**Spend-O-Cart: Online Shopping Platform Using MERN Stack**

**PROJECT REPORT**

*by*  
**HEMA T G**

**PROJECT DESCRIPTION**

Spend-O-Cart is a full-stack e-commerce application built using the MERN stack (MongoDB, Express.js, React.js, and Node.js). The platform offers a seamless online shopping experience with features like product management, shopping cart functionality, order placement. Spend-O-Cart allows users to browse products, add items to their cart.

The backend, powered by Node.js and Express, handles server-side operations and integrates with MongoDB for data storage. The frontend, developed using React, offers an intuitive and responsive user interface. Spend-O-Cart aims to provide a scalable and modern solution for online retail, incorporating best practices in web development and e-commerce functionality.

**PROJECT DETAILS**

**Spend-O-Cart MERN project setup:**

Spend-O-Cart project is organised into folders for backend (server) and frontend (client).

A close up of words

Description automatically generated

**Backend setup:**

In backend folder, initialize a new Node.js project using following commands.

cd backend

npm init

Install necessary dependencies for backend development:

* **Express**: For creating server routes.
* **Mongoose**: To interact with MongoDB.
* **dotenv**: For environment variables.
* **cors**: To handle Cross-Origin Resource Sharing.

npm install express

npm install mongoose

npm install dotenv

npm install cors

Environment Variable Configuration:

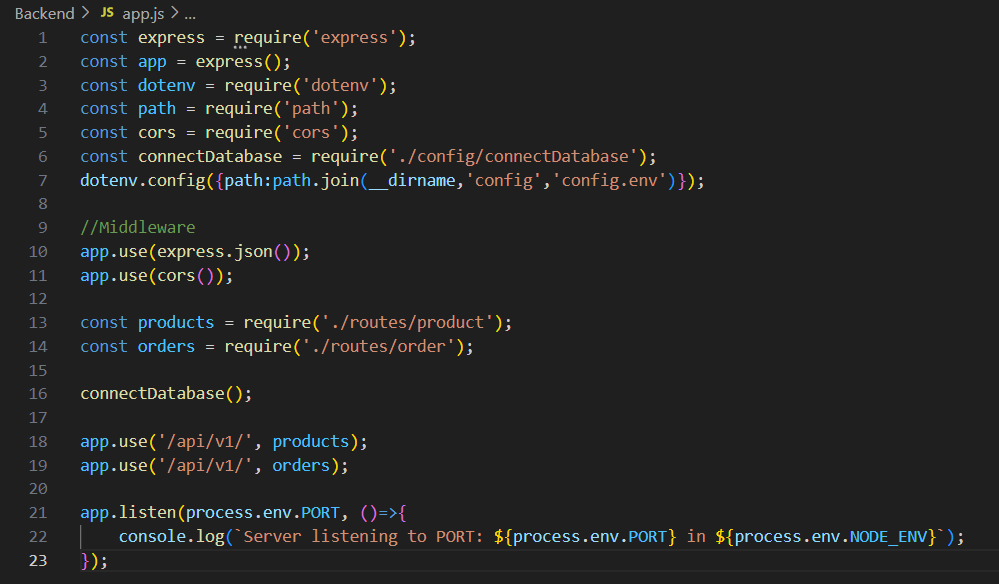
Create Config.env file in backend folder, set port number and production environment.

A screen shot of a computer

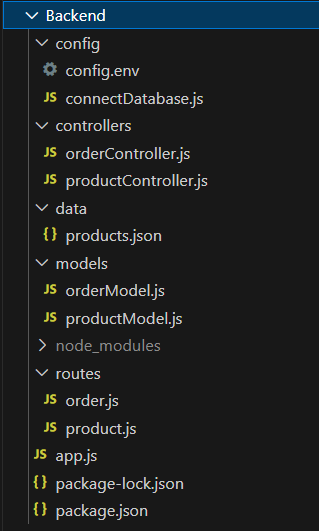
Description automatically generated

Server Entry Point:

***app.js***



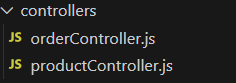
Create the following folders and files for server- models, routes, controllers, data:



