# CHAPTER 1

# INTRODUCTION

## 1.1 OVERVIEW OF THE PROJECT

The Online Tyre Shopping Platform for Sakthi Tyres aims to revolutionize the tyre business by transitioning from manual processes to an efficient digital platform. Built with the MERN stack (MongoDB, Express, React, Node), this web application offers features like Tyre Management, Customer Booking, Billing, and User Authentication to streamline operations and enhance customer experience. With React.JS providing a user-friendly interface, customers can easily search for tyres across brands and make bookings online. The Node.JS and Express.JS back-end ensures smooth server-side performance, while MongoDB manages tyre listings and customer data. This platform boosts productivity, reduces delays, and offers scalability for future growth.

## 1.2 PROBLEM DEFINITION

In the traditional tyre retail industry, many businesses struggle with inefficiencies caused by manual processes and fragmented systems. This often leads to delays in customer service, challenges in managing inventory, and difficulty tracking customer preferences and maintenance needs. Additionally, the lack of an integrated online platform limits the ability of tyre shops to offer convenient services like online tyre browsing, booking, and secure payment options. As a result, businesses miss opportunities to enhance customer satisfaction and streamline their operations.

The goal of this project is to develop the Online Tyre Shopping Platform for Sakthi Tyres using the MERN stack. This platform will address these challenges by providing an efficient, scalable, and user-friendly solution that integrates essential features such as Tyre Management, Customer Booking, Billing, and User Authentication. By offering a seamless online shopping experience, this web application will help Sakthi Tyres improve operational

efficiency, reduce manual errors, and enhance customer engagement, positioning the business for future growth in a competitive market

## 1.3 OBJECTIVE OF THE PROJECT

The objective of this project is to develop the Online Tyre Shopping Platform for Sakthi Tyres, aimed at modernizing the tyre business and improving both operational efficiency and customer experience. By utilizing the MERN stack (MongoDB, Express, React, Node), the platform will ensure high performance, scalability, and security. It will offer key features such as Tyre Management, Customer Booking, Billing, and User Authentication, providing users with an easy and intuitive way to browse, compare, and purchase tyres from various brands. The platform will also enhance customer engagement by enabling direct online bookings and offering a seamless shopping experience. Additionally, it will include features like maintenance tracking and reporting to help the business manage its operations more efficiently.

The project further aims to improve Sakthi Tyres online visibility and customer reach by offering a user-friendly, secure, and feature-rich platform, ensuring the business can cater to both existing and new customers with ease. Overall, the objective is to create a digital solution that not only supports the company’s growth but also positions Sakthi Tyres as a modern, competitive player in the tyre retail industry.

# CHAPTER 2

# SYSTEM ANALYSIS

## 2.1 EXISTING SYSTEM

The current system at Sakthi Tyres relies on manual processes and in-person transactions, leading to inefficiencies in managing tyre inventory, bookings, and customer service. Customers must visit or call the store to inquire about tyre availability and pricing, making the experience inconvenient and time-consuming. Tyre information is shared verbally or through printed materials, limiting the shop's ability to reach a wider audience. Additionally, the lack of a centralized platform for managing listings, tracking customer interactions, and handling online bookings results in poor record-keeping and delayed support. This underscores the need for a digital solution to streamline operations and enhance the overall customer experience.

### 2.1.1 DISADVANTAGES OF EXISTING SYSTEM

The existing system has following disadvantages:

- Manual and inefficient tyre booking process.
- Inconvenient for customers who must visit or call for information.
- No centralized system for managing listings, bookings, or customer data.
- Limited customer reach due to the absence of an online presence.
- Increased risk of errors in inventory and sales management.

## 2.2 PROPOSED SYSTEM

The proposed Online Tyre Shopping Platform for Sakthi Tyres will transition the business to a modern, digital platform that enhances customer engagement and streamlines

business operations. The new system will allow customers to browse available tyres, check prices, and make bookings online, significantly improving the shopping experience.

With an intuitive and user-friendly interface powered by the MERN stack (MongoDB, Express, React, Node), customers will easily search for tyres across various brands and make direct bookings. The system will also provide features like Tyre Management, Customer Booking, Billing, and User Authentication. Administrators will have real-time control over inventory, bookings, and customer data, enabling better decision-making and efficient management.

### 2.2.1 ADVANTAGES OF PROPOSED SYSTEM

The proposed system offers the following advantages:

- Streamlined tyre booking and purchasing process, making it convenient for customers to order online.
- User-friendly interface for easy navigation and browsing of tyres.
- Real-time inventory management and updates for administrators.
- Centralized platform for managing tyre listings, customer bookings, and data.
- Enhanced customer reach through a scalable online presence, leading to potential growth opportunities.

## 2.3 FEASIBILITY STUDY

Feasibility studies are crucial for evaluating the proposed Online Tyre Shopping Platform for Sakthi Tyres, assessing its functionality, potential impact on the business, ability to meet customer needs, and resource efficiency. While any project may appear feasible with unlimited resources and time, it is vital to analyze the practicality and associated risks of the software development process. The proposed system will undergo a proof-of-concept phase to determine its viability before proceeding with full development. Understanding the interplay between feasibility and risk is essential, particularly when project risks are significant and the feasibility of development is uncertain.

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

### 2.3.1 TECHNICAL FEASIBILITY

Technical feasibility assesses whether the necessary technology and resources are available for the successful development and implementation of the Online Tyre Shopping Platform for Sakthi Tyres. This evaluation is crucial as it requires simultaneous analysis and definition alongside the assessment of technical feasibility. A key consideration is whether the organization possesses sufficient resources for both development and implementation. Given that the proposed system utilizes existing technologies and requires minimal resources, it is classified as technically feasible.

### 2.3.2 OPERATIONAL FEASIBILITY

Operational feasibility evaluates the effectiveness of the proposed system in meeting the needs of both customers and administrators. The Online Tyre Shopping Platform is designed to provide robust support to users while enhancing overall operational performance. If it successfully fulfills these criteria, it can be regarded as operationally feasible. The system aims to deliver a convenient, user-friendly experience accessible to customers globally, thereby expanding its reach. Additionally, by creating better market opportunities for service providers, the proposed system further reinforces its operational viability.

### 2.3.3 ECONOMIC FEASIBILITY

Economic feasibility examines the project's development costs in relation to the potential revenue and benefits it will generate. The proposed Online Tyre Shopping Platform is economically advantageous as it does not require any additional hardware or software investments, ensuring its financial viability for the organization. By carefully analyzing costs against expected returns, the project demonstrates a strong economic rationale, ensuring that the investment will yield positive financial outcomes for Sakthi Tyres. This thorough assessment of economic feasibility underscores the project's capacity to deliver value and sustainability.

# CHAPTER 3

# SYSTEM SPECIFICATION

## 3.1 HARDWARE SPECIFICATION

- Processor : Intel® Core™ i3- 2.00GHz
- Ram : 8GB
- System type : 64-bit operating system
- Hard disk : 512GB
- Keyboard : Standard 102

## 3.2 SOFTWARE SPECIFICATION

- Operating System : Windows
- Front End : HTML, CSS, REACT JS
- Back End : NODEJS, Mongo DB
- Environment : Visual Studio, Chrome

### 3.2.1 FRONT END

#### 3.2.1.1 HTML

Hypertext Markup Language (HTML) is the foundation of web development and is essential for creating the structure of a website. For the Sakthi Tyres website, HTML will be used to organize the layout, including sections for product listings, customer information, and booking forms. HTML defines elements such as headings, paragraphs, images, and

links, ensuring that the website is easy to navigate and visually appealing. Semantic HTML will be employed to ensure that the content is well-structured and search engine-friendly, improving the site's visibility online. HTML also allows for the integration of Cascading Style Sheets (CSS) to style the site and JavaScript for adding interactive features, making the Sakthi Tyres website more engaging and user-friendly. Additionally, HTML attributes provide extra information about elements, facilitating the use of CSS and JavaScript to manipulate the content and enhance the overall user experience.

### 3.2.1.2 CSS

Cascading Style Sheets (CSS) are crucial for defining the visual presentation of HTML elements on the Sakthi Tyres website. CSS allows developers to control the design aspects such as fonts, colors, margins, borders, background images, and the overall layout. By separating content (HTML) from presentation (CSS), the code becomes cleaner and easier to maintain. For the Sakthi Tyres project, CSS ensures that the website has a visually appealing and user-friendly interface, making it easy for customers to explore products and book services. CSS also enables responsive design, ensuring the website is accessible and functional across various devices such as desktops, tablets, and smartphones. This is vital for customer convenience, as users may access the website on different devices. The use of external, internal, and inline style sheets provides flexibility in managing styles, with external style sheets ensuring consistency and reducing development time by applying the same styles across multiple pages.

### 3.2.1.3 JAVASCRIPT

JavaScript is a client-side scripting language that enhances the interactivity and functionality of web applications. It allows developers to create dynamic content that responds to user actions, such as form validation, animations, and real-time updates. JavaScript also enables asynchronous communication with the server through AJAX, allowing data to be fetched and updated without reloading the page. For the Sakthi Tyres website, JavaScript plays a vital role in creating interactive features like product search, booking tyres, and real-time notifications. It improves the user experience by providing

instant feedback and smooth interactions. JavaScript can be embedded directly into HTML using internal scripts or written in external files for better organization and maintainability. External JavaScript files help keep the code structured and easier to debug as the website grows in complexity. Its versatility enables real-time interaction, ensuring users can access up-to-date product details, product bookings, and enjoy a seamless shopping experience.

### 3.2.1.4 REACT JS

React JS is a powerful JavaScript library used for building user interfaces, especially for single-page applications. React allows developers to create reusable components, which are self-contained pieces of code that can be combined to build dynamic and complex web applications. For the Sakthi Tyres website, React will be used to create a highly interactive and responsive user interface that can handle and display dynamic data such as tyre listings, user profiles, and booking details without compromising performance. React's use of the virtual DOM (Document Object Model) ensures efficient updates and rendering, enhancing the site's performance by reducing direct manipulations of the real DOM. This ensures that even when new products are added or bookings are updated, the site remains fast and responsive, providing a seamless user experience. Additionally, React's component-based structure promotes code reuse, making the development process faster and more organized. This modularity makes it easier to maintain and scale the website as the platform grows. By managing the application state efficiently, React allows smooth handling of user interactions such as product selection, booking management, and personalized recommendations, offering an engaging and dynamic shopping experience for customers.

