INNOMATICS
RESEARCH LABS

MERN Stack E-Commerce Project Documentation

OVERVIEW

We are excited to give you a challenging and innovative project that involves developing a comprehensive e-commerce application using the MERN stack (MongoDB, Express.js, React, and Node.js). This document outlines the project requirements, goals, specifications, system design, and step-by-step implementation guide.

PROBLEM STATEMENT

You are required to develop a full-fledged e-commerce application that includes a variety of features to provide an engaging shopping experience for users. The application should support product listing, user authentication, shopping cart functionality, and order processing.

GOALS

1. User Authentication: Implement user registration, login, and logout functionality.
2. Product Management: Enable administrators to add, update, and delete products.
3. Product Listing: Display products in an organized and user-friendly manner.
4. Shopping Cart: Allow users to add, update, and remove products from their shopping cart.
5. Order Processing: Implement order placement and management.
6. Order History: Provide users with access to their order history.

INNOMATİCS
RESEARCH LABS

SPECIFICATIONS

Frontend

1. **Framework**: React.js
2. **Components:**
    - Product List: Display a list of products with images, titles, prices, and short descriptions.
    - Product Details: Show detailed information about a selected product.
    - Shopping Cart: Display the items in the user's cart and allow for quantity adjustments and item removal.
    - User Authentication: Include forms for user registration and login.
    - Order Summary: Summarize the order before finalizing the purchase.
    - Order History: Show a list of past orders for the logged-in user.
3. **Styling:** Use CSS/SCSS for styling components to ensure a responsive and visually appealing UI.

Backend

1. **Framework**: Node.js with Express.js
2. **Database**: MongoDB
3. **APIs:**
    - User API: Handle user registration, login, and authentication.
    - Product API: Manage CRUD operations for products.
    - Cart API: Handle shopping cart functionality.
    - Order API: Manage order placement and retrieval.

