**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW OF THE PROJECT**

The MANIYAN STORES E-commerce Platform is designed to modernize the shopping experience for departmental goods by providing a seamless and efficient online shopping solution. Developed using the MERN stack (MongoDB, Express.js, React, and Node.js), this platform ensures a dynamic user interface and a robust backend for smooth performance. Key features include Real-Time Inventory Management, Personalized Product Recommendations, Customer Reviews, and Advanced Search and Filtering options, allowing users to find products quickly. Secure authentication is implemented using Firebase and JWT, ensuring safe access and data protection. The integration of Stripe enables secure and hassle-free transactions, while MongoDB efficiently manages product listings and customer data. With responsive design, order tracking, and dedicated customer support, the platform enhances user convenience, builds trust, and supports scalability for future growth.

**1****.2 PROBLEM DEFINITION**

In the traditional departmental stores often face challenges due to manual operations and disconnected systems, leading to inefficiencies in inventory management, customer service, and order processing. Without an integrated e-commerce platform, businesses struggle to provide a seamless shopping experience, resulting in limited customer reach, delayed order fulfillment, and difficulties in tracking product availability. Additionally, the absence of secure online payment options and real-time inventory updates restricts the ability to offer a smooth and reliable purchasing process.

The goal of this project is to develop the MANIYAN STORES E-commerce Platform using the MERN stack. This platform will address these challenges by providing a scalable, secure, and user-friendly online shopping solution. Key features include Real-Time Inventory Management , Personalized Product Recommendations, Secure Payment

Integration with Stripe, and Advanced Search and Filtering. By ensuring seamless order tracking, secure authentication using Firebase and JWT, and an intuitive user interface, the platform will enhance customer satisfaction, streamline business operations, and position MANIYAN STORES for long-term growth in the competitive e-commerce market.

**1.3 OBJECTIVE OF THE PROJECT**

The objective of this project is to develop the MANIYAN STORES E-commerce Platform, designed to enhance the shopping experience for departmental goods while improving operational efficiency and scalability. Built using the MERN stack (MongoDB, Express, React, Node.js), the platform ensures high performance, security, and seamless user interactions. Key features include Real-Time Inventory Management, Secure Authentication with Firebase and JWT, Personalized Product Recommendations, Customer Reviews, Advanced Search and Filtering, and Secure Payment Integration via Stripe. These features will provide users with a smooth and intuitive shopping experience, enabling them to browse, compare, and purchase products effortlessly.

Additionally, the platform will enhance customer engagement by offering easy order tracking, responsive customer support, and personalized shopping suggestions. By ensuring a user-friendly and feature-rich online presence, MANIYAN STORES will expand its customer reach, streamline business operations, and establish itself as a competitive and modern player in the e-commerce industry.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

The current system at MANIYAN STORES relies on traditional in-store shopping and manual processes for inventory, billing, and transactions, causing inefficiencies. Customers must visit the store physically, limiting accessibility and reach. Without an online catalog, the store struggles to adapt to modern shopping trends. Manual inventory updates lead to stock mismatches, while payments rely on cash or basic POS systems, lacking secure digital options. The absence of a customer engagement system results in no personalized recommendations or targeted promotions. Without an integrated e-commerce platform, MANIYAN STORES faces challenges in scaling operations, streamlining processes, and enhancing customer satisfaction, emphasizing the need for a digital solution.

**2.1.1 DISADVANTAGES OF EXISTING SYSTEM**

The existing system has following disadvantages:

* + Inefficient order management, causing crowding and delays at the billing counter.
  + Inconvenient for customers needing to visit the store.
  + No centralized system for managing inventory, orders, or customer data.
  + Limited customer reach due to the absence of an online presence.
  + Higher risk of stock mismatches and sales errors due to manual record-keeping.

**2.2 PROPOSED SYSTEM**

The proposed MANIYAN STORES E-commerce Platform will modernize the business by enabling customers to browse products, check prices, and make purchases online, enhancing the shopping experience.

With features like real-time inventory management, secure payments, and personalized recommendations, the platform will streamline operations, expand reach, and support future growth.

With an intuitive, user-friendly interface powered by the MERN stack (MongoDB, Express, React, Node), customers will easily browse products, check prices, and make purchases online. The system will include features such as Product Management, Customer Orders, Billing, and User Authentication. Administrators will have real-time control over inventory, orders, and customer data, enabling efficient management and informed decision-making.

**2.2.1 ADVANTAGES OF PROPOSED SYSTEM**

The proposed system offers the following advantages:

* Streamlined product browsing and purchasing process, allowing customers to order online conveniently.
* User-friendly interface for easy navigation and browsing of products.
* Real-time inventory management and updates for administrators.
* Centralized platform for managing product listings, customer orders, and data.
* Enhanced customer reach through a scalable online presence, supporting potential growth opportunities.

**2.3 FEASIBILITY STUDY**

Feasibility studies are crucial for evaluating the proposed MANIYAN STORES E-commerce Platform, assessing its functionality, potential impact on the business, ability to meet customer needs, and resource efficiency. While any project may seem feasible with unlimited resources and time, it is essential 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 relationship between feasibility and risk is essential, especially 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 MANIYAN STORES E-commerce Platform. This evaluation is critical, as it involves the analysis of both technology availability and resource requirements. A key consideration is whether the organization has sufficient resources for both development and implementation. Given that the proposed system utilizes existing technologies (such as the MERN stack) and requires minimal additional resources, it is classified as technically feasible.

