API Documentation

v1.0

Table of Contents

1. General Information

This section provides an overview of the API, including its base URL, versioning strategy, supported tools, and key features that make it a developer-friendly backend solution.

1.1 Base URL

The API can be accessed through the following production URL:

```
https://grocery-app-backend-45m3.onrender.com/api/v1
```

If the backend project is running locally on your machine, the API is accessible at:

```
http://localhost:8080/api/v1
```

1.2 API Version

The current version of the API is **v1.0**.
All endpoints are prefixed with `/api/v1`, ensuring backward compatibility and simplifying future updates or version management.

1.3 Postman

Postman is a popular tool for testing and debugging APIs. To assist frontend developers in testing this backend API, a comprehensive Postman collection is available. This collection includes all API endpoints, example requests, and sample responses.

Download the Collection:

The Postman collection is available in JSON format and can be downloaded using the provided icon.

Setting Up Postman for the API:

Follow these steps to configure Postman and test the API:

1. **Import the Postman Collection**:
   - Open the Postman application.
   - Click on the **Import** button (top-left corner).
   - Select the downloaded JSON file of the Postman collection and click **Import**.
2. **Set the Base URL**:
   - Open the collection settings or the individual request in the collection.
   - Replace `{{baseURL}}` with the production or local development URL.
   - Alternatively, create a **Postman environment**:
     - Go to **Environments > Add New**.
     - Add a variable named `baseURL` with the value of production URL.
3. **Authentication**:
   - For secured endpoints, you need to include a valid access token in the request headers.

1.4 Authentication Type

The Grocery App Backend API secures its endpoints using JSON Web Tokens (JWT) for authentication. JWT is a compact and self-contained way to securely transmit information as a JSON object.

Key Details:

- **Token Type**: The API uses Bearer tokens for authentication.
- **Signing Algorithm**: Tokens are signed using the HS512 (HMAC-SHA-512) algorithm, ensuring robust security.
- **Token Header**: Secured endpoints require the `Authorization` header in the format: `Authorization: Bearer <accessToken>`
- **Access Token Expiration**: Access tokens are valid for 24 hours.
- **Refresh Token**: Refresh tokens are valid for 7 days and allow users to obtain new access tokens without re-authenticating.

1.5 Token Expiration

Access Tokens are valid for 24 hours (86400000 milliseconds). Once expired:

- Clients can use the `/refresh-token` endpoint with a valid refresh token to obtain a new access token.
- If the refresh token is also expired, the client must re-authenticate using the `/login` endpoint.

Refresh Tokens are valid for 7 days (604800000 milliseconds). Once expired:

- The user must log in again to generate new access and refresh tokens.
- Refresh tokens should be securely stored by the client (e.g., in HTTP-only cookies or secure storage).

1.6 Response Format

All API responses except the `Unauthorized` and `Forbidden` ones adhere to a standardized structure using the `ApiResponse<T>` class. This approach ensures consistency and readability across all endpoints, whether successful or error responses.

Overview of the ApiResponse Class:

The `ApiResponse<T>` class includes the following fields:

Field Name	Type	Description
status	`int`	HTTP status code (e.g., `200, 404`).
message	`String`	A brief description of the response.
data	`T (Generic Type)`	The main payload of the response
timestamp	`LocalDateTime`	The time the response was generated.
errors	`List<ErrorDetails>`	Optional; detailed error messages, if applicable.

Successful `ApiResponse<T>` Example:

```
{
    "status": 200,
    "message": "User registered. Activation code sent via email.",
    "data": {
        "userId": "673a0d83c2e3a1234d640766",
        "email": "testuser@gmail.com",
        "message": "Please check (testuser@gmail.com) for the activation code"
    },
    "timestamp": "2024-11-17T15:36:44.202192695",
    "errors": null
}
```

Unsuccessful `ApiResponse<T>` Example:

```
{
    "status": 400,
    "message": "Validation failed",
    "data": null,
    "timestamp": "2024-11-17T15:29:30.49744992",
    "errors": [
        {
            "field": "email",
            "errorMessage": "Invalid email format",
            "rejectedValue": "asdasdmail.com"
        }
    ]
}
```

2. Authentication

Authentication is a fundamental component of the Grocery App Backend API. It ensures that only authorized users can access secured endpoints, while public endpoints remain open to all users. The API implements JWT (JSON Web Token) for secure authentication and authorization processes. Below is a detailed explanation of the authentication system, including token management and usage.

