## PROJECT REPORT ON

**E-Commerce Using Mern**

## SUBMITTED IN PARTIAL FULFILLMENT

## OF

## THE REQUIREMENTS OF

## THE AWARDS OF THE

#### PG-Diploma in Advanced Secure Software Development

**Offered By**

**C-DAC Hyderabad**

BY

**NAME (ROLL NO)**

**Dhulipalla Sri Charan 250250329002**

**Harsh Patel 250250329003**

**Advanced Computing Training School  
C-DAC  
Hyderabad-500005**

**FEBRUARY 2025**



Certificate

This is to certify that this is a bonafide record of project entitled **“E-commerce using MERN ”**. Dhulipalla Sri Charan bearing rollno. 250250329002 and Harsh Patel bearing rollno. 250250329003 has completed project work as part of **Diploma in Advanced Secure Software Development (February 2025 Batch)**, a PG course offered by C-DAC Hyderabad. They have completed project work under the supervision of Mr. Deven Puttagunta. Their Performance found to be good.

**Name of Project guide**

(Mr. Deven Puttagunta)

**Date**  : 11-08-2025

**Place**  : C-DAC, Hardware Park, HYDERABAD.

**ACKNOWKEDGEMENT**

E-commerce using MERN has presented, an objective, a goal, a challenge of data security. This project marks the final hurdle that we tackle, of hopefully what would be one of the many challenges we have taken upon and am yet to take.

However, we could not have made it without the support and guidance from the following. Firstly I want to take this opportunity to have special thanks to our guide **Mr. Deven Puttagunta** who helped us throughout this project by providing valuable guidance and advice as well as acquiring all components needed for this project to become a success.

(PG-DASSD FEBRUARY 2025)

**NAME (ROLL NO)**

**Dhulipalla Sri Charan 250250329002**

**Harsh Patel 250250329003**

**TABLE OF CONTENTS**

**TOPIC PAGE**

**Abstract I**

**Introduction 1**

**1. Project Requirements 2**

**1.2 Problem Statement 2**

**1.3 Software & hardware Requirements 2**

**1.3.1 Hardware Specification 2**

**1.3.2 Software Specification 2**

**1.3 Coding and Testing 2**

**2.Implementation 3**

**2.1 Frontend 3**

**2.1.1 ReactJS 3**

**2.1.2 React Router 3**

**2.1.3 Tailwind CSS 3**

**2.1.4 Axios 3**

**2.2 Backend 4**

**2.2.1 NodeJS and ExpressJS 4**

**2.2.2 Bcrypt, Json Web Token(cookie), 4**

**2.3 Database 4**

**2.4 Configuration 4**

**3. System Architecture and modules 5**

**3.1 Core Modules 5 3.2 Key definition 5**

**4.Objectives 6**

**5.ER diagram 7**

**5.1 What is ER 7 5.2 Data flow diagram 8-9**

**6. Admin/User Workflow 10**

**6.1 User workflow 10-14**

**6.2 Admin workflow 14-17**

**7. Admin Dashboard Functionality 18**

**7.1 Flow of Revelation Process 18**

**8. Project Summary 19**

**8.1 List of Featured Implemented 19**

**8.2 Advantages 19**

**8.3 Application 19**

**8.4 Future Scope 20**

**8.5 Conclusion 20**

**9. Literature Survey 21**

**10. References 22**

**E-Commerce using MERN stack**

**Abstract :**

**The project** is a modern, full-stack e-commerce web application built using the MERN (MongoDB, Express.js, React.js, Node.js) stack. The project aims to provide a seamless and secure online shopping experience for users, while offering a comprehensive and easy-to-use dashboard for administrators to manage the store's inventory, orders, and users.

The primary function of the application is to allow users to browse a wide range of products, filter them by category, and search for specific items. Users can view detailed product information, add items to their shopping cart, and proceed through a complete checkout process. The system features robust user authentication, ensuring that personal information and order history are kept secure. For administrators, a separate, protected dashboard provides full control over the e-commerce platform, including the ability to add, update, and delete products, and oversee the user base.

**Implementation :**

This application is developed using **React.js** for a dynamic and responsive frontend, with **Tailwind CSS** for styling. The backend is powered by **Node.js** and **Express.js**, providing a RESTful API for all frontend operations. **MongoDB** is used as the database, with **Mongoose** for object data modeling. User authentication is handled securely using JSON Web Tokens (JWT) stored in httpOnly cookies. The result is a scalable, secure, and modern e-commerce solution that serves both customers and administrators effectively.

**INTRODUCTION**

In the current digital era, e-commerce has become an indispensable part of the global retail landscape. A well-designed e-commerce platform is crucial for businesses to reach a wider audience and provide a convenient shopping experience. The **E-commerce** project is a web application designed to simulate a real-world online store, incorporating all the essential features from product browsing to a secure checkout and administrative management alongside.

This project leverages the power of the MERN stack to create a fast, responsive, and scalable single-page application (SPA). By separating the frontend (client-side) and backend (server-side), we ensure a modular architecture that is easier to maintain and extend. The application provides two distinct user experiences: one for the customer, focused on a seamless shopping journey, and another for the administrator, focused on efficient store management.

**1.PROJECT REQUIREMENTS**

**1.1 Problem Statement:**

* User registration and login with secure authentication.
* A public-facing product catalog with search and filtering capabilities.
* A persistent shopping cart for authenticated users.
* A complete, multi-step checkout process.
* A protected admin dashboard for managing products, and users.
* Conditional routing to differentiate between regular users and administrators.

**1.2 Software & hardware Requirements :**

**1.2.1 Hardware Specification:**

* Processor – Multi-core processor (e.g., Intel i3/i5/i7, AMD Ryzen series)
* Hard Disk – Minimum 50 GB of free space
* Memory – Minimum 8 GB RAM

**1.2.2Software Specification**:

* Operating System : Windows, macOS, or a Linux (preferably Linux )
* Runtime Environment : NodeJS ( v22.x or later)
* Package Manager : npm(v9.x or later)
* Database Server : MongoDB (v6.x or later) with MongoDB Compass
* Web Browser : A modern browser like Google Chrome, Firefox, or Microsoft Edge
* Code Editor : Visual Studio Code (or any other modern editor)

**1.3 Coding and Testing :**

In this we develops a code using following software languages:

* **Frontend** : JavaScript (ES6+), React.js, HTML5, Tailwind CSS.
* **Backend** : JavaScript (ES6+), Node.js, Express.js
* **Database** : MongoDB (using Mongoose ODM)
* Testing : Manual testing was conducted throughout the development process to ensure all features work as expected.

**2.IMPLEMENTATION**

The **E-commerce using MERN** project was developed following the MERN stack architecture, which allows for a clean separation of concerns between the client-side and server-side.

* 1. **Frontend :**

The frontend is a single-page application (SPA) built with **React.js**. This approach ensures a fast and fluid user experience, as page reloads are not required for navigation.

* + 1. **ReactJS :** React.js is a JavaScript linrary for building user interfaces. We used it to create reusable UI components for various parts of the application, such as navbar, product cards, and forms. State management was handled using React Hooks (useState, useEffect, useContext) to manage component-level state and global state (like user authentication and the shopping cart).
    2. **React Router :** react-router-dom was used to handle all client-side routing. This allowed us to create different "pages" (e.g., /products, /cart, /admin/dashboard) and protect certain routes from unauthorized access. We implemented both standard protected routes for logged-in users and a specialized protected route for administrators.
    3. **Tailwind CSS :** All styling for the application was done using **Tailwind CSS**. This utility-first CSS framework allowed us to build a modern, responsive, and consistent design directly within our JSX markup, without writing custom CSS files.
    4. **Axios : Axios** was used as the HTTP client to make requests from the frontend to our backend API. We created organized service modules (authService, productService, etc.) to manage all API calls, ensuring that the authentication token (cookie) was sent with every protected request.
  1. **Backend :**

The backend is a RESTful API built with **Node.js** and the **Express.js** framework. It handles all business logic, database interactions, and user authentication.

* + 1. **NodeJS & ExpressJS : Node.js** provides the Javascript runtime environment for our server. **Express.js** is a minimal and flexible Node.js web application framework that we used to define our API routes and handle HTTP requests and responses. We structured our backend with clear separation for routes, controllers (business logic), models (database schemas).
    2. **Middleware :** We used several middleware functions in our Express application:
       - * **cors**: To enable Cross-Origin Resource Sharing, allowing our frontend (on a different port) to communicate with the backend.
         * **cookie-parser**: To parse cookies sent from the browser, which is essential for our JWT authentication.
         * **Custom Authentication Middleware**: We wrote a custom middleware ( isAuthenticated.js ) to protect our backend routes. It decodes the JWT from the cookie and attahces the user’s information to the request object, ensuring only authenticated and authorized users can access protected endpoints.
  1. **Database :**

The database for our application is **MongoDB**, a NoSQL, document-oriented database. We used **Mongoose**, an Object Data Modeling (ODM) library, to define schemas for our data collections (users, products, and orders) and to interact with the MongoDB database in a structured way.

**2.4 Configuration :**

The application uses a .env file to store all environment variables, such as the database connection string, JWT secret, and admin credentials. This ensures that sensitive information is kept separate from the source code.

**3. SYSTEM ARCHITECTURE & MODULES**

### 3.1 Core Modules

The E-Commerce using MERN application is divided into several core modules:

* **Authentication Module**: Handles user registration, login (for both users and admins), and logout. It uses JWT for secure session management.
* **Product Module**: Manages all product-related operations. This includes fetching all products with filtering and pagination for users, and full CRUD (Create, Read, Update, Delete) functionality for admins.
* **Cart Module**: Manages the user's shopping cart. This is a persistent cart that is saved to the database for each user.
* **Order Module**: Handles the entire checkout process, from creating a new order to viewing past orders.
* **Admin Module**: A protected section of the application that provides a dashboard for managing products, orders, and users.

### 3.2 Key Definitions

* **MERN Stack**: A technology stack composed of MongoDB, Express.js, React.js, and Node.js.
* **RESTful API**: An architectural style for designing networked applications, where the frontend communicates with the backend using standard HTTP methods (GET, POST, PUT, DELETE).
* **JSON Web Token (JWT)**: A compact, URL-safe means of representing claims to be transferred between two parties. We use it to manage user authentication sessions.
* **Single-Page Application (SPA):** A web application that interacts with the user by dynamically rewriting the current web page with new data from the web server, instead of the default method of the browser loading entire new pages.