**2.3.2 OPERATIONAL FEASIBILITY**

Operational feasibility evaluates how effectively the MANIYAN STORES E-commerce Platform meets the needs of both customers and administrators. The 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, expanding its reach. Additionally, by creating better market opportunities for the business, 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 MANIYAN STORES E-commerce Platform is economically advantageous as it does not require additional hardware or software investments, ensuring its financial viability. By carefully analyzing costs against expected returns, the project demonstrates a strong economic rationale, ensuring that the investment will yield positive financial outcomes for the business. This thorough assessment of economic feasibility highlights the project’s potential 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) forms the foundation of web development and is crucial for creating the structure of a website. For the MANIYAN STORES E-commerce Platform, HTML will be used to organize the layout, including sections for product listings,

customer information, and order forms. HTML defines elements like headings, paragraphs, images, and links, ensuring that the website is easy to navigate and visually appealing. Semantic HTML will be employed to ensure well-structured content that is search engine-friendly, improving the platform’s online visibility. HTML also integrates with Cascading Style Sheets (CSS) to style the site and JavaScript to add interactive features, enhancing the user experience. Additionally, HTML attributes provide extra information about elements, supporting the manipulation of content through CSS and JavaScript for a more engaging and dynamic platform.

**3.2.1.2 CSS**

Cascading Style Sheets (CSS) are essential for defining the visual presentation of HTML elements on the MANIYAN STORES E-commerce Platform. CSS allows developers to control 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 this platform, CSS ensures a visually appealing and user-friendly interface, enabling easy navigation for customers to explore products and place orders. CSS also supports responsive design, ensuring the website is accessible and functional across various devices, including desktops, tablets, and smartphones. This is crucial for customer convenience, as users may access the platform from different devices. The use of external, internal, and inline style sheets offers 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 MANIYAN STORES E-commerce Platform, JavaScript plays a vital role in creating interactive features like product search, order placement, 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 MANIYAN STORES E-commerce Platform, React will be used to create a highly interactive and responsive user interface that can handle and display dynamic data such as product listings, user profiles, and order 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 orders 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, order 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 MANIYAN STORES E-commerce Platform, where the system needs to manage various types of data, such as product listings, customer profiles, and order 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 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 MANIYAN STORES E-commerce Platform, Node.js will serve as the server-side engine that processes incoming customer requests, such as product 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 products, placing orders, 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
* Search and Filter
* Cart
* Order
* Payment
* Contact
* Edit Profile
* Admin

**4.1.1 LOGIN/REGISTER**

The Login module allows users to securely log in by entering their username and password, with credentials validated and stored in an encrypted format. The Register module enables users to create an account by providing a username, email, password, and a confirmation password (cpassword). Both modules are designed to ensure secure user access, safeguarding personal information from unauthorized access.

**4.1.2 HOME**

The home page acts as the main entry point for users, offering an intuitive and visually engaging interface for easy navigation through sections like products, about us, contact us, and help. A dynamic top bar displays the logo, user profile options, and a

shopping cart icon with real-time item updates, improving user engagement. If users try to access product features without logging in, a prompt appears to ensure secure access. The page also uses animations and effects to provide a lively, enjoyable user experience.

**4.1.3 PRODUCT**

The product module provides a user-friendly interface for browsing a variety of grocery items, such as fresh produce, packaged goods, dairy products, beverages, and more. Users can easily filter products by categories, brands, and types using dropdown menus. Each product is displayed with high-quality images, detailed specifications (including product name, quantity, brand, and price), and options to add items to the cart. While direct online purchasing may not be available in this module, users can add products for future consideration or use the "Buy Now" feature for quick checkout, enhancing the overall shopping experience.

**4.1.4 SEARCH AND FILTER**

The search and filter module on the product page allows users to efficiently browse and find the grocery items they need. The search bar lets users enter keywords to quickly locate specific products, making the shopping experience faster and more convenient. Alongside the search function, a comprehensive filtering system enables users to narrow down products based on category, type, price range, and brand. This filter system ensures that customers can easily explore products that meet their specific preferences. Each filter is designed to be intuitive, offering clear dropdown options, and the selected filters are prominently displayed for users to review. Additionally, users can remove or reset individual filters with ease, providing full control over their search process. The module enhances overall user experience by offering both flexibility and precision, helping customers find exactly what they’re looking for in a timely manner.

**4.1.5 CART**

The cart module manages the shopping cart functionality for an online grocery shopping platform. It uses React hooks to manage the cart's state, including item quantities, pricing, and payment options. Upon component mount, it fetches cart items from the backend API, initializing their quantities and prices. Users can update item quantities, remove products, and initiate the checkout process for selected items. The component displays loading animations while fetching data and a notification if the cart is empty. Visual effects are enhanced with Lottie animations, and Axios is used for API requests, ensuring a seamless shopping experience with options for order and payment processing.

**4.1.6 ORDER**

The order module is an essential component of the online grocery shopping platform, streamlining the process from cart to final purchase. Once users are ready to check out, they can view a detailed summary of their selected items, including product names, quantities, prices, and total cost. The module enables users to enter their shipping information and choose from available payment options, ensuring a smooth and secure transaction. After the order is placed, users receive an order confirmation, and the backend processes the transaction, updating inventory and notifying users about order status. This module enhances the overall shopping experience by offering an intuitive and efficient flow, allowing users to easily track and complete their grocery orders with confidence.

**4.1.7 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.

**4.1.8 CONTACT**

The contact module allows users to easily get in touch for assistance regarding their grocery shopping experience. Users can fill out a form to submit inquiries, feedback, or requests, which will be forwarded to the relevant support team for quick resolution. Additionally, key contact details such as a dedicated support phone number and email address are provided to enable direct communication. This module ensures that user inquiries and concerns are addressed promptly, contributing to improved customer satisfaction and overall support.