2.1 Overview

The authentication process generates two tokens after a successful login:

- **Access Token**: Used for accessing secured endpoints, sent in the `Authorization` header.
- **Refresh Token**: Used to obtain a new access token without re-authentication.

Workflow:

1. **User Login**:
    - Users authenticate by providing valid credentials.
    - Upon successful authentication, both access and refresh tokens are returned.
2. **Token Validation**:
    - The access token is validated with every request to secured endpoints.
    - If the access token has expired, the refresh token is sent to `/refresh-token` to get a new access token.
3. **Session Expiry**:
    - If both access and refresh tokens have expired, the user must re-authenticate by logging in again.

2.2 Authentication Workflow

2.2.1 User Registration and Login

Registration Endpoint:

- Allows users to create an account by providing basic information such as name, email, and password.

Login Endpoint:

- Authenticates the user and generates both an access token and a refresh token. These tokens must be securely stored by the client.

2.2.2 Token Usage

To access secured endpoints, include the JWT token in the `Authorization` header. There are two ways to add the JWT token in Postman:

Option 1: Using the Authorization Tab

- Select the request in Postman.
- Go to the **Authorization** tab.
- Set the **Type** to `Bearer Token.`
- Paste your JWT token (e.g., `eyJhbGciOiJUzUxMiIsnR4I6IkpXVCJ9...`) into the input field.

Option 2: Adding Manually in Headers

- Select the request in Postman.
- Go to the **Headers** tab.

Add a new key-value pair as follows:
`Authorization: Bearer eyJhbGciOiJUzUxMiIsnR4I6IkpXVCJ9...`

2.2.3 Token Expiration

Access Tokens:

- Valid for **24 hours**.
- Once expired:
  - Use the `/refresh-token` endpoint with a valid refresh token to obtain a new access token.
  - If the refresh token is also expired, the user must log in again.

Refresh Tokens:

- Valid for **7 days**.
- Can be used to renew access tokens via the `/refresh-token` endpoint.
- If the refresh token is expired, the user must log in again to generate new tokens.

2.2.4 Public and Secured Endpoints

- **Public Endpoints**: Accessible without a token.
  Example:
  - `GET /api/v1/products`
  - `GET /api/v1/products/{productId}`
- **Secured Endpoints**: Require a valid token.
  Example:
  - `GET /api/v1/cart`
  - `POST /api/v1/orders`

2.3 Common Authentication Error Codes

Code	Description	Example Scenario
`401`	Unauthorized	Missing or invalid token.
`403`	Forbidden	User lacks the required permissions.
`400`	Bad Request	Invalid login credentials.

Important Note before the "Endpoints Overview" Section

Requesting any other endpoint with any other HTTP method except the provided below, in the third section, will cause responses with no response body with **`Forbidden 403`** status code. Only the endpoints when used with given particular HTTP methods covered in this section will work properly.

3. Endpoints Overview

This section provides detailed documentation for all API endpoints. Each endpoint includes the URL, description, request/response models, and example payloads.

3.1 Auth

The **Auth** endpoints handle all authentication-related operations, including user registration, account activation, login, password reset, and token management.

3.1.1 Register User

URL: `{{baseURL}}/auth/register`
Method: `POST`

This endpoint allows a new user to create an account by providing their username, email, and password. Upon successful registration, an activation code is sent to the provided email address. The user must activate their account using the code before accessing any secured endpoints.

- Public endpoint (no JWT required).
- The activation code sent to the provided email address is valid for **15 minutes**. If the code expires, a new one can be requested by attempting to register again with the same email.

No path variable or header is required.

