A REPORT OF ONE MONTH TRAINING

at

STEP GNDEC

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Science and Engineering)

JUNE- JULY, 2025

SUBMITTED BY:

NAME : Prabhdeep Kaur
UNIVERSITY ROLL NO.(s) : 2435240

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

(An Autonomous College Under UGC ACT)

A REPORT OF ONE MONTH TRAINING

at

STEP GNDEC

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Science and Engineering)

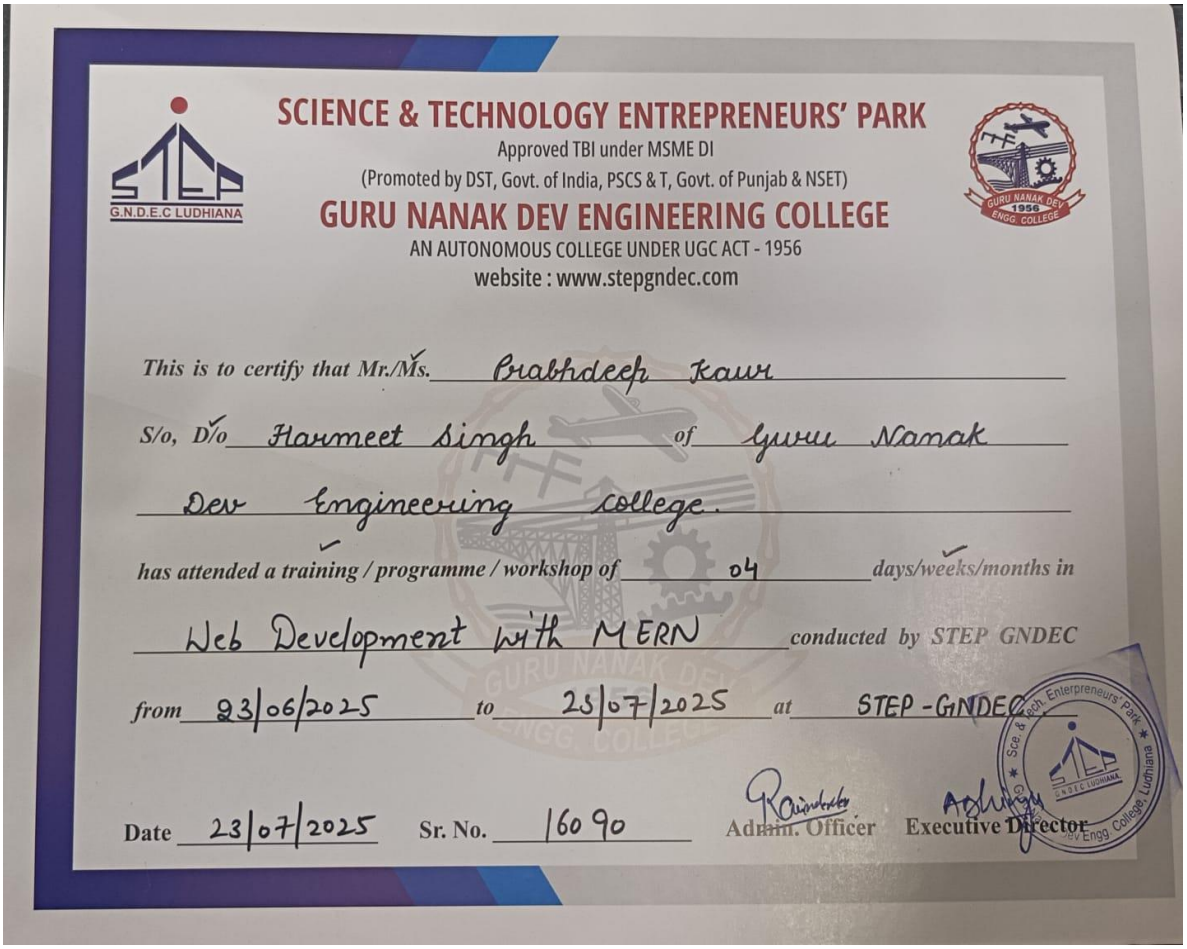
JUNE- JULY, 2025

SUBMITTED BY:

NAME : Prabhdeep Kaur
UNIVERSITY ROLL NO.(s) : 2435240

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING
GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA
(An Autonomous College Under UGC ACT)

CERTIFICATE BY INSTITUTE



SCIENCE & TECHNOLOGY ENTREPRENEURS' PARK
Approved TBI under MSME DI
(Promoted by DST, Govt. of India, PSCS & T, Govt. of Punjab & NSET)
GURU NANAK DEV ENGINEERING COLLEGE
AN AUTONOMOUS COLLEGE UNDER UGC ACT - 1956
website : www.stepgndec.com

This is to certify that Mr./Ms. Prabhdeep Kaur
S/o, D/o Harmeet Singh of Guru Nanak
Dev Engineering college.
has attended a training / programme / workshop of 04 days/weeks/months in
Web Development with MERN conducted by STEP GNDEC
from 23/06/2025 to 25/07/2025 at STEP-GNDEC

Date 23/07/2025 Sr. No. 1690 Admin. Officer Executive Director

GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA

CANDIDATE'S DECLARATION

I **PRABHDEEP KAUR**, hereby declare that I have undertaken one month training at **STEP GNDEC** during the period from **24 JUNE 2025** to **24 JULY 2025**, in partial fulfillment of the requirements for the award of the degree of **B.Tech (Computer Science Engineering)** at **Guru Nanak Dev Engineering College, Ludhiana**.

The work which is being presented in the training report submitted to the **Department of Electronics and Communication Engineering** at **Guru Nanak Dev Engineering College, Ludhiana**, is an **authentic record of training work** carried out by me.

Signature of the Student

The one month industrial training Viva–Voce Examination of_____has been held on _____and accepted.

Signature of Internal Examiner

Signature of External Examiner

ABSTRACT

During my one-month industrial training at STEP GNDEC, I gained hands-on experience in **Web Development with MERN STACK**, learning both **frontend and backend technologies**. The training focused on building a full-stack online medicine ordering platform, *MediKart*, using the **MERN Stack** — MongoDB, Express.js, React.js, and Node.js.

As part of the training, I learned and applied **HTML, CSS, JavaScript, and Bootstrap** to design responsive and user-friendly web pages. I developed the frontend using **React.js**, creating interactive pages such as Home, Products, Cart, Checkout, and My Orders. On the backend, I worked with **Node.js and Express.js**, learning to handle API requests, secure user authentication with JWT, and integrate a **MongoDB** database for managing users, products, orders, and addresses.

This project provided practical exposure to **state management in React**, **frontend-backend integration**, **database operations**, and **version control using Git and GitHub**. Through this training, I enhanced my understanding of building scalable, dynamic web applications, bridging the gap between theoretical knowledge and real-world web development skill.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to the management and mentors at STEP GNDEC for providing me the opportunity to undergo industrial training in web development. This training gave me practical exposure to full-stack development using the **MERN stack**, including **HTML, CSS, JavaScript, Bootstrap, React.js** and the basics of **MongoDB,Node.js, Express.js**,.

I would also like to express my sincere gratitude to the **Department of Computer Science and Engineering, Guru Nanak Dev Engineering College, Ludhiana**, for incorporating this industrial training as a valuable part of the curriculum. I am especially thankful to **Jaswant Sir** for his constant guidance and motivation, and to **Ms. Kiran Jyoti(HOD)** for her encouragement and support throughout my academic journey.

This training has given me hands-on experience in **web development, full-stack project implementation, and industry-standard workflows**, and I am truly grateful to everyone who contributed to making this learning experience meaningful and enriching.

ABOUT INSTITUTION

STEP GNDEC is an initiative by **Guru Nanak Dev Engineering College** to foster innovation, entrepreneurship, and practical learning in technology. The organization provides students with exposure to real-world projects, modern development tools, and industry-standard workflows, bridging the gap between academic learning and practical application.

STEP GNDEC focuses on nurturing talent in software development, web technologies, and entrepreneurial ventures. It provides a structured environment for students to work on live projects, gain hands-on experience in web development with MERN STACK, and develop skills that are crucial for the IT industry.

During my training at STEP GNDEC, I got the opportunity to work on the **MediKart project**, a full-stack web application for online medicine ordering. This exposure helped me understand project development cycles, database integration, frontend-backend communication, and best practices in software engineering.

Through such initiatives, **STEP GNDEC** continues to play a vital role in empowering students with technical knowledge, practical experience, and professional skills essential for real-world success.

TABLE OF CONTENTS

Topics **Page No.**

Certificate by Institute…….. **i**

Candidate's Declaration…… **ii**

Abstract……. **iii**

Acknowledgement…… **vi**

About the Institute……... **v**

Table of Content……... **vi**

CHAPTER 1: INTRODUCTION **9-22**

1.0 Foundational Concepts……10

1.1 Evolution of Web Technologies……..11

1.2 Importance of Frontend Development……12

1.3 Introduction to MERN Stack……...13

1.3.1 Overview of MERN (MongoDB, Express, React, Node)……14

1.3.2 Role of Each Component in Web Applications……...15

1.3.3 React.js as the Frontend Framework……15

1.4 Frontend Architecture and Component Planning……16

1.5 User Interface & User Experience Design……...17

1.6 Software, Tools, and Technologies Learned……18

1.7 Training Benefits and Learning Outcomes……...20

1.8 Report Structure Overview……...21

1.9 Summary……...22

CHAPTER 2: TRAINING WORK UNDERTAKEN **23-27**

2.1 Week1- Introduction to Web Development and Version Control …….....23

2.2 Week 2 – JavaScript and Server Environment Basics …….24

2.3 Week 3 – React.js and Backend Technologies ……....25

2.4 Week 4 – MERN Stack Project Development (MediKart) …….27

CHAPTER 3: RESULTS AND DISCUSSION **28-51**

3.1 Functional Outcomes of MediKart Application……...28

3.2 UI Components and Interaction Flow……..30

3.3 Technical Results and Backend Integration…….35

3.4 Discussion of Learning Outcomes……36

3.5 Summary……..40

CHAPTER 4: CONCLUSION AND FUTURE SCOPE

4.1 Conclusion……....46

4.2 Future Scope…….47

REFERENCES **48**

CHAPTER 1
INTRODUCTION

Today, online platforms are widely used across almost every field. People prefer online services because they are easy to access, save time, and deliver quick results. Therefore, websites need to be simple, secure, and user-friendly.

During my web development training at STEP GNDEC, I worked on a project named MediKart. MediKart is a full-stack web application that allows users to browse medicines, add them to the cart, and place orders online. The main purpose of this project was to understand how a real-world web application works using both frontend and backend technologies.

The project focused on two aspects. First, creating a responsive and easy-to-use frontend so that users can search for products and place orders without difficulty. Second, understanding backend operations such as user login, order handling, and data storage. Features like authentication, product listing, cart system, and order confirmation were implemented step by step during the training.

Through this training, I learned the following technologies:

- React.js for frontend development
- Node.js and Express.js for backend APIs
- MongoDB for storing application data
- GitHub for version control

Initially, I faced difficulty connecting the frontend with the backend, but with practice and guidance, I was able to understand the application flow. This project helped me gain practical knowledge of full-stack web development and improved my understanding of how frontend and backend work together in a real application .

1.0 Foundational Concepts

During my web development training at **STEP GNDEC**, I learned the basic technologies required to develop a full-stack web application. The training helped me understand both frontend and backend development, which I applied while working on my project **MediKart**.

The main technologies and tools that I used during this training are listed below:

HTML and CSS were used to understand basic webpage structure and styling, which helped in writing JSX and designing layouts in React.

JavaScript was used to implement application logic and handle user interactions in the frontend.

React.js was used to develop the frontend of MediKart, including pages like product listing, cart, checkout, and user authentication.

Node.js and Express.js were used to create backend APIs for user login, product management, cart handling, and order placement.

MongoDB was used as the database to store user details, products, orders, and addresses.

GitHub was used for version control to manage project code and track changes during development.

Initially, I found it difficult to connect the frontend with backend APIs and manage application state. However, after practice and debugging, I was able to understand the complete flow of the application. This training helped me move from basic web development concepts to building a complete MERN stack project and improved my understanding of how frontend and backend work together.

1.1 Evolution of Web Technologies

Here is a shortened, simple, and student-written version with the same meaning:Web technologies have evolved from simple static websites to modern dynamic applications. Earlier, websites were built using only HTML and CSS with fixed content and no user interaction. Later, server-side technologies made websites dynamic by allowing data processing and features like login and database connectivity. The introduction of JavaScript added interactivity, improving user experience without reloading pages. Today, modern frameworks like React.js and full-stack technologies are used to build fast and efficient applications, making projects like **MediKart** possible through smooth frontend and backend integration.

1.2 Importance of Frontend Development

The frontend of a web application is the part that users interact with directly. In MediKart, the frontend was developed using React.js, which helped create a dynamic and interactive user interface.

Key frontend features I implemented in MediKart include:

- **Product Browsing and Cart:** Users can view available medicines, add products to the cart, and remove items easily.
- **User Authentication:** Login and registration pages were created using React forms.
- **Dynamic Updates:** Cart items and order status update in real-time using React state management.
- **Responsive Design:** The layout adapts to different screen sizes using Tailwind CSS.

The frontend communicates with the backend through APIs built with Node.js and Express, allowing users to fetch product data, place orders, and manage their accounts.

Through this work, I gained hands-on experience in building responsive React components, managing state, and integrating frontend with backend services to make MediKart fully functional and user-friendly.