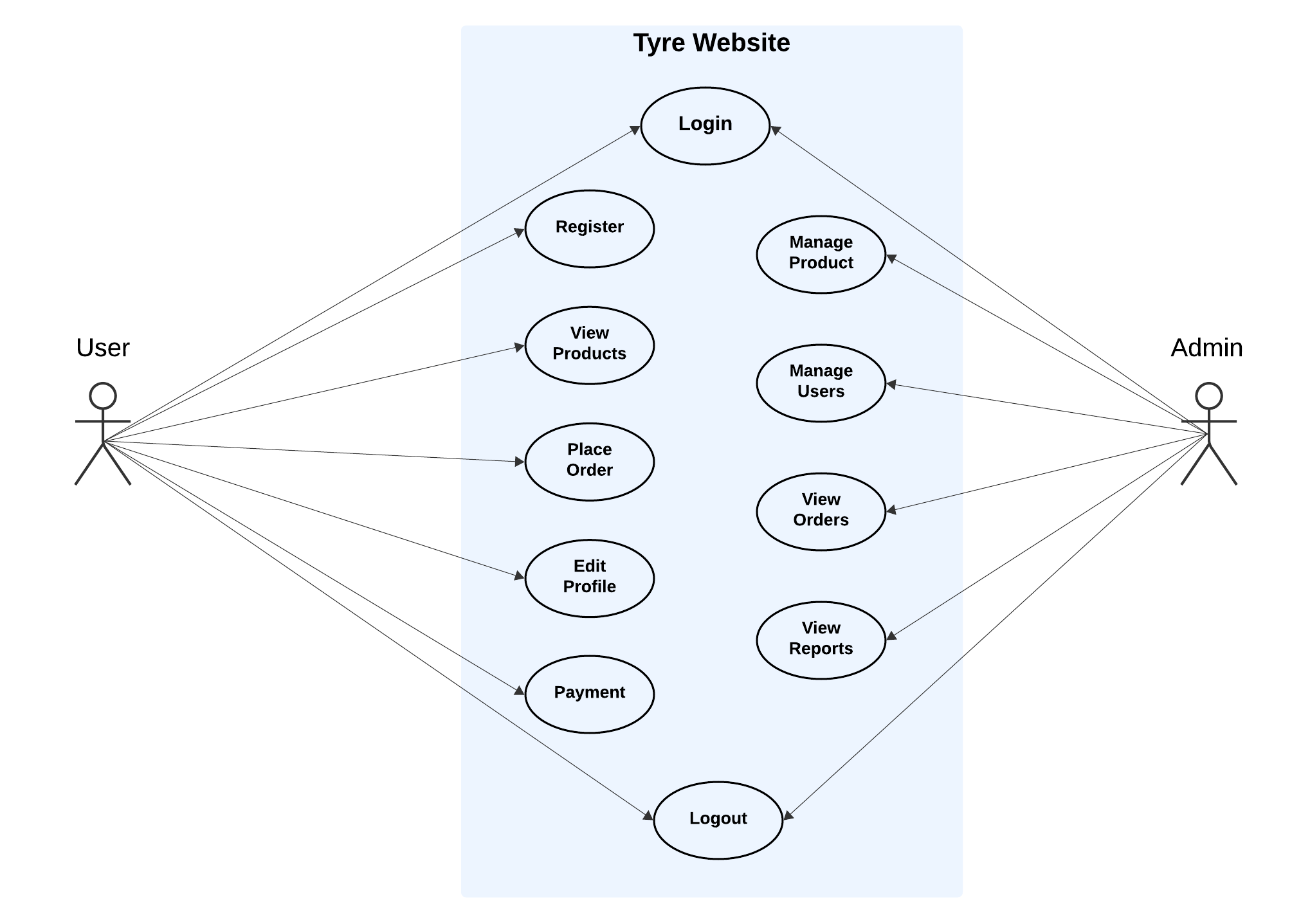
**4.1.9 EDIT PROFILE**

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.