Request Body Field	Type	Mandatory
username	String	Yes
email	String	Yes
password	String	Yes

Example Request:

```
{
   "username": "testuser",
   "email": "test-user@gmail.com",
   "password": "12345678"
}
```

Example Response:

```
{
   "status": 200,
   "message": "User registered successfully. Activation code sent via email.",
   "data": {
       "userId": "673a2494c2e3a2636d640767",
       "email": "test-user@gmail.com",
       "message": "Please check email (test-user@gmail.com) for activation code."
   },
   "timestamp": "2024-11-17T17:15:01.505630637",
```

```
    "errors": null
}
```


3.1.2 Activate Account

URL: `{{baseURL}}/auth/activate`

Method: `PATCH`

This endpoint allows users to activate their account using the activation code sent to their email after the registration. The activation code must be valid and not expired. Once activated, the user can log in and access their tokens to access the secured resources.

- Public endpoint (no JWT required).
- The user must provide their email and the activation code.
- If the activation code has expired, the user must register again to receive a new code.

No path variable or header is required.

Request Body Field	Type	Mandatory
email	String	Yes
activationCode	String	Yes

Example Request:


```
{
    "email": "test-user@gmail.com",
    "activationCode": "609040"
}
```


Example Response:


```
{
    "status": 200,
    "message": "Account activated successfully",
    "data": null,
    "timestamp": "2024-11-17T17:27:16.893505158",
    "errors": null
}
```


3.1.3 Login User

URL: `{{baseURL}}/auth/login`

Method: `POST`

This endpoint authenticates users by validating their credentials (username and password). Upon successful authentication, it returns an access token and a refresh token. Access token must be used in the `Authorization` header for all secured endpoints. Refresh token, on the other hand, might be used for longer authentication sessions across the application context.

- Public endpoint (no JWT required).
- The access token returned from this endpoint is valid for **24 hours**,
- If the user enters invalid credentials, they will receive a `401 Unauthorized` response.

No path variable or header is required.

Request Body Field	Type	Mandatory
username	String	Yes
password	String	Yes

Example Request:

```
{
    "username": "testuser",
    "password": "12345678"
}
```

Example Response:

```
{
    "status": 200,
    "message": "Logged in successfully.",
    "data": {
        "username": "testuser",
        "accessToken": "eyJhbGciOiJIUzUxMiJ9.eyJ0eXBlIjoiYWNjZXNzIiwiciIjo...",
        "refreshToken": "eyJhbGciOiJIUzUxMiJ9.eyJ0eXBlIjoicmVmcmVzaCIImNhV..."
    },
    "timestamp": "2024-11-20T14:37:14.935221727",
    "errors": null
}
```

3.1.4 Refresh Token

URL: `{{baseURL}}/auth/refresh-token`

Method: `POST`

This endpoint generates a new access token and a refresh token. It allows the client to maintain a seamless user session without requiring re-authentication when the access token expires.

- Public endpoint (no JWT required).
- If the user enters invalid credentials, they will receive a `401 Unauthorized` response.
- The access token returned from this endpoint is valid for **24 hours**.
- The refresh token returned from this endpoint is valid for **7 days**. Before the refresh token expires, this endpoint might be requested from the client side if the user should be logged in for a longer time.

No path variable or header is required.

Request Body Field	Type	Mandatory
refreshToken	String	Yes

Example Request:

```
{
    "refreshToken": "eyJhbGciOiJIUzUxMiJ9abcJzdWIiOiJc3VkbGUi...",
}
```

Example Response:

```
{
    "status": 200,
    "message": "Token refreshed successfully.",
    "data": {
        "accessToken": "eyJhbGciOiJIUzUxMiJ9.eyJ0eXBlIjabcWNjZXNzIiwiciIjo...",
        "refreshToken": "eyJhbGciOiJIUzUxMiJ9.eyJ0eXBlIjoixyzmcmVzaCIImNhV..."
    },
    "timestamp": "2024-11-20T14:37:14.935221727",
    "errors": null
}
```

3.1.5 Forgot Password

URL: `{{baseURL}}/auth/forgot-password`

Method: `POST`

This endpoint initiates the password reset process by sending a reset code to the user's registered email. The user can then use this code to reset their password.

- Public endpoint (no JWT required).
- The email provided must be associated with an existing account.

No path variable or header is required.