**4. OBJECTIVES**

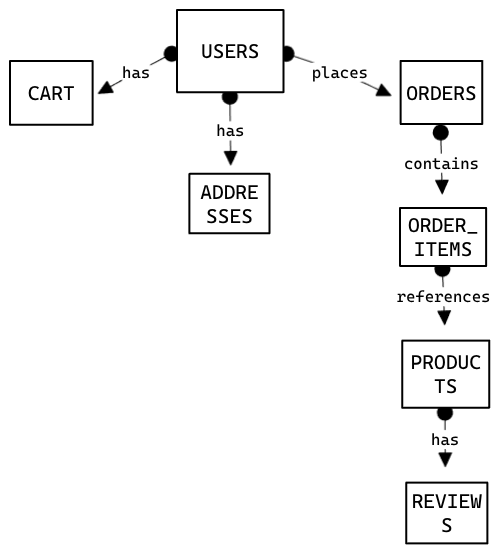
The primary objectives of the **E-commerce using MERN** project were to:

* Develop a fully functional, end-to-end e-commerce platform using the MERN stack.
* Implement a secure user authentication system with role-based access control (user vs. admin).
* Create a clean, responsive, and intuitive user interface for a seamless shopping experience.
* Build a comprehensive admin dashboard for complete control over the store's data.
* Ensure the application is scalable and maintainable by following best practices in code structure and separation of concerns.

**5. ER Diagram**

**5.1 What is E-R :**

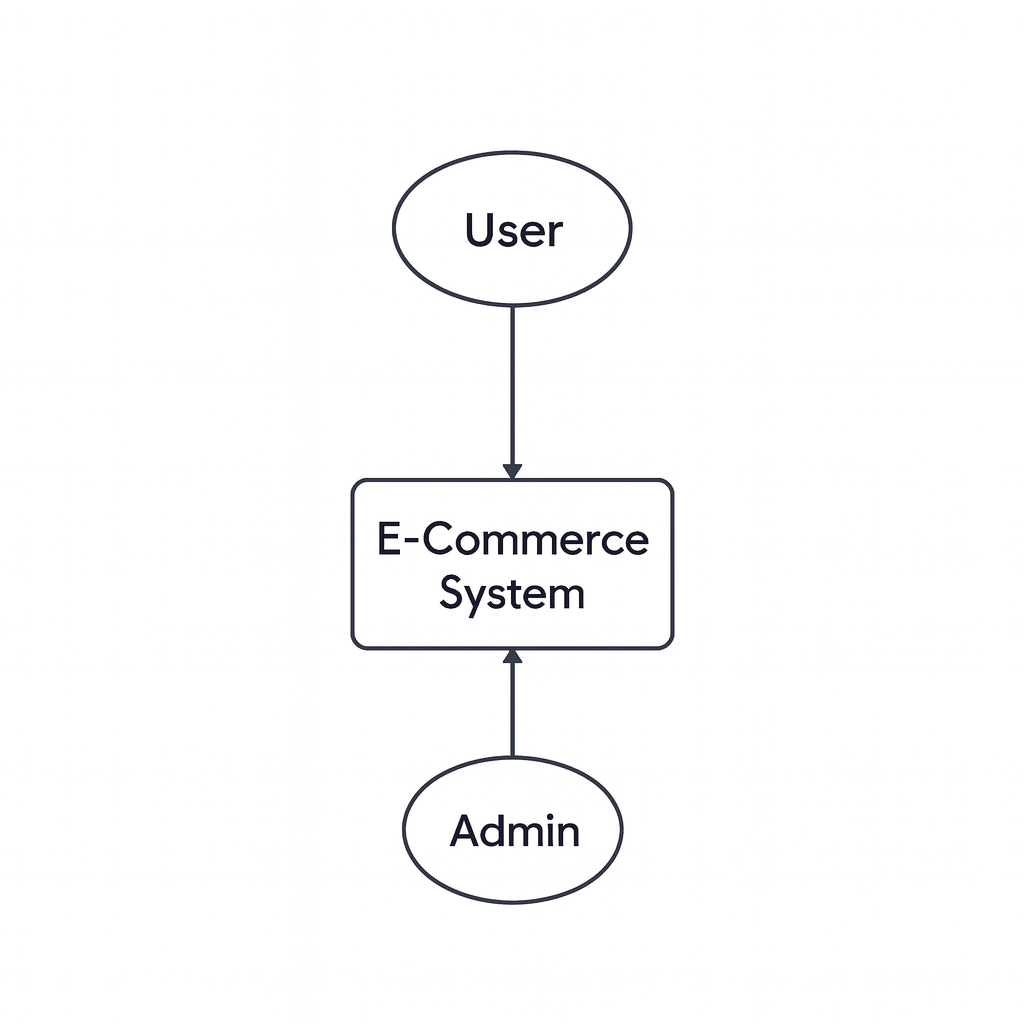
Database is absolutely an integral part of software system. To fully utilize ER Diagram in database engineering guarantee you to produce high quality database design to use in database creation, management and maintenance. An ER model also provides a means for communication.

**~~~~**

**Fig 5.1.a : ER Diagram**

**5.2 Data Flow Diagram :**

**Level 0 :**

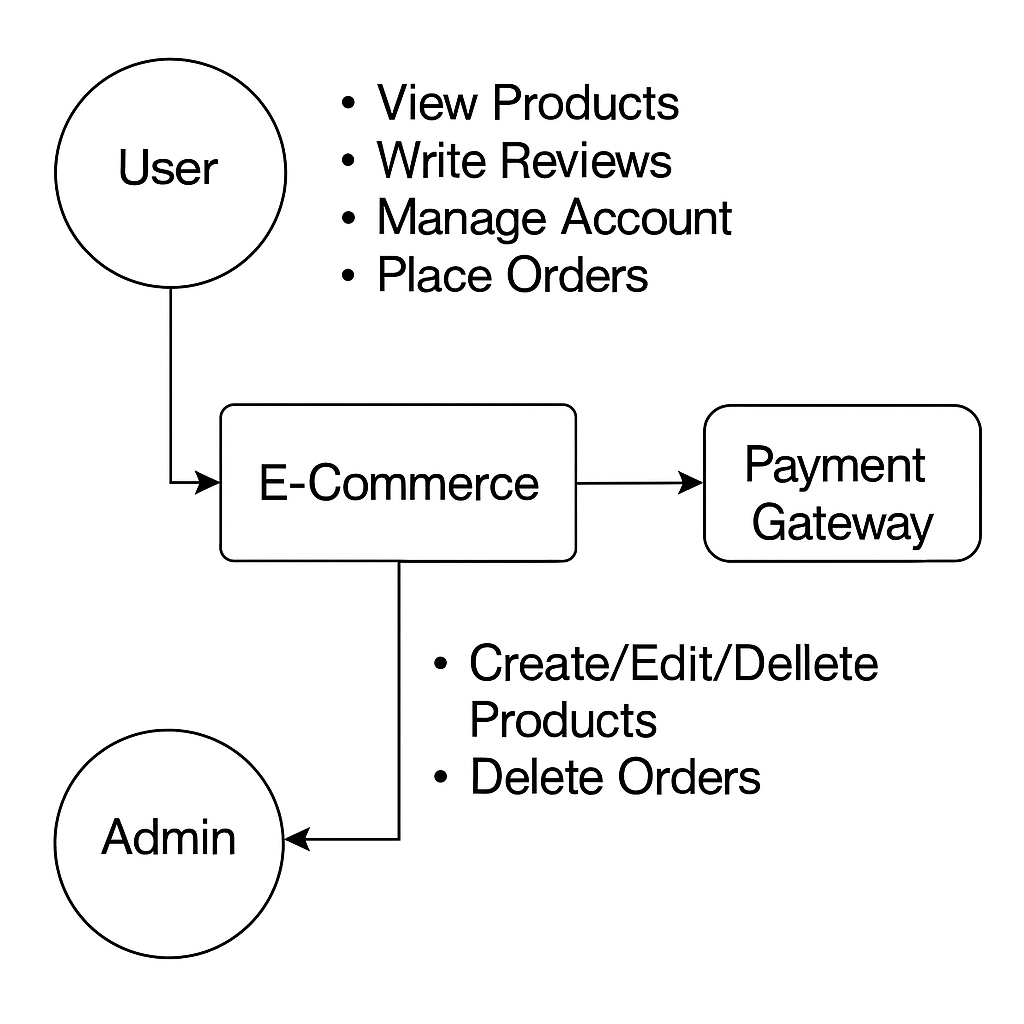


**Fig 5.2.1 :level 0 Diagram**

**Description:**

The Level 0 DFD shows the overall system interaction. The **User** and the **Admin** are the two external entities that interact with the **E-Commerce System**. The user can browse products, manage their cart, and place orders, while the admin manages the store's data.

