

# **WEBSITE DEVELOPMENT FOR ONLINE MEDICAL RETAIL SHOP**

## **A PROJECT REPORT**

**Submitted by**

**RAJARAM S**

(Reg. No: 24MCR080)

**SACHIN S**

(Reg. No: 24MCR084)

**SANGEETHA M**

(Reg. No: 24MCR089)

*in partial fulfilment of the requirements for*

*the award of the degree of*

**MASTER OF COMPUTER APPLICATIONS**

**DEPARTMENT OF COMPUTER APPLICATIONS**



**KONGU ENGINEERING COLLEGE**

**(Autonomous)**

**PERUNDURAI, ERODE – 638 060**

**DECEMBER 2024**

**DEPARTMENT OF COMPUTER APPLICATIONS****KONGU ENGINEERING COLLEGE****(Autonomous)****PERUNDURAI, ERODE – 638 060****DECEMBER 2024****BONAFIDE CERTIFICATE**

This is to certify that the project report entitled “**WEBSITE DEVELOPMENT FOR ONLINE MEDICAL RETAIL SHOP**” is the bonafide record of project work done by **RAJARAM S (24MCR080)**, **SACHIN S (24MCR084)** and **SANGEETHA M (24MCR089)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications of Anna University, Chennai during the year 2024-2025.

**SUPERVISOR****HEAD OF THE DEPARTMENT****(Signature with seal)****Date:**

Submitted for the end semester viva-voice examination held on \_\_\_\_\_

**INTERNAL EXAMINER****EXTERNAL EXAMINER**

## **DECLARATION**

I affirm that the project report entitled “**WEBSITE DEVELOPMENT FOR ONLINE MEDICAL RETAIL SHOP**” being submitted in partial fulfilment of the requirements for the award of Master of Computer Applications is the original work carried out by us. It has not formed the part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**DATE:**

**RAJARAM S**

**(Reg. No: 24MCR080)**

**SACHIN S**

**(Reg. No: 24MCR084)**

**SANGEETHA M**

**(Reg. No. 24MCR089)**

I certify that the declaration made by the above candidates is true to the best of my knowledge.

**Date:**

**Name and Signature of the Supervisor**

## **ABSTRACT**

This project involves the healthcare industry is rapidly evolving through technological advancements, driving the demand for efficient and reliable access to medical supplies. Medicare Shop is a console-based online store designed to transform medical product procurement by offering a secure, user-friendly, and scalable solution. Leveraging modern web technologies such as React.js for a responsive frontend and Firebase for real-time backend services, the platform addresses challenges like limited stock availability and delayed pricing updates.

Key features include a well-organized product catalog, real-time updates on stock and pricing, and robust security measures to protect user data. With an intuitive console-based design, Medicare Shop ensures accessibility for users of all technical levels, streamlining the process of purchasing medical supplies and bridging the gap between healthcare and technology.

In addition to its core functionalities, Medicare Shop emphasizes inclusivity and adaptability, making it suitable for both individual users and healthcare institutions. By streamlining medical supply procurement, the platform reduces inefficiencies, ensures timely updates, and enhances user satisfaction. This innovative integration of advanced technologies into healthcare procurement positions Medicare Shop as a vital tool in improving the accessibility and reliability of medical products in an increasingly digital world.

## ACKNOWLEDGEMENT

We respect and thank our correspondent **THIRU.A.K.ILANGO, B.Com.,M.B.A.,LLB.,**and our Principal **Dr.V.BALUSAMY B.E(Hons)., M.Tech, PhD.** Kongu Engineering College, Perundurai for providing us with the facilities offered.

We convey our gratitude and heartfelt thanks to our Head of the Department **Dr.A.TAMILARASI MSc., MPhil., PhD.,** Department of Computer Applications, Kongu Engineering College, for her perfect guidance and support that made this work to be completed successfully.

We also wish to express my gratitude and sincere thanks to our project coordinator **Mrs.S.HEMALATHA MCA., PhD(pursuing) .** Associate Professor(s), Department of Computer Applications, Kongu engineering College, who have motivated us in all aspects for completing the project in the scheduled time.

We would like to express our gratitude and sincere thanks to our project guide **Dr. K.CHITRA MCA., M.Phil., PhD** Department of Computer Applications, Kongu Engineering College for giving her valuable guidance and suggestions which helped us in the successful completion of the project.

We owe a great deal of gratitude to our parents for helping overwhelm in all proceedings. We bow our heart with heartfelt thanks to all those who thought us their warm services to succeed and achieve our work.

## TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	iv
	ACKOWLEGEMENT	v
	LIST OF FIGURES	vi
	LIST OF ABBREVIATIONS	vii
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 ABOUT THE PROJECT	1
	1.2 EXISTING SYSTEM	2
	1.3 DRAWBACKS OF EXISTING SYSTEM	2
	1.4 PROPOSED SYSTEM	3
	1.5 ADVANTAGES OF PROPOSED SYSTEM	4
<b>2</b>	<b>SYSTEM ANALYSIS</b>	<b>5</b>
	2.1 IDENTIFICATION OF NEED	5
	2.2 FEASIBILITY STUDY	5
	2.2.1 Technical Feasibility	6
	2.2.2 Operational Feasibility	7
	2.2.3 Economic Feasibility	7
	2.3 SOFTWARE REQUIREMENTS	
	SPECIFICATION	8
	2.3.1 Hardware Requirements	10
	2.2.2 Software Requirements	11
	2.4 SOFTWARE DESCRIPTION	11
<b>3</b>	<b>SYSTEM DESIGN</b>	<b>15</b>
	3.1 MODULE DESCRIPTION	15
	3.2 DATAFLOW DIAGRAM	16
	3.3 DATABASE DESIGN	18
	3.4 TABLE DESIGN	21
	3.5 INPUT DESIGN	22

	3.6 OUTPUT DESIGN	24
<b>4</b>	<b>IMPLEMENTATION</b>	25
	4.1 SYSTEM IMPLEMENTATION	25
	4.2 STANDARDIZATION OF THE CODING	26
	4.3 ERROR HANDLING	26
<b>5</b>	<b>TESTING AND RESULTS</b>	27
	5.1 TESTING	27
	5.1.1 Unit Testing	28
	5.1.2 Integration Testing	28
	5.1.3 Validation Testing	28
<b>6</b>	<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	30
	6.1 CONCLUSION	30
	6.2 FUTURE ENHANCEMEN	30
	<b>APPENDICES</b>	31
	A. SAMPLE CODING	31
	B. SCREENSHOTS	40
	<b>REFERENCES</b>	45

## LIST OF FIGURES

<b>FIGURE No.</b>	<b>FIGURE NAME</b>	<b>PAGE No.</b>
3.1	Level 0	17
3.2	Level 1 admin side	17
3.3	Level 1 user side	18
3.4	Class Based Diagram	20
3.8	User Registration Form	22
3.9	User Login Page	23
3.10	Admin Login Page	23
3.11	Admin Dashboard	24
5.1	Register Page User exists	29
5.2	Login Page	29
B.1	Sign-Up page	40
B.2	Login	40
B.3	Home Page	41
B.4	Product Category	41
B.5	Add to Cart	42
B.6	View Order Page	42
B.7	Payment Page	43
B.8	Payment verification Page	43
B.9	Admin Login Page	44
B.10	Add Product	44



**LIST OF ABBREVIATIONS**

JS	Java Script
VS CODE	Visual Studio Code
JWT	JSON Web Tokens
DB	DataBase
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
JSX	JavaScript XML
JSON	JavaScript Object Notation
API	Application Programming Interface
DOM	Document Object Model
NPM	Node Package Manager

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 ABOUT THE PROJECT**

The healthcare industry is undergoing rapid technological advancements, driven by the need to provide efficient and reliable access to medical products. In this context, Medicare Shop emerges as a console-based online store designed to revolutionize the way users shop for medical supplies. This platform leverages modern web technologies such as React.js and Firebase, combining them to deliver a user- friendly, secure, and scalable solution tailored to meet the diverse needs of both individual and institutional users.

Medicare Shop addresses common issues in traditional medical product procurement, such as limited stock availability and delayed updates on pricing. It offers an intuitive and streamlined console-based design, making it accessible to users of all technical backgrounds. Powered by React.js, the frontend ensures a dynamic and responsive user experience, while Firebase serves as the backend, providing real- time updates on stock and pricing, secure user authentication, and a real-time database.

Key features of Medicare Shop include a well-organized product catalog that categorizes items for easy navigation, real-time updates to keep users informed about the latest stock and pricing, and robust security measures to protect user data. Its console- based interface is designed with accessibility in mind, allowing users with varying levels of technical expertise to shop with confidence and ease. By integrating these advanced technologies, Medicare Shop bridges the gap between healthcare and technology, making it easier for users to purchase medical supplies quickly and securely.

## **1.2 EXISTING SYSTEM**

### **1. Physical Stores:**

Users are limited to local inventory and geographical locations, requiring visits to stores in person. This can be especially inconvenient for users in remote areas or those with limited access to transportation. Additionally, physical stores often have long wait times, limited stock, and no real-time updates on product availability or pricing.

### **2. General E-commerce Platforms:**

Many e-commerce platforms that sell medical products lack features specifically tailored for the healthcare sector, such as product categorization by medical specialty or detailed regulatory information. Stock and pricing updates are often delayed, leading to issues like out-of-stock products at checkout or incorrect pricing. Furthermore, these platforms are not designed with healthcare needs in mind, making it harder for users to find the right products quickly.

### **3. Complex Interfaces and Security Issues:**

The user interfaces of many existing online medical stores are often overly complicated, with unclear navigation or unintuitive design. This presents a barrier for less tech-savvy users, particularly the elderly or those unfamiliar with online shopping. Security is another major concern, as many platforms do not implement adequate encryption or secure payment gateways, making users hesitant to share sensitive information such as medical details and payment data.

## **1.3 DRAWBACKS OF EXISTING SYSTEM**

- Limited Accessibility
- Lack of Real-Time Updates
- Security Risks

- Complicated Interfaces
- Limited Specialization
- Inefficient Search and Navigation
- Lack of Customer Support

## **1.4 PROPOSED SYSTEM**

The proposed Medicare Shop system is a comprehensive platform designed to enhance the online medical product procurement process with features that ensure efficiency, safety, and user satisfaction. The system provides real-time stock updates, allowing users to view and add only in-stock items to their cart, while marking out-of-stock products as unavailable and sending notifications when these items are restocked. Medicines are categorized by age group to ensure safety, restricting users from purchasing products unsuitable for their age.

