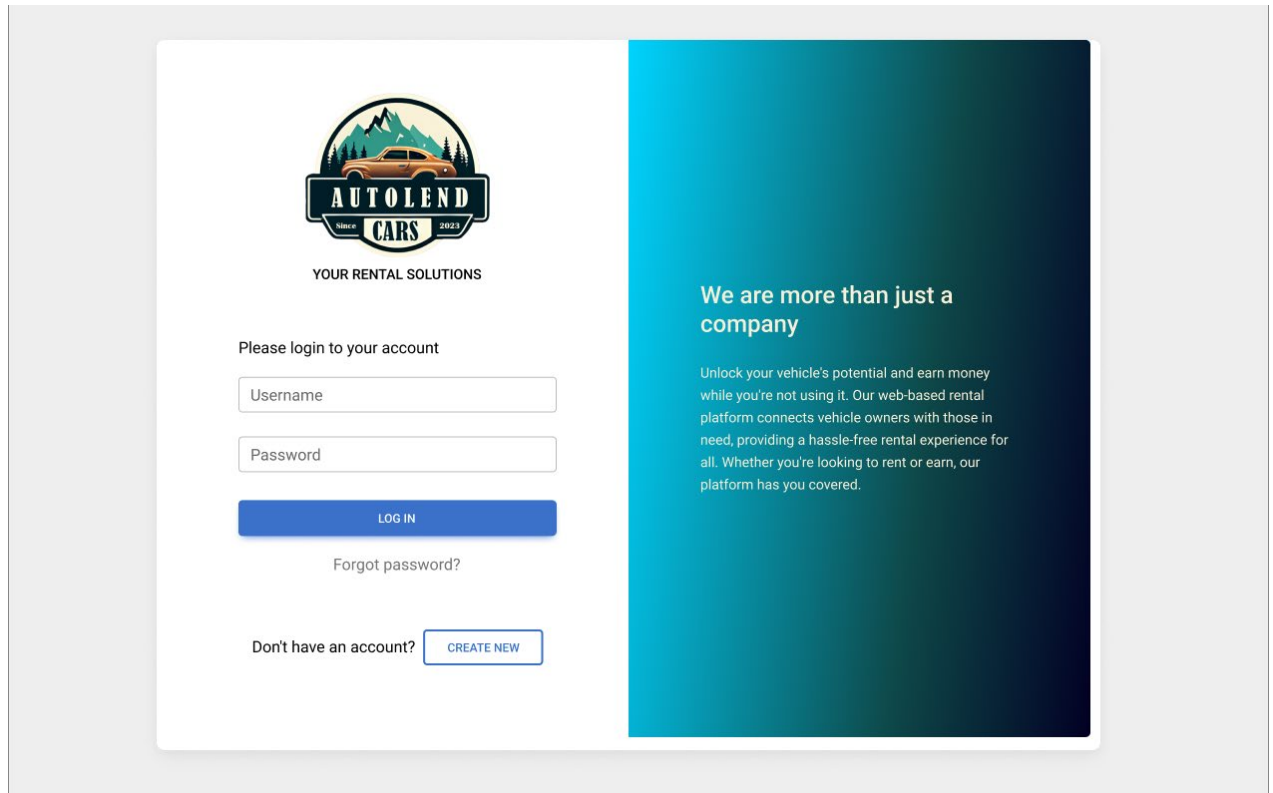


Figure 6: figma design login page

Form Name: Registration Page

Registration Form

First Name	User Name
DOB	Location
Email	Phone Number
Password	Confirm Password
Choose option	image

Already have an account? [LOGIN](#)

Figure 7: Figma Design Registration Page

4.4 DATABASE DESIGN

4.4.1 Relational Database Management System (RDBMS)

A Relational Database Management System (RDBMS) is a software that allows users to create, manage and manipulate data in a relational database. A relational database is a type of database that organizes data into tables, where each table consists of rows and columns. Each row represents a record or an entity, and each column represents an attribute or a property of that entity. The tables are linked by common fields or keys, which enable users to access data across tables using queries. An RDBMS provides various functions and features to facilitate data operations, such as:

- Data definition: Users can define the structure, schema and constraints of the tables using a data definition language (DDL).
- Data manipulation: Users can insert, update, delete and retrieve data from the tables using a data manipulation language (DML), such as SQL.
- Data security: Users can control the access and permissions of different users or groups to the tables or specific data using a data control language (DCL).
- Data integrity: Users can ensure the accuracy, consistency and validity of the data using rules, constraints, and triggers.
- Data backup and recovery: Users can protect the data from loss or damage by creating backups and restoring them in case of failure.
- Data concurrency: Users can allow multiple users to access and modify the data simultaneously without compromising its integrity using locking mechanisms and transactions.

Some examples of popular RDBMSs are MySQL, PostgreSQL, Oracle, SQL Server and SQLite. Each RDBMS may have its own variations or extensions of SQL syntax and features, but they all follow the basic principles of the relational model.

4.4.2 Normalization

Normalization is a process of organizing the data in a database to reduce redundancy and improve data integrity. Normalization also simplifies the database design so that it achieves the optimal structure composed of atomic elements (i.e., elements that cannot be broken down into smaller parts).

There are several types of normalization rules or normal forms that can be applied to a database. The most common ones are:

- **First Normal Form (1NF):** This rule ensures that each attribute in a table has a single value and does not contain repeating groups of values. For example, a table that stores the

names and phone numbers of customers should not have multiple phone numbers in one column, but rather have a separate column for each phone number.

- **Second Normal Form (2NF):** This rule ensures that each attribute in a table depends overall primary key and not on a part of it. For example, a table that stores the order details of customers should not have the customer's name as an attribute, since it depends only on the customer ID, which is part of the primary key. The customer's name should be stored in a separate table that relates to the order table by the customer ID.
- **Third Normal Form (3NF):** This rule ensures that each attribute in a table depends only on the primary key and not on any other attribute. For example, a table that stores the order details of customers should not have the product price as an attribute, since it depends on the product ID, which is another attribute in the table. The product price should be stored in a separate table that relates to the order table by the product ID.

Normalization helps to avoid data anomalies, such as insertion, deletion, and update anomalies, that can cause inconsistency and errors in the database. Normalization also makes it easier to query and manipulate data, since it reduces the number of joins and calculations required.

4.4.3 Sanitization

An automated procedure called "sanitization" is used to get a value ready for use in a SQL query. This process typically involves checking the value for characters that have a special significance for the target database. To prevent a SQL injection attack, you must sanitize(filter) the input string while processing a SQL query based on user input. For instance, the user and password input are a typical scenario. In that scenario, the server response would provide access to the 'Target user' account without requiring a password check.

4.4.4 Indexing

By reducing the number of disk accesses needed when a query is completed, indexing helps databases perform better. It is a data structure method used to locate and access data in a database rapidly. Several database columns are used to generate indexes. The primary key or candidate key of the table is duplicated in the first column, which is the Search key. To make it easier to find the related data, these values are kept in sorted order. Recall that the information may or may not be kept in sorted order.

4.5 TABLE DESIGN

1.Tbl_login

Primary key: **login_id**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	login_id	int	Primary Key	Primary key in the table
2	username	varchar	Not Null	Username for login
3	password	varchar	Not Null	Password to login

2.Tbl_user

Primary key: **User_id**

Foreign key: **login_id** references table **Tbl_login**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	User id	Int	Primary key	Primary key of the table
2	Login_id	Int	Foreign key	Foreign key reference to login table
3	name	Varchar (50)	Not Null	Name of the user
4	dob	Date	Not Null	Date of birth of user
5	location	Varchar (50)	Not Null	Location of the user
6	email	Varchar (100)	Not Null	Email of the user
7	mobile	Int	Not Null	Mobile number of users
8	User_type	Int	Not Null	Type of user(owner/renter)
9	User_status	Int	Not Null	Status of user active or blocked
10	image	varchar	Not Null	Image of the user

3.Tbl_vehicle

Primary key: **Vehicle_id**

Foreign key: **Category_id** references table **Tbl_category**, **User_id** references table **Tbl_users**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	Vehicle_id	Int	Primary key	Primary key of the table

