AUTOMATE HUB

Mini Project Report

Submitted by

KEERTHI U

Reg. No.: AJC22MCA-2055

In Partial fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS

(MCA TWO YEAR)

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]

2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Project report, "**AUTOMATE HUB"** is the bona fide work of **KEERTHI U(Regno:AJC22MCA-2055)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Rini Kurian

Internal Guide

Ms. Meera Rose Mathew

Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department

DECLARATION

I hereby declare that the project report **“AUTOMATE HUB”** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: **KEERTHI U**

KANJIRAPPALLY **Reg: AJC22MCA-2055**

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 whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for his 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. Rini Kurian** 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.

KEERTHI U

ABSTRACT

The current system for procuring commercial vehicle spare parts and booking service appointments involves customers directly visiting the company's physical location to purchase parts and schedule vehicle maintenance. This process lacks convenience and accessibility for customers seeking an efficient solution.

This project aims to revolutionize the automotive industry by introducing an online platform designed to facilitate the seamless purchase of spare parts and the hassle-free booking of vehicle services. The website will provide a user-friendly interface that allows customers to effortlessly browse, select, and purchase the required spare parts with ease.

CONTENT

List of Abbreviation

IDE - Integrated Development Environment

HTML - Hyper Text Markup Language.

CSS - Cascading Style Sheet

SQLite - Structured Query Language

UML - Unified Modeling Language

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

This project aims to provide a one-stop solution for the purchase, maintenance, and protection of commercial vehicles. The platform will offer a wide range of services, including the sale of new commercial vehicles, parts delivery .To offer efficient and reliable parts delivery services, reducing vehicle downtime and maintenance costs for our customers.This project stands to revolutionize the automotive industry by leveraging an innovative online platform designed explicitly for the seamless purchase of commercial vehicle spare parts and hassle-free service booking. The envisioned website will boast a user-friendly interface, empowering customers to effortlessly navigate, select, and procure the necessary spare parts with utmost convenience.

1.2 PROJECT SPECIFICATION

Traditional methods of procuring commercial vehicle spare parts and booking service appointments have been inconvenient for customers, requiring physical visits to company locations. This project endeavors to transform the automotive industry by introducing an online platform. This platform aims to streamline the purchase of spare parts and simplify the process of scheduling vehicle services. The proposed website will offer a user-friendly interface, enabling customers to effortlessly browse, select, and purchase the required spare parts, thereby enhancing convenience and accessibility.

Admin Module Functionalities:

The admin module acts as the overseer, responsible for monitoring and managing user profiles, service branch information, and staff details. This includes the ability to add new branches and manage associated staff members.

User Module Functionalities:

Users, on the other hand, engage with the platform by registering, browsing spare parts and services, managing their profiles, and facilitating transactions for spare parts purchase and vehicle service bookings.

Service Branch Module Functionalities:

Service branches are equipped with functionalities to oversee service bookings, providing a dashboard to view all appointments and the ability to confirm or reject bookings based on various criteria like availability or service specifics.

Parts Manager Module Functionalities:

The parts manager module handles the inventory of spare parts, allowing for the addition of new parts and the maintenance of existing product details to ensure an up-to-date inventory list.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

The current system for procuring commercial vehicle spare parts and booking service appointments involves customers directly visiting the company's physical location to purchase parts and schedule vehicle maintenance. This process lacks convenience and accessibility for customers seeking an efficient solution.This project aims to revolutionize the automotive industry by introducing an online platform designed to facilitate the seamless purchase of spare parts and the hassle-free booking of vehicle services. The website will provide a user-friendly interface that allows customers to effortlessly browse, select, and purchase the required spare parts with ease.

2.2 EXISTING SYSTEM

The current system for commercial vehicle spare parts procurement and service booking relies on physical visits to the company's brick-and-mortar locations. Customers are required to personally attend these locations to purchase spare parts and schedule vehicle maintenance. This traditional approach results in inconvenience and limited accessibility, hindering customers seeking efficient solutions. The system lacks the convenience of remote access and online transactions, leading to potential time constraints and geographical limitations for customers.

2.2.1 NATURAL SYSTEM STUDIED

The natural system studied encompasses the broader environment and factors impacting the automotive industry and its customers. It involves an analysis of consumer behavior, market trends, technological advancements, and the growing shift towards online transactions and remote services. Understanding the customer's increasing inclination towards digital solutions, the need for convenience, and the competitive landscape within the automotive industry are crucial components of this analysis. It considers the evolving expectations of customers regarding accessibility, ease of use, and efficiency in procuring spare parts and scheduling vehicle services.

2.2.2 DESIGNED SYSTEM STUDIED

The proposed designed system aims to revolutionize the current process by introducing an online platform dedicated to commercial vehicle spare parts procurement and service booking. The system's design will focus on providing a user-friendly interface accessible via a website. This platform will offer customers the convenience of remotely browsing, selecting, and purchasing required spare parts hassle-free. Additionally, the platform will facilitate seamless booking of vehicle services, providing users with the ability to schedule maintenance appointments efficiently.

2.3 DRAWBACKS OF EXISTING SYSTEM

- Customers are constrained by the necessity of physically visiting company locations, which can be time-consuming and inconvenient
- The system restricts access to spare parts and services solely through physical outlets, creating barriers for customers who prefer or require remote or online solutions.
- The current method lacks efficiency due to manual processes involved in purchasing spare parts and scheduling vehicle maintenance, leading to potential delays and errors.
- Customers located in distant areas might face difficulties accessing the company's physical locations, causing inconvenience and delays in procuring necessary parts and services.
- The system's rigid structure does not adapt to the changing needs and preferences of modern customers who increasingly seek convenient, on-demand services.
- The inconvenience of the current system might result in decreased customer satisfaction and retention due to the inability to meet their expectations for easy and accessible solutions.
- In an evolving market where competitors offer online solutions and enhanced customer experiences, the existing system's limitations might place the company at a disadvantage.

2.4 PROPOSED SYSTEM

The current system for procuring commercial vehicle spare parts and booking service appointments involves customers directly visiting the company's physical location to purchase parts and schedule vehicle maintenance. This process lacks convenience and accessibility for customers seeking an efficient solution.This project aims to revolutionize the automotive industry by introducing an online platform designed to facilitate the seamless purchase of spare parts and the hassle-free booking of vehicle services. The website will provide a user-friendly interface that allows customers to effortlessly browse, select, and purchase the required spare parts with ease.

2.5 ADVANTAGES OF PROPOSED SYSTEM

- Enhanced Accessibility
- Convenience for Customers
- Time Efficiency
- Improved Customer Satisfaction
- Adaptability and Modernization
- Competitive Edge
- Geographical Flexibility

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Planning, organizing, and managing resources to ensure the achievement of particular project goals and objectives is the process of project management. A feasibility study is a preliminary examination of a prospective project or end to determine its merits and viability. A feasibility study aims to provide an objective assessment of the technical, economic, financial, legal, and environmental elements of a proposed project. The information can then be used by decisionmakers to decide whether to proceed with the project or not. The findings of the feasibility study can also be used to develop a practical project plan and budget. It cannot be simple to determine whether or not a proposed project is worthwhile pursuing without a feasibility study. The document provides the feasibility of the project that is being designed and lists. Various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibility. The following are its features:

3.1.1 Economical Feasibility

The economical feasibility assessment will evaluate the financial viability of transitioning from the current physical procurement system to the proposed online platform. It will delve into the costs involved in developing the website, implementing the necessary infrastructure, ongoing maintenance, and operational expenses. This assessment will weigh these costs against potential revenue gains, cost savings from streamlined processes, and increased customer engagement. Financial projections will assess the return on investment (ROI) and the platform's profitability in the long run.

3.1.2 Technical Feasibility

Technical feasibility will explore the technological aspects needed to develop and implement the online platform. This assessment will consider the compatibility of the proposed platform with existing systems, the required software and hardware components, cybersecurity measures, and the scalability of the technology. It will also evaluate whether the infrastructure and technical expertise are available to support the platform's development and maintenance.

3.1.3 Behavioral Feasibility

Behavioral feasibility will focus on understanding how stakeholders, including customers, staff, and management, will adapt to the transition from a physical to an online platform. It will assess the readiness of customers to embrace digital solutions, the training needs of staff to manage the platform, and the impact on existing work processes. Surveys, interviews, or user studies will be conducted to gauge stakeholder attitudes, preferences, and potential resistance to change.

3.1.4 Feasibility Study Questionnaire

1. **Project Overview:** The project aims to introduce an online platform revolutionizing commercial vehicle spare parts procurement and service booking. This web-based system targets enhancing accessibility and convenience by allowing users to purchase spare parts and schedule vehicle services effortlessly.
2. **To what extent the system is proposed for?**
   The system is proposed to cater to the entire process of commercial vehicle spare parts procurement and service booking. It aims to facilitate both purchasing spare parts and scheduling vehicle maintenance services.
3. **Specify the Viewers/Public involved in the System?**
   The viewers/public involved in the system include customers/users seeking to procure spare parts and schedule vehicle services, service branch staff managing service bookings, parts managers handling inventory, and the system administrators overseeing the entire platform.
4. **List the Modules included in your System?**
   - Admin Module
   - User Module
   - Service Branch Module
   - Parts Manager Module
5. **Identify the users in your project?**
   Users encompass customers seeking spare parts and services, service branch staff managing bookings, parts managers overseeing inventory, and administrators responsible for system supervision.
6. **Who owns the system?**
   The ownership of the system lies with the entity responsible for its development and maintenance. This could be the automotive company or the organization spearheading the project.

7. **System is related to which firm/industry/organization?**
   The system is related to the automotive industry, particularly targeting commercial vehicle spare parts procurement and service booking. It might be associated with an automotive company, dealership, or service provider within this industry.
8. **Details of person that you have contacted for data collection?**
   The specific details of the person contacted for data collection are not provided in the project overview. However, data collection for this project would likely involve collaborating with Commercial vehicle dealerships , software developers, maintenance experts

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor	- intel core i5
RAM	- 1 6 . 0 0 G B
Hard disk	- 5 1 2 G B

3.2.2 Software Specification

Front End	-	HTML, CSS
Back End	-	PYTHON DJANGO
Client on PC	-	Windows 11 and above.
Database	-	SQLite
Technologies used	-	JS, HTML5, AJAX, J Query, Python Django, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 PYTHON

Python is a high-level, versatile programming language renowned for its simplicity, readability, and versatility. Created by Guido van Rossum in the late 1980s, Python emphasizes code readability and ease of use, fostering a clean and concise syntax that facilitates rapid development. Known for its extensive standard library, Python offers a rich set of modules and packages, empowering developers to accomplish diverse tasks, from web development and data analysis to artificial intelligence and scientific computing.

Python's interpreted nature allows for interactive and dynamic coding, making it an ideal choice for beginners and seasoned developers alike. Its object-oriented approach promotes code reusability and modularity. Python's community-driven ethos has led to a vast ecosystem of third-party libraries and frameworks, such as Django, Flask, NumPy, and TensorFlow, contributing to its widespread adoption across industries. Its cross-platform compatibility and support for multiple programming paradigms, including procedural, functional, and object-oriented programming, underscore Python's status as a go-to language for various applications.

3.3.2 DJANGO

Django is a high-level, open-source web framework written in Python, designed to simplify and expedite the creation of robust web applications. Developed with a "batteries-included" philosophy, Django provides a comprehensive set of tools and functionalities, enabling developers to build secure, scalable, and maintainable web applications swiftly.

At its core, Django follows the Model-View-Controller (MVC) architectural pattern, emphasizing the Model-View-Template (MVT) structure. It offers an ORM (Object-Relational Mapping) layer, abstracting database interactions and allowing developers to work with database models using Python classes. This simplifies database-related tasks, enhancing code readability and maintainability.

3.3.3 MYSQL SERVER

MySQL Server is a widely used open-source relational database management system (RDBMS) known for its reliability, scalability, and ease of use. Developed by Oracle Corporation, MySQL serves as a robust platform for storing, managing, and retrieving data. It employs a client-server architecture, allowing multiple users to access databases concurrently, making it suitable for various applications, from small-scale web applications to enterprise-level systems.

This server utilizes Structured Query Language (SQL) to interact with databases, enabling users to create, modify, and query data efficiently. MySQL supports multiple storage engines, offering flexibility in handling different data types and requirements. Its features include strong data security measures, ACID (Atomicity, Consistency, Isolation, Durability) compliance for transaction support, and high performance through indexing and caching mechanisms. MySQL's versatility makes it a popular choice among developers and businesses seeking a stable and cost-effective solution for managing their data needs.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997. UML stands for Unified Modeling Language. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design. After some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete. UML includes the following nine diagrams.

• Class diagram

• Object diagram

• Use case diagram

• Sequence diagram

• Collaboration diagram

• Activity diagram

• State chart diagram

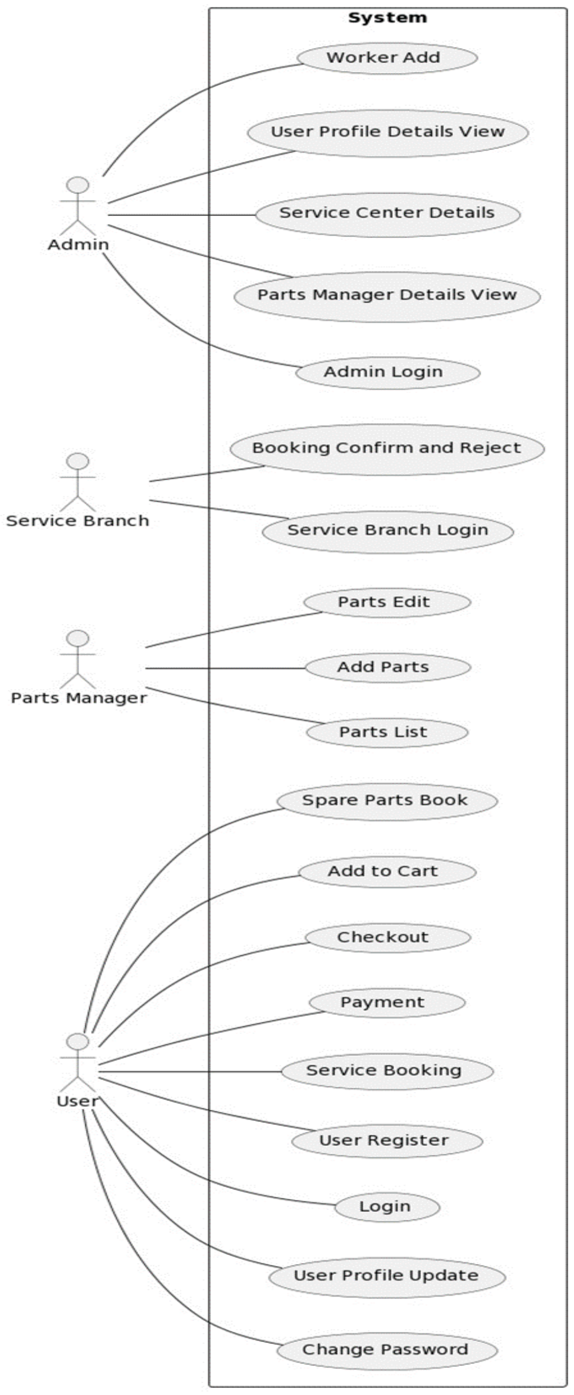
• Deployment diagram

• Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refersto something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems. System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customerrelations.Ause case diagram containsfour components.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases


System
Worker Add
User Profile Details View
Service Center Details
Parts Manager Details View
Admin Login
Admin
Booking Confirm and Reject
Service Branch Login
Service Branch
Parts Edit
Add Parts
Parts List
Parts Manager
Spare Parts Book
Add to Cart
Checkout
Payment
Service Booking
User Register
Login
User Profile Update
Change Password
User

4.2.1 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

Sequence Diagram Notations –

i. **Actors** – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.

ii. **Lifelines** – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.

iii. **Messages** – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

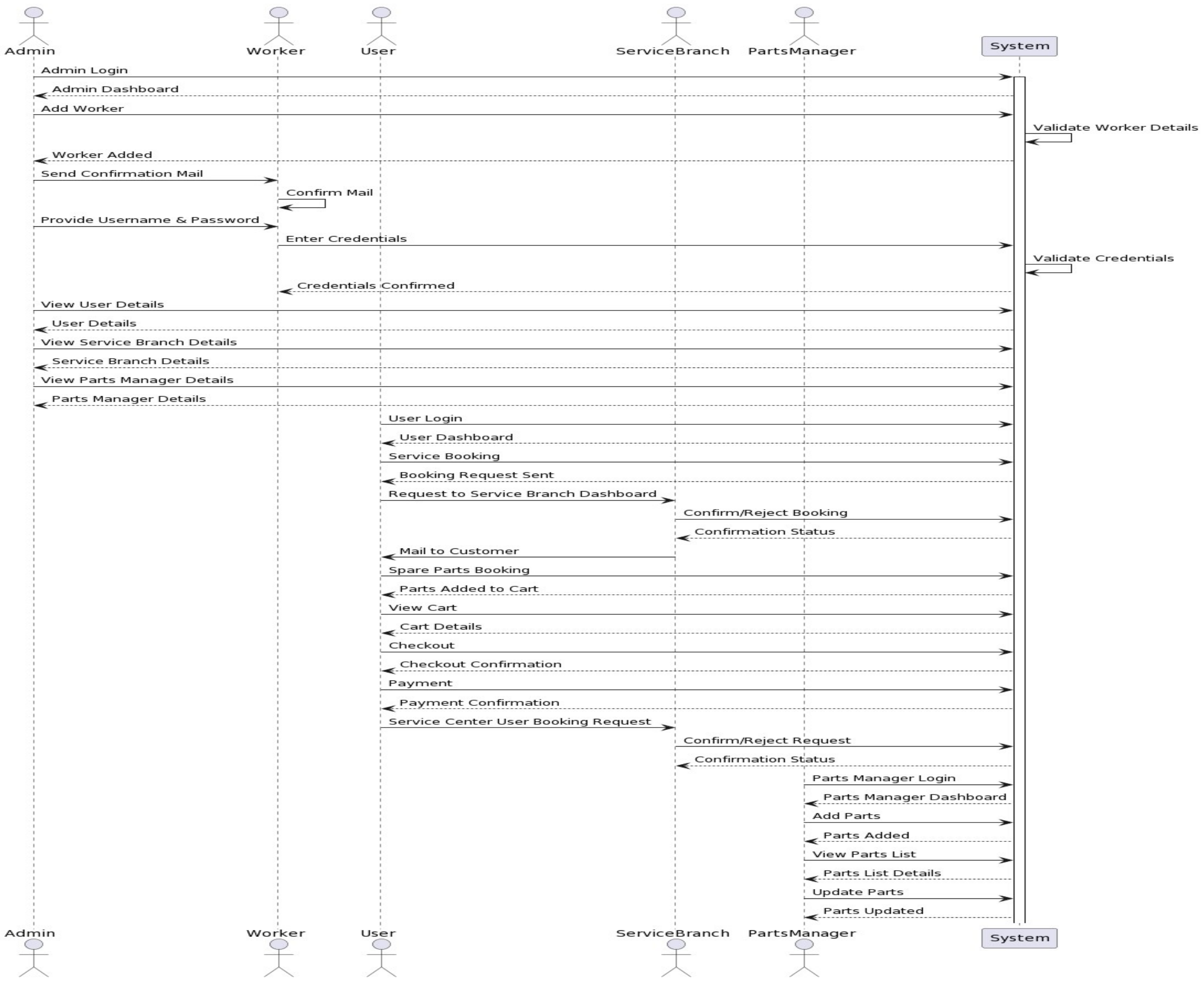
Messages can be broadly classified into the following categories:

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

iv. **Guards –** To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met.Guards play an important role in letting software developers know the constraints attached to a system ora particular process

Uses of sequence diagrams –

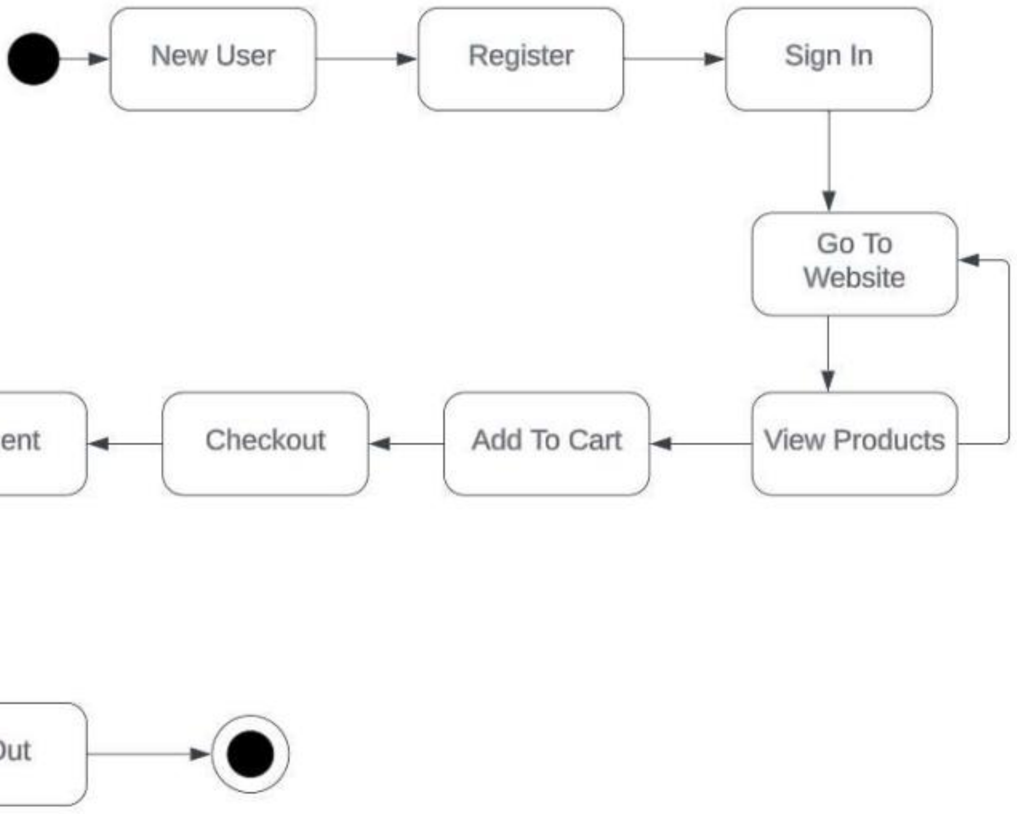
- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.



4.2.2 State Chart Diagram

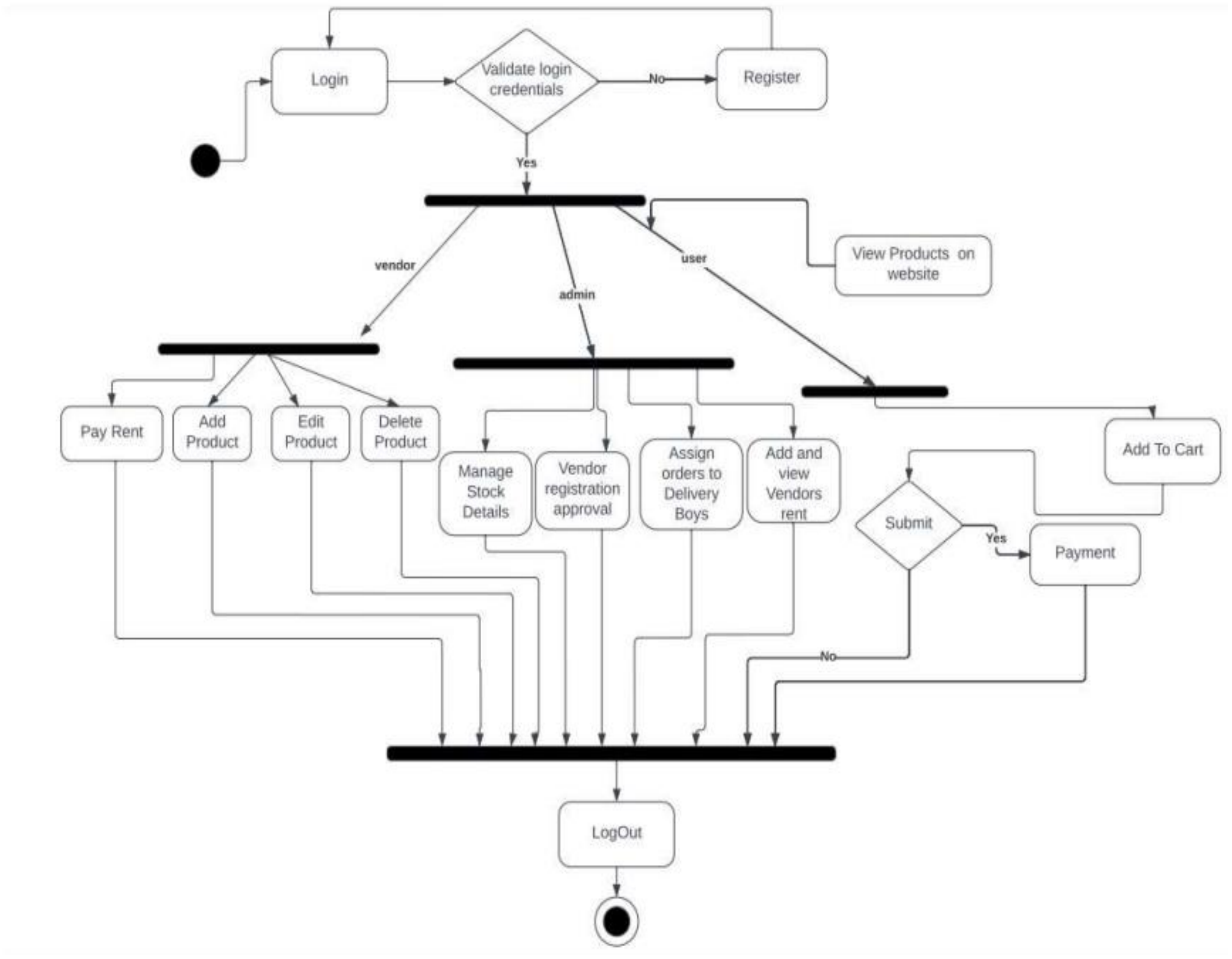
State Chart Diagram A particular form of diagram used in computer science and related subjects to explain how systems behave is called a state diagram. State diagrams call for the system being represented to consist of a finite number of states; occasionally, this is the case, and other times, it's only an acceptable abstraction. State diagrams come in a variety of shapes and sizes, each with a distinct meaning. State diagrams are there to provide a system's behaviour an abstract explanation. In order to evaluate and demonstrate this behaviour, a sequence of events that could occur in one or more hypothetical states is employed. "Each diagram typically depicts objects of a single class and monitor the different states of its objects across the system," according to this.

- **Initial State -** This state represents the starting point of the system or object and is denoted by a solid black circle.
- **State -** This element describes the current state of the system or object at a specific point in time and is represented by a rectangle with rounded corners.
- **Transition** - This element shows the movement of the system or object from one state to another and is represented by an arrow.
- **Event and Action** - An event is a trigger that causes a transition to occur, and an action is the behavior or effect of the transition.
- **Signal -** A message or trigger caused by an event that is sent to a state, causing a transition to occur.
- **Final State -** The State Chart Diagram ends with a Final State element, which is represented by a solid black circle with a dot inside. It indicates that the behavior of the system or object has completed



4.2.2 Activity Diagram

Activity diagrams depict how different levels of abstraction of activities are linked to provide a service. Typically, an event should be completed by some activities, particularly when the activity is intended to do multiple separate goals that need coordination. Another typical requirement is how the events in a single use case interact with one another, particularly in use cases where operations may overlap and require coordination. It may also be used to show how a collection of interrelated use cases interacts to reflect business operations.



4.2.3 Class Diagram

Class diagram is a static diagram. It represents the static view of the application. Class diagrams are useful for visualising, describing, and documenting various system components as well as for writing executable code for software applications. A class diagram describes the constraints imposed on the system together with the properties and operations of a class. The only UML diagrams that can be directly converted into objectoriented languages are class diagrams, which are extensively utilised in the designing of object-oriented systems. An assortment of classes, interfaces, affiliations, partnerships, and limitations are displayed in a class diagram. It also goes by the name "structural diagram."

4.2.4 Object Diagram

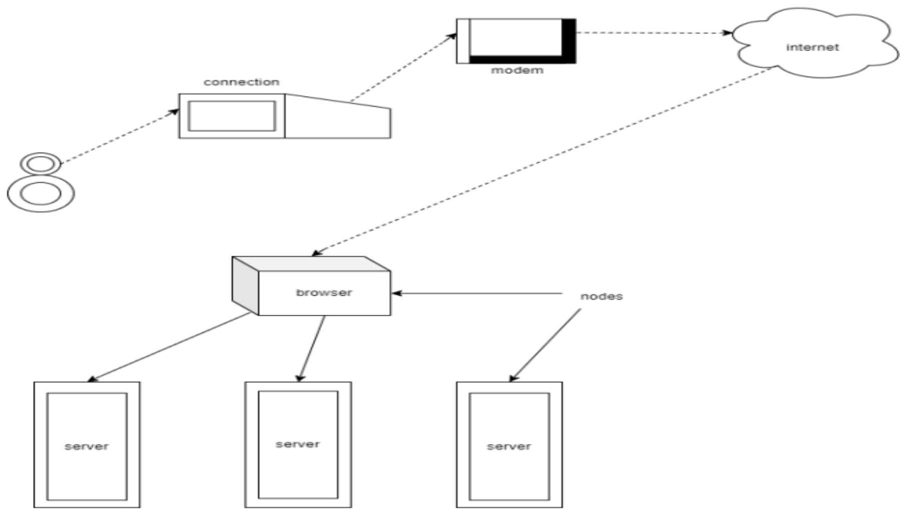
Class diagrams are necessary before object diagrams can be created since they are the ancestor of object diagrams. An object diagram represents a particular instance of a class diagram. The underlying concepts used in class and object diagrams are the same. Object diagrams may also describe the static view of a system, although this static view only depicts a current state of the system. Object diagrams are used to show links between a set of things.