To enhance security on Medicare Shop, each user account is linked to a unique combination of phone number and email address. This setup prevents unauthorized login attempts using alternate email addresses associated with the same phone number, adding an extra layer of protection against account theft and unauthorized access. The system also includes cart stock reservation, which temporarily holds selected items during the checkout process.

Additional features like purchase limits are implemented to encourage responsible usage, preventing bulk purchases that could lead to shortages or disrupt supply chains. Dynamic notifications alert users to changes in stock levels, allowing them to make informed decisions about purchasing. A saved cart functionality enables users to store items they wish to purchase later, further simplifying the shopping process. These features collectively make Medicare Shop a user-friendly and reliable platform for online medical product procurement, providing a seamless and secure experience for users while ensuring efficient stock.

## 1.5 ADVANTAGES OF PROPOSED SYSTEM

- **Real-Time Inventory Control**

Ensures that users only see products currently available, reducing frustration and preventing failed purchases due to stock unavailability.

- **Cart Stock Reservation**

Reserves items in the user's cart during checkout, preventing others from purchasing the same products and ensuring a smooth shopping experience.

- **Restock Alerts**

Notifies users when out-of-stock products are replenished, increasing customer satisfaction and promoting repeat visits.

- **Saved Cart Functionality**

Allows users to save their cart for later, offering flexibility and convenience for customers who need more time to decide.

- **Dynamic Notifications**

Provides instant updates about stock changes or price modifications, enhancing transparency and building trust with customers.

## SUMMARY

This chapter describes about Medicare Shop is a console-based platform for buying medical supplies, using React.js and Firebase for real-time updates and security. It addresses issues like limited accessibility, outdated inventory, and complex interfaces. Key features include stock reservation, restock alerts, and saved carts. The platform ensures a user-friendly, secure, and efficient shopping experience.

## **CHAPTER 2**

### **SYSTEM ANALYSIS**

#### **2.1 IDENTIFICATION OF NEED**

The Medicare Shop project addresses key challenges in the current online medical product procurement process. Traditional platforms often provide inaccurate stock availability, leading to frustration when items are out of stock. Users also face issues with cart reservation, losing items in their cart if they are bought by someone else before checkout. Existing systems often lack real-time stock updates, causing confusion about product availability, and have inconsistent pricing information. Additionally, many platforms have complex user interfaces that make it difficult for non-technical users to navigate. The Medicare Shop project resolves these issues by offering real-time stock updates, cart reservation features, and dynamic notifications for stock and pricing changes. It also provides an intuitive, user-friendly interface, ensuring a smoother and more reliable online shopping experience for medical supplies.

#### **2.2 FEASIBILITY STUDY**

The feasibility study deals with all analysis that takes up in developing the project. Each structure has to be thought of in the developing of the project, as it has to serve the end user in friendly manner. One must know the type of information to be gathered and the system analysis consists of collecting, organizing and evaluating facts about a system and its environment.

The results of this analysis are used in making the decision whether to proceed with the project or not. This analytical tool used during the project planning phase shows how a institute would operate under a set of assumption, such as technology used, the facilities and equipment. The study is the first time in a project development process that show whether the project create a technical and economic feasible concept.

A feasible project is one where the project could generate adequate number of profits, withstand the risk it will encounter, remain viable in the long-term and meet the goals. The main objective of the system analysis is to study operation and to learn and accomplish the processing activities. The details are processed through coding themselves. It will be controlled by the program alone.

Three considerations involved in feasibility are

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

### **2.2.1 Technical Feasibility**

The development of Medicare Shop is technically feasible due to the use of advanced technologies like React.js for the frontend and Firebase for the backend. React.js provides a dynamic, responsive user interface, while Firebase offers a real-time database for stock and pricing updates, along with secure authentication. These technologies integrate seamlessly, ensuring efficient communication between the frontend and backend.

The platform requires minimal infrastructure, as Firebase is cloud-based, reducing setup costs and allowing for easy scalability. React.js and Firebase offer the tools needed for a streamlined development process, ensuring the platform is accessible across devices and operating systems. Firebase also provides strong security features to protect user data and transactions. Overall, Medicare Shop benefits from reliable, scalable, and secure technologies, making it technically feasible to develop an efficient and accessible platform for online medical product procurement.

### **2.2.2 Operational Feasibility**

Medicare Shop is operationally feasible due to its user-friendly design and efficient performance. The platform's console-based interface makes it accessible to users of all technical levels, ensuring easy adoption. Powered by React.js, it offers fast, responsive browsing, while Firebase provides real-time updates on stock and pricing, ensuring accurate information.

Security is guaranteed through Firebase's secure authentication and transaction protection. The platform is scalable, with the ability to handle growing user traffic and product catalogues. It can also integrate with existing healthcare systems, making it adaptable for institutions. The use of React.js and Firebase minimizes infrastructure needs and keeps operational costs low, while ensuring compliance with necessary regulations. In conclusion, Medicare Shop is a cost-effective, scalable, and secure solution that effectively meets user needs.

### **2.2.3 Economic Feasibility**

The economic feasibility of Medicare Shop is strong, with low initial investment and operational costs. By using React.js and Firebase, the platform eliminates the need for expensive infrastructure, as Firebase is cloud-based, reducing setup and maintenance expenses. The pay-as-you-go model of Firebase keeps operational costs scalable and affordable.

Medicare Shop is designed for scalability, meaning it can grow without proportional increases in costs. Revenue can be generated through direct product sales, subscription services, and partnerships with healthcare providers. Digital marketing strategies like SEO and social media promotion further reduce marketing costs and enhance profitability. With the growing demand for online medical supplies, Medicare Shop is well-positioned to generate consistent revenue and expand into new markets. Its low costs, scalability, and diverse revenue opportunities make it a financially sustainable platform.



## **2.3 SOFTWARE REQUIREMENT SPECIFICATION**

### **Functional Requirements**

System requirements describe the capabilities a system must have to meet user needs effectively and efficiently. In the case of Medicare Shop, the system needs to fulfill specific requirements for both hardware and software to ensure seamless operation and a user-friendly experience. These requirements are designed to improve product management and customer experience, while also addressing the drawbacks and limitations of existing systems.

### **System Feature 1 :**

#### **User Authentication and Access Management**

##### **1.User Login:**

Customers can log in using their correct username and password. If there is no match, a warning message is displayed. During account creation, if the chosen username already exists, the system alerts the user to choose a different one.

##### **2.Username Verification:**

After logging in, the system will check the username against the database and display the appropriate user interface based on the user credentials.

##### **3.User Logout:**

Customers can log out after completing their tasks.

### **System Feature 2 :**

#### **Product Management and Viewing**

##### **1.Product Display:**

The system will display a list of available products, including their defined features retrieved from the database.

**1. Product Viewing:**

Registered customers can view the homepage and browse through the product catalog. The product details will include real-time updates on stock availability and pricing.

**2. Real-Time Updates:**

Product stock and pricing will be updated in real-time across all user interfaces, ensuring that customers always see accurate data.

**3. Cart Management:**

- ◆ Items added to the cart will be held for a specified time, ensuring availability during checkout.
- ◆ If an item is out of stock, the system will notify the user and offer alternatives.
- ◆ Inventory will update in real-time as items are added or removed from the cart.

**System Feature 3:****Admin Panel and Data Management****1. Admin Authentication:**

Admins can securely log in to manage product data, ensuring that only authorized personnel can make changes to the product catalogue.

**2. Product Management:**

Admins can add, modify, and remove products, including stock levels, descriptions, and prices.

**3. Order and Customer Management:** Admins can view transaction history, manage orders, and handle customer information.**4. Notifications:** Admins will receive notifications about stock levels, out-of-stock products in the system.

#### **System Feature 4 :**

##### **Notifications and Alerts**

1. **Stock Alerts:** Customers will receive notifications if items in their cart are about to go out of stock or if they have been restocked.
2. **Price Change Alerts:** Customers will be notified when there are changes in product pricing or availability.
3. **Order Updates:** Customers will also receive notifications regarding order statuses.

#### **System Feature 5:**

##### **Security and Data Protection**

1. **Secure Login and Registration:** The system will use encryption for passwords and sensitive data, ensuring secure authentication for both customers and admins.
2. **Data Protection:** Users personal and payment information will be securely stored and processed to prevent unauthorized access.

#### **2.3.1 Hardware Requirements**

The hardware for the system is selected considering the factors such as CPU processing speed, memory access, peripheral channel access speed, printer speed; seek time & relational data of hard disk and communication speed etc. Below is the minimum hardware requirement of the project.

- **Processor:** Intel Core i5 (or equivalent)
- **RAM:** 4GB or higher
- **Hard Disk:** 320GB or higher (preferably SSD for better performance)
- **Monitor:** 15'' or larger VGA/HD monitor
- **Network:** Broadband internet connection for seamless data.

### 2.3.2 Software Requirements

The software for the project is selected considering the factors such as working front end environment, flexibility in the coding language, database knowledge of enhances in backend technology etc.

- **Operating System:** Windows 10 or Windows 11
- **Front End:** React.js , HTML5 , CSS3
- **Back End:** Firebase
- **Database:** Firebase
- **Tools:** Visual Studio, Firebase Console

## 2.4 SOFTWARE DESCRIPTION

### Front End: React.js

React.js is a powerful, widely-used JavaScript library developed by Facebook for building user interfaces, particularly for single-page applications (SPAs). React allows developers to build dynamic, efficient, and scalable web applications with a component-based architecture. This modular approach enables better code reuse, maintenance, and organization, making React an ideal choice for building modern web applications like Medicare Shop.

React.js operates by rendering components, which are the building blocks of a React application. These components allow for the creation of dynamic, responsive user interfaces that can efficiently handle updates without reloading the entire page, thanks to React's Virtual DOM. The Virtual DOM is a lightweight copy of the actual DOM and allows React to quickly compare and update only the parts of the user interface that have changed. This results in improved performance and a smoother user experience.

React declarative syntax enables developers to describe how the UI should look for any given state of the application. React then takes care of updating the UI when the underlying data changes. This makes it easier to manage complex user interfaces where data flows between various components and needs to be updated frequently. React also offers features such as hooks, which allow developers to manage state and lifecycle events in functional components, and JSX, a syntax extension that allows developers to write HTML-like code within JavaScript. JSX makes it easier to create and visualize the component structure.

With React, the front end of Medicare Shop can efficiently display and update product information, handle user inputs like adding items to the cart, and provide real-time updates about product availability and pricing. React flexibility also allows easy integration with other tools and libraries, making it suitable for building interactive and user-friendly web applications.