**Level 1:**



****Fig 5.2.2 :level 1 Diagram****

**Description:**

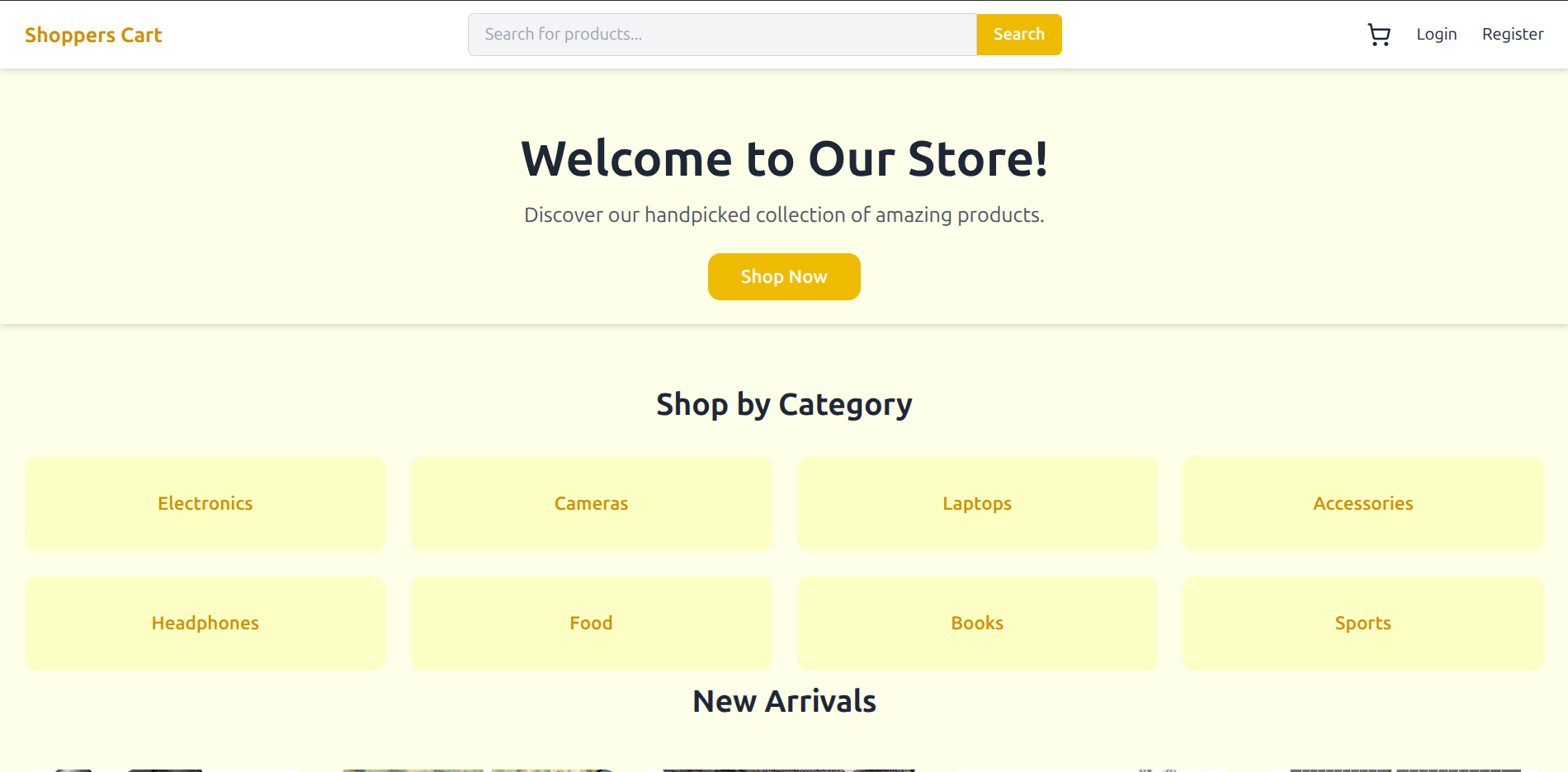
The Level 1 DFD breaks down the main system into its core processes:

1. **1.0 Manage User Account**: Handles login, registration, and profile updates.
2. **2.0 Browse Products**: Manages fetching and displaying products with search and filtering.
3. **3.0 Manage Cart & Orders**: Handles adding items to the cart and the entire checkout process.
4. **4.0 Admin Functions**: Manages all administrative tasks, such as product, order, and user management.

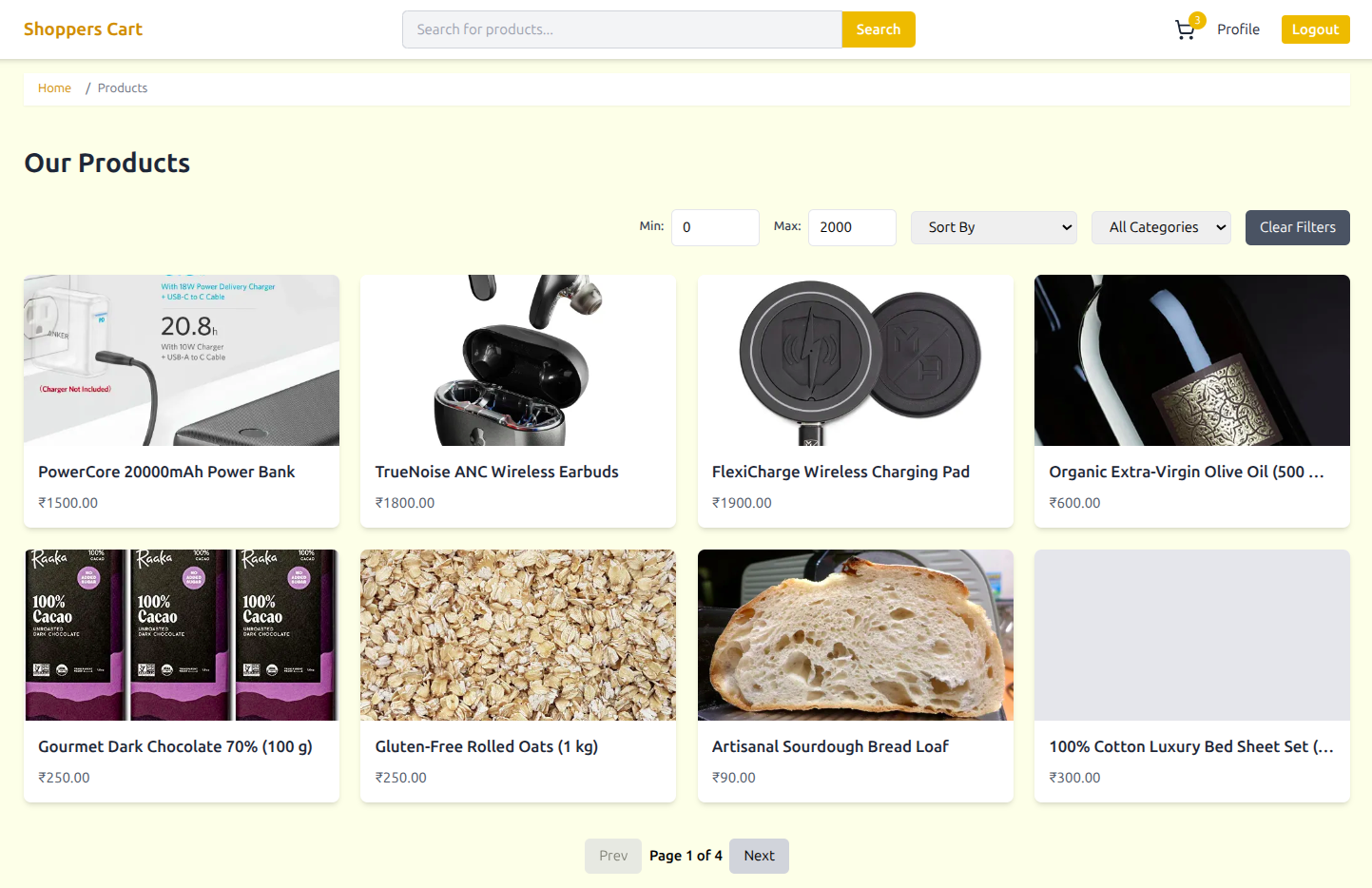
**6. User Workflow**

### 6.1 Customer Flow

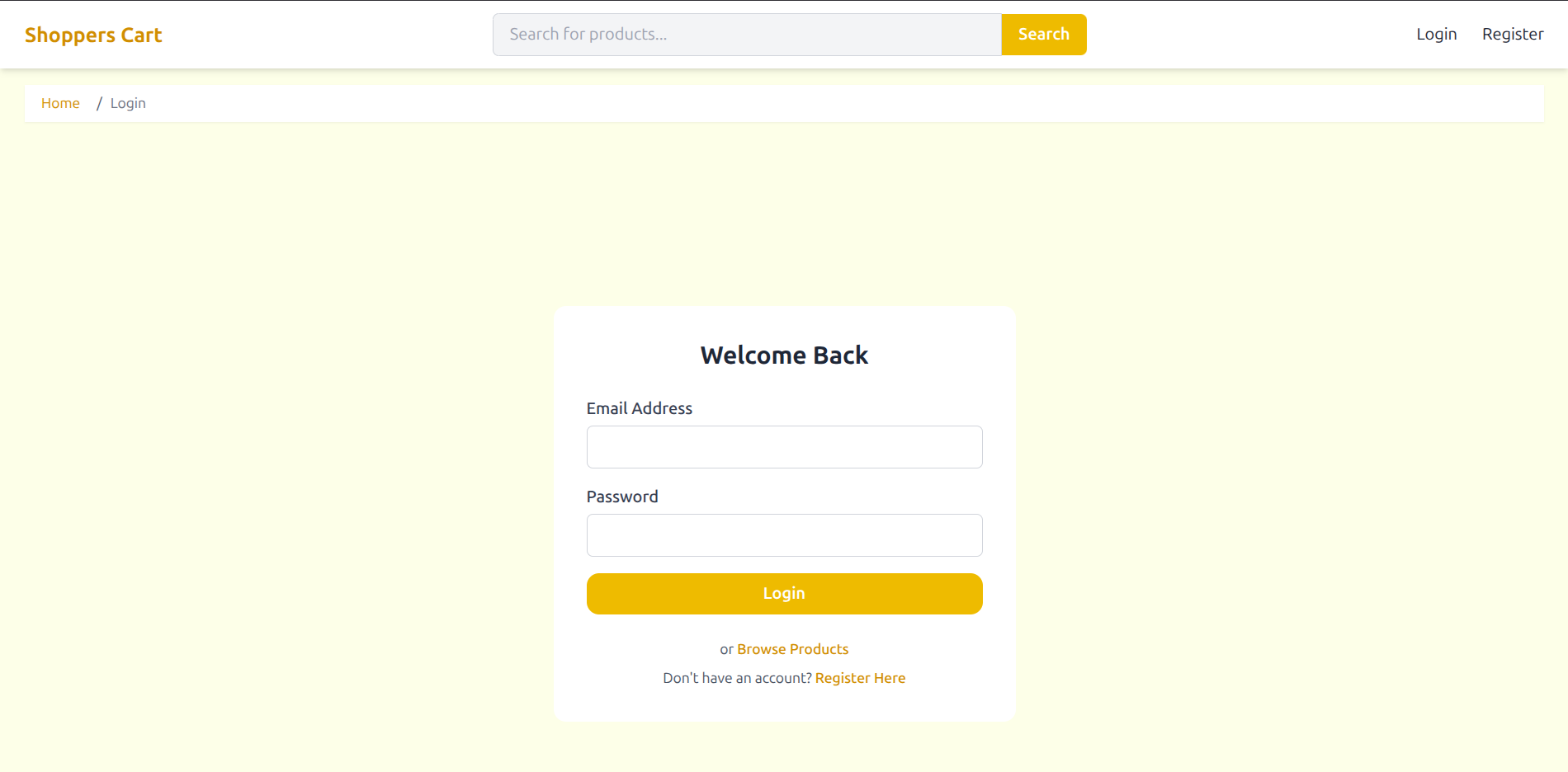
1. A new user visits the homepage. The homepage has New arrivals, Featured, Best seller, Top rated, Recently Viewed for more categorization.