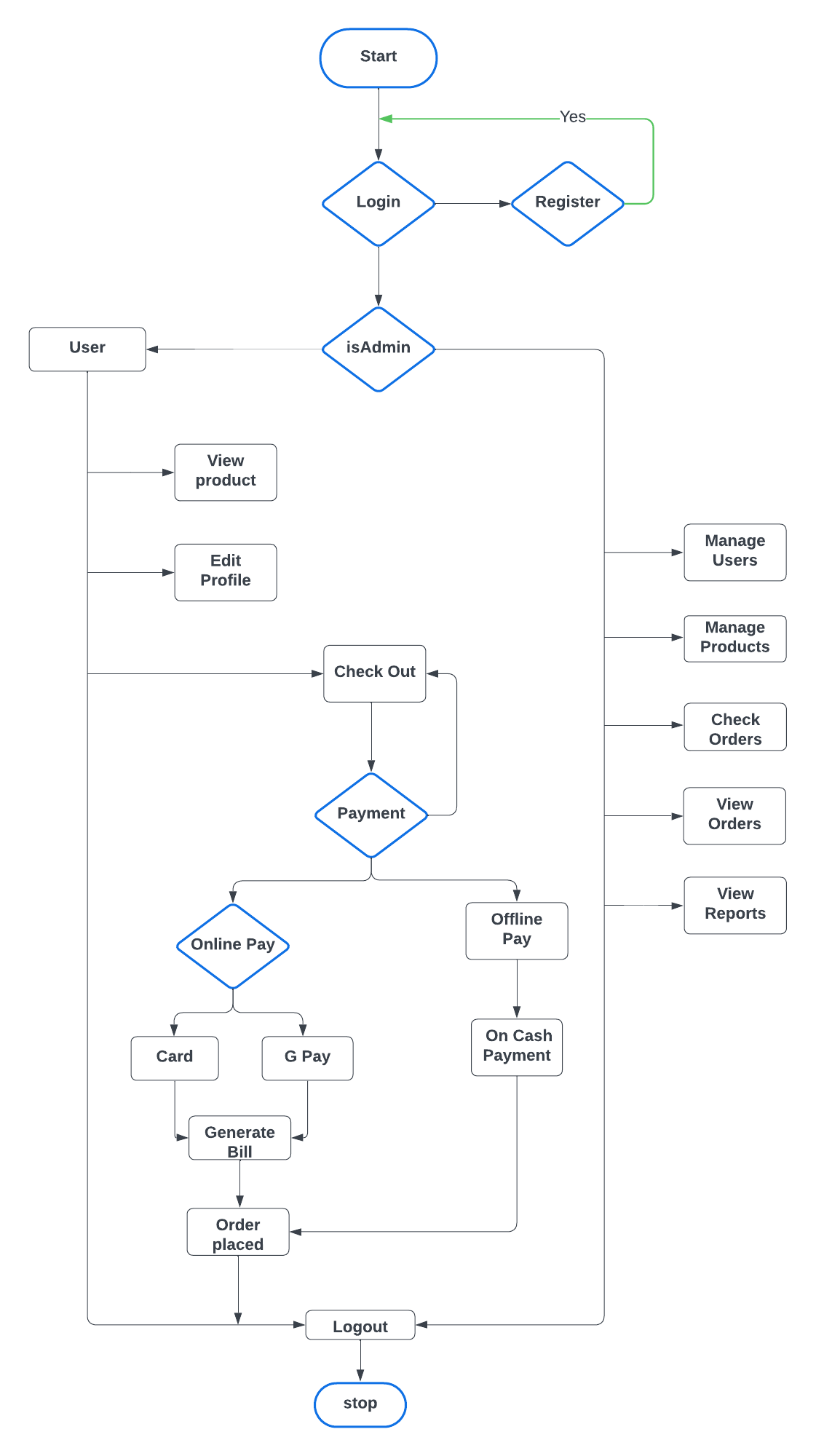
**4.1.10 ADMIN**

The Admin module provides administrators with full control over the platform's operations. Admins can view, manage, and update user profiles and product listings, ensuring the platform remains organized and up-to-date. They also have the ability to track payment records, including transactions made via UPI or cash, ensuring accurate financial logs. In addition, admins can edit or remove product content as necessary, maintaining the accuracy and relevance of the product database. This module streamlines the management and monitoring of platform activities, enhancing overall efficiency and ensuring the system runs smoothly.

**4.2 USE CASE DIAGRAM**

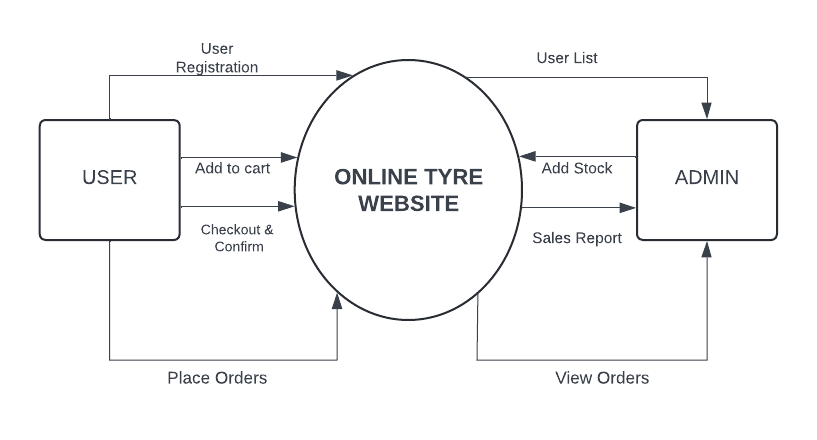


**4.3 SYSTEM FLOW DIAGRAM**

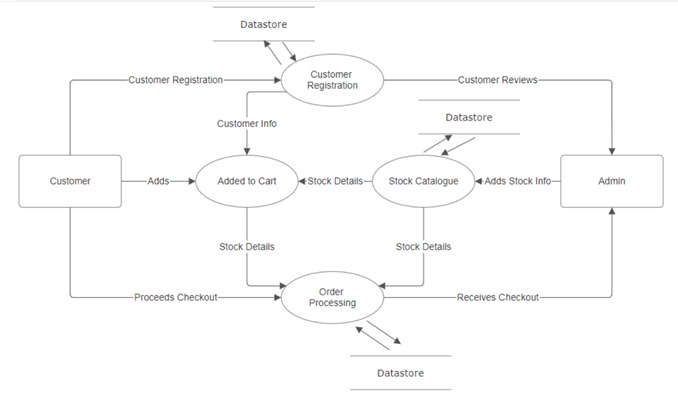


**4.4 DATA FLOW DIAGRAM**

**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** |
| name | String | Username |
| email | String | Email |
| password | String | Password |
| isGoogleUser | Boolean | User Login Method |
| isAdmin | Boolean | User / Admin Login |

**TABLE NO** : 4.5.2

**TABLE NAME** : products

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

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| productImage | String | Product image |
| productCategory | String | Product Category |
| productType | String | Product Type |
| productBrand | String | Product Brand |
| productName | String | Product Name |
| productUnit | String | Product Unit |
| productContainerType | String | Product Container Type |
| productExpirationPeriod | String | Product Expiration Period |
| productPrice | Number | Product Price |
| productPreviousPrice | Number | Product Previous Price |
| productStock | Number | Product Stock |

**TABLE NO** : 4.5.3

**TABLE NAME** : carts

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

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| userId | String | User id |
| products | Array | Product items |

**TABLE NO** : 4.5.4

**TABLE NAME** : orders

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

|  |  |  |
| --- | --- | --- |
| **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| userId | String | User id |
| items | Array | Product item |
| totalPrice | Number | Total price |
| paymentType | String | Payment type |
| paymentMethod | String | Payment method |
| productState | String | Product state |

**4.6 INPUT DESIGN**

In the proposed work**,** the input forms designed include the Customer Registration Form. The Customer Registration Form collects essential details such as Email ID and Password. Validation is implemented for the password to ensure it meets the required length criteria. If the input is invalid or incomplete, appropriate error messages are displayed to guide users. Following is the Input design:

* + Login form
  + Registration form
  + Edit profile form

**LOGIN FORM**

The Login Form is the gateway for users to access their accounts, requiring an email ID and password for authentication. It ensures a seamless and secure login process with validation checks, guiding users through clear error messages if credentials are incorrect or missing. The form also includes a forgot password option for account recovery and supports Google Sign-In for convenience. With a responsive design, it offers a smooth experience across different devices, encouraging effortless access to the platform.

**REGISTER FORM**

The Register Form in Maniyan Stores enables users to create an account by providing their name, email, password, and confirmation password. It ensures secure registration with email validation, password length checks, and a mismatch warning to prevent errors. Clear messages guide users through corrections, and a success notification confirms registration. The form maintains a user-friendly design, featuring a responsive layout and engaging animations, ensuring accessibility across devices. Additionally, users can quickly navigate to the sign-in page if they already have an account, enhancing the overall experience.