4.2.5 Component Diagram

Component diagrams have different behaviours and personalities. The physical parts of the system are represented using component diagrams. Executables, libraries, files, documents, and other items that are physically present in a node are just a few examples. Component diagrams are used to show how the components of a system are connected and arranged. These diagrams may also be used to construct systems that can be run

4.2.8 Deployment Diagram

An execution architecture of a system, containing nodes like hardware or software execution environments, and the middleware linking them, is shown in a deployment diagram, a form of UML diagram. Typically, deployment diagrams are used to represent the actual hardware and software of a system. By using it, you can comprehend how the hardware will physically deliver the system. In contrast to other UML diagram types, which primarily depict the logical components of a system, deployment diagrams assist describe the hardware structure of a system.



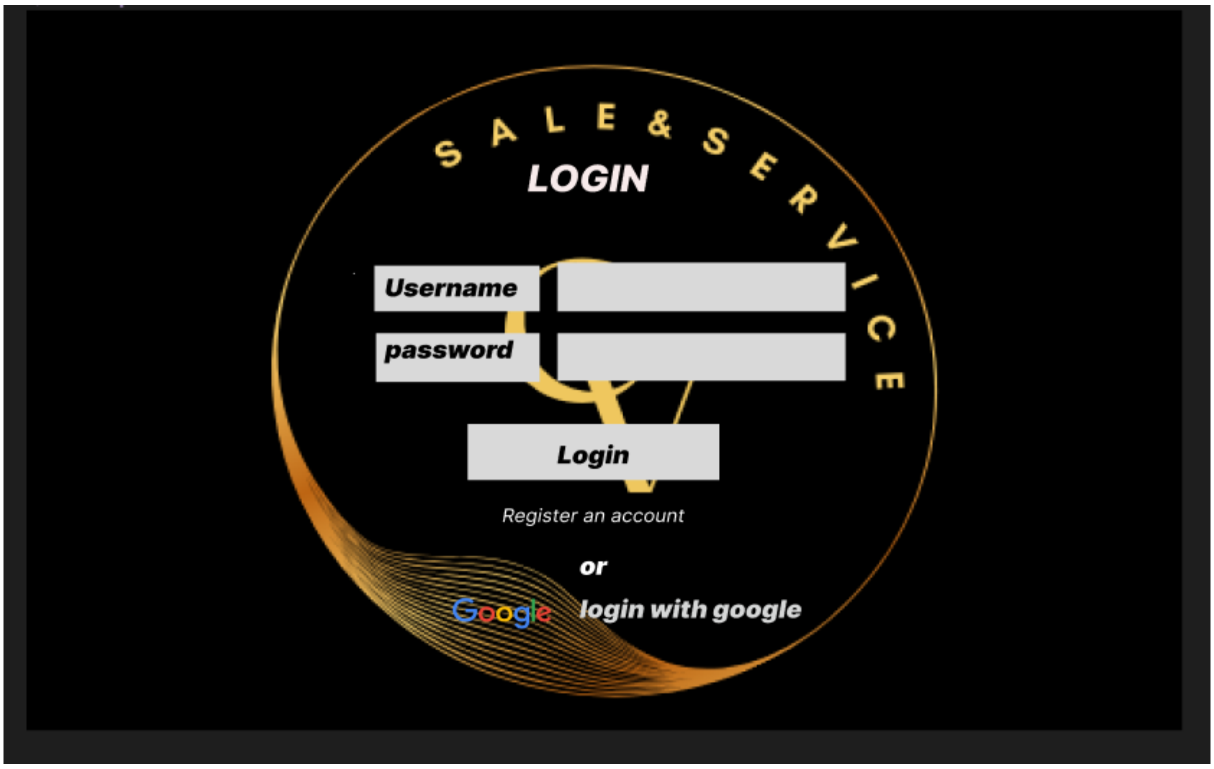
4.2.9 Collaboration Diagram

A collaboration diagram is a diagram that is used to represent the relationships between objects in a system. It is similar to a sequence diagram in that it represents the same information, but it does so in a different way. Instead of showing the flow of messages between objects, it depicts the structure of the objects in the system. This is because collaboration diagrams are based on objectoriented programming, where objects have various attributes and are connected to each other. Thus, collaboration diagrams are a visual representation of the object architecture in a system.

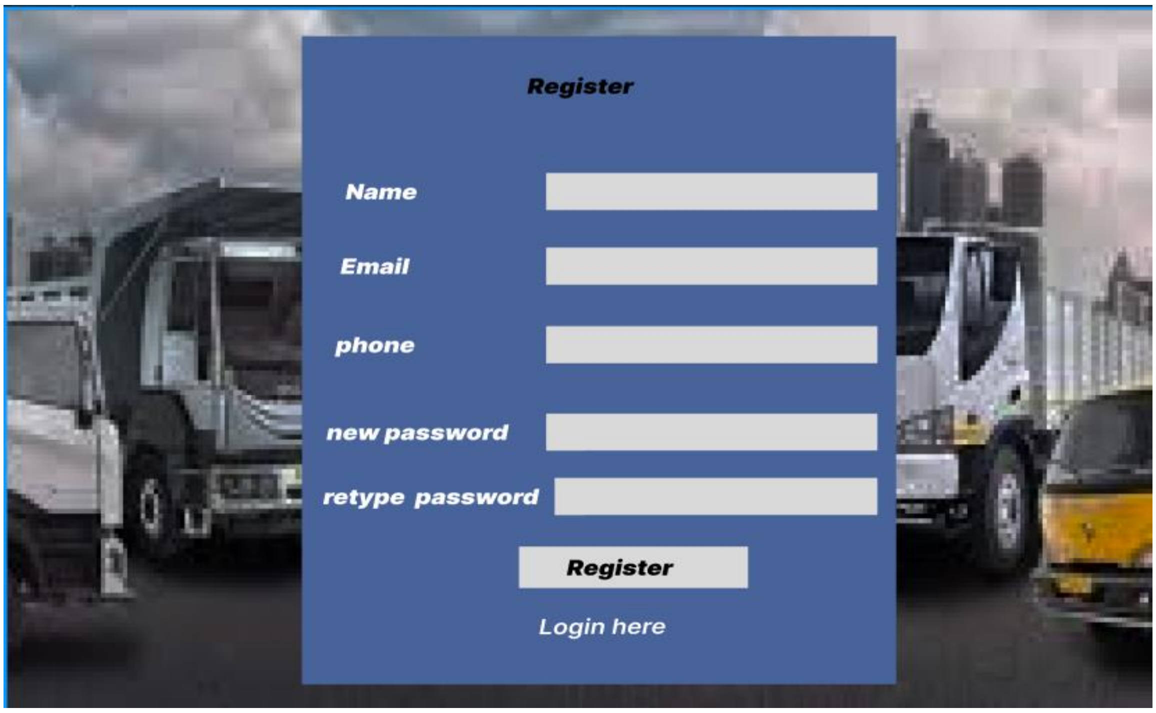
A component diagram includes the following components:

4.3 USER INTERFACE DESIGN USING FIGMA

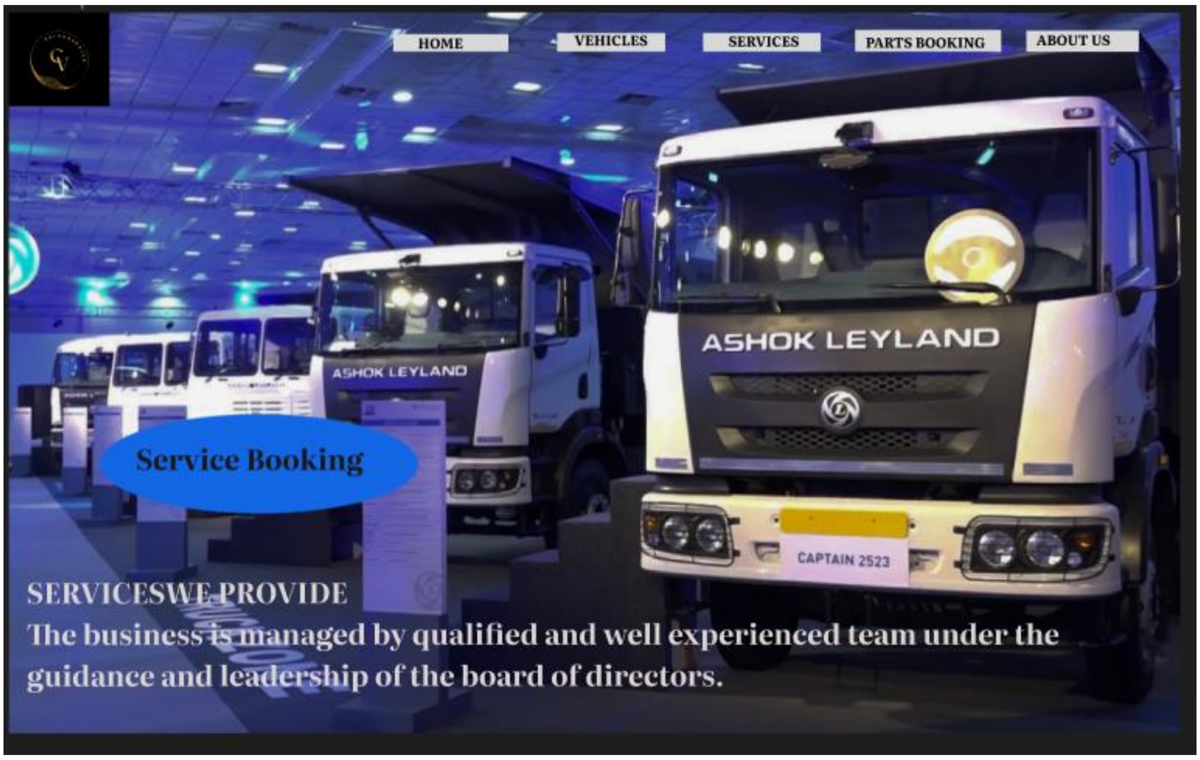
Form Name: Login page



Form Name: Registration page

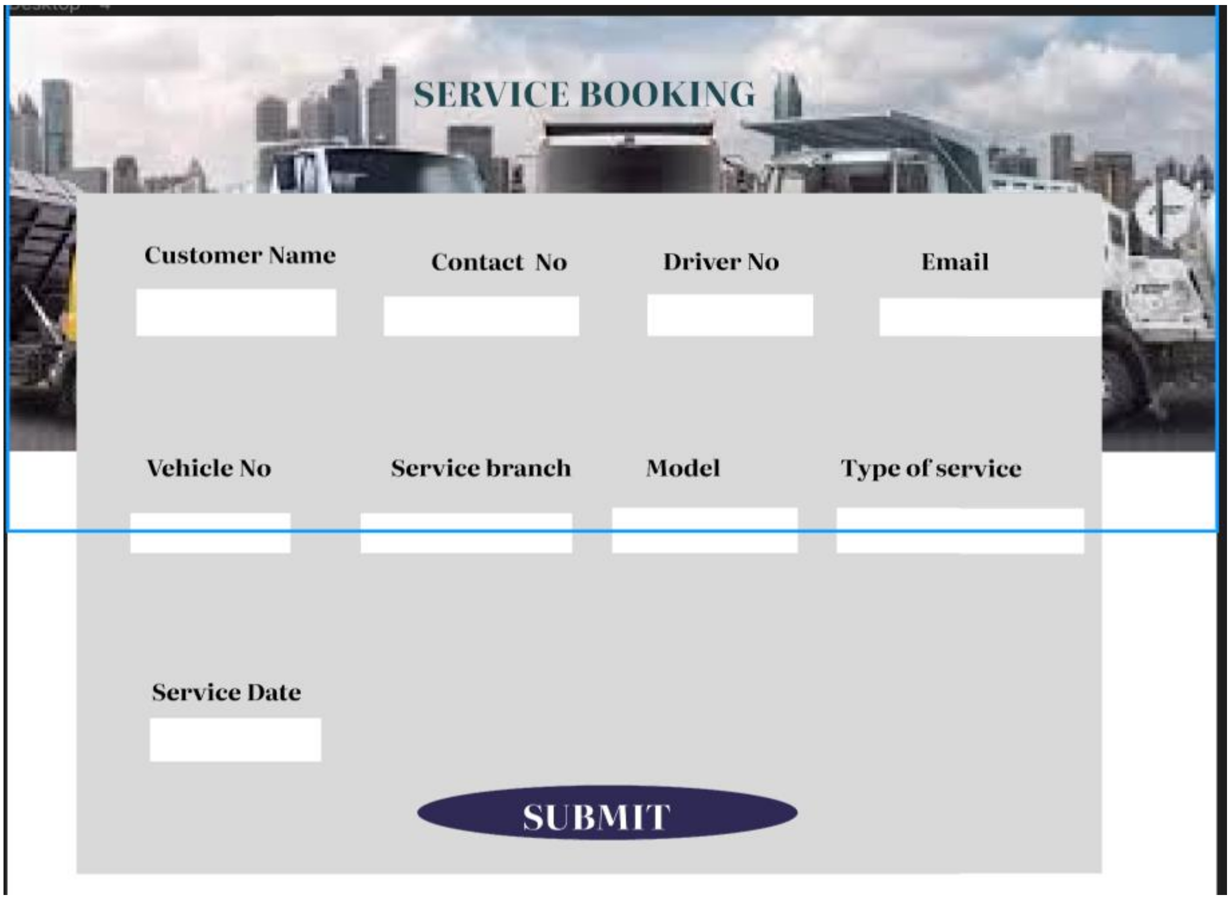


Form Name: Home Page



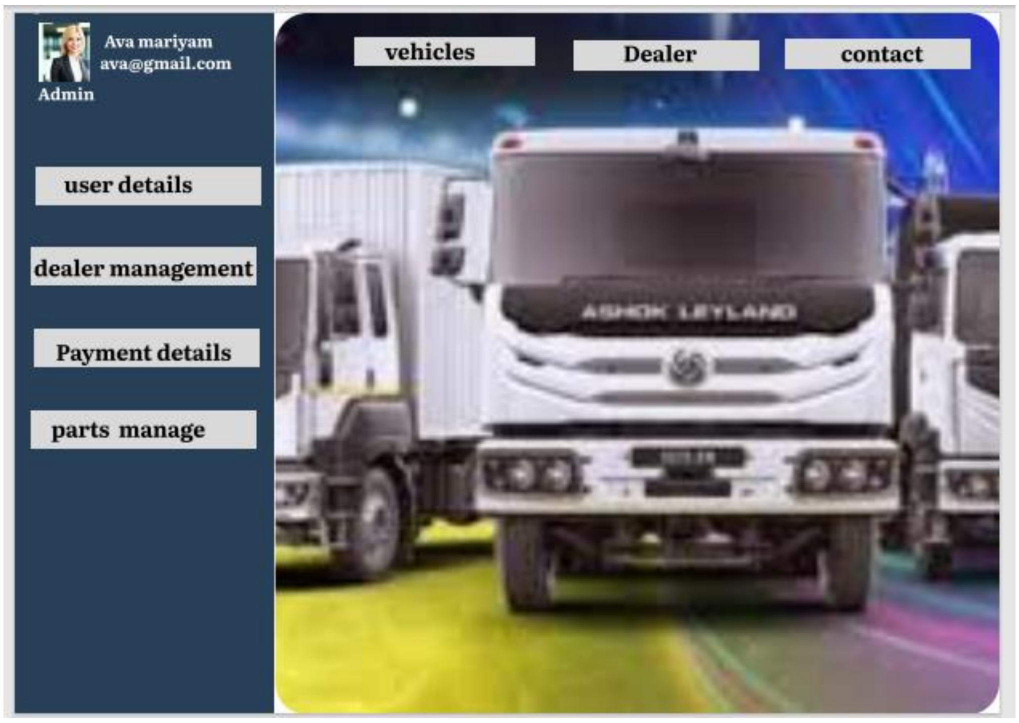
SERVICESWE PROVIDE
The business is managed by qualified and well experienced team under the guidance and leadership of the board of directors.