1.3 Introduction to MERN Stack

During my web development training, I learned the MERN stack (MongoDB, Express.js, React.js, Node.js) and applied it to the MediKart project. The MERN stack allows building a complete web application using JavaScript for frontend, backend, and database.

1.3.1 Technologies Used

- **MongoDB (Database):**
  Used to store user details, products, orders, and addresses. I learned to create collections, insert data, and perform queries for the project.
- **Express.js (Backend Framework):**
  Used to create APIs for MediKart, such as product listing, cart management, user authentication, and order processing.

- **React.js (Frontend Library):**
  Used to build the user interface, including product pages, cart, checkout, login, and profile management. React components and hooks (useState, useEffect) were used to manage state and user interactions.
- **Node.js (Runtime Environment):**
  Run backend JavaScript code and connected the frontend with MongoDB through Express APIs.

1.3.2 Role of Each Component in MediKart

- MongoDB: Stores and retrieves all application data.
- Express.js: Handles backend logic and API requests.
- React.js: Displays products, manages cart and orders, and handles user interactions.
- Node.js: Runs backend code and connects frontend with the database.

Summary:
Using the MERN stack, I was able to develop a fully functional MediKart e-commerce platform. This training gave me hands-on experience in integrating frontend, backend, and database to build a complete web application.

1.4 Frontend Architecture and Component Planning

During my training, I developed **MediKart** using **React.js** with a component-based architecture, which made the frontend modular and easier to manage.

Key Points of MediKart Frontend

- **Component Structure:**
  Each part of the UI, like Navbar, ProductCard, Cart, CheckoutPage, and Footer, was created as a separate React component. This made it easier to reuse code and update features.
- **Page Organization:**
  MediKart has organized pages such as Home, Product Category, Product Details, Cart, Checkout, My Orders, Seller Dashboard, and Admin. This helped in managing navigation

and state efficiently.

- **State Management:**
  React hooks like useState, useEffect, and useContext were used to manage state. Shared data such as user information, cart items, and product listings were handled through AppContext.
- **Routing and Navigation:**
  React Router was used to navigate between pages. Public and protected routes ensured that users and sellers could access only the pages meant for them.
- **Integration with Backend APIs:**
  Frontend components communicated with Node.js/Express APIs to fetch product data, manage cart items, place orders, and handle user authentication.
- **Responsive Layout:**
  Pages were styled with Tailwind CSS to make them look good on mobile, tablet, and desktop devices.

Through this architecture, I learned how to build a modular and functional frontend, manage state efficiently, and connect the interface with backend APIs to make MediKart work smoothly.

1.5 User Interface & User Experience Design

A smooth **user interface (UI)** and good **user experience (UX)** are important in MediKart, where users browse products, manage their cart, and place orders easily.

UI Design in MediKart

- **Consistency:** Same color scheme and design style used across all pages.
- **Simplicity:** Minimalistic and easy-to-navigate interface for product browsing and checkout.
- **Responsiveness:** Layout adapts to different screen sizes using Tailwind CSS, making MediKart usable on desktops, tablets, and mobiles.

UX Features Implemented

- **Intuitive Navigation:** Clear menus, categories, search bar, and product listings for quick access.
- **Interactive Components:** Product cards, cart, and forms respond to user actions in real-

time.

- **Checkout Flow:** Step-by-step process for adding addresses, selecting payment options, and confirming orders.
- **User Authentication:** Secure login and registration to protect user data.
- **Feedback Mechanisms:** Notifications and alerts for successful actions or invalid inputs.

Technologies Used

- **React.js:** Built reusable components for product cards, navigation, cart, checkout, and dashboards.
- **Tailwind CSS:** Styled pages and components, ensuring responsive design.

This approach ensured that MediKart is easy to use, interactive, and visually consistent, providing a smooth experience for users.

1.6 Software, Tools, and Technologies Learned

During my MERN stack training, I learned various web development tools and technologies, which I applied in building **MediKart**.

Frontend

- **React.js & JSX:** Built reusable components for Navbar, product cards, cart, checkout, dashboards, and order management. Managed state using React Hooks (useState, useEffect) for live cart updates, order tracking, and user sessions.
- **Tailwind CSS:** Used for styling responsive and modern UI components across all pages, including home, product listings, cart, checkout, and dashboards.

Backend & Database

- **Node.js:** Server-side runtime to handle requests and run backend code.
- **Express.js:** Created RESTful APIs for managing products, users, orders, and authentication.
- **MongoDB:** Stored and retrieved data for users, products, carts, orders, and addresses.
- **npm:** Managed project dependencies such as React Router, Axios, and bcrypt.

Version Control & Development Tools

- **GitHub:** Tracked code changes, managed versions, and collaborated on project development.
- **VS Code:** Used as the main IDE for coding, debugging, and project management.
- **Browser Developer Tools:** Debugged frontend functionality and monitored API calls.

Other Knowledge

- **HTML & CSS basics:** Though not used directly, helped in understanding JSX and component styling in React.
- **Bootstrap basics:** Not used in MediKart, but helped understand modern CSS frameworks like Tailwind CSS.

1.7 Training Benefits and Learning Outcomes

The MERN stack training at **STEP GNDEC** significantly helped me develop both practical and theoretical skills, enabling me to successfully create the **MediKart** project. The key learning outcomes are as follows:

- **1.7.1 Application of Knowledge to Practice**
  I applied theoretical concepts of web development directly to MediKart. This included integrating frontend with backend APIs and managing database operations.
- **1.7.2 Understanding Full-Stack Workflow**
  I gained a clear understanding of how React interacts with Node.js/Express APIs and MongoDB, forming a complete full-stack workflow for MediKart.
- **1.7.3 Problem-Solving and Logical Thinking**
  Developing MediKart involved challenges such as managing product data, handling cart and orders, and implementing user authentication, which improved my problem-solving skills.
- **1.7.4 Project Planning and Component Structuring**
  I learned to plan frontend components, backend routes, and database schemas, enabling an organized and modular development process.

1.8 Report Structure Overview

The report is structured to provide a clear and comprehensive understanding of the web development training with the MERN stack and the development of the MediKart project. Each chapter focuses on a specific aspect of the training and project work:

- **Chapter 1 – Introduction:**
  Covers the background of the training, theoretical concepts, MERN stack fundamentals, software and tools learned, and key learning outcomes.
- **Chapter 2 – Training Work Undertaken:**

Details the sequential learning process, methodology followed, and hands-on project development activities for MediKart. Includes weekly progress, component development, and backend/database integration.

- **Chapter 3 – Results and Discussion:**
  Presents the outcomes of the MediKart project, including frontend functionalities, UI interactions, backend operations, and database performance. Discusses challenges faced and solutions implemented during development.
- **Chapter 4 – Conclusion and Future Scope:**
  Summarizes the overall learning experience, project achievements, and potential enhancements or future developments for MediKart.
- **References and Appendix:**
  Includes the sources consulted during the training and additional information such as code snippets, diagrams, or configuration details related to MediKart.

This structure ensures a logical flow from theoretical knowledge to practical application, highlighting the skills and understanding gained through the training.

1.9 Summary

The web development training with the MERN stack provided a solid foundation in both frontend and backend development, along with database management and version control. Through this training:

- I gained theoretical knowledge and practical experience in **React.js, Node.js, Express.js, and MongoDB**.
- Learned to plan and develop frontend components and integrate them with backend APIs.
- Acquired skills in state management, user interface design, and responsive layouts using Tailwind CSS in MediKart.
- Built a complete full-stack application by combining frontend, backend, and database workflows.
- Learned to use GitHub for version control and collaborative project management.

Overall, the training bridged the gap between theory and practice, equipping me with the skills to develop a professional web application like **MediKart**, while understanding the complete full-stack development lifecycle.

CHAPTER 2
TRAINING WORK UNDERTAKEN

Throughout the training, I worked on developing the project in a step-by-step manner, focusing on both frontend implementation with React and backend basics using Node.js, Express.js, and MongoDB.
The training was divided into four weeks, each covering specific skills and deliverables.

The MERN stack training at **STEP GNDEC** provided structured, hands-on learning in web development, covering both frontend and backend development. The focus was on building **MediKart**, an online medicine ordering platform, allowing me to apply concepts learned in a real-world project.

Training Overview

The training was conducted over four weeks, each week focusing on specific skills and project deliverables:

- **Version Control with GitHub:**
  - Created repositories, managed branches, and committed changes efficiently.
  - Practiced collaborative workflows to track project progress and maintain code history.
- **Frontend Development with React.js:**
  - Built dynamic and reusable components for pages like product listing, shopping cart, checkout, and user dashboards.
  - Learned component structuring, state management (useState, useEffect, useContext), and routing with React Router.
- **Backend Basics with Node.js and Express.js:**
  - Developed RESTful APIs to handle users, products, carts, and orders.
  - Integrated backend APIs with the React frontend for dynamic data display and real-time updates.
- **Database Integration with MongoDB:**
  - Managed collections for users, products, carts, orders, and addresses.
  - Performed CRUD operations to store and retrieve data efficiently for MediKart.

- **Development Environment Setup:**
    - Configured VS Code, Node.js, npm, and created the React project folder structure.
    - Prepared the environment for seamless frontend-backend integration.

2.1 Week 1 – Introduction to Web Development and Version Control

During the first week of training, I was introduced to the basics of web development and full stack technologies. I learned what the MERN stack is and how it is used to build modern web applications. This week helped me understand the overall flow of a website, from frontend design to backend support.

I started by learning HTML, which is used to create the structure of web pages. I practiced creating pages using headings, paragraphs, lists, forms, and images. After that, I learned CSS, which helped me style the web pages and improve their appearance. I worked on layouts, colors, fonts, spacing, and basic responsiveness.

Along with frontend basics, I learned Git and GitHub, which are important tools for version control. I understood how to create repositories, upload projects, commit changes, and manage different versions of code. I also learned about forking and branching, which are useful when working on projects in a team.

Later in the week, I learned Bootstrap, a CSS framework that makes web pages responsive and attractive. Using Bootstrap components like buttons, cards, navigation bars, and grids, I created a small responsive project.

Outcome:

This week helped me build a strong foundation in HTML, CSS, Bootstrap, and GitHub. I became comfortable with basic frontend design and understood how to manage code properly.

2.2 Week 2 – JavaScript and Server Environment Basics

In the second week, the focus was on JavaScript, which is used to add functionality and logic to web applications. I learned basic programming concepts such as variables, data types, operators, conditions, loops, functions, arrays, and objects.

I practiced writing JavaScript programs to understand how logic works in web development. This helped me learn how data is processed and how user actions can be handled in a program.

I also learned about XAMPP and understood how a local server environment works. This gave me

a basic idea of how websites run on servers. Along with this, I was introduced to Drupal, where I learned about content management systems and how to set up a project locally.

Outcome:

This week improved my problem-solving skills and helped me understand the role of JavaScript and servers in web development.

2.3 Week 3 – React.js and Backend Technologies

The third week focused on React.js, which is a JavaScript library used for building user interfaces. I learned how to set up a React environment and understood concepts like components, JSX, and component-based structure.

I learned how to create reusable components and manage application data using React Hooks. This helped me understand how modern frontend applications work and how data changes are handled efficiently.

Along with React, I was introduced to backend technologies. I learned Node.js, which is used to run JavaScript on the server side, and Express.js, which helps in creating APIs. I also learned MongoDB, a NoSQL database used to store application data.

I understood how frontend, backend, and database communicate with each other in a MERN stack application.

Outcome:

This week gave me a clear understanding of full-stack development and prepared me for working on a real project.

2.4 Week 4 – MERN Stack Project Development (MediKart)

The fourth week was fully dedicated to developing my MERN stack project MediKart, an online medicine ordering platform. In this week, I applied all the concepts learned during training in a practical way.

I developed the frontend using React.js and created components such as Navbar, Product Listing, Product Card, Cart, Checkout Page, Address Form, and Order History. I used React Hooks like useState for managing component data and useEffect for fetching data from the backend. React Router was used to navigate between different pages smoothly.

For global data sharing, I used Context API, which helped manage user login information and cart data across multiple components without passing props repeatedly.

On the backend side, I used Node.js and Express.js to create APIs for user registration, login, product listing, cart operations, and order placement. I implemented JWT authentication to ensure secure login and protected routes.

I connected the backend with MongoDB, where all user data, medicine details, cart items, addresses, and order information were stored. I used Axios to connect the frontend with backend APIs and handle responses properly.

Throughout the project, I tested different features, fixed errors, and improved the structure of the application. GitHub was used regularly to save progress and manage project versions.

Outcome:

The MediKart project was successfully completed as a fully functional MERN stack application. Users can register, log in, browse medicines, add items to the cart, place orders, and view their order history.

Conclusion:

This four-week MERN stack training helped me understand web development from basic concepts to complete project development. Working on the MediKart project gave me real-world experience in frontend, backend, and database integration. The training improved my technical skills, confidence, and understanding of full-stack web development.

CHAPTER 3

RESULTS AND DISCUSSION

This chapter presents the implementation outcomes, functionalities, and discussions based on the MediKart project developed during my MERN stack training at STEP GNDEC. MediKart is a full-stack online medicine ordering platform, designed to provide users with a secure and convenient way to order medicines while allowing sellers to manage products efficiently.

The project involved frontend design, backend API development, MongoDB integration, and testing to create a complete e-commerce solution.

3.1 Functional Outcomes of MediKart

MediKart successfully achieved the project objectives, offering both user and seller functionalities for a comprehensive e-commerce experience.