## 3.2.2 BACKEND

### 3.2.2.1 MONGO DB

MongoDB is a highly flexible and scalable NoSQL database system, ideal for managing large and complex datasets. Unlike traditional relational databases, MongoDB stores data in JSON-like documents, providing a more dynamic and adaptable structure. This flexibility is essential for the Sakthi Tyres platform, where the system needs to manage various types of data, such as tyre listings, customer profiles, and booking details. MongoDB efficiently handles unstructured or semi-structured data, making it easy to store and retrieve

product and customer information. Additionally, MongoDB's horizontal scaling capabilities ensure that as the platform grows, it can manage increased traffic and data volumes without sacrificing performance. This makes MongoDB a perfect choice for supporting the scalability and flexibility required for the online tyre shopping platform.

### 3.2.2.2 NODE JS

Node.js is a powerful runtime environment that enables JavaScript to be executed on the server side, making it an ideal choice for building scalable and high-performance web applications. For the Sakthi Tyres platform, Node.js will serve as the server-side engine that processes incoming customer requests, such as tyre inquiries, bookings, and purchases, while efficiently communicating with the database to provide real-time responses. Its non-blocking, event-driven architecture allows Node.js to handle multiple customer interactions simultaneously, ensuring smooth performance even during peak traffic times. This makes Node.js an excellent choice for the platform, where multiple users may be browsing tyre options, booking services, or making purchases at the same time.

# CHAPTER 4

# SYSTEM DESCRIPTION

## 4.1 MODULE DESCRIPTION

The project contains the following modules such as:

- Login/Register
- Home
- Product
- Cart
- Contact
- Payment
- Edit Profile
- Admin

### 4.1.1 LOGIN/REGISTER

The login module allows users to log in by entering their username and password. It ensures security through encrypted storage and validation of credentials. The registration module requires users to provide a username, email, password, and confirmation password (cpassword) to create an account. Both modules are designed to offer secure access and protect user data from unauthorized access.

### 4.1.2 HOME

The home page serves as the central hub for users interacting with the platform, offering an intuitive and visually appealing interface for easy navigation through various sections, such as products, about us, contact us, and help. Featuring a dynamic top bar that displays the logo, user profile options, and a shopping cart icon with real-time item updates, the page enhances user engagement by providing quick access to essential features. If users

attempt to access product-related functionalities without being logged in, a login prompt is displayed to ensure secure access. Additionally, the home page employs animations and effects to create a lively and enjoyable user experience.

### 4.1.3 PRODUCT

The product module provides a user-friendly interface for exploring a diverse range of tyres available for various vehicle types, including cars, two-wheelers, trucks, and more. Users can easily navigate through categories and select brands and models using dropdown menus, which filter the displayed products accordingly. Each tyre is showcased with high-quality images and detailed specifications, including the tyre model, size, brand, and pricing information. While users cannot purchase products online directly from this module, they have the option to add items to their cart for later consideration. Additionally, a "Buy Now" feature allows for quick access to the payment process, enhancing the overall shopping experience.

### 4.1.4 CART

The cart module manages the shopping cart functionality for an online tyre shopping platform. It utilizes React hooks to maintain the cart's state, loading status, and payment modal visibility. On component mount, it fetches cart items from the backend API, initializing their quantities and prices. Users can change item quantities, remove items from the cart, and trigger a payment modal for selected items. The component displays loading animations when fetching data and a message if the cart is empty. It also integrates Lottie animations for visual effects and uses Axios for API requests, ensuring a smooth user experience while providing options for ordering and purchasing tyres.

### 4.1.5 CONTACT

The contact module enables users to easily reach out for assistance regarding their tyre shopping experience. Users can fill out a form to submit inquiries, feedback, or requests, which will be directed to the relevant support team for prompt review. Key contact details, including a dedicated support phone number and email address, may also be provided to facilitate direct communication. This module ensures that user inquiries and concerns are handled efficiently, enhancing overall customer satisfaction.

### 4.1.6 PAYMENT

The payment module facilitates seamless transactions for users purchasing tyres through various methods, including UPI QR codes and cash. Users can conveniently scan a QR code with their mobile device to complete the payment or opt for cash payment for offline orders. The module exclusively supports UPI payments and cash transactions, ensuring a straightforward payment process without the inclusion of credit or debit card options. Upon successful transactions, the system generates payment confirmations and receipts, which are promptly sent to users. This approach guarantees an efficient payment experience, allowing users to finalize their purchases with ease.

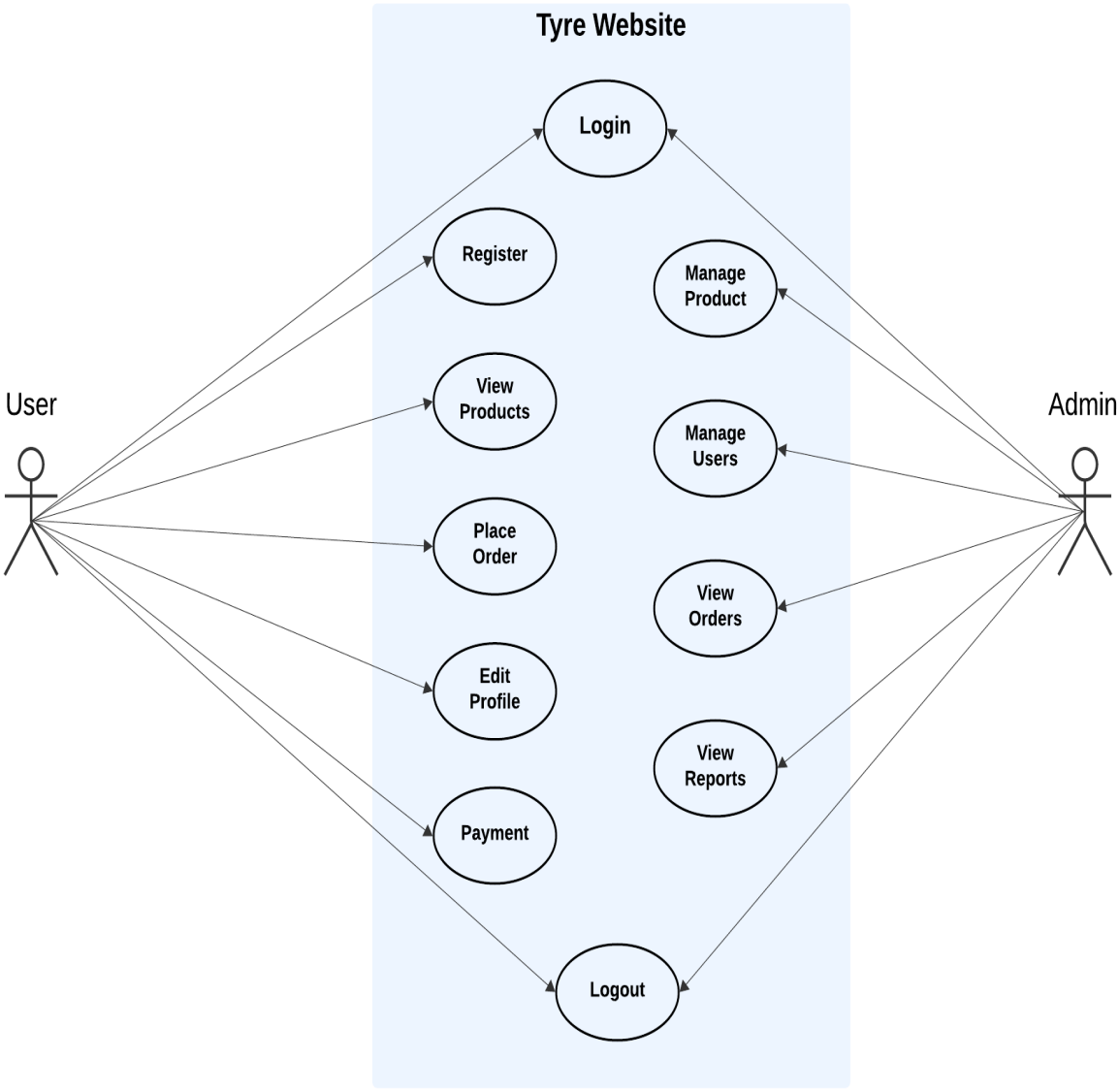
### 4.1.7 EDIT PROFILE

This module allows users to update their personal details such as username, email, and password. Users can modify their profile information at any time, including uploading a profile picture. The module also provides an option to change security settings like the password, ensuring users can maintain the security of their accounts. Profile changes are subject to authentication checks for user safety.