Form Name: Booking page

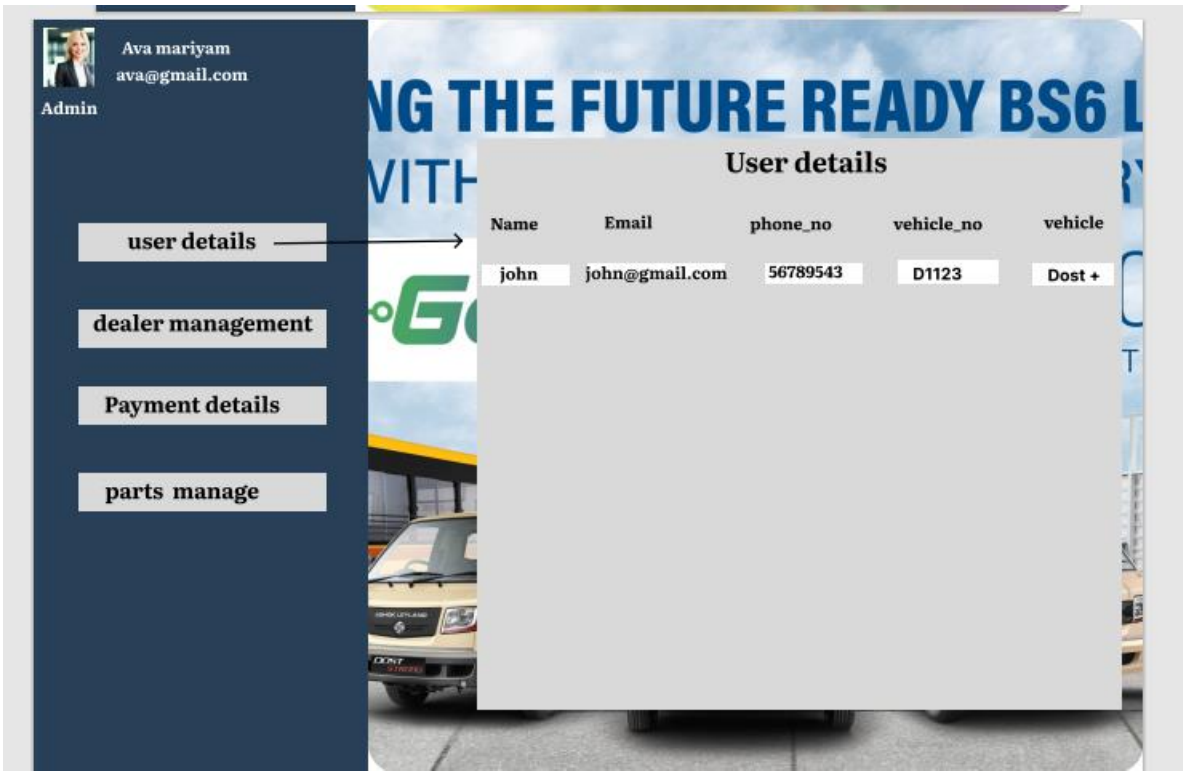


Customer Name	Contact No	Driver No	Email
Vehicle No	Service branch	Model	Type of service
Service Date			

Form Name: Admin page



Form Name: User details page



User details

Name	Email	phone_no	vehicle_no	vehicle
john	john@gmail.com	56789543	D1123	Dost +

4.4 DATABASE DESIGN

A database is a collection of data that has been organized to make it simple to manage and update. Information security could be one of the main aims of any database. The database design process is divided into two steps. The goal of the first step is to gather user requirements to create a database that as clearly as possible satisfies user needs. It is known as information-level design and is carried out without the aid of any DBMS. An information-level design is changed to a specific DBMS design that will be used to develop the system in the following stage. The physical-level design phase is where the characteristics of the specific DBMS are considered.

4.4.1 Relational Database Management System (RDBMS)

A Relational Database Management System (RDBMS) is a critical software application for organizing and managing data in a structured manner. It stores data in tables with rows and columns, where each row represents a record, and each column is a specific attribute or field. RDBMS systems like MySQL, Oracle, or Microsoft SQL Server ensure data integrity, enforce relationships between tables, and allow for efficient data retrieval and manipulation through Structured Query Language (SQL). These systems are widely used in various applications, such as e-commerce websites, financial systems, and inventory management, due to their robustness, scalability, and ability to handle complex data structures, making them essential for modern data-driven environments. A Relational Database Management System (RDBMS) is a cornerstone of modern data management. It structures data into tables, where each row represents a unique entry, and each column corresponds to a specific attribute, ensuring a logical and organized storage system. RDBMS systems employ complex algorithms for data retrieval and manipulation, guaranteeing data consistency and adherence to predefined relationships between tables. Popular RDBMS software, including PostgreSQL, MySQL, and Microsoft SQL Server, provides robust features for transactions, data security, and scalability. These systems are integral to a myriad of applications, from healthcare records and customer databases to inventory control and financial platforms, supporting the efficient storage, retrieval, and analysis of vast datasets, making them essential tools in today's data-driven world.

4.4.2 Normalization

Normalization is a database design technique used in relational database management systems (RDBMS) to eliminate data redundancy and improve data integrity. It involves organizing data in a way that reduces data duplication and ensures that relationships between tables are defined and maintained.

The primary goals of normalization are

Minimizing Data Redundancy: By breaking down data into separate tables and ensuring that each piece of data is stored only once, normalization helps reduce the chances of data inconsistencies or errors.

Ensuring Data Integrity: Normalization enforces the rules of referential integrity, which means that relationships between tables are well-defined and maintained. This ensures that data remains accurate and consistent.

Normalization typically involves dividing a database into multiple related tables and using primary keys and foreign keys to establish relationships between these tables. There are several normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF), each with specific rules and requirements. The level of normalization achieved depends on the specific needs of the database and the trade-off between data redundancy and query performance.

By applying normalization principles, designers can create efficient and reliable database structures that support data integrity and ease data maintenance and manipulation.

There are several normal forms (NF) that define specific rules and requirements for achieving progressively higher levels of normalization. Here are the most common normal forms, from First Normal Form (1NF) to Fifth Normal Form (5NF)

First Normal Form (1NF)

• Each table has a primary key.
• All columns contain atomic (indivisible) values.
• There are no repeating groups or arrays in columns.

Second Normal Form (2NF):

• The table is in 1NF.

• All non-key attributes are fully functionally dependent on the entire primary key. In other words, all non-key attributes must be dependent on the entire primary key, not just part of it.

Third Normal Form (3NF):

• The table is in 2NF.

• There is no transitive dependency, meaning that non-key attributes are not dependent on other non-key attributes.

4.4.3 Sanitization

In Python Django, sanitization refers to the process of cleaning and validating data to ensure that it is safe and free from malicious content before it is used or stored in a database. Sanitization is a crucial security practice to prevent various forms of attacks, such as cross-site scripting (XSS) and SQL injection, which can compromise the security and integrity of a web application.

4.4.4 Indexing

Indexing in Django is a fundamental database optimization technique. It involves creating data structures, or indexes, on specific fields to expedite data retrieval. These indexes act as signposts that enable the database to swiftly locate and fetch relevant data, especially in tables with substantial amounts of information. Django offers automatic index creation for primary keys, unique fields, and foreign keys. Additionally, developers can define custom indexes for fields frequently used in filtering, sorting, or searching. The choice of proper indexing plays a pivotal role in improving query performance, resulting in more responsive and scalable web applications. It's essential to consider application-specific query patterns and create indexes accordingly, as well as to understand the capabilities and limitations of the chosen database backend. Effective indexing is a cornerstone of efficient database operations in Django, contributing to enhanced application performance.

4.5 TABLE DESIGN

1.tbl_registration

Primary Key: **id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	User_Id	AutoField	Primary Key	Id for registration table
2	Username	CharField(150)	Not Null	User’s username
3	Email	EmailField	Unique	User’s email
4	Address	TextField	Not Null	User’s address
5	Phone Number	CharField(12)	Not Null	User’s Phone Number
6	Password	CharField	Not Null	User’s Password
7	Role	CharField(100)	Default:””	User’s role type

2. tbl_Userprofile

Foreign Key : **User_id**
Primary Key : **Id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Id	BigAutoField	Foreign Key	Reference to user_id
2	First_name	CharField(255)	Not null	User firstname
3	Last_name	CharField(255)	Not Null	Users last name
4	Phone_number	CharField(12)	Not Null	User phoneno
5	Address	TextField	Not Null	User Address
6	pincode	CharField(10)	Not Null	User pincode
7	city	CharField(50)	Not Null	User city
8	state	CharField(50)	Not Null	User state
9	User_id	OneToOneField	Not Null	Unique identification

3. tbl_Booking

Foreign Key : **User_id**
Primary Key : **Booking_Id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Booking_Id	BigAutoField	Primary key	Unique identifier
2	Customer name	CharField(15)	Not Null	Customer name is username
3	driver_number	CharField(15)	Not Null	Vehicle driver number
4	vehicle_number	CharField(50)	Not Null	User vehicle number
5	service_branch	CharField(50)	Not Null	Service branch
6	vehicle_model	CharField(50)	Not Null	Vehicle model
7	service_type	CharField(10)	Not Null	Service type
8	service_date	DateField	Not Null	Service date
9	email	EmailField	Not Null	User email i
10	User_id	OneToOneField	Foreign key	Reference to register page

4. tbl_Parts

Primary Key : **Parts_Id**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	partsname	CharField(100)	Primary key	Parts name
2	description	TextField	Not Null	Parts description
3	price	DecimalField	Not Null	Parts price
4	parts_image	ImageField	Not Null	Parts image
5	quantity	PositiveInteger	Not Null	Parts quantity
6	categories	CharField	Not Null	Parts categories

5. tbl_CartItem

Primary Key : **CartItem_id**
Foreign key : **Cart, product**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	CartItem_id	BigAutoField	Primary key	Cartitem id
2	quantity	PositiveIntegerField	Not Null	Product quantity
3	cart	ManyToManyField	Foreign key	Reference to cart table
4	product	PositiveIntegerField	Foreign key	Reference to parts table

6. tbl_Order

Primary Key : **Order_id**
Foreign key : **User_id ,product**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Order_id	BigAutoField	Primary key	Unique identification
2	products	PositiveIntegerField	Foreign key	Reference to parts table
3	total_amount	DecimalField	Not Null	Total amount
4	payment_id	CharField(100)	Not Null	Not Null
5	payment_status	BooleanField	Not Null	Not Null
6	created_at	DateTimeField	Not Null	Not Null
7	User_id	OneToOneField	Foreign key	Reference to register table

7. tbl_OrderItem

Primary Key : **Order_id**
Foreign key : **quantity,product,Item_total**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Order_id	CharField	Primary key	Order id unique
2	product	PositiveIntegerField	ForeignKey	Reference to parts table
3	quantity	PositiveIntegerField	ForeignKey	Reference to parts table
4	item_total	DecimalField	Not Null	Reference to parts table

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question – Does the software behave as specified? Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted. Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information. Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are: Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error. If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met. There are three ways to test program.
- For correctness
- For implementation efficiency
- For computational complexity. Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

5.2 TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print 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 mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours 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 focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is whitebox oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested. Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries

5.2.2 Integration Testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

5.2.3 Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests. Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions thatwill fully exercise all functional requirements for a program. Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

5.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm’s need. The software should keep in touch with perspective system; user at the time ofdeveloping and making changes whenever required. This done with respect to the following points: ¬ Input Screen Designs, ¬ Output Screen Designs, The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

5.2.5 Automation Testing

Automated testing is a process that validates if software is functioning appropriately and meeting requirements before it is released into production. This software testing method uses scripted sequences that are executed by testing tools. UI automation testing is a technique where these testing processes are performed using an automation tool. Instead of having testers click through the application to verify data and action flows visually, test scripts are written for each test case.
A series of steps to follow when the verifying data is then added. Automatic testing is required when you want to run the same test cases across multiple machines at the same time.

5.2.6 Selenium Testing

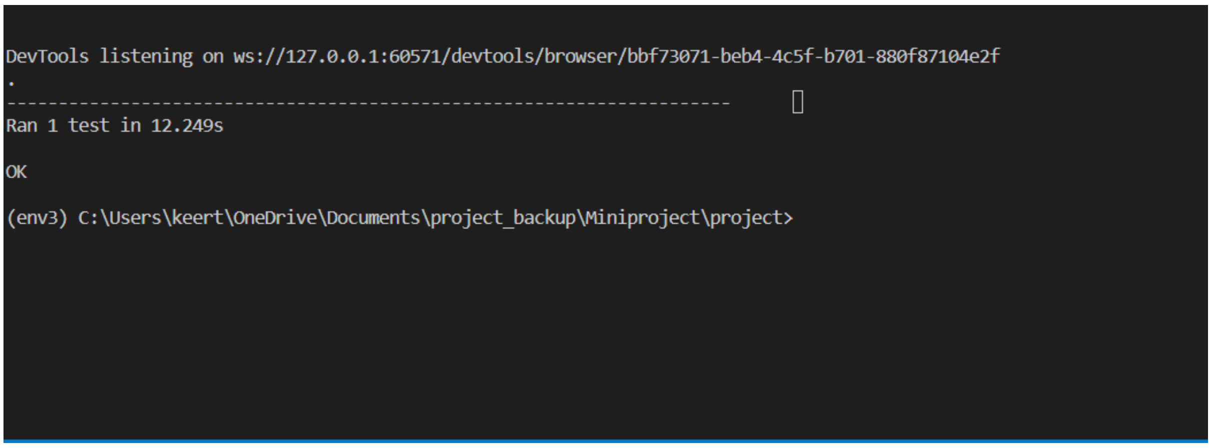
Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python, etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing. Selenium Integrated Development Environment (IDE) is the simplest framework in the Selenium suite and is the easiest one to learn. It is a Chrome and Firefox plugin that you can install as easily as you can with other plugins. However, because of its simplicity, Selenium IDE should only be used as a prototyping tool. If you want to create more advanced test cases, you will need to use either Selenium RC or WebDriver. Selenium RC was the flagship testing framework of the whole Selenium project for a long time. This is the first automated web testing tool that allows users to use a programming language they prefer. RC can support the following programming languages JavaC#, PHP, Python, Perl, Ruby. Selenium can be used to automate functional tests and can be integrated with automation test tools such as Maven, Jenkins, & Docker to achieve continuous testing. It can also be integrated with tools such as TestNG, & JUnit for managing test cases and generating reports. For example, Selenium is used to test the login functionality of the patient portal by automating the entry of login credentials and verifying that the correct patient dashboard is displayed after successful authentication. Selenium can also be used to automate the testing of appointment scheduling by programmatically entering various input scenarios and verifying that the resulting output matches the expected behavior.