1. User Module

- **Authentication and Authorization:**
  Secure signup and login using JWT tokens. Each session is authenticated to protect user data.
- **Product Browsing and Searching:**
  Users can view medicines fetched from MongoDB and explore products by categories.
- **Cart and Checkout:**
  Users can add, update, or remove items. Cart data persists during navigation.
  Checkout allows adding delivery addresses and confirming orders.
- **Order Placement and Tracking:**
  Orders are stored in MongoDB and displayed in the My Orders section for tracking.

2. Seller Module

- **Seller Login and Authentication:**
  Secure login via a dedicated portal using JWT authentication.
- **Product Management:**
  Sellers can add, edit, or delete products. Product data is handled via Express.js APIs.
- **Order Management:**
  Sellers can view all customer orders for their products, helping manage inventory and

deliveries.

3. Application-Wide Features

- **Dynamic API Integration:**
  Frontend uses **Axios** to fetch and display real-time data from backend APIs.
- **Data Validation:**
  Inputs in forms (registration, address, product addition) are validated on both frontend and backend.
- **Error Handling:**
  Displays appropriate messages for invalid inputs, authentication errors, or network issues.

Result:

By the end of the project, MediKart operated as a fully functional MERN stack e-commerce platform, enabling users to browse, purchase, and track medicines, while sellers managed products and orders efficiently.

3.2 UI Components and Interaction Flow

The frontend of MediKart was built using React.js for dynamic components, Tailwind CSS for responsive design, and React Router DOM for smooth single-page navigation. The interface was designed to be intuitive and user-friendly, with seamless updates from backend APIs.

Key UI Components

- **Navbar:** Provides navigation links to Home, Categories, Cart, and Dashboard.
- **Product Cards:** Display product details dynamically with Add to Cart functionality.
- **Cart Page:** Shows selected products with options to update quantity or remove items.
- **Checkout Page:** Collects delivery details, confirms order, and displays total price.
- **User Dashboard:** Allows users to view past orders and track deliveries.
- **Seller Dashboard:** Enables product management and viewing customer orders.

The following subsections describe each major UI component in logical order of user interaction, along with recommended screenshot placements.

1. Navigation and Layout

The **layout components** define the overall structure of the application and are consistent across all pages.

- **Navbar:**
    - Provides quick access to Home, Products, Cart, My Orders, and Login/Signup pages.
    - Dynamically updates based on login status:
        - Shows user options if a user is logged in.
        - Shows seller options if a seller is logged in.
    - Ensures a consistent navigation experience throughout the site.

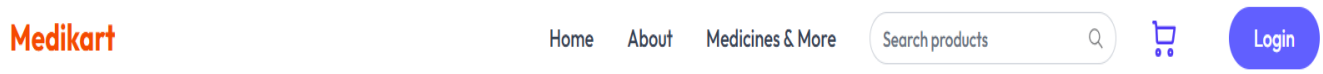


Figure 1.1- Navbar

- **Footer:**
    - Displays contact information, newsletter subscription, and policy links.
    - Provides additional navigation for users who scroll to the bottom of the page.

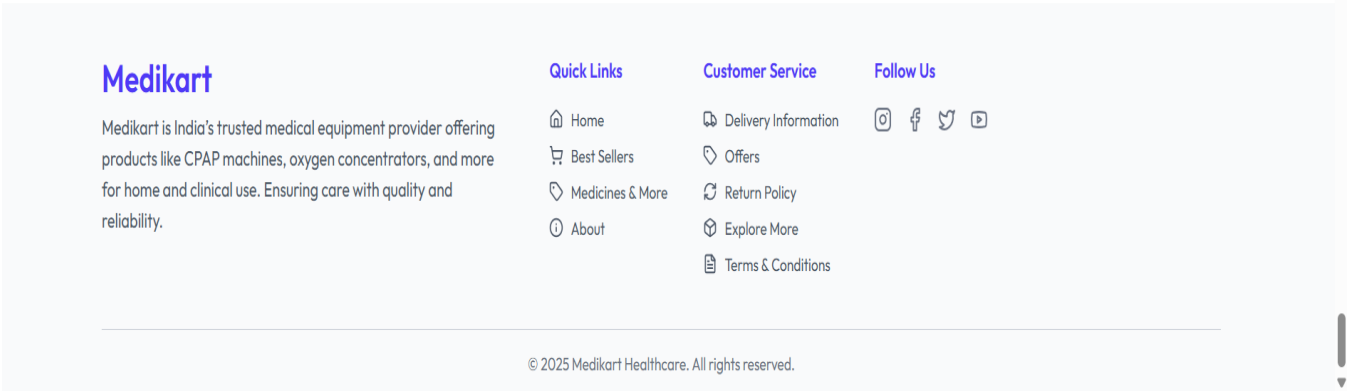


Figure 1.2 - Footer

2. Home Page

The Home Page serves as the first impression of the platform and introduces users to the services offered.

- **Hero Section:**
    - Highlights the platform with promotional banners and call-to-action buttons like *"Shop Now"*.
    - Sets the visual theme for the website.

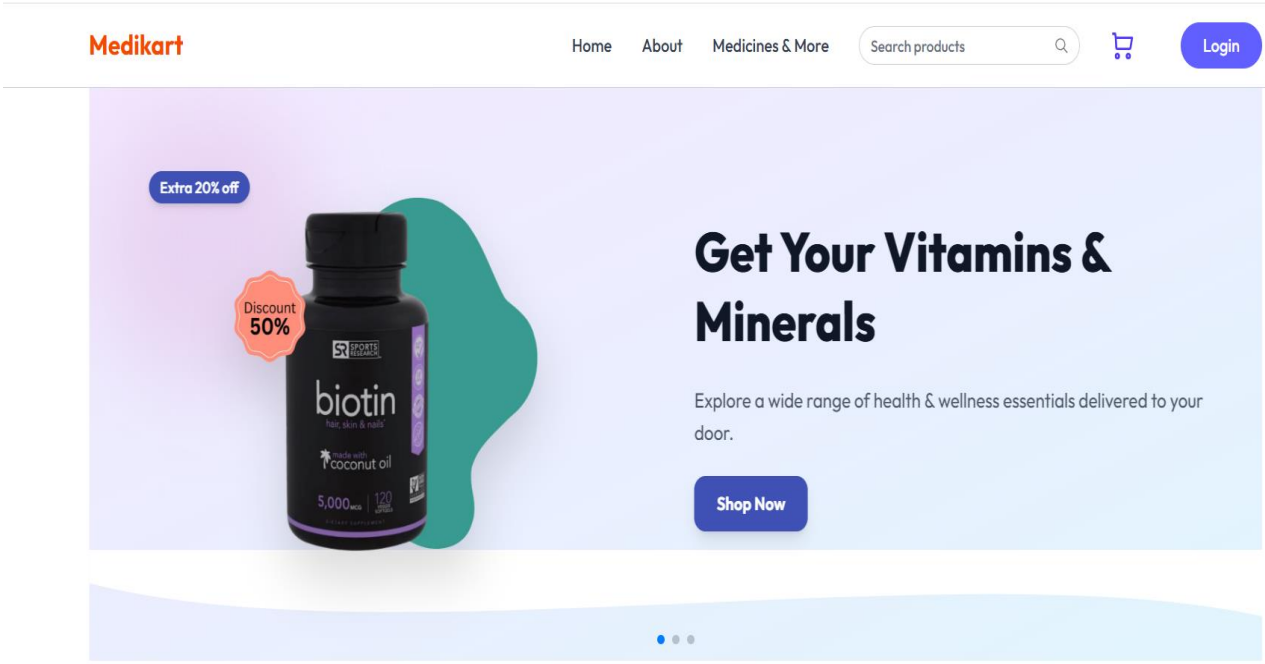
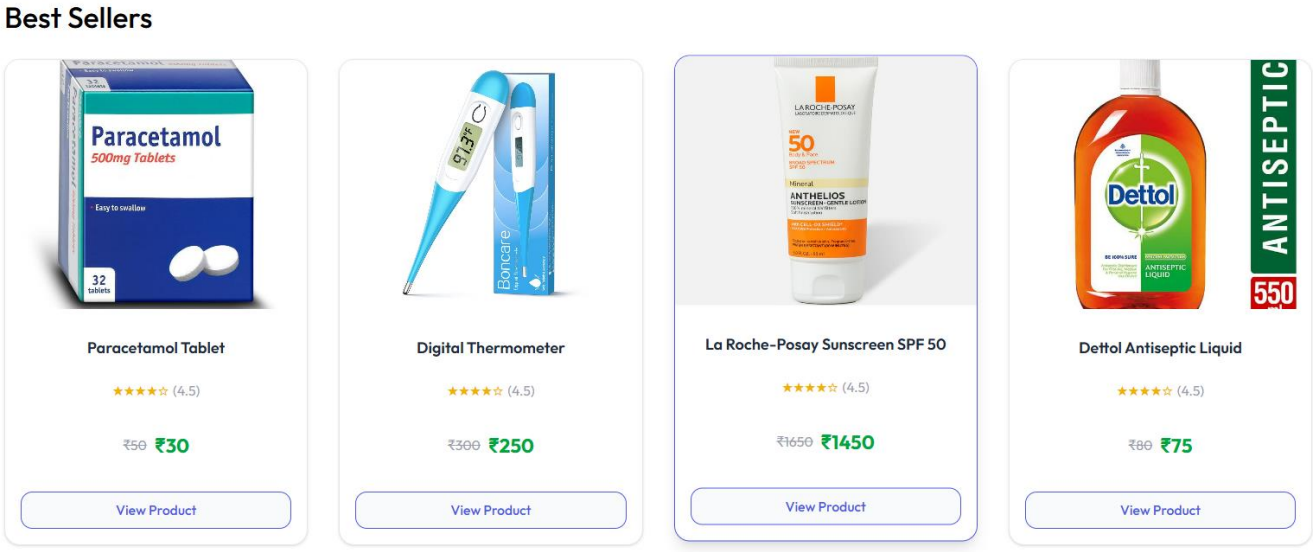


Figure: 1.3 – Hero section

- **Best Seller Section:**
  - Showcases popular medicines with real-time data fetched from MongoDB.
  - Encourages users to quickly access frequently purchased items.



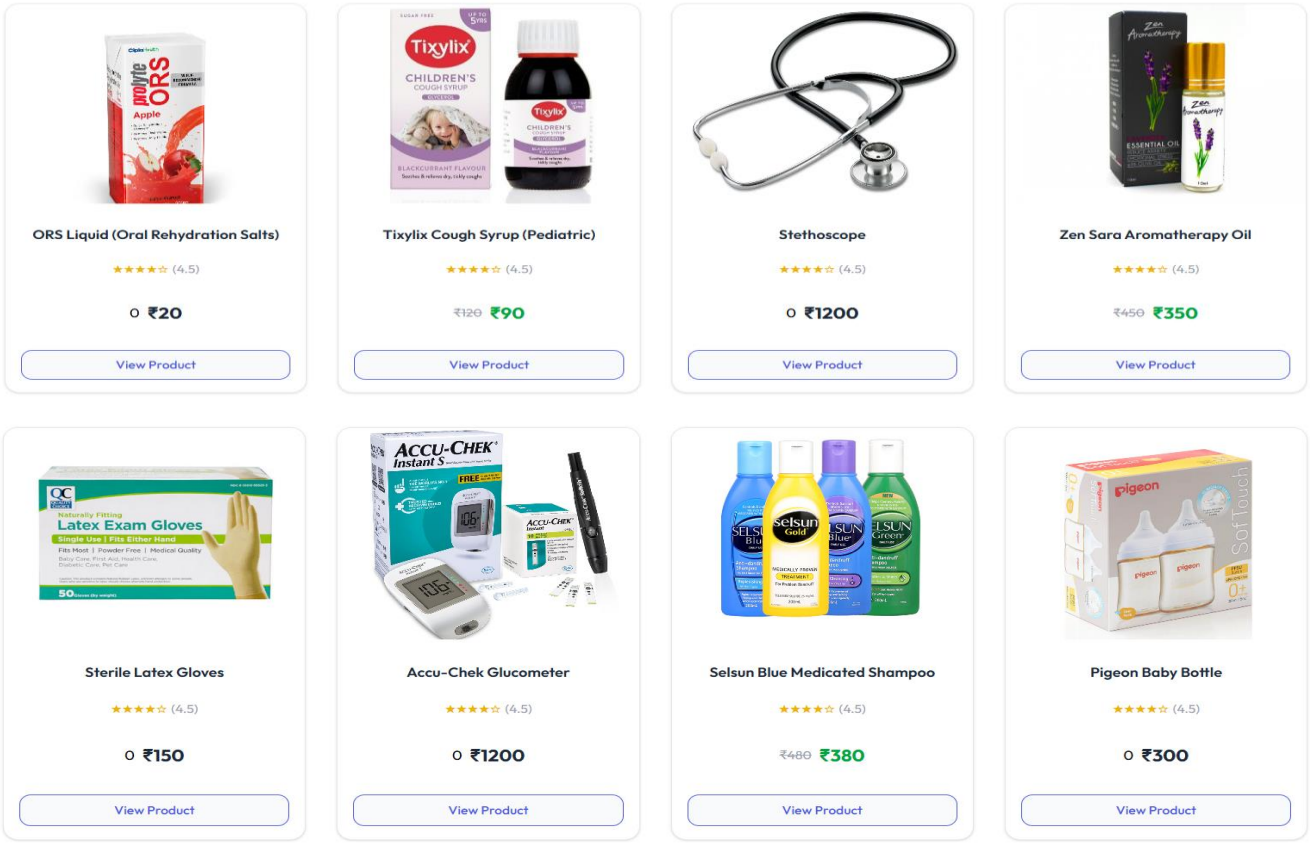


Figure 1.4 – Best Seller

- **Newsletter Subscription Section:**
    - This section of the website is designed to encourage users to subscribe to the platform's newsletter.

