Flipr Labs – Backend Development Task

Statement- You are tasked with developing the backend of an e-commerce platform.

In this task, you will build the core functionalities that allow an user to manage products and to interact with these products through purchasing and shipping processes. The goal is to create a robust and scalable backend that can handle various operations required in a typical e-commerce environment.

Authentication

1. Sign Up

- **Endpoint**: `/signup`
- **Description**: This endpoint allows a new user to register on the platform. The user needs to provide basic information such as name, email, password, and optionally, an address. The system should ensure that the email provided is unique and securely store the password.
- **Things to Keep in Mind**:
    - Validate the input data (e.g., email format, password strength).
    - Ensure the email provided is unique and not already registered.
    - Hash the password using a secure hashing algorithm (e.g., bcrypt) before storing it in the database.
    - If the signup is successful, return a success message along with the customer's ID.
    - If the email is already registered, return an error message indicating the conflict.

2. Sign In

- **Endpoint**: `/signin`
- **Description**: This endpoint allows an existing customer to log in to the platform using their email and password. The system should verify the credentials and provide a session token if the login is successful.
- **Things to Keep in Mind**:
    - Validate the input data (e.g., email format).
    - Check if the email exists in the system and if the provided password matches the stored hashed password.
    - If the login is successful, generate and return a session token (e.g., JWT) along with a success message.
    - If the credentials are incorrect, return an error message indicating the failure.

Product Management

1. Add Product

- **Endpoint**: `/addproduct`
- **Description**: This endpoint allows the user to add a new product to the platform. The user must provide product details such as name, description, price, and category. The product will then be added to the catalog for customers to view and purchase.
- **Things to Keep in Mind**:
    - Validate the input data (e.g., price should be a positive number).
    - Ensure that all required fields are provided.
    - If the product is added successfully, return a success message along with the product ID.
    - Handle cases where input data is invalid by returning an appropriate error message.

2. Update Product

- **Endpoint**: `/updateproduct/:productId`
- **Description**: This endpoint allows the admin to update details of an existing product. The admin can update fields such as name, description, price, and category. The `productId` parameter in the URL specifies the product to be updated.
- **Things to Keep in Mind**:
    - Validate the input data and ensure that the product ID exists.
    - Only update the fields that are provided in the request.
    - If the product is updated successfully, return a success message.
    - Handle cases where the product ID is not found or input data is invalid by returning appropriate error messages.

3. Delete Product

- **Endpoint**: `/deleteproduct/:productId`
- **Description**: This endpoint allows the admin to delete an existing product from the platform. The `productId` parameter in the URL specifies the product to be removed.
- **Things to Keep in Mind**:
    - Verify that the product ID exists before attempting to delete.
    - If the product is deleted successfully, return a success message.
    - Handle cases where the product ID is not found by returning an appropriate error message.

4. Get All Products

- **Endpoint**: `/products`
- **Description**: This endpoint retrieves a list of all products available on the platform. Users can use this endpoint to view the entire product catalog, which includes details like product name, description, price, and category.
- **Things to Keep in Mind**:
    - Return an appropriate message if no products are found.

Cart Management

1. Add Product to Cart

- **Endpoint**: `/cart/add`
- **Description**: This endpoint allows a customer to add a product to their shopping cart. The customer specifies the product ID to add to the cart. If the product is already in the cart, the endpoint should return an error indicating that the product already exists in the cart. No quantity needs to be specified, as each product is only allowed once in the cart.
- **Things to Keep in Mind**:
    - Validate that the product ID exists.
    - Only one quantity of each product is allowed in the cart; any attempt to add the same product again should result in an error.
    - Ensure the product is in stock before adding to the cart.
    - Return a success message if the product is added successfully, and an appropriate error message if the product already exists in the cart.

2. Delete Product from Cart

- **Endpoint**: `/cart/delete`
- **Description**: This endpoint allows a customer to remove a product from their shopping cart. The customer needs to specify the product ID of the item to be removed.
- **Things to Keep in Mind**:
    - Validate that the product ID exists in the customer's cart.
    - If the product is successfully removed, return the updated cart details.
    - Handle errors such as invalid product ID or product not found in the cart.

3. Get Cart

- **Endpoint**: `/cart`
- **Description**: This endpoint retrieves the details of a customer's shopping cart, including each product's description, quantity, price, and the total amount for the cart. It provides a comprehensive view of all items currently in the cart, along with any relevant product details.
- **Things to Keep in Mind**:

    - Validate that the customer ID is provided and is valid.
    - Retrieve the cart details, including product descriptions and current quantities.
    - Calculate the total amount based on the quantities and prices of the items in the cart.
    - Handle cases where the cart is empty by returning an appropriate message.

Order Management

1. Place Order

- **Endpoint**: `/placeorder`
- **Description**: This endpoint processes the purchase of the items in a customer's cart. It stores the customer's shipping details, generates an order ID, and finalizes the order.
- **Things to Keep in Mind**:
    - Validate that the cart belongs to the customer and that the cart is not empty.
    - Capture and store the customer's shipping details.
    - Generate a unique order ID for the purchase.
    - Return an order confirmation message with the order ID.

2. Get Orders

- **Endpoint**: `/getorders`
- **Description**: This endpoint retrieves a list of orders placed by customer. The response should include order details such as order ID, order date, items purchased, shipping details, and order status.
- **Things to Keep in Mind**:
    - Returns only their own orders based on their customer ID.

Additionally

- Create API documentation using Postman or a similar tool. This should include details about the endpoints, requests, and responses.
- If you don't know how to publish documentation you can use youtube.

Evaluation Criteria

- **Functionality:** All required features are implemented and working correctly.
- **Code Quality:** Clean, organized, and well-documented code.