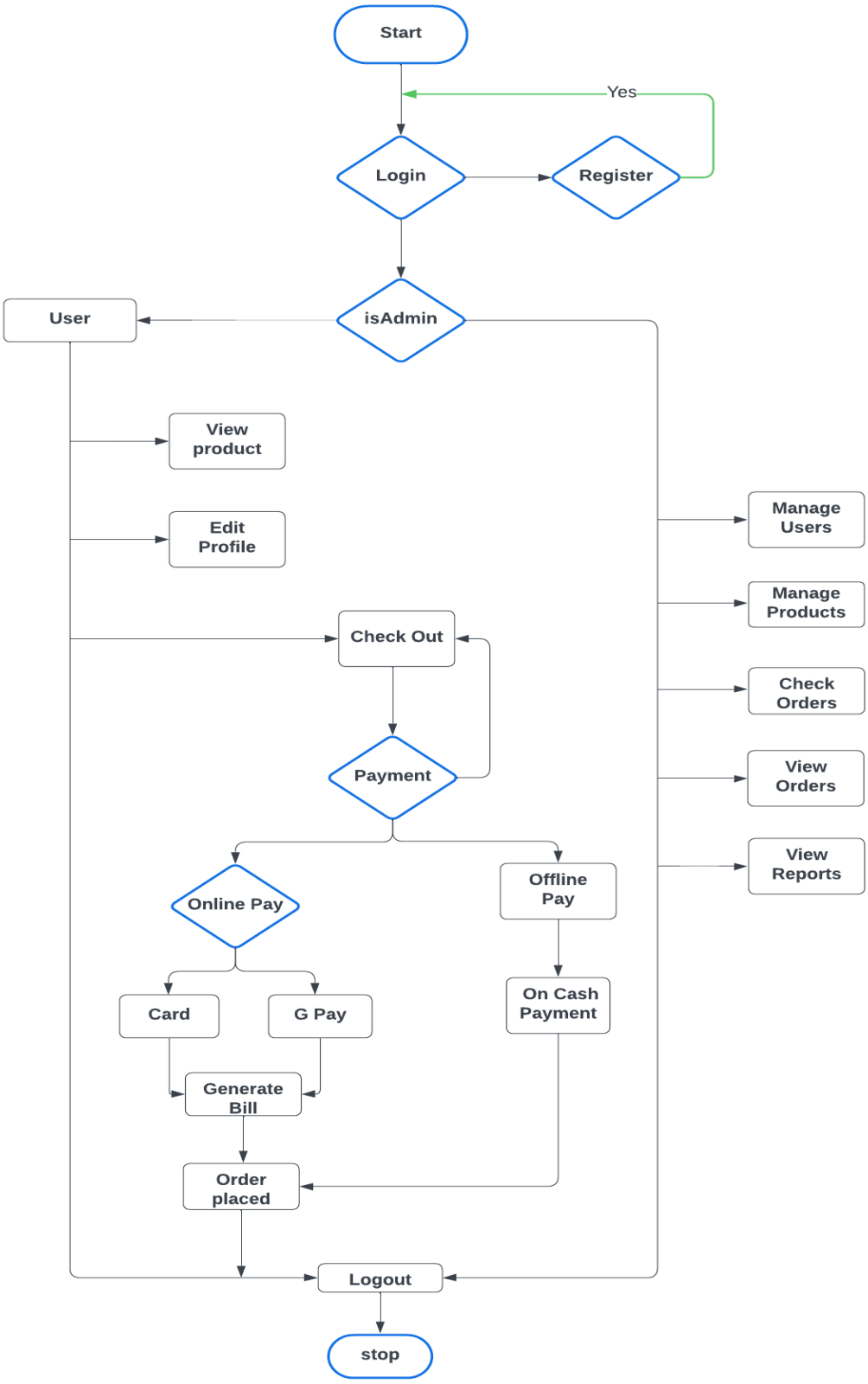
### 4.1.8 ADMIN

The admin module enables administrators to manage and monitor all aspects of the platform. Admins can view and manage user and products, and manage product listings. They also have the ability to track payment records, ensuring transactions through UPI or cash are logged. Additionally, admins can edit or remove content, ensuring that the system stays up to date with accurate information on products.

## 4.2 USE CASE DIAGRAM

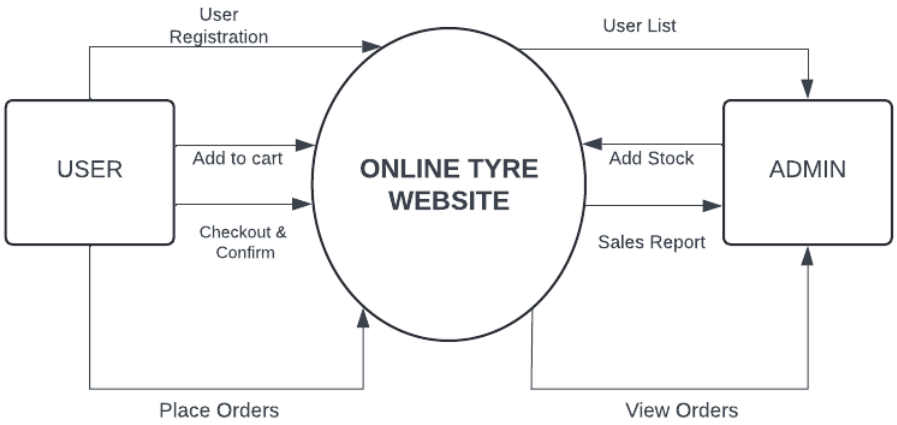


## 4.3 SYSTEM FLOW DIAGRAM

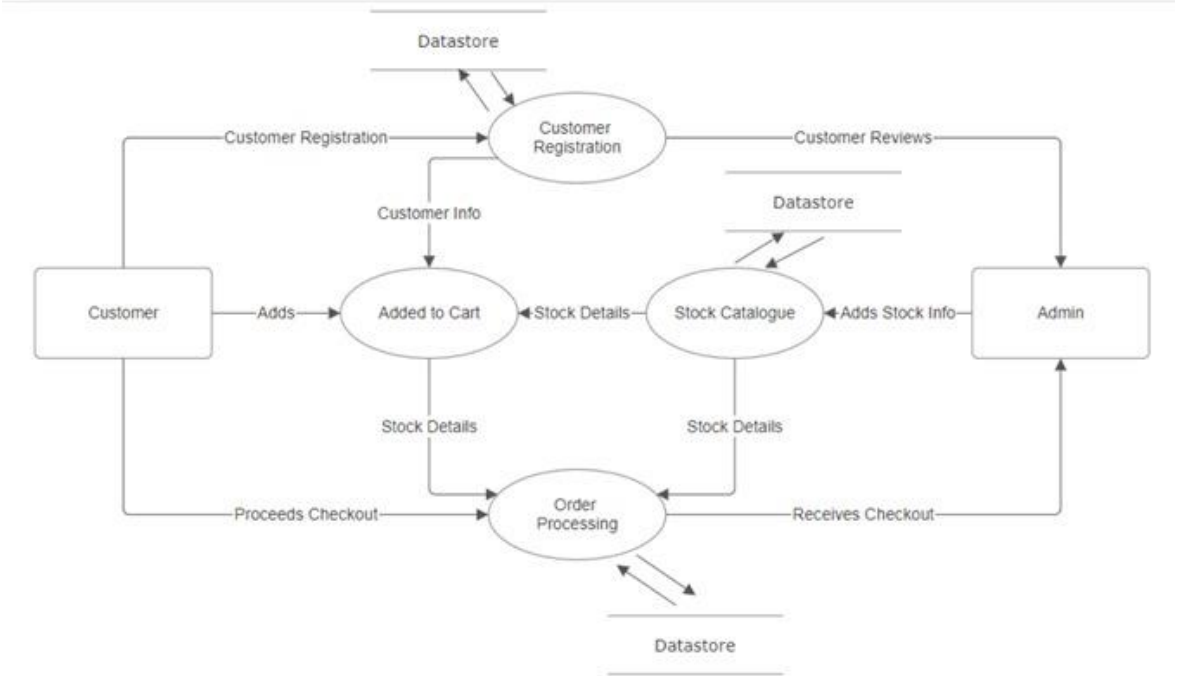


## 4.4 DATA FLOW DIAGRAM

### 4.4.1 DATA FLOW DIAGRAM (LEVEL 0)




### 4.4.2 DATA FLOW DIAGRAM (LEVEL 1)



## 4.5 DATABASE DESIGN

**TABLE NO** : 4.5.1

**TABLE NAME** : users

**DESCRIPTION** **:** The table is used to store the login details of admin and the user.

| FIELD NAME | DATA TYPE | DESCRIPTION |
|---|---|---|
| Username | String | Username |
| Email | String | Email |
| Password | String | Password |
| Profile Image | String | User's profile image |
| Address | String | Address |
| Fullname | String | Fullname |
| Location | String | Location |
| Phno | String | Phone number |

**TABLE NO** : 4.5.2

**TABLE NAME** : tyres

**DESCRIPTION** **:** The table is used to store the tyre details.

| FIELD NAME | DATA TYPE | DESCRIPTION |
|---|---|---|
| Image | String | Tyre image |
| Vehicle Type | String | Vehicle Type |
| Vehicle Model | String | Vehicle Model |
| Vehicle Brand | String | Vehicle Brand |

| | | |
|---|---|---|
| Tyre Brand | String | Tyre Brand |
| Tyre Model | String | Tyre Model |
| Tyre Size | String | Tyre Size |
| Price | String | Price |

**TABLE NO** : 4.5.3

**TABLE NAME** : carts

**DESCRIPTION** **:** The table is used to store the cart details.

| FIELD NAME | DATA TYPE | DESCRIPTION |
|---|---|---|
| User Id | String | User id |
| Items | Array | Product item |

**TABLE NO** : 4.5.4

**TABLE NAME** : orders

**DESCRIPTION** **:** The table is used to store the order details.

| FIELD NAME | DATA TYPE | DESCRIPTION |
|---|---|---|
| User Id | String | User id |
| Items | Array | Product item |
| Total Price | Number | Total price |
| Payment Type | String | Payment type |
| Payment Method | String | Payment method |
| Tyre State | String | Tyre state |

## 4.6 INPUT DESIGN

In the proposed work, the input forms that are designed are the Customer registration form and product details form. In the customer registration form, information such as First Name, Last Name, Email id, Password are collected. Validation is done for password and mobile number. If the data is not valid, proper error message are displayed. Following is the Input design:

- Login form
- Registration form
- Edit profile form

## LOGIN FORM

The login form is the initial step for users to access their accounts, typically requiring a username and password. It should be designed for simplicity and security, incorporating features like password recovery options and two-factor authentication. Clear error messages help guide users in case of incorrect entries, enhancing their experience. A responsive layout ensures usability across various devices, encouraging users to log in frequently.

## REGISTER FORM

The registration form facilitates the creation of new accounts by collecting essential information such as name, email, and password. It should emphasize data security with features like email verification and strong password requirements. A user-friendly layout, possibly with social media login options, can streamline the process for new users. By clearly outlining terms and privacy policies, the form helps build trust and encourages user engagement.