Request Body Field	Type	Mandatory
email	String	Yes

Example Request:

```
{
   "email": "test-user@gmail.com"
}
```

Example Response:

```
{
   "status": 200,
   "message": "Password reset code sent successfully.",
   "data": {
       "message": "Password reset code sent to test-user@gmail.com"
   },
   "timestamp": "2024-11-17T17:38:15.211248806",
   "errors": null
}
```

3.1.6 Reset Password

URL: `{{baseURL}}/auth/reset-password`

Method: `PATCH`

This endpoint allows users to reset their password by providing their email, reset code sent to their email and a new password.

- Public endpoint (no JWT required).
- The reset code is valid for **15 minutes**.
- If the reset code is invalid or expired, the user must request a new code using the **Forgot Password** endpoint.

No path variable or header is required.

Request Body Field	Type	Mandatory
email	String	Yes
resetPasswordCode	String	Yes
newPassword	String	Yes

Example Request:

```
{
   "email": "test-user@gmail.com",
   "resetPasswordCode": "243643",
   "newPassword": "135798642"
}
```

Example Response:

```
{
   "status": 200,
   "message": "Password reset successfully.",
   "data": null,
   "timestamp": "2024-11-17T17:41:32.397421933",
   "errors": null
}
```


3.2 Cart

The **Cart** endpoints allow authenticated users to manage their shopping cart. Users can retrieve their cart, add items, update quantities, remove specific items, and clear the entire cart.

3.2.1 Get User's Cart

URL: `{{baseURL}}/cart`

Method: `GET`

This endpoint retrieves the current state of the user's shopping cart, including all items, their quantities, and the total price.

- Secured endpoint (JWT token required).
- If the cart is empty, the response will contain an empty list for `cartItems` and a `totalPrice` of `0.0`.

No request body or path variable is required.

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Response:


```
{
   "status": 200,
   "message": "Cart fetched successfully",
   "data": {
       "cartItems": [
           {
               "productId": "672ca178803e6ff577ae0b3a",
               "productName": "Whole Wheat Bread",
               "quantity": 3,
               "price": 2.5
           }
       ],
       "totalPrice": 7.5
   },
   "timestamp": "2024-11-18T10:42:06.427504579",
   "errors": null
}
```


3.2.2 Add Item to Cart

URL: `{{baseURL}}/cart`

Method: `POST`

This endpoint allows authenticated users to add a product to their cart. If the product is already in the cart, its quantity will be updated.

- Secured endpoint (JWT token required).
- A valid product ID and quantity are required.
- If the product does not exist or is unavailable, the API will return an appropriate error.

No path variable is required.

Request Body Field	Type	Mandatory
productId	String	Yes
quantity	int	Yes

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Request:
(with Authorization Header)

```
{
   "productId": "672ca178803e6ff577ae0b3a",
   "quantity": 3
}
```

Example Response:

```
{
   "status": 200,
   "message": "Item added to cart successfully",
   "data": {
       "cartItems": [
           {
               "productId": "672ca178803e6ff577ae0b3a",
               "productName": "Whole Wheat Bread",
               "quantity": 3,
               "price": 2.5
           }
       ],
       "totalPrice": 7.5
   },
   "timestamp": "2024-11-18T10:41:50.33330172",
   "errors": null
}
```

3.2.3 Update Cart Item Quantity

URL: `{{baseURL}}/cart/{productId}`
Method: PATCH
This endpoint allows authenticated users to update the quantity of a specific product in their cart.

- Secured endpoint (JWT token required).
- The `productId` must be a valid product in the user's cart.
- If the product is not in the cart, the API will return a `404 Not Found` error.

Request Body Field	Type	Mandatory
quantity	int	Yes

Path Variable	Type	Mandatory
productId	String	Yes

Header	Value	Mandatory
Authorization	`Bearer <JWT_TOKEN>`	Yes

Example Request:
(with Authorization Header to `/api/cart/672ca178803e6ff577ae0b3a` URL)

```
{
    "quantity": 5
}
```

Example Response:

```
{
    "status": 200,
    "message": "Cart item quantity updated successfully",
    "data": {
        "cartItems": [
            {
                "productId": "672ca178803e6ff577ae0b3a",
                "productName": "Whole Wheat Bread",
                "quantity": 5,
                "price": 2.5
            }
        ],
        "totalPrice": 12.5
    },
    "timestamp": "2024-11-18T11:13:19.444878657",
    "errors": null
}
```

3.2.4 Remove Item From Cart

URL: `{{baseURL}}/cart/{productId}`
Method: DELETE
This endpoint allows authenticated users to remove a specific product from their cart.

