



ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

School of Electrical Engineering and Computing

Department of Computer Science and Engineering

Advanced Programming

Group Project

No	Name	ID	Sec
1	Eleni Tadese	UGR/30450/15	3
2	Marya Getu	UGR/30857/15	3
3	Samuel Zenebe	UGR/31182/15	3
4	Mohammed Sadik	UGR/30960/15	3
5	Medhanit Tesfaye	UGR/30862/15	3
6	Yohannis Kifle	UGR/31426/15	3
7	Radyat Daniel	UGR/31115/15	3
8	Moti Alemu	UGR/30963/15	3
9	Osman Sultan	UGR/31107/15	3

Submitted to:- Mrs. Yared

Submitted date:- 01/Jan/2025G.C

Online Shopping Platform - Backend API

This backend serves as the foundation for an online shopping platform where users can view product details and read customer reviews. The backend is built using Java and interacts with a MySQL database. This backend exposes RESTful APIs that can be consumed by a frontend application built using **Vite**, **React**, and **Redux**. The goal is to enable users to interact with the platform by retrieving products and reviews, which are then displayed on the frontend.

Introduction

Welcome to the documentation for the Online Shopping Platform! This comprehensive guide provides everything you need to get started with both the backend API and frontend integration. Whether you're a developer building the platform from scratch or someone looking to understand the entire workflow, this documentation will serve as your reference point.

The Online Shopping Platform is a complete system designed to facilitate a seamless online shopping experience. It allows users to browse a variety of products, view product details, and read reviews from other customers. The system is built with Java on the backend, using Servlets to handle HTTP requests, and is integrated with a MySQL database to store and retrieve product information and customer reviews.

The frontend of the platform is developed using React for creating the user interface, Redux for managing the application's state, and Vite as the build tool to optimize development speed and performance. The frontend interacts with the backend through RESTful APIs, which are designed to allow the frontend to fetch product details, customer reviews, and other information seamlessly.

Why This Platform?

Online shopping platforms have become a crucial part of the e-commerce industry. Whether you're purchasing electronics, clothing, or groceries, an intuitive and fast system is required to ensure a smooth shopping experience. This platform was created to not only provide the basics of an online store but also demonstrate how to integrate the frontend and backend in a real-world e-commerce setup. It serves as a functional and extensible example for developers looking to build and customize their own shopping platforms.

Technologies Used

This system integrates multiple technologies to provide a robust, scalable, and easy-to-maintain online shopping experience. Here's a quick overview of the key technologies used in both the frontend and backend:

- **Java 11 (or higher):** The backend is built using Java with Servlets to handle HTTP requests and responses. Java is a widely-used programming language that offers stability and scalability for server-side development.
- **Servlets:** Java Servlets are used to manage HTTP requests, process data, and return appropriate responses, such as JSON data or HTML content.
- **JDBC:** The backend uses JDBC (Java Database Connectivity) to interact with a MySQL database. JDBC enables secure and efficient communication between the Java application and the database, allowing for CRUD (Create, Read, Update, Delete) operations on data.
- **MySQL:** A relational database management system (RDBMS) used to store product information, customer reviews, and other relevant data for the shopping platform.
- **Tomcat:** A popular web server for Java applications that hosts the servlet-based backend.
- **Vite:** A modern build tool that provides fast hot module replacement and optimized bundling for frontend development.
- **React:** A JavaScript library for building dynamic and interactive user interfaces. It allows developers to create reusable components that manage their own state.
- **Redux:** A state management library for JavaScript applications. It is used in this platform to manage global application state, such as the current shopping cart, user authentication, and product details.
- **Axios:** A promise-based HTTP client for the browser and Node.js. It is used to send requests from the frontend React app to the backend API.

Purpose of This Documentation

This documentation serves as a guide to help you:

1. **Understand the System Architecture:** It provides an overview of how the different components of the platform interact with each other, from the frontend to the backend, and how data flows between them.
2. **Set Up the Backend:** Instructions for setting up the backend API, configuring the database, and deploying the backend using a Java web server (e.g., Apache Tomcat).
3. **Integrate the Frontend:** Clear steps on how to set up the React app, integrate Redux for state management, and communicate with the backend through API calls to display product details and reviews.
4. **Customize and Extend the Platform:** Whether you're adding new features, modifying the user interface, or integrating with third-party services, this guide will help you extend the functionality of the platform.
5. **Deploy and Maintain the Platform:** Learn how to deploy both the frontend and backend to production environments and keep the system running smoothly over time.

By following this documentation, you'll be able to fully understand the workings of the Online Shopping Platform, from backend development to frontend integration. You will also gain the skills necessary to extend, modify, and deploy similar systems for your own e-commerce projects.