## EDIT PROFILE FORM

The edit profile form allows users to update their personal information easily and securely. It should pre-fill existing data to simplify the editing process while including validation messages to ensure accurate input. Requiring password re-entry for significant changes enhances security and prevents unauthorized modifications. A straightforward

design and clear instructions contribute to a positive user experience during the profile update.

## 4.7 OUTPUT DESIGN

The source of information to be generated through this work is referred to as the output. The systems should be improved through efficient and understandable output design.

Following is the output design:

- Generated Bill

## GENERATING BILL

The Generating Bill section effectively compiles and presents key information regarding each order in a clear and organized format. It includes details such as the Tyre Model, Tyre Brand, Tyre Size, Quantity, and Total Price, allowing users to easily review their purchases. The bill also highlights the Payment Type and Payment Method, ensuring transparency about how the transaction was completed, whether through online card payments or offline cash payments. Additionally, users can download the bill as a PDF, providing them with a tangible record of their transaction. This comprehensive approach not only simplifies the user experience but also aids in managing their financial commitments related to tyre purchases.

# CHAPTER 5

# SYSTEM TESTING

System testing is a type of testing that evaluates the overall functionality and performance of a complete and fully integrated software. System testing is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

## 5.1 UNIT DESIGN

In unit testing, we must test the programs making up the system. By giving improper inputs, the errors occurred are noted and eliminated. This enables, to detection of errors in coding and logic that are contained within the module alone. The testing was carried out during the programming. In this system each form is considered as a separate unit and tested for errors. Every user inut is unit tested for a valid accepted range.

### Test Case 1

Module : Admin Login

Login Type : Loading of the appropriate form for the administrator

Input : Username and Password

Expected Output : Display admin menu

### Sample Test Case

Output : Redirect to Main Page and display the admin menus

Analysis : In this form, username and password of the admin are tested. If it is correct, it

## 5.2 INTEGRATION TESTING

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions.

### Test Case 1

Module : Admin

Login Type : Tyre Management and User Management

Input : Navigation between Admin options

Expected Output : Navigation between modules is completed

### Sample Test Case

Output : On Clicking Login and other Admin modules the respective pages will open correctly.

Analysis : Respective pages will be open.

## 5.3 VALIDATION TESTING

Verification and validation testing are two important tests, which are carried out before the product has been handed over to the customer. It determines whether the software function as the user expected.

### Test Case 1

Module : Register

Login Type : Register new user

Input : Input to all fields

Expected Output : Required field should not be empty

## Sample Test Case

| | |
|---|---|
| Input | : Input for a required field is not provided |
| Output | : Provide all the required fields. |
| Analysis | : It should navigate to the next page. |

# CHAPTER 6

# SYSTEM IMPLEMENTATION

System implementation is the stage of the project where the theoretical design is turned into a working system. If the implementation stage is not properly planned and controlled, it can cause errors. Thus, it can be the most crucial stage in achieving a successful new system and in giving the user confidence that the new system will work and be effective.

Implementation is the process of converting a new or revised system design into an operational one when the initial design was done by the system, a demonstration was given to the end user about the working system.

This process is used to verify and identify any logical mess working in the system by feeding various combinations of test data. After the approval of the system by both the end user and management the system was implemented. System implementation is made up of many activities. The six major activities are as follows.

## CODING

Coding is the process whereby the physical design specification created by the analysis team is turned into working computer code by the programming team.

## TESTING

Once the coding process is beginning and proceeds in parallel, each program module can be tested. Testing ensures a quality product is delivered to the customers.

## INSTALLATION

Installation is the process during which the current system is replaced by the new system. This includes the conversion of existing data, software and documentation, and work procedures to those consistent with the new system.

## DOCUMENTATION

It results from the installation process; user guides provide information on how and the system and its flow.

## TRAINING AND SUPPORT

A training plan is a strategy for training users so they quickly learn the new system. The development of the training plan probably began earlier in the project.

# CHAPTER 7

# CONCLUSION AND FUTURE ENHANCEMENTS

## 7.1 CONCLUSION

Implementing a system is crucial for turning design ideas into real solutions that work for users and meet the goals of the organization. With careful planning and execution, we can reduce the chances of errors and ensure a smooth transition to a working system. By thoroughly testing and validating the system, we help build confidence among users and set the stage for the system to perform well and be improved in the future. Overall, a successful implementation enhances efficiency and productivity, contributing to the organization's success.

Thus the website for the Online Tyre Shopping Platform has been successfully developed for Sakthi Tyres, utilizing React JS for the frontend and Node JS with Express JS for the backend.

## 7.2 FUTURE ENHANCEMENT

The application becomes useful if the below enhancement is made in the future.

- Add WhatsApp integration to send users real-time updates and notifications.
- Track user activity to understand how they interact with the system and what they prefer.
- Add functionality to display the shipping and delivery status to the user.

# APPENDIX 1- SAMPLE CODING

## App.jsx

```
import React, { useState } from 'react';
import { BrowserRouter, Routes, Route, useLocation } from 'react-router-dom';
import { Elements } from '@stripe/react-stripe-js';
import { loadStripe } from '@stripe/stripe-js';
import Login from './Form/Login';
import Registration from './Form/Registration';
import Home from './Pages/Home';
import Product from './Pages/Product';
import Cart from './Pages/Cart/Cart';
import About from './Pages/About/About';
import Contact from './Pages/Contact_Us/Contact';
import Help from './Pages/Help/Help';
import Welcome from './Pages/Welcome';
import Footer from './Pages/Footer';
import ScrollToTop from './Pages/Scroll/ScrollToTop';
import LoadingComponent from './Pages/Animation/Loading';
import EditProfile from './Pages/Edit_Profile/EditProfile';
import JK from './Pages/Brand_About/JK';
import Michelin from './Pages/Brand_About/Michelin';
import Payment from './Pages/Payment/Payment';
import Orders from './Pages/Orders/Orders'

// Admin
import Admin from './Admin/Admin'
import Dashboard from './Admin/Dashboard'
import ManageProducts from './Admin/ManageProducts'
```

```
import ManageUsers from './Admin/ManageUsers'

// Initialize Stripe with your publishable key once

const stripePromise =
loadStripe("pk_test_51Q5g2GD7L0PMiSZDyh1Slqidawdli8iWnIGxx69koWIyfEpliXrlqP
BaDqtTtiiiee6upIoioleWHdwXxZDzTvdU00LXTGyT2G");

function App() {

  const [cartItems, setCartItems] = useState([]);

  const addToCart = (item) => {

    setCartItems((prevItems) => [...prevItems, item]);

  };

  const removeFromCart = (id) => {

    setCartItems((prevItems) => prevItems.filter(item => item.id !== id));

  };

  return (

    <BrowserRouter>

      <ScrollToTop />

      <Elements stripe={stripePromise}>

        <MainContent cartItems={cartItems} addToCart={addToCart}
removeFromCart={removeFromCart} />

      </Elements>

    </BrowserRouter>

  );

}

function MainContent({ cartItems, addToCart, removeFromCart }) {

  const location = useLocation();

  // Show footer unless on specific pages

  const showFooter = !['/login', '/register', '/edit-profile', '/cart', '/payment', '/admin',
'/admin/manageuser', '/admin/manageproduct', '/cart/orders'].includes(location.pathname);

  return (

    <>

      <LoadingComponent />
```

```
      <Routes>
        <Route path='/' element={<Home />}>
          <Route index element={<Welcome />} />
          <Route path='product' element={<Product addToCart={addToCart} />} />
          <Route path='cart' element={<Cart cartItems={cartItems}
removeFromCart={removeFromCart} />} />
          <Route path='about' element={<About />} />
          <Route path='contact' element={<Contact />} />
          <Route path='help' element={<Help />} />
          <Route path='apollo' element={<Apollo />} />
          <Route path='bridgestone' element={<Bridgestone />} />
          <Route path='jk' element={<JK />} />
          <Route path='michelin' element={<Michelin />} />
          <Route path='cart/orders' element={<Orders />} />
        </Route>
        <Route path='/login' element={<Login />} />
        <Route path='/register' element={<Registration />} />
        <Route path='/edit-profile' element={<EditProfile />} />
        <Route path='/payment' element={<Payment />} />
        <Route path="/admin" element={<Admin />}>
          <Route index element={<Dashboard />} />
          <Route path="manageuser" element={<ManageUsers />} />
          <Route path="manageproduct" element={<ManageProducts />} />
        </Route>
      </Routes>
      {showFooter && <Footer />}
    </>
  );
}
export default App;
```