**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 systematically compiles product details, including product name, brand, category, unit, container type, expiration period, and pricing, ensuring a transparent and organized billing process. It highlights essential transaction information such as stock availability and previous vs. current pricing, allowing users to make informed purchase decisions. The bill further includes timestamps for tracking and supports downloading a detailed invoice in PDF format, making it convenient for users to maintain a record of their grocery 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 input 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, the admin's username and password are verified. If correct, the Admin Page will be displayed.

will redirect to the main page.

**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 : Product 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 of setting up a new system, configuring software, organizing data, preparing documentation, establishing procedures, and ensuring a smooth transition to the new digital environment.

**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 essential to transform design concepts into functional solutions that serve both users and the business. With meticulous planning and execution, we can minimize errors and ensure a smooth transition into a fully operational system. By conducting comprehensive testing and validation, we build trust with users and lay the foundation for future system improvements. A successful implementation not only boosts efficiency but also plays a pivotal role in the long-term success of the organization.

Thus, the website for Maniyan Stores has been successfully developed, leveraging 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.

* + Implement real-time notifications to keep users informed about their orders, offers, and updates.
  + Allow customer reviews and ratings to help others make informed purchase decisions.
  + Offer promotions and discounts to engage customers and drive more sales.
  + Develop a mobile app to extend the platform’s reach and offer more convenience for users on the go.

**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\_51Q5g2GD7L0PMiSZDyh1Slqidawdli8iWnIGxx69koWIyfEpliXrlqPBaDqtTtiiiee6upIoioleWHdwXxZDzTvdU00LXTGyT2G");

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')}>&nbsp;&nbsp;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} />}

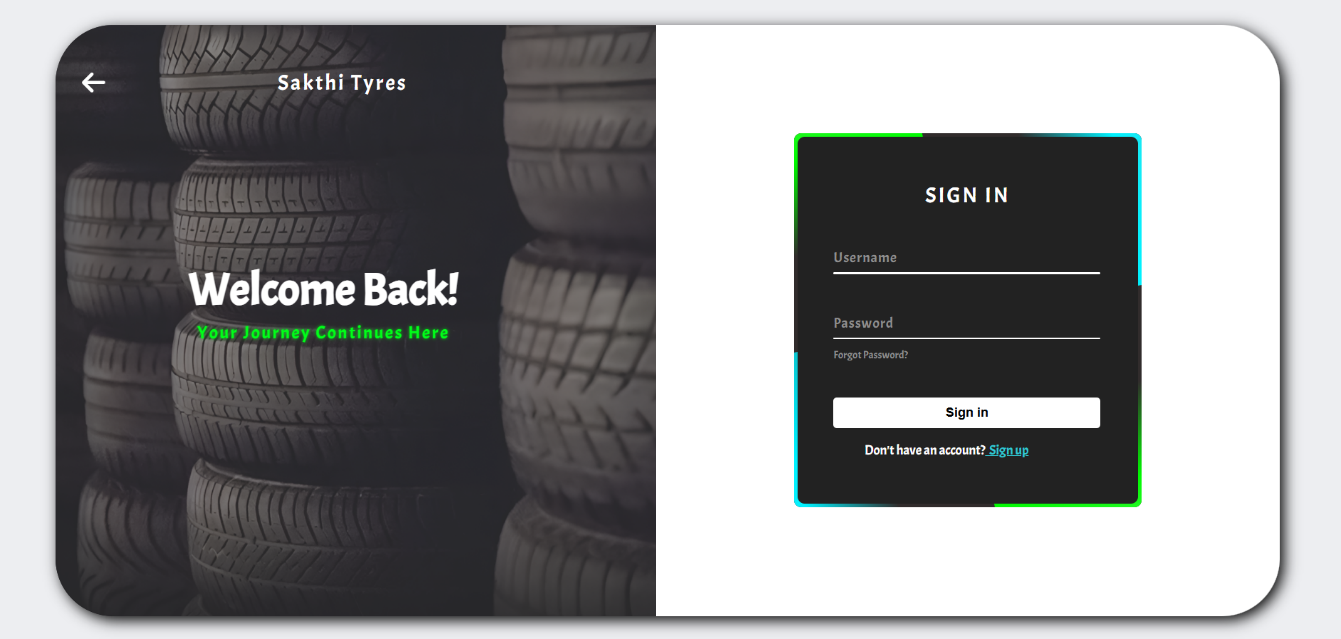
    </>

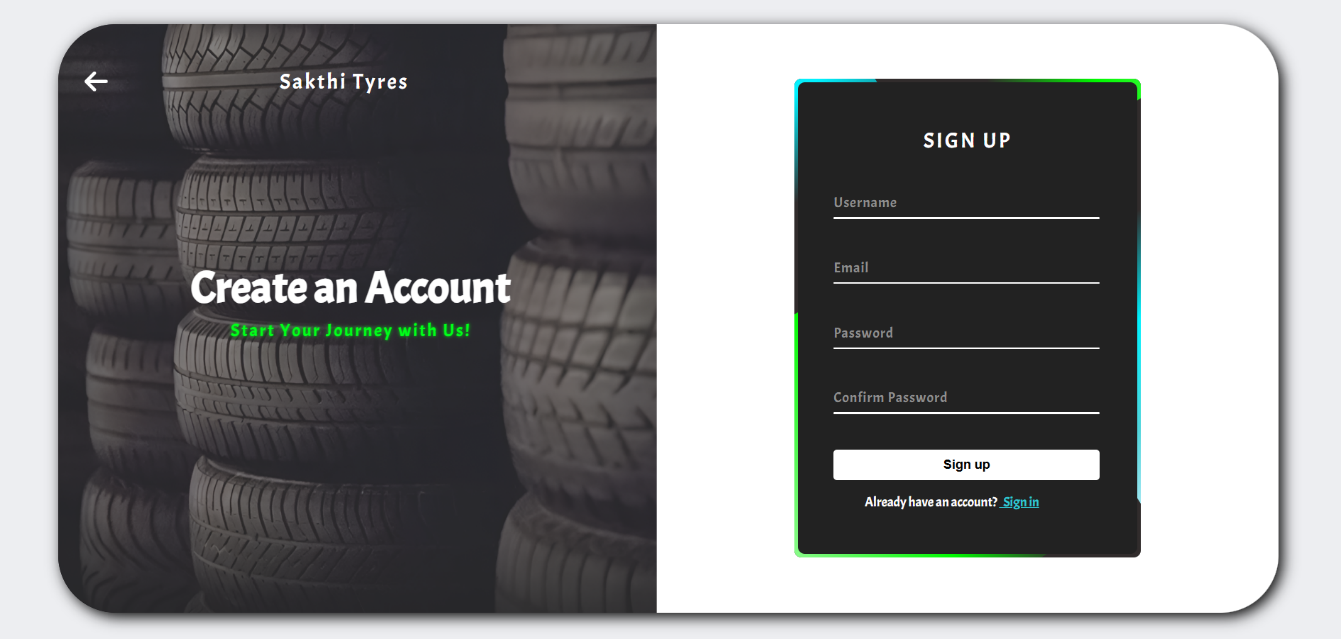
  );