### Features of React.js in the Medicare Shop Project

- **Component-Based Architecture:** React's component-based structure allows developers to break the UI into reusable, self-contained pieces, making the codebase more organized and maintainable.
- **Real-Time UI Updates:** React's Virtual DOM ensures that the user interface is updated efficiently and quickly, providing a seamless experience when users interact with the platform.
- **State Management:** React's hooks like `useState` and `useEffect` enable easy handling of component states and side effects, such as fetching data from Firebase or updating the UI based on user interactions.
- **Declarative UI:** React's declarative approach makes it easier to build complex user interfaces by defining how the UI should look for different states, reducing the complexity of managing UI updates manually.
- **JSX (JavaScript XML):** React uses JSX, a syntax extension that allows

HTML structures to be written directly in JavaScript, making it easier to design and visualize the components' layout.

- **Efficient Rendering with Virtual DOM:** Reacts efficient rendering mechanism ensures that only the parts of the interface that need to be updated are re-rendered, optimizing performance.
- **Integration with APIs and Firebase:** React easily integrates with external APIs, including Firebase for real-time database synchronization, user authentication, and secure transactions.
- **Enhanced User Experience:** Reacts flexibility and performance enable the creation of highly interactive and responsive web pages, providing users with a smooth and engaging experience while browsing medical products or managing their orders.

### **Back End: Firebase**

Firebase is a comprehensive platform developed by Google that provides a variety of services and tools for building and managing modern web and mobile applications. Firebase is especially popular for its real-time database and cloud-based backend services, making it an ideal choice for applications like Medicare Shop that require efficient real-time data synchronization and secure user authentication.

Firebase eliminates the need for complex server-side infrastructure by offering backend-as-a-service (BaaS). With its robust set of tools, Firebase provides everything needed to manage user authentication, store and synchronize product data, handle transactions, and scale the application as needed.

Key features of Firebase include real-time data synchronization, user authentication, and cloud storage, which work seamlessly together to provide a highly scalable, secure, and fast back-end infrastructure for Medicare Shop. Firebase's intuitive dashboard also makes it easier for developers to manage data, monitor performance, and optimize the app. Firebase supports both NoSQL databases (Firestore) and Real-time databases, making it highly suitable for applications that

need to handle a high volume of data with real-time interactions. This is crucial for Medicare Shop where inventory, stock updates, and pricing must be dynamically reflected to the users.

### **Features of Firebase in the Medicare Shop Project**

- **Real-time Database :** Firebase's Firestore database provides real-time synchronization of data across all connected clients, ensuring that the product information, stock availability, and pricing updates are reflected instantly on the user interface.
- **User Authentication:** Firebase offers a secure authentication system, supporting multiple sign-in methods such as email/password, Google, Facebook, etc. This allows users to securely log in to their accounts, manage their personal data, and make purchases in a safe environment.
- **Cloud Storage:** Firebase's Cloud Storage allows for secure file storage and access. For Medicare Shop, it can be used to store product images and other assets, ensuring they are available for users without compromising security or performance.
- **Firebase Hosting:** Firebase Hosting provides fast and secure web hosting for your application. With its global content delivery network (CDN), the hosting ensures low- latency performance, making sure users across the world can access Medicare Shop quickly.
- **Security Rules:** Firebase allows the setting of security rules to ensure that only authorized users can access or modify sensitive data. This ensures that all transactions, user data, and product details are protected.

### **SUMMARY:**

This chapter presents the issues in online medical procurement by offering real-time stock updates, cart reservation, and dynamic notifications. It uses React.js for a responsive frontend and Firebase for secure, real-time backend management. The platform ensures scalability, security, and cost-efficiency.

## **CHAPTER 3**

### **SYSTEM DESIGN**

#### **3.1 MODULE DESCRIPTION**

A module description provides detailed information about the module and its supported components, which is accessible in different manners. In this application, it contains many sub- modules.

##### **ADMIN MODULE:**

The Admin module is managed by the system administrator, who holds full control over the platform. The admin can:

- Add, modify, or delete products listed on the website.
- View user details, including personal and order information.
- Set the price, description, and images for products.
- Assign unique product IDs to avoid duplicate entries.
- Ensure the availability and update of products as needed.

##### **USER MODULE:**

The User module allows new customers to register and existing customers to log in. Key functions include:

- User registration, where users provide their username, password, email, phone number, and address.
- Authentication of login credentials to provide secure access to the system.
- Ability to browse the available medical products and add them to their cart.
- View product details, including price and description, as provided by the admin.



## **ORDER MODULE**

This module is where users can place their orders for the products they want to purchase. It includes:

- The option for users to add items to their cart and proceed with checkout.
- Users can review their selected products before finalizing the order.
- This module ensures that the products selected by the user are accurately processed for purchase.

## **PAYMENT MODULE**

The Payment module allows customers to complete their purchase through an online payment gateway. It includes:

- A secure payment processing system, where users can select their preferred payment method (credit card, debit card, etc.).
- After the payment is processed, the order is confirmed, and the amount is debited.

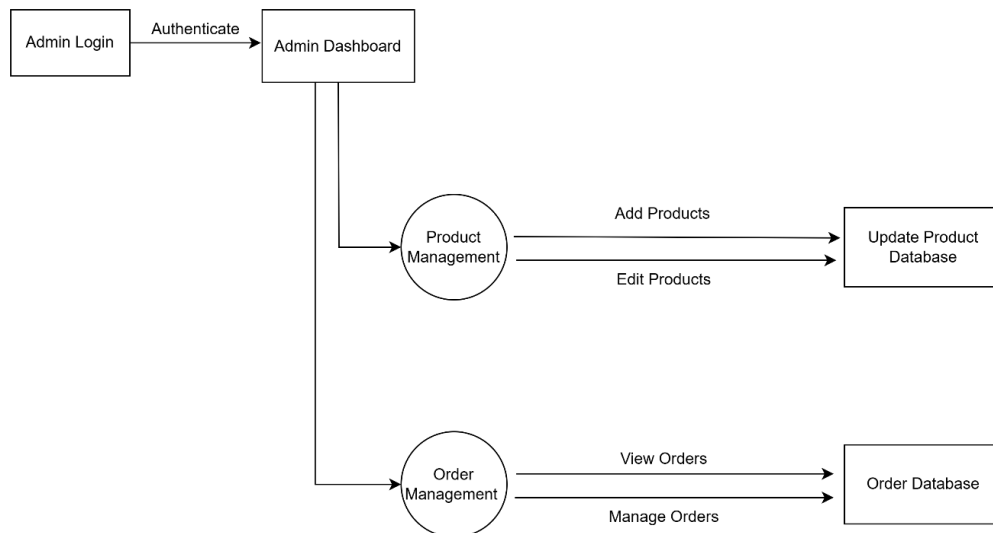
## **3.2 DATAFLOW DIAGRAM**

A Data Flow Diagram (DFD) visually represents how data flows within a system, highlighting inputs, outputs, and processes. It uses symbols to map interactions between entities, processes, and data stores, providing clarity on data transformations. Each process shows how data is transformed and its sources and destinations. The DFD helps manage system complexity by breaking down processes into simpler steps, ensuring traceability and clarity in data flow. This structured visualization aids in debugging, testing, and validating the system, making it easier to understand and optimize.

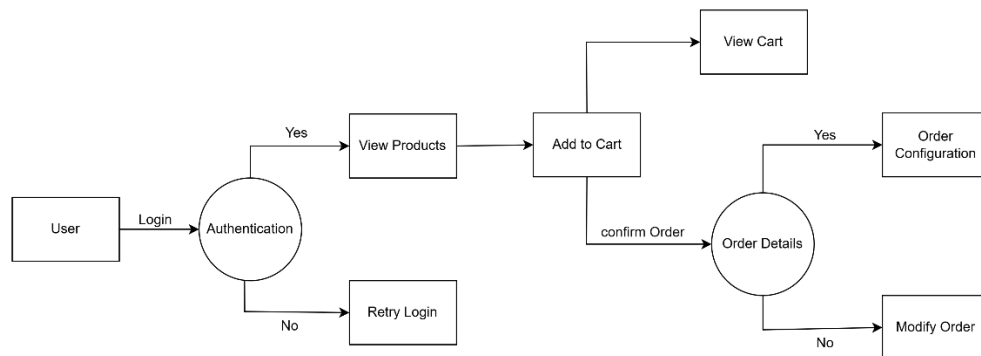
**LEVEL 0****Figure3.1 0th Level DFD diagram**

Admin can view the products in the web page and also manage the orders.

User can view the products and they can place their orders.

**LEVEL 1****Figure3.2 Admin side 1st Level DFD diagram**

Admin can login and maintain the items to be displayed, and add the product or edit the product and checks for the orders placed by the user.



**Figure3.3 User side 1st Level DFD diagram**

User can login and view the items and user can add the products into the cart and if the user wants to buy the products and the order can be confirmed.

- **User Authentication:** The user logs in, and credentials are verified. If unsuccessful, the system allows retrying.
- **Viewing Products:** Once logged in, users browse available products.
- **Adding to Cart:** Users add items to their cart for purchase.
- **Order Details Management:** Users can view and modify their cart before placing an order.
- **Cart Management:** Includes features like reviewing the cart, configuring the order, and making necessary modifications.

### 3.3 DATABASE DESIGN:

Database design is an essential aspect of the system, ensuring proper data management, security, and accessibility.

#### Database Overview

The Medicare Shop database serves as the backbone for storing and managing critical information required for the web application's operations.

This includes user data, product information, and order history. The database structure is meticulously designed to handle the complexities of a retail system, supporting high performance and scalability. It ensures that all data whether it's customer information, inventory details, or transaction records is efficiently stored and accessible when needed.

## **Data Integration**

Data integration is a key feature of the Medicare Shop database, allowing all critical information to be centrally stored and easily accessible through the database system. This centralized approach is vital for maintaining consistency across the system, reducing data redundancy, and ensuring that updates to records are accurate and propagated throughout the application.

For example, when a product's stock level is adjusted, the change must be reflected immediately in the product catalog, in user orders, and in the cart system.

## **Data Independence**

The Medicare Shop system is designed with data independence in mind, allowing the physical data structure (such as the layout of tables, the arrangement of columns, and the indexing of data) to be modified without impacting the application's functionality.

This is particularly important for maintaining the longevity and flexibility of the platform as new features and updates are introduced. For example, if the system needs to add a new field for product specifications or adjust the relationship between user tables to accommodate new roles, these changes can be made at the database level without rewriting existing code.