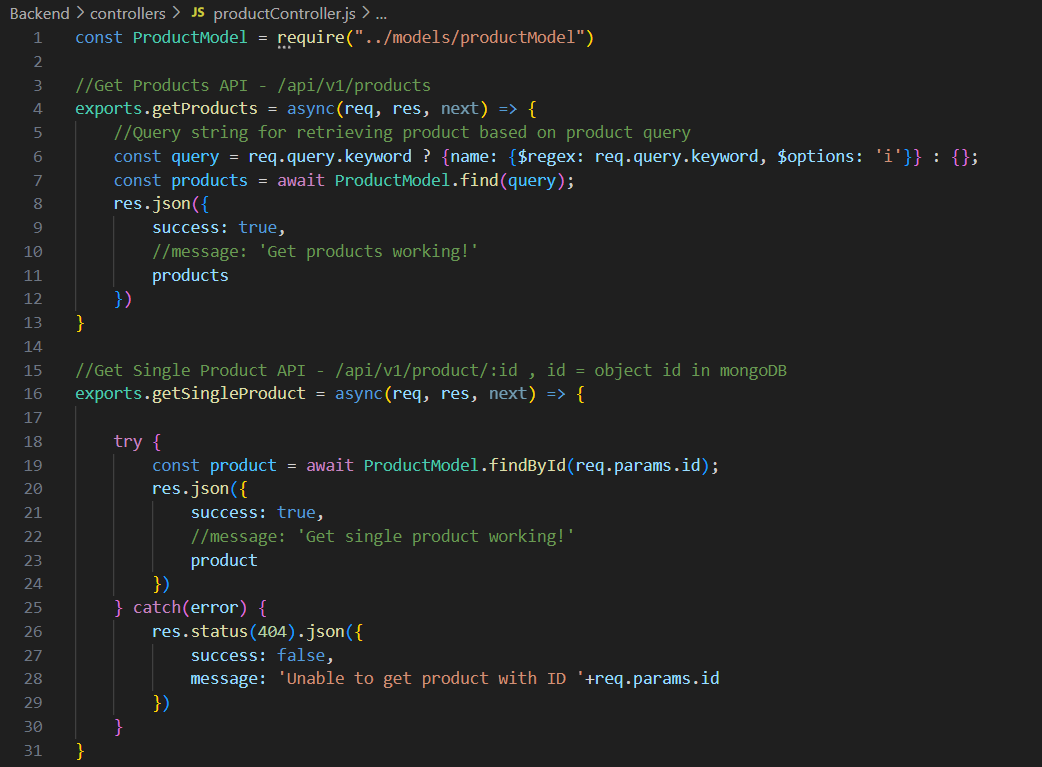
**Model:** The model will handle data-related logic.

**Route:** The routes define how different endpoints interact with the controller.

**Controller**: Acts as an intermediary between the Model and the View. It processes incoming requests, interacts with the Model, and returns the View.



***productController.js***

******

***orderController.jsA screen shot of a computer code

Description automatically generated***

***REST API Integration and Working:***

***A screenshot of a computer

Description automatically generated***

***A screenshot of a computer program

Description automatically generated***

***A screenshot of a computer

Description automatically generated***

**Frontend setup:**

In frontend folder, initialize a new React.js project using following commands.

cd frontend

npx create-react-app .

Install necessary dependencies for frontend development:

npm install bootstrap

npm install react-router-dom

npm install react-toastify

Environment Variable Configuration:

Create .env file in frontend folder, set React Application API URL.