Test Case 1

Code

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
import time

 class LoginTest(unittest.TestCase):
     def setUp(self):
        # Start the Selenium WebDriver
        self.driver = webdriver.Chrome()  # Adjust based on your WebDriver configuration
        self.driver.get("http://127.0.0.1:8000/login")
        time.sleep(10)

    def test_login_successful(self):
        # Find the username, password, and login button elements
        username_input = self.driver.find_element(By.ID, "username")
        password_input = self.driver.find_element(By.ID, "password")
        login_button = self.driver.find_element(By.ID, "submitBtn")

        # Enter valid credentials
        username_input.send_keys("Sisira12")
        password_input.send_keys("Sisira@12")

        # Click the login button
        login_button.click()
        time.sleep(2)
        self.assertEqual(self.driver.current_url, 'http://127.0.0.1:8000/')
 def tearDown(self):
        self.driver.close()

if __name__ == "__main__":
    unittest.main()
```

Output



```
DevTools listening on ws://127.0.0.1:60571/devtools/browser/bbf73071-beb4-4c5f-b701-880f87104e2f
.
----------------------------------------------------------------------
Ran 1 test in 12.249s

OK

(env3) C:\Users\keert\OneDrive\Documents\project_backup\Miniproject\project>
```

Test Report

Test Case 1

Project Name: Commercial vehicle service and Spare parts Management					
Login Test Case					
Test Case ID: Test_1			**Test Designed By: Keerthi u**		
Test Priority(Low/Medium/High):High			**Test Designed Date: 4/12/2023**		
Module Name: Login page			**Test Executed By :** RINI KURIAN		
Test Title : Verify login with email and password			**Test Execution Date: 5/12/2023**		
Description: Testing the login page					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login page		Login Page should be displayed	Login Page displayed	Pass
2	Provide valid username	Username: Keerthi	User should be able to login	User logged in and navigated to dashboard	Pass
3	Provide valid password	Keerthi@1			
4	Click on Login Button				
5	Provide Invalid username or password	Username: keerthi1 Password: keerthi123	User should not be able in Login	Message for enter valid email id or password displayed	Pass
6	Provide null username or password	username:null Password: null			
7	Click on Login In button				
Post-Condition: Post-Condition: User is validated with database and login to the website. The account session details are logged in database					

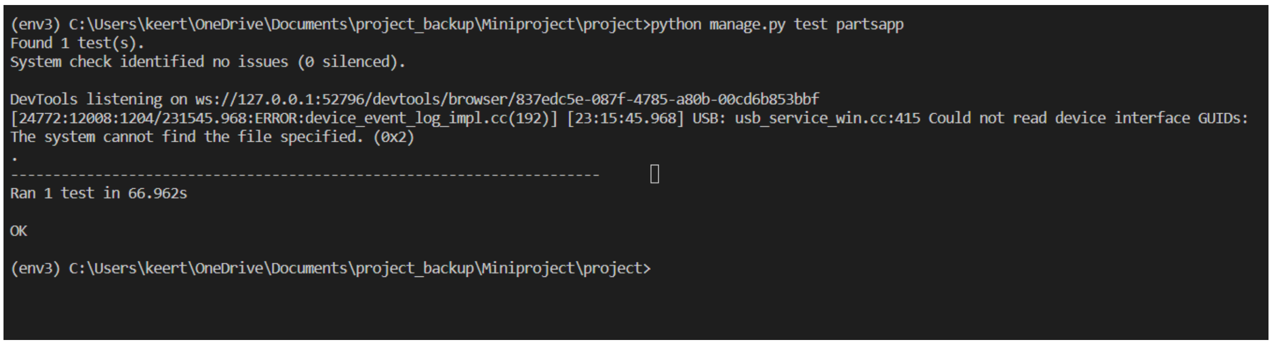
Test Case 2:

Code

```
import unittest

from selenium import webdriver

from selenium.webdriver.common.by import By

import time

class LoginAndNavigationTest(unittest.TestCase):

    def setUp(self):

        self.driver = webdriver.Chrome()

        self.driver.get("http://127.0.0.1:8000/login")

        time.sleep(5)

    def test_login_successful(self):

        # Login process (similar to your previous test)

        username_input = self.driver.find_element(By.ID, "username")

        password_input = self.driver.find_element(By.ID, "password")

        login_button = self.driver.find_element(By.ID, "submitBtn")

        username_input.send_keys("Sisira12")

        password_input.send_keys("Sisira@12")

        login_button.click()

        time.sleep(5)

        self.assertEqual(self.driver.current_url, 'http://127.0.0.1:8000/')

        self.driver.get("http://127.0.0.1:8000/booking/")

        time.sleep(5)
        email_input = self.driver.find_element(By.ID, "email")
        driver_number_input = self.driver.find_element(By.ID, "driver_number")
        vehicle_number_input = self.driver.find_element(By.ID, "vehicle_number")
        service_branch_dropdown = self.driver.find_element(By.ID, "service_branch")
        vehicle_model_dropdown = self.driver.find_element(By.ID, "vehicle_model")
        service_type_radio = self.driver.find_element(By.XPATH, "//input[@name='service_type' and
@value='free']")
        service_date_input = self.driver.find_element(By.ID, "service_date")
        submit_button = self.driver.find_element(By.XPATH, "//button[@name='submit']")
        email_input.send_keys("Sisiramohan2000@gmail.com")
        driver_number_input.send_keys("9234567890")
        vehicle_number_input.send_keys("KL11234")
        service_branch_dropdown.send_keys("Trivandrum")
        vehicle_model_dropdown.send_keys("Dost+")
        service_type_radio.click()
        service_date_input.send_keys("5-12-2023")
        # Submit the form
        submit_button.click()
        time.sleep(10)
        # Add assertions for successful submission or subsequent page validation
    def tearDown(self):
        self.driver.close()
if __name__ == "__main__":
```

```
unittest.main()
```

OUTPUT



```
(env3) C:\Users\keert\OneDrive\Documents\project_backup\Miniproject\project>python manage.py test partsapp
Found 1 test(s).
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:52796/devtools/browser/837edc5e-087f-4785-a80b-00cd6b853bbf
[24772:12008:1204/231545.968:ERROR:device_event_log_impl.cc(192)] [23:15:45.968] USB: usb_service_win.cc:415 Could not read device interface GUIDs:
The system cannot find the file specified. (0x2)
.
----------------------------------------------------------------------
Ran 1 test in 66.962s

OK

(env3) C:\Users\keert\OneDrive\Documents\project_backup\Miniproject\project>
```

Test Case 2

Project Name: Commercial vehicle service and Spare parts Management					
Login Test Case					
Test Case ID: Test_1			**Test Designed By: Keerthi u**		
Test Priority(Low/Medium/High):High			**Test Designed Date: 4/12/2023**		
Module Name: booking page			**Test Executed By :** RINI KURIAN		
Test Title : Verify booking details			**Test Execution Date: 5/12/2023**		
Description: Testing the booking					
Pre-Condition : User has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigation to Login page		Login Page should be displayed	Login Page displayed	Pass
2	Provide valid username	Username: Keerthi	User should be able to login	User logged in and navigated to dashboard	Pass
3	Provide valid password	Keerthi@1			
4	Click on Login Button				
5	Provide Invalid username or password	Username: keerthi1 Password: keerthi123	User should not be able in Login	Message for enter valid email id or password displayed	Pass
6	Provide null username or password	username:null Password: null			
7	Click on Login In button				
Post-Condition: Post-Condition: User is validated with database and login to the website. The account session details are logged in database					

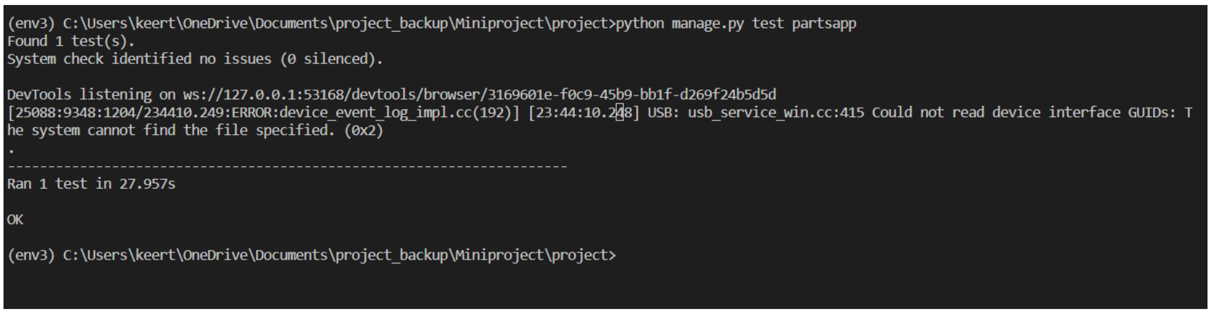
Test Case 3:

Code

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
import time

class ChangePasswordTest(unittest.TestCase):
def setUp(self):
self.driver = webdriver.Chrome()
self.driver.get("http://127.0.0.1:8000/login")
time.sleep(5)

def test_change_password(self):
# Login process (similar to your previous test)
username_input = self.driver.find_element(By.ID, "username")
password_input = self.driver.find_element(By.ID, "password")
login_button = self.driver.find_element(By.ID, "submitBtn")

username_input.send_keys("Sisira12")
password_input.send_keys("Sisira@12")
login_button.click()

time.sleep(5)
self.assertEqual(self.driver.current_url, 'http://127.0.0.1:8000/')

# After successful login, navigate to change password page
self.driver.get("http://127.0.0.1:8000/change_password/")  # Replace with the actual change
password URL
time.sleep(5)

# Fill in the password change form
current_password_input = self.driver.find_element(By.ID, "current_password")
new_password_input = self.driver.find_element(By.ID, "new_password")
```

```
confirm_new_password_input = self.driver.find_element(By.ID, "confirm_new_password")
change_password_button = self.driver.find_element(By.XPATH, "//button[contains(text(),
'Change Password')]")

current_password_input.send_keys("Sisira@12")
new_password_input.send_keys("Sisira@123")
confirm_new_password_input.send_keys("Sisira@123")

# Submit the form to change the password
change_password_button.click()

time.sleep(5)
# Add assertions for successful password change or subsequent page validation

def tearDown(self):
self.driver.close()

if __name__ == "__main__":
unittest.main()
```

OUTPUT



```
(env3) C:\Users\keert\OneDrive\Documents\project_backup\Miniproject\project>python manage.py test partsapp
Found 1 test(s).
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:53168/devtools/browser/3169601e-f0c9-45b9-bb1f-d269f24b5d5d
[25088:9348:1204/234410.249:ERROR:device_event_log_impl.cc(192)] [23:44:10.248] USB: usb_service_win.cc:415 Could not read device interface GUIDs: T
he system cannot find the file specified. (0x2)
.
----------------------------------------------------------------------
Ran 1 test in 27.957s

OK

(env3) C:\Users\keert\OneDrive\Documents\project_backup\Miniproject\project>
```

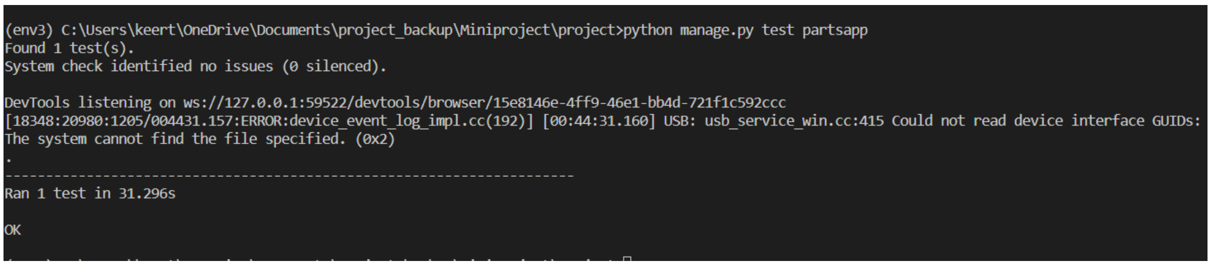
Test Case 4:

Code

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
import time

class AddToCartTest(unittest.TestCase):
def setUp(self):
self.driver = webdriver.Chrome()
self.driver.get("http://127.0.0.1:8000/login")
time.sleep(5)

def test_add_to_cart(self):
# Login process (similar to your previous test)
username_input = self.driver.find_element(By.ID, "username")
password_input = self.driver.find_element(By.ID, "password")
login_button = self.driver.find_element(By.ID, "submitBtn")

username_input.send_keys("Sisira12")
password_input.send_keys("Sisira@123")
login_button.click()

time.sleep(5)
self.assertEqual(self.driver.current_url, 'http://127.0.0.1:8000/')

# After successful login, navigate to the specific part or product page
self.driver.get("http://127.0.0.1:8000/partsorder/Dost+/")  # Replace with the actual product URL
time.sleep(5)

# Find and click the "Add to Cart" button
add_to_cart_buttons = self.driver.find_elements(By.CLASS_NAME, "add-to-cart-button")
add_to_cart_buttons[0].click()  # Click the first "Add to Cart" button
time.sleep(5)
```

```
# Navigate to the cart view explicitly
self.driver.get("http://127.0.0.1:8000/view_cart")  # Replace with the actual cart URL
time.sleep(5)

# Add assertions to validate the presence of the added item in the cart or any other cart-related validations

def tearDown(self):
self.driver.close()

if __name__ == "__main__":
unittest.main()
```

OUTPUT



```
(env3) C:\Users\keert\OneDrive\Documents\project_backup\Miniproject\project>python manage.py test partsapp
Found 1 test(s).
System check identified no issues (0 silenced).