## CLASS BASED DIAGRAM

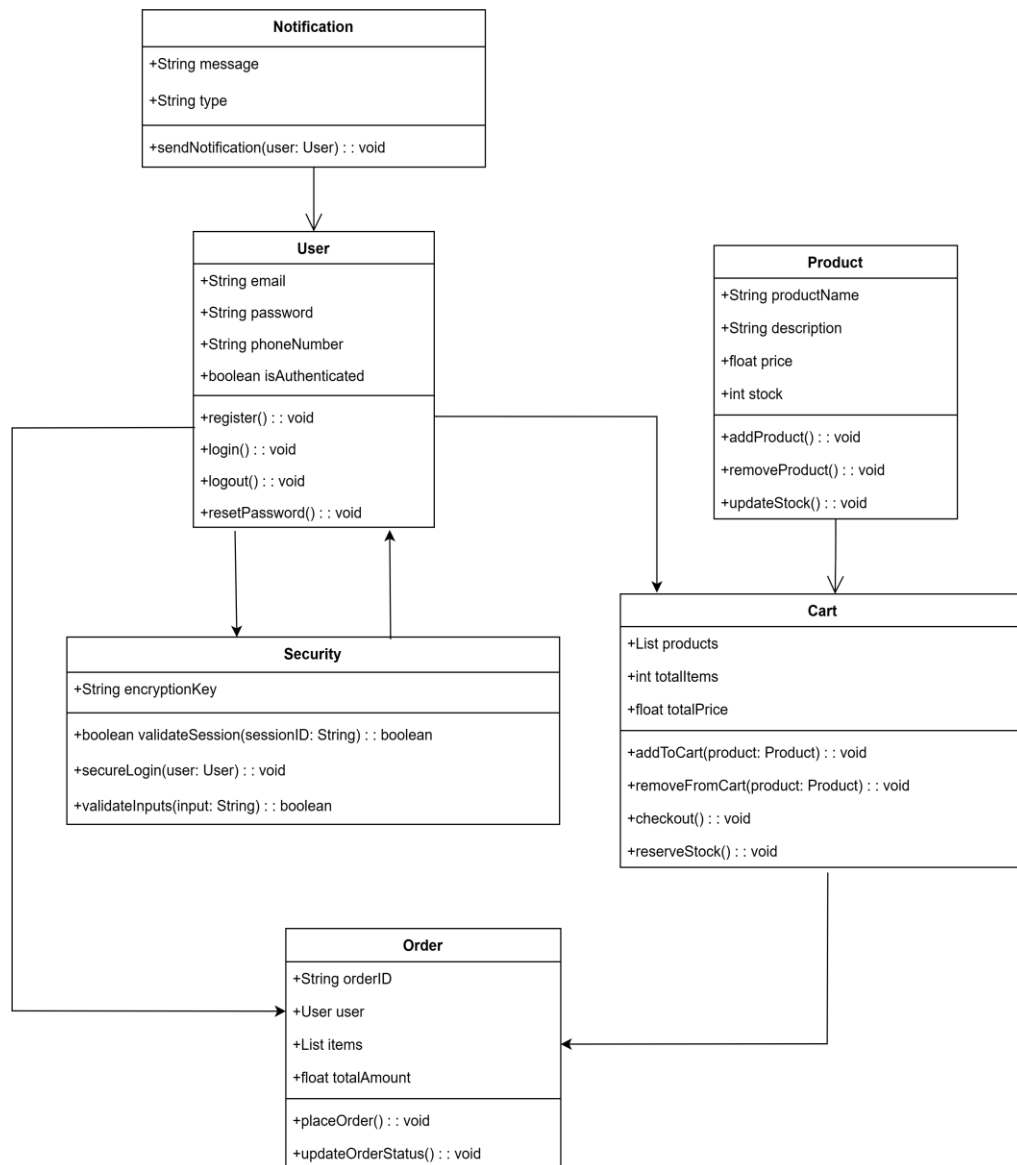


Figure 3.4 Class Based Diagram

### 3.4 TABLE DESIGN

FIELD NAME	DATA TYPE	SIZE	CONSTRAINTS
User_name	VARCHAR	15	PRIMARY KEY
email	VARCHAR	15	UNIQUE
password	VARCHAR	8	NOT NULL
Mobile_no	VARCHAR	10	NOT NULL

**Table 3.5 user details**

In the above table 3.5 user details, where the customer can register their details like user\_name, email, mobile number and their password.

FIELD NAME	DATA TYPE	SIZE	CONSTRAINTS
EMAIL	VARCHAR	15	NOT NULL
PASSWORD	VARCHAR	8	NOT NULL

**Table 3.6 admin details**

In the above table 3.6 user details, where admin can register their details like product\_name, email, mobile number and their password.

FIELD NAME	DATA TYPE	SIZE	CONSTRAINTS
Product_name	varchar	20	NOT NULL
Age_category	int	2	NOT NULL
price	int	100	NOT NULL
Stock_qty	int	100	NOT NULL
Image_url	varchar	100	NOT NULL
description	varchar	200	NOT NULL

**Table 3.7 shop\_product**

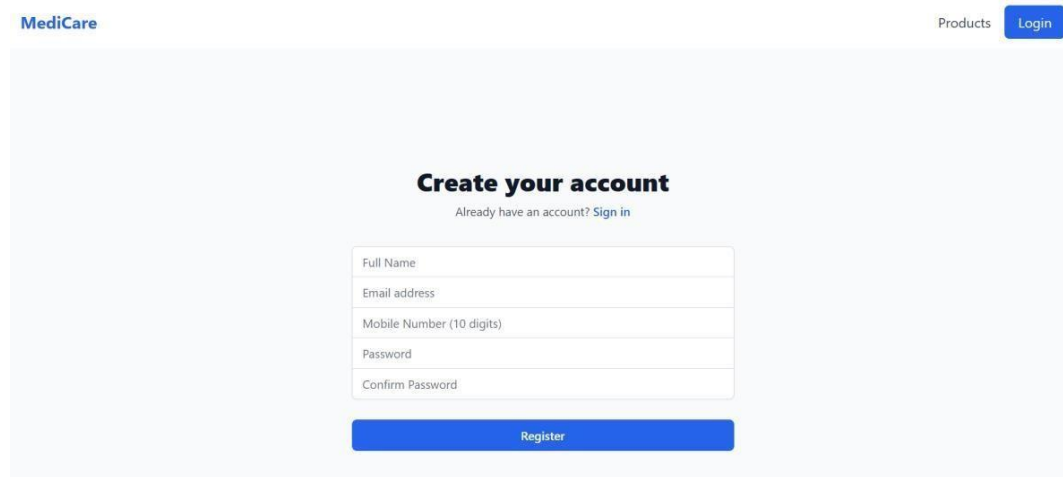
In the above table 3.7 product details are stored like product name, which category it belongs to price, etc.

### 3.5 INPUT DESIGN

Input design is the process of converting user-originated inputs to a computer understandable format. Input design is one of the most expensive phases of the operation of computerized system and is often the major problem of a system. A large number of problems with a system can usually be tracked back to fault input design and method.

Every moment of input design should be analysed and designed with utmost care. The decisions made during the input design are the project gives the low time consumption to make sensitive application made simple. Thus, the developed system is well within the budget. This was achieved because most of the technologies used are freely available.

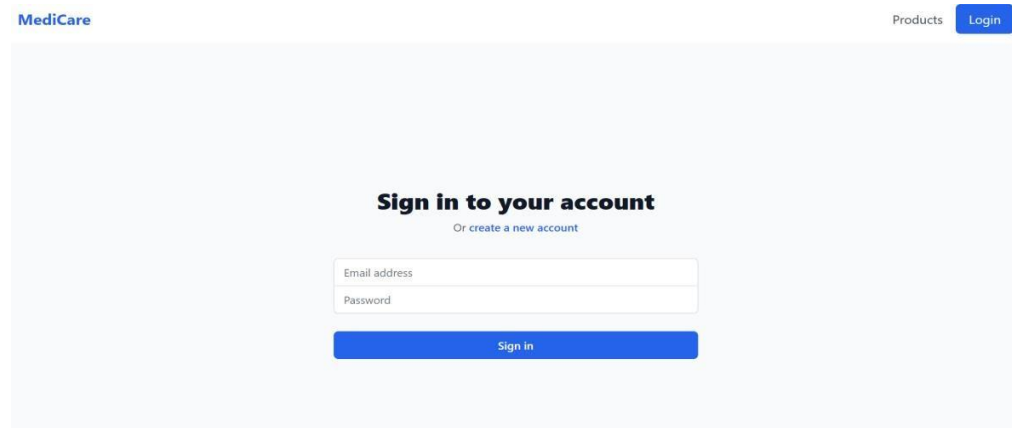
#### User Registration Page



**Figure 3.8 User Registration Page**

The figure 3.5.1 represents a user registration page.. The password must contain a minimum of 6 characters or digits. I Name, Email Address, Mobile Number, Password, and Confirm Password.

## User Login Page

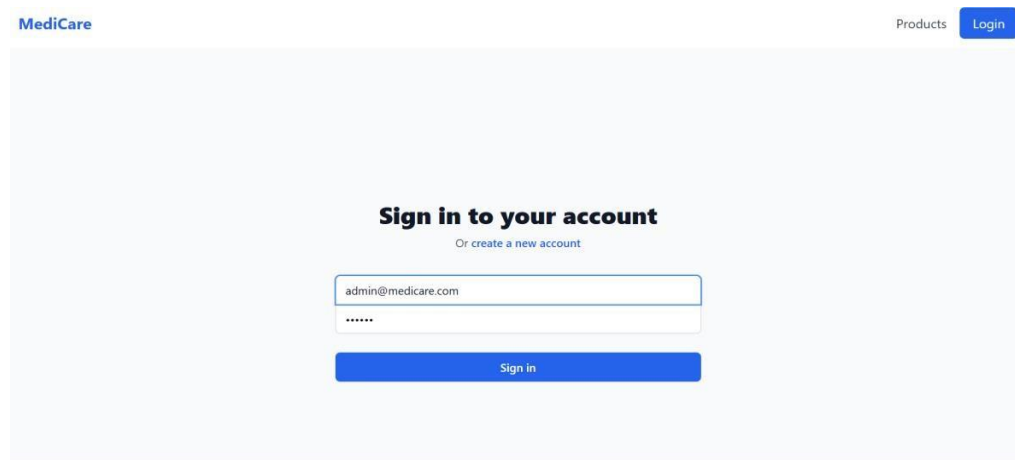


The image shows the user login page for the MediCare platform. At the top left is the "MediCare" logo, and at the top right are the links "Products" and "Login". The main heading is "Sign in to your account", with a link "Or create a new account" below it. There are two input fields: "Email address" and "Password". Below these fields is a blue "Sign in" button.

