Filipino Food Store E-commerce Platform

A modern e-commerce platform built with Next.js, TypeScript, and Neon, featuring Filipino cuisine products.

API DOCUMENTATION

The API follows RESTful principles and uses JSON for request/response bodies. All endpoints return appropriate HTTP status codes and JSON responses. Authentication is required for protected endpoints using JWT tokens in the Authorization header.

AUTHENTICATION ENDPOINTS

1. User Registration
   Endpoint: POST /api/auth/register
   Description: Creates a new user account in the system. Validates email format and password strength.

   Request Body:
   ```
   {
    "email": "string",
    "password": "string",
    "name": "string"
   }
   ```

   Response:
   ```
   {
    "user": {
     "id": "string",
     "email": "string",
     "name": "string",
     "createdAt": "string"
    },
    "session": {
     "token": "string",
     "expiresAt": "string"
    }
   }
   ```

   Status Codes:
   - 201: Created successfully
   - 400: Invalid input
   - 409: Email already exists

2. User Login

Endpoint: POST /api/auth/login
Description: Authenticates a user and returns a JWT token for subsequent requests.

Request Body:
```
{
  "email": "string",
  "password": "string"
}
```

Response:
```
{
  "user": {
    "id": "string",
    "email": "string",
    "name": "string"
  },
  "session": {
    "token": "string",
    "expiresAt": "string"
  }
}
```

Status Codes:
- 200: Login successful
- 401: Invalid credentials
- 400: Invalid input

PRODUCT ENDPOINTS

1. Get All Products
Endpoint: GET /api/products
Description: Retrieves a paginated list of products with optional filtering and sorting.

Query Parameters:
- category: Filter products by category (e.g., "desserts", "main-dishes")
- search: Search products by name or description
- sort: Sort products by field (e.g., "price", "name", "createdAt")
- page: Page number for pagination (default: 1)
- limit: Items per page (default: 10)

Response:
```
{
  "products": [
    {
      "id": "string",
```

```
      "name": "string",
      "description": "string",
      "price": "number",
      "category": "string",
      "imageUrl": "string",
      "stock": "number",
      "createdAt": "string"
    }
  ],
  "pagination": {
    "total": "number",
    "pages": "number",
    "currentPage": "number",
    "limit": "number"
  }
}
```


Status Codes:
- 200: Success
- 400: Invalid query parameters

2. Get Product by ID
Endpoint: GET /api/products/[id]
Description: Retrieves detailed information about a specific product.

Response:


```
{
  "product": {
    "id": "string",
    "name": "string",
    "description": "string",
    "price": "number",
    "category": "string",
    "imageUrl": "string",
    "stock": "number",
    "ingredients": "string[]",
    "nutritionalInfo": "object",
    "createdAt": "string",
    "updatedAt": "string"
  }
}
```


Status Codes:
- 200: Success
- 404: Product not found

CART ENDPOINTS

1. Add to Cart
   Endpoint: POST /api/cart
   Description: Adds a product to the user's shopping cart. If the product already exists, updates the quantity.

   Request Body:
   ```
   {
    "productId": "string",
    "quantity": "number"
   }
   ```

   Response:
   ```
   {
    "cart": [
      {
       "id": "string",
       "productId": "string",
       "quantity": "number",
       "product": {
         "name": "string",
         "price": "number",
         "imageUrl": "string"
       },
       "subtotal": "number"
      }
    ],
    "total": "number"
   }
   ```

   Status Codes:
   - 200: Success
   - 400: Invalid input
   - 404: Product not found
   - 401: Unauthorized

2. Get Cart
   Endpoint: GET /api/cart
   Description: Retrieves the current user's shopping cart with all items and total.

   Response:
   ```
   {
    "items": [
      {
       "id": "string",
   ```

```
      "productId": "string",
      "quantity": "number",
      "product": {
        "name": "string",
        "price": "number",
        "imageUrl": "string"
      },
      "subtotal": "number"
    }
  ],
  "total": "number"
}
```

Status Codes:
- 200: Success
- 401: Unauthorized

3. Update Cart Item
   Endpoint: PUT /api/cart/[itemId]
   Description: Updates the quantity of a specific item in the cart.

   Request Body:
```
{
  "quantity": "number"
}
```

   Response:
```
{
  "cart": [
    {
      "id": "string",
      "productId": "string",
      "quantity": "number",
      "product": {
        "name": "string",
        "price": "number",
        "imageUrl": "string"
      },
      "subtotal": "number"
    }
  ],
  "total": "number"
}
```

   Status Codes:
   - 200: Success

- 400: Invalid quantity
- 404: Cart item not found
- 401: Unauthorized

4. Remove from Cart
Endpoint: DELETE /api/cart/[itemId]
Description: Removes a specific item from the cart.

Response:
```
{
 "cart": [
  {
   "id": "string",
   "productId": "string",
   "quantity": "number",
   "product": {
    "name": "string",
    "price": "number",
    "imageUrl": "string"
   },
   "subtotal": "number"
  }
 ],
 "total": "number"
}
```

Status Codes:
- 200: Success
- 404: Cart item not found
- 401: Unauthorized

ORDER ENDPOINTS

1. Create Order
Endpoint: POST /api/orders
Description: Creates a new order from the current cart items and processes payment.

Request Body:
```
{
 "items": [
  {
   "productId": "string",
   "quantity": "number"
  }
 ],
 "shippingAddress": {
```

```
    "street": "string",
    "city": "string",
    "state": "string",
    "zipCode": "string",
    "country": "string"
  },
  "paymentMethod": "string"
}
```

Response:

```
{
  "order": {
    "id": "string",
    "userId": "string",
    "items": [
      {
        "productId": "string",
        "quantity": "number",
        "price": "number",
        "subtotal": "number"
      }
    ],
    "shippingAddress": "object",
    "total": "number",
    "status": "string",
    "paymentStatus": "string",
    "createdAt": "string"
  }
}
```

Status Codes:
- 201: Order created
- 400: Invalid input
- 401: Unauthorized
- 402: Payment failed
- 422: Insufficient stock

2. Get User Orders
   Endpoint: GET /api/orders
   Description: Retrieves all orders for the authenticated user.

   Query Parameters:
   - page: Page number (default: 1)
   - limit: Items per page (default: 10)
   - status: Filter by order status

Response:

```
{
  "orders": [
    {
      "id": "string",
      "items": [
        {
          "productId": "string",
          "quantity": "number",
          "price": "number",
          "subtotal": "number"
        }
      ],
      "total": "number",
      "status": "string",
      "paymentStatus": "string",
      "createdAt": "string"
    }
  ],
  "pagination": {
    "total": "number",
    "pages": "number",
    "currentPage": "number",
    "limit": "number"
  }
}
```

Status Codes:
- 200: Success
- 401: Unauthorized