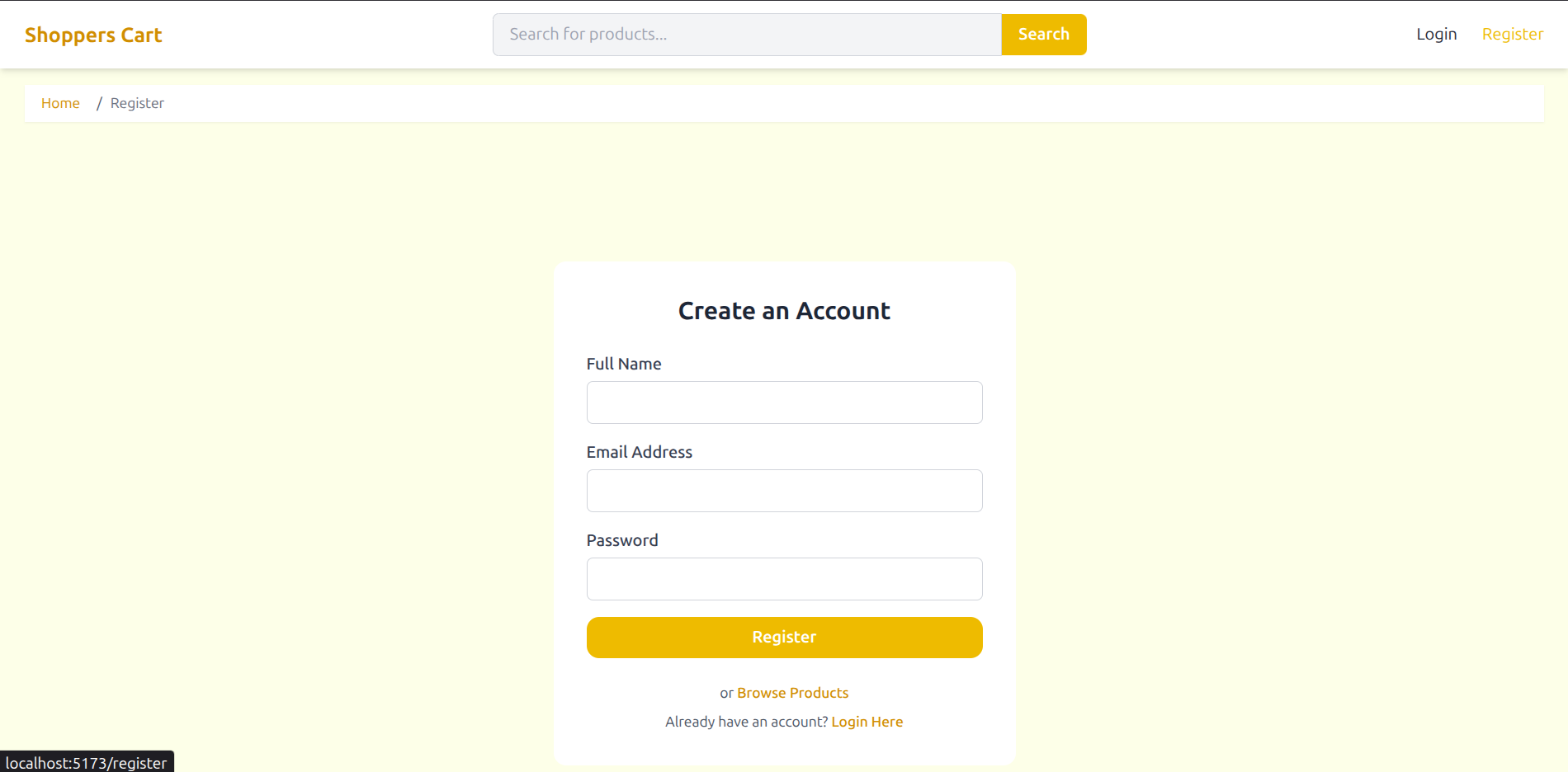
1. Users can browse all products, use the dynamic search bar, or filter by category, sort by prices, apply price ranges.



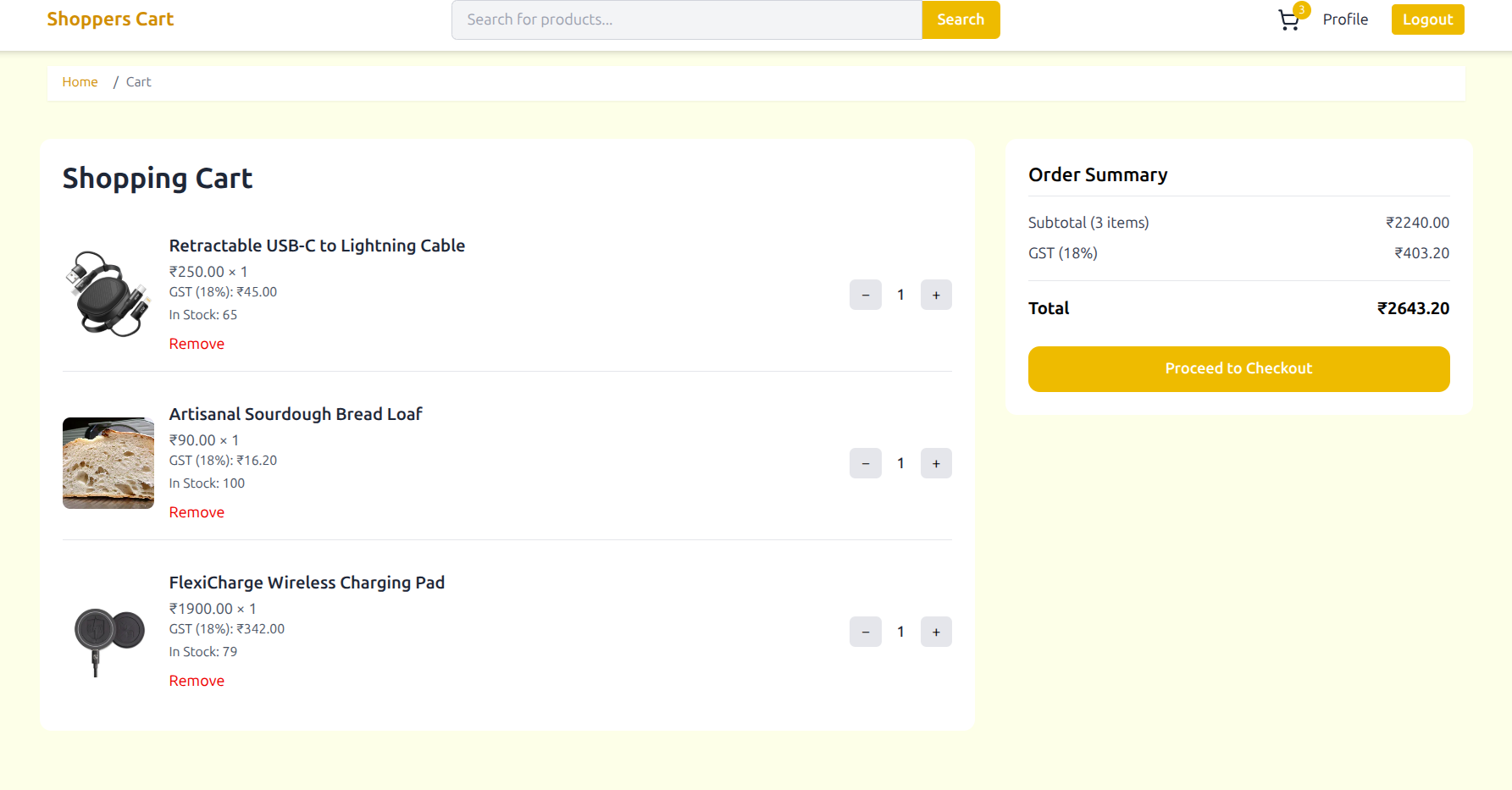
1. To add an item to the cart, they are prompted to log in, if not they can register.