**Figure 3.9 User Login Page**

The figure 3.5.2 shows the user login process for the MediCare platform. Users with an existing account can enter their email and password, then click "Sign in" to access their profile and use the platform's features.

## Admin Login Page



The image shows the admin login page for the MediCare platform. At the top left is the "MediCare" logo, and at the top right are the links "Products" and "Login". The main heading is "Sign in to your account", with a link "Or create a new account" below it. There are two input fields: "Email address" (containing "admin@medicare.com") and "Password" (containing "\*\*\*\*\*"). Below these fields is a blue "Sign in" button.

**Figure 3.10 Admin Login Page**

Figure 3.5.3 shows the admin login page for the MediCare platform, allowing admins to manage accounts, configure settings, and access analytics tools.



### 3.6 OUTPUT DESIGN

Output design generally refers to the results and information that are generated by the system for many end-users; it should be understandable with the enhanced format. Computer output is most important direct source of information to the user. Output design deals with form design. Efficient output design should improve the interfacing with user.

The screenshot displays the 'Admin Dashboard' page of the 'MediCare' application. At the top, there is a navigation bar with links for 'Products', 'Admin', 'Admin Dashboard', and 'Logout'. The main content area is titled 'Admin Dashboard' and contains a form for 'Add New Product'. The form includes the following fields:

- Product Name:** A text input field containing 'Corex dx'.
- Price (₹):** A text input field containing '150'.
- Stock Quantity:** A text input field containing '50'.
- Age Category:** A dropdown menu with '12+ years' selected.
- Image URL:** A text input field containing a base64 encoded image URL.
- Description:** A text area containing the text: 'Corex DX Syrup is a combination of two medicines used in the treatment of dry cough. It can provide temporary relief of cough due to throat irritation'.

Below the form, there is an 'Image Preview' section showing a small image of a bottle of Corex DX Syrup. At the bottom of the form, there is a blue button labeled 'Add Product'.

**Figure 3.11 Admin Dashboard Page**

In figure 3.6.1 is the home page of admin. Whatever the activity performed in food ordering application will be stored in this page. Each user login, products, order details will be managed by the admin.

### SUMMARY

This chapter includes Admin, User, Order, and Payment modules, with Data Flow Diagrams illustrating data interactions. The database and interface designs ensure efficient data management and user-friendly operations.

## **CHAPTER 4**

### **IMPLEMENTATION**

#### **4.1SYSTEM IMPLEMENTATION**

Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and giving confidence in the new system for the users, that it will work efficiently and effectively, It involves careful planning, investing of the current system, and its constraints on implementation, design of methods to achieve the changeover methods The implementation process begins with preparing a plan for the implementation of the system.

The coding step translates a detailed design representation into a programming language realization. Programming languages are vehicles for communication between humans and computers programming language characteristics and coding styles can profoundly affect software quality and maintainability.

The coding is done with the following characteristics in

- 4.1.1 Ease of design to code translation.
- 4.1.2 Code efficiency
- 4.1.3 Memory efficiency & Maintainability

The user should be very careful while implementing a project to ensure what they have planned is properly implemented. The user should not change the purpose of project while implementing. The user should not go in a roundabout way to achieve a solution; it should be direct, crisp and clear and up to the point.

## **4.2 STANDARDIZATION OF THE CODING**

React.js emphasizes clean and consistent coding practices, ensuring clarity and maintainability. Proper indentation defines control structures, improving code readability. Consistent naming conventions for variables standardize the codebase. Thoughtful data descriptions enhance understanding, while comments provide valuable insights for developers. This approach helps developers quickly grasp the purpose of different code sections. It prioritizes readability and organization, enabling efficient development. Overall, this promotes seamless collaboration while maintaining core functionality.

## **4.3 ERROR HANDLING**

Error and exception handling is crucial for application reliability. Errors cause crashes and require monitoring, bug tracking, and developer intervention. Exceptions are manageable with mechanisms like try-catch blocks, allowing graceful recovery. However, unhandled exceptions can disrupt normal operations. Implementing robust error and exception handling ensures smooth execution and minimal disruption. It improves the application's resilience, allowing it to recover from issues. Proactive strategies help maintain a seamless user experience.

## **SUMMARY**

This chapter involves turning the design into a working system, with careful planning and coding for efficiency and maintainability. React.js emphasizes clean, consistent coding practices for clarity and collaboration. Error and exception handling are crucial for reliability, ensuring smooth execution and resilience. Robust strategies prevent disruptions, allowing applications to recover gracefully.

## **CHAPTER 5**

### **TESTING AND RESULTS**

#### **5.1 TESTING**

Software testing is a crucial phase in software quality assurance, where specifications, designs, and coding are assessed. Test data is prepared and used to test the system thoroughly, targeting potential errors. After the source code is completed, relevant data structures are documented for the testing process. The system undergoes testing and validation to ensure it meets the required standards. Developers are responsible for testing individual modules, ensuring each unit functions correctly. They also perform integration testing, combining modules into a complete system. This ensures the system works as intended and is free of errors.

This project has undergone the following testing procedures to ensure its correctness

- Unit testing
- Integration Testing
- Validation Testing

##### **5.1.1 UNIT TESTING**

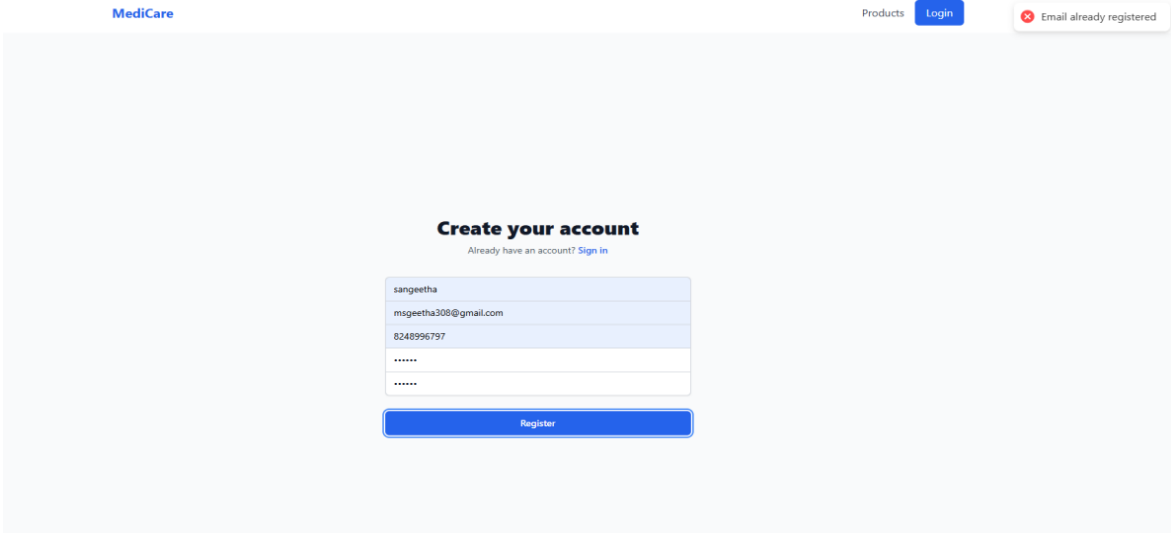
In unit testing, each unit of the system is tested, and its functionality is verified and ensures that it functions as expected. Unit testing is performed by the web developer while developing the application. The software units in a system are the modules and routines that are assembled and integrated to perform a specific function, Unit testing is first done on the modules independently of one another, to locate errors.

### 5.1.2 INTEGRATION TESTING

Integration testing is done to test itself if the individual modules work together as one single unit. In integration testing, the individual modules that are to be integrated are available for testing. Thus, the manual test data that used to test the interfaces replaced by that which in generated automatically from the various modules. It can be used for testing how the module would actually interact with the proposed system. The modules are integrated and tested to reveal the problem interfaces.

### 5.1.3 VALIDATION TESTING

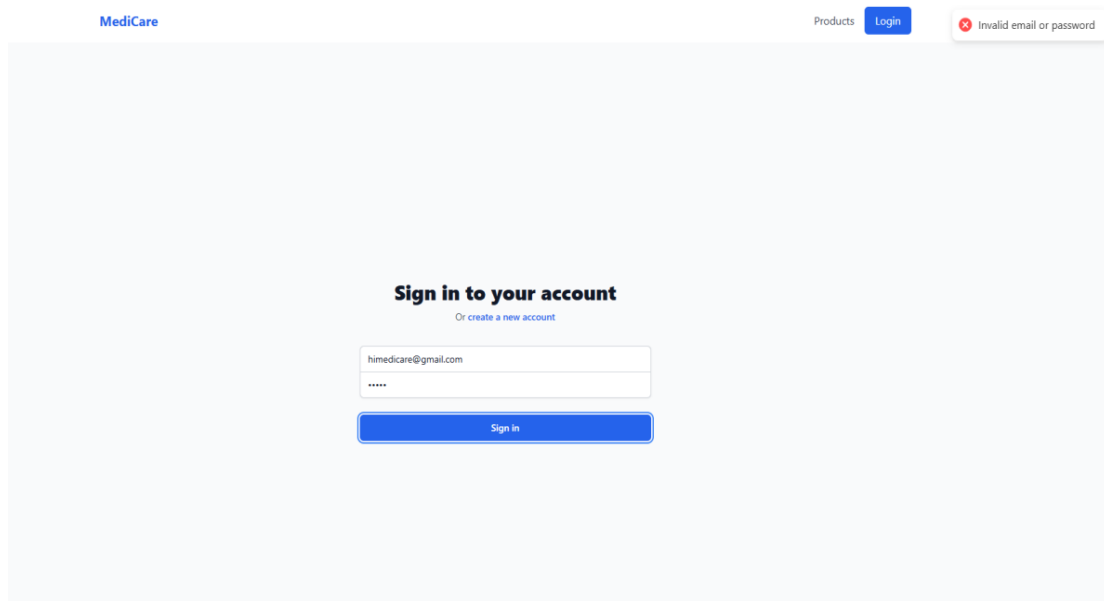
Validation testing can be defined in many ways, but a simple definition is that can be reasonably expected by the customer. After the validation test has been conducted. one of two possible conditions exists. The functions or performance characteristics confirm to specification and are accepted. A deviation from the specification is uncovered and a deficiency list is created.