## Login.jsx

```
import React, { useState } from 'react';

import left from '../assets/tyre2.png';

import axios from 'axios';

import { useNavigate } from 'react-router-dom';

import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';

import { faArrowLeft } from '@fortawesome/free-solid-svg-icons';

import Toaster from './Toaster';

import './style.css';



function Login() {

  const [loading, setLoading] = useState(false);

  const [data, setData] = useState({ name: "", password: "" });

  const [logInStatus, setLogInStatus] = useState(null);

  const navigate = useNavigate();

  const handleChange = (e) => {

    const { name, value } = e.target;

    setData({ ...data, [name]: value });

  };
```

```
const display = async () => {
  setLoading(true);
  try {
    const config = {
      headers: {
        'Content-Type': 'application/json',
      },
    };
    const res = await axios.post("http://localhost:8080/user/login", data, config);
    console.log("Login response:", res.data);
    // Store the token in localStorage
    localStorage.setItem("token", res.data.token);
    localStorage.setItem("userdata", JSON.stringify(res.data));
    localStorage.setItem("userStatus", "true");
    setLogInStatus({ msg: "Login Successful! 😎", key: Math.random(), severity:
"success" });
    setTimeout(() => {
      navigate('/');
    }, 2000);
```

```
    } catch (err) {
      console.error("Axios Error -> ", err.response ? err.response.data : err.message);
      setLogInStatus({ msg: err.response ? err.response.data.message : "Invalid username or password. Please try again.", key: Math.random(), severity: "error" });
    } finally {
      setLoading(false);
    }
  };
  const ValidateData = (e) => {
    e.preventDefault();
    if (!data.name || !data.password) {
      setLogInStatus({ msg: "Fill in all fields!", key: Math.random(), severity: "error" });
    } else if (data.password.length < 6) {
      setLogInStatus({ msg: "Password must be at least 6 characters long", key: Math.random(), severity: "error" });
    } else {
      display();
    }
  };
  const handleToasterClose = () => {
```

```
  if (logInStatus?.severity === 'success') {
    navigate('/');
  }
};
return (
  <>
    <div className='main-container'>
      <div className="container">
        <div className="left">
          <div className='back-button' onClick={() => navigate('/')}>
            <FontAwesomeIcon icon={faArrowLeft} className='left-arrow'/>
          </div>
          <img className="bg" src={left} alt="Background" />
          <div className="logo">
            <h2>Sakthi Tyres</h2>
          </div>
          <div className="left-content">
            <h1>Welcome Back!</h1>
            <h2>Your Journey Continues Here</h2>
```

```
          </div>
        </div>
        <div className="right">
          <div style={{ height: '420px' }} className="box">
            <span className="borderLine"></span>
            <form onSubmit={ValidateData}>
              <h2>SIGN IN</h2>
              <div className="inputBox">
                <input type="text" name='name' value={data.name}
onChange={handleChange} required />
                <span>Username</span>
                <i></i>
              </div>
              <div className="inputBox">
                <input type="password" name='password' value={data.password}
onChange={handleChange} required />
                <span>Password</span>
                <i></i>
              </div>
              <div className="links">
```

```
            <a href="#">Forgot Password?</a>
          </div>
          <input type="submit" value="Sign in" disabled={loading}/>
          <div className='last'>
            <p>Don't have an account?<u onClick={() =>
navigate('/register')}>  Sign up</u></p>
          </div>
        </form>
        {logInStatus && (
          <Toaster
            key={logInStatus.key}
            message={logInStatus.msg}
            severity={logInStatus.severity}
            onClose={handleToasterClose}
          />
        )}
      </div>
    </div>
  </div>
</div>
```

```
    </>
  );
}
export default Login;
```

## Home.jsx

```
import React, { useState, useEffect } from 'react';
import ReactDOM from 'react-dom';
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome';
import { faCircleUser, faCartShopping, faArrowRightFromBracket, faPen, faGear, faCircleQuestion } from '@fortawesome/free-solid-svg-icons';
import './myStyle.css';
import Sakthi from '../assets/Sakthi.png';
import { NavLink, BrowserRouter as Router, useNavigate, useLocation } from 'react-router-dom';
import { Outlet } from 'react-router-dom';
import Lottie from 'react-lottie';
import logoutAnimation from '../assets/Home/logout.json';
import Login_Gif from '../assets/Home/Login_Animation.gif';
import axios from 'axios';
import DefaultProfile from './Edit_Profile/default-profile.png'
import "aos/dist/aos.css";
import AOS from 'aos';

function Home() {
    const [isProfileDropdownVisible, setIsProfileDropdownVisible] = useState(false);
    const [cartCount, setCartCount] = useState(0);
```

```
const [showLoginPrompt, setShowLoginPrompt] = useState(false);
const [showLogoutAnimation, setShowLogoutAnimation] = useState(false);
const location = useLocation();
const navigate = useNavigate();
const [username,setUserName] = useState(null);
const [profileImg, setProfileImg] = useState(null);

useEffect(() => {
   AOS.init({
     once: true,
     disable: "phone",
     duration: 750,
     easing: "ease-out-cubic",
   });
 }, []);

useEffect(() => {
   const fetchCartCount = async () => {
      try {
         const token = localStorage.getItem('token');

         if (token) {
            const { data } = await axios.get('http://localhost:8080/user/cart', {
               headers: {
                  Authorization: `Bearer ${token}`,
               },
            });
            setCartCount(data.length);
         } else {
            setCartCount(0);
```

```
            }
        } catch (error) {
            console.error('Failed to fetch cart count:', error);
            setCartCount(0);
        }
    };
    fetchCartCount();

    const userStatus = JSON.parse(localStorage.getItem("userStatus"));
    if ((location.pathname === '/product' || location.pathname === '/cart') && !userStatus)
{
        setShowLoginPrompt(true);
        navigate('/login');
    }
  }, [location, navigate]);

  const handleProfileClick = () => {
    setIsProfileDropdownVisible(!isProfileDropdownVisible);
  };

  const handleProductClick = () => {
    const userStatus = JSON.parse(localStorage.getItem("userStatus"));
    if (!userStatus) {
        setShowLoginPrompt(true);
    } else {
        navigate('/product');
    }
  };

  const handleCartClick = () => {
```

```
    const userStatus = JSON.parse(localStorage.getItem("userStatus"));
    if (!userStatus) {
        setShowLoginPrompt(true);
    } else {
        navigate('/cart');
    }
};

const handleLoginClick = () => {
    navigate('/login');
};

const handleEditProfileClick = () => {
    navigate('/edit-profile');
};

const closeLoginPrompt = () => {
    setShowLoginPrompt(false);
};

const handleLogoutClick = () => {
    localStorage.setItem("userStatus", false);
    localStorage.removeItem('token');
    localStorage.removeItem('userdata');
    localStorage.removeItem('userData');
    setShowLogoutAnimation(true);
    setTimeout(() => {
        navigate('/login');
    }, 3000);
};
```

```
const isCartActive = location.pathname === '/cart';
const isProductActive = location.pathname === '/product';

const defaultOptions = {
    loop: true,
    autoplay: true,
    animationData: logoutAnimation,
    rendererSettings: {
        preserveAspectRatio: 'xMidYMid slice'
    }
};

useEffect(()=>{
    if(localStorage.getItem('userdata')){
        setUserName(JSON.parse(localStorage.getItem('userdata')).name);
    } else {
        setUserName("Profile");
    }
},[]);

useEffect(() => {
    if (localStorage.getItem('userData')) {
        const userData = JSON.parse(localStorage.getItem('userData')).data;
        if(userData.profileImage != "")
        {
            setProfileImg(userData.profileImage);
        }
        else{
            setProfileImg(DefaultProfile);
```

```
        }
    } else {
        setProfileImg(DefaultProfile);
    }
}, []);
return (
    <>
        {showLoginPrompt && (
            <div data-aos="zoom-in" className='login-prompt-container'>
                <div className='login-prompt-box'>
                    <img src={Login_Gif} alt="Login_Gif" />
                    <h2>Please login to continue </h2>
                    <button className='login-button' onClick={handleLoginClick}>
                        Login
                    </button>
                    <button className='close-button' onClick={closeLoginPrompt}>
                        Cancel
                    </button>
                </div>
            </div>
        )}
        {showLogoutAnimation && (
            <>
                <div className='dark-overlay'></div>
                <div className='logout-animation-container'>
                    <Lottie options={defaultOptions} height={400} width={400} />
                </div>
            </>
        )}
        <div className={`top-bar ${showLoginPrompt ? 'blur-background' : ''}`}>
```

```
            <div>
              <img className='tyre-logo' src={Sakthi} alt='Tyres' />
            </div>

            {isProfileDropdownVisible && (
              <div className='dropDownProfile'>
                {JSON.parse(localStorage.getItem("userStatus")) ? (
                  <>
                    <h2>
                      {username}
                    </h2>
                    <ul>
                      <hr />
                      <li onClick={handleEditProfileClick}>
                        <FontAwesomeIcon className="icons" icon={faPen} />
                        <span className="edit-profile-text">Edit Profile</span>
                      </li>
                      <li>
                        <FontAwesomeIcon className="icons"
icon={faCircleQuestion}/>
                        <span className="edit-profile-text">Help & Support</span>
                      </li>
                      <hr />
                      <li onClick={handleLogoutClick} className='logout-item'>
                        <FontAwesomeIcon className="icons-log"
icon={faArrowRightFromBracket} />
                        <span className="edit-profile-text">Logout</span>
                      </li>
                    </ul>
                  </>
                ) : (
```

```
                    <ul>
                        <li onClick={handleLoginClick} className='login-item'>
                            <FontAwesomeIcon className="icons-log"
icon={faArrowRightFromBracket} />
                            <span className="edit-profile-text">Login</span>
                        </li>
                    </ul>
                )}
            </div>
        )}
        <div className='nav-item'>
            <nav>
                <ul id='navbar'>
                    <li><NavLink className='page-link' to='/'>Home</NavLink></li>
                    <li>
                        <span
                            className={`page-link ${isProductActive ? 'active' : ''}`}
                            onClick={handleProductClick}
                        >
                            Product
                        </span>
                    </li>
                    <li><NavLink className='page-link' to='about'>About
Us</NavLink></li>
                    <li><NavLink className='page-link' to='contact'>Contact
Us</NavLink></li>
                    <li><NavLink className='page-link' to='help'>Help</NavLink></li>
                    <img
                        src={profileImg}
                        alt="Profile"
                        className='profile'
```

```
                        onClick={handleProfileClick}
                      />
                    </ul>
                  </nav>
                </div>
                {/* Cart Icon */}
                <div
                  className={`cart-icon-container ${isCartActive ? 'active' : ''}`}
                  onClick={handleCartClick}>
                  <FontAwesomeIcon icon={faCartShopping} className={`cart-icon
${isCartActive ? 'active-icon' : ''}`} />
                  {cartCount > 0 && (
                    <div className='cart-badge'>{cartCount}</div>
                  )}
                </div>
              </div>
              <div className={`Outlet ${showLoginPrompt ? 'blur-background' : ''}`}>
                <Outlet />
              </div>
            </>
        );
}
const App = () => (
    <Router>
        <Home />
    </Router>
);
ReactDOM.render(<App />, document.getElementById('root'));
export default Home;
```