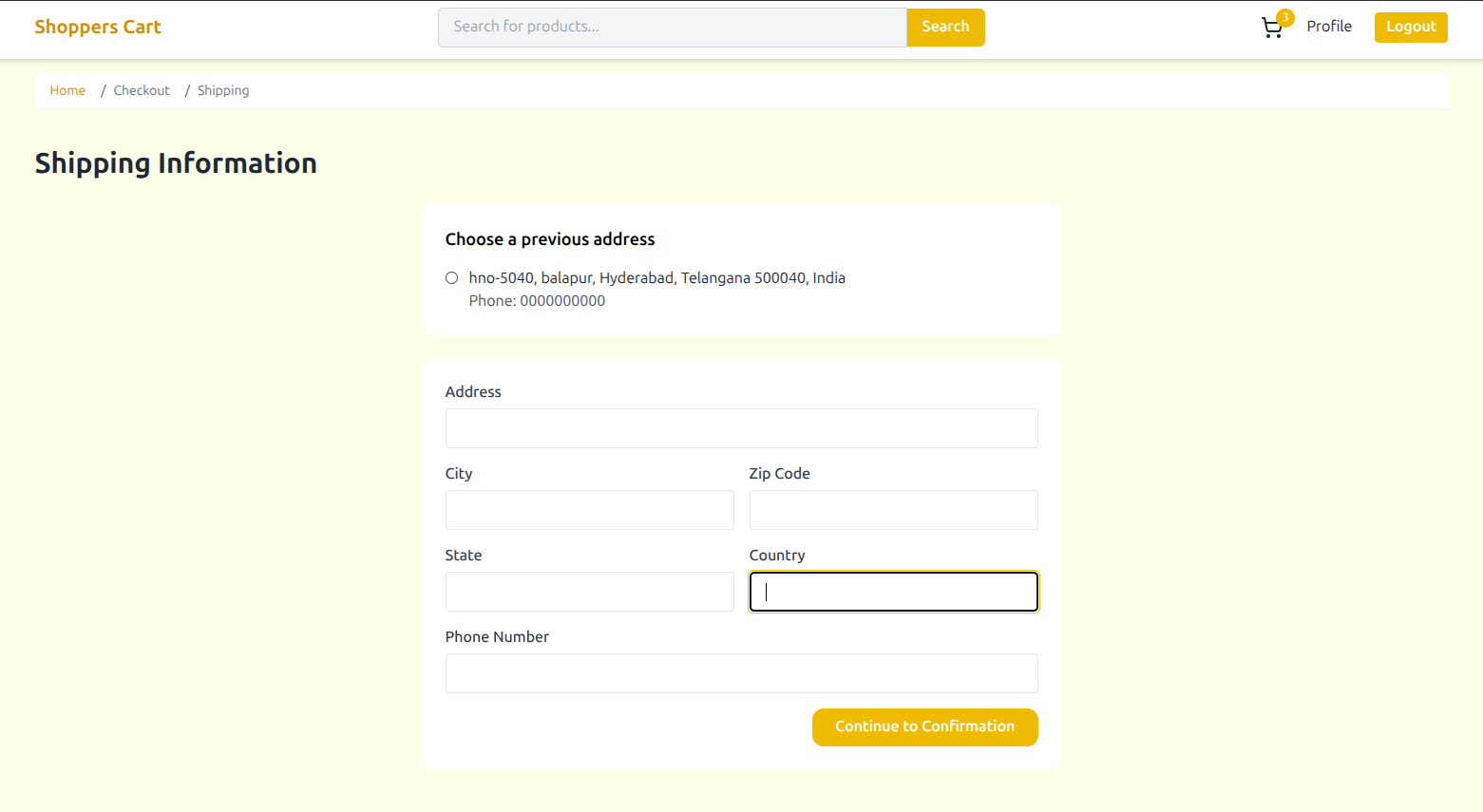
1. If not registered, user can register in the register page.



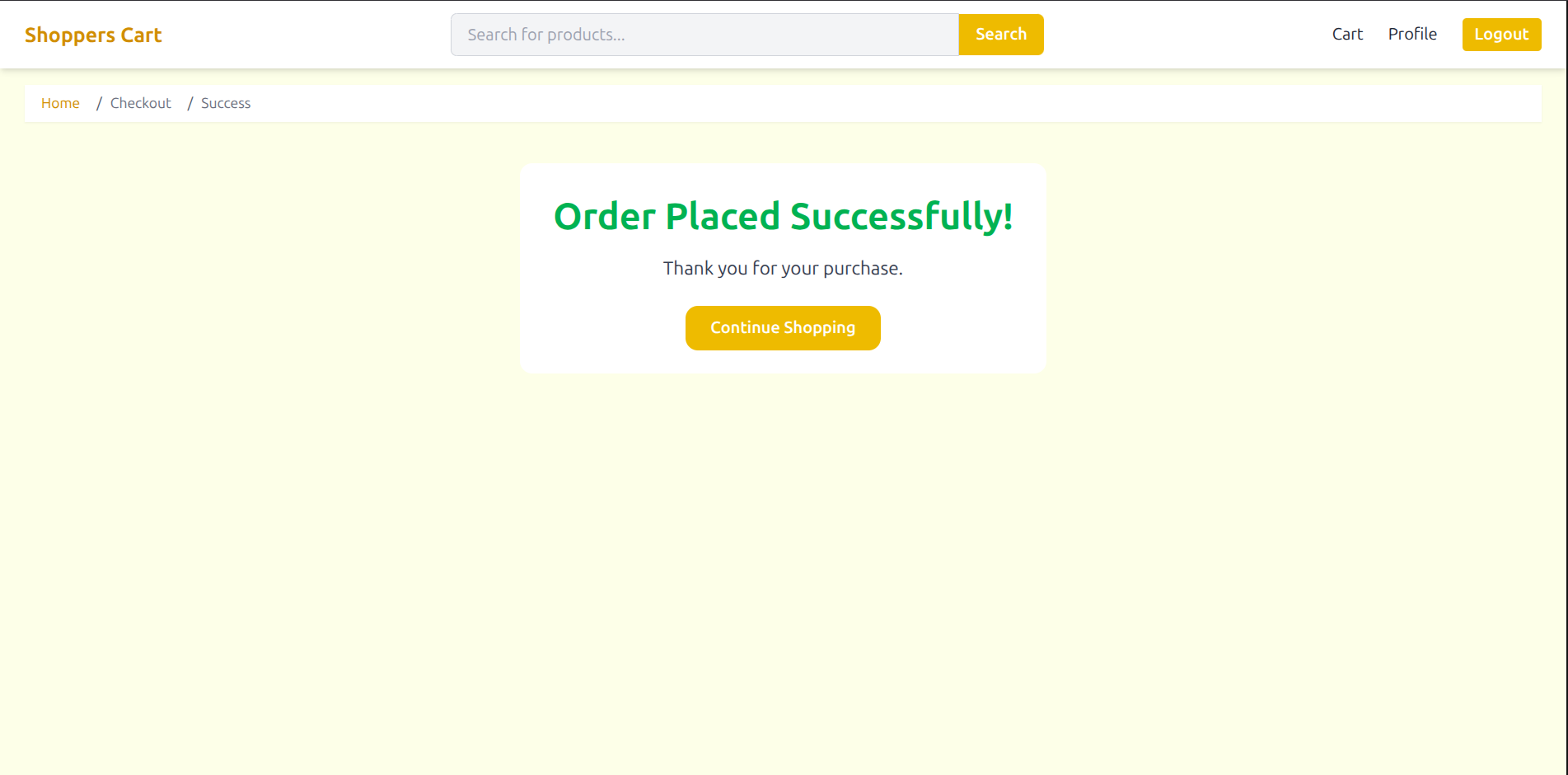
1. After logging in, they can add items to their cart, update quantities, or remove items.



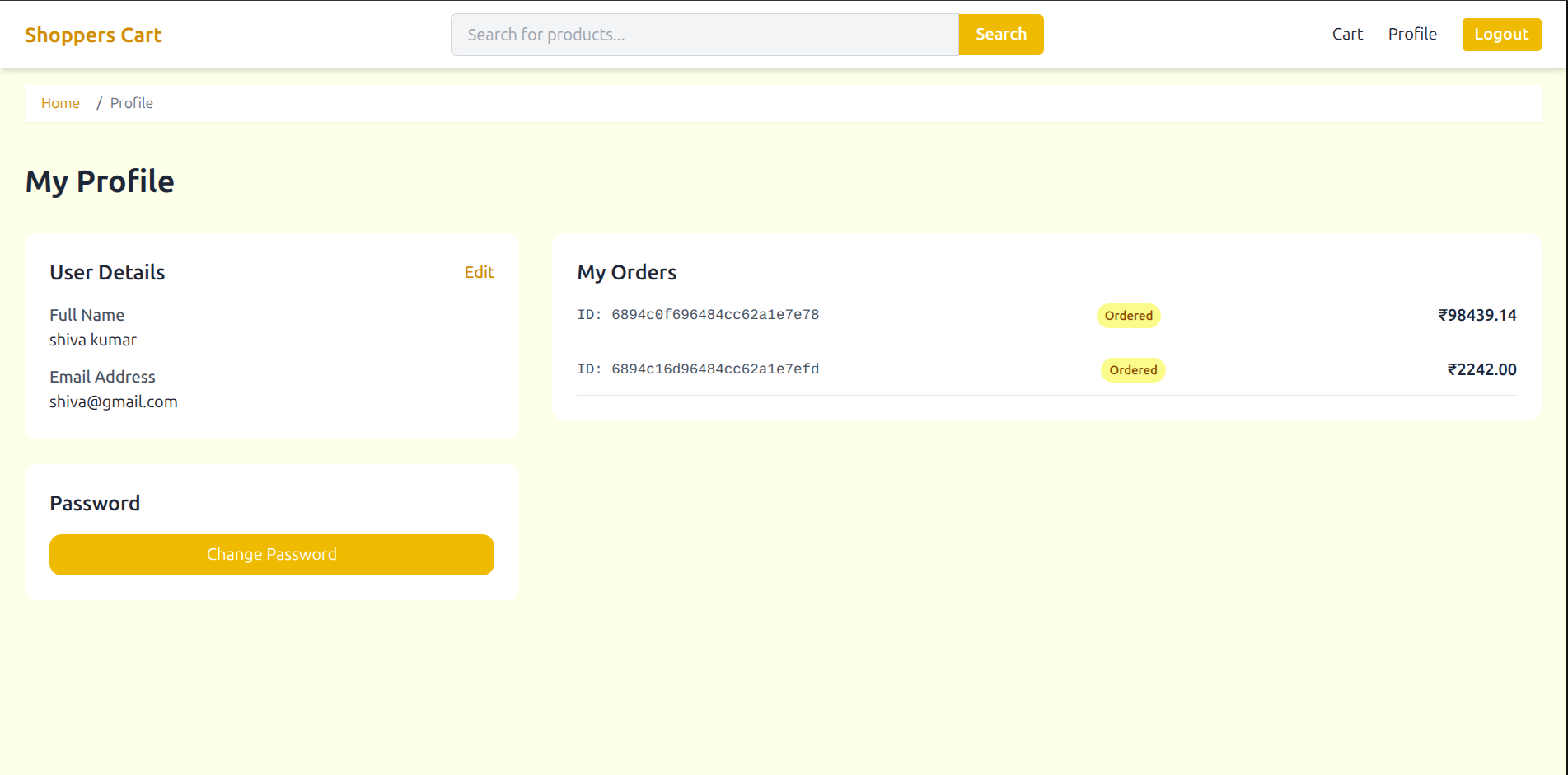
1. The user proceeds to checkout, enters their shipping information, and in next page confirms the order.