The screenshot shows the MediCare registration page. At the top left is the 'MediCare' logo. At the top right are links for 'Products' and 'Login', and a red error message 'Email already registered'. The main heading is 'Create your account' with a link 'Already have an account? Sign in'. Below this is a registration form with fields for 'sangeetha' (username), 'msggeetha308@gmail.com' (email), and '8248996797' (password). A blue 'Register' button is at the bottom of the form.

**Figure 5.1 Register Page with user already exists**

In Figure 5.1, the sign-up page for MediCare shows an error message if the user's email or password already exists in the database.



**Figure 5.2 Login Page with email or password**

In Figure 5.2, when a user attempts to log in to the MediCare platform with incorrect either the username or password—an error message is displayed. This message reads, "Invalid Username/Password! Please try again."

## SUMMARY

This chapter outlines the correctness and quality of a system. It includes unit testing, integration testing, and validation testing. Unit testing checks individual modules for functionality, while integration testing ensures that modules work together as a unified system. Validation testing verifies that the system meets customer expectations and specifications. During testing, error messages are displayed if issues like existing users or incorrect login credentials occur. This process helps ensure the system functions as intended and meets all requirements.

## CHAPTER 6

### CONCLUSION AND FUTURE ENHANCEMENT

#### 6.1 CONCLUSION

The project **Medical Retail Shop** represents a significant advancement in the healthcare industry, blending cutting-edge web technologies with the critical need for efficient medical supply procurement. By leveraging React.js and Firebase, the platform not only simplifies the shopping experience but also enhances security, ensuring that users can access real-time updates on product availability and pricing with ease. The console-based design makes it accessible to a wide range of users, regardless of their technical expertise. Medical Retail Shop not only addresses the inefficiencies of traditional medical supply shopping but also sets a precedent for future innovations in healthcare technology, ultimately improving the quality of care and accessibility for all users.

#### 6.2 FUTURE ENHANCEMENT

In envisioning the further enhance Medicare Shop, several advancements can be made to improve both the user experience and the system's functionality. Future enhancements could include integrating advanced multi-factor authentication to strengthen security during user login processes. Introducing personalized product recommendations powered by AI and machine learning would make it easier for users to discover relevant medical supplies. For administrators, more sophisticated analytics tools and real-time collaboration features would enable better inventory management and marketing strategies.

An enhanced notification system, tailored to specific product categories, and user education features, such as instructional videos and usage guides, would further improve the user experience by providing critical information directly on the platform. These enhancements would not only address existing limitations but also position Medicare Shop as a leader in the healthcare e-commerce space, offering a more personalized, secure, and efficient shopping experience.

## APPENDICES

### A. SAMPLE CODING

#### LOGIN PAGE

```
import { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { useAuthStore } from '../store/authStore';
import toast from 'react-hot-toast';
export default function Login() {
  const [credentials, setCredentials] = useState({ email: "", password: "" }); const login =
  useAuthStore((state) => state.login);
  const navigate = useNavigate();
  const handleSubmit = (e) => {
    e.preventDefault();
    const _result=
    login(credentials);if
    (result.success) {
      toast.success(`Welcome back, ${result.user.name}!`); navigate('/');
    } else {
      toast.error(result.message || 'Invalid email or password');
    }
  };
  return (
    <div className="min-h-screen flex items-center justify-center bg-gray-50 py-12 px-4
sm:px-6 lg:px-8">
      <div className="max-w-md w-full space-y-8">
        <div>
          <h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900">
            Sign in to youraccount</h2>
```



```

<p className="mt-2 text-center text-sm text-gray-600"> Or{ ' ' }
  <Linkto="/register" className="font-medium text-blue-600 hover:text-blue- 500">

    create a new account

  </Link>
</p>
</div>
<form className="mt-8 space-y-6" onSubmit={handleSubmit}>
  <div className="rounded-md shadow-sm -space-y-px">
    <div>
      <input type="email"
        required
        className="appearance-none rounded-none relative block w-full px-3 py- 2
border border-gray-300 placeholder-gray-500 text-gray-900 rounded-t-md focus:outline-
none focus:ring-blue-500 focus:border-blue-500 focus:z-10 sm:text- sm"

        placeholder="Email   address"
        value={credentials.email}

        onChange={(e) => setCredentials({ ...credentials, email: e.target.value })}/>
    </div>
    <div>
      <input
        type="password"
        required
        className="appearance-none rounded-none relative block w-full px-3 py- 2
border border-gray-300 placeholder-gray-500 text-gray-900 rounded-b-md focus:outline-
none focus:ring-blue-500 focus:border-blue-500 focus:z-10 sm:text- sm"

        sword}
        onChange={(e) => setCredentials({ ...credentials, password: e.target.value

place
holde
r="Pa
sswor
d"
value
={cre
dientia
ls.pas

```

```

    />
  </div>
</div>
<div>
  <button
    type="submit"
    className="group relative w-full flex justify-center py-2 px-4 border border-transparent
text-sm font-medium rounded-md text-white bg-blue-600 hover:bg-
blue-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500"
  >
    Sign in
  </button>
</div>
</form>
</div>
</div>
);
}

```

## ADMIN PAGE

```

import AdminProductForm from '../components/AdminProductForm';
import StockManagement from '../components/StockManagement';
export default function Admin() {
  return (
    <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-8">
      <h2 className="text-2xl font-bold mb-6">Admin Dashboard</h2>
      <AdminProductForm />
      <StockManagement />
    </div>);}

```

**REGISTRATION PAGE**

```

import { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { useAuthStore } from '../store/authStore';
import toast from 'react-hot-toast';

export default function Register() {
  const [formData, setFormData] = useState({
    email: "",
    password: "",
    confirmPassword: "",
    name: "",
    mobile: ""
  });

  const registerUser = useAuthStore((state) => state.registerUser);
  const navigate = useNavigate();

  const handleSubmit = (e) => {
    e.preventDefault();

    if (formData.password !== formData.confirmPassword) {
      toast.error('Passwords do not match');
      return;
    }

    if (!formData.mobile.match(/^[6-9]\d{9}$/)) {
      toast.error('Please enter a valid 10-digit Indian mobile number');
      return;
    }

    const result = registerUser({
      email: formData.email,
      password: formData.password,
      name: formData.name,
      mobile: formData.mobile
    });
  }

```

```

if (result.success) {
  toast.success('Registration successful! Please login.');
```

    navigate('/login');

```

}
else { toast.error(result.message);
}
};

const handleMobileChange = (e) => {
  const value = e.target.value.replace(/\D/g, '').slice(0, 10);
  setFormData({ ...formData, mobile: value });
};

return (
  <div className="min-h-screen flex items-center justify-center bg-gray-50 py-12 px-4
sm:px-6 lg:px-8">

    <div className="max-w-md w-full space-y-8">
      <div>
        <h2 className="mt-6 text-center text-3xl font-extrabold text-gray-900">
          Create your account
        </h2>
        <p className="mt-2 text-center text-sm text-gray-600">
          Already have an account?{' '}
          <Link to="/login" className="font-medium text-blue-600 hover:text-blue-
500">
            Sign in
          </Link>
        </p>
      </div>
      <form className="mt-8 space-y-6" onSubmit={handleSubmit}>
        <div className="rounded-md shadow-sm -space-y-px">
          <div>

```

```

        className="appearance-none rounded-none relative block w-full px-3 py-2
border border-gray-300 placeholder-gray-500 text-gray-900 rounded-t-md focus:outline-
none focus:ring-blue-500 focus:border-blue-500 focus:z-10 sm:text-sm"

```

```

        placeholder="Full Name"

```

```

        value={formData.name}

```

```

        onChange={(e) =>

```

```

        setFormData({

```

```

        ...formData,      name:

```

```

        e.target.value })}

```

```

    />

```

```

</div>

```

```

<div>

```

```

    <input

```

```

        type="email"

```

```

        required

```

```

        className="appearance-none rounded-none relative block w-full px-3 py-2
border border-gray-300 placeholder-gray-500 text-gray-900 focus:outline-none
focus:ring-blue-500 focus:border-blue-500 focus:z-10 sm:text-sm"

```

```

        placeholder="Email address"

```

```

        value={formData.email}

```

```

        onChange={(e) => setFormData({ ...formData, email: e.target.value })}

```

```

    />

```

```

</div>

```

```

<div>

```

```

    <input

```

```

        type="tel"

```

```

        required

```

```

        pattern="[6-9][0-9]{9}"

```

```

        className="appearance-none rounded-none relative block w-full px-3 py-2
border border-gray-300 placeholder-gray-500 text-gray-900 focus:outline-none
focus:ring-blue-500 focus:border-blue-500 focus:z-10 sm:text-sm"

```

```

        placeholder="Mobile Number (10 digits)"

```

```

        value={formData.mobile}

```

```

        onChange={handleMobileChange}

```

```

        maxLength={10}

```

```

    />

```

```

</div>

```

```

<div>

```

```

    <input

```

```

        type="password"

```

```

        required

```

```

        className="appearance-none rounded-none relative block w-full px-3 py-2
border border-gray-300 placeholder-gray-500 text-gray-900 focus:outline-none
focus:ring-blue-500 focus:border-blue-500 focus:z-10 sm:text-sm"

```

```

        placeholder="Password"

```

```

        value={formData.password}

```

```

        onChange={(e) => setFormData({ ...formData, password: e.target.value })}

```

```

        minLength={6}

```

```

    />

```

```

</div>

```

```

<div>

```

```

    <input

```

```

        type="password"