## Product.jsx

```
import React, { useState, useEffect } from 'react';
import './Product.css';
// import Cars from './Vehicle/Cars';
import vehicleData from './Vehicle/VehicleData';
import { useNavigate } from 'react-router-dom';
//Company Logos
import apollo from '../assets/Welcome/Dealers/apollo.png';
import bridgestone from '../assets/Welcome/Dealers/bridgestone.png';
import jk from '../assets/Welcome/Dealers/jk.png';
import Car from '../assets/Product/car.svg';
import Bike from '../assets/Product/bike.svg';
import Truck from '../assets/Product/truck.svg';
import SCV from '../assets/Product/scv.svg';
import LCV from '../assets/Product/lcv.svg';
import Pickup from '../assets/Product/pickup-van.svg';
import MCV from '../assets/Product/mcv.svg';
import ICV from '../assets/Product/icv.svg';
import Car_active from '../assets/Product/car-active.svg';
import Bike_active from '../assets/Product/bike-active.svg';
import Truck_active from '../assets/Product/truck-active.svg';
import SCV_active from '../assets/Product/scv-active.svg';
import LCV_active from '../assets/Product/lcv-active.svg';
import Pickup_active from '../assets/Product/pickup-van-active.svg';
import MCV_active from '../assets/Product/mcv-active.svg';
import ICV_active from '../assets/Product/icv-active.svg';
import AddShoppingCartTwoToneIcon from '@mui/icons-
material/AddShoppingCartTwoTone';
import { IconButton } from '@mui/material';
```

```
import ShoppingCartOutlinedIcon from '@mui/icons-material/ShoppingCartOutlined';
import Payment from './Payment/Payment';
import { ToastContainer, toast, Bounce } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import axios from 'axios';

function Product() {
  const [activeCategory, setActiveCategory] = useState('Car');
  const [selectedBrand, setSelectedBrand] = useState('');
  const [selectedModel, setSelectedModel] = useState('');
  const [cart, setCart] = useState([]);
  const [Cars,setCars]=useState([]);
  const navigate = useNavigate(null);
  const [isPaymentModalOpen, setIsPaymentModalOpen] = useState(false);

  useEffect(()=>{

    const fetch=async()=>{
      try{
        const res=await axios.get("http://localhost:8080/user/displaytyredata");
        console.log(res.data.Data);
        setCars(res.data.Data);
      }
      catch(err){
        console.log(err.response.data.message);
      }
    }
    fetch();
  },[]);
```

```
  useEffect(() => {
    const storedCart = JSON.parse(localStorage.getItem('cart')) || [];
    setCart(storedCart);
  }, []);
  const categories = [
    { name: 'Car', icon: Car, activeIcon: Car_active },
    { name: 'Two Wheeler', icon: Bike, activeIcon: Bike_active },
    { name: 'Truck', icon: Truck, activeIcon: Truck_active },
    { name: 'SCV', icon: SCV, activeIcon: SCV_active },
    { name: 'LCV', icon: LCV, activeIcon: LCV_active },
    { name: 'Pick Up', icon: Pickup, activeIcon: Pickup_active },
    { name: 'MCV', icon: MCV, activeIcon: MCV_active },
    { name: 'ICV', icon: ICV, activeIcon: ICV_active }
  ];
  const handleBrandChange = (e) => {
    setSelectedBrand(e.target.value);
    setSelectedModel('');
  };
  const filteredItems = Cars.filter(item => {
    const isCategoryMatch = item.vehicle_type === activeCategory;
    const isBrandMatch = !selectedBrand || item.vehicle_brand === selectedBrand;
    const isModelMatch = !selectedModel || item.vehicle_model === selectedModel;
    return isCategoryMatch && (selectedBrand || selectedModel) && isBrandMatch &&
isModelMatch;
  });

  const addToCart = async (item) => {
    try {
      const token = localStorage.getItem('token');
      if (!token) {
```

```
    throw new Error('No token found, please log in again.');
   }

   await axios.post('http://localhost:8080/user/cart', item, {
    headers: {
     Authorization: `Bearer ${token}`,
    },
   });

   toast.info("The item has been added to Cart", {
    position: "top-right",
    autoClose: 1500,
    hideProgressBar: false,
    closeOnClick: true,
    pauseOnHover: true,
    draggable: true,
    theme: "dark",
    transition: Bounce,
   });
  } catch (error) {
   toast.error("The item already added to Cart !", {
    position: "top-right",
    autoClose: 1500,
    hideProgressBar: false,
    closeOnClick: true,
    pauseOnHover: true,
    draggable: true,
    theme: "dark",
    transition: Bounce,
   });
```

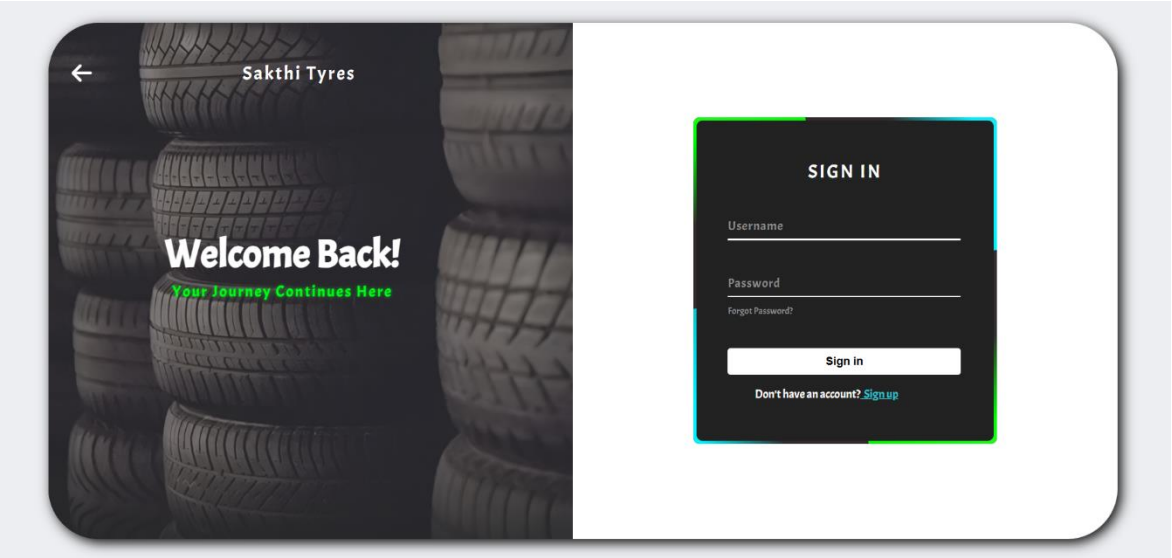
```
    console.error('Error adding item to cart:', error);
  }
};

const handleBuyClick = () => {
  setIsPaymentModalOpen(true);
};
// Function to close the payment modal
const handleClosePaymentModal = () => {
  setIsPaymentModalOpen(false);
};
return (
  <>
    <div className="product">
      <div className="category-row">
        {categories.map((category) => (
          <div
            key={category.name}
            className={`category-item ${activeCategory === category.name ? 'active' : ''}`}
            onClick={() => setActiveCategory(category.name)}
          >
            <img
              src={activeCategory === category.name ? category.activeIcon : category.icon}
              alt={`${category.name} icon`}
              className="category-icon"
            />
            {category.name}
          </div>
        ))}
      </div>
```

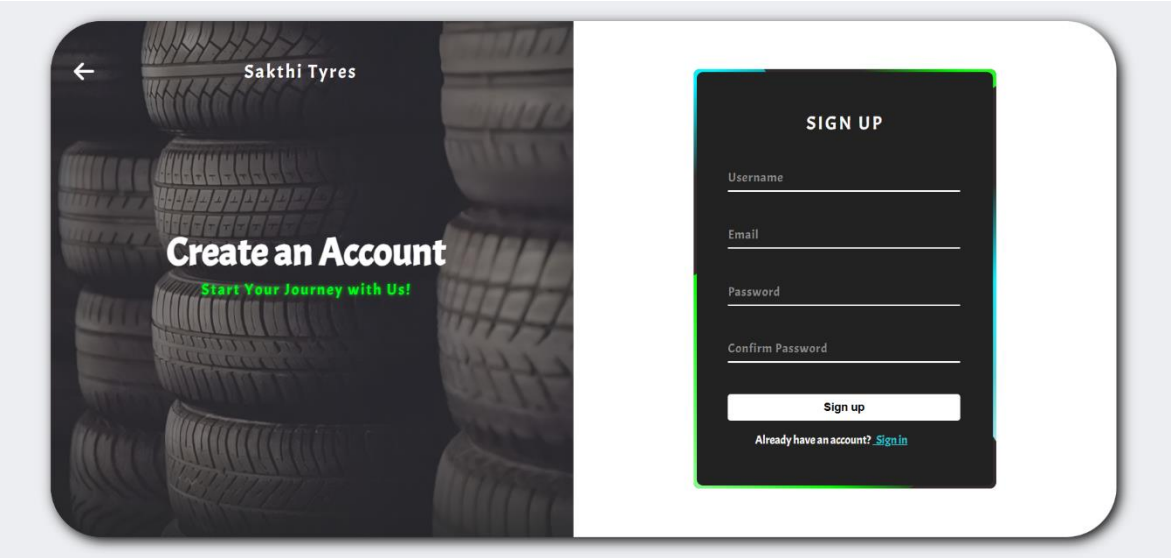
```
      <div className="category-content">
        {activeCategory && (
          <div className="category-details">
            <h2>Select a Tyre For your {activeCategory}</h2>
            <div className="dropdown-container">
              <div className="dropdown-left">
                <label>Select Brand</label>
                <select className='select-type' value={selectedBrand}
onChange={handleBrandChange}>
                  <option value="">Select Brand</option>
                  {vehicleData[activeCategory]?.brands.sort().map((brand) => (
                    <option key={brand} value={brand}>{brand}</option>
                  ))}
                </select>
              </div>
              <div className="dropdown-right">
                <label>Select Model</label>
                <select className='select-type' value={selectedModel} onChange={(e) =>
setSelectedModel(e.target.value)} disabled={!selectedBrand}>
                  <option value="">Select Model</option>
                  {selectedBrand &&
vehicleData[activeCategory]?.models[selectedBrand]?.map((model) => (
                    <option key={model} value={model}>{model}</option>
                  ))}
                </select>
              </div>
            </div>
            <div className="category-images">
              {filteredItems.map((item) => {
                const { image, tyre_model, tyre_size, tyre_brand, price } = item;
                return (
```