1. After placing the order, they are redirected to a success page.



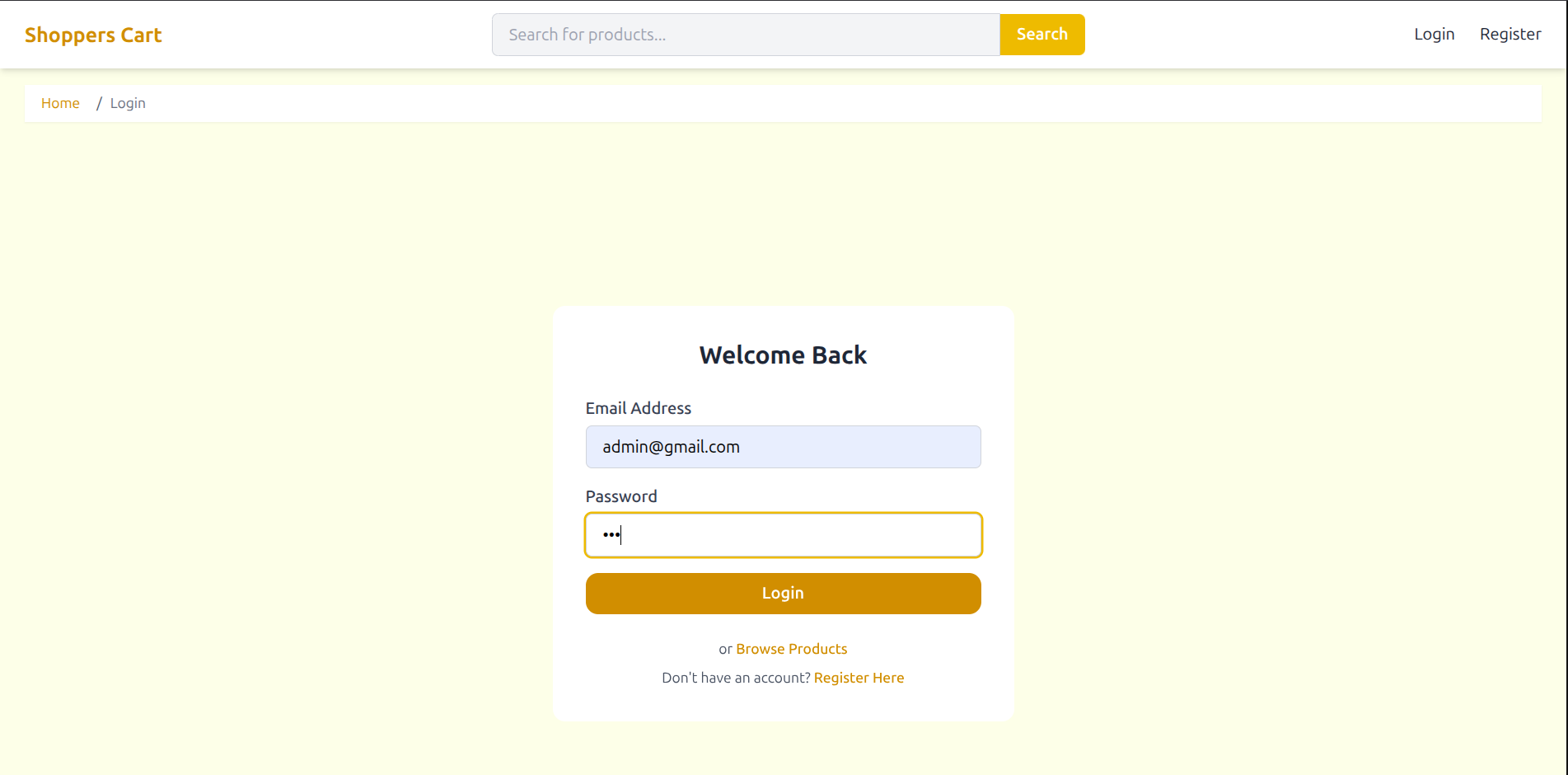
1. The user can view their profile and see their order history at any time.



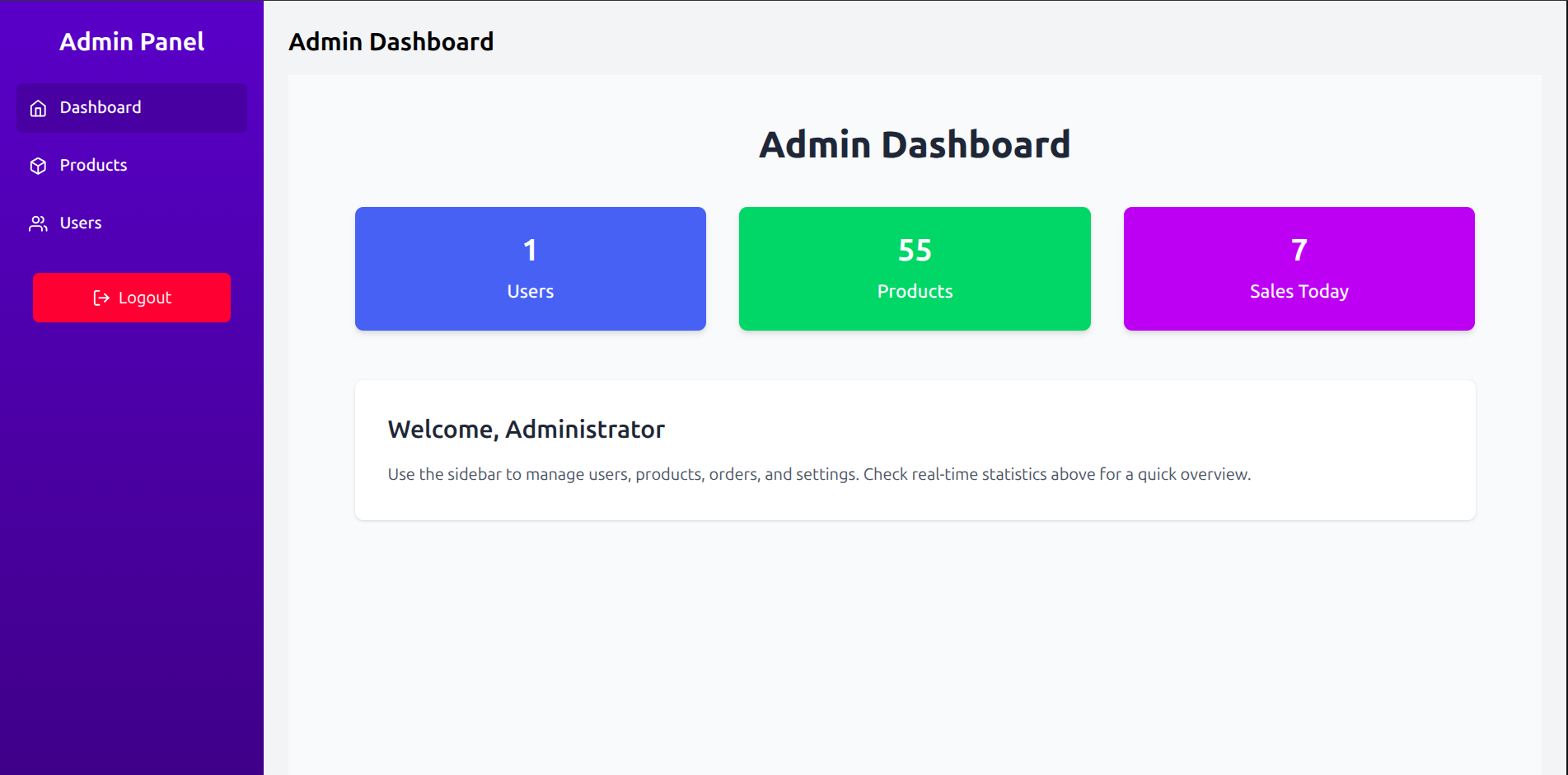
### 

### 6.2 Admin Flow

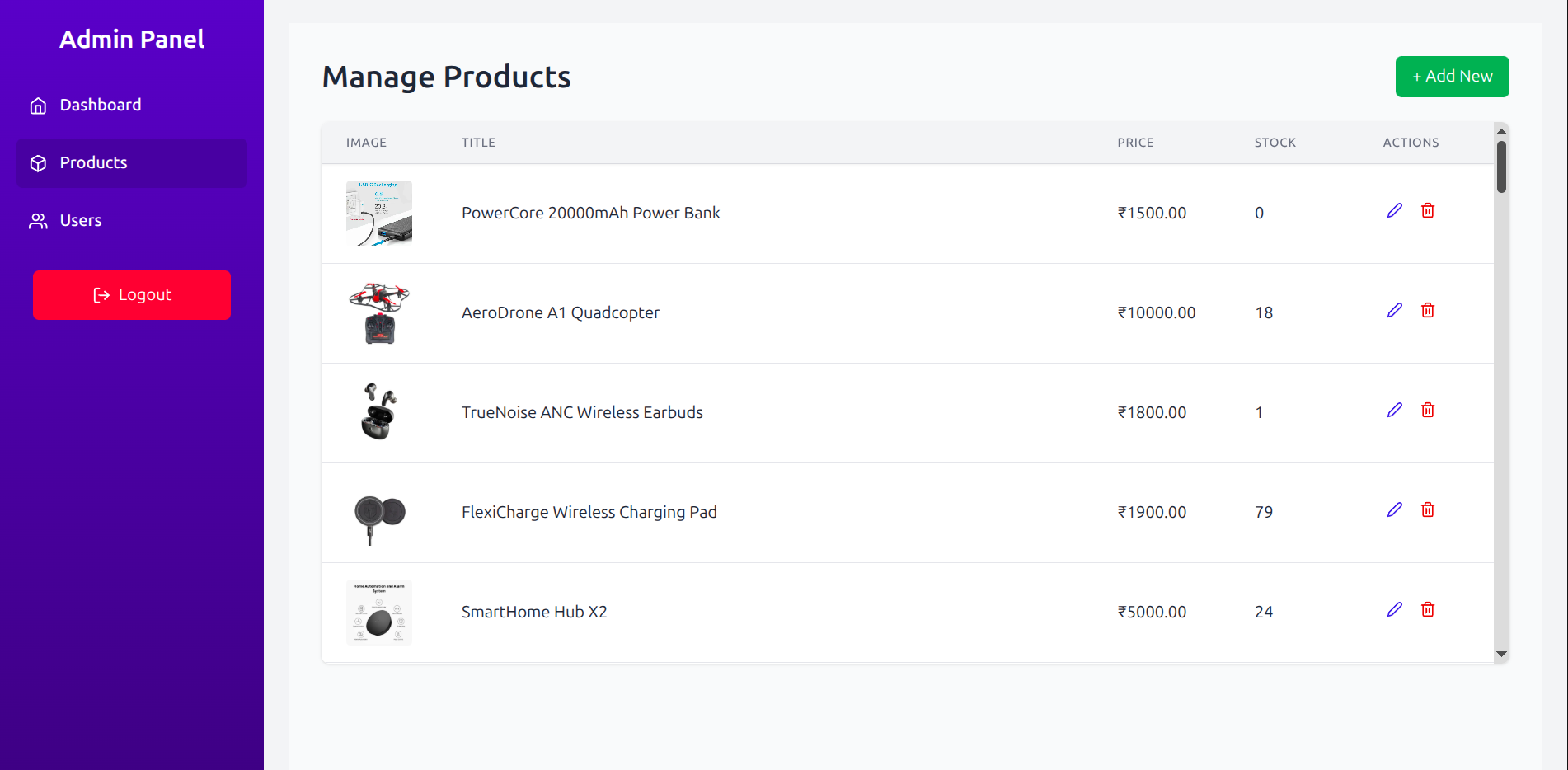
1. The admin logs in using their special credentials on the same login page as users.