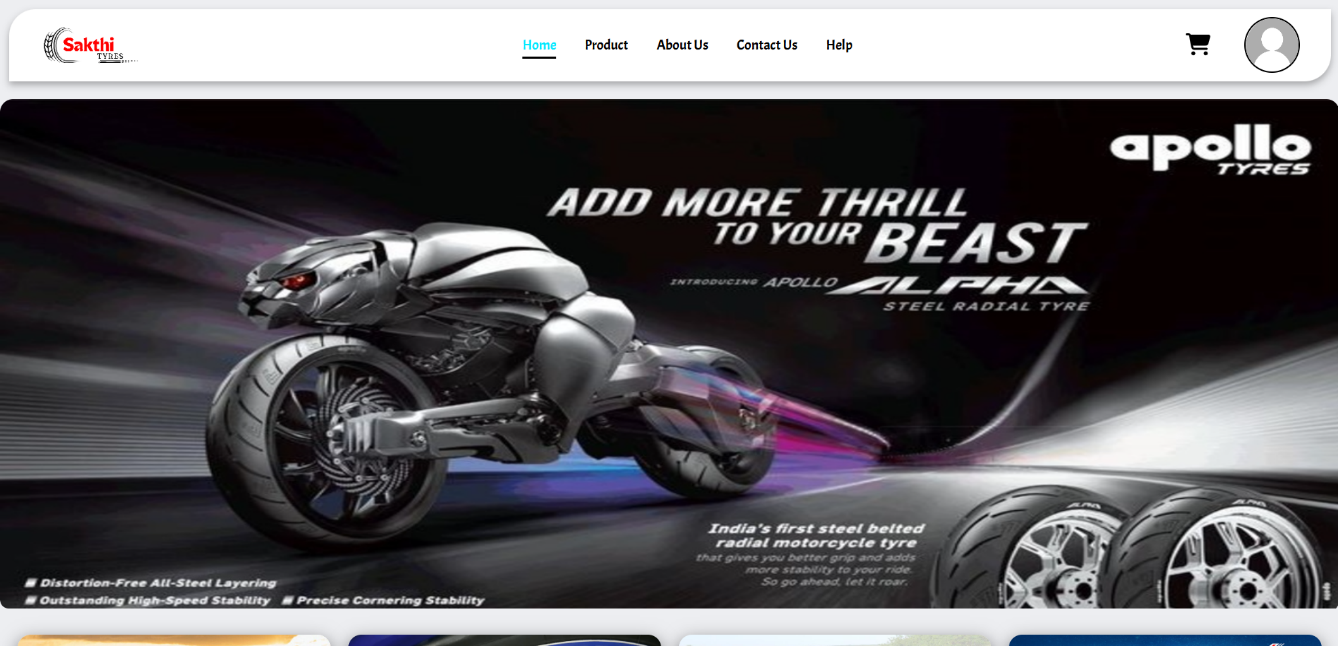
}

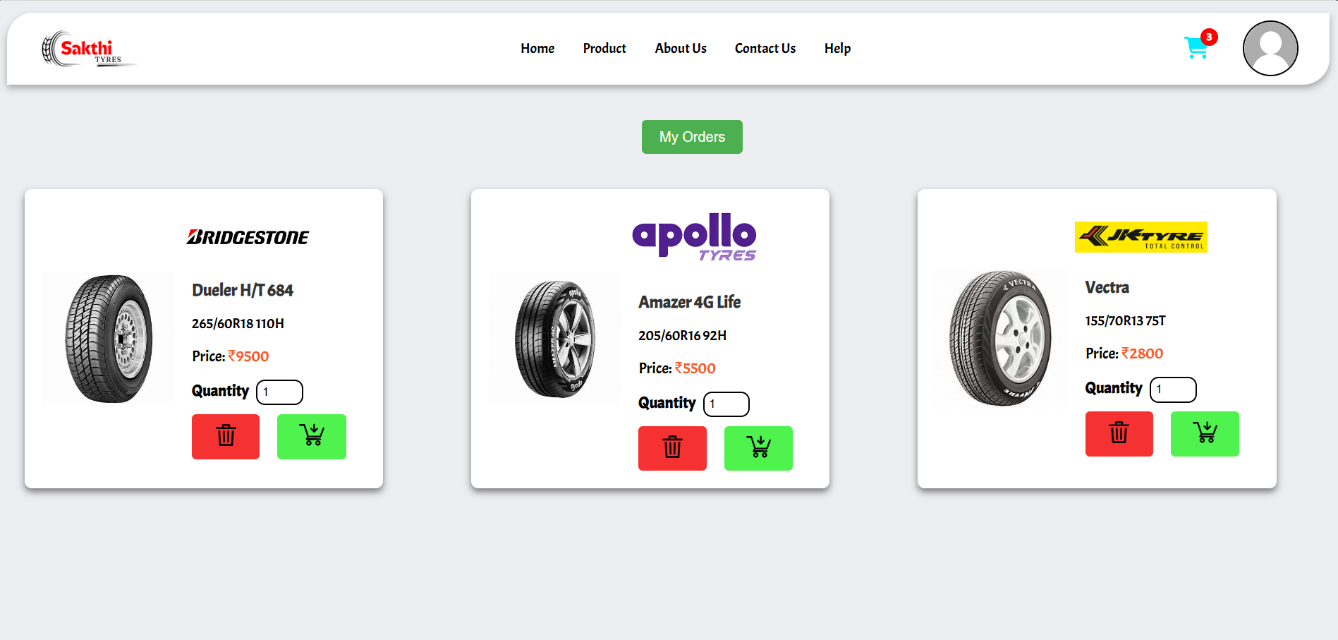
export default Product;