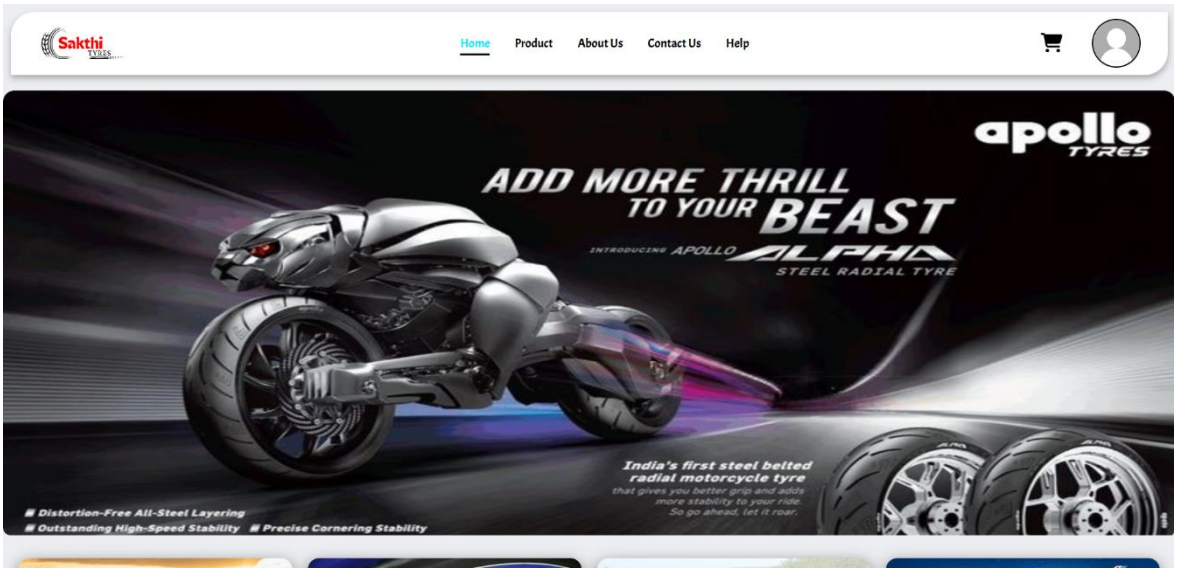
```
                <div className="tire-box" >
                  <img src={image} alt={tyre_model} />
                  <div className="content">
                  <h2>
                    {tyre_brand === 'Apollo' && <img src={apollo} alt="Apollo" />}
                    {tyre_brand === 'Bridgestone' && <img src={bridgestone}
alt="Bridgestone" />}
                    {tyre_brand === 'JK' && <img src={jk} alt="JK" />}
                    {(tyre_brand !== 'Apollo' && tyre_brand !== 'Bridgestone' && tyre_brand
!== 'JK') && tyre_brand}
                  </h2>
                    <h3>{tyre_model}</h3>
                    <p>{tyre_size}</p>
                    <div className='price'>
                      <span className="price-label">Price : </span>
                      <span className="price-value">{price}</span>
                    </div>
                    <div className="buttons">
                      <IconButton className="add-to-cart" onClick={() => addToCart(item)}
title='Add to cart'><AddShoppingCartTwoToneIcon/></IconButton>
                      <a href="#">
                        <IconButton className="buy-now" title='Buy Now'
onClick={handleBuyClick}>
                          <ShoppingCartOutlinedIcon/>
                        </IconButton>
                      </a>
                    </div>
                  </div>
                </div>
              );
            })}
```

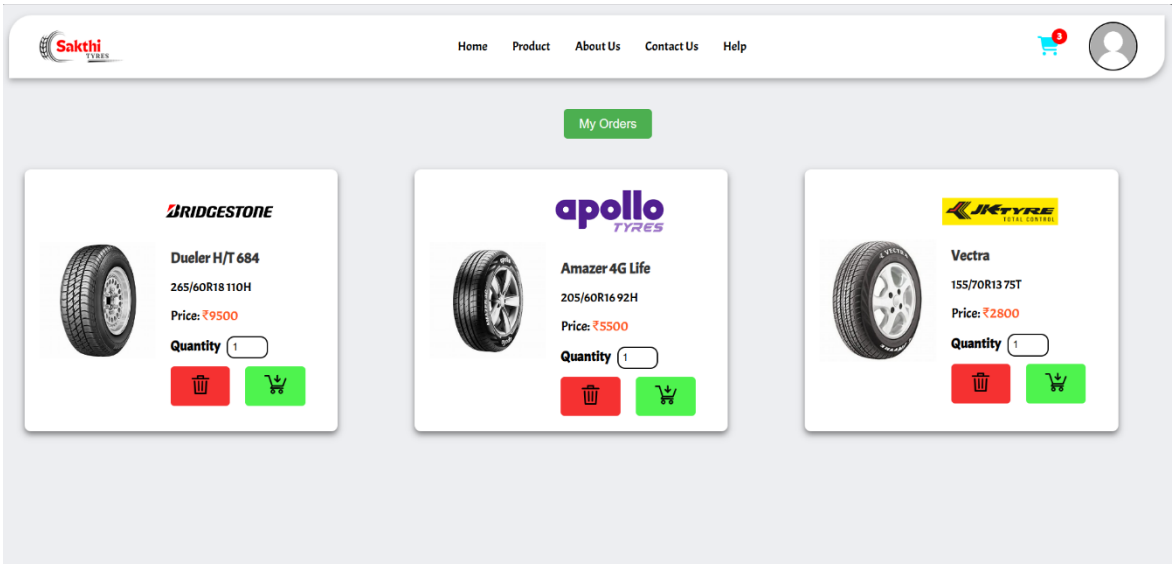
```
            </div>
          </div>
        )}
      </div>
    </div>
    <ToastContainer
      position="top-right"
      autoClose={3000}
      limit={1}
      hideProgressBar={false}
      newestOnTop={false}
      closeOnClick
      rtl={false}
      pauseOnFocusLoss
      draggable
      pauseOnHover
      theme="dark"
    />
        {isPaymentModalOpen && <Payment onClose={handleClosePaymentModal} />}

  </>
 );
}
export default Product;
```