2	Brand_name	Varchar (50)	Not Null	Brand name of vehicle
3	Modal_name	Varchar (50)	Not Null	Modal name of vehicle
4	Mileage	Int	Not Null	Mileage of the vehicle
5	Year	Varchar (10)	Not Null	Year of vehicle
6	Fuel_type	Varchar (50)	Not Null	Fuel type of the vehicle
7	Transmission_type	Varchar (50)	Not Null	Transmission type of the vehicle
8	Seat	Int	Not Null	Number of seats in the vehicle
9	Rate	Int	Not Null	Rate of vehicle per day
10	Status	Int	Not Null	Vehicle availability
11	Location	Varchar (50)	Not Null	Location of the vehicle
12	Category_id	Int	Foreign key	References to category table
13	User_id	Int	Foreign key	References to user table
14	Rc_book	Varchar (50)	Not Null	RC book to upload
15	Puc	Varchar (50)	Not Null	PUC certificate to upload
16	Insurance	Varchar (50)	Not Null	Insurance certificate to upload
17	permit	Varchar (50)	Not Null	permit certificate to upload
18	image	Varchar (50)	Not Null	Image of the vehicle

4.Tbl_vehicle_booking

Primary key: **Booking_id**

Foreign key: vehicle_id references table **Tbl_vehicle**, User_id references table Tbl_users

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	Booking_id	Int	Primary key	Primary key of the table
2	Vehicle_id	Int	Foreign key	reference to vehicle table
3	User_id	Int	Foreign key	reference to user table
4	Start_date	Date	Not Null	Starting date of booking
5	End_date	Date	Not Null	End date of booking
6	Pick_up_location	Varchar (50)	Not Null	Pick up location of vehicle

7	Pick_up_time	Varchar (50)	Not Null	Pick time of vehicle
8	Drop_of_location	Varchar (50)	Not Null	Drop of location of vehicle
9	Drop_of_time	Varchar (50)	Not Null	Drop of time of vehicle
10	Booking_status	Int	Not Null	Status of booking
11	Booking_date	date	Date	Date of booking

5.Tbl_vehicle_category

Primary key: Category_id

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	Category_id	Int	Primary key	Primary key of user
2	Category_name	Varchar (50)	Not Null	Name of category
3	Status	Int	Not null	Status of the category
4	Image	Varchar (50)	Not null	Image of the category

6.Tbl_verify_user

Primary key: Verify_id

Foreign key: User_id references table **Tbl_user**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	Verify_id	Int	Primary key	Primary key of the table
2	User_id	Int	Foreign key	References to user table
3	Verify_status	Int	Not null	Status of verification
4	License_no	Int	Not null	Number of licenses
5	Expiry_date	Date	Not null	Expiry date if license
6	License_file	Varchar (50)	Not null	To upload license

7.Tbl_request_vehicle

Primary key request_id

Foreign key: User_id references table **Tbl_user**, Vehicle_id references table **Tbl_vehicle**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	request_id	Int	Primary key	Primary key of the table
2	Vehicle_id	Int	Foreign key	reference to vehicle table
3	User_id	Int	Foreign key	reference to user table
4	Start_date	Date	Not Null	Starting date of booking
5	End_date	Date	Not Null	End date of booking
6	Pick_up_location	Varchar (50)	Not Null	Pick up location of vehicle
7	Pick_up_time	Varchar (50)	Not Null	Pick time of vehicle
8	Drop_of_location	Varchar (50)	Not Null	Drop of location of vehicle
9	Drop_of_time	Varchar (50)	Not Null	Drop of time of vehicle
10	request_status	Int	Not Null	Status of booking
11	request_date	date	Date	Date of booking

8.Tbl_review

Primary key review_id

Foreign key: User_id references table **Tbl_user**, Vehicle_id references table **Tbl_vehicle**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	review_id	Int	Primary key	Primary key of the table
2	Accuracy	Int	Not Null	To rate accuracy
3	Cleanliness	Int	Not Null	To rate cleanliness
4	Communication	Int	Not Null	To rate Communication
5	Vehicle_Condition	Int	Not Null	To rate vehicle condition
6	review_msg	Varchar (300)	Not Null	Write a short review
7	user_id	Int	Foreign key	References to user table

8	vehicle_id	Int	Foreign key	References to vehicle table
9	review_date	timestamp	Not Null	Date of the review

9.Tbl_Payment

Primary key :payment_id

Foreign key: request_id references table **Tbl_vehicle_booking**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	payment_id	Int	Primary key	Primary key of the table
2	amount	Int	Not null	Total amount paid
3	razorpaymentid	Int	Not null	Id of payment gateway
4	request_id	Int	Foreign key	References to booking table

10.Tbl_available_time

Primary key: Time_id

Foreign key Vehicle_id references table **Tbl_vehicle**

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	Time_id	Int	Primary key	Primary key of the table
2	Available_time	Varchar (10)	NOT null	Available times
3	Vehicle_id	int	Foreign key	References to vehicle table

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is the process of running software in a controlled way, to answer the question - Does the software work as expected? Software testing is often related to the term's verification and validation. Validation is the process of checking or testing items, including software, for compliance and consistency with a related specification. Software testing is only one type of verification, which also uses methods such as reviews, analysis, inspections, and walkthroughs. Validation is the process of verifying that what has been defined is what the user really wanted.

Other activities that are often related to software testing are static analysis and dynamic analysis. Static analysis examines the source code of software, looking for issues and collecting metrics without running the code. Dynamic analysis examines the behavior of software while it is running, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be prepared in advance and performed in a systematic way. Testing starts at the module level and moves towards the integration of the whole computer-based system. Testing is essential for completing anything, as it is crucial for the system's success. There are some rules that can serve as testing objectives, based on the system testing objectives. They are: Testing is a process of running a program with the aim of finding a mistake. • A good test case is one that has a high chance of finding a hidden error. • A successful test is one that reveals a hidden error.

If testing is done successfully according to the objectives as mentioned above, it would discover mistakes in the software. Also testing shows that the software function seems to be working according to the specification, and that performance requirement seems to have been achieved. There are three ways to test program.

- For accuracy
- For implementation effectiveness
- For computational complexity

Tests for accuracy are supposed to check that a program does exactly what it was intended to do. This is much harder than it may seem at first, especially for large programs.

5.2 TEST PLAN

A test plan means a series of planned actions to be taken in performing various testing methods. The Test Plan serves as a guide for the action that is to be followed. The software engineers create a computer program, its documentation, and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove

the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the average time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test workhours per regression test all should be stated within the test plan.

The levels of testing include:

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

5.2.1 Unit Testing

Unit testing is a software testing technique that verifies the functionality and quality of individual units of code, such as functions, methods, classes, or modules. Unit testing helps developers to find and fix bugs early in the development process, as well as to ensure that the code meets the design and requirements specifications. Unit testing also facilitates code refactoring, debugging, and maintenance.

Unit testing can be done manually or automatically, using tools and frameworks that support various programming languages and platforms. Some examples of unit testing tools are JUnit for Java, NUnit for .NET, PyTest for Python, and Mocha for JavaScript. Unit testing tools typically provide features such as test runners, assertion libraries, mocking libraries, code coverage analysis, and reporting.

Unit testing best practices include:

- Writing unit tests that are independent of each other and of external dependencies, such as databases, web services, or user interfaces.
- Testing only one aspect of the code at a time, using clear and descriptive names for the test cases and methods.
- Following the Arrange-Act-Assert pattern for structuring the test cases, where arrange sets up the initial conditions and inputs, Act executes the code under test, and Assert verifies the expected outcomes and behaviors.
- Using test doubles, such as mocks, stubs, fakes, or spies, to isolate the code under test from its dependencies and to simulate different scenarios and behaviors.
- Writing unit tests before or along with the code under test, following the Test-Driven Development (TDD) or Behavior-Driven Development (BDD) methodologies.
- Running unit tests frequently and automatically, using continuous integration tools or scripts.

- Refactoring and updating unit tests as the code evolves, ensuring that they are readable, maintainable, and consistent with the current functionality and requirements.

5.2.2 Integration Testing

Integration testing is a type of software testing that aims to verify that different modules or components of a system work together correctly. It is usually performed after unit testing, which tests individual modules in isolation, and before system testing, which tests the whole system in its intended environment.

Integration testing can help to identify and fix errors that may arise when different modules interact with each other, such as data loss, interface mismatch, or functionality failure. Integration testing can also ensure that the system meets the user's expectations and requirements.