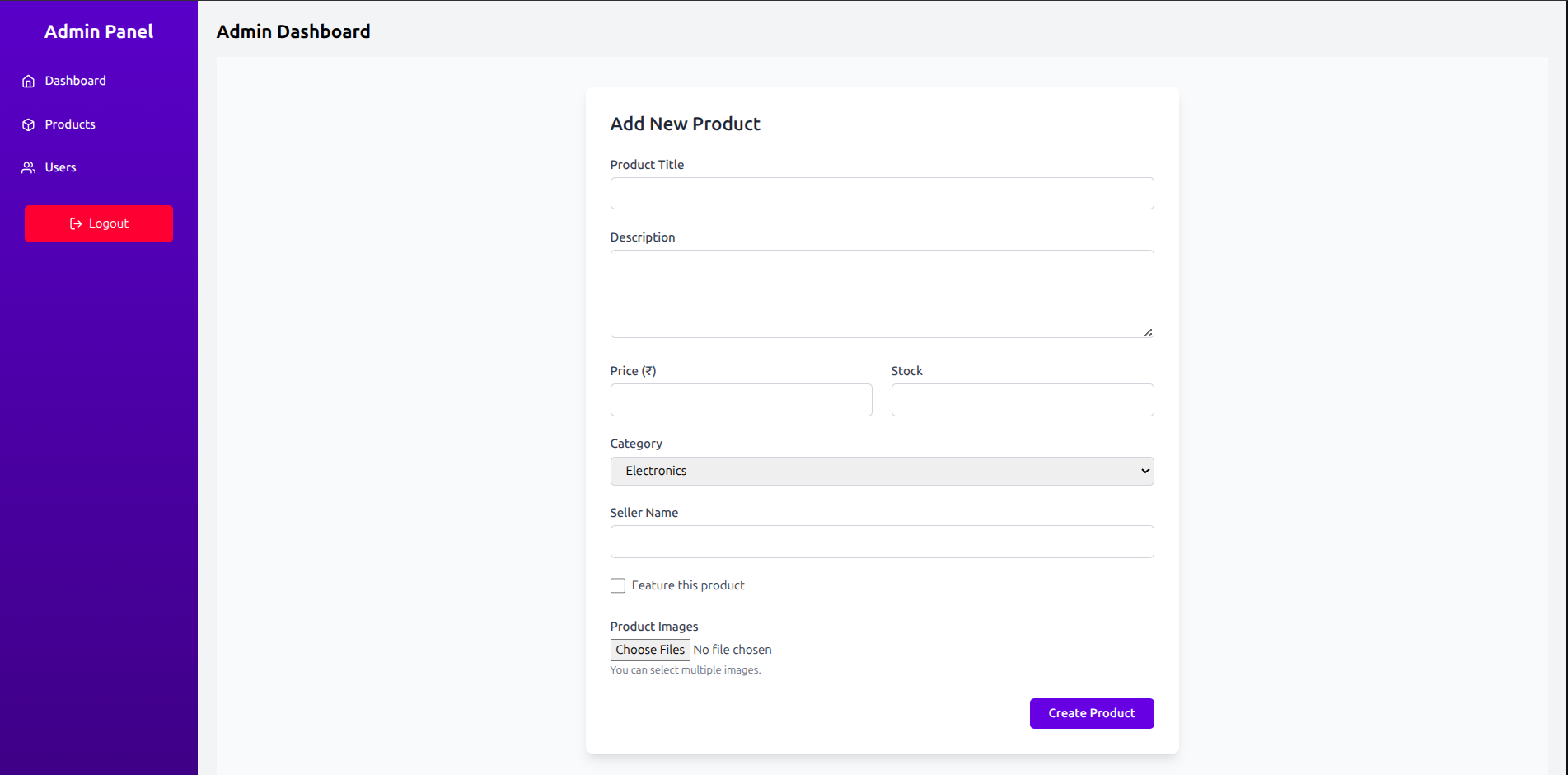
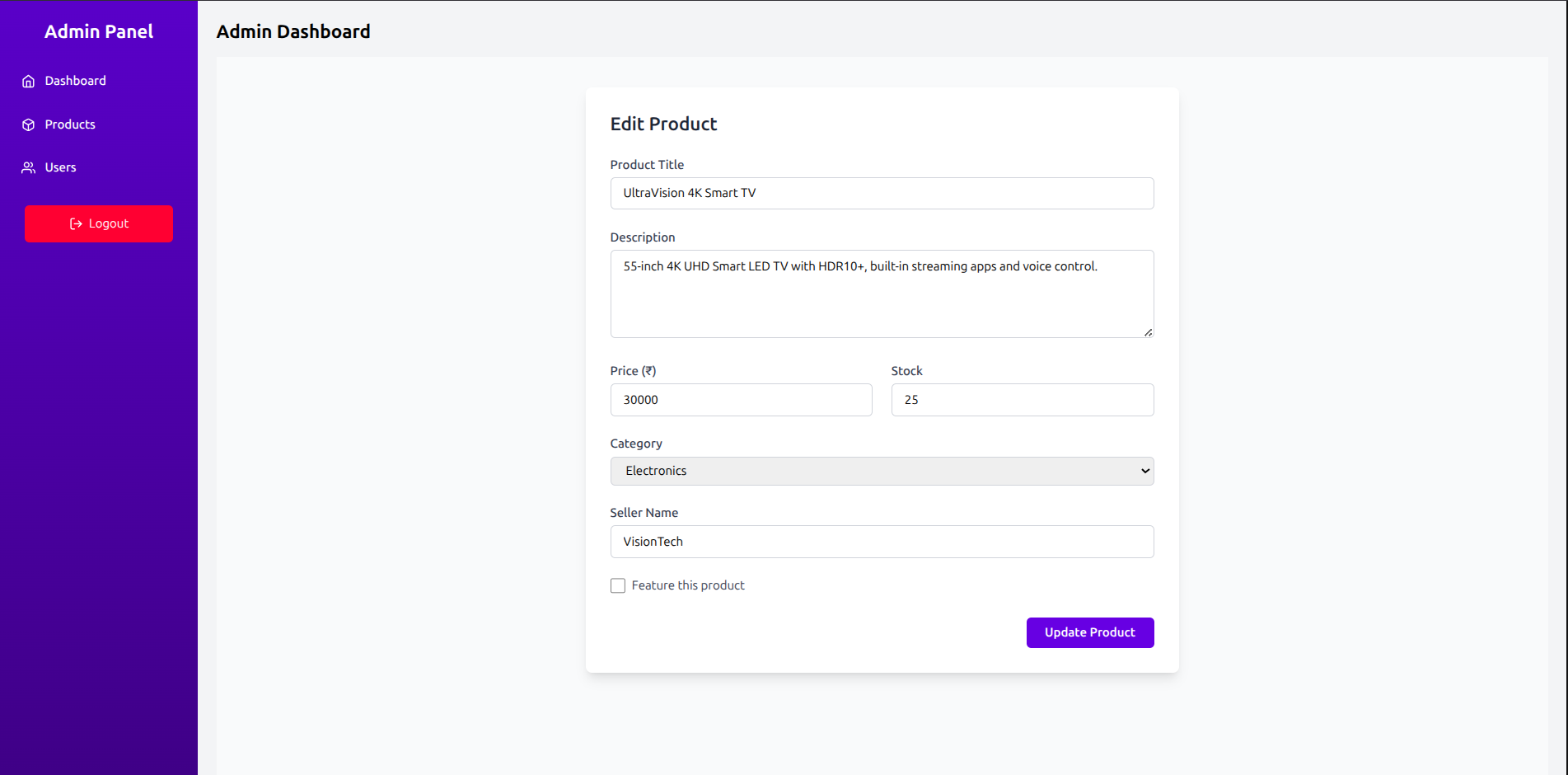
2. Upon successful login, they are redirected to the protected /admin/dashboard. Where admin can see real time data like users registered, products, sales today.