- Secured endpoint (JWT token required).
- If the product is not in the cart, the API will return a `404 Not Found` error.

No request body is required.

Path Variable	Type	Mandatory
productId	String	Yes

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Request:
(with Authorization Header to `/api/cart/672ca178803e6ff577ae0b3a` URL)

```
{
    "productId": "672ca178803e6ff577ae0b3a",
}
```

Example Response:

```
{
    "status": 200,
    "message": "Product removed from cart successfully.",
    "data": {
        "cartItems": [], // Product with ID "672ca178803e6ff577ae0b3a" removed.
        "totalPrice": 0.0
```

```
    },
    "timestamp": "2024-11-18T11:22:02.229374109",
    "errors": null
}
```


3.2.5 Clear Cart Items

URL: `{{baseURL}}/cart`

Method: `DELETE`

This endpoint allows authenticated users to remove all items from their cart.

- Secured endpoint (JWT token required).
- If the cart is already empty, the response will confirm the operation.
- This operation is irreversible; the user will need to re-add items if they change their mind.

No request body or path variable is required.

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Response:


```
{
    "status": 200,
    "message": "Cart cleared successfully",
    "data": {
        "cartItems": [],
        "totalPrice": 0.0
    },
    "timestamp": "2024-11-18T11:29:34.831769868",
    "errors": null
}
```


3.3 Order

The **Order** endpoints allow authenticated users to manage their purchase orders. Users can place new orders, retrieve details of specific orders, and view a list of all past orders. These endpoints ensure seamless integration with the cart and payment functionalities.

3.3.1 Place an Order

URL: `{{baseURL}}/orders`

Method: `POST`

This endpoint allows authenticated users to place an order for the items in their cart. Once the order is successfully placed, the cart is cleared, and an order confirmation is returned. A unique order ID in the format `ORD-12345...` is created in this confirmation process. Later on, the orders might be retrieved using these unique IDs.

- Secured endpoint (JWT token required).
- The order will only be processed if the cart contains at least one item.

No path variable is required.

Request Body Field	Type	Mandatory
address	String	Yes

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Request:

```
{
   "address": "Groove St. No: 25, USA"
}
```

Example Response:


```
{
   "status": 200,
   "message": "Order created successfully.",
   "data": {
       "orderId": "ORD-1731929627280",
       "address": "Groove St. No: 25, USA",
       "date": "2024-11-18",
       "totalAmount": 7.5,
       "items": [
           {
               "productId": "672ca178803e6ff577ae0b3a",
               "productName": "Whole Wheat Bread",
               "quantity": 3,
               "price": 2.5
           }
       ]
   },
   "timestamp": "2024-11-18T11:33:47.438014358",
   "errors": null
}
```


3.3.2 Get User's Order by ID

URL: `{{baseURL}}/orders/{orderId}`

Method: `GET`

This endpoint allows authenticated users to retrieve the details of a specific order they have placed.

- Secured endpoint (JWT token required).
- The `orderId` must belong to the authenticated user.

No request body is required.

Path Variable	Type	Mandatory
orderId	String	Yes

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Response:


```
{
   "status": 200,
   "message": "Order fetched successfully.",
   "data": {
       "orderId": "ORD-1732187837593",
       "address": "Groove St. No: 25, USA",
       "date": "2024-11-21",
       "totalAmount": 2.4,
       "items": [
           {
               "productId": "672ca178803e6ff577ae0b39",
```

```
                "productName": "Organic Apple",
                "quantity": 2,
                "price": 1.2
            }
        ]
    },
    "timestamp": "2024-11-21T14:20:05.174722593",
    "errors": null
}
```


3.3.3 Get User's All Orders

URL: `{{baseURL}}/orders`

Method: `GET`

This endpoint allows authenticated users to retrieve a list of all their past orders, including basic details such as order IDs, dates, and total prices.