There are different approaches to integration testing, such as top-down, bottom-up, sandwich, and big bang. Each approach has its own advantages and disadvantages, depending on the complexity and structure of the system. Some of the factors that influence the choice of integration testing approach are:

- The availability and dependency of modules
- The level of test coverage and risk analysis
- The time and resources available for testing
- The testability and maintainability of the system

Integration testing can be done manually or with the help of automated tools. Manual integration testing involves executing test cases by human testers, while automated integration testing involves using software tools to run test scripts. Automated integration testing can save time and effort, improve accuracy and consistency, and facilitate regression testing. However, automated integration testing also requires more upfront investment, technical expertise, and maintenance.

Integration testing is an important part of software development because it ensures that the system works as a whole and delivers the expected functionality and quality to the users.

5.2.3 Validation Testing or System Testing

Validation testing or system testing is a type of software testing that aims to verify that the software meets the specified requirements and satisfies the needs of the end-users. Validation testing is performed after the software has passed the integration testing and before it is deployed to the production environment. Validation testing involves testing the software, using realistic scenarios and data, in an environment that simulates the actual operating conditions. Validation testing can

include functional testing, non-functional testing, usability testing, security testing, performance testing, and acceptance testing. The main objective of validation testing is to ensure that the software meets the expectations and requirements of the stakeholders and delivers the intended value and benefits.

5.2.4 Output Testing or User Acceptance Testing

Output Testing or User Acceptance Testing is a process of verifying that a software system meets the requirements and expectations of the end users. It is usually performed after the system has passed functional and integration testing, and before the system is deployed to the production environment. Output Testing or User Acceptance Testing involves testing the system with real or simulated data, scenarios, and users, to ensure that the system behaves as intended and delivers the desired outputs. Output Testing or User Acceptance Testing can be conducted by the developers, testers, clients, or end users, depending on the project scope and methodology. Output Testing or User Acceptance Testing can help to identify any defects, errors, or usability issues that might affect the user satisfaction and acceptance of the system. Output Testing or User Acceptance Testing can also provide feedback and suggestions for improving the system design and functionality.

5.2.5 Automation Testing

Automation testing is a process of using software tools to execute predefined test cases on a software application, without human intervention. Automation testing can help to improve the quality, efficiency, and reliability of software testing, by reducing the time and cost of manual testing, detecting defects earlier in the development cycle, and increasing the test coverage and accuracy. Automation testing can also support continuous integration and delivery, by enabling faster feedback and deployment of software changes.

Automation testing can be applied to different types of software testing, such as functional testing, performance testing, security testing, compatibility testing, regression testing, etc. Depending on the scope and complexity of the software application, automation testing can be performed at different levels, such as unit testing, integration testing, system testing, or acceptance testing. Automation testing can also use different approaches and techniques, such as data-driven testing, keyword-driven testing, behavior-driven testing, model-based testing, etc.

Automation testing requires the use of automation tools that can interact with the software application under test and verify the expected outcomes against the actual results. There are many automation tools available in the market, such as Selenium, Test Complete, QTP/UFT, Appium, Cucumber, etc. Each tool has its own features and limitations and can support different platforms

and technologies. Automation testers need to select the appropriate tool for their specific needs and requirements.

Automation testing also requires the design and development of automation scripts or test cases that can instruct the automation tool on how to perform the testing tasks. Automation scripts or test cases can be written in different programming languages or frameworks, such as Java, Python, Ruby, C#, etc. Automation testers need to have good programming skills and knowledge of the software application under test.

Automation testing is not a one-time activity, but a continuous process that needs to be maintained and updated throughout the software development life cycle. Automation testers need to review and modify the automation scripts or test cases whenever there are changes or enhancements in the software application under test. Automation testers also need to monitor and analyze the results of automation testing and report any issues or defects to the developers or stakeholders.

5.2.6 Selenium Testing

Selenium Testing is a process of verifying the functionality and performance of web applications using an open-source automation framework called Selenium. Selenium Testing allows testers to write test scripts in various programming languages such as Java, C#, Python, etc. and run them on different browsers and platforms. Selenium Testing has many advantages over manual testing, such as faster execution, higher coverage, lower cost, and better reliability. Selenium Testing consists of four main components: Selenium IDE, Selenium WebDriver, Selenium Grid, and Selenium RC. Selenium IDE is a browser extension that helps testers to record and playback test cases. Selenium WebDriver is a library that enables testers to interact with web elements using a common API. Selenium Grid is a tool that allows testers to run parallel tests on multiple machines and browsers. Selenium RC is a legacy component that supports older browsers and platforms.

Example:**Login Test****Code:**

```
seleniumlogin.py > ...
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.common.keys import Keys
4 from time import sleep
5
6 from colorama import init, Fore, Style
7
8 # initialize colorama
9 init()
10
11 # set color for success and failure messages
12 success_color = Fore.GREEN
13 failure_color = Fore.RED
14
15 # set up the driver
16 driver = webdriver.Chrome()
17
18 # navigate to the login page
19 driver.get("http://localhost/Autolend/login.php")
20
21 # enter the username
22 username_input = driver.find_element(By.ID, "form2Example11")
23 username_input.send_keys("aron12")
24 sleep(2)
25 # enter the password
26 password_input = driver.find_element(By.ID, "form2Example22")
27 password_input.send_keys("ARong123")
28 sleep(2)
29
30 # submit the form
31 submit_button = driver.find_element(By.ID, "subbtn")
32 submit_button.click()
33 sleep(4)
34 welcome_message = driver.find_element(By.XPATH, "//small[contains(text(),'Autolend')]").text
35 print(Fore.YELLOW + "*****LOGIN TEST*****")
36 print("*****")
37 print("*****")
38 if welcome_message == 'Autolend':
39     print(success_color + "Tested successfully")
40 else:
41     print(failure_color + "Testing failed")
42
43 # reset colorama
44 Style.RESET_ALL
45
46 # close the browser window
47 driver.quit()
```

Screenshot

```
*****
*****LOGIN TEST*****
*****
Tested successfully
```

Figure 8: Login test result

Test Report

Test Case 1					
Project Name: AutoLend					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Bibin P Daniel		
Test Priority (Low/Medium/High):			Test Designed Date: 19/04/2023		
Module Name: login			Test Executed By: Bibin		
Test Title: User login			Test Execution Date: 19/04/2023		
Description: This test checks whether a user can login into the system if use valid username and password					
Pre-Condition: User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to login page	N/A	Login page loads successfully	Login page loaded successfully	Pass
2	Fill in username and password fields	"aron12" for username and "ARon@123" for password	Username and password fields are successfully filled in	Username and password fields are successfully filled in	Pass
3	Submit the login form	N/A	User is successfully logged in and "AutoLend" welcome message is displayed	"AutoLend" welcome message is displayed	Pass
4	Close the browser window	N/A	Browser window is successfully closed	Browser window is successfully closed	Pass
Post-Condition: User has successfully logged in.					

Test Case 2: Registration

Code:

```

1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.common.keys import Keys
4 from time import sleep
5 import unittest
6 from colorama import init, Fore, Style
7
8 class LoginTest(unittest.TestCase):
9     def setUp(self):
10         self.driver = webdriver.Chrome()
11
12     def tearDown(self):
13         self.driver.quit()
14
15     def test_login(self):
16         # navigate to the registration page
17         self.driver.get("http://localhost/Autolend/registration.php")
18
19         # enter the registration details
20         username_input = self.driver.find_element(By.ID, "firstName")
21         username_input.send_keys("Nikhil")
22         sleep(3)
23         password_input = self.driver.find_element(By.ID, "lastName")
24         password_input.send_keys("Nikhil12")
25         sleep(3)
26         input = self.driver.find_element(By.ID, "DOB")
27         input.send_keys("19-02-1997")
28         sleep(3)
29         input = self.driver.find_element(By.ID, "Location")
30         input.send_keys("kollam")
31         sleep(3)
32         input = self.driver.find_element(By.ID, "emailAddress")
33         input.send_keys("nikhill12@gmail.com")
34         sleep(3)
35         input = self.driver.find_element(By.ID, "phoneNumber")
36         input.send_keys("9675453554")
37
38         input = self.driver.find_element(By.ID, "phoneNumber")
39         input.send_keys("9675453554")
40
41         input = self.driver.find_element(By.ID, "password")
42         input.send_keys("Pass@1234")
43         sleep(3)
44         input = self.driver.find_element(By.ID, "Cpassword")
45         input.send_keys("Pass@1234")
46         sleep(3)
47         input = self.driver.find_element(By.ID, "inputfileupload")
48         input.send_keys("D:/ADCE/S6/Mini Project/Autolend/Images/users/aatik-tasneem")
49         sleep(3)
50
51         # submit the registration form
52         submit_button = self.driver.find_element(By.ID, "btn")
53         submit_button.click()
54         print(Fore.YELLOW + "*****REGISTRATION TEST*****")
55         print("*****REGISTRATION TEST*****")
56         print("*****REGISTRATION TEST*****")
57         # check if registration was successful
58         if self.driver.current_url == "http://localhost/Autolend/login.php":
59             print(Fore.GREEN + "Registration successful!")
60         else:
61             print(Fore.RED + "Registration failed!")
62
63 if __name__ == '__main__':
64     unittest.main()
65