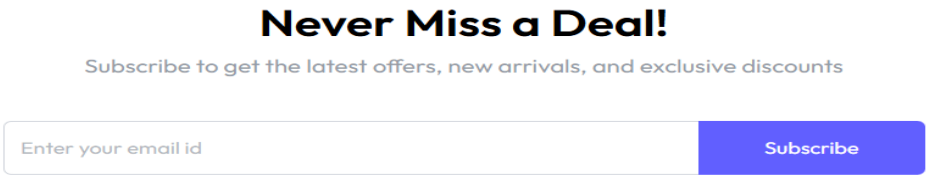


Figure 1.5 – newsletter Subscription section

- **Category Section:**
    - Organizes products into categories such as Tablets, Syrups, Supplements, and Personal Care.

- Allows users to filter products easily based on type.

Explore Our Categories

Medicines

Buy prescription & OTC medicines safely and conveniently.

Wellness

Products to support your mental and physical well-being.

Personal Care

Skincare, haircare & hygiene essentials for everyday health.

Health Devices

BP monitors, oximeters, thermometers and more.

Baby Care

Gentle and safe products for your baby's needs.

Skin Care

Nourish and protect your skin with trusted solutions.

Diabetic Care

Essential tools and supplies for diabetic support.

Fitness & Supplements

Nutrition and gear to power your health journey.

Ayurveda

Explore Ayurvedic remedies for holistic healing.

Medical Equipments

From nebulizers to walkers — essential devices for care.

Elderly Care

Adult diapers, support belts & more for senior comfort.

Surgical Supplies

Sterile instruments and surgical products for clinics & homes.

Figure 1.6 – category section

- **Feature & Mission Sections:**
    - Communicates the core values, safety standards, and benefits of using MediKart.

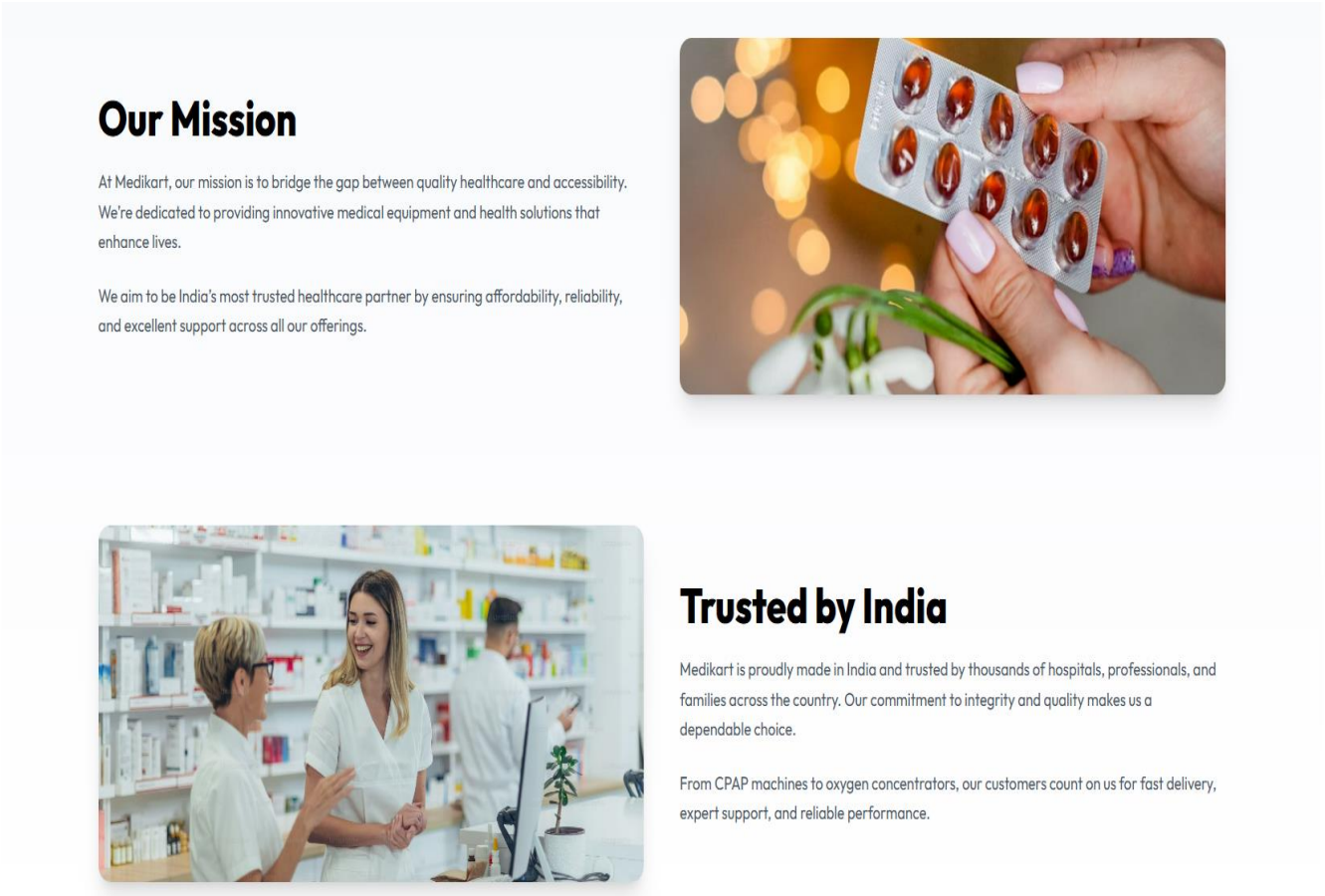


Figure 1.7 – Feature & Mission section

- **FAQ (Frequently Asked Questions) Section:**
    - The FAQ section is intended to address common queries related to the system, provide guidance to users, and improve overall usability and understanding of the application.

Figure 1.8 – Frequently asked questions

3. Product Pages

These pages are central to the shopping experience, combining frontend display and backend data interaction.

- **Product Listing Page:**
    - Dynamically fetches all products from the backend API using **Axios**.
    - Displays products in a grid layout using the reusable ProductCard component.
    - Allows sorting and searching for better accessibility.

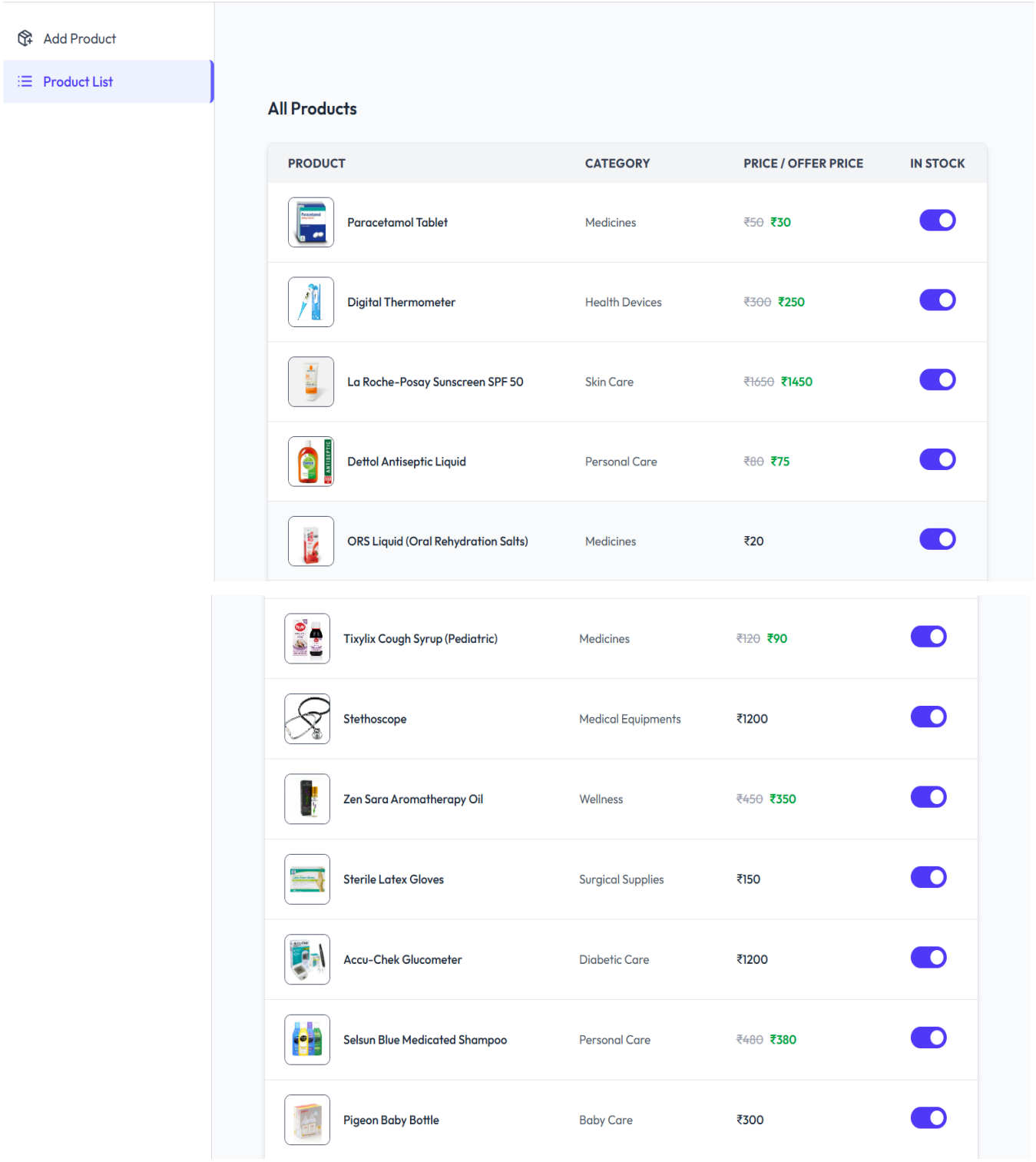


Figure 2.0 – Product Listing Page

- **Product Details Page:**
    - Shows detailed information about a selected product:
        - Product images
        - Name and description
        - Price and available quantity
    - Includes an Add to Cart button, which updates the cart in real time.

Figure 2.1 – Product Listing Page

4. Cart and Checkout

The Cart and Checkout system integrates user selections with backend data storage to maintain order integrity.

- **Cart Page:**
    - Displays selected products with quantity adjustment options and total price updates dynamically.
    - Users can remove items before proceeding to checkout.

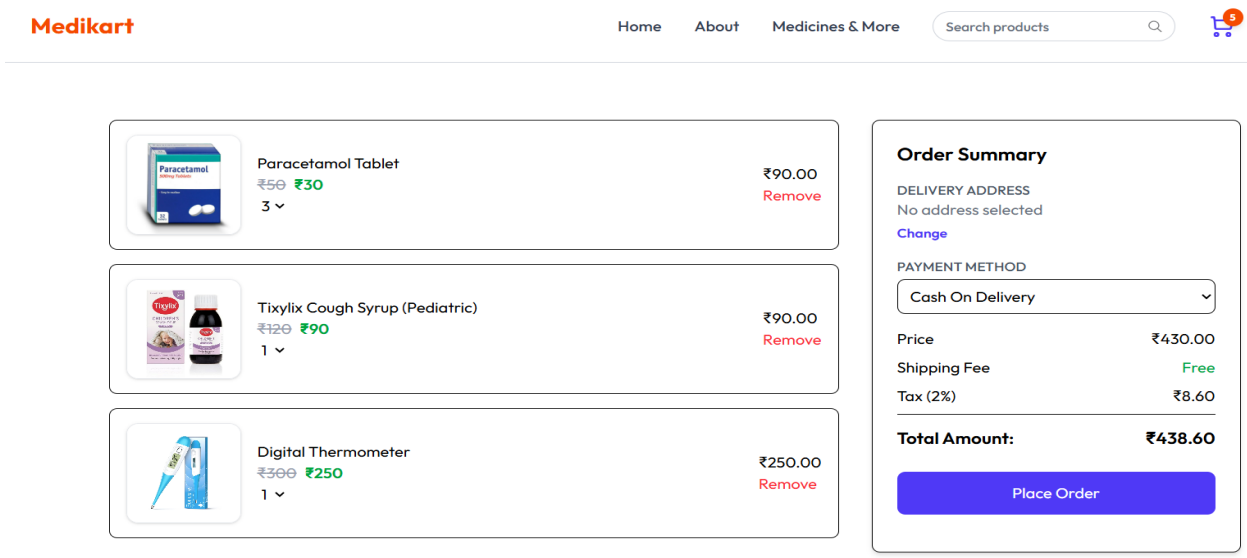


Figure 3.1 – cart & checkout

- **Add Address Page:**
  - Allows users to save new addresses directly into the database using backend APIs.
  - Ensures smooth integration between the frontend form and the backend server.