- Secured endpoint (JWT token required).
- The response includes an array of orders; each order includes a summary of its items.

No request body or path variable is required.

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Response:


```
{
    "status": 200,
    "message": "Orders fetched successfully.",
    "data": {
        "orders": [
            {
                "orderId": "ORD-1732187837593",
                "address": "Groove St. No: 25, USA",
                "date": "2024-11-21",
                "totalAmount": 2.4,
                "items": [
                    {
                        "productId": "672ca178803e6ff577ae0b39",
                        "productName": "Organic Apple",
                        "quantity": 2,
                        "price": 1.2
                    }
                ]
            }
        ]
    },
    "timestamp": "2024-11-21T14:20:35.218536136",
    "errors": null
}
```


3.4 Product

The **Product** endpoints allow users to interact with the product catalog. Users can view all available products, retrieve detailed information about a specific product, and access reviews associated with a product.

3.4.1 Get All Products

URL: `{{baseURL}}/products`
Method: `GET`
This endpoint retrieves a list of all products available in the system and supports optional query parameters for pagination. Pagination can be utilized primarily by client applications to minimize the load on the backend server.

- Public endpoint (no JWT required).
- If no query parameters are provided, the API will return the first 10 products by default (`offset=0`, `limit=10`).
- If `offset` is greater than the total number of products, the response will return an empty list.

No request body, path variable, or header is required.
However, at this endpoint, query parameters can be used.

Query Parameter	Type	Mandatory
offset	int	No
limit	int	No

Example Response: (from the `/api/products?offset&limit` URL)


```
{
   "status": 200,
   "message": "Products retrieved successfully",
   "data": {
       "products": [
           {
               "id": "672ca178803e6ff577ae0b3a",
               "name": "Whole Wheat Bread",
               "description": "Healthy whole wheat bread with no preservatives.",
               "price": 2.5,
               "quantity": 50,
               "imageUrl": "https://res.cloudinary.com/image/products/...",
               "category": "Bakery",
               "brand": "Baker's Delight",
               "weight": "500g",
               "reviews": [
                   {
                       "username": "alice_m",
                       "rating": 3,
                       "comment": "It was okay, a bit dry for my taste.",
                       "date": "2023-11-03"
                   }
               ]
           },
           {
               "id": "672ca178803e6ff577ae0b3d",
               "name": "Tomato",
               "description": "Fresh, organic tomatoes perfect for salads.",
               "price": 0.9,
               "quantity": 150,
               "imageUrl": "https://res.cloudinary.com/image/products/...",
               "category": "Vegetables",
               "brand": "Nature's Farm",
               "weight": "1kg",
               "reviews": [
                   {
                       "username": "mark_z",
                       "rating": 4,
```

```
                    "comment": "Very fresh tomatoes, loved them in my salad.",
                    "date": "2023-11-05"
                }
            ]
        }
    ],
    "totalProducts": 10,
    "totalPages": 1,
    "currentPage": 0
},
"timestamp": "2024-11-19T17:11:41.386428755",
"errors": null
}
```


3.4.2 Get Product by ID

URL: `{{baseURL}}/products/{productId}`

Method: `GET`

This endpoint retrieves detailed information about a specific product by its ID.

- Public endpoint (no JWT required).
- The `productId` must correspond to an existing product.

No request body or header is required.

Path Variable	Type	Mandatory
productId	String	Yes

Example Response:


```
{
    "status": 200,
    "message": "Product retrieved successfully",
    "data": {
        "id": "672ca178803e6ff577ae0b40",
        "name": "Banana",
        "description": "Ripe and sweet bananas full of potassium.",
        "price": 1.1,
        "quantity": 120,
        "imageUrl": "https://res.cloudinary.com/your-cloud-name/image/...",
        "category": "Fruits",
        "brand": "Tropical Harvest",
        "weight": "1kg",
        "reviews": [
            {
                "username": "david_l",
                "rating": 4,
                "comment": "Tasty bananas, ripe and sweet.",
                "date": "2023-11-09"
            }
        ]
    },
    "timestamp": "2024-11-18T19:58:40.712185774",
    "errors": null
}
```


3.5 Review

The **Review** endpoints allow users to interact with reviews associated with products. Users can view existing reviews, add their own reviews, and delete reviews they have written.