```

Screenshot

```

*****
*****REGISTRATION TEST*****
*****
Registration successful!
.
-----
Ran 1 test in 37.171s
OK

```

Figure 9: Registration test result

Test report

Test Case 2					
Project Name: AutoLend					
Registration Test Case					
Test Case ID: Test_2			Test Designed By: Bibin P Daniel		
Test Priority (Low/Medium/High):			Test Designed Date: 19/04/2023		
Module Name: Registration			Test Executed By: Bibin		
Test Title: User registration			Test Execution Date: 19/04/2023		
Description: This test whether a user can successfully register if he uses valid details					
Pre-Condition: Filling valid registration fields					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to registration page	http://localhost/Autolend/registration.php	Registration page should be loaded	Registration page was loaded	Pass
2	Enter first name	"Nikhil"	The first name field should be filled with "Nikhil"	First name field was filled with "Nikhil"	Pass
3	Enter last name	"Nikhil12"	The last name field should be filled with "Nikhil12"	Last name field was filled with "Nikhil12"	Pass
4	Enter date of birth	"19-02-1997"	The date of birth field should be filled with "19-02-1997"	Date of birth field was filled with "19-02-1997"	Pass
5	Enter location	"Kollam"	The location field should be filled with "Kollam"	Location field was filled with "Kollam"	Pass
6	Enter email address	" nikhil12@gmail.com "	The email address field should be filled with " nikhil12@gmail.com "	Email address field was filled with " nikhil12@gmail.com "	Pass
7	Enter phone number	"9675453554"	The phone number field should be filled with "9675453554"	Phone number field was filled with "9675453554"	Pass

8	Enter password	"Pass@1234"	The password field should be filled with "Pass@1234"	Password field was filled with "Pass@1234"	Pass
9	Confirm password	"Pass@1234"	The confirm password field should be filled with "Pass@1234"	Confirm password field was filled with "Pass@1234"	Pass
10	Upload profile picture	"D:/AJCE/S6/Mini Project/AutoLend/Images/users/aatik-tasneem-7omHUGghmZ0-unsplash.jpg"	The profile picture should be uploaded successfully	Profile picture was uploaded successfully	Pass
11	Click on the register button		If registration is successful, user should be redirected to login page	User was redirected to login page	Pass
Post-Condition: Registration successfully completed					

Test Case 3: Profile Updating

Code:

```

selenium.userprofile.py > ...
1  from selenium import webdriver
2  from selenium.webdriver.common.by import By
3  from selenium.webdriver.common.keys import Keys
4  from time import sleep
5
6  from colorama import init, Fore, Style
7
8  # initialize colorama
9  init()
10
11 # set color for success and failure messages
12 success_color = Fore.GREEN
13 failure_color = Fore.RED
14
15 # set up the driver
16 driver = webdriver.Chrome()
17
18 # navigate to the login page
19 driver.get("http://localhost/Autolend/login.php")
20
21 # enter the username
22 username_input = driver.find_element(By.ID, "form2Example11")
23 username_input.send_keys("user12")
24 sleep(2)
25 # enter the password
26 password_input = driver.find_element(By.ID, "form2Example22")
27 password_input.send_keys("User@1234")
28
29 # submit the form
30 submit_button = driver.find_element(By.ID, "subbtn")
31 submit_button.click()
32 sleep(2)
33 driver.get("http://localhost/Autolend/userprofileedit.php")
34
35 sleep(4)
36 username_input = driver.find_element(By.ID, "firstName")
37 username_input.clear()
38 username_input.send_keys("user")
39 sleep(2)
38  username_input.send_keys("user")
39  sleep(2)
40
41  submit_button = driver.find_element(By.ID, "btn")
42  submit_button.click()
43  sleep(3)
44
45  element = driver.find_element(By.ID, "firstName")
46  value = element.get_attribute("value")
47  print(value)
48  sleep(3)
49
50  print(Fore.YELLOW + "*****")
51  print("*****PROFILE TEST*****")
52  print("*****")
53  if value == "user" :
54      print(success_color + "Tested successfully")
55  else:
56      print(failure_color + "Testing failed")
57
58
59 # reset colorama
60 Style.RESET_ALL
61
62 # close the browser window
63 driver.quit()

```

Screenshot

```

*****
*****PROFILE TEST*****
*****
Tested successfully
PS D:\AJCE\S6\Open Source>

```

Figure 10: Profile updating test result.

Test report

Test Case 3					
Project Name: AutoLend					
User profile edit Test Case					
Test Case ID: Test_3			Test Designed By: Bibin P Daniel		
Test Priority (Low/Medium/High):			Test Designed Date: 19/04/2023		
Module Name: User profile edit			Test Executed By: Bibin		
Test Title: Update profile			Test Execution Date: 19/04/2023		
Description: This test aim to change the name of user in profile updating page					
Pre-Condition: Changing the name in user profile page					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to the login page	http://localhost/Autolend/login.php	The login page should be displayed	Login page was loaded	Pass
2	Enter the username	user12	The username field should be filled with "user12"	Username entered	Pass
3	Enter the password	USer@1234	The password field should be filled with "USer@1234"	Password was entered	Pass
4	Click on the submit button		The user should be logged in and redirected to their profile edit page	Page redirected to profile edit page	Pass

5	Edit the first name field	"user"	The first name field should be updated with "user"	Name field changed	Pass
6	Click on the save button		The updated profile information should be saved	Button clicked	Pass
7	Retrieve the updated first name		The updated first name should be displayed on the page as "user"	Retrieved the name	Pass
8	Compare the expected and actual results	Expected: "user" Actual: "user"	The test should pass	The expected and actual was same	Pass
Post-Condition: Successfully updated the name					

Test Case 4: changing category status

Code:

```

seleniumreg.py > ...
1 from selenium import webdriver
2 from selenium.webdriver.common.by import By
3 from selenium.webdriver.common.keys import Keys
4 from time import sleep
5 from selenium.webdriver.common.action_chains import ActionChains
6
7 from colorama import init, Fore, Style
8
9 # initialize colorama
10 init()
11
12 # set color for success and failure messages
13 success_color = Fore.GREEN
14 failure_color = Fore.RED
15
16 driver = webdriver.Chrome()
17
18 # navigate to the login page
19 driver.get("http://localhost/Autolend/login.php")
20
21 # enter the username
22 username_input = driver.find_element(By.ID, "form2Example11")
23 username_input.send_keys("admin")
24 sleep(3)
25 # enter the password
26 password_input = driver.find_element(By.ID, "form2Example22")
27 password_input.send_keys("admin")
28 sleep(2)
29 # submit the form
30 submit_button = driver.find_element(By.ID, "subbtn")
31 submit_button.click()
32
33 # wait for the new page to load
34 driver.implicitly_wait(10)
35
36 driver.get("http://localhost/Autolend/admin-add-cat.php")
37
38 rows = driver.find_elements(By.XPATH, "//*[@id='mytable']/tbody/tr")
39
seleniumreg.py > ...
38 rows = driver.find_elements(By.XPATH, "//*[@id='mytable']/tbody/tr")
39
40 # iterate over each row and click on the toggle-status link
41 for row in rows:
42     sleep(2)
43     toggle_link = row.find_element(By.CSS_SELECTOR, "a.toggle-status")
44     toggle_link.click()
45
46 for row in rows:
47     sleep(2)
48     status_span = row.find_element(By.CSS_SELECTOR, "span.user-status")
49     assert status_span.text == "Blocked"
50
51 for row in rows:
52     sleep(2)
53     toggle_link = row.find_element(By.CSS_SELECTOR, "a.toggle-status")
54     toggle_link.click()
55
56 for row in rows:
57     sleep(2)
58     status_span = row.find_element(By.CSS_SELECTOR, "span.user-status")
59     if status_span.text == "Active":
60         flag=1
61     else:
62         flag=0
63
64 print(Fore.YELLOW + "*****TEST*****")
65 print("*****")
66 if flag == 1:
67     print(success_color + "Tested successfully")
68 else:
69     print(failure_color + "Testing failed")
70
71 # wait for the page to load after clicking the link
72 driver.implicitly_wait(10)
73 # wait for the page to load after clicking the links
74 driver.implicitly_wait(10)
75 sleep(5)
76 # close the browser window
77 driver.quit()

```

Screenshot

```

error_code_1; net_error_101
*****
*****TEST*****
*****
Tested successfully
PS D:\AJCE\S6\Open Source> INFO: Created TensorFlow Lite XNNPACK delegate for CPU.

