**Documentation:**

User Routes:

1. Get all users

- Route: `GET /user`

- Description: This route retrieves all users stored in the database.

- Controller Function: `getUsers`

- Response:

- Status Code: 200 OK

- Body: An array of user objects.

2. Get a single user by userId

- Route: `GET /user/:userId`

- Description: This route retrieves a single user based on the provided `userId`.

- Controller Function: `getSingleUser`

- Parameters:

- `userId`: The unique identifier of the user to retrieve.

- Response:

- Status Code:

- 200 OK if the user is found.

- 404 Not Found if the user with the specified `userId` does not exist.

- Body: A JSON object representing the user.

3. Create a new user

- Route: `POST /user`

- Description: This route creates a new user with the provided user data.

- Controller Function: `createUser`

- Request Body: A JSON object containing user information such as username, email, password, etc.

- Response:

- Status Code:

- 201 Created if the user is successfully created.

- 400 Bad Request if the request body is invalid.

- Body: A JSON object representing the created user.

4. Delete a user by userId

- Route: `DELETE /user/:userId`

- Description: This route deletes a user based on the provided `userId`.

- Controller Function: `deleteUser`

- Parameters:

- `userId`: The unique identifier of the user to delete.

- Response:

- Status Code:

- 204 No Content if the user is successfully deleted.

- 404 Not Found if the user with the specified `userId` does not exist.

5. Authentication Token Generation

- Route: `POST /user/login`

- Description: This route generates an authentication token for a user based on provided credentials.

- Controller Function: `getAuthToken`

- Request Body: A JSON object containing user credentials such as username/email and password.

- Response:

- Status Code:

- 200 OK if the authentication is successful.

- 401 Unauthorized if the provided credentials are invalid.

- Body: A JSON object containing the authentication token.

**Product Routes :**

1. Get all products

- Route: `GET product/getProducts`

- Description: This route retrieves all products available in the system.

- Controller Function: `getProducts`

- Response:

- Status Code: 200 OK

- Body: An array of product objects.

2. Get a single product by productId

- Route: `GET product/:productId`

- Description: This route retrieves a single product based on the provided `productId`.

- Controller Function: `getSingleProduct`

- Parameters:

- `productId`: The unique identifier of the product to retrieve.

- Response:

- Status Code:

- 200 OK if the product is found.

- 404 Not Found if the product with the specified `productId` does not exist.

- Body: A JSON object representing the product.

3. Create a new product

- Route: `POST product /getProducts`

- Description: This route creates a new product with the provided product data.

- Controller Function: `createProduct`

- Request Body: A JSON object containing product information such as name, price, description, etc.

- Response:

- Status Code:

- 201 Created if the product is successfully created.

- 400 Bad Request if the request body is invalid.

- Body: A JSON object representing the created product.

4. Delete a product by productId

- Route: `DELETE product/:productId`

- Description: This route deletes a product based on the provided `productId`.

- Controller Function: `deleteProduct`

- Parameters:

- `productId`: The unique identifier of the product to delete.

- Response:

- Status Code:

- 204 No Content if the product is successfully deleted.

- 404 Not Found if the product with the specified `productId` does not exist.

**Cart Routes :**

1. Create Cart

- Route: `POST cart/createCart/:username`

- Description: This route creates a new shopping cart for the specified user.

- Controller Function: `createCart`

- Parameters:

- `username`: The username of the user for whom the cart is being created.

- Response:

- Status Code:

- 201 Created if the cart is successfully created.

- 400 Bad Request if there's an issue with the request.

2. Get Cart by User ID

- Route: `GET cart/:userId`

- Description: This route retrieves the shopping cart for the specified user.

- Controller Function: `getCartByUserId`

- Parameters:

- `userId`: The unique identifier of the user.

- Response:

- Status Code:

- 200 OK if the cart is found.

- 404 Not Found if the cart for the specified user does not exist.

- Body: A JSON object representing the user's shopping cart.

3. Add Product to Cart

- Route: `POST cart/add`

- Description: This route adds a product to the user's shopping cart.

- Controller Function: `addToCart`

- Request Body: A JSON object containing the product details to be added to the cart.

- Response:

- Status Code:

- 200 OK if the product is successfully added to the cart.

- 400 Bad Request if there's an issue with the request.

4. Remove Product from Cart

- Route: `DELETE cart/:userId/:productId`

- Description: This route removes a product from the user's shopping cart.

- Controller Function: `removeFromCart`

- Parameters:

- `userId`: The unique identifier of the user.

- `productId`: The unique identifier of the product to be removed from the cart.

- Response:

- Status Code:

- 204 No Content if the product is successfully removed from the cart.

- 404 Not Found if the specified user or product is not found.