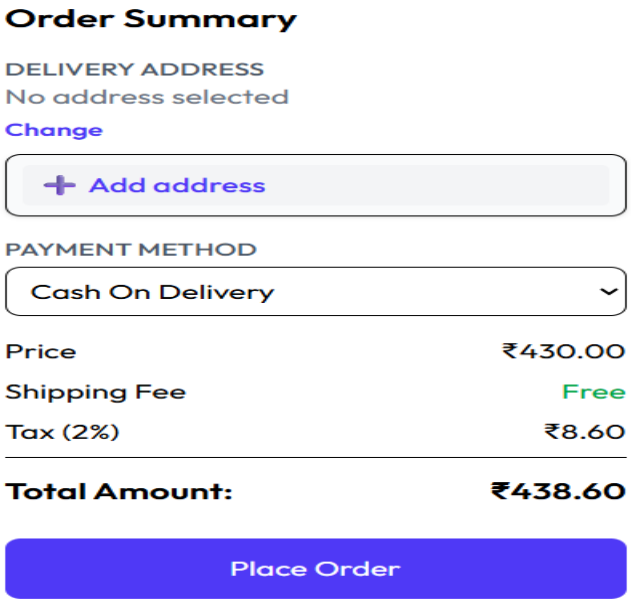


Figure 3.2 – Add address

Clicking on add address button :

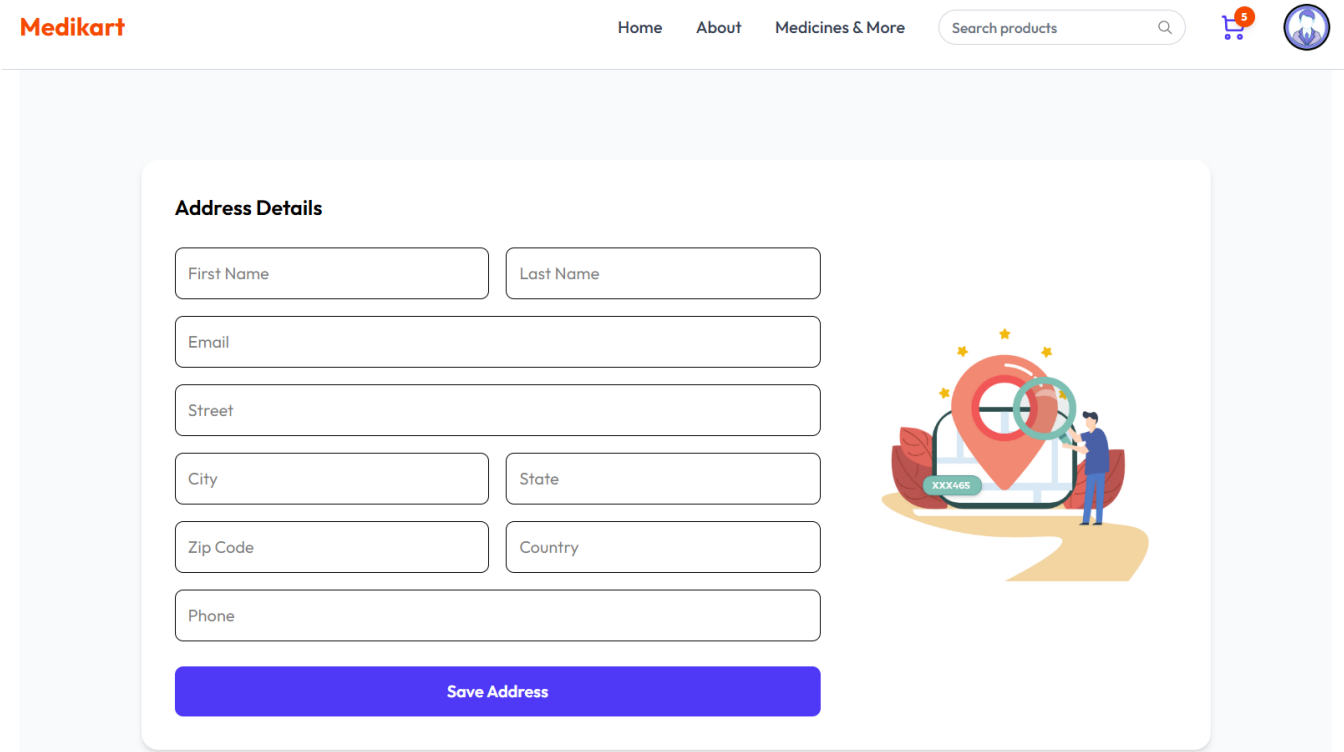
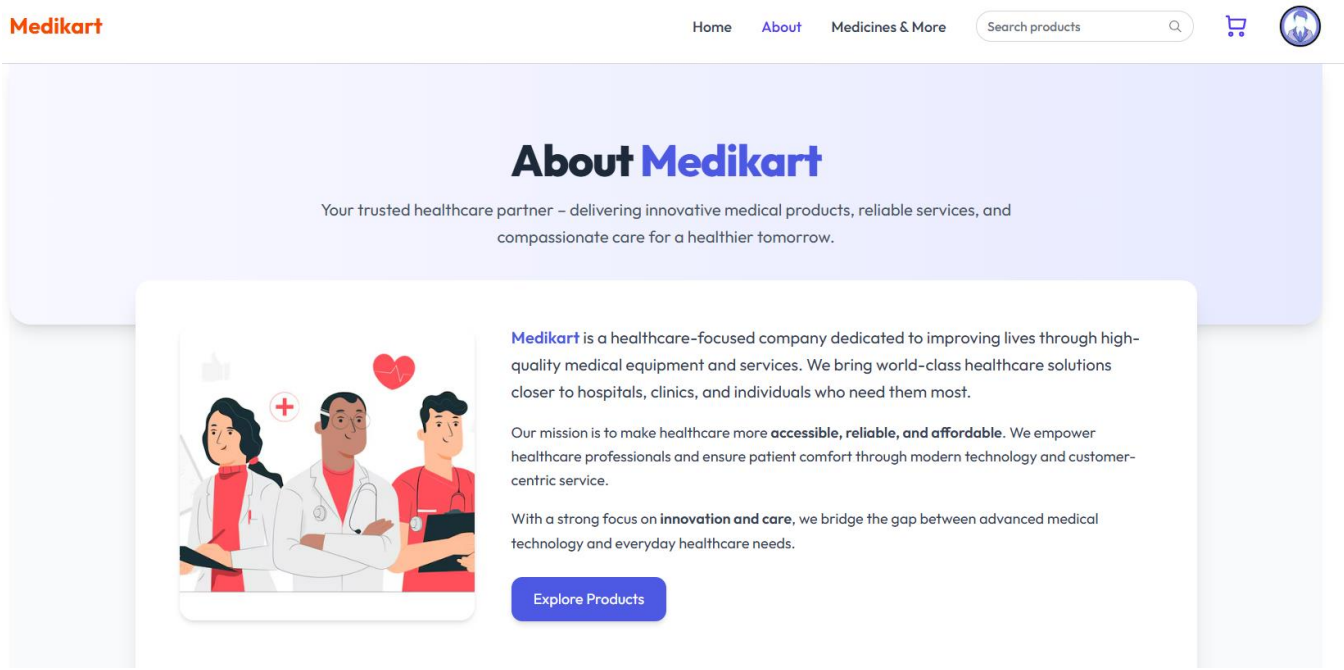


Figure 3.3 – Add address

About Page

- Provides information about MediKart, its vision, services, and team.
- May include contact details and links to social media.



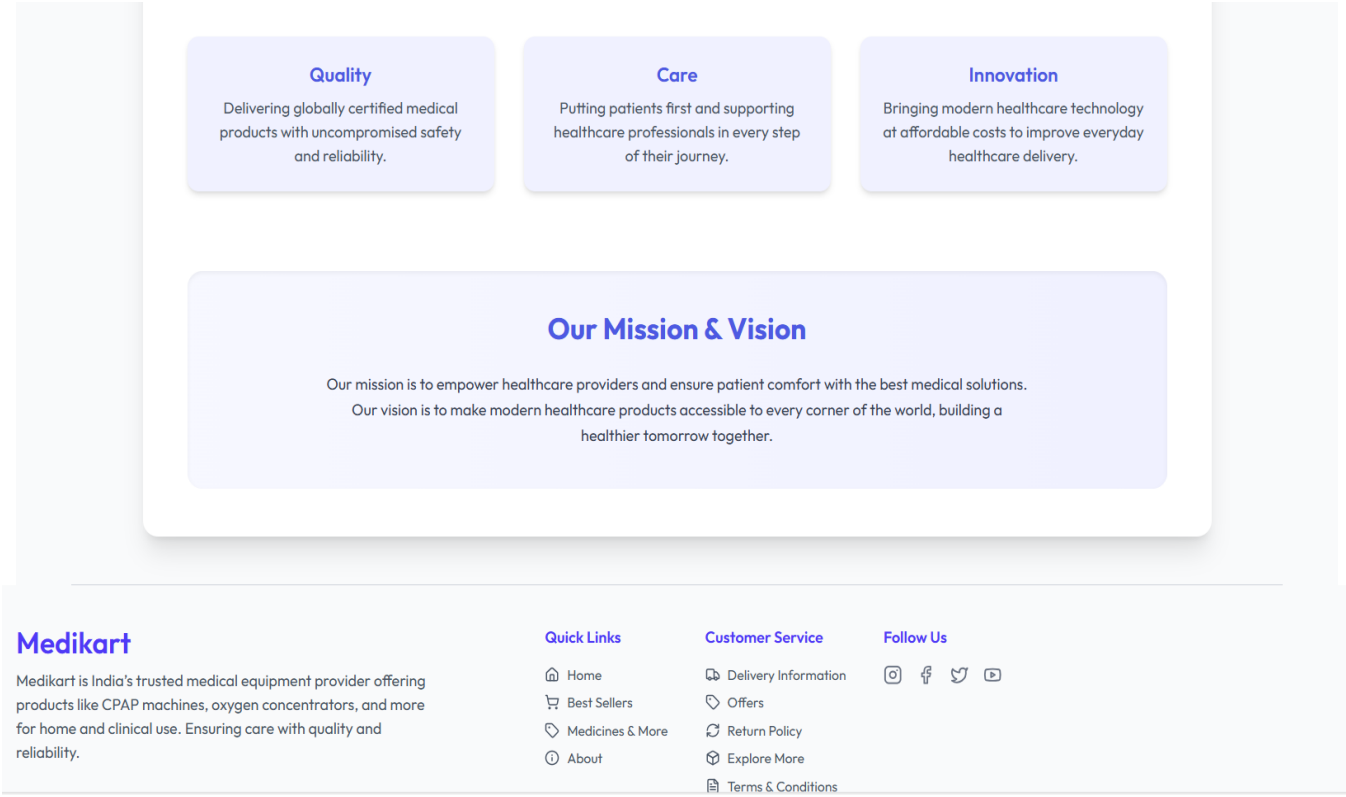


Figure 4.1 – About Page

Medicines & More

- Displays all medicines in categories like Medicines, Skin Care, Health Care and Personal Care etc.
- Users can search and filter products by name, type, or price.
- Clicking a product opens its details page for more information**.**

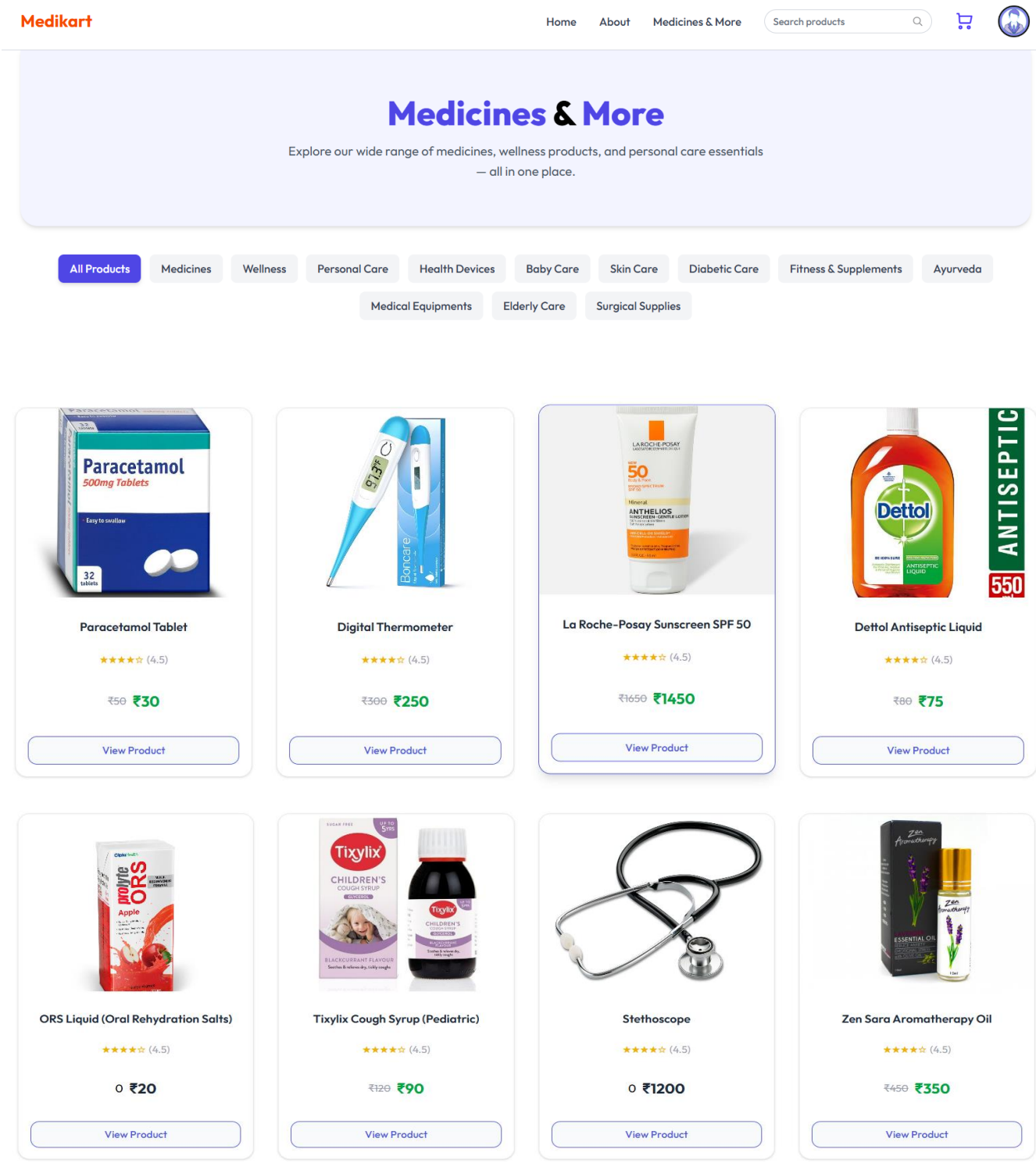


Figure 5.1 – Medicines & More

Search Functionality

- Allows users to quickly find products by typing keywords.
- Integrated with backend APIs to fetch live product data.