```

Figure 11: toggle status test result

Test report

Test Case 4					
Project Name: AutoLend					
changing category status Test Case					
Test Case ID: Test_4			Test Designed By: Bibin P Daniel		
Test Priority (Low/Medium/High):			Test Designed Date: 19/04/2023		
Module Name: changing category status			Test Executed By: Bibin		
Test Title: Toggling category status			Test Execution Date: 19/04/2023		
Description: This test toggles the status vehicle category					
Pre-Condition: Toggling the category status					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigate to the login page	N/A	Login page is loaded	Login page is loaded	Pass
2	Enter the username	admin	Username is entered	Username is entered	Pass
3	Enter the password	admin	Password is entered	Password is entered	Pass
4	Submit the form	N/A	Dashboard is loaded	Dashboard is loaded	Pass
5	Navigate to admin-add-cat page	N/A	Page is loaded	Page is loaded	Pass
6	Click on toggle-status link for each row	N/A	Rows are toggled	Rows are toggled	Pass
7	Verify status of each row is "Blocked"	N/A	Status is "Blocked"	Status is "Blocked"	Pass

8	Click on toggle-status link for each row	N/A	Rows are toggled	Rows are toggled	Pass
9	Verify status of each row is "Active"	N/A	Status is "Active"	Status is "Active"	Pas
Post-Condition: All status toggled sucessfully					

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

The implementation of a project involves the actual development and deployment of the proposed system. It is a crucial phase where the proposed design and specifications are transformed into an operational system. The implementation process requires a proper plan, coordination, and execution of different tasks involved in developing the system.

In the case of the AutoLend project, the implementation process involves developing the software application, creating the database, and integrating the system with hardware components. The implementation phase follows the design and testing phases and involves the actual coding and configuration of the system.

During the implementation process, the developers need to ensure that the system is developed according to the proposed specifications and design. They also need to test the system for any bugs or issues that may arise during the development process. Once the software is developed, it needs to be deployed on the target hardware and tested thoroughly before it can be made available to end-users.

The implementation of the AutoLend project involves creating a user-friendly and secure platform for users to search and book vehicles. The implementation phase will involve the development of the user interface, the database schema, and the integration of the payment gateway.

Overall, the implementation phase is a critical stage in the software development life cycle, and proper planning, execution, and testing are essential to ensure the successful deployment of the system.

6.2 IMPLEMENTATION PROCEDURES

Installation of Hardware: The system hardware specifications, including the processor, RAM, and hard disk, will be installed to ensure that they meet the required standards for the proper functioning of the system.

Installation of Software: The necessary software, including the MySQL database and programming languages such as HTML5, AJAX, J Query, PHP, and CSS, will be installed to support the functionality of the AutoLend system.

System Configuration: After installation, the system will be configured to ensure that all the components are correctly integrated and working correctly. This process will include setting up the database, creating user accounts, and configuring the system settings.

Testing: The system will undergo rigorous testing to ensure that it is functioning correctly and that all the features are working as intended. Testing will include unit testing, integration testing, system testing, and user acceptance testing.

Deployment: Once the system has been thoroughly tested, it will be deployed in its actual environment. The system will be made available to users, and proper guidance will be provided to ensure that they are comfortable using the application.

Maintenance: Finally, regular maintenance of the system will be carried out to ensure that it continues to function correctly and that any issues that arise are resolved promptly. This process will include regular backups, updates, and bug fixes.

By following these implementation procedures, the AutoLend system will be successfully implemented, and users will be able to use the application to rent vehicles easily and conveniently.

6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people, who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database, and call up routine that will produce reports and perform other necessary functions.

6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

6.2.3 System Maintenance

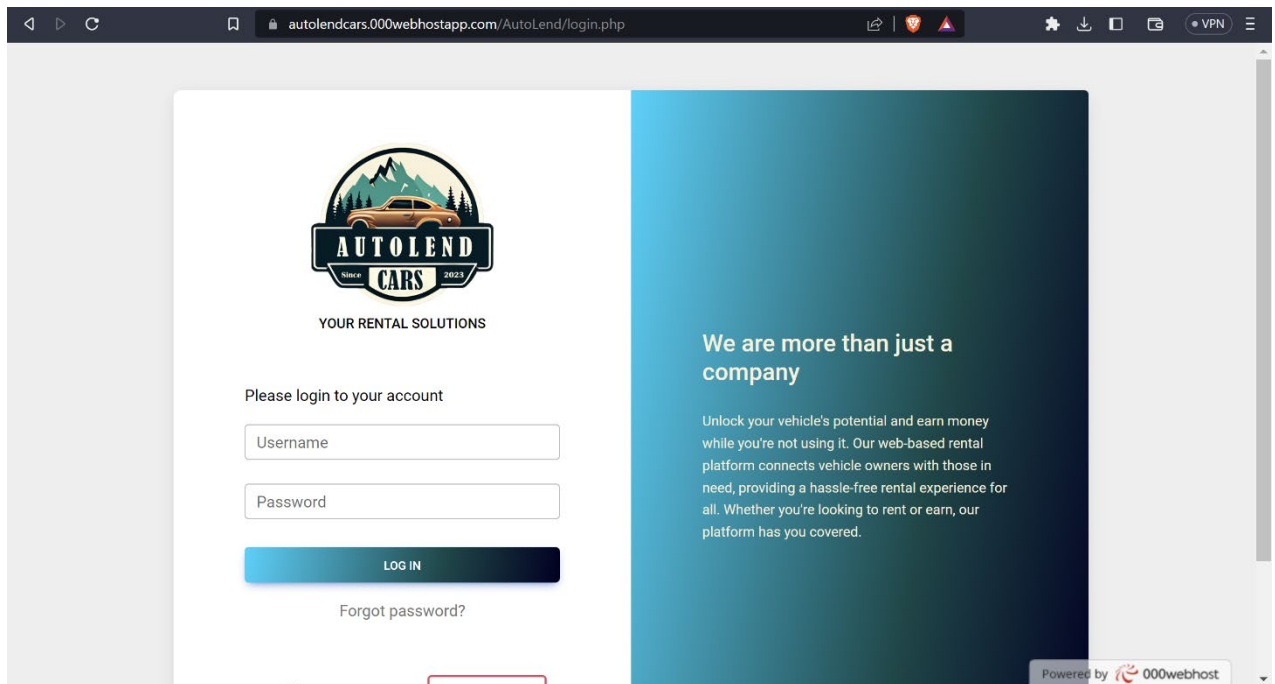
Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is Labor Management System 54 Amal Jyothi College of Engineering, Kanjirappally Department of Computer Applications an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

6.2.4 Hosting

Web hosting is a service that allows individuals and organizations to make their websites accessible over the internet. In simple terms, a web hosting service provides the necessary infrastructure, software, and technologies to store and serve websites to internet users.