DevTools listening on ws://127.0.0.1:59522/devtools/browser/15e8146e-4ff9-46e1-bb4d-721f1c592ccc
[18348:20980:1205/004431.157:ERROR:device_event_log_impl.cc(192)] [00:44:31.160] USB: usb_service_win.cc:415 Could not read device interface GUIDs:
The system cannot find the file specified. (0x2)
.
----------------------------------------------------------------------
Ran 1 test in 31.296s

OK
```

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one. At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion. Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover. The implementation state involves the following tasks: Careful planning. Investigation of system and constraints. Design of methods to achieve the changeover

6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that: The active user must be aware of the benefits of using the new system. Their confidence in the software is built up. Proper guidance is imparted to the user so that he is comfortable in using the application. Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won’t take place.

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 thencover 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 an important aspect in the software development life cycle. The need for system maintenance is for it to make E-Store Management System 64 Amal Jyothi College of Engineering, Kanjirappally Department of Computer Applications adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes"

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The abstract paints a clear picture of the inefficiencies in the current system and proposes an innovative solution. By leveraging technology, the project aims to enhance accessibility and convenience for customers, bridging the gap between traditional physical interactions and the evolving digital landscape. The focus on a seamless user experience signals an intent to prioritize customer satisfaction and streamline the entire process.

7.2 FUTURE SCOPE

Looking ahead, the future scope of this initiative appears promising. The envisioned online platform could potentially evolve beyond spare parts procurement and service bookings. It might integrate features like personalized recommendations, predictive maintenance, or real-time inventory updates. Expanding services to include user accounts, loyalty programs, or customer support portals could further enhance user engagement and retention.

Additionally, the platform might explore incorporating AI-driven functionalities for smarter part recommendations or automated service scheduling. Embracing emerging technologies like IoT (Internet of Things) for vehicle diagnostics or AR/VR (Augmented Reality/Virtual Reality) for interactive experiences could push the boundaries of customer engagement and service quality.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- PankajJalote, “Software engineering: a precise approach”
- Gary B. Shelly, Harry J. Rosenblatt, “System Analysis and Design”, 2009
- Ken Schwaber, Mike Beedle, Agile Software Development with Scrum, Pearson(2008)
- Roger S Pressman, “Software Engineering”
- IEEE Std 1016 Recommended Practice for Software Design Descriptions

WEBSITES:

- www.w3schools.com
- www.geeksforgeeks.com
- www.jquery.com
- https://chat.openai.com/chat

CHAPTER 9
APPENDIX

9.1 Sample Code

Login.html

```
{% load static %}
{% load socialaccount %}
<!DOCTYPE html>
<html lang="en">
<head>
<link rel="icon" type="image/x-icon" href="favicon.ico">

<!-- Bootstrap CSS -->
<link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
<link rel="stylesheet" href="{% static 'css/owl.carousel.min.css' %}">
<link rel="stylesheet" href="{% static 'css/owl.theme.default.min.css' %}">
<link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css' rel='stylesheet'>
<link rel="stylesheet" href="{% static 'css/style.css' %}">
<link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
<!-- Design by foolishdeveloper.com -->
{% comment %} <title>Glassmorphism login Form Tutorial in html css</title>

<link rel="preconnect" href="https://fonts.gstatic.com">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css">
<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&display=swap"
rel="stylesheet">
<!--Stylesheet--> {% endcomment %}
<style>
body {
/* Set background properties for the entire page */
background-image: url({% static 'img/Untitled_design.jpg' %}); /* Replace
'path_to_your_image.jpg' with the actual path to your image */
background-size: cover; /* Adjust the background size */
/* Add other necessary properties */
}
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-white sticky-top">
<div class="container">
<a class="navbar-brand" href="#">Sparx motors<span class="dot">.</span></a>
<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav"
aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
<ul class="navbar-nav ms-auto">
{% if user.is_authenticated %}
<li class="nav-item">
<a class="nav-link" href="#home">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#about"></a>
</li>
<li class="nav-item">
<a class="nav-link" href="#services">vehicles</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#portfolio">Services</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#team">Spare parts</a>
</li>
{% else %}
<li class="nav-item">
<a class="nav-link" href="{% url 'login' %}">signin</a>
</li>
```

```
<li class="nav-item">
<a class="btn btn-brand" href="{% url 'signup' %}">signup</a>
</li>
{% endif %}
</ul>

</div>
</div>
</nav>
{% if messages %}
<div class="container">
{% for message in messages %}
<div class="alert alert-{{ message.tags }} alert-dismissible fade show" role="alert">
{{ message }}
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
<span aria-hidden="true">&times;</span>
</button>
</div>
{% endfor %}
</div>
{% endif %}
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
<form action="" method="POST">
{% csrf_token %}
<h2 align="center"><b>Login</b></h2>

<label for="username">Username</label>
<input type="text" name="username" placeholder="Username" id="username" required>
<span id="usernameError" style="color: red;"></span>

<label for="password">Password</label>
<input type="password" name="password" placeholder="Password" id="password" required>
<span id="passwordError" style="color: red;"></span>
```

```
<button type="submit" id="submitBtn" style="background-color: #007bff; color:
white;">Login</button>
{% if error_message %}
<div style="color: red;">
{{ error_message }}
</div>
{% endif %}

<a href="{% provider_login_url 'google' %}" class="google-login-button"><i class="fab fa-
google"></i> Google</a>

<p>Don't have an account? <a href="{% url 'signup' %}">Sign Up</a></p>
<p class="forgot-password"><a href="{% url 'password_reset' %}">Forgot Password?</a></p>
</div>

</form>

</body>
</html>
```

Booking.html

```
{% load static %}
<!doctype html>
<html lang="en">

<head>
   <!-- Required meta tags -->
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="icon" type="image/x-icon" href="favicon.ico">

   <!-- Bootstrap CSS -->
   <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
   <link rel="stylesheet" href="{% static 'css/owl.carousel.min.css' %}">
   <link rel="stylesheet" href="{% static 'css/owl.theme.default.min.css' %}">
   <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css' rel='stylesheet'>
   <link rel="stylesheet" href="{% static 'css/style.css' %}">

   <title>sparx motors light vehicles</title>
```

```
</head>

<body data-bs-spy="scroll" data-bs-target=".navbar" data-bs-offset="70">


  <!-- TOP NAV -->
  <div class="top-nav" id="home">
    <div class="container">
      <div class="row justify-content-between">
        <div class="col-auto">
          <p> <i class='bx bxs-envelope'></i> keerthi@gmail.com</p>
          <p> <i class='bx bxs-phone-call'></i> 9947601695</p>
        </div>
        <div class="col-auto social-icons">
          <a href="#"><i class='bx bxl-facebook'></i></a>
          <a href="#"><i class='bx bxl-twitter'></i></a>
          <a href="#"><i class='bx bxl-instagram'></i></a>
          <a href="#"><i class='bx bxl-pinterest'></i></a>
        </div>
      </div>
    </div>
  </div>

  <!-- BOTTOM NAV -->
  <nav class="navbar navbar-expand-lg navbar-light bg-white sticky-top">
    <div class="container">
      <a class="navbar-brand" href="#">Sparx motors<span class="dot">.</span></a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav"
        aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ms-auto">
        {% if user.is_authenticated %}
          <li class="nav-item">
            <a class="nav-link" href="#home">Home</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#about">vehicles</a>
          </li>

          <li class="nav-item">
            <a class="nav-link" href="{% url 'booking' %}">Services</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="partsorder.html">Spare parts</a>
          </li>
          <li class="nav-item" style="display: flex; align-items: center;">
            <div id="user-dropdown" class="user-profile">
              {% if request.session.profile_image_updated %}
```

```
                        <!-- Use the updated profile image -->
                        <img src="{{ user_profile.image.url }}" alt="" class="profile-image">
                    {% else %}
                        <!-- Use the default image -->
                        <img src="{% static 'img/img2.png' %}" alt="" class="profile-image">
                    {% endif %}
                    <strong style="font-weight: bold;">{{ user.username }}</strong>
                    <div id="user-dropdown-content" class="dropdown" style="position: absolute;
top: 100%; left: 0; display: none;">
                        <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
                            <a class="dropdown-item" href="">My Profile</a>
                            <a class="dropdown-item" href="{% url 'edit_profile' %}">Edit
Profile</a>
                            <a class="dropdown-item" href="{% url 'logout' %}"
onclick="confirmLogout()">Logout</a>
                            {% csrf_token %}
                        </div>
                    </div>
                </div>
            </li>
            {% if user.is_authenticated %}
            <div class="cart-icon">
            <a href="{% url 'view_cart' %}" class="view-cart-link">
            <img src="{% static 'img/cart.png' %}" alt="Cart">
        </a>
            {% if cart_count == 0 %}
                <span></span>
            {% else %}
             <span class="cart-count">{{ cart_count }}</span>
            {% endif %}
        </div>
        {% endif %}
            {% comment %} <li class="nav-item">
                <a class="btn btn-outline-light ms-3" href="{% url 'logout' %}"
onclick="confirmLogout()">Logout</a>
            </li> {% endcomment %}
            <script>
                function confirmLogout() {
                    if (confirm("Are you sure you want to log out?")) {
                        window.location.href = "{% url 'logout' %}";  // Redirect to the logout URL
                    }
                }


                const userDropdown = document.getElementById("user-dropdown");
                const userDropdownContent = document.getElementById("user-dropdown-
content");

                userDropdown.addEventListener("click", () => {
                    userDropdownContent.style.display = userDropdownContent.style.display ===
"block" ? "none" : "block";
```

```
                });
                document.addEventListener("DOMContentLoaded", function () {
                    // Disable the back button
                    window.history.pushState(null, null, location.href);
                    window.onpopstate = function () {
                        window.history.go(0);
                    };
                });

            </script>



            {% else %}

            <li class="nav-item">
                <a class="btn btn-brand" href="{% url 'login' %}">signin</a>
            </li>

            <li class="nav-item">
                <a class="btn btn-outline-light ms-3" href="{% url 'signup' %}">signup</a>
            </li>

        {% endif %}
        </ul>

      </div>
    </div>
  </nav>
<!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>User Service Booking</title>
    <style>
  {% comment %} booking table {% endcomment %}
  <title>User Service Booking</title>
  <!-- Link to your CSS file -->
</head>
<body>
  <h2 align="center">Service Booking</h2>
<form method="POST" action="{% url 'booking' %}">
  {% csrf_token %}



  <!-- User Information -->
  <div class="form-group">
    <label for="username">Customer Name:</label>
    <input type="text" id="username" name="username" value="{{ request.user.username }}"
readonly>
```

```
</div>

<div class="form-group">
  <label for="email">Email:</label>
  <input type="text" id="email" name="email" required>
  <span id="emailError" style="color: red;"></span>
</div>

<div class="form-group">
  <label for="driver_number">Driver Number:</label>
  <input type="text" id="driver_number" name="driver_number" required>
</div>

<div class="form-group">
  <label for="vehicle_number">Vehicle Number:</label>
  <input type="text" id="vehicle_number" name="vehicle_number" required>
</div>

<!-- Service Details -->
<div class="form-group">
  <label for="service_branch">Service Branch:</label>
  <select id="service_branch" name="service_branch" required>
    <option value=""></option>
    <option value="branch1">Trivandrum</option>
    <option value="branch2">Kollam</option>
    <option value="branch3">Alappuzha</option>
    <option value="branch4">Kottayam</option>
    <option value="branch5">Eranakulam</option>
    <option value="branch6">Idukki</option>
    <!-- Add more options as needed -->
  </select>
</div>

<div class="form-group">
  <label for="vehicle_model">Vehicle Model:</label>
  <select id="vehicle_model" name="vehicle_model" placeholder="vehiclemodel" required>
    <option value=""></option>
    <option value="Dost+">Dost+</option>
    <option value="Dost strong">Dost strong</option>
    <option value="Partner">Partner</option>
    <option value="Dost CNG">Dost CNG</option>
    <option value="MITR Staff Bus">MITR Staff Bus</option>
    <option value="Dost Express">Dost Express</option>
    <option value="Bada Dost i3">Bada Dost i3</option>
    <!-- Add more options as needed -->
  </select>
</div>

<div class="form-group">
  <label>Service Type:</label>
  <label><input type="radio" name="service_type" value="free" required> Free
```

```
Service</label>
        <label><input type="radio" name="service_type" value="paid" required> Paid
Service</label>
        <label><input type="radio" name="service_type" value="running" required> Running
Service</label>
        <label><input type="radio" name="service_type" value="accident" required> Accident
Service</label>
    </div>

    <div class="form-group">
        <label for="service_date">Service Date:</label>
        <input type="date" id="service_date" name="service_date" required>
    </div>

    <button type="submit" name="submit">Submit</button>
</form>

    <script>
        {% comment %} window.onload = function() {
            document.getElementById('driver_number').addEventListener('input', function() {
                // Validate driver number to be exactly 10 digits
                let driverNumber = this.value;
                if (driverNumber.length !== 10 || isNaN(driverNumber)) {
                    this.setCustomValidity('Driver number should be a 10-digit number.');
                } else {
                    this.setCustomValidity('');
                }
            });

            document.getElementById('vehicle_number').addEventListener('blur', function() {
                // Check vehicle number uniqueness (validate against database via API)
                let vehicleNumber = this.value;
                // Make an API call to check uniqueness, handle accordingly
            });

            document.getElementById('service_date').addEventListener('input', function() {
                // Prevent selection of past dates
                let currentDate = new Date();
                let selectedDate = new Date(this.value);
                if (selectedDate < currentDate) {
                    this.setCustomValidity('Please select a date from today onwards.');
                } else {
                    this.setCustomValidity('');
                }
            });
        };
    </script>
     {% endcomment %}
</body>
</html>
```

Partsorder.html

```
{% load static %}
<!doctype html>
<html lang="en">

<head>
   <!-- Required meta tags -->
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <link rel="icon" type="image/x-icon" href="favicon.ico">

   <!-- Bootstrap CSS -->
   <link rel="stylesheet" href="{% static 'css/bootstrap.min.css' %}">
   <link rel="stylesheet" href="{% static 'css/owl.carousel.min.css' %}">
   <link rel="stylesheet" href="{% static 'css/owl.theme.default.min.css' %}">
   <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css' rel='stylesheet'>
   <link rel="stylesheet" href="{% static 'css/style.css' %}">

   <title>sparx motors light vehicles</title>
   <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css"
rel="stylesheet">
</head>

<body data-bs-spy="scroll" data-bs-target=".navbar" data-bs-offset="70">

   <!-- TOP NAV -->
   <div class="top-nav" id="home">
      <div class="container">
         <div class="row justify-content-between">
            <div class="col-auto">
               <p> <i class='bx bxs-envelope'></i> keerthi@gmail.com</p>
               <p> <i class='bx bxs-phone-call'></i> 9947601695</p>
            </div>
            <div class="col-auto social-icons">
               <a href="#"><i class='bx bxl-facebook'></i></a>
               <a href="#"><i class='bx bxl-twitter'></i></a>
               <a href="#"><i class='bx bxl-instagram'></i></a>
               <a href="#"><i class='bx bxl-pinterest'></i></a>
            </div>
         </div>
      </div>
   </div>