3.5.1 Get Product Reviews

URL: `{{baseURL}}/products/{productId}/reviews`

Method: `GET`

This endpoint retrieves all reviews associated with a specific product.

- Public endpoint (no JWT required).
- Reviews include the reviewer's username, rating, and comments.
- If there is no review for the particular product, the API response will return an empty array.

No request body or header is required.

Path Variable	Type	Mandatory
productId	String	Yes

Example Response:


```
{
    "status": 200,
    "message": "Product reviews fetched successfully",
    "data": [
        {
            "username": "susan_b",
            "rating": 5,
            "comment": "High-quality olive oil, perfect for salads and cooking!",
            "date": "2023-11-06"
        },
        {
            "username": "tom_h",
            "rating": 4,
            "comment": "Good flavor, though a bit pricey.",
            "date": "2023-11-07"
        }
    ],
    "timestamp": "2024-11-18T16:18:13.170152603",
    "errors": null
}
```


3.5.2 Add Product Review

URL: `{{baseURL}}/products/{productId}/reviews`

Method: `POST`

This endpoint allows authenticated users to add a review for a specific product.

- Secured endpoint (JWT token required).
- The user must provide a valid `productId` and the `rating` must be between 1 and 5.
- Users can only submit one review per product.

Request Body Field	Type	Mandatory
rating	int	Yes
comment	String	Yes

Path Variable	Type	Mandatory
productId	String	Yes

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Request:
(with Authorization Header to `/api/products/{productId}/reviews` URL)

```
{
    "rating": 3,
    "comment": "Not bad, but could have been better"
}
```

Example Response:

```
{
    "status": 200,
    "message": "Review added successfully",
    "data": {
        "username": "testuser",
        "rating": 3,
        "comment": "Not bad, but could have been better",
        "date": "2024-11-18"
    },
    "timestamp": "2024-11-18T16:21:46.810956324",
    "errors": null
}
```

3.5.3 Delete Product Review

URL: `{{baseURL}}/products/{productId}/reviews`
Method: `DELETE`
This endpoint allows authenticated users to delete their review for a specific product.

- Secured endpoint (JWT token required).
- Users can only delete reviews they have written.

No request body is required.

Path Variable	Type	Mandatory
productId	String	Yes

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Response:

```
{
    "status": 200,
    "message": "Review deleted successfully",
    "data": null,
    "timestamp": "2024-11-18T16:31:02.799014875",
    "errors": null
}
```

3.6 User

The **User** endpoints allow authenticated users to manage their profile information. Users can view their profile details and update specific information, such as their name or contact details.

3.6.1 Get User Profile

URL: `{{baseURL}}/users/profile`

Method: `GET`

This endpoint allows authenticated users to retrieve their profile details including their current cart items and previous order details.

- Secured endpoint (JWT token required).

No request body or path variable is required.

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Response:


```
{
   "status": 200,
   "message": "User profile fetched successfully.",
   "data": {
       "username": "test-user",
       "email": "test-user@gmail.com",
       "password": "$2a$10$abc5OzREh48ylvuo2eUE21eEV6autJeEFmAuhVu...",
       "cart": [
           {
               "productId": "672ca178803e6ff577ae0b3c",
               "productName": "Brown Eggs",
               "quantity": 5,
               "price": 3.2
           }
       ],
       "orders": [
           {
               "orderId": "ORD-1732187837593",
               "address": "Groove St. No: 25, USA",
               "date": "2024-11-21",
               "total": 2.4,
               "orderItems": [
                   {
                       "productId": "672ca178803e6ff577ae0b39",
                       "productName": "Organic Apple",
                       "quantity": 2,
                       "price": 1.2
                   }
               ]
           }
       ]
   },
   "timestamp": "2024-11-21T14:24:10.15831496",
   "errors": null
}
```

3.6.2 Update User Profile

URL: `{{baseURL}}/users/profile`

Method: `PATCH`

This endpoint allows authenticated users to update specific details in their profile, such as their username or password information.

