Get Ready Shopping Mall for Men’s Shirt Web Application (Backend Documentation)

Description:-

Get Ready Shopping Mall web application Below, you'll find detailed descriptions of the functionalities and API endpoints available that enable various e-commerce operations,such as Inserting,Updating ,Deleting , fetching , Product Users as well as Cart, We are using authentication for users using jwt token and handling orders. We have used the following technologies in our project JavaScript and Nodejs express and Database Mongodb , cors , express -middlewares . This documentation includes token-based authentication for securing the API endpoints, ensuring that only authenticated users can access protected functionalities.

Here's a detailed guide to understanding the functionalities and API calls that power our exciting project.

Products Examples:-

Product 1:

```
{
  "p_id": 21,
  "p_name": "spykar jeans",
  "p_cost": 1750,
  "p_category": "mens jeans",
  "p_img": "https://spykar.com/collections/men-jeans",
  "p_desc": "spykar jeans"
}
```

Product 2:

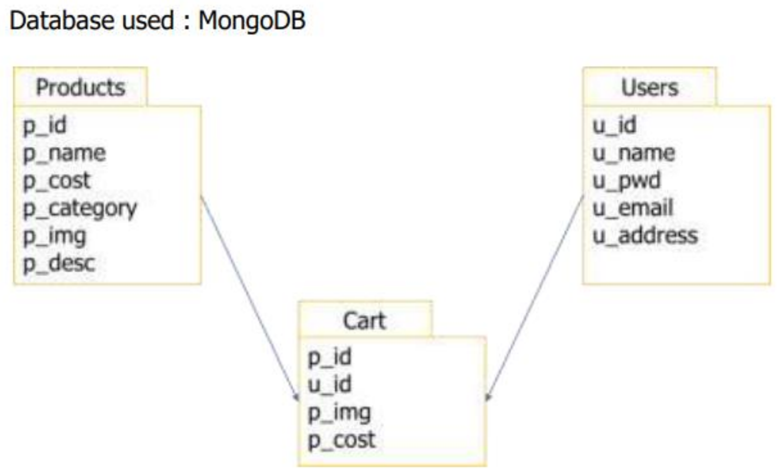
```
{
  "p_id": 24,
  "p_name": "armani jeans",
  "p_cost": 2250,
  "p_category": "mens jeans",
  "p_img": "https://www.armani.com/en-us/armani-exchange/man/clothing/jeans",
  "p_desc": "armani jeans"
}
```

Users Example:-

```
{
  "u_id": 2,
  "u_name": "Vivek",
  "u_pwd": "asd123",
  "u_email": "vivek@gmail.com",
  "u_address": "BlueYonder, Sattava Knowledge City",
"u_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1X25hbWUiOiJTaGl2YW5zaHUiLCJ
1X3B3ZCI6ImFzZDEyMyJ9.NPv969XciUZkV3NQ5L5rAJ2IrZUPvj1RDyHzo8fzGMA",
  "u_contact": "8770976496"
}
```

Cart Example:-

```
{
  "p_id": "2",
  "p_cost": 23000,
  "qty": 1,
  "p_img": "https://m.media-amazon.com/images/I/71ShBw4g6GL._SX679_.jpg",
  "u_name": "Shivanshu"
}
```



Authentication

All endpoints, except for user registration and login, require a valid JSON Web Token (JWT) in the Authorization header of the HTTP request.

Authorization: <token>

User Functionality

1. Create User

- Endpoint: POST /insert/createUser
- Description: Register a new user in the system.
- Request Body: { u_name, u_pwd, u_email ,u_address,u_contact}
- Response:
  - Success: { message: "User created successfully", status: "insert:success" }
  - Failure: { message: "Failed to create user", status: "insert:failed" }

2. Login

- Endpoint: POST /fetch/auth
- Description: Authenticate user credentials and generate a JWT token.
- Request Body: { u_name, u_pwd }
- Response:
  - Success: { message: "Login successful", status: "login:success", token, user_data }
  - Failure: { message: "Invalid credentials", status: "login:failed" }

3. Show All Products

- Endpoint: GET /fetch
- Description: Retrieve all available products.
- Response:

    - Success: { message: "Products fetched successfully", status: "fetch:success", products_array }
    - Failure: { message: "Error fetching products", status: "fetch:error" }

4. Add to Cart

- Endpoint: POST /insert/cartInsert
- Description: Add a product to the user's shopping cart.
- Request Body: { p_id, p_cost, qty, p_img, u_name, u_token}
- Response:
    - Success: { message: "Product added to cart", status: "insert:success" }
    - Failure: { message: "Failed to add product to cart", status: "insert:failed" }

5. Reduce from Cart

- Endpoint: DELETE/delete/cartDelete
- Description: Update or remove a product from the user's cart.
- Request Body (Update): { u_name, p_id }
- Response:
    - Success (Update): { message: "Cart updated successfully", status: "update:success" }
    - Failure (Update): { message: "Failed to update cart", status: "update:failed" }
    - Success (Delete): { message: "Product deleted from cart", status: "delete:success" }

6. Buy Now

- Endpoint: DELETE/cartDelete
- Description: Finalize the purchase and update product quantities.
- Request Body: { u_name }
- Response:
    - Success: { message: "Product quantity updated", status: "update:success" }
    - Failure: { message: "Failed to update product quantity", status: "update:failed" }

Admin Functionality

1. Create Products

- Endpoint: POST /insertproduct
- Description: Add a new product to the store.
- Request Body: { p_name, p_cost, p_category, p_img, p_desc }
- Response:
    - Success: { message: "Product created successfully", status: "insert:success" }
    - Failure: { message: "Failed to create product", status: "insert:failed" }

2. Update Product

- Endpoint: PUT /update
- Description: Modify an existing product's details.
- Request Body: { p_id, p_name,p_cost }
- Response:
    - Success: { message: "Product updated successfully", status: "update:success" }
    - Failure: { message: "Failed to update product", status: "update:failed" }

3. **Delete Product**

- Endpoint: DELETE /delete
- Description: Remove a product from the store.
- Request Body: { p_id }
- Response:
    - Success: { message: "Product deleted successfully", status: "delete:success" }
    - Failure: { message: "Failed to delete product", status: "delete:failed" }

4. **Create User (Admin)**

- Endpoint: POST /insert
- Description: Register a new admin user.
- Request Body: { u_name, u_pwd, u_email }
- Response:
    - Success: { message: "Admin user created successfully", status: "insert:success" }
    - Failure: { message: "Failed to create admin user", status: "insert:failed" }

5. **Delete User (Admin)**

- Endpoint: DELETE /delete
- Description: Remove a user (including admins) from the system.
- Request Body: { u_id }
- Response:
    - Success: { message: "User deleted successfully", status: "delete:success" }
    - Failure: { message: "Failed to delete user", status: "delete:failed" }