A screen shot of a computer

Description automatically generated

Client Entry point

***App.js***

***A screen shot of a computer code

Description automatically generated***

**React Project Structure:**

**A screenshot of a computer

Description automatically generated**

**React View pages:**

***Home.js***

**A computer screen shot of text

Description automatically generated**

Home page displaying products

***ProductDetails.js***

***A screen shot of a computer program

Description automatically generated***

******

***A screen shot of a computer code

Description automatically generated***

Product Details Page

***Cart.js***

***A screen shot of a computer program

Description automatically generated***

***A screen shot of a computer program

Description automatically generated***

***A computer screen with many colorful text

Description automatically generated***

***A computer code on a black background

Description automatically generated***

Cart details and order processing page

**React Reusable Components:**

***Header.js***

******

***ProductDetails.js***

***A screen shot of a computer code

Description automatically generated***

***Search.js***

***A screen shot of a computer program

Description automatically generated***

***Footer.js***

***A screenshot of a computer program

Description automatically generated***

***A computer screen with text

Description automatically generated***

***A computer screen with text

Description automatically generated***

**Connecting Frontend and Backend:**

In our React app, we use the fetch () method to make requests to the back end. Since, the backend have routes defined then the Client can interact with Server.

***Making a GET Request Using fetch ()***

A screen shot of a computer code

Description automatically generated

fetch(http://localhost:8000/api/v1/products) makes a request to the backend server at the specified endpoint.

Then, backend server responds with products list. And, the products data is displayed on the front end.

***Making a POST Request Using fetch ()***

A screen shot of a computer code

Description automatically generated

fetch(http://localhost:8000/api/v1/order), {method: 'POST', headers, body}) sends a POST request to the backend server to place order for cart items. Then server handles the request by parsing the JSON body and adding order details to the database.

With this setup, we can effectively connect the front end and back end using fetch () method for fetching and sending data.

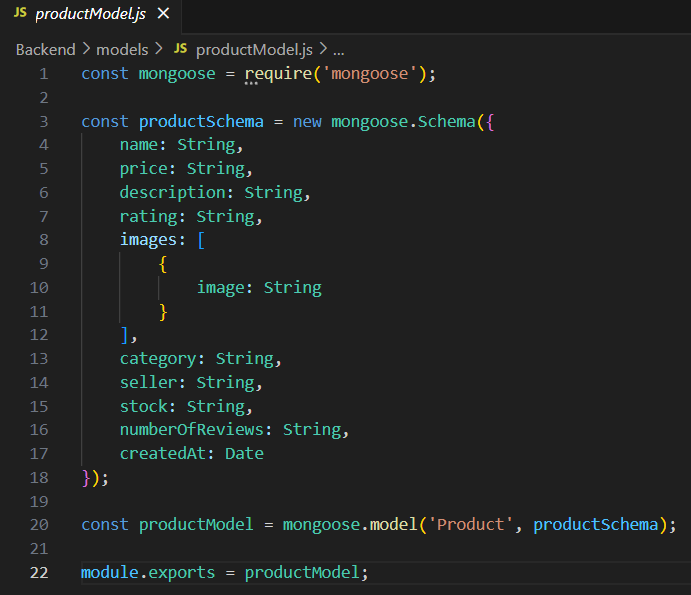
**Database Setup (MongoDB):**

For setting up database, we used MongoDB Atlas.

And, created collections for products and orders.

Then, used Mongoose schemas to define the structure of each collection in backend.

***productModel.js***



***orderModel.js***

***A screen shot of a computer program

Description automatically generated***

**Running the Project:**

Execute the following command in backend and frontend terminal:

npm start

Backend

A screen shot of a computer

Description automatically generated

Frontend

A computer screen shot of a computer code

Description automatically generated

**VISUAL HIGHLIGHTS OF PROJECT**

**Home page**

**A screenshot of a website

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A blue background with white text

Description automatically generated**

**Product Details page**

**A computer with a person on the screen

Description automatically generated**

**A screen shot of a phone

Description automatically generated**

**Product Search**

**A screenshot of a website

Description automatically generated**

**A screenshot of a website

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Cart Details page**

**A screenshot of a computer

Description automatically generated**

**Order placement page**

**A screenshot of a computer

Description automatically generated**

**CONCLUSION**

In conclusion, Spend-O-Cart successfully demonstrates the capabilities of the MERN stack for building a modern, full-featured e-commerce platform. By integrating MongoDB for data storage, Express and Node.js for backend operations, and React for the frontend, the project delivers a responsive and scalable solution for online shopping.

**PROJECT LINK**

<https://github.com/Hemagowdham/spendOcart>