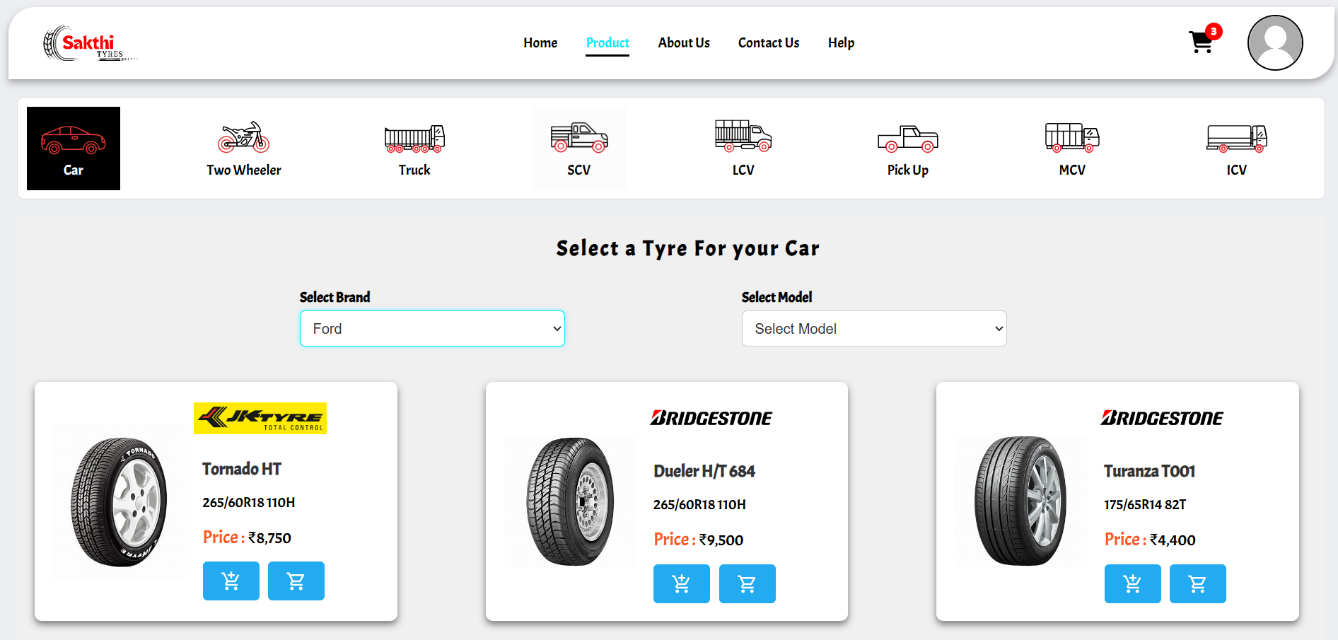
**APPENDIX 2- SCREENSHOTS**

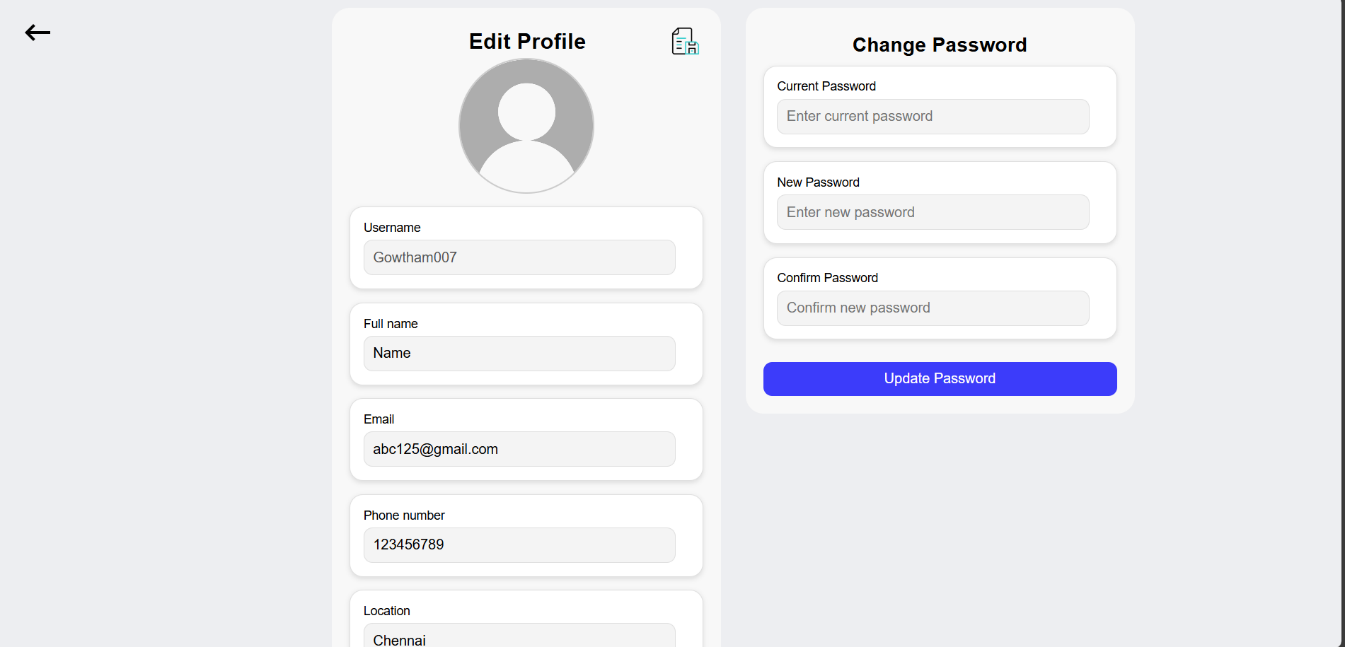
****

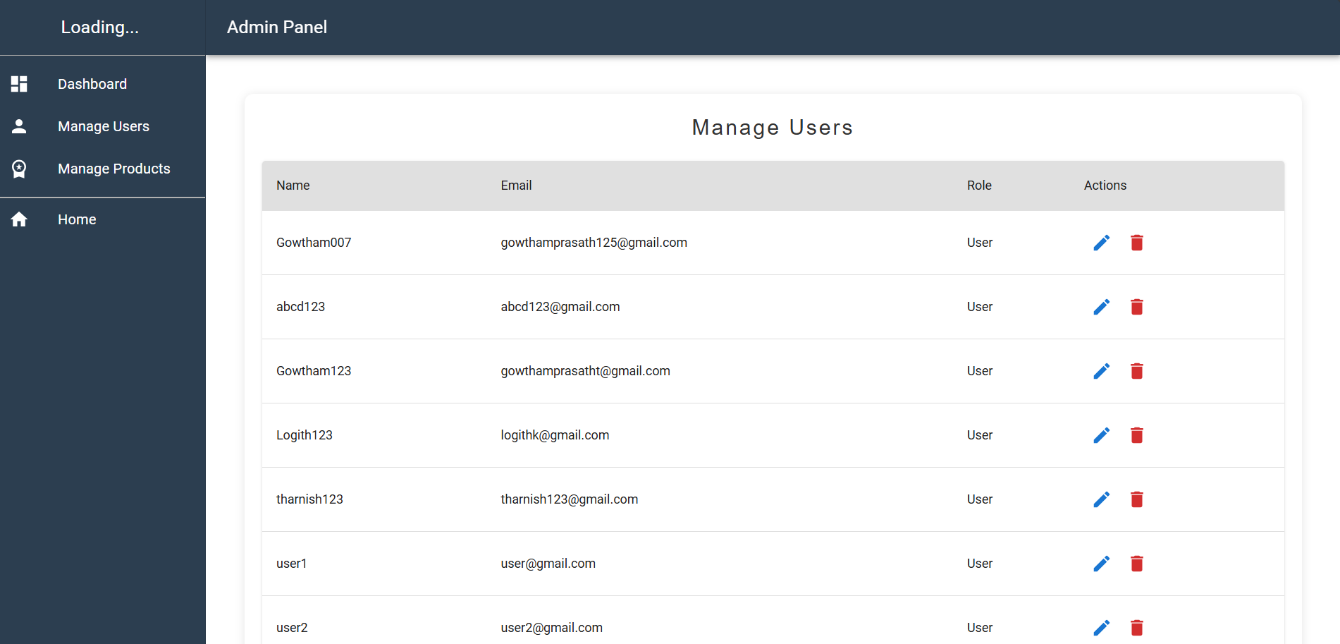