- Improves user experience by making product discovery faster.

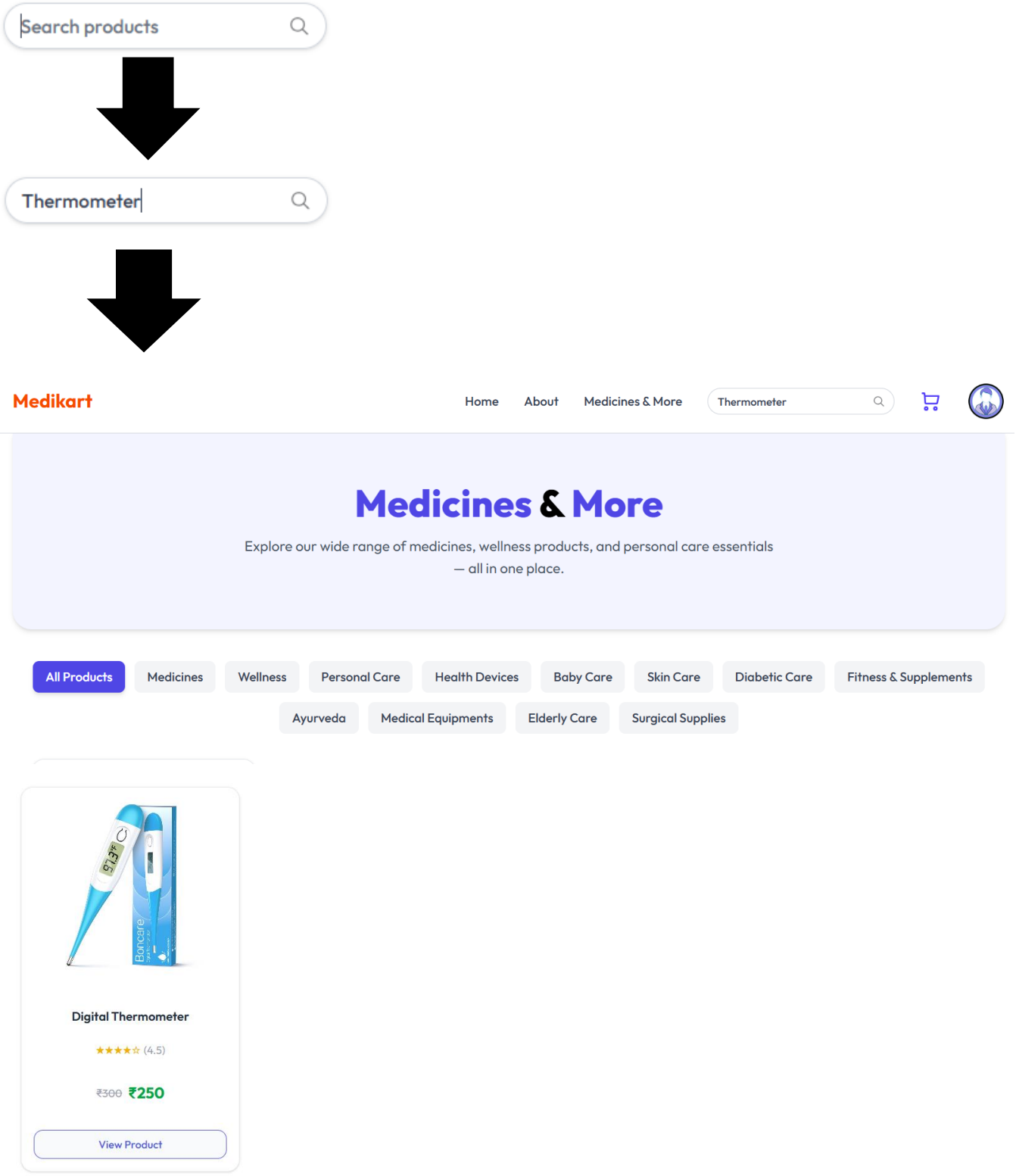
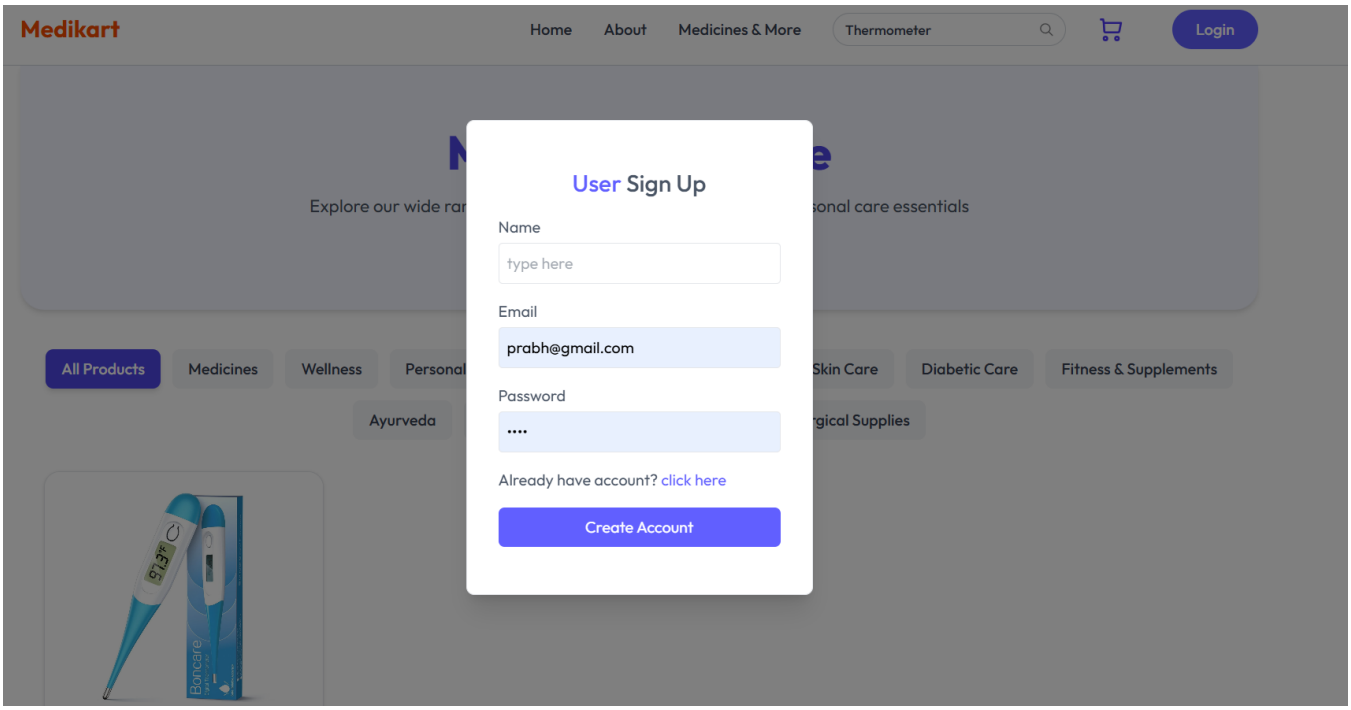


Figure 5.2 – Search Functionality

Login and Register User

- Login allows existing users to access their accounts.
- Register lets new users create an account with email and password.
- Authentication is connected to the backend for secure data storage.

Register:



Login:

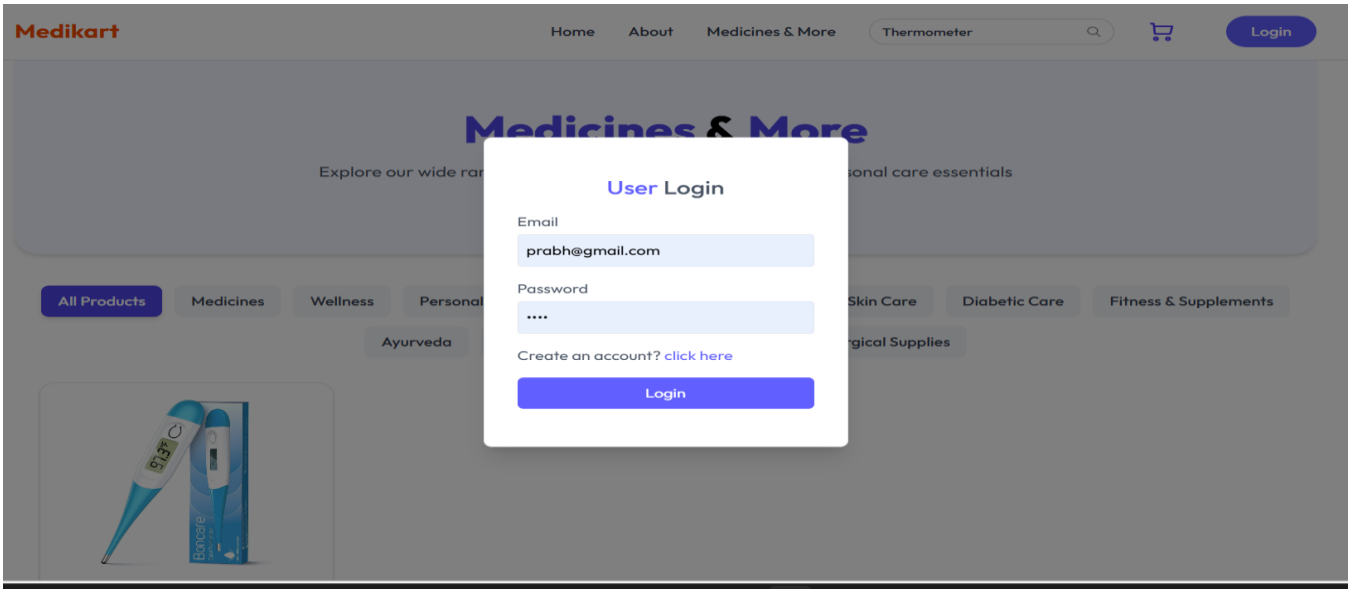


Figure 6.1 – Login and Register

5. Seller Dashboard

The Seller Dashboard enables sellers to manage their inventory and track customer orders.

- **Add Product Page:**
    - Sellers can upload new medicines, including name, price, description, and image.
    - Data is sent to the backend and stored in MongoDB.

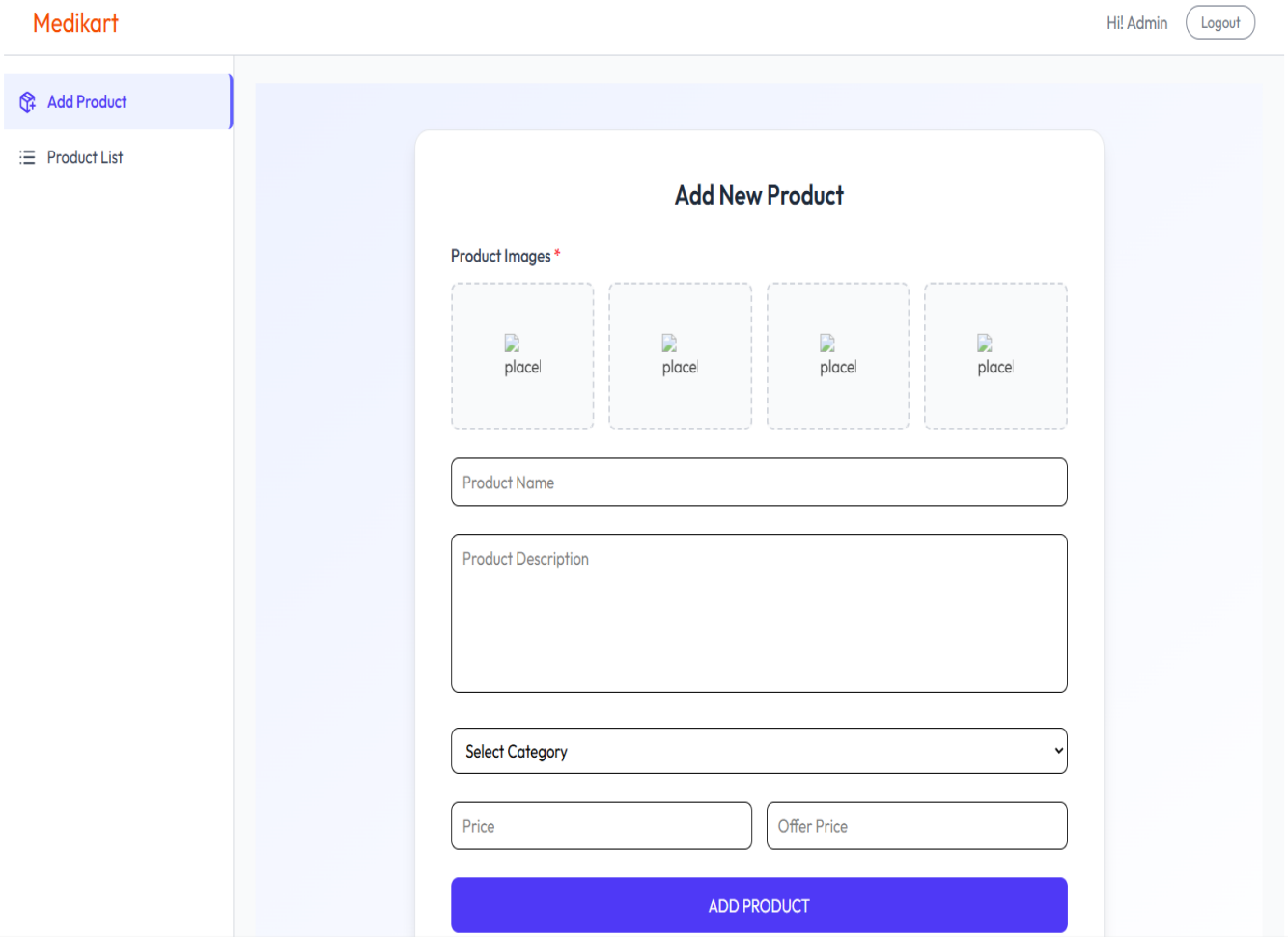


Figure 7.1 – Add Product

- **Product List Page:**
    - Displays all products uploaded by the seller.
    - Sellers can edit or delete products as needed.

