API Documentation

Base URL

```
http://localhost:5000
```

Authentication

All protected endpoints require a JWT token in the Authorization header:

```
Authorization: Bearer <jwt_token>
```

Response Format

All responses follow this structure:

```
{
  "message": "Response message",
  "data": {}, // Optional data object
  "errors": [] // Optional validation errors array
}
```

Status Codes

- `200` - Success
- `201` - Created
- `400` - Bad Request (validation errors)
- `401` - Unauthorized (invalid/missing token)
- `403` - Forbidden (insufficient permissions)
- `404` - Not Found
- `500` - Internal Server Error

Authentication Endpoints

Register User

POST `/api/auth/register`

Register a new user account.

Request Body:

```
{
  "name": "string (min: 2 chars)", // required
  "email": "string (valid email)", // required
  "password": "string (min: 6 chars)", // required
  "role": "Admin | User" // required
}
```

Success Response (201):

```
{
  "message": "User registered successfully",
  "token": "jwt_token_string",
  "user": {
    "id": "uuid",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "User"
  }
}
```

Error Response (400):

```
{
  "message": "User already exists with this email"
}
```

Login User

POST `/api/auth/login`

Authenticate user and receive JWT token.

Request Body:

```
{
  "email": "string (valid email)", // required
  "password": "string" // required
}
```

Success Response (200):

```
{
  "message": "Login successful",
```

```
  "token": "jwt_token_string",
  "user": {
    "id": "uuid",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "User"
  }
}
```

Error Response (400):

```
{
  "message": "Invalid credentials"
}
```

Get Current User

GET `/api/auth/me`

Get current authenticated user information.

Headers:

```
Authorization: Bearer <jwt_token>
```

Success Response (200):

```
{
  "user": {
    "id": "uuid",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "User"
  }
}
```

Logout User

POST `/api/auth/logout`

Logout current user (client should remove token).

Headers:

```
Authorization: Bearer <jwt_token>
```

Success Response (200):

```
{
  "message": "Logout successful"
}
```

Product Endpoints

Get All Products

GET `/api/products`

Retrieve all products (available to both Admin and User).

Headers:

```
Authorization: Bearer <jwt_token>
```

Success Response (200):

```
{
  "message": "Products retrieved successfully",
  "products": [
    {
      "id": "uuid",
      "name": "Laptop",
      "price": 999.99,
      "stock": 10,
      "description": "Gaming laptop",
      "createdBy": "admin_user_id",
      "createdAt": "2023-12-01T10:00:00.000Z",
      "updatedAt": "2023-12-01T10:00:00.000Z"
    }
  ]
}
```

Get Single Product

GET `/api/products/:id`

Retrieve a specific product by ID.

Headers:

```
Authorization: Bearer <jwt_token>
```

URL Parameters:

- `id` - Product UUID

Success Response (200):

```
{
  "message": "Product retrieved successfully",
  "product": {
    "id": "uuid",
    "name": "Laptop",
    "price": 999.99,
    "stock": 10,
    "description": "Gaming laptop",
    "createdBy": "admin_user_id",
    "createdAt": "2023-12-01T10:00:00.000Z",
    "updatedAt": "2023-12-01T10:00:00.000Z"
  }
}
```

Error Response (404):

```
{
  "message": "Product not found"
}
```

Create Product (Admin Only)

POST `/api/products`

Create a new product (Admin role required).

Headers:

```
Authorization: Bearer <admin_jwt_token>
```

Request Body:

```
{
  "name": "string (min: 2 chars)", // required
  "price": "number (min: 0)", // required
  "stock": "integer (min: 0)", // required
  "description": "string" // optional
}
```

Success Response (201):

```
{
  "message": "Product created successfully",
  "product": {
    "id": "uuid",
    "name": "Laptop",
    "price": 999.99,
    "stock": 10,
    "description": "Gaming laptop",
    "createdBy": "admin_user_id",
    "createdAt": "2023-12-01T10:00:00.000Z",
    "updatedAt": "2023-12-01T10:00:00.000Z"
  }
}
```

Error Response (403):

```
{
  "message": "Access denied. Admin only."
}
```

Update Product (Admin Only)

PUT `/api/products/:id`

Update an existing product (Admin role required).

Headers:

```
Authorization: Bearer <admin_jwt_token>
```

URL Parameters:

- `id` - Product UUID

Request Body:

```
{
  "name": "string (min: 2 chars)", // optional
  "price": "number (min: 0)", // optional
  "stock": "integer (min: 0)", // optional
  "description": "string" // optional
}
```

Success Response (200):

```
{
  "message": "Product updated successfully",
  "product": {
    "id": "uuid",
    "name": "Updated Laptop",
    "price": 1099.99,
    "stock": 15,
    "description": "Updated gaming laptop",
    "createdBy": "admin_user_id",
    "createdAt": "2023-12-01T10:00:00.000Z",
    "updatedAt": "2023-12-01T11:00:00.000Z"
  }
}
```

Delete Product (Admin Only)

DELETE `/api/products/:id`

Delete a product (Admin role required).