The following are the steps involved in hosting a website:

- **Choose a web hosting provider:** There are many web hosting providers available, each with different plans and features. Choose a provider that offers the features and resources that meet your website's needs, such as bandwidth, storage, and support.
- **Select a hosting plan:** Most web hosting providers offer a range of hosting plans, including shared hosting, VPS hosting, and dedicated hosting. Select a plan that suits your budget and website requirements.
- **Choose a domain name:** A domain name is the address that users type in their web browser to access your website. Choose a unique and memorable domain name that reflects your brand or website content.
- **Configure your website:** Depending on your hosting plan, you may need to install and configure your website's software, such as WordPress or Joomla. You can also upload your website files using an FTP client or the hosting provider's file manager.
- **Test and launch your website:** Before launching your website, test it thoroughly to ensure that it functions correctly and is optimized for speed and usability. Once you are satisfied with your website's performance, launch it and make it accessible to the public.
- **Maintain and update your website:** Regularly maintain and update your website to ensure that it remains secure, performs well, and meets your users' needs. You may need to install updates, backup your website regularly, and optimize your website's content and structure to improve its search engine rankings.

Screenshot:**Figure 12:** screenshot of hosted projected

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In conclusion, AutoLend is an online car rental system that provides a user-friendly platform for car owners and renters to interact seamlessly. The proposed system offers several advantages over the existing system, such as improved security, better accessibility, and more extensive features. The system specification includes hardware and software specifications that are necessary for the smooth running of the system. The software specification includes the use of HTML, CSS, MySQL, JS, PHP, AJAX, and J Query technologies. The implementation procedures include ensuring that the active user is aware of the benefits of using the new system and imparting proper guidance to them to ensure comfort while using the application.

Overall, AutoLend is a comprehensive solution for car owners and renters seeking a reliable platform to interact, rent and lend cars. The project offers several benefits to its users and has the potential to revolutionize the car rental industry. The economic and behavioral feasibility analysis shows that the project is economically and behaviorally viable, and with the proper implementation procedures, it is bound to succeed. Therefore, AutoLend is a valuable addition to the car rental market, and its success is highly anticipated.

7.2 FUTURE SCOPE

The AutoLend project has great potential for future enhancements and expansions. Here are some possible areas of improvement and development:

Mobile application: With the increasing use of smartphones, developing a mobile application can increase the accessibility and convenience of the AutoLend platform.

Integration with third-party services: Integration with third-party services like Google Maps, payment gateways, and social media platforms can provide additional functionalities and enhance user experience.

Machine learning and data analytics: Implementing machine learning and data analytics techniques can provide insights into user behavior, booking patterns, and customer preferences. This can help improve the system's efficiency and decision-making capabilities.

Multi-language support: Incorporating multi-language support can make the platform more accessible and user-friendly for a wider audience.

Fleet management system: Developing a fleet management system can allow car owners to manage their vehicles more efficiently and track their performance and maintenance.

By implementing these enhancements, AutoLend can become a more comprehensive and advanced car-sharing platform, providing users with a seamless experience and greater convenience.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009.
- Roger S Pressman, “Software Engineering”, 1994.
- PankajJalote, “Software engineering: a precise approach”, 2006.
- James lee and Brent ware Addison, “Open source web development with LAMP”,2003
- The Complete reference PHP by Steven Holzner
- The Complete reference MySQL by Vikram Vaswani
- CSS Cookbook byChristopher Schmitt

WEBSITES:

- www.w3schools.com
- www.geeksforgeeks.com
- www.tutorialspoint.org
- <https://getbootstrap.com>

CHAPTER 9

APPENDIX

9.1 Sample Code

Admin view and verify users:

```
<!DOCTYPE html>
<html lang="en">
<?php
session_start();
if ($_SESSION['logout'] == "") {
    header("location:login.php");
}
?>

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Autolend</title>
    <link rel="icon" href="Images/Logo.png">
    <!-- Font Awesome -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css"
rel="stylesheet" />
    <!-- Google Fonts -->
    <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700&display=swap"
rel="stylesheet" />
    <!-- MDB -->
    <link href="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/6.1.0/mdb.min.css"
rel="stylesheet" />
    <!-- jquery cdn -->
    <script src="https://code.jquery.com/jquery-3.6.3.slim.min.js" integrity="sha256-
ZwqZIVdD3iXNyGHbSYdsmWP//UBokj2FHAKuSBKDSO="
crossorigin="anonymous"></script>
    <!-- ajax -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js"></script>
    <link rel="stylesheet" href="adminpanel.css">
    <!-- data table -->
    <link rel="stylesheet" type="text/css"
href="https://cdn.datatables.net/1.13.2/css/jquery.dataTables.css">
</head>
<style>
.gradient-custom {
    background: rgb(2, 0, 36);
    background: linear-gradient(280deg, rgba(2, 0, 36, 1) 0%, rgba(14, 72, 73, 1) 37%, rgba(0,
212, 255, 1) 100%);
}
</style>

<body>
<script>
$(document).ready(function() {
    $(document).on('click', "#btn",function() {
```

```

    $('#exampleModal').modal('toggle')
  })
  var table = $('#mytable').DataTable({
    "lengthChange": false,
    pageLength: 6,
    lengthMenu: [
      [5, 10, 20, -1],
      [5, 10, 20, 'Todos']
    ]
  })
  $(document).ready(function() {
    // toggle user status
    $(document).on('click', '.toggle-status1', function(e) {
      e.preventDefault();
      var userId = $(this).data('user-id');
      var statusElm = $('.user-v-status[data-user-id="' + userId + '"]');
      var status = statusElm.text();
      // update user status and icon
      if (status === 'verified') {
        statusElm.text('Not verified').removeClass('badge-success').addClass('badge-
danger');
        $(this).html('<i class="fa fa-times text-danger"></i>');
      } else if (status === 'verification Pending') {
        statusElm.text('verified').removeClass('badge-danger').addClass('badge-success');
        $(this).html('<i class="fa fa-check text-success"></i>');
      } else {
        $(this).attr('data-mdb-toggle', 'popover');
        $(this).attr('data-mdb-container', 'body');
        $(this).attr('title', '');
        $(this).attr('data-mdb-content', 'Details Not Updated');
        $(this).popover({
          trigger: 'hover'
        });
        $(this).popover('show');
      }

      // send ajax request to update user status in database
      $.ajax({
        url: 'update_status1.php',
        type: 'POST',
        data: {
          user_id: userId,
          status: statusElm.text()
        },
        success: function(response) {
          console.log(response);
        }
      });
    });
  });
  $(document).ready(function() {

```

```

// toggle user status
$(document).on('click', '.toggle-status', function(e) {
    e.preventDefault();
    var userId = $(this).data('user-id');
    var statusElm = $('<div>user-status[data-user-id="' + userId + '"]</div>');
    var status = statusElm.text();

    // update user status and icon
    if (status === 'Active') {
        statusElm.text('Blocked').removeClass('badge-success').addClass('badge-danger');
        $(this).html('<i class="fa fa-times text-danger"></i>');
    } else if (status === 'Blocked') {
        statusElm.text('Active').removeClass('badge-danger').addClass('badge-success');
        $(this).html('<i class="fa fa-check text-success"></i>');
    }

    // send ajax request to update user status in database
    $.ajax({
        url: 'update_status.php',
        type: 'POST',
        data: {
            user_id: userId,
            status: statusElm.text()
        },
        success: function(response) {
            console.log(response);
        }
    });
});

$(document).on('click', '.view-btn', function(e) {
    e.preventDefault();
    var userId = $(this).data('user-id');
    console.log("Triggering")
    $.ajax({
        url: 'retrieve_data.php',
        type: 'POST',
        data: {
            id: userId
        },
        dataType: 'json',
        success: function(data) {
            console.log("data: ", data)
            var name = '<p>' + data.value1 + '</p>';
            var email = '<p>' + data.value2 + '</p>';
            var mob = '<p>' + data.value3 + '</p>';
            var dob = '<p>' + data.value4 + '</p>';
            var image = data.value5;
            var lno = '<p>' + data.value6 + '</p>';
            var exdate = '<p>' + data.value7 + '</p>';
            var loc = '<p>' + data.value8 + '</p>';