Key Concepts Covered in This Documentation

- **Backend API Development:** Learn how to develop RESTful APIs in Java using Servlets, interact with MySQL, and expose endpoints for the frontend to consume.
- **Frontend Development:** Understand how to build a dynamic and responsive user interface using React, with data fetched from the backend API to display product details and reviews.
- **State Management:** Learn how to manage global application state in React using Redux, including handling the shopping cart and product details.
- **Database Interaction:** Understand how the backend communicates with the MySQL database to store and retrieve product and review data efficiently.
- **API Integration:** See how to integrate the frontend with backend APIs using Axios to send and receive data, ensuring smooth communication between the client and server.

Who This Documentation Is For

This guide is intended for:

- **Backend Developers:** Those who want to understand how to build a Java-based backend with Servlets, handle HTTP requests, interact with databases, and expose RESTful APIs.
- **Frontend Developers:** Developers who want to build modern, dynamic web applications using React and integrate them with backend APIs. It's also helpful for those learning how to use Redux for state management in React.
- **Full-Stack Developers:** Developers who want to work on both the frontend and backend of the platform and understand how to tie everything together.
- **Beginners:** Even if you're just getting started with web development, this documentation will guide you through setting up and understanding both the frontend and backend components of an online shopping platform.

What You Will Learn

Throughout this guide, you will:

- Set up a Java-based backend using Servlets and MySQL to manage product and review data.
- Build a React frontend using Redux for state management and Vite for fast development and bundling.
- Learn how to handle API calls using Axios to fetch data from the backend and display it dynamically on the frontend.
- Understand how to create a shopping cart feature and manage global state with Redux.
- Deploy the entire system to production using best practices for Java, MySQL, React, and Vite.

Whether you are building a simple prototype or a full-fledged e-commerce platform, this documentation will guide you through every step of the development process.

This enhanced introduction should provide users with a deeper understanding of the project's purpose, technologies used, and the intended audience for the documentation.

Frontend Technologies:

1. Vite:

- Vite is a modern build tool used to bundle and serve the frontend application.
- It enables fast development and offers features like hot module replacement (HMR) for a smoother development experience.

2. React:

- React is a powerful library for building user interfaces, especially single-page applications (SPAs).
- It allows developers to create reusable UI components and efficiently manage state and user interactions.

3. Redux:

- Redux is used for state management in React applications. It provides a central store to hold the application's state and allows components to access and update the state in a predictable manner.
- With Redux, state can be shared globally, making it easier to manage the product details and reviews across multiple components.

4. Axios or Fetch:

- Axios (or the native Fetch API) is used for making HTTP requests from the frontend to the backend.
- Axios simplifies sending GET, POST, PUT, and DELETE requests and handling responses, such as retrieving product details or posting new reviews.

API Endpoints

This backend exposes a set of RESTful API endpoints that the frontend can interact with. The main endpoint currently provided is for fetching product details and reviews.

1. GET /productDetailsServlet?id={productId}

This endpoint retrieves detailed information about a product and its reviews based on the provided product ID.

Request:

- **URL:** /productDetailsServlet?id={productId}
- **Method:** GET
- **Query Parameters:**
 - id: The product ID of the item you want to retrieve (this is a required parameter).

Response (JSON):

When the API receives the request, it will return a JSON response with the following information:

- **Product Information:** The product's details, such as its name, description, price, image, stock quantity, discount percentage, etc.
- **Reviews:** A list of reviews associated with the product, each containing:
 - id: The review ID.
 - rating: The customer's rating for the product (from 1 to 5).
 - reviewText: The textual review left by the customer.
 - customerId: The ID of the customer who submitted the review.
 - createdAt: The date and time when the review was created.

Example Request:

GET <http://localhost:8080/olineshoppingplatform/ProductDetailsServlet?id=1>

Example Response:

```
{
  "product": {
    "productId": 1,
    "name": "Smartphone",
    "description": "Latest model with all the advanced features.",
    "imageUrl": "https://example.com/images/smartphone.jpg",
    "price": 499.99,
    "discount": 10.00,
    "categoryId": 2,
    "stock": 25,
    "sellerId": 1
  },
  "reviews": [
    {
      "id": 101,
      "rating": 4.5,
      "reviewText": "Great phone with excellent battery life!",
      "customerId": 1,
      "createdAt": "2024-12-20T12:34:56"
    },
    {
      "id": 102,
      "rating": 3.0,
      "reviewText": "Good phone but overpriced for the features.",
      "customerId": 2,
      "createdAt": "2024-12-21T14:20:30"
    }
  ]
}
```

- **Product Details:** The response includes the name, price, description, stock, and seller information for the product.
- **Reviews:** A list of reviews is included, with each review containing the customer's rating, written feedback, and the time it was created.

Frontend Integration Example

To connect the backend with the frontend, the frontend application will need to send requests to the backend API and display the product details and reviews to the user. Below is a detailed explanation of how to integrate the backend with a React-based frontend using **Redux** for state management.