Headers:

```
Authorization: Bearer <admin_jwt_token>
```

URL Parameters:

- `id` - Product UUID

Success Response (200):

```
{
  "message": "Product deleted successfully",
  "deletedProduct": {
    "id": "uuid",
    "name": "Laptop",
    "price": 999.99,
    "stock": 10,
```

```
    "description": "Gaming laptop"
  }
}
```

Cart Endpoints

Get User Cart

GET `/api/cart`

Retrieve current user's cart.

Headers:

```
Authorization: Bearer <jwt_token>
```

Success Response (200):

```
{
  "message": "Cart retrieved successfully",
  "cart": {
    "items": [
      {
        "productId": "uuid",
        "name": "Laptop",
        "price": 999.99,
        "quantity": 2
      }
    ],
    "total": 1999.98
  }
}
```

Add Item to Cart

POST `/api/cart/add`

Add a product to the user's cart.

Headers:

```
Authorization: Bearer <jwt_token>
```

Request Body:

```
{
  "productId": "string (uuid)", // required
  "quantity": "integer (min: 1)" // required
}
```

Success Response (200):

```
{
  "message": "Item added to cart successfully",
  "cart": {
    "items": [
      {
        "productId": "uuid",
        "name": "Laptop",
        "price": 999.99,
        "quantity": 2
      }
    ],
    "total": 1999.98
  }
}
```

Error Response (400):

```
{
  "message": "Insufficient stock. Only 5 items available"
}
```

Update Cart Item

PUT `/api/cart/update`

Update quantity of an item in cart (set to 0 to remove).

Headers:

```
Authorization: Bearer <jwt_token>
```

Request Body:

```
{
  "productId": "string (uuid)", // required
```

```
  "quantity": "integer (min: 0)" // required
}
```

Success Response (200):

```
{
  "message": "Cart updated successfully",
  "cart": {
    "items": [
      {
        "productId": "uuid",
        "name": "Laptop",
        "price": 999.99,
        "quantity": 3
      }
    ],
    "total": 2999.97
  }
}
```

Remove Item from Cart

DELETE `/api/cart/remove/:productId`

Remove a specific item from cart.

Headers:

```
Authorization: Bearer <jwt_token>
```

URL Parameters:

- `productId` - Product UUID

Success Response (200):

```
{
  "message": "Item removed from cart successfully",
  "cart": {
    "items": [],
    "total": 0
  }
}
```

Clear Cart

DELETE `/api/cart/clear`

Remove all items from cart.

Headers:

```
Authorization: Bearer <jwt_token>
```

Success Response (200):

```
{
  "message": "Cart cleared successfully",
  "cart": {
    "items": [],
    "total": 0
  }
}
```

Checkout Cart

POST `/api/cart/checkout`

Process cart checkout (updates product stock and clears cart).

Headers:

```
Authorization: Bearer <jwt_token>
```

Success Response (200):

```
{
  "message": "Checkout successful",
  "purchasedItems": [
    {
      "productId": "uuid",
      "name": "Laptop",
      "price": 999.99,
      "quantity": 2
    }
  ],
  "totalAmount": 1999.98,
  "cart": {
    "items": [],
```

```
    "total": 0
  }
}
```

Error Response (400):

```
{
  "message": "Cart is empty"
}
```

Error Handling

Validation Errors

When validation fails, the response includes detailed error information:

Response (400):

```
{
  "message": "Validation failed",
  "errors": [
    {
      "type": "field",
      "msg": "Name must be at least 2 characters",
      "path": "name",
      "location": "body"
    }
  ]
}
```

Authentication Errors

Response (401):

```
{
  "message": "No token, authorization denied"
}
```

Response (401):

```
{
  "message": "Token is not valid"
}
```

Authorization Errors

Response (403):

```
{
  "message": "Access denied. Admin only."
}
```

Example Usage with cURL

1. Register a new admin user

```
curl -X POST http://localhost:5000/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Admin User",
    "email": "admin@example.com",
    "password": "admin123",
    "role": "Admin"
  }'
```

2. Login and get token

```
curl -X POST http://localhost:5000/api/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "admin@example.com",
    "password": "admin123"
  }'
```

3. Create a product (use token from login)

```
curl -X POST http://localhost:5000/api/products \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -d '{
    "name": "Gaming Laptop",
    "price": 1299.99,
    "stock": 15,
    "description": "High-performance gaming laptop"
  }'
```

4. Register a regular user

```
curl -X POST http://localhost:5000/api/auth/register \
  -H "Content-Type: application/json" \
  -d '{
    "name": "John Doe",
    "email": "user@example.com",
    "password": "user123",
    "role": "User"
  }'
```

5. Add item to cart (use user token)

```
curl -X POST http://localhost:5000/api/cart/add \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer USER_JWT_TOKEN" \
  -d '{
    "productId": "PRODUCT_UUID_FROM_STEP_3",
    "quantity": 2
  }'
```

6. Checkout cart

```
curl -X POST http://localhost:5000/api/cart/checkout \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer USER_JWT_TOKEN"
```

Development Notes

- The server runs on port 5000 by default
- JWT tokens expire after 24 hours
- Cart data is stored in memory (resets on server restart)
- User and product data is persisted in JSON files
- All prices are in USD with 2 decimal places
- Stock quantities are whole numbers only