   <!-- BOTTOM NAV -->
   <nav class="navbar navbar-expand-lg navbar-light bg-white sticky-top">
      <div class="container">
         <a class="navbar-brand" href="#">Sparx motors<span class="dot">.</span></a>
         <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav"
```

```
            aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav ms-auto">
            {% if user.is_authenticated %}
                <li class="nav-item">
                    <a class="nav-link" href="#home">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="#about">vehicles</a>
                </li>

                <li class="nav-item">
                    <a class="nav-link" href="{% url 'booking' %}">Services</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="partsorder.html">Spare parts</a>
                </li>
                <li class="nav-item" style="display: flex; align-items: center;">
                    <div id="user-dropdown" class="user-profile">
                        {% if request.session.profile_image_updated %}
                            <!-- Use the updated profile image -->
                            <img src="{{ user_profile.image.url }}" alt="" class="profile-image">
                        {% else %}
                            <!-- Use the default image -->
                            <img src="{% static 'img/img2.png' %}" alt="" class="profile-image">
                        {% endif %}
                        <strong style="font-weight: bold;">{{ user.username }}</strong>
                        <div id="user-dropdown-content" class="dropdown" style="position: absolute;
top: 100%; left: 0; display: none;">
                            <div class="dropdown-menu" aria-labelledby="dropdownMenuButton">
                                <a class="dropdown-item" href="">My Profile</a>
                                <a class="dropdown-item" href="{% url 'edit_profile' %}">Edit
Profile</a>
                                <a class="dropdown-item" href="{% url 'logout' %}"
onclick="confirmLogout()">Logout</a>
                                {% csrf_token %}
                            </div>
                        </div>
                    </div>
                </li>

                {% if user.is_authenticated %}
                <div class="cart-icon">
                <a href="{% url 'view_cart' %}" class="view-cart-link">
                <img src="{% static 'img/cart.png' %}" alt="Cart">
            </a>
                {% if cart_count == 0 %}
                    <span></span>
                {% else %}
```

```
                 <span class="cart-count">{{ cart_count }}</span>
                {% endif %}
            </div>
            {% endif %}
                {% comment %} <li class="nav-item">
                    <a class="nav-link" href="{% url 'view_cart' %}">
                        <i class="fas fa-shopping-cart"></i> <!-- Assuming you're using Font Awesome
for icons -->
                        Cart
                    </a>
                </li> {% endcomment %}
                {% comment %} <li class="nav-item">
                    <a class="btn btn-outline-light ms-3" href="{% url 'logout' %}"
onclick="confirmLogout()">Logout</a>
                </li> {% endcomment %}
                <script>
                    function confirmLogout() {
                        if (confirm("Are you sure you want to log out?")) {
                            window.location.href = "{% url 'logout' %}";  // Redirect to the logout URL
                        }
                    }


                    const userDropdown = document.getElementById("user-dropdown");
                    const userDropdownContent = document.getElementById("user-dropdown-
content");

                    userDropdown.addEventListener("click", () => {
                        userDropdownContent.style.display = userDropdownContent.style.display ===
"block" ? "none" : "block";
                    });
                    document.addEventListener("DOMContentLoaded", function () {
                        // Disable the back button
                        window.history.pushState(null, null, location.href);
                        window.onpopstate = function () {
                            window.history.go(0);
                        };
                    });

                </script>




                {% else %}

                <li class="nav-item">
                    <a class="btn btn-brand" href="{% url 'login' %}">signin</a>
                </li>

                <li class="nav-item">
                    <a class="btn btn-outline-light ms-3" href="{% url 'signup' %}">signup</a>
                </li>

              {% endif %}
              </ul>

          </div>
        </div>
    </nav>
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Product Booking Dashboard</title>
    <link rel="stylesheet" href="{% static 'styles.css' %}">
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.0/css/all.min.css">
</head>
<body>

    <div class="search-container">
        <input type="text" id="search-bar" placeholder="Search products">
        <button id="search-button">Search</button>
    </div>
        {% comment %} <div id="suggestions" class="suggestions">
            <div class="suggestion">Light Vehicle</div>
            <div class="suggestion">Tracks</div>
            <div class="suggestion">Bus</div>
        </div> {% endcomment %}
    <!-- Add more filter options as needed -->
    </div>


        <section class="product-list" id="product-list">
            <!-- Sample product cards -->
            {% for part in parts_orders %}
                <div class="product-card">
                    <!-- Wishlist icon -->
                    {% comment %} <div class="wishlist-container">
                        <button class="add-to-wishlist-button" data-product-id="{{ part.id }}">
                            <i class="fas fa-heart"></i>
                        </button>
                    </div> {% endcomment %}

                    <!-- Product details -->
                    <div class="product-image">
                        <img src="{{ part.parts_image.url }}" alt="{{ part.partsname }} Image">
                    </div>
                    <div class="product-info">
                        <h2>{{ part.partsname }}</h2>
                        <p>{{ part.description }}</p>
                        <p>{{ part.price }}</p>

                        <form action="{% url 'add_to_cart' part.id %}" method="post">
                            {% csrf_token %}
                            <button class="add-to-cart-button">Add to Cart</button>
                        </form>

                        <a href="{% url 'view_cart' %}" class="add-to-cart-link">
                            <button class="buy-now-button" data-product-id="{{ part.id }}">Buy
Now</button>
                        </a>



                    </div>
                </div>
            {% empty %}
                <p>No parts orders yet.</p>
            {% endfor %}
        </section>
    </div>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script>
        $(document).ready(function() {
            // Function to filter products based on the search query
            function filterProducts(searchQuery) {
                const productList = document.getElementById('product-list');
                const products = productList.getElementsByClassName('product-card');

                for (let i = 0; i < products.length; i++) {
                    const productName = products[i].querySelector('h2').textContent.toLowerCase();
                    const displayStyle = productName.includes(searchQuery.toLowerCase()) ? 'block' :
'none';
                    products[i].style.display = displayStyle;
                }
            }

            // Event listener for the search button click
            $('#search-button').on('click', function() {
                const searchValue = $('#search-bar').val();
                filterProducts(searchValue);
            });

            // Event listener for typing into the search bar
            $('#search-bar').on('input', function() {
                const searchValue = $(this).val();
                filterProducts(searchValue);
            });
        });
       // Function to filter products based on the search query
function filterProducts(searchQuery) {
   const productList = document.getElementById('product-list');
   const products = productList.getElementsByClassName('product-card');

   for (let i = 0; i < products.length; i++) {
      const productName = products[i].querySelector('h2').textContent.toLowerCase();
      const displayStyle = productName.includes(searchQuery.toLowerCase()) ? 'block' : 'none';
      products[i].style.display = displayStyle;
   }
}

// Event listener for the search button click
document.getElementById('search-button').addEventListener('click', function() {
   const searchValue = document.getElementById('search-bar').value;
   filterProducts(searchValue);
});

// Event listener for typing into the search bar
document.getElementById('search-bar').addEventListener('input', function() {
   const searchValue = this.value;
   filterProducts(searchValue);
});

</body>
</html>
```

View_cart.html

```
{% load static %}
<html>
<head>
<style>
   {% comment %} .container {
      max-width: 800px;
      margin: 0 auto;
      padding: 20px;
      background-color: #f4f4f4;
      border-radius: 8px;
      box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
   } {% endcomment %}

   .cart-header {
      text-align: center;
      margin-bottom: 20px;
   }

    .cart-item {
      display: flex;
      justify-content: space-between;
      align-items: center;
      padding: 10px;
      border-bottom: 1px solid #ddd;
   }

   .cart-item-details {
      display: flex;
      align-items: center;
   }

    .cart-item-name {
      font-weight: bold;
      margin-right: 10px;
      flex: 1;
      white-space: nowrap;
      overflow: hidden;
      text-overflow: ellipsis;
   }

   .remove-from-cart-btn,
   .quantity-btn {
      background-color: #ff6347;
      color: white;
      border: none;
      border-radius: 6px;
      padding: 5px 10px;
      cursor: pointer;
      transition: background-color 0.3s;
   }

    .remove-from-cart-btn:hover,
   .quantity-btn:hover {
      background-color: #e74c3c;
   }

   .cart-item-quantity {
      display: flex;
      align-items: center;
      margin-right: 30px;
   }

    .cart-item-price {
      font-weight: bold;
   }

   .continue-shopping-link {
      display: inline-block;
      margin-top: 20px;
      text-decoration: none;
      color: #3498db;
      font-weight: bold;
```

```
   }

   .continue-shopping-link:hover {
      text-decoration: underline;
   }

   .checkout-button {
      display: inline-block;
      background-color: #28a745;
      color: white;
      padding: 10px 20px;
      border-radius: 4px;
      margin-top: 20px;
      margin-left: 400px;
      text-decoration: none;
      font-weight: bold;
      transition: background-color 0.3s;
   }

   .checkout-button:hover {
      background-color: #218838;
   }
</style>

<body>

<div class="container">

   <div class="cart-header">
      <h1>Your Cart</h1>
   </div>

   <ul>
      {% for item in cart_items %}
      <li class="cart-item">
         <div class="cart-item-details">
            <div class="cart-item-image">
               <img src="{{ item.product.parts_image.url }}" alt="{{ item.product.partsname }}
Image">
            </div>



         <div class="cart-item-quantity">
            <div class="cart-item-name">{{ item.product.partsname }}</div>
            <form action="{% url 'remove_from_cart' item.product.id %}" method="post">
               {% csrf_token %}
               <button class="remove-from-cart-btn" type="submit">Remove</button>
            </form>
         </div>
            <form action="{% url 'increase-cart-item' item.product.id %}" method="post">
               {% csrf_token %}
```

```
                <button class="quantity-btn increase-quantity" type="submit">+</button>
            </form>
            <span class="item-quantity">{{ item.quantity }}</span>
            <form action="{% url 'decrease-cart-item' item.product.id %}" method="post">
                {% csrf_token %}
                <button class="quantity-btn decrease-quantity" type="submit">-</button>
            </form>
        </div>
        <div class="cart-item-price" data-price="{{ item.product.price }}">
            ${{ item.product.price }}
        </div>


    </li>
    {% endfor %}
  </ul>
  <p class="total-price-data"><span id="total-price"></span></p>

  <a class="continue-shopping-link" href="{% url 'partsorder' %}">Continue Shopping</a>

  <a class="checkout-button" href="{% url 'checkout' %}">Checkout</a>
</div>
<script>
  function updateCartCount() {
  var cartCount = document.querySelector('.cart-count');
  fetch('{% url "fetch-cart-count" %}')
      .then(response => response.json())
      .then(data => {
          cartCount.textContent = data.cart_count;
      });
}

updateCartCount();
document.addEventListener("DOMContentLoaded", function () {

  updateTotalPrice();

const increaseButtons = document.querySelectorAll(".increase-quantity");
const decreaseButtons = document.querySelectorAll(".decrease-quantity");
const quantityElements = document.querySelectorAll(".item-quantity");
const priceElements = document.querySelectorAll(".cart-item-price");

increaseButtons.forEach((button, index) => {
  button.addEventListener("click", (event) => {
      event.preventDefault();
      const currentItem = event.target.closest(".cart-item");
      const quantityElement = currentItem.querySelector(".item-quantity");
      const priceElement = currentItem.querySelector(".cart-item-price");
      const pricePerItem = parseFloat(priceElement.getAttribute("data-price"));
      const currentQuantity = parseInt(quantityElement.textContent);
```

```
        quantityElement.textContent = currentQuantity + 1;
        updateCartItemPrice(priceElement, pricePerItem, currentQuantity + 1);
        updateTotalPrice();
    });
});

decreaseButtons.forEach((button, index) => {
    button.addEventListener("click", (event) => {
        event.preventDefault();
        const currentItem = event.target.closest(".cart-item");
        const quantityElement = currentItem.querySelector(".item-quantity");
        const priceElement = currentItem.querySelector(".cart-item-price");
        const pricePerItem = parseFloat(priceElement.getAttribute("data-price"));
        const currentQuantity = parseInt(quantityElement.textContent);

        if (currentQuantity > 1) {
            quantityElement.textContent = currentQuantity - 1;
            updateCartItemPrice(priceElement, pricePerItem, currentQuantity - 1);
            updateTotalPrice();
        }
    });
});

function updateCartItemPrice(priceElement, pricePerItem, quantity) {
    const totalPrice = (pricePerItem * quantity).toFixed(2);
    priceElement.textContent = "$" + totalPrice;
}

function updateTotalPrice() {
    const itemElements = document.querySelectorAll(".cart-item");

    itemElements.forEach(function (itemElement) {
        const price = parseFloat(itemElement.querySelector(".cart-item-price").getAttribute("data-
price"));
        const quantity = parseInt(itemElement.querySelector(".cart-item-quantity").textContent);
        total += price * quantity;
    });

    const totalPriceElement = document.getElementById("total-price");
    totalPriceElement.textContent = total.toFixed(2);
}
});

</script>
</body>
</html>
```

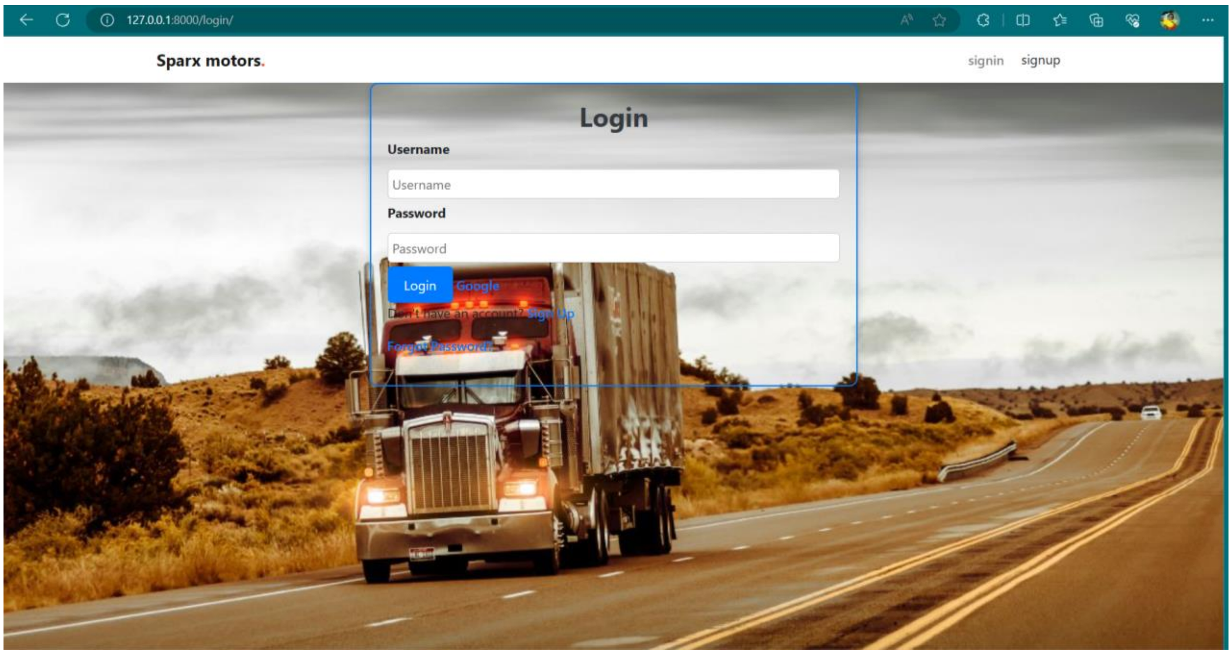
Checkout.html