- Public endpoint (no JWT required).
- Returns the user's current profile information.
- After updating their profile information, the user session should be terminated and they should log in to the application again.

No path variable is required.

Request Body Field	Type	Mandatory
username	String	Yes
password	String	Yes

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Request:

```
{
    "username": "testuser-updated",
    "password": "updatedPassword123"
}
```

Example Response:

```
{
    "status": 200,
    "message": "Profile updated successfully.",
    "data": {
        "message": "User profile updated successfully."
    },
    "timestamp": "2024-11-18T16:41:08.922034691",
    "errors": null
}
```

3.7 Wishlist

The **Wishlist** endpoints allow authenticated users to manage their profile information. Users can view their profile details and update specific information, such as their name or contact details.

3.7.1 Get User's Wishlist

URL: `{{baseURL}}/wishlist`

Method: `GET`

This endpoint allows authenticated users to retrieve the products they have added to their wishlist.

- Secured endpoint (JWT token required).
- If the wishlist is empty, the response will contain an empty wishlist array.

No request body or path variable is required.

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Response:


```
{
    "status": 200,
    "message": "Wishlist retrieved successfully.",
    "data": {
        "wishlist": [
            {
                "id": "672ca178803e6ff577ae0b3a",
                "name": "Whole Wheat Bread",
                "description": "Healthy whole wheat bread with no preservatives.",
                "price": 2.5,
                "imageUrl": "https://res.cloudinary.com/image/..."
            }
        ]
    },
    "timestamp": "2024-11-20T20:40:38.577717864",
    "errors": null
}
```


3.7.2 Add Product to User's Wishlist

URL: `{{baseURL}}/wishlist`

Method: `PATCH`

This endpoint allows authenticated users to add a product to their wishlist.

- Secured endpoint (JWT token required).
- A proper product ID should be provided in the request body.

No path variable is required.

Request Body Field	Type	Mandatory
productId	String	Yes

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Request:

```
{
    "productId": "672ca178803e6ff577ae0b3a",
}
```

Example Response:


```
{
    "status": 200,
    "message": "Product added to wishlist successfully.",
    "data": {
        "wishlist": [
            {
                "id": "672ca178803e6ff577ae0b3a",
                "name": "Whole Wheat Bread",
                "description": "Healthy whole wheat bread with no preservatives.",
                "price": 2.5,
                "imageUrl": "https://res.cloudinary.com/image/..."
            }
```

```
        ]
    },
    "timestamp": "2024-11-20T20:40:35.536087643",
    "errors": null
}
```


3.7.3 Remove Product from User's Wishlist

URL: `{{baseURL}}/wishlist/{productId}`

Method: `DELETE`

This endpoint allows authenticated users to remove a specific product from their wishlist.

- Secured endpoint (JWT token required).
- A proper product ID should be provided as a path variable.

No path variable is required.

Path Variable	Type	Mandatory
productId	String	Yes

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Response: (from the `/api/wishlist/672ca178803e6ff577ae0b39` URL)


```
{
    "status": 200,
    "message": "Product removed from wishlist successfully.",
    "data": {
        "wishlist": [
            {
                "id": "672ca178803e6ff577ae0b3a",
                "name": "Whole Wheat Bread",
                "description": "Healthy whole wheat bread with no preservatives.",
                "price": 2.5,
                "imageUrl": "https://res.cloudinary.com/image/..."
            },
            // the product with ID 672ca178803e6ff577ae0b39 removed.
        ]
    },
    "timestamp": "2024-11-20T20:40:35.536087643",
    "errors": null
}
```


3.7.4 Clear User's Wishlist

URL: `{{baseURL}}/wishlist`

Method: `DELETE`

This endpoint allows authenticated users to clear all products from their wishlist.

- Secured endpoint (JWT token required).
- A proper product ID should be provided in the request body.

No request body or path variable is required.

Header	Value	Mandatory
Authorization	Bearer <JWT_TOKEN>	Yes

Example Response:


```
{
    "status": 200,
    "message": "Wishlist created successfully.",
    "data": {
        "wishlist": []
    },
    "timestamp": "2024-11-20T19:34:03.423758366",
    "errors": null
}
```