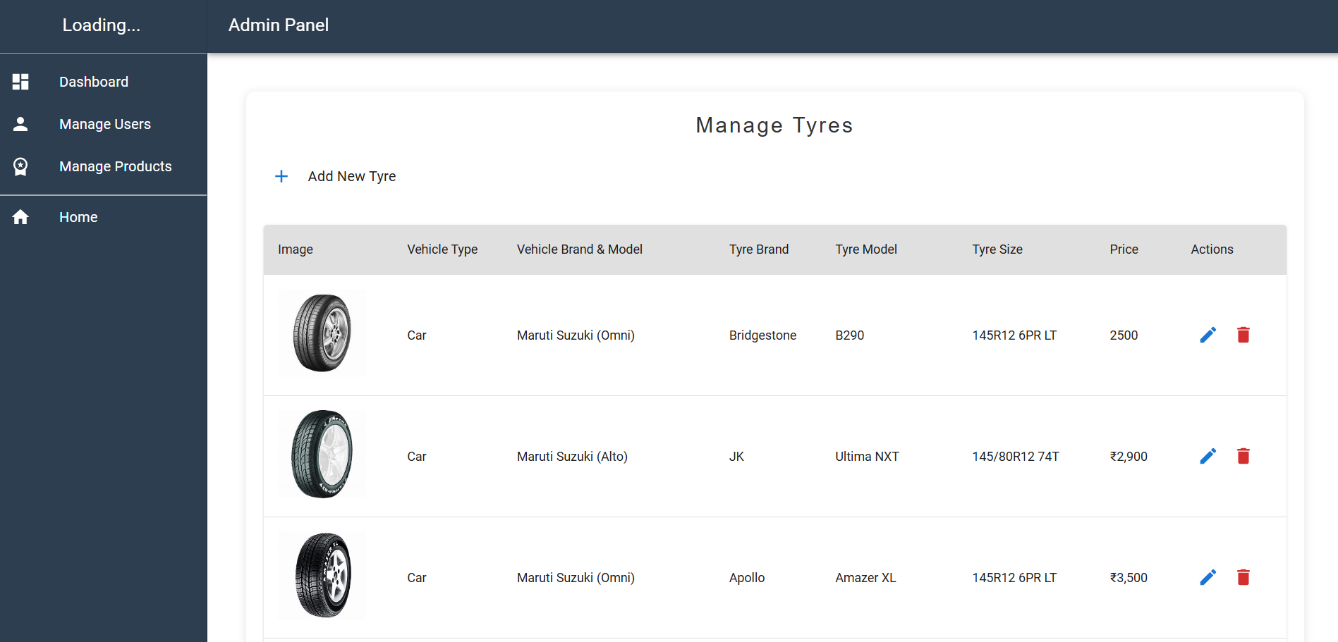










****

**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/>