Setting Up the Frontend

You will use **Vite** to bundle your React application, and **Redux** to manage the global state (such as storing product details and reviews).

1. Create a New Vite Project with React:

To get started, create a new Vite project with the React template. Open your terminal and run:

```
npm create vite@latest my-app --template react cd my-app
```

2. Install Necessary Dependencies:

For managing state with Redux and making API calls, install the necessary dependencies:

```
npm install redux react-redux axios
```

This will install:

- **Redux:** The state management library.
- **React-Redux:** The bindings that allow React to interact with Redux.
- **Axios:** A promise-based HTTP client for making API requests.

Creating Redux Store

In Redux, we use a **store** to manage the application's state. The store will hold the product information and reviews.

store.js:

```
import { createStore } from 'redux';

const initialState = {
  product: null,
  reviews: [],
};

function productReducer(state = initialState, action) {
  switch (action.type) {
    case 'SET_PRODUCT_DETAILS':
      return { ...state, product: action.product };
    case 'SET_REVIEWS':
      return { ...state, reviews: action.reviews };
    default:
      return state;
  }
}

const store = createStore(productReducer);

export default store;
```

- The store has an initial state with a product object and an empty reviews array.
- The reducer listens for two actions: SET_PRODUCT_DETAILS and SET_REVIEWS, which update the product and reviews in the state.

Creating Redux Actions

Actions are used to send data to the store. In this case, we need actions to update the product and review details.

actions.js:

```
export const setProductDetails = (product) => ({
  type: 'SET_PRODUCT_DETAILS',
  product,
});

export const setReviews = (reviews) => ({
```

```
type: 'SET_REVIEWS',
reviews,
});
```

- `setProductDetails`: This action updates the store with the product details fetched from the backend.
- `setReviews`: This action updates the store with the list of reviews for the product.

Fetching Product Details and Reviews

Once the Redux store is set up, you can use **React-Redux** to connect your React components to the store. You'll fetch product details and reviews from the backend API and update the state.

ProductDetails Component:

```
import React, { useEffect } from "react";
import { useDispatch, useSelector } from "react-redux";
import { setProductDetails, setReviews } from "../actions";
import axios from "axios";

const ProductDetails = ({ productId }) => {
  const dispatch = useDispatch();
  const { product, reviews } = useSelector((state) => state);

  useEffect(() => {
    const fetchProductDetails = async () => {
      const response = await
        axios.get(`http://localhost:8080/olineshoppingplatform/ProductDetailsServlet?id=${productId}`);
      dispatch(setProductDetails(response.data.product));
      dispatch(setReviews(response.data.reviews));
    };

    fetchProductDetails();
  }, [productId, dispatch]);

  if (!product) {
    return <div>Loading...</div>;
  }

  return (
    <div>
      <h2>{product.name}</h2>
      <img src={product.imageUrl} alt={product.name} />
      <p>{product.description}</p>
      <p>Price: ${product.price}</p>
      <p>Discount: {product.discount}%</p>
    </div>
  );
};
```

```

    <p>Stock: {product.stock}</p>

    <h3>Reviews:</h3>
    <ul>
      {reviews.map((review) => (
        <li key={review.id}>
          <p>Rating: {review.rating}</p>
          <p>{review.reviewText}</p>
          <p>Reviewed by customer {review.customerId} on {new
Date(review.createdAt).toLocaleString()}</p>
        </li>
      ))}
    </ul>
  </div>
);
};

```

```
export default ProductDetails;
```

- **useDispatch**: A hook from React-Redux to dispatch actions (like updating product details and reviews).
- **useSelector**: A hook to access the state from Redux, allowing you to use the product and reviews in your component.
- **useEffect**: This React hook fetches the product data when the component is first rendered and dispatches the results to the Redux store.

Main Entry Point (index.js)

The **index.js** file is where you connect your app to the Redux store using the Provider component.

index.js:

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { Provider } from 'react-redux';
import store from './store';

ReactDOM.createRoot(document.getElementById('root')).render(
  <Provider store={store}>
    <App />
  </Provider>
);

```

- **Provider:** This component from React-Redux makes the Redux store available to all components in the app.
- The app is wrapped in the Provider so that every component can access the Redux store and dispatch actions or read state.

Setup Instructions

These instructions guide you through setting up the backend and frontend of the online shopping platform.

1. Clone the Repository

Begin by cloning the backend repository to your local machine:

git clone <https://github.com/Moh-Sad/Advanced-Java-Project>

This will download the backend code to your computer, allowing you to work on it.

2. Set Up the Database

1. **Create a New MySQL Database:** In your MySQL client (e.g., MySQL Workbench or command line), run:
2. **CREATE DATABASE** online_shopping;
3. **Import Schema:** Import the schema for the Products and Ratings tables from the provided SQL file or write your own schema to create the necessary tables.

3. Configure Database Connection

Open the DBHelper.java file and configure the database connection with your MySQL credentials. Make sure to replace the placeholder values with your actual database connection details.