```

```

        required

```

```

        className="appearance-none rounded-none relative block w-full px-3 py-2
border border-gray-300 placeholder-gray-500 text-gray-900 rounded-b-md
focus:outline-none focus:ring-blue-500 focus:border-blue-500 focus:z-10 sm:text-sm"

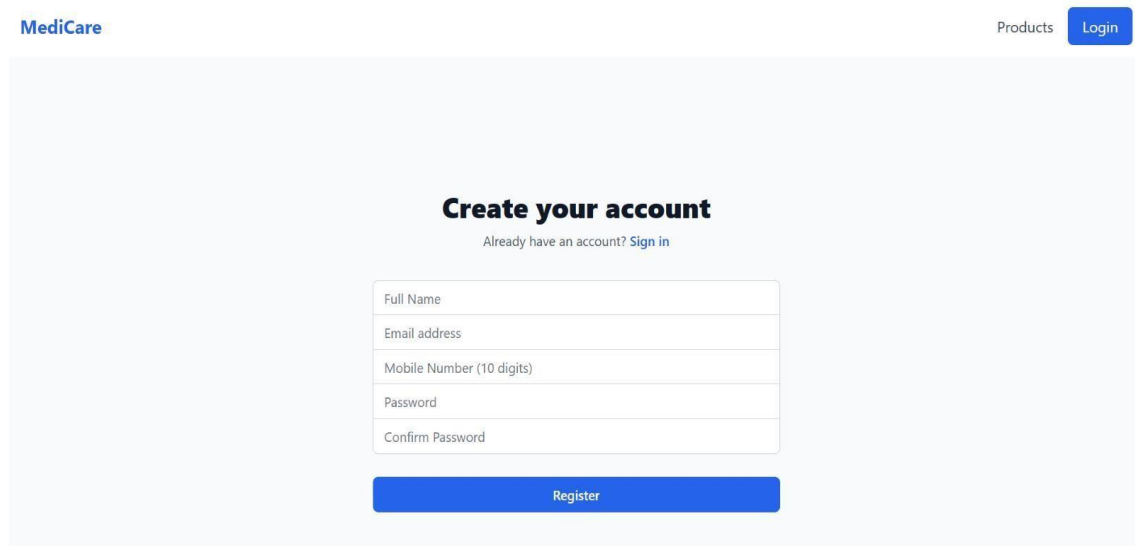
        placeholder="Confirm Password"
        value={formData.confirmPassword}
        onChange={(e) => setFormData({ ...formData, confirmPassword:
e.target.value })}

        minLength={6}
    />
</div>
</div>
<div>
    <button
        type="submit"
        className="group relative w-full flex justify-center py-2 px-4 border border-
transparent text-sm font-medium rounded-md text-white bg-blue-600 hover:bg-blue-700
focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-blue-500"

        >
        Register
    </button>
</div>
</form>
</div>
</div>
);
}

```

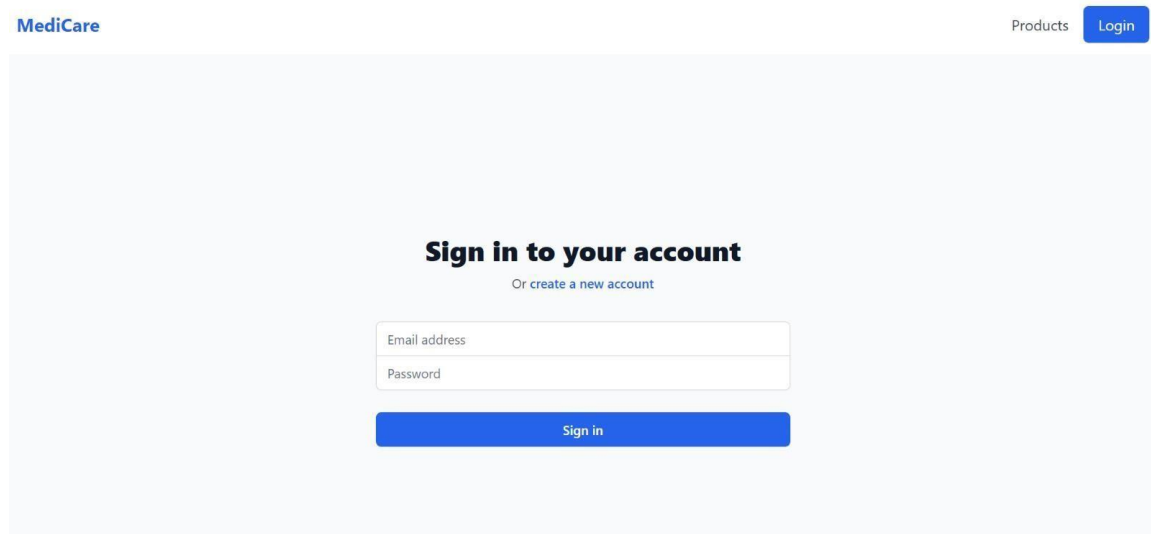
## B. SCREENSHOTS



The screenshot shows the Medicare website's sign-up page. At the top left is the 'MediCare' logo, and at the top right are links for 'Products' and a blue 'Login' button. The main heading is 'Create your account' in bold, with a link 'Already have an account? Sign in' below it. The registration form consists of five input fields: 'Full Name', 'Email address', 'Mobile Number (10 digits)', 'Password', and 'Confirm Password'. Below these fields is a large blue 'Register' button.

**Figure B.1 Sign-Up Page**

In Figure B.1 When a new customer comes first time to the website, they have to do registration first. They have to fill all the details and click on register button.

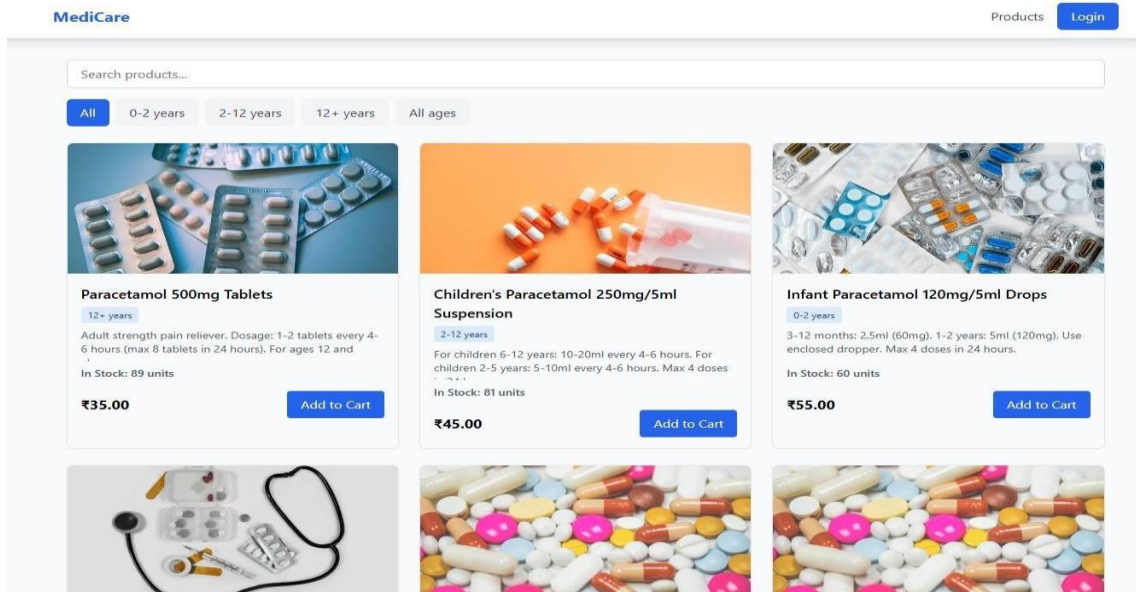


The screenshot shows the Medicare website's login page. At the top left is the 'MediCare' logo, and at the top right are links for 'Products' and a blue 'Login' button. The main heading is 'Sign in to your account' in bold, with a link 'Or create a new account' below it. The login form consists of two input fields: 'Email address' and 'Password'. Below these fields is a large blue 'Sign in' button.

**Figure B.2 Login Page**

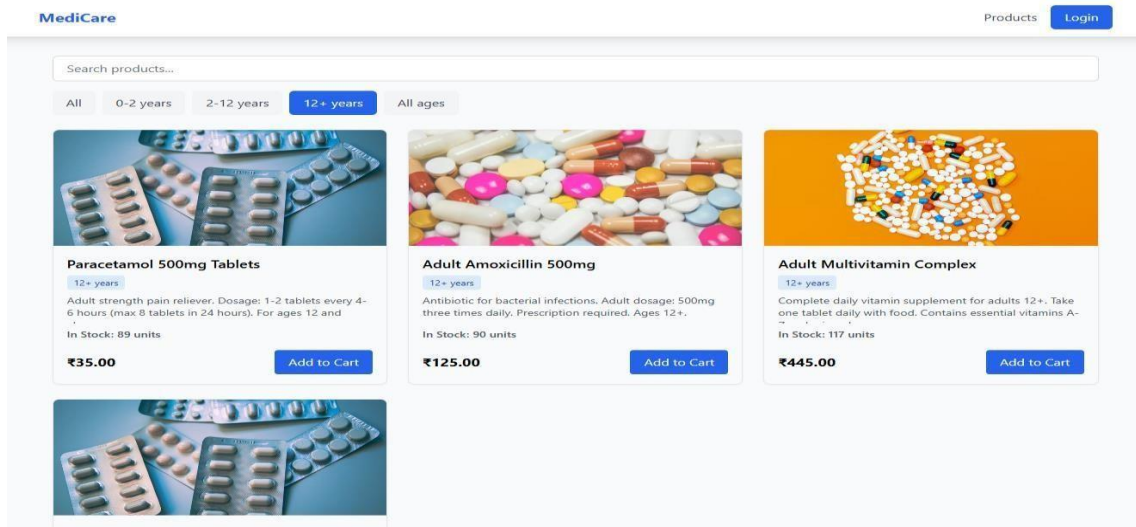
In Figure B.2, the customer logs in with their email and password to checkout and order products.