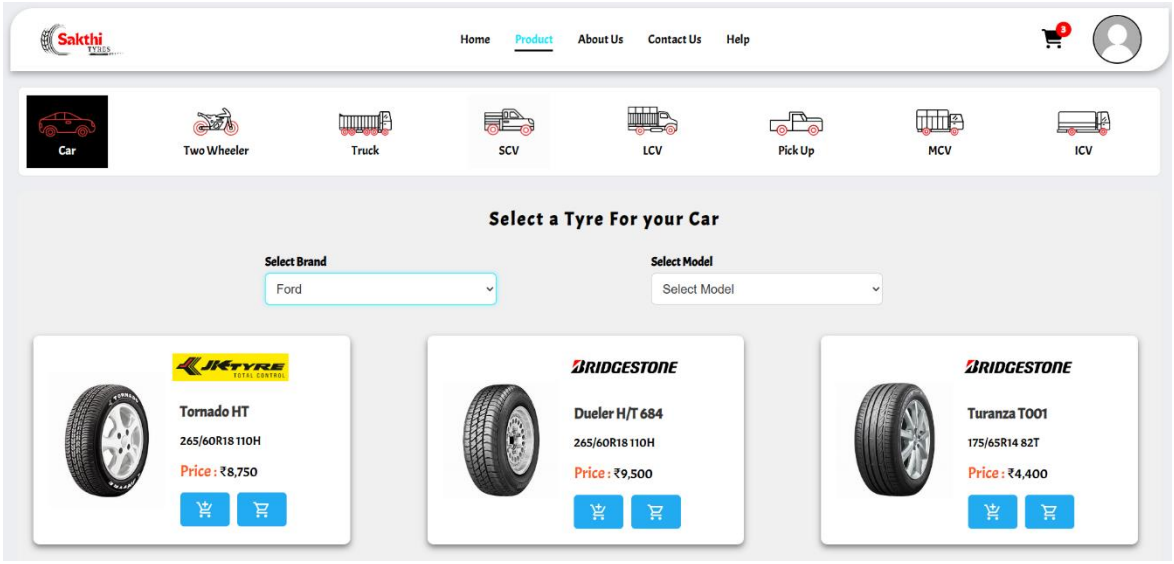
# APPENDIX 2- SCREENSHOTS

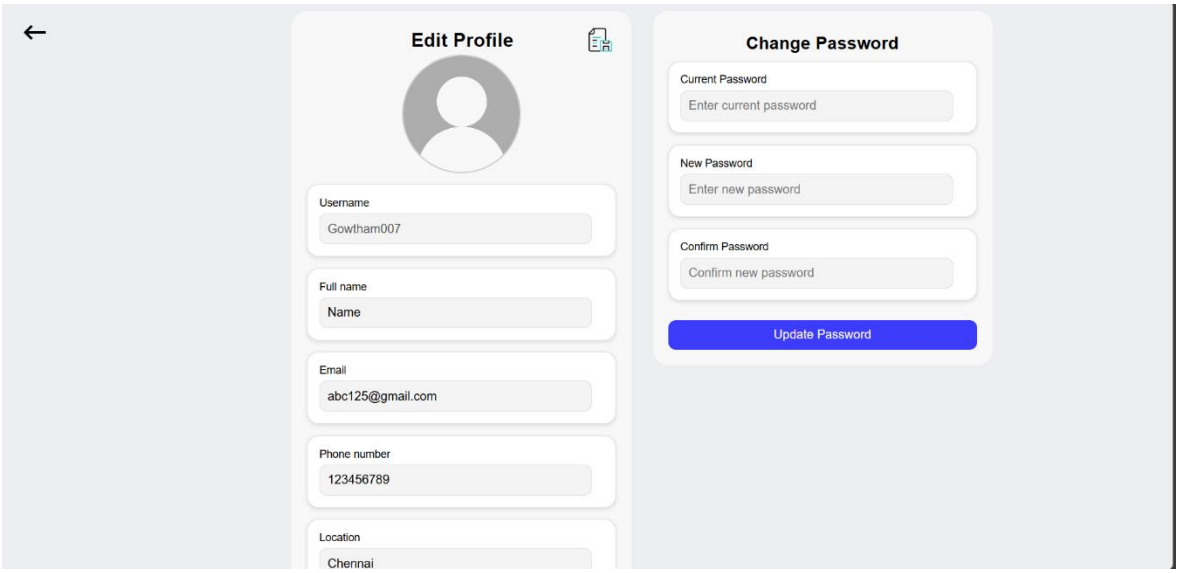


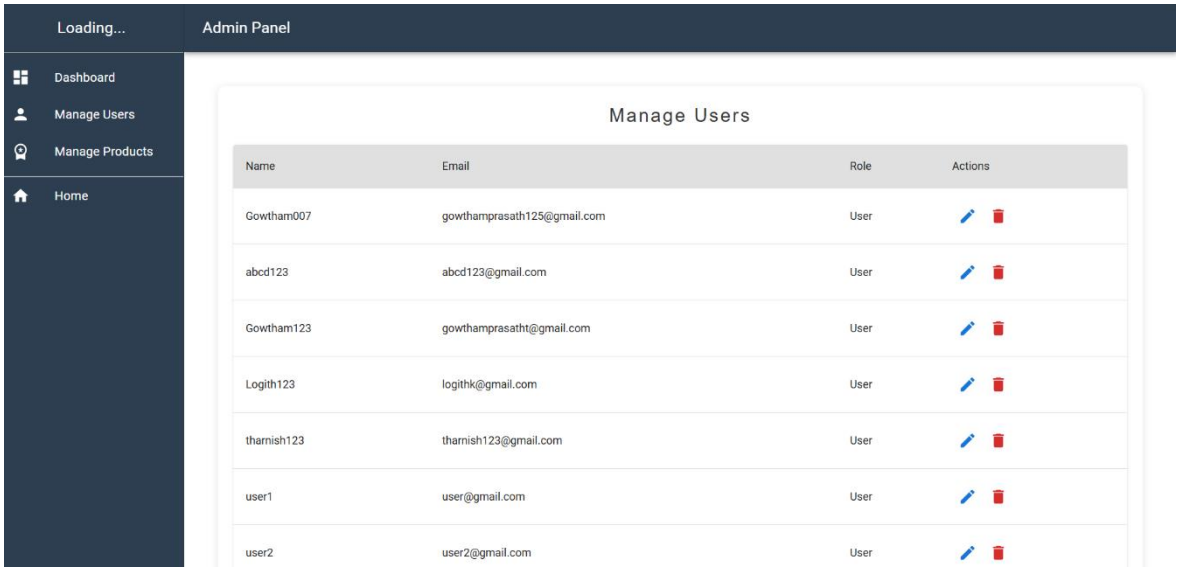



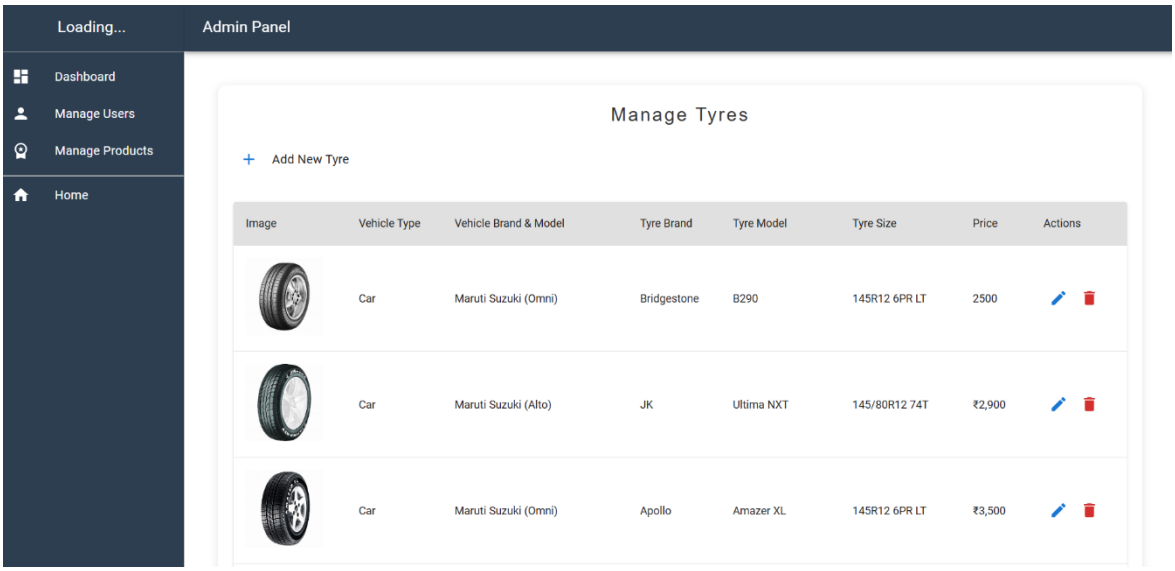

Sakthi
Home
Product
About Us
Contact Us
Help
apollo
TYRES
ADD MORE THRILL
TO YOUR BEAST
INTRODUCING APOLLO ALPHA
STEEL RADIAL TYRE
India's first steel belted
radial motorcycle tyre
that gives you better grip and adds
more stability to your ride.
So go ahead, let it roar.
Distortion-Free All-Steel Layering
Outstanding High-Speed Stability
Precise Cornering Stability


Sakthi
Home
Product
About Us
Contact Us
Help
My Orders
BRIDGESTONE
Dueler H/T 684
265/60R18 110H
Price: ₹9500
Quantity 1
apollo
TYRES
Amazer 4G Life
205/60R16 92H
Price: ₹5500
Quantity 1
JK TYRE
Vectra
155/70R13 75T
Price: ₹2800
Quantity 1


Sakthi
Home
Product
About Us
Contact Us
Help
Car
Two Wheeler
Truck
SCV
LCV
Pick Up
MCV
ICV
Select a Tyre For your Car
Select Brand
Ford
Select Model
Select Model
JK TYRE
Tornado HT
265/60R18 110H
Price : ₹8,750
BRIDGESTONE
Dueler H/T 684
265/60R18 110H
Price : ₹9,500
BRIDGESTONE
Turanza T001
175/65R14 82T
Price : ₹4,400


Edit Profile
Username
Gowtham007
Full name
Name
Email
abc125@gmail.com
Phone number
123456789
Location
Chennai
Change Password
Current Password
Enter current password
New Password
Enter new password
Confirm Password
Confirm new password
Update Password


Loading...
Admin Panel
Dashboard
Manage Users
Manage Products
Home
Manage Users
Name
Email
Role
Actions
Gowtham007
gowthamprasath125@gmail.com
User
abcd123
abcd123@gmail.com
User
Gowtham123
gowthamprasatht@gmail.com
User
Logith123
logithk@gmail.com
User
tharnish123
tharnish123@gmail.com
User
user1
user@gmail.com
User
user2
user2@gmail.com
User


Loading...
Admin Panel
Dashboard
Manage Users
Manage Products
Home
Manage Tyres
Add New Tyre
Image
Vehicle Type
Vehicle Brand & Model
Tyre Brand
Tyre Model
Tyre Size
Price
Actions
Car
Maruti Suzuki (Omni)
Bridgestone
B290
145R12 6PR LT
2500
Car
Maruti Suzuki (Alto)
JK
Ultima NXT
145/80R12 74T
₹2,900
Car
Maruti Suzuki (Omni)
Apollo
Amazer XL
145R12 6PR LT
₹3,500

# REFERENCES

**[1] Ethan Brown**, "Web Development with Node and Express: Leveraging the JavaScript Stack," O'Reilly Media, First Edition, 2014.

**[2] Alex Banks & Eve Porcello**, "Learning React: Modern Patterns for Developing React Apps." O'Reilly Media, Third Edition, 2023.

**[3] Vasan Subramanian**, "Pro MERN Stack: Full Stack Web App Development with Mongo. Express. React, and Node," Apress, Second Edition, 2019.

**[4] Robin Wieruch**, "The Road to React: Your Journey to Master Plain yet Pragmatic React.js." Independent Publishing, Fifth Edition, 2022.

**[5] Andrew Mead & Rob Percival**, "The Complete Node.js Developer Course: Build RESTful APIs with Node, Express, Mongo DB." Udemy, First Edition, 2020.

## WEBSITE LINKS

**[1]** https://stackoverflow.com/

**[2]** https://Tutorialspoint.com/

**[3]** https://github.com/

**[4]** https://www.w3schools.com/

**[5]** https://medium.com/

SAKTHI TYRES
591/1, Mysore Trunk Road,
Rangasamudram,
Sathyamangalam - 638 402.
Mail: chithranatarajan99@gmail.com

Chithra Natarajan
94430 88649
93633 29176
GSTIN NO : 33BVUPC8O2 1P1Z4

We thank that the following students of CT-UG department, Kongu Engineering College, Perundurai for making an Web Application for our Tyre shop.

The project is successfully implemented in our environment.

1. Mr. Gowtham Prasath T (22BIR014)
2. Mr. Logith K (22BIR027)
3. Mr. Tharnish P (22BIR053)

Project Guide: Mr B Ravisankar BE. ME.,

N. chithra

GSTIN No. 33BNBPS6814D1Z7
SAKTHI TYRES
591/1, Mysore Trunk Road, Rangasamudram,
Sathyamangalam - 638 402
Cell: 94430 88649

N. chithra
(Mrs.N.Chithra)
Owner