```
public class DBHelper {  
    public static Connection getConnection() throws SQLException {  
        String url = "jdbc:mysql://localhost:3306/online_shopping";  
        String username = "root";  
        String password = "yourpassword";  
        return DriverManager.getConnection  
  
(url, username, password); } }
```

Ensure that the database is accessible and that the server is running.

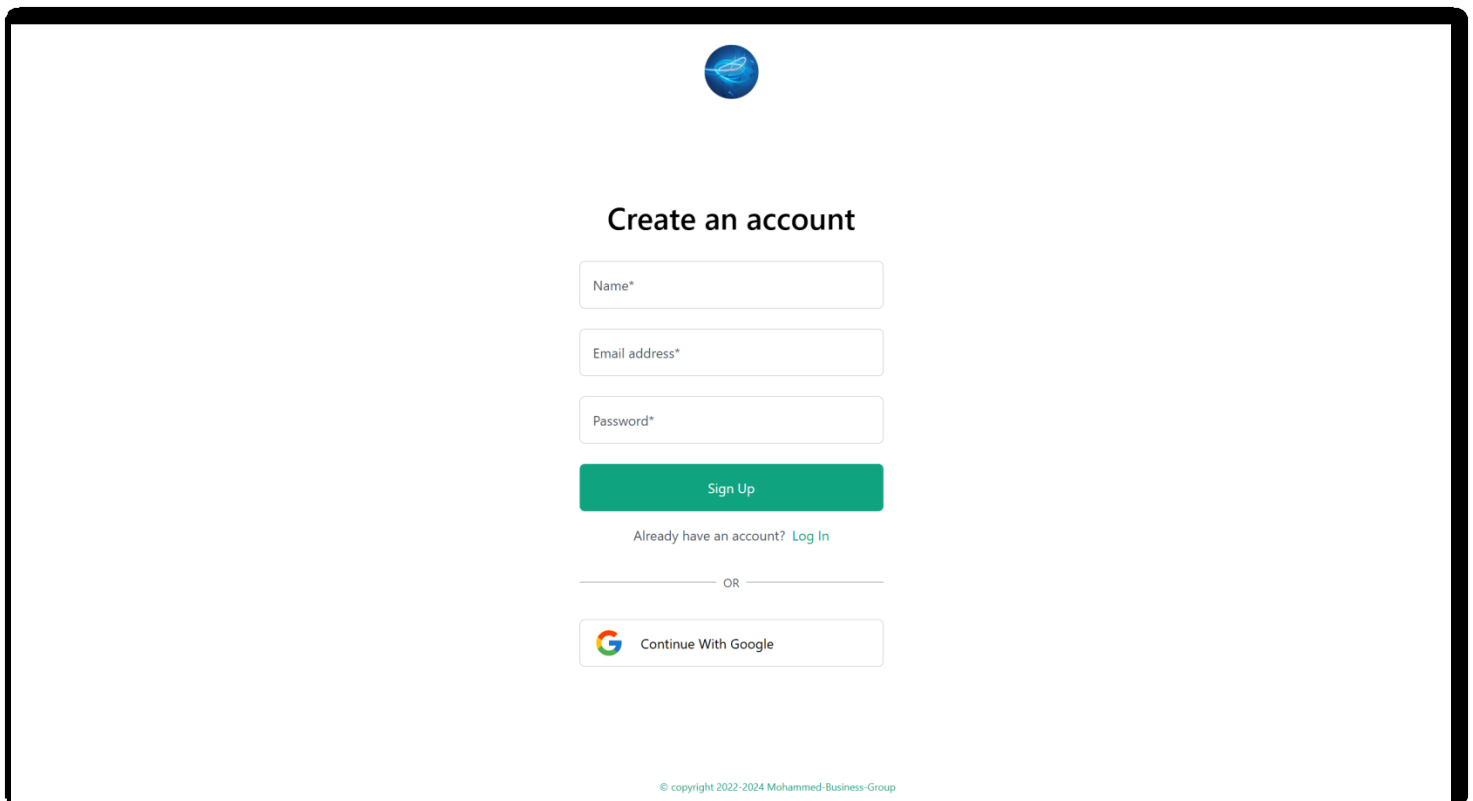
4. Deploy the Backend