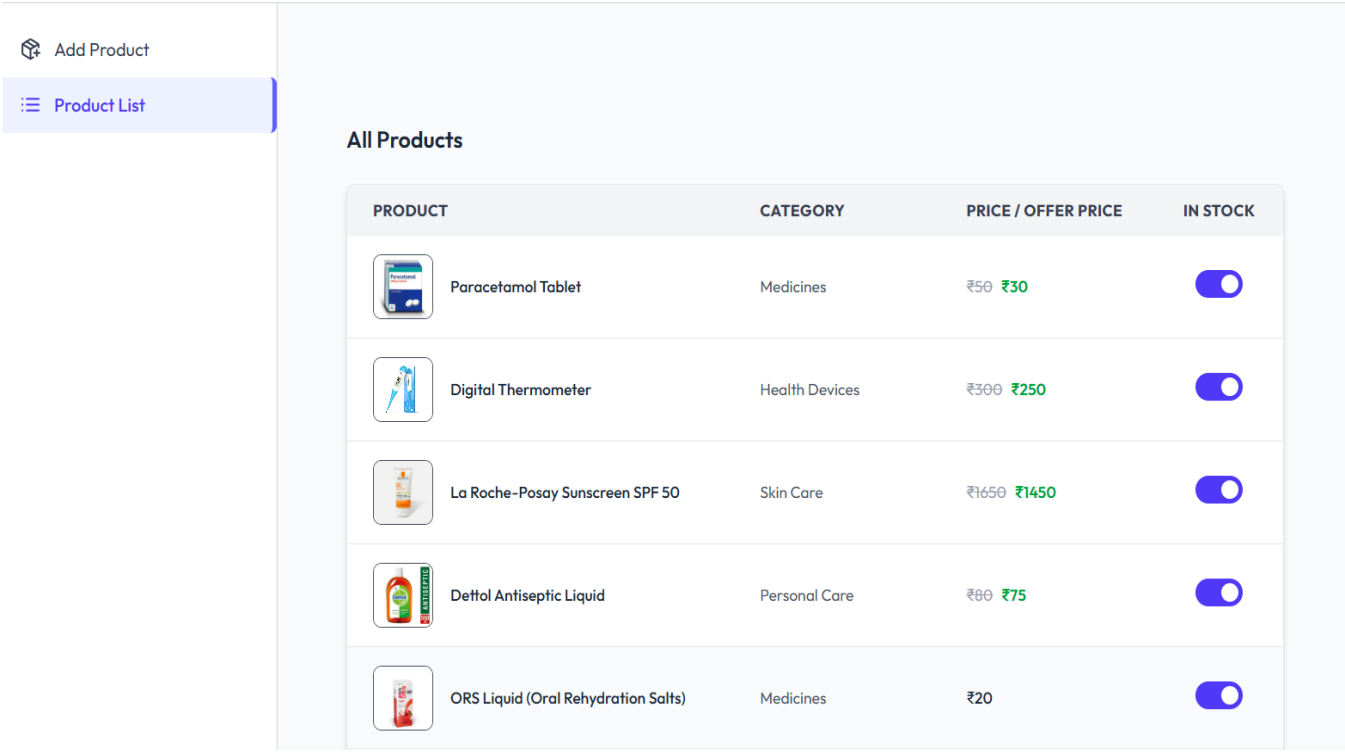


Figure 7.2 – Product list

7. Summary

The MediKart application provides a simple, responsive, and user-friendly interface for online medicine ordering. The frontend and backend are well integrated to ensure smooth functionality for both users and sellers.

UI Summary

- **Component-Based Design:**
  The user interface is built using reusable React components such as Navbar, Product Cards, Cart, Checkout, Address, and Dashboard.
- **Dynamic Interaction:**
  Actions like adding items to the cart, placing orders, and viewing order history update dynamically using backend data.
- **Smooth Navigation:**
  React Router DOM is used to navigate between pages without reloading the application.
- **Responsive Layout:**
  The interface adapts well to different screen sizes, providing usability on desktops and mobile devices.

- **User-Friendly Experience:**
  Clear buttons, readable text, and simple layouts make the application easy to use.

Overall Result:

The frontend works smoothly with the backend APIs, allowing users to browse medicines, manage carts, and place orders efficiently. Sellers can manage products and view orders without complexity.

Backend Integration

The backend of MediKart is developed using **Node.js** and **Express.js**, with **MongoDB** as the database. It manages all server-side logic and data storage.

Key Backend Functionalities

- **User and Seller Authentication:**
  Secure signup and login functionality is implemented using JWT tokens to protect user data and sessions.
- **Product Management:**
  Sellers can add, update, and delete medicine products. All product data is stored in MongoDB.
- **Cart and Order Management:**
  User carts and orders are saved in the database and updated in real time.
- **Address Management:**
  Users can add and manage delivery addresses, which are stored securely in MongoDB.
- **API Communication:**
  React frontend communicates with the backend using Axios to perform CRUD operations through Express APIs.

Final Outcome:

The backend successfully supports all application features and ensures reliable data handling. The integration between frontend, backend, and database results in a fully functional MERN stack application.

