ShoppyGlobal-Backend

This repository contains the backend server for ShoppyGlobal, an e-commerce platform. It is built with Node.js and Express, using MongoDB for the database. The API provides user authentication, product management, and shopping cart functionalities.

Table of Contents

Features

- **User Authentication**: Secure user registration and login using JWT (JSON Web Tokens).
- **Product Management**: CRUD operations for managing products.
- **Shopping Cart**: Functionality for users to add, update, view, and remove items from their cart.
- **Authorization**: Middleware to protect specific routes, ensuring only authenticated users can access them.

Technologies Used

- **Node.js**: JavaScript runtime environment.
- **Express.js**: Web application framework for Node.js.
- **MongoDB**: NoSQL database for storing application data.
- **Mongoose**: ODM library for MongoDB and Node.js.
- **jsonwebtoken (JWT)**: For generating and verifying access tokens.
- **bcrypt**: For hashing user passwords.

Getting Started

Follow these instructions to get a copy of the project up and running on your local machine for development and testing.

Prerequisites

- Node.js (v14 or newer)
- npm (usually comes with Node.js)
- MongoDB installed and running, or a MongoDB Atlas connection string.

Installation

1. **Clone the repository:**
   git clone
   https://github.com/anupam1310/ShoppyGlobal-Backend.git
   cd ShoppyGlobal-Backend

2. **Install dependencies:**
   npm install

3. Configure your database:
   Open Server.js and replace the MongoDB connection string with your own. use the pre-existing one
   ```
   // In Server.js
   mongoose.connect('YOUR_MONGODB_CONNECTION_STRING_HERE')
   .then(() => {
      console.log("Connected to Database");
   })
   //...
   ```

Running the Server

Start the development server with nodemon (which will automatically restart on file changes):

npm start

The server will start on http://localhost:3050.

API Endpoints Guide

Here is a detailed breakdown of the available API routes.

Authentication

- The /cart routes are protected.
- To access them, you first need to log in via POST /login to receive a JWT token.
- This token must be included in the Authorization header for all subsequent requests to

protected routes, prefixed with Bearer .
    - **Header Key**: Authorization
    - **Header Value**: Bearer <your_jwt_token>

User Routes

1. Register User

- **Endpoint**: POST /register
- **Description**: Creates a new user account.
- **Body (raw/json)**:
  ```
  {
    "email": "testuser@example.com",
    "password": "yourstrongpassword"
  }
  ```

2. Login User

- **Endpoint**: POST /login
- **Description**: Authenticates a user and returns a JWT token.
- **Body (raw/json)**:
  ```
  {
    "email": "testuser@example.com",
    "password": "yourstrongpassword"
  }
  ```

- **Successful Response**:
  ```
  {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  }
  ```

Product Routes

1. Create a Product

- **Endpoint**: POST /product
- **Description**: Adds a new product to the database.
- **Body (raw/json)**:
  ```
  {
    "name": "Laptop Pro",
    "price": 1200,
    "discription": "A high-performance laptop for professionals.",
    "stock": 50
  }
  ```

2. Get All Products

- **Endpoint**: GET /product
- **Description**: Retrieves a list of all products.
- **Body**: None

3. Get Product by ID

- **Endpoint**: GET /product/:id
- **Description**: Retrieves a single product by its unique ID.
- **URL Parameter**:
  - id: The _id of the product you want to fetch (e.g., 60c72b2f5f1b2c001f8e4d3a).
- **Example URL**: http://localhost:3050/product/60c72b2f5f1b2c001f8e4d3a
- **Body**: None

Cart Routes (Protected)

Remember: All these routes require the Authorization: BEARER <token> header.

1. Get User's Cart

- **Endpoint**: GET /cart
- **Description**: Fetches all items in the currently logged-in user's cart.
- **Body**: None

2. Add Item to Cart

- **Endpoint**: POST /cart
- **Description**: Adds a product to the user's cart. If the product already exists, it increases the quantity.
- **Body (raw/json)**:

```
{
  "productId": "68da66d29bedf6e9af4ffaba",
  "quantity": 1
}
```

  Note: productId is the _id of a product from the products collection.

3. Update Cart Item Quantity

- **Endpoint**: PUT /cart/:id
- **Description**: Updates the quantity of a specific item in the cart.
- **URL Parameter**:
  - id: The _id of the cart item (not the product ID).
- **Example URL**: http://localhost:3050/cart/60c72b8d5f1b2c001f8e4d3b
- **Body (raw/json)**:

```
{
```

```
  "quantity": 3
}
```

4. Remove Item from Cart

- **Endpoint**: DELETE /cart/:id
- **Description**: Removes an item completely from the user's cart.
- **URL Parameter**:
  - id: The _id of the cart item.
- **Example URL**: http://localhost:3050/cart/60c72b8d5f1b2c001f8e4d3b
- **Body**: None

Testing with Thunder Client

1. **Register a user**: Send a POST request to /register with an email and password.
2. **Log in**: Send a POST request to /login with the same credentials.
3. **Copy the Token**: From the login response, copy the JWT token value.
4. **Access Protected Routes**: For any cart route:
   - Go to the **Auth** tab.
   - Select **Bearer**.
   - Paste the copied token into the "Token" field.
   - Now you can send requests to /cart, /cart/:id, etc.
5. **Create a Product**: Send a POST request to /product to create a product. Copy its _id from the response.
6. **Add to Cart**: Send a POST request to /cart, using the product _id you just copied in the request body.