```

```

        var file = data.value9;
        var status = data.status;
        var id = data.id;
        $('#modal-name').html(name);
        $('#modal-email').html(email);
        $('#modal-mob').html(mob);
        $('#modal-dob').html(dob);
        $('#modal-lno').html(lno);
        $('#modal-exdate').html(exdate);
        $('#modal-loc').html(loc);
        $('#modal-image').attr('src', 'Uploads/' + image);
        if (status == '-1') {
            $('.modal-file1').hide()
            $('.modal-file2').show()
        } else {
            $('.modal-file1').show()
            $('.modal-file2').hide()
            $('.modal-file1').attr('href', 'Licence/' + file);
        }
        if (status == '-1') {
            $('#verify_btn').show()
            $('#verify_btn1').hide()
            $('#verify_btn2').hide()
        } else if (status == '0') {
            $('#verify_btn').hide()
            $('#verify_btn1').show()
            $('#verify_btn2').hide()
            $('#verify_btn1').attr('data-user-id', id)
        }
        } else {
            $('#verify_btn').hide()
            $('#verify_btn1').hide()
            $('#verify_btn2').show()
            $('#verify_btn2').attr('data-user-id', id)
        }
    },
    error: function(jqXHR, textStatus, errorThrown) {
        console.log("Catch", textStatus, errorThrown);
    }
});
});
});
</script>
<!--Main Navigation-->
<?php
$tmp_id = $_SESSION['id'];
include 'dbconnect.php';
$query = "SELECT `image` FROM `tbl_user` WHERE `login_id`='".$tmp_id'";
$result = mysqli_query($con, $query);
$row = mysqli_fetch_array($result);

```

```

$img = $row['image'];
?>
<?php
include("adminsidebar.php");
?>
<!--Main Navigation-->

<!--Main layout-->
<main style="margin-top: 58px">
  <div class="container pt-4">
    <table class="table align-middle mb-0 bg-white table-hover" id="mytable">
      <thead class="bg-light">
        <tr>
          <th>Name</th>
          <th>Verification</th>
          <th>Action</th>
          <th>Status</th>
          <th>Action</th>
          <th>View</th>
        </tr>
      </thead>
      <tbody>
        <?php
        $query = "SELECT * FROM `tbl_user` where user_type =1";
        $result = mysqli_query($con, $query);
        while ($row = mysqli_fetch_array($result)) {
          ?>
          <tr>
            <td>
              <div class="d-flex align-items-center">
                
                <div class="ms-3">
                  <p class="fw-bold mb-1"><?php echo $row['first_name'] ?></p>
                  <p class="text-muted mb-0"><?php echo $row['email'] ?></p>
                </div>
              </div>
            </td>
            <!-- <td>
              <span class="badge badge-success rounded-pill d-inline">verified</span>
            </td> -->
            <?php $id = $row['user_id'];
            $query1 = "SELECT `verify_status` FROM `tbl_verify_user` where `user_id`
=id";

            $result1 = mysqli_query($con, $query1);
            $row1 = mysqli_fetch_array($result1);
            if ($row1['verify_status'] == -1) {
              ?>
              <td>
                <span class="user-v-status badge badge-danger rounded-pill d-inline" data-
user-id="<?php echo $row['user_id']; ?>">Not verified</span>

```



```

        <a href="#"><i class="fa fa-info-circle"></i></a>
    </td>
    <?php
    } elseif ($row1['verify_status'] == 0) {
    ?>
    <td>
        <span class="user-v-status badge badge-primary rounded-pill d-inline" data-
user-id="<?php echo $row['user_id']; ?>">verification Pending</span>
        <a href="#"><i class="fa fa-info-circle"></i></a>
    </td>
    <?php
    } elseif ($row1['verify_status'] == 1) {
    ?>
    <td>
        <span class="user-v-status badge badge-success rounded-pill d-inline" data-
user-id="<?php echo $row['user_id']; ?>">verified</span>
        <a href="#"><i class="fa fa-info-circle"></i></a>
    </td>
    <?php
    }
    ?>
    <td>
        <a href="#" class="toggle-status1" title="Toggle Status" data-user-id="<?php
echo $row['user_id']; ?>" data-toggle="tooltip">
        <?php if ($row1['verify_status'] == -1) { ?>
        <i class="fa fa-times text-danger "></i>
        <?php } elseif ($row1['verify_status'] == 0) { ?>
        <i class="fa fa-times text-danger"></i>
        <?php } else { ?>
        <i class="fa fa-check text-success"></i>
        <?php } ?>
        </a>
    </td>
    <td>
        <?php if ($row['user_status'] == 1) { ?>
        <span class="user-status badge badge-success rounded-pill d-inline" data-
user-id="<?php echo $row['user_id']; ?>">Active</span>
        <?php } elseif ($row['user_status'] == 0) { ?>
        <span class="user-status badge badge-danger rounded-pill d-inline" data-
user-id="<?php echo $row['user_id']; ?>">Blocked</span>
        <?php } ?>
    </td>
    <td>
        <a href="#" class="toggle-status" title="Toggle Status" data-user-id="<?php
echo $row['user_id']; ?>" data-toggle="tooltip">
        <?php if ($row['user_status'] == 1) { ?>
        <i class="fa fa-check text-success"></i>
        <?php } elseif ($row['user_status'] == 0) { ?>
        <i class="fa fa-times text-danger"></i>
        <?php } ?>
        </a>

```

```

        </td>
        <td> <button type="button" class="btn btn-info view-btn" data-mdb-
toggle="modal" data-mdb-target="#exampleModal" data-user-id="<?php echo $row['user_id'];
?>">
            View
        </button></td>
    </tr>
<?php } ?>
</tbody>
</table>
</div>
</main>
<!-- Modal -->
    <div class="modal fade" id="exampleModal" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
        <div class="modal-dialog modal-dialog-scrollable">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="exampleModalLabel">User Profile</h5>
                    <button type="button" class="btn-close" data-mdb-dismiss="modal" aria-
label="Close"></button>
                </div>
                <div class="modal-body">
                    <div class="row g-0">
                        <div class="col-md-12 gradient-custom text-center text-white d-flex" style="border-
top-left-radius: .5rem; border-bottom-left-radius: .5rem;">
                            <div class="my-auto mx-auto">
                                
                                <h5 id="modal-name"></h5>
                            </div>
                            <p id="modal-uname"></p>
                        </div>
                    <div class="col-md-12">
                        <div class="card-body p-4">
                            <h6>Information</h6>
                            <hr class="mt-0 mb-4">
                            <div class="row pt-1">
                                <div class="col-6 mb-3">
                                    <h6>Email</h6>
                                    <p class="text-muted" id="modal-email"></p>
                                </div>
                                <div class="col-6 mb-3">
                                    <h6>Phone</h6>
                                    <p class="text-muted" id="modal-mob"></p>
                                </div>
                                <div class="col-6 mb-3">
                                    <h6>DOB</h6>
                                    <p class="text-muted" id="modal-dob"></p>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

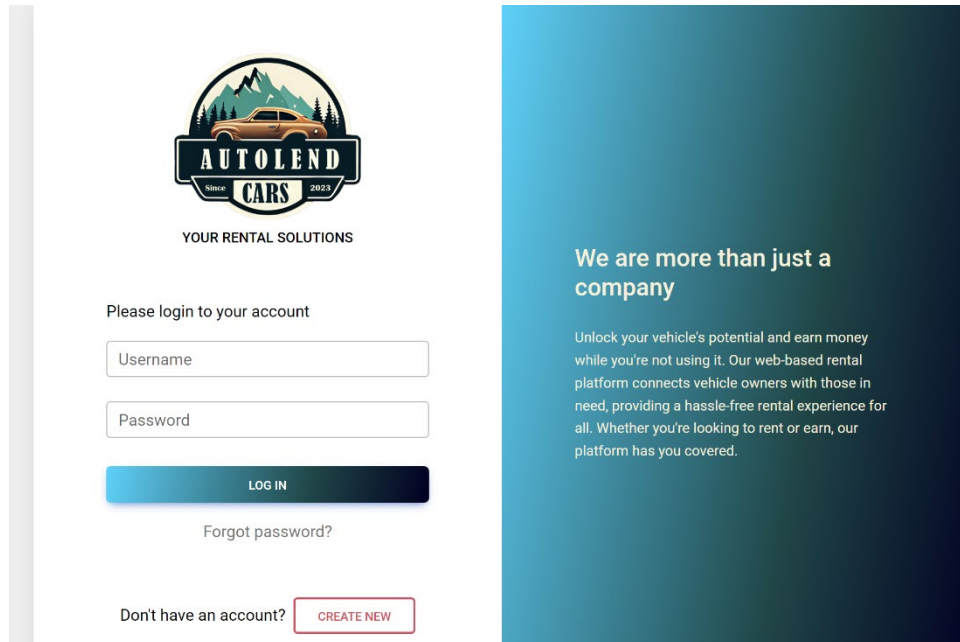
```