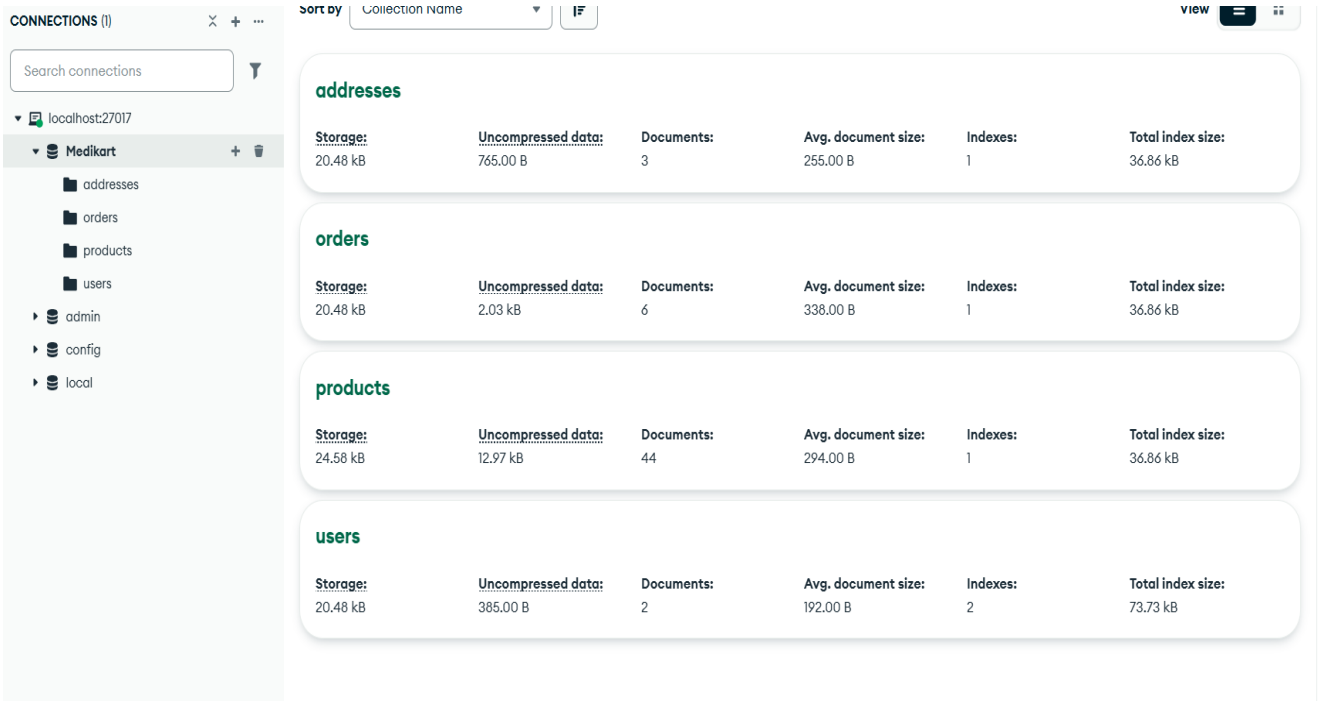


Figure 8.1– Product list

1. **Users collection** –

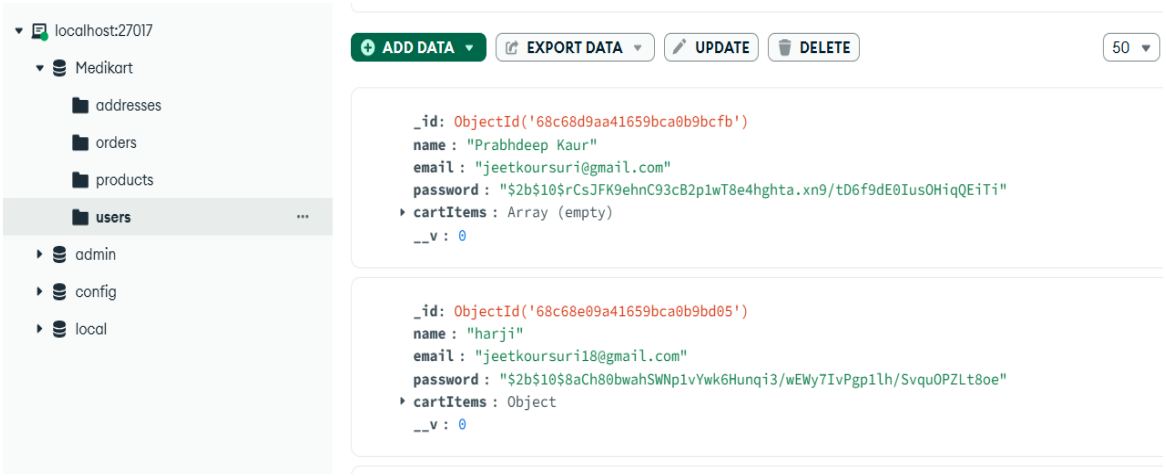


Figure 8.2- user collection

2. **Products collection** –

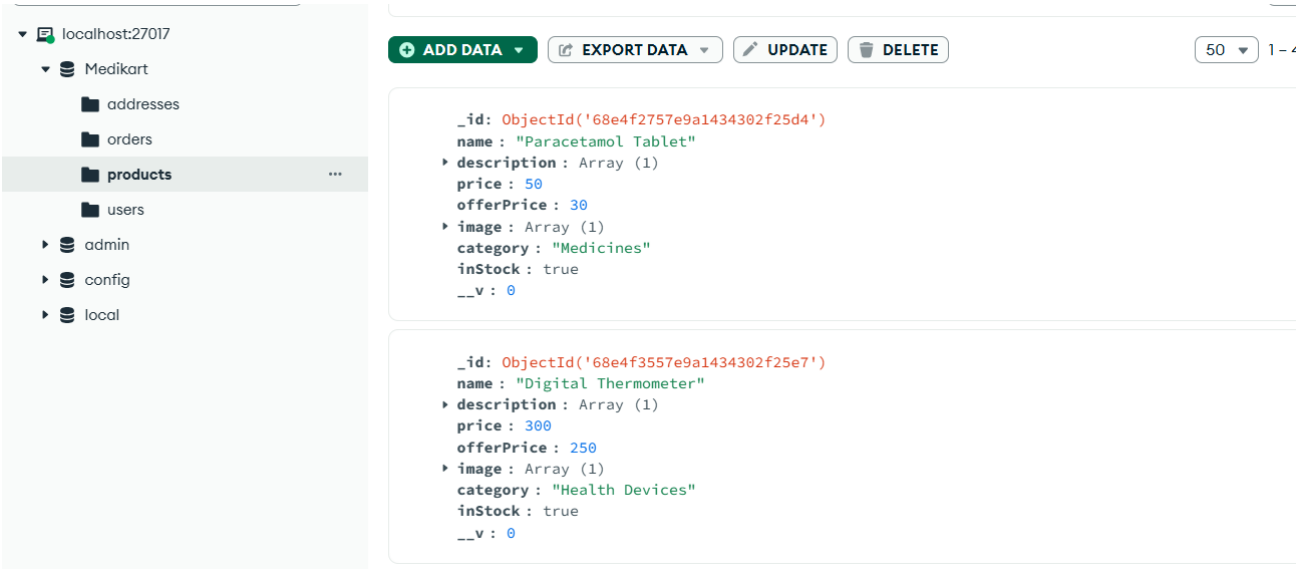


Figure 8.3- Products Collection

3. **Orders collection** –

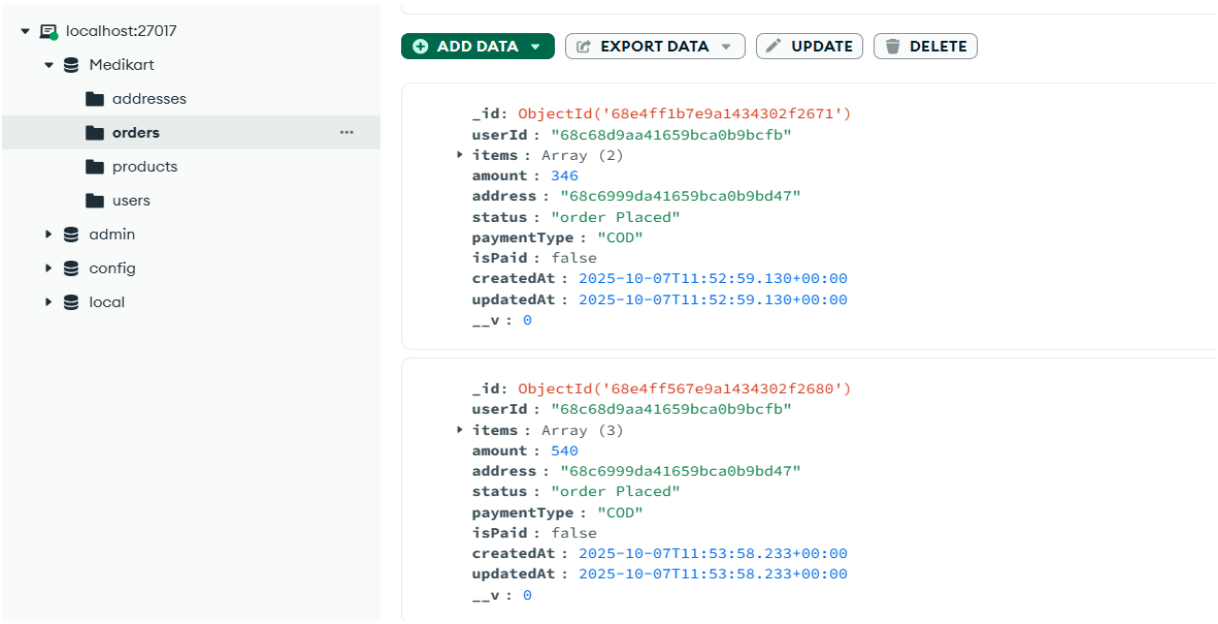


Figure 8.4 – Order Collections

4. **Addresses collection** –

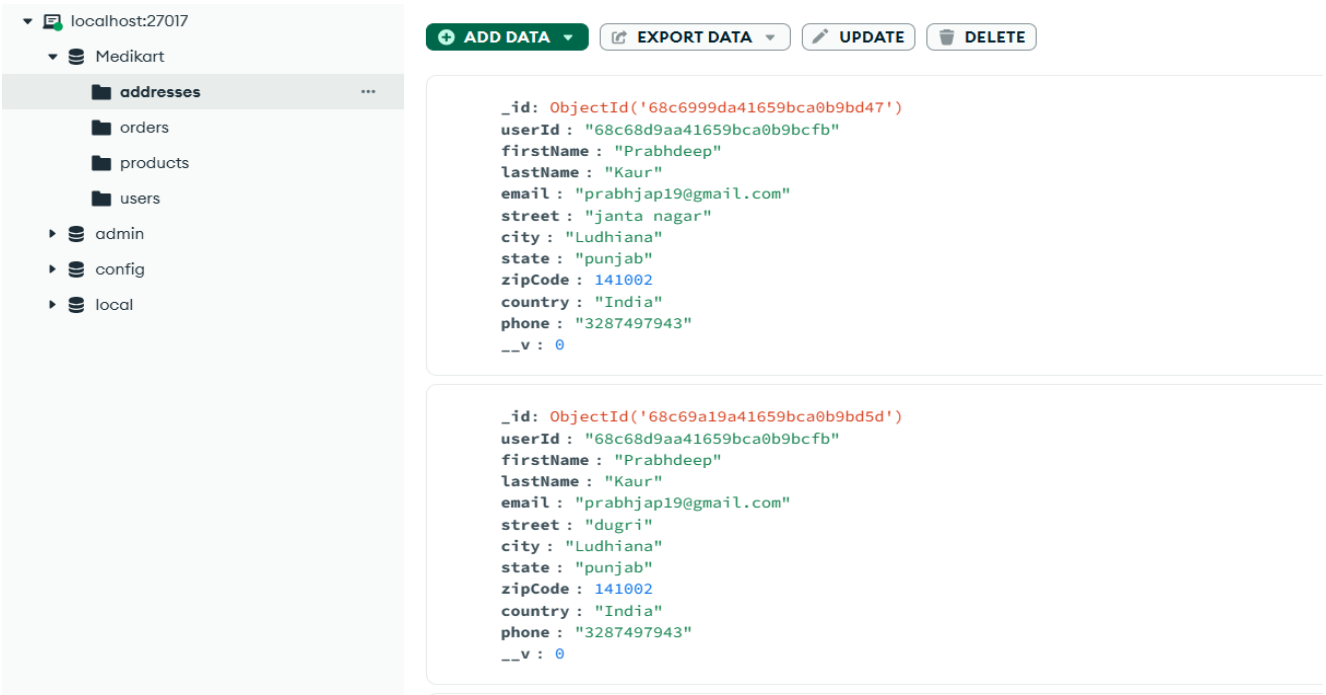


Figure 8.5 –Address Collections

Summary:

The backend ensures that all user, seller, product, and order data is securely stored and managed in MongoDB. The frontend interacts smoothly with these collections via APIs, providing real-time updates for the application.

3.3 Technical Results and Backend Integration

The MediKart project is developed using the MERN stack, where the frontend built with React.js communicates with a backend developed using Node.js and Express.js, and MongoDB is used for database storage. This integration allows smooth data flow between the user interface, server logic, and database.

Backend Implementation

The backend of MediKart handles all server-side operations required for the application.

User Authentication

- User signup and login are implemented using JWT (JSON Web Tokens).
- Authentication ensures that only logged-in users can access cart, checkout, and order pages.

Product Management

- Products (medicines) are stored in MongoDB.
- Sellers can add, update, and delete products using backend APIs.
- Product data is fetched dynamically and displayed on the frontend.

Cart and Order Management

- Users can add products to the cart, update quantities, and remove items.
- Order details such as products, quantity, total price, and delivery address are stored in the database.
- Users can view their order history in the My Orders section.

Address Management

- Users can add delivery addresses during checkout.
- Addresses are saved in MongoDB and linked to the user account.

Database Integration

- MongoDB is used to store:
    - Users
    - Products
    - Cart data
    - Orders
    - Addresses
- All database operations are performed through Express.js APIs.

Frontend–Backend Communication

- The React frontend communicates with the backend using Axios.
- API responses are rendered dynamically in the UI.
- JWT tokens are used to protect private API routes.

Result

- Frontend and backend are successfully integrated.
- Data is stored and retrieved correctly from MongoDB.
- Users can register, log in, browse medicines, manage cart, place orders, and view order history.
- Sellers can manage products and view orders.
- The MediKart application works as a complete and functional MERN stack project.

3.4 Discussion of Learning Outcomes

Developing the MediKa**rt** project gave me practical experience in full-stack web development. Through this project, I learned how the frontend, backend, and database work together to form a complete web application. Working on MediKart helped me understand the actual development process used in real projects.

Technical Learnings

1. Frontend Development

- I designed user interfaces using React.js and Bootstrap to make the application responsive and easy to use.
- I used React Hooks such as useState, useEffect, and Context API to manage data like user login, cart items, and product lists.
- I implemented page navigation using React Router, which allowed users to move between pages without reloading the website.

2. Backend Development

- I created backend APIs using Node.js and Express.js to manage users, products, carts, orders, and addresses.
- I implemented JWT authentication to secure login and restrict access to protected pages.
- I learned to organize backend code using models, controllers, and middleware, which made the backend easier to manage.

3. Database Integration

- I designed MongoDB collections for users, products, carts, orders, and addresses.
- I connected MongoDB with the backend to store and retrieve data whenever required.
- I used MongoDB Compass to view and verify the stored data during development.

4. API Integration

- I connected the frontend with the backend using Axios.
- I handled API responses and fixed errors related to data fetching, authentication, and form submission.

Professional and Practical Skills

- I learned how to plan a project by dividing work into frontend, backend, and database tasks.
- I used Git and GitHub to manage code, track changes, and maintain project versions.

Result

By completing the MediKart project, I was able to:

Develop a fully working MERN stack application with features like product listing, cart, and order management.

Understand the complete development cycle from UI design to backend and database integration.

3.5 Summary

The **MediKart** project successfully met its technical and functional goals. It is a complete MERN stack application that allows users to browse and order medicines, while sellers can manage products and orders.

Key highlights of the project:

- **Frontend:** A responsive interface built using React.js and Bootstrap with smooth navigation using React Router.
- **Backend:** Secure APIs developed with Node.js and Express.js, including JWT-based authentication.
- **Database:** MongoDB used to store users, products, carts, orders, and addresses efficiently.
- **Integration:** Frontend and backend connected using Axios for dynamic data handling.
- **Learning Outcome:** Gained practical experience in full-stack development, debugging, testing, and version control using GitHub.

Overall, this project helped me apply theoretical concepts in a real-world application and improved my confidence in developing full-stack web projects.

CHAPTER 4:
CONCLUSION, FUTURE SCOPE, AND REFERENCES

4.1 Conclusion

The MediKart project represents the successful completion of my Full Stack Web Development training using the MERN Stack (MongoDB, Express.js, React.js, and Node.js). This project demonstrates how these technologies were practically used to develop an online medicine ordering platform for users and sellers.

Through this project, I gained hands-on experience in frontend and backend development. I worked on creating user interfaces using React, developing backend APIs with Node.js and Express.js, implementing authentication, and integrating the application with MongoDB. The system was designed to ensure smooth functionality, proper data handling, and secure access.

Key Outcomes:

1. **Project Implementation:**
   Implemented user and seller authentication using JWT, developed features for product listing, cart management, address handling, and order placement, and connected frontend and backend using Axios.
2. **Technical Skills:**
   Used MongoDB for storing application data, applied React Hooks and Context API for state management, and used CSS and Bootstrap to create a responsive user interface.
3. **Skill Improvement:**
   Gained a clear understanding of frontend-backend communication, REST API working, debugging using browser developer tools and Postman, and version control using GitHub.
4. **Practical Experience:**
   Learned proper project structuring, code organization, testing features, and preparing the application for deployment.

Overall, the MediKart project helped me apply theoretical concepts in a real-world application and strengthened my understanding of full-stack web development.

4.2 Future Scope

The current version of **MediKart** includes basic e-commerce features such as user login, product listing, cart management, and order placement. In the future, the platform can be improved further by adding the following features:

1. **Online Payment Integration:**
   Payment gateways like Razorpay or Stripe can be added to support UPI, debit/credit cards, and net banking.
2. **Admin Dashboard:**
   An admin panel can be introduced to manage users, sellers, products, and view sales reports.
3. **Product Recommendation System:**
   Suggestions can be provided to users based on their previous orders to improve user experience.
4. **Inventory Management:**
   Sellers can be given real-time stock management with low-stock alerts.
5. **Mobile Application:**
   A mobile app version can be developed to make MediKart accessible on smartphones.
6. **Multi-Language Support:**
   Regional language options and accessibility features like dark mode can be added.
7. **Cloud Deployment:**
   The application can be deployed on cloud platforms for better performance and scalability.
8. **Notifications:**
   Email or SMS notifications can be implemented for order confirmation and delivery updates.
9. **Data Analytics:**
   User activity and sales data can be analyzed to improve business decisions.
10. **Improved Security:**
    Additional security features like two-factor authentication can be added.

These improvements will make MediKart more scalable, secure, and user-friendly in the future.

4.3 References

The following resources were used during the development of the **MediKart** project for understanding concepts, writing code, and fixing errors:

[1] MongoDB Official Documentation – https://www.mongodb.com/docs/

[2] Express.js Official Documentation – https://expressjs.com/

[3] React.js Official Documentation – https://react.dev/

[4] Node.js Documentation – https://nodejs.org/en/docs/

[5] Bootstrap Documentation – https://getbootstrap.com/docs/

[6] GeeksforGeeks(HTML, CSS, JavaScript) – https://developer.mozilla.org/

[7] GitHub Guides for Version Control – https://docs.github.com/

[8] YouTube Tutorials:

- Code With Harry– MERN Stack Tutorials
- freeCodeCamp.org – Full MERN Stack Course
- Programming with Mosh – React and Node.js Tutorials

[9] Online learning websites such as GeeksforGeeks, W3Schools, and Stack Overflow were also referred for problem-solving and practical implementation.