```
<body>

<div class="container">
   <div id="order-placed-section" style="display: none;">
      <h2>Order Placed</h2>
      <p id="order-success-message"></p>
   </div>
   <P><a href="{% url 'bill_invoice'  %}" target="_blank">View Invoice</a></P>
   <h1>Checkout</h1>
   <p>Order Summary :</p>
   <ul>

      {% for item in cart_items %}
      <li><img src="{{ item.product.parts_image.url }}" alt="{{ item.product.partsname }} Image"></li>
      <li>Total quantity:{{ item.product.partsname }} x {{ item.quantity }}</li>
      {% endfor %}
   </ul>
   <p>Total Amount: &#8377<span id="total_item_amount">{{total_amount}}</span></p>

   <button id="rzp-button1">Payment</button>
</div>
<div id="checkout-data" data-email="{{ email }}" data-fullname="{{ full_name }}">
</div>
<script>
   var handlePaymentUrl = "{% url 'handle-payment' %}";
</script>
<script src="https://checkout.razorpay.com/v1/checkout.js"></script>
<script src="{% static 'js/checkout.js' %}"></script>
<script >
   document.addEventListener('DOMContentLoaded', function () {
      const checkoutButton = document.getElementById('rzp-button1');
      const checkoutData = document.getElementById("checkout-data");
      const email = checkoutData.getAttribute("data-email");
      const fullName = checkoutData.getAttribute("data-fullname");
      const totalItemAmount = document.getElementById("total_item_amount");

      function getCookie(name) {
         let value = "; " + document.cookie;
         let parts = value.split("; " + name + "=");
         if (parts.length === 2) return parts.pop().split(";").shift();
      }

      checkoutButton.addEventListener('click', function (event) {
         event.preventDefault();

         const rawPrice = totalItemAmount.textContent;
         const totalPrice = parseFloat(rawPrice.replace('₹', '').trim());

         fetch("http://127.0.0.1:8000/create-order/", {
            method: 'POST',
            headers: {
               'Content-Type': 'application/json',
               'X-CSRFToken': getCookie('csrftoken')
```

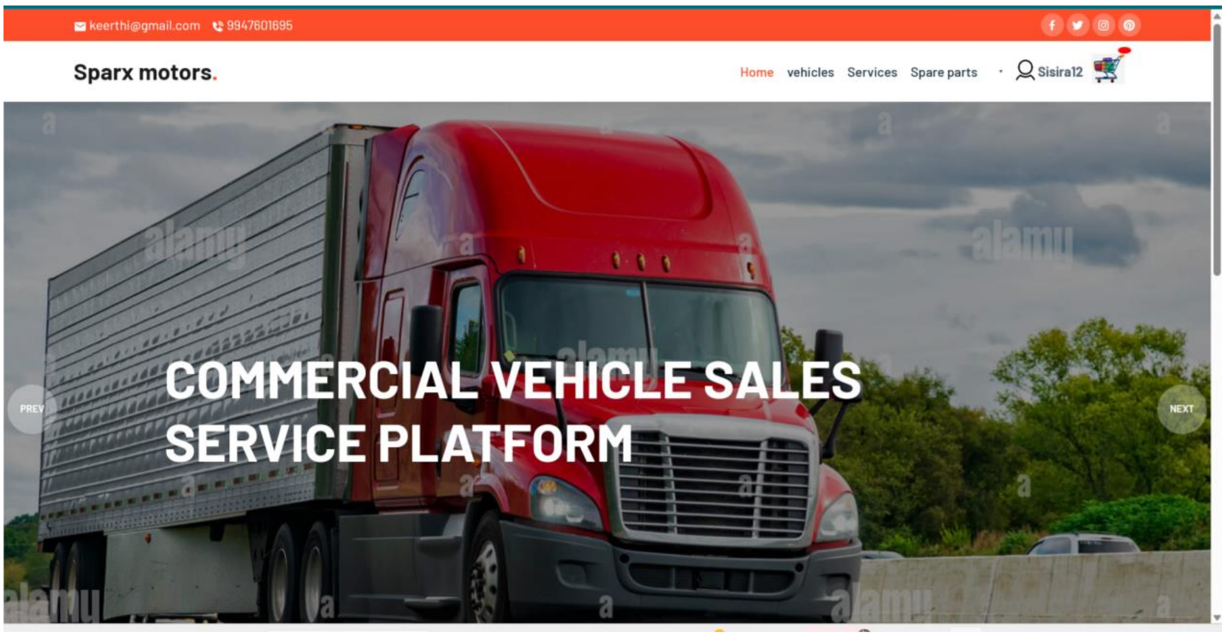
```
        }
    })
        .then(response => {
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            return response.json();
        })
        .then(data => {
            var options = {
                "key": "rzp_test_wbkpfdufxSi09S",
                "amount": totalPrice * 100,
                "currency": "INR",
                "name": "Ecommerce",
                "description": "Order Payment",
                "order_id": data['order_id'],
                "prefill": {
                    "name": fullName,
                    "email": email
                },
                "handler": function (response) {
                    const razorpay_order_id = response.razorpay_order_id;
                    const payment_id = response.razorpay_payment_id;
                    console.log(razorpay_order_id, payment_id)

                    fetch("http://127.0.0.1:8000/handle-payment/", {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json',
                            'X-CSRFToken': getCookie('csrftoken')
                        },
                        body: JSON.stringify({
                            'order_id': razorpay_order_id,
                            'payment_id': payment_id
                        })
                    })
                        .then(response => {
                            if (!response.ok) {
                                throw new Error('Network response was not ok during payment handling');
                            }
                            return response.json();
                        })
                        .then(data => {
                            if (data.message === 'Payment successful') {
                                const orderSection = document.getElementById('order-placed-section');
                                const orderMessage = document.getElementById('order-success-message');

                                orderMessage.textContent = "Successfully placed order!";
                                orderSection.style.display = "block";
                            } else {
                                alert('Payment failed');
                            }
                        })
                        .catch(error => {
                            console.error('An error occurred while processing the payment.', error);
                        alert('There was an issue processing your payment. Please try again.');
                    });
                }
            };

            var rzp1 = new Razorpay(options);
            rzp1.open();
        })
        .catch(error => {
            console.error('Error creating order:', error);
            alert('There was an issue initiating your order. Please try again.');
        });
    });
  });



</script>
</body>
</html>
```

9.1 Screen Shots

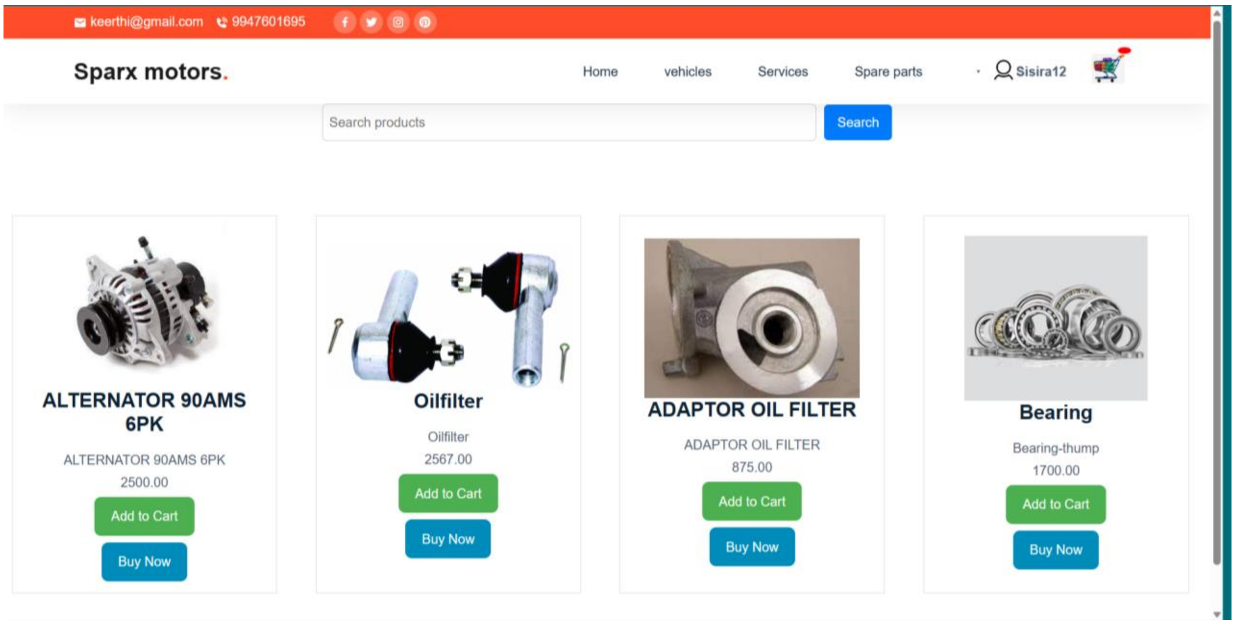
9.1.1 Login page



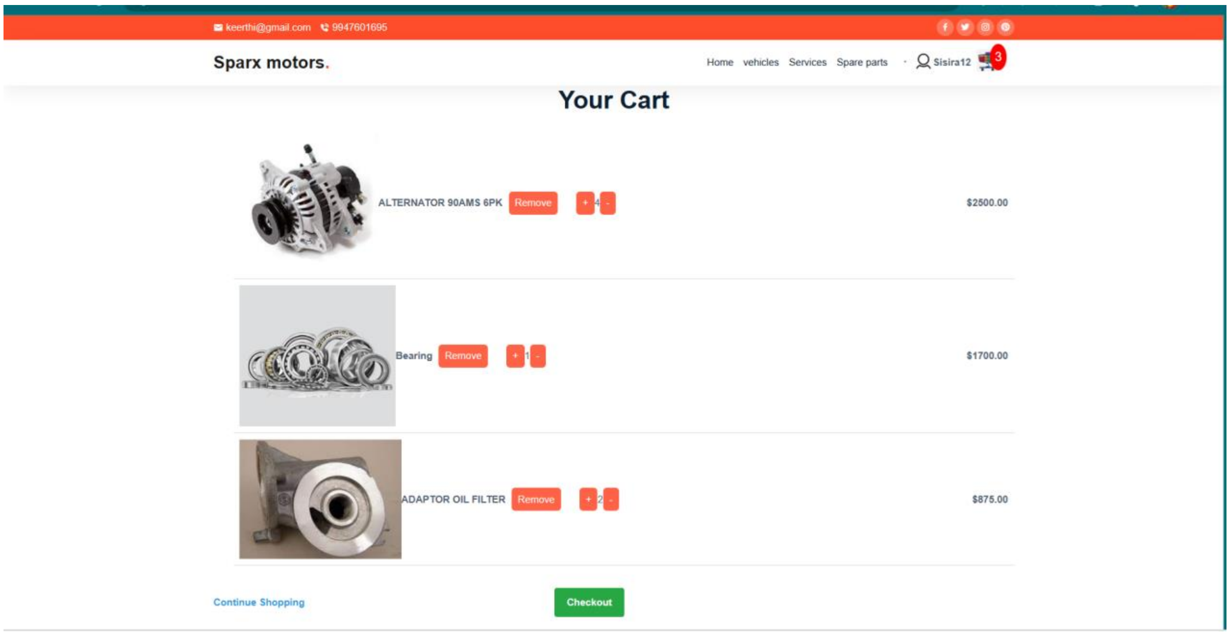
9.1.2 index page



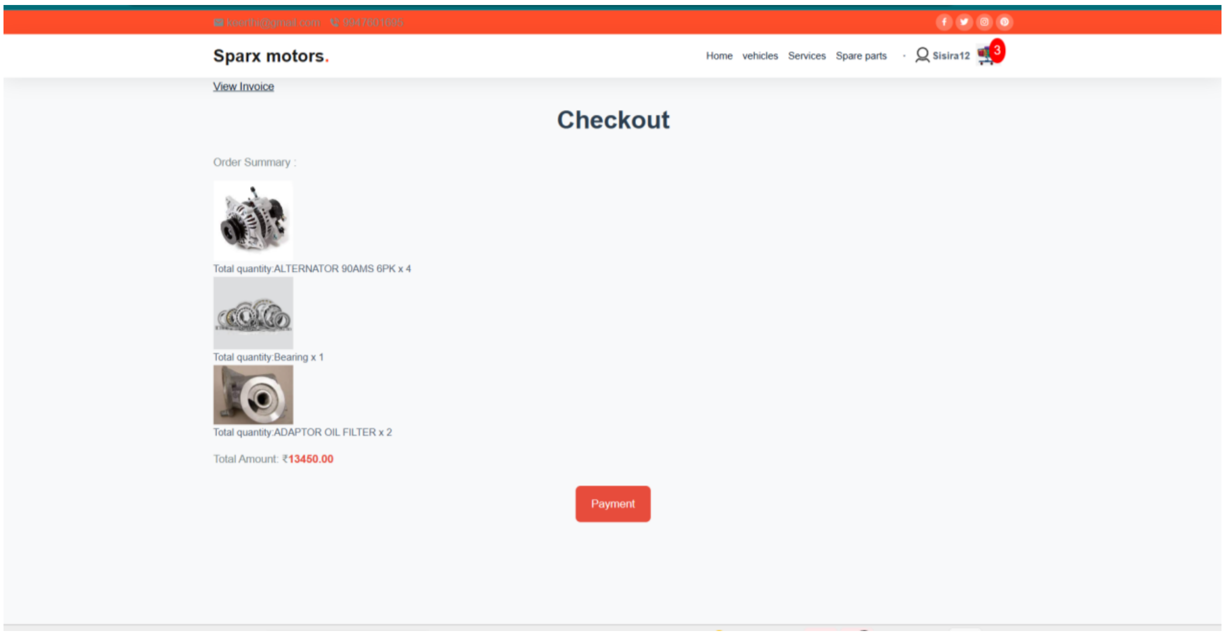
9.1.3 spare parts page



9.1.4 cart page



9.1.5 checkout page



Checkout

Order Summary :

Total quantity:ALTERNATOR 90AMS 6PK x 4

Total quantity:Bearing x 1

Total quantity:ADAPTOR OIL FILTER x 2

Total Amount: ₹13450.00