        <div class="col-6 mb-3">
            <h6>Location</h6>
            <p class="text-muted" id="modal-loc"></p>
        </div>
        <div class="col-6 mb-3">
            <h6>Licence No</h6>
            <p class="text-muted" id="modal-lno"></p>
        </div>
        <div class="col-6 mb-3">
            <h6>Expiry Date</h6>
            <p class="text-muted" id="modal-exdate"></p>
        </div>
        <div class="col-12 mb-3">
            <div class="d-flex justify-content-center">
                <a class="modal-file1" href="#" target="_blank" data-mdb-ripple-
color="dark"><i class="fas fa-address-card fa-5x"></i></a>
                <a class="modal-file2" data-mdb-toggle="popover" data-mdb-
placement="top" data-mdb-content="licence not Uploaded" data-mdb-ripple-color="dark"><i
class="fas fa-address-card fa-5x"></i></a>
            </div>
        </div>
    </div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
<!-- data table -->
        <script type="text/javascript" charset="utf8"
src="https://cdn.datatables.net/1.13.2/js/jquery.dataTables.js"></script>
        <!--Main layout-->
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwXhTBTkF7CXvN"
crossorigin="anonymous"></script>
    </body>
    <script src="adminpanel.js"></script>
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-
kit/6.1.0/mdb.min.js"></script>
</html>

```

9.1 ScreenShots

Login Page:



The screenshot shows the AutoLend login page. On the left, there is a vertical grey bar. The main content area features the AutoLend logo (a car in a mountain landscape) with the text "YOUR RENTAL SOLUTIONS" below it. A login form is centered, asking the user to "Please login to your account". It includes input fields for "Username" and "Password", a "LOG IN" button, and a link for "Forgot password?". Below the login form is a link for "Don't have an account?" and a "CREATE NEW" button. On the right, there is a blue gradient box with the text "We are more than just a company" and a paragraph describing the platform's mission to connect vehicle owners with renters.

AUTOLEND CARS
Since 2020

YOUR RENTAL SOLUTIONS

Please login to your account

Username

Password

LOG IN

Forgot password?

Don't have an account? [CREATE NEW](#)

We are more than just a company

Unlock your vehicle's potential and earn money while you're not using it. Our web-based rental platform connects vehicle owners with those in need, providing a hassle-free rental experience for all. Whether you're looking to rent or earn, our platform has you covered.

Figure 13:Login Page

Search Cars:

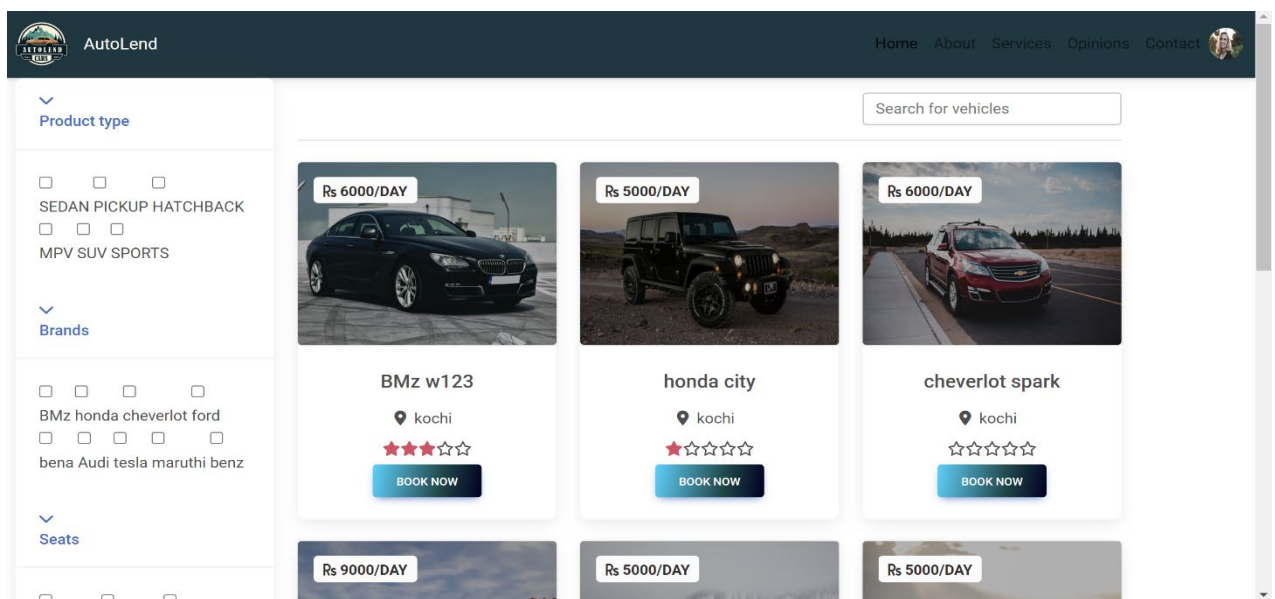
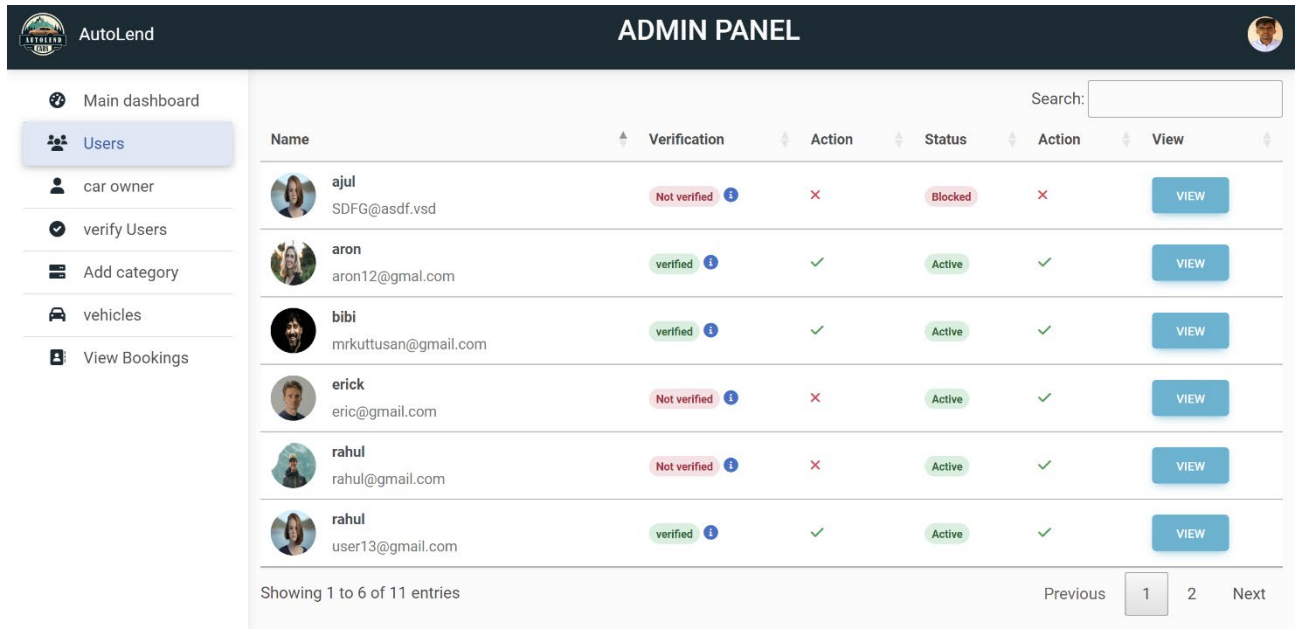


























Figure 14: Product Display Page

Admin view and verify user:

Name	Verification	Action	Status	Action	View
 ajul SDFG@asdf.vsd	Not verified 		Blocked		VIEW
 aron aron12@gmail.com	verified 		Active		VIEW
 bibi mrkuttusan@gmail.com	verified 		Active		VIEW
 erick eric@gmail.com	Not verified 		Active		VIEW
 rahul rahul@gmail.com	Not verified 		Active		VIEW
 rahul user13@gmail.com	verified 		Active		VIEW

Showing 1 to 6 of 11 entries

Previous 1 2 Next

Figure 15: Admin Toggle Status Page