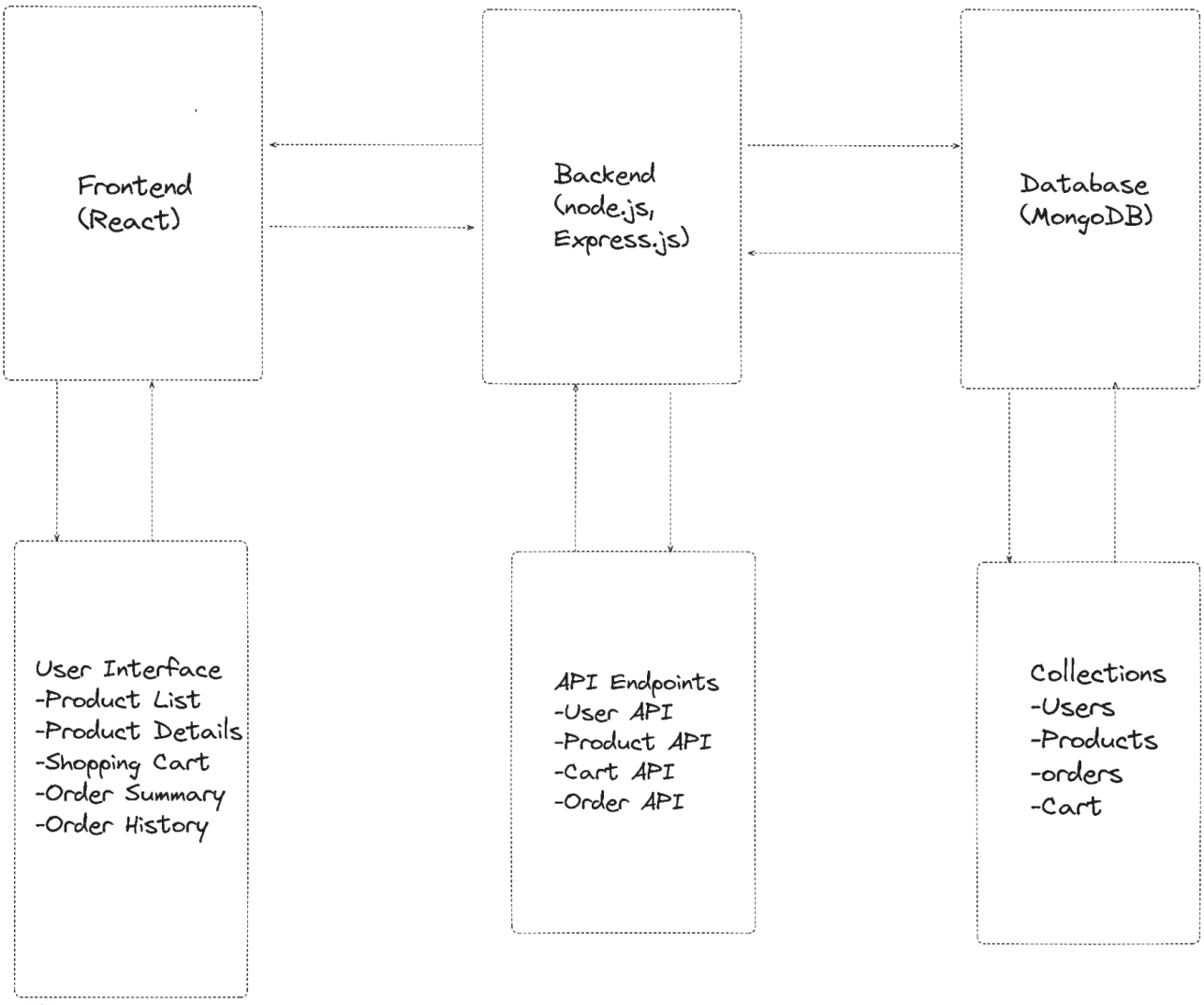
Additional Features

1. **User Profile:** Allow users to update their profile information and view their order history.
2. **Search and Filter:** Implement search functionality to find products and filters to narrow down product lists.
3. **Responsive Design**: Ensure the application is mobile-friendly and works well on various screen sizes.

INNOMATICS
RESEARCH LABS

SYSTEM DESIGN

Architecture Diagram:



Components

1. **Frontend:**
    - React.js: A JavaScript library for building user interfaces.
    - Redux: For state management.
    - React Router: For routing.
2. **Backend:**
    - Express.js: A minimal and flexible Node.js web application framework.
    - MongoDB: A NoSQL database for storing product, user, and order data.
    - Mongoose: An ODM (Object Data Modeling) library for MongoDB and Node.js.

Data Flow

1. **User Requests:** The user interacts with the frontend React application.
2. **API Calls:** The frontend makes API calls to the backend Express server.
3. **Database Operations:** The backend processes the requests, interacts with the MongoDB database, and returns the necessary data.
4. **Response:** The backend sends the processed data back to the frontend, which updates the UI accordingly.

STEP-BY-STEP IMPLEMENTATION GUIDE

1. Set Up the Project

- Initialize the Frontend:
    - Create a new React application using Create React App:

      ```
      npx create-react-app ecommerce-frontend
      cd ecommerce-frontend
      ```

    - Install necessary dependencies: Redux, React Router, Axios.

      ```
      npm install redux react-redux react-router-dom axios
      ```

- Initialize the Backend:
    - Set up a Node.js project:
      ```
      mkdir ecommerce-backend
      ```
    - `cd ecommerce-backend`
    - `npm init -y`

    - Install necessary dependencies: Express, Mongoose, bcrypt (for password hashing), JWT (for authentication).

      ```
      npm install express mongoose bcryptjs jsonwebtoken
      ```

- Set Up MongoDB:
    - Create a MongoDB cluster using MongoDB Atlas.
    - Create necessary collections: Users, Products, Orders, Cart.

2. Implement User Authentication

1. **Frontend:**
    - Create registration and login forms.
    - Set up Redux for managing authentication state.
    - Implement authentication actions and reducers.
2. **Backend:**
    - Create user schema and model using Mongoose.
    - Implement registration and login endpoints.
    - Use bcrypt for hashing passwords.
    - Use JWT for generating and verifying tokens.

3. Implement Product Management

1. **Backend:**
    - Create product schema and model using Mongoose.
    - Implement CRUD operations for products (Create, Read, Update, Delete).
2. **Frontend:**
    - Create components for displaying product lists and product details.
    - Implement actions and reducers for fetching and managing products.

4. Implement Shopping Cart

1. **Frontend:**
    - Create a shopping cart component.
    - Implement actions and reducers for managing cart state.
2. **Backend:**
    - Create endpoints for adding, updating, and removing items from the cart.
    - Store cart data in the database.

5. Implement Order Processing

1. **Frontend:**
    - Create order summary and checkout components.
    - Implement actions and reducers for placing orders.
2. **Backend:**
    - Create order schema and model using Mongoose.
    - Implement endpoints for creating and retrieving orders.

6. Implement Order History

1. **Frontend:**
    - Create an order history component.
    - Implement actions and reducers for fetching past orders.
2. **Backend:**
    - Implement an endpoint for retrieving user-specific orders.

MOCK PRODUCTS DATASET

implement the Products API, a mock dataset of products is provided. This dataset can be downloaded and used to populate the Products collection in MongoDB.

Download the Dataset

You can download the mock products dataset from the following link:

Mock Products Dataset

How to Use the Dataset:

1. **Download the Dataset:**
    - Download the JSON file from the provided link and save it to your local machine.
2. **Import the Dataset into MongoDB:**
    - Use MongoDB Atlas or a script to import the dataset into your MongoDB database.

SUBMISSION

You are required to create a project folder containing all the necessary files for the frontend and backend. The project folder should include:

1. **Frontend:**
    - React.js components
    - CSS/SCSS files for styling
    - Configuration files (e.g., package.json)
2. **Backend:**
    - Express.js application
    - MongoDB schema and models
    - API routes and controllers
    - Configuration files (e.g., package.json)
3. **Documentation:**
    - README.md file with instructions on how to set up and run the project
    - API documentation

Note: Ensure that your submission includes all necessary dependencies and configuration files to enable smooth setup and execution of the project.

Cheers! All the best!