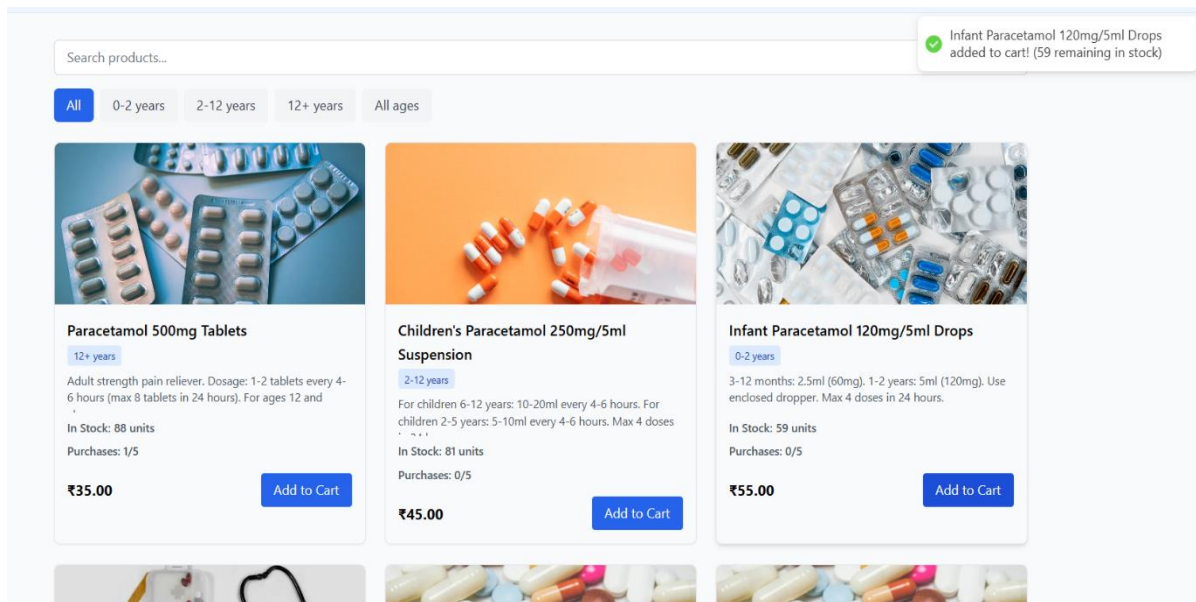
**Figure B.3 Home Page**

In Figure B.3, after logging in, users are directed to the MediCare home page, where they can access features and manage their session.



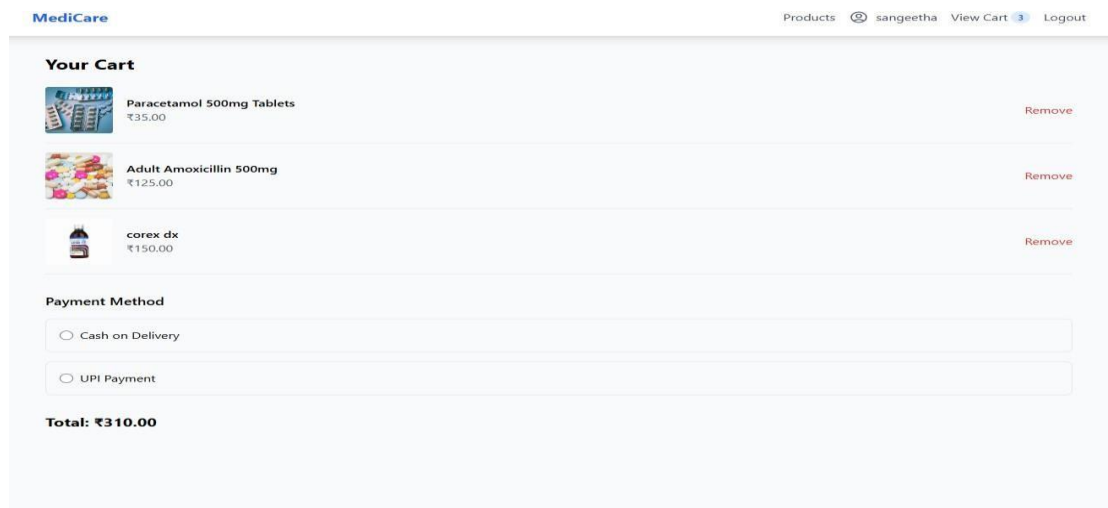
**Figure B.4 Product Category**

In Figure B.5, displays product categories, allowing users to view and order products from their chosen category.



**Figure B.5 Add to Cart screen**

In Figure B.5 represents the product added to cart screen. In this page user can see their Orders.



**Figure B.6 View Order Page**


In Figure B.6 represents the view order page. In this page user can see his present and past orders.

☐ Cash on Delivery

☒ UPI Payment

**Total: ₹55.00**

Scan QR Code to Pay ₹55.00



Open your Google Pay app and scan this QR code


I have completed the payment

Choose another payment method

**Figure B.7 Payment Page**

In Figure B.7 represents the payment page. In this page user can use QR scanner to pay their amount.

MediCare

Products  sangeetha View Cart 

Payment verified successfully!

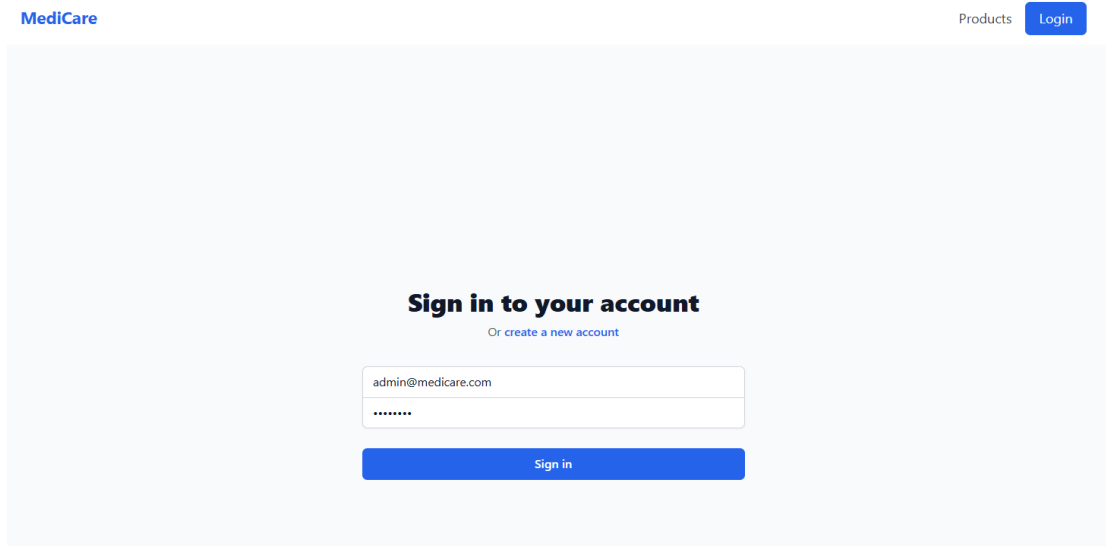
**Welcome to  
MediCare**

Your trusted online pharmacy for all your medical needs.

[Browse Products](#)

**Figure B.8 Payment verification Page**

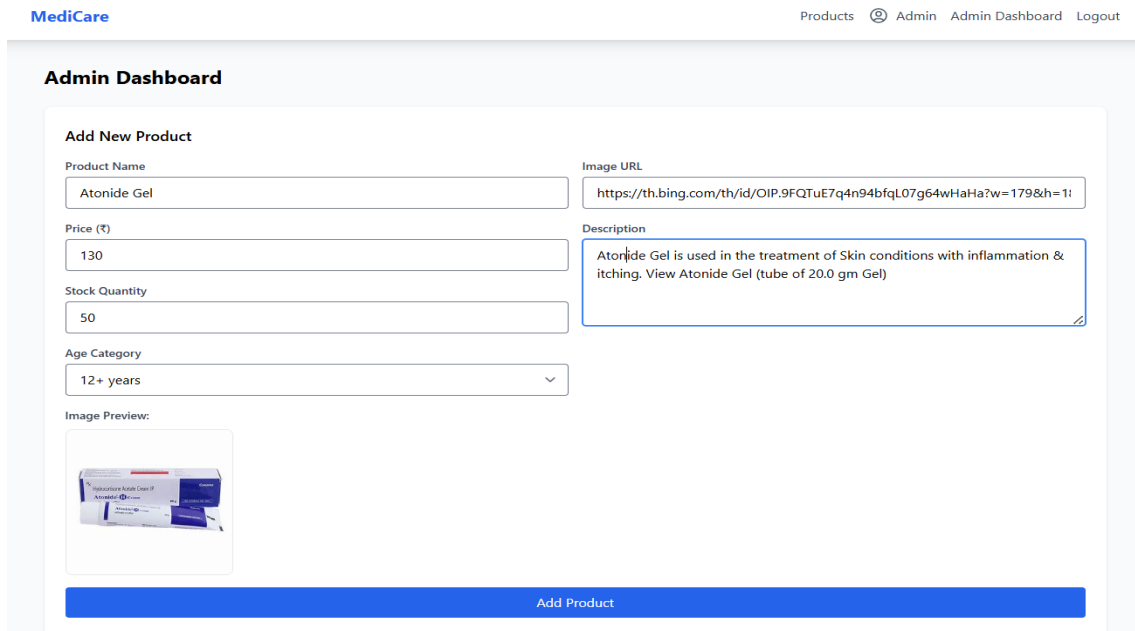
In Figure B.8 represents the payment verification page. In this page user can able to know the verification of payment.



The image shows the 'Sign in to your account' page for MediCare. At the top left is the 'MediCare' logo. At the top right are links for 'Products' and a 'Login' button. The main heading is 'Sign in to your account' with a sub-link 'Or create a new account'. Below this is a form with two input fields: the first contains 'admin@medicare.com' and the second contains a masked password '\*\*\*\*\*'. A blue 'Sign in' button is positioned below the password field.

**Figure B.9 Admin Login Page**

In Figure B.2, the admin logs in with their email and password to checkout and order products.



The image shows the 'Admin Dashboard' with a sub-section titled 'Add New Product'. The dashboard header includes 'MediCare' on the left and navigation links 'Products', 'Admin', 'Admin Dashboard', and 'Logout' on the right. The 'Add New Product' form contains the following fields:
 

- Product Name:** Atonide Gel
- Image URL:** https://th.bing.com/th/id/OIP.9FQTuE7q4n94bfqL07g64wHaHa?w=1798&h=11
- Price (₹):** 130
- Stock Quantity:** 50
- Age Category:** A dropdown menu currently showing '12+ years'.
- Description:** A text area containing 'Atonide Gel is used in the treatment of Skin conditions with inflammation & itching. View Atonide Gel (tube of 20.0 gm Gel)'.
- Image Preview:** A small image of the Atonide Gel product packaging.

 A blue 'Add Product' button is located at the bottom of the form.

**Figure B.10 Add Product**

In Figure B.10 represents, in which admin can add product and product category with price, description, etc.

## REFERENCES

1. <https://www.emedstore.in/pharmacy-website-development>
2. <https://www.weblinkindia.net/pharmacy-online-store-builder.htm>
3. [https://themeforest.net/search/medical%20store?srsltid=AfmBOodr6ar6RswHo3XC4P9LZ8WYBFjpXpr7q8TsIIyXOxEjcScxS\\_R](https://themeforest.net/search/medical%20store?srsltid=AfmBOodr6ar6RswHo3XC4P9LZ8WYBFjpXpr7q8TsIIyXOxEjcScxS_R)
4. <https://biz4commerce.com/medical-eCommerce-solution>