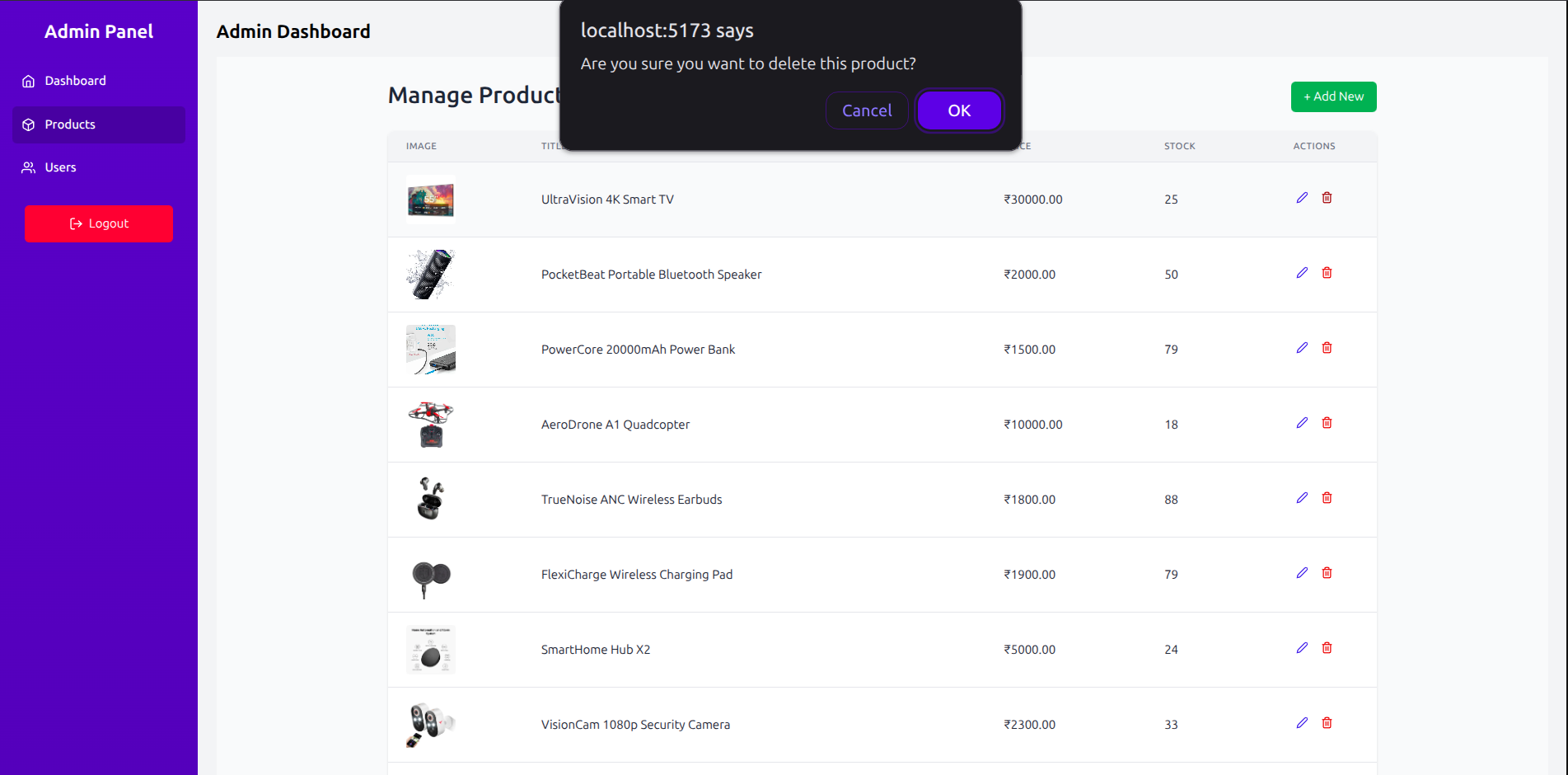


3. From the sidebar, the admin can navigate to manage products, or users.

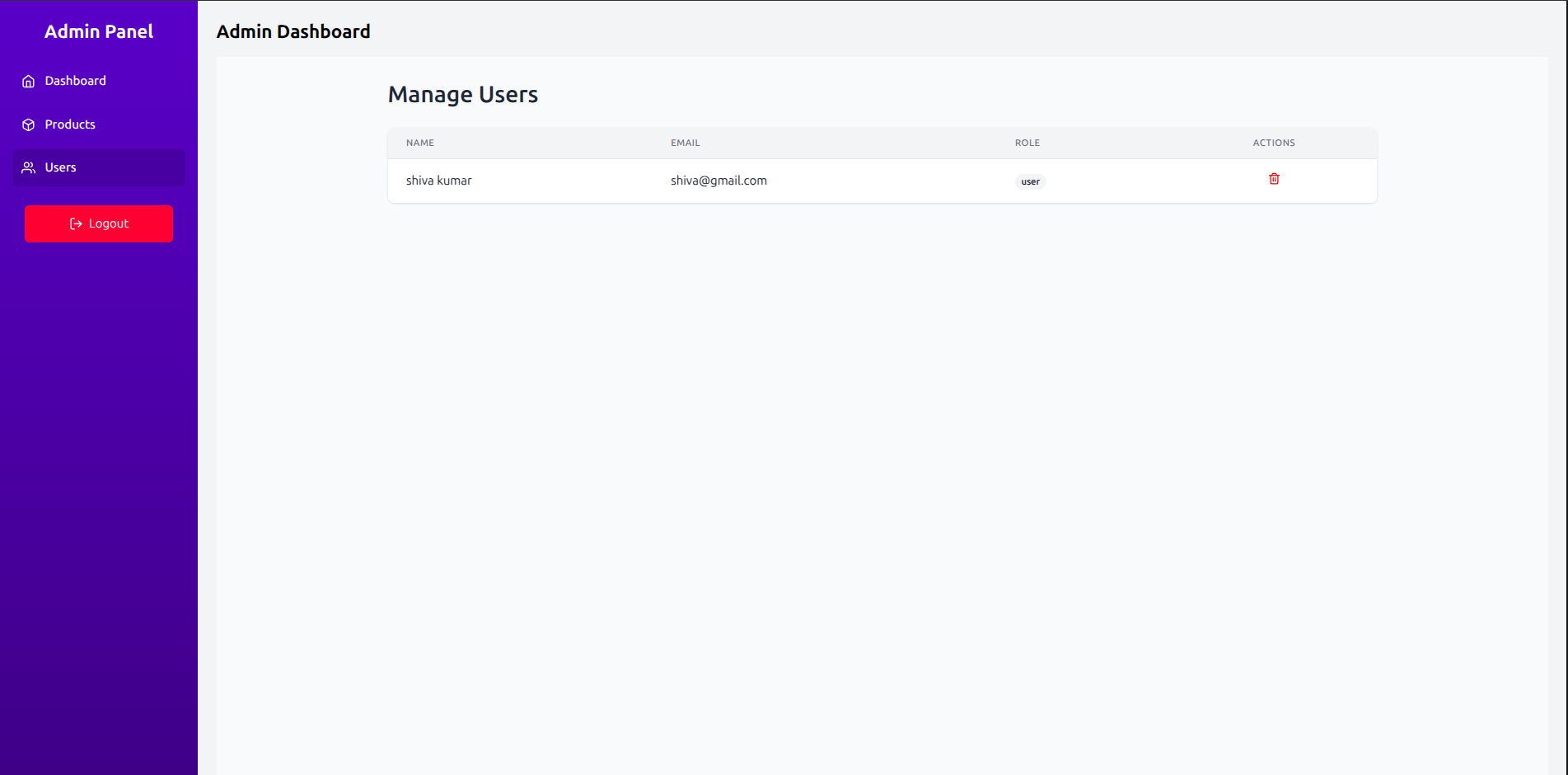


4. The admin can create new products, edit existing ones, or delete them.





5. The admin can view a list of all registered users.



## **7. ADMIN DASHBOARD FUNCTIONALITY**

### 7.1 Key Features

The admin dashboard is a protected area accessible only to users with the "admin" role. It includes:

* **Product Management**: A table view of all products with options to **Add**, **Edit**, and **Delete** any product.
* **User Management**: A view of all registered users in the database.

**8. Project Summary**

### 8.1 List of Features Implemented

* User and Admin Authentication (Single Login)
* Product Catalog with Search, Filter, and Pagination, Sort by and price ranges
* Persistent Shopping Cart in database so cart persists between logins.
* Multi-Step Checkout Flow
* User Profile Page with Order History
* Users can update their name and password(with mean time between reattempts) in the Profile Page.
* Users can access their orders in the Profile Page, where upon clicking they are redirected to a dedicated order summary page.
* Admin Dashboard with Product, and User Management where they can add, edit, delete products and delete users.
* Responsive Design with Tailwind CSS

### 8.2 Advantages

* **Full-Stack Solution**: A complete, end-to-end application built with a modern technology stack.
* **Secure**: Uses JWT and httpOnly cookies for secure authentication and protected routes.
* **User-Friendly**: An intuitive and responsive UI for a seamless experience on any device.
* **Scalable**: The MERN stack and modular architecture allow for easy expansion and addition of new features.

### 8.3 Application

This project serves as a robust template for building real-world e-commerce websites for small to medium-sized businesses. It can be adapted for various types of online stores, from selling physical goods to digital products by the UI is simple and user friendly.

### 8.4 Future Scope

The project can be extended with more advanced features, such as:

* Integration with a real payment gateway like Stripe or PayPal.
* Personalized product recommendations for users by integrating ml models.
* More advanced admin analytics and reporting like order status management integrated to the delivery system.

**8.5 Conclusion**

The **E-commerce using MERN** project successfully demonstrates the development of a modern, secure, and feature-rich e-commerce platform using the MERN stack. It meets all the core objectives, providing a seamless experience for customers and a powerful management tool for administrators. The project serves as a strong foundation that can be further developed into a production-ready application.

**9. LITERATURE SURVEY ON E-Commerce using MERN:**

Here is the summary of the various scholarly articles which we read.

1) **"Building Modern E-commerce Websites with the MERN Stack"** (2020) by A. Kamal - This paper provides an overview of the MERN stack's components and discusses its application in developing Ecommerce platforms. It highlights the benefits of using MongoDB, Express.js, React.js, and Node.js for building scalable and efficient web applications.

2) **"Scalability Testing of MongoDB for E-commerce Applications"** (2019) by S. Rathore et al. - This paper evaluates MongoDB's performance in handling large-scale E-commerce datasets. It discusses scalability testing methodologies and presents findings on MongoDB's suitability for E-commerce database management.

3) **"Enhancing Security in MongoDB for E-commerce Database Management"** (2021) by V. Kumar et al. - This paper focuses on enhancing security measures in MongoDB for E-commerce applications. It discusses encryption techniques, access control mechanisms, and data privacy considerations in the context of Ecommerce database management.

4) **"Performance Evaluation of Node.js for Real-time Web Applications"** (2021) by F. Ahmad et al. - This paper examines Node.js's performance characteristics in real-time web applications, including its suitability for handling E-commerce transaction processing and server-side logic.

5) **"User Interface Design with React.js for E-commerce Websites"** (2020) by A. Hussain et al. - This paper explores best practices for designing user interfaces with React.js in the context of E-commerce websites. It discusses component-based architecture, responsive design principles, and performance optimization techniques.

**References**

* [React.js Official Documentation](https://react.dev/)
* [Express.js Official Website](https://expressjs.com/)
* [MongoDB Official Documentation](https://www.mongodb.com/docs/)
* [Node.js Official Website](https://nodejs.org/)
* [Tailwind CSS Official Documentation](https://tailwindcss.com/docs/)