To deploy the backend, you need a web server, such as **Tomcat**, to run your Java servlets.

1. Set up Tomcat and ensure that it's configured to serve the backend properly.
2. Deploy your servlet project to Tomcat, ensuring that the servlets are accessible at the correct endpoints.

Project Screenshots:

Signup Page:






Create an account

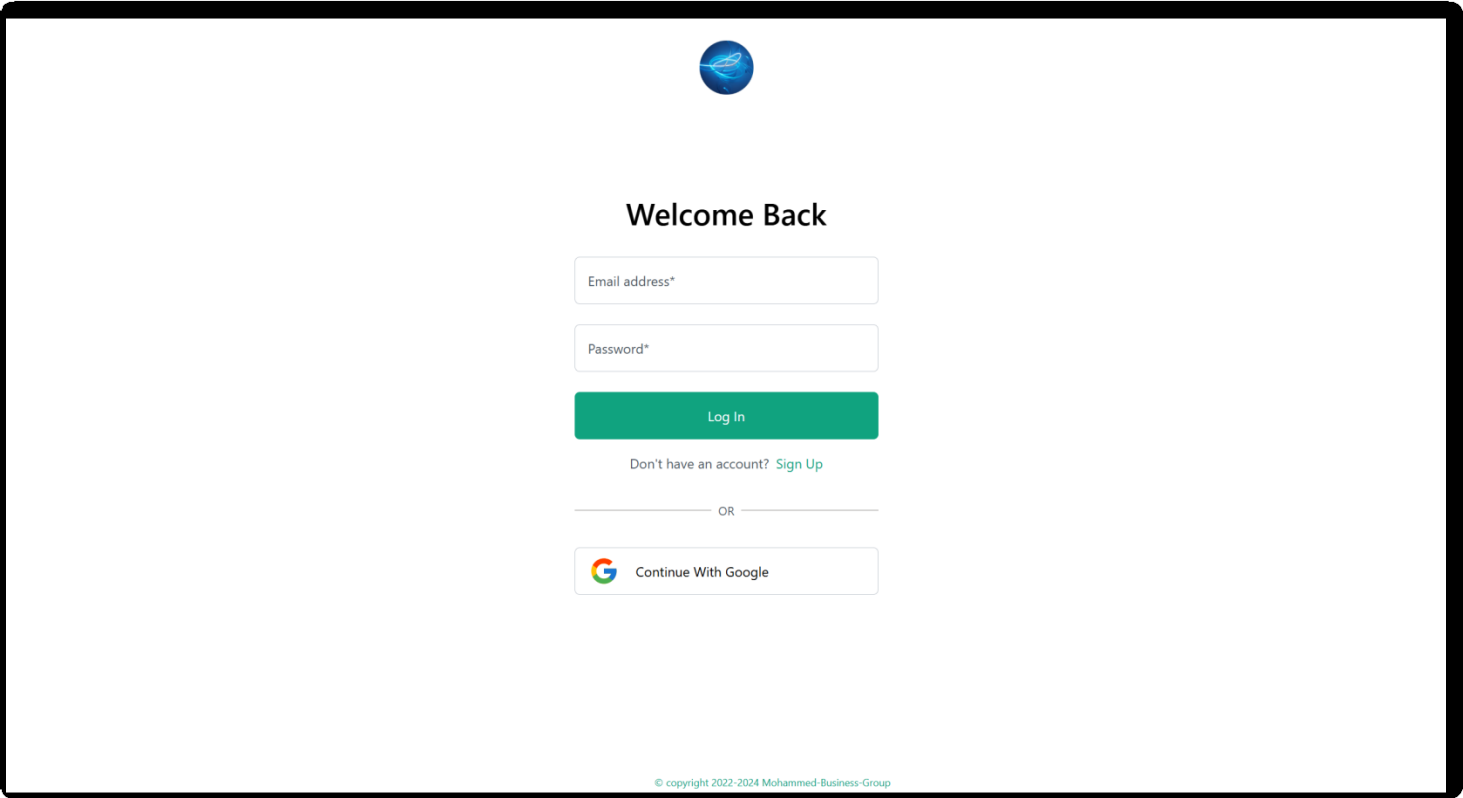
Already have an account? [Log In](#)

OR

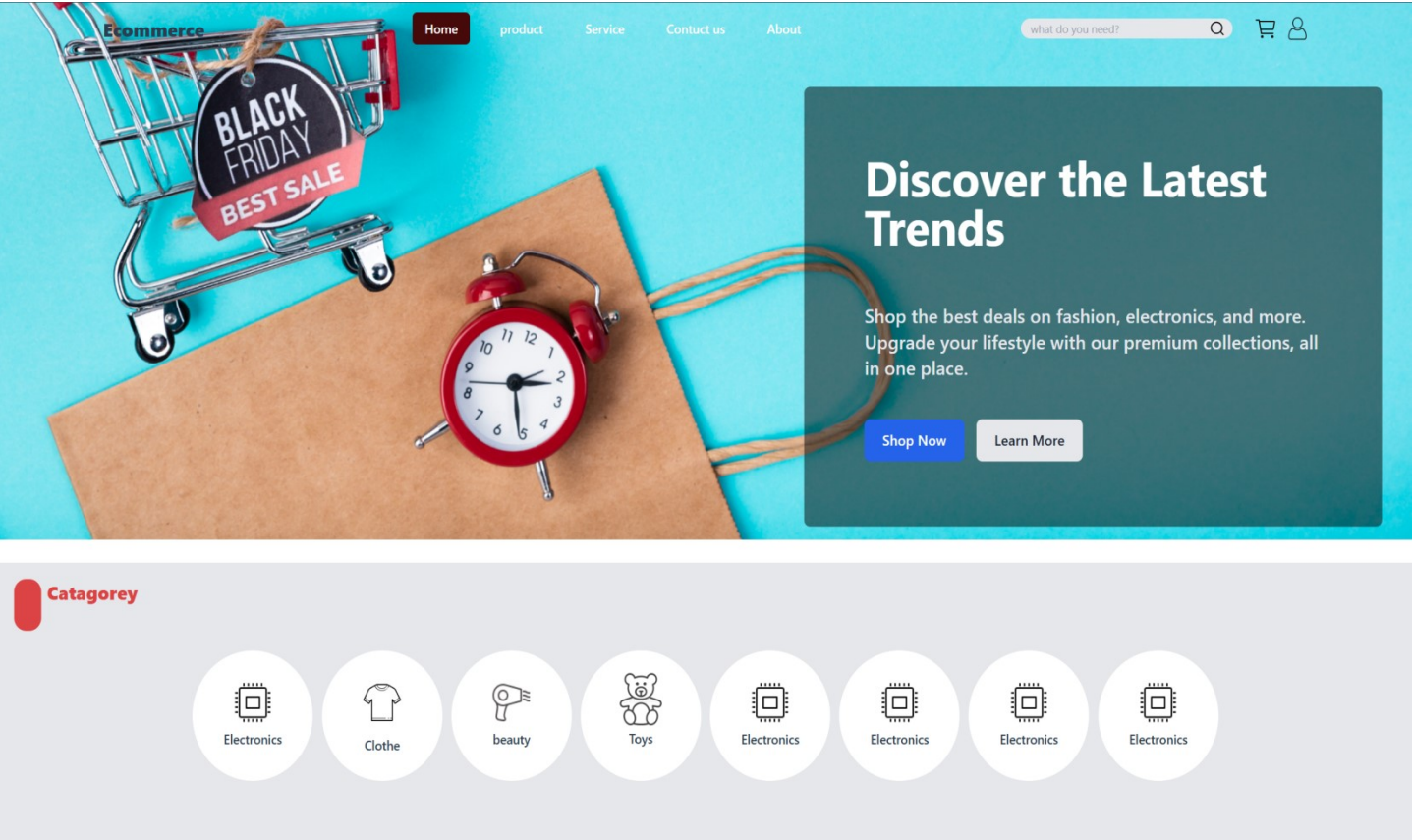


© copyright 2022-2024 Mohammed-Business-Group

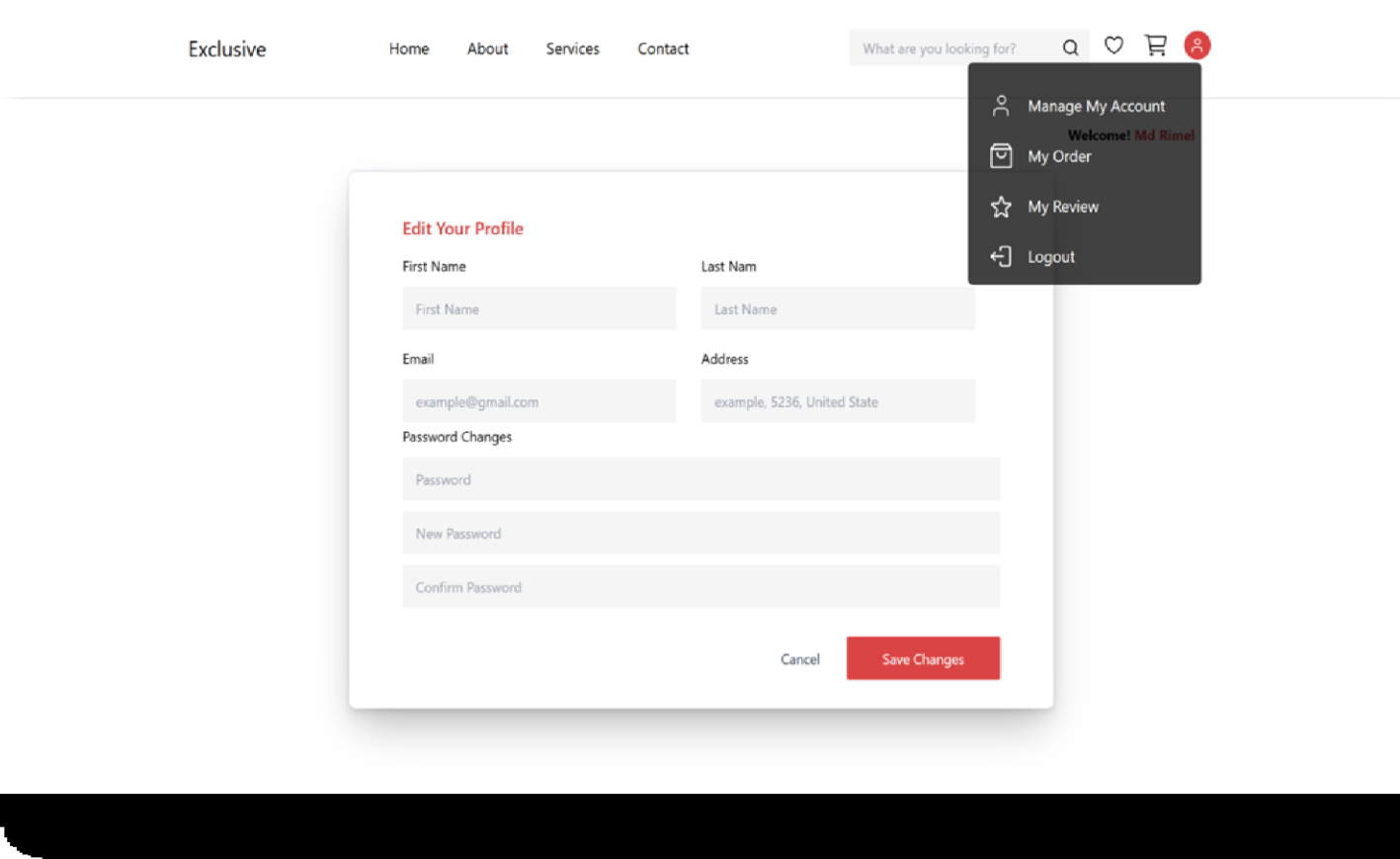
Login Page:



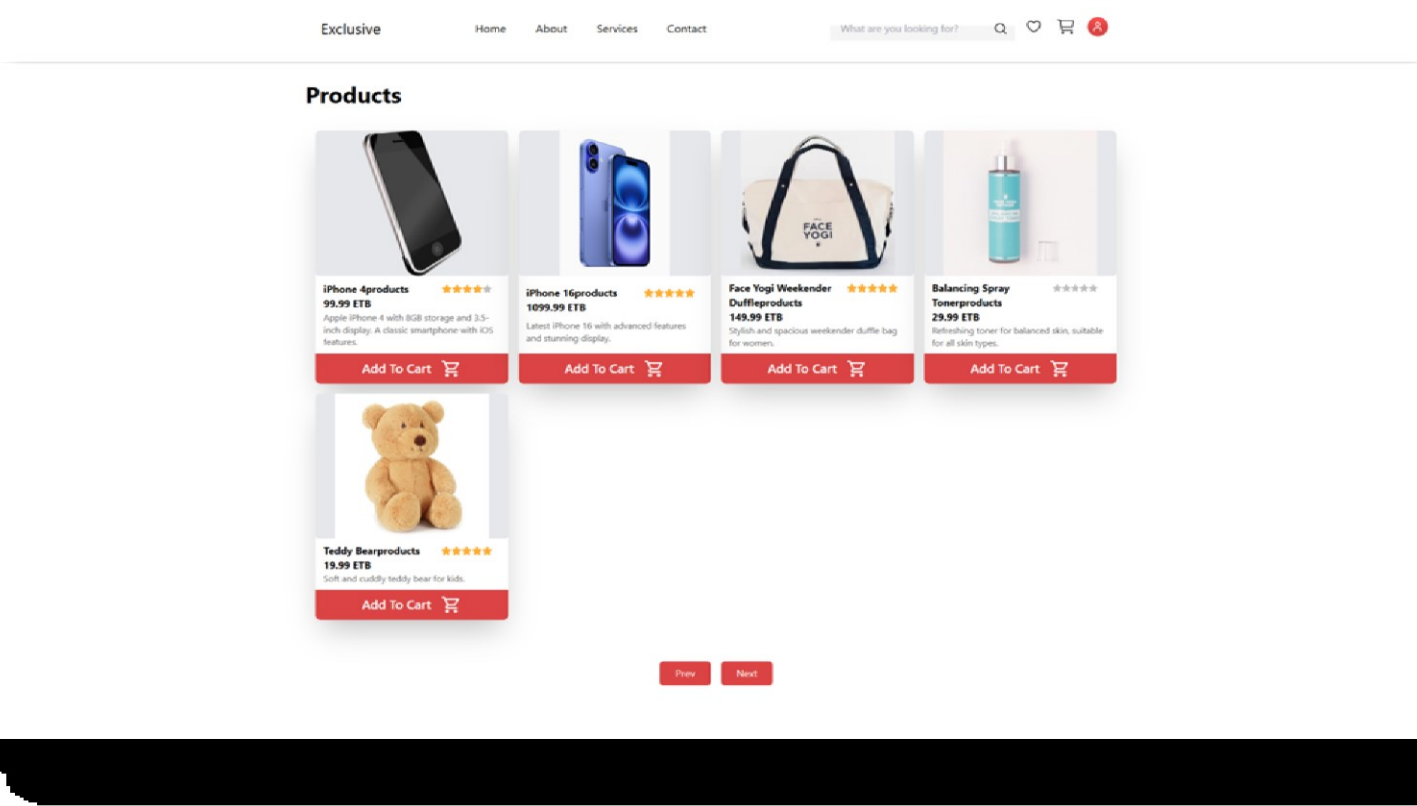
Landing Page:



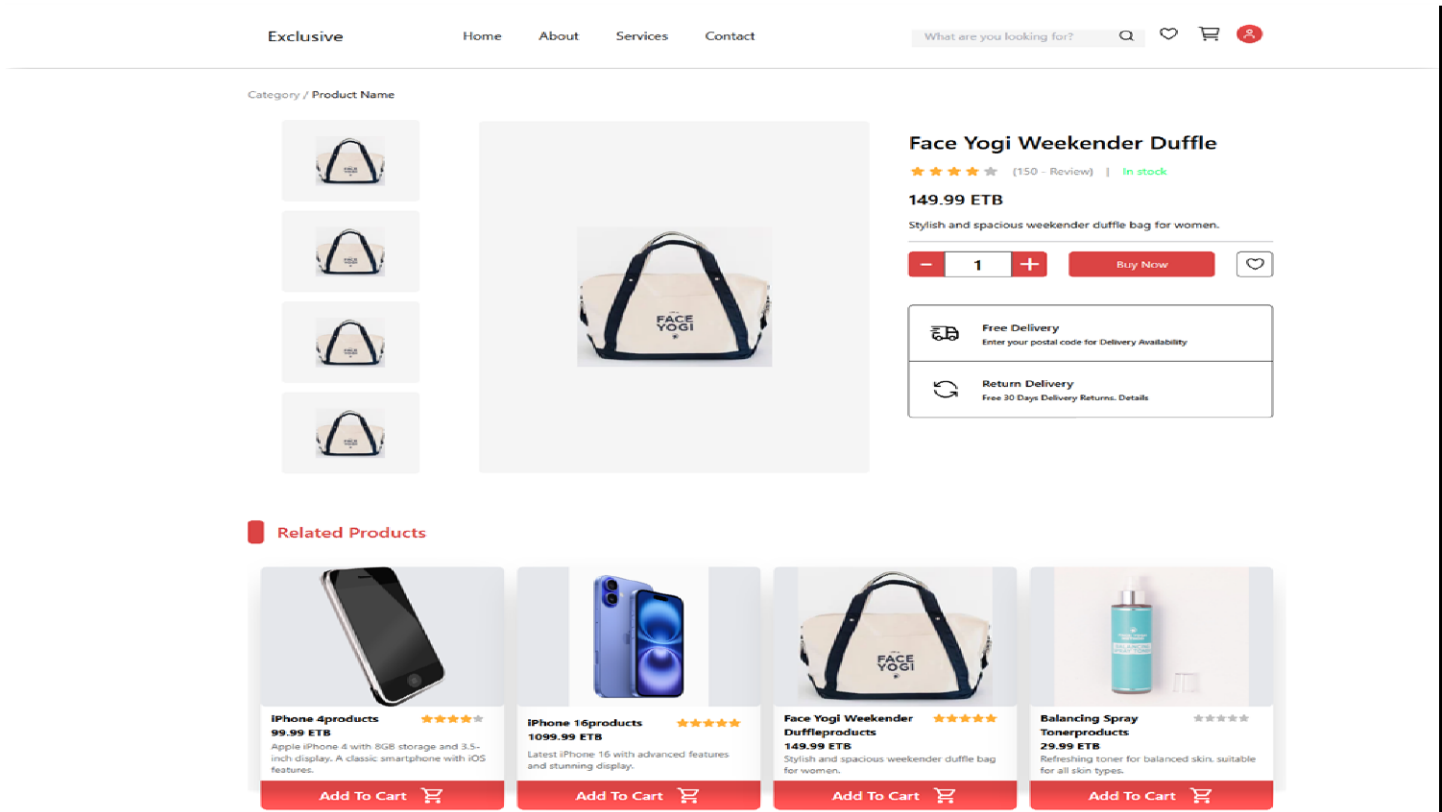
User Profile Page:



Products Page:



Product Detail Page:



Cart Page:

