EcoBazaarX API Documentation

Base URL

```
http://localhost:8080
```

Table of Contents

Authentication Endpoints

1. Register New User

POST `/register`

Description: Register a new user account (customer, seller, or admin).

Request Body:

```
{
  "username": "john_doe",
  "password": "SecurePass@2024",
  "email": "john@example.com",
  "firstName": "John",
  "lastName": "Doe",
  "phone": "+91-9876543210",
  "role": "USER"
}
```

Request Fields:

Field	Type	Required	Description
`username`	String	Yes	Unique username (max 50 chars)
`password`	String	Yes	User password (will be BCrypt hashed)

Field	Type	Required	Description
`email`	String	Yes	Unique email address
`firstName`	String	No	User's first name
`lastName`	String	No	User's last name
`phone`	String	No	Contact phone number
`role`	Enum	No	USER, SELLER, or ADMIN (default: USER)

Response (201 Created):

```
{
  "id": 1,
  "username": "john_doe",
  "email": "john@example.com",
  "firstName": "John",
  "lastName": "Doe",
  "phone": "+91-9876543210",
  "role": "USER",
  "createdAt": "2024-11-11T10:30:00"
}
```

Tables Modified:

- `users` - New user record inserted

---

2. Login

POST `/login`

Description: Authenticate user with email/username and password. Returns JWT token.

Request Body:

```
{
  "username": "john_doe",
  "password": "SecurePass@2024"
}
```

Alternative (Email-based login):

```
{
  "email": "john@example.com",
  "password": "SecurePass@2024"
}
```

Request Fields:

Field	Type	Required	Description
`username`	String	No*	Username for login
`email`	String	No*	Email for login
`password`	String	Yes	User password

*At least one of username or email is required.

Response (200 OK):

```
{
  "success": true,
  "message": "Login successful",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "user": {
    "id": 1,
    "username": "john_doe",
    "email": "john@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "role": "USER",
    "createdAt": "2024-11-11T10:30:00"
  }
}
```

Response (401 Unauthorized):

```
{
  "success": false,
  "message": "Invalid credentials"
}
```

Tables Read:

- `users` - User authentication

User Endpoints

3. Get All Products

GET `/all-products`

Description: Fetch all available products from the platform.

Authentication: Not required (public endpoint)

Query Parameters: None

Response (200 OK):


```
[
  {
    "productId": 1,
    "name": "Eco-Friendly Water Bottle",
    "description": "Reusable stainless steel water bottle",
    "category": "Home & Kitchen",
    "price": 29.99,
    "stockQuantity": 50,
    "imageUrl": "https://example.com/image1.jpg",
    "carbonFootprint": 2.5,
    "carbonExplanation": "Manufacturing and shipping carbon footprint",
    "carbonPoints": 30,
    "postedBy": {
      "id": 2,
      "username": "seller_xyz",
      "email": "seller@example.com"
    },
    "createdAt": "2024-11-10T14:20:00",
    "updatedAt": "2024-11-11T09:15:00"
  },
  {
    "productId": 2,
    "name": "Bamboo Cutting Board Set",
    "description": "Set of 3 organic bamboo cutting boards",
    "category": "Kitchen",
    "price": 45.99,
    "stockQuantity": 30,
    "imageUrl": "https://example.com/image2.jpg",
    "carbonFootprint": 3.2,
    "carbonExplanation": "Sustainable bamboo sourced and processed",
    "carbonPoints": 25,
    "postedBy": {
      "id": 2,
      "username": "seller_xyz"
    },
    "createdAt": "2024-11-09T11:45:00",
    "updatedAt": "2024-11-11T09:15:00"
  }
]
```


Tables Read:

- `products` - All product records

Product Endpoints

4. Get Filtered Products

GET `/api/products`

Description: Get products with optional filtering by name, category, price, and carbon footprint.

Authentication: Not required

Query Parameters:

Parameter	Type	Required	Description
`name`	String	No	Product name (partial match)
`category`	String	No	Product category
`minPrice`	BigDecimal	No	Minimum price filter
`maxPrice`	BigDecimal	No	Maximum price filter
`maxCarbon`	BigDecimal	No	Maximum carbon footprint (kg CO_2e)

Example Request:

```
GET /api/products?category=Kitchen&minPrice=10&maxPrice=50&maxCarbon=5
```

Response (200 OK):

```
[
  {
    "productId": 1,
    "name": "Eco-Friendly Water Bottle",
    "category": "Home & Kitchen",
    "price": 29.99,
    "carbonFootprint": 2.5,
    "carbonPoints": 30,
    ...
  }
]
```

Tables Read:

- `products` - Filtered product records

Seller Endpoints

5. Add Product (Seller)

POST `/seller/add-product`

Authentication: Required (Bearer token in Authorization header)

Description: Sellers add new eco-friendly products with carbon footprint details.

Request Body:

```
{
  "name": "Bamboo Coffee Mug",
  "description": "Eco-friendly bamboo and cork coffee mug",
  "category": "Kitchen",
  "price": 18.99,
  "stockQuantity": 100,
  "imageUrl": "https://example.com/mug.jpg",
  "carbonDetails": {
    "material": "bamboo",
    "manufacturingCarbon": 0.5,
    "shippingCarbon": 0.3,
    "packagingCarbon": 0.2,
    "recyclePercentage": 80,
    "country": "India"
  }
}
```

Request Fields:

Field	Type	Required	Description
`name`	String	Yes	Product name
`description`	String	No	Product description
`category`	String	No	Product category
`price`	BigDecimal	Yes	Product price
`stockQuantity`	Integer	No	Available stock
`imageUrl`	String	No	Product image URL
`carbonDetails`	Object	Yes	Carbon footprint details
`carbonDetails.material`	String	Yes	Material type
`carbonDetails.manufacturingCarbon`	Float	Yes	Manufacturing CO_2 (kg)
`carbonDetails.shippingCarbon`	Float	Yes	Shipping CO_2 (kg)
`carbonDetails.packagingCarbon`	Float	No	Packaging CO_2 (kg)
`carbonDetails.recyclePercentage`	Integer	No	Recyclable percentage
`carbonDetails.country`	String	No	Manufacturing country

Response (200 OK):

```
{
  "message": "✅ Product added successfully by seller_xyz | Carbon footprint: 1.0 kg CO₂e | Points: 50"
}
```

Tables Modified:

- `products` - New product record inserted
- `users` - (postedBy relationship)

---

6. Get Seller's Products

GET `/seller/my-products`

Authentication: Required (Bearer token)

Description: Fetch all products posted by the authenticated seller.

Response (200 OK):

```
[
  {
    "productId": 5,
    "name": "Bamboo Coffee Mug",
    "price": 18.99,
    "stockQuantity": 100,
    "carbonFootprint": 1.0,
    "carbonPoints": 50,
    "createdAt": "2024-11-11T10:30:00"
  },
  {
    "productId": 6,
    "name": "Organic Tea Set",
    "price": 35.99,
    "stockQuantity": 45,
    "carbonFootprint": 1.8,
    "carbonPoints": 40,
    "createdAt": "2024-11-10T15:20:00"
  }
]
```

Tables Read:

- `products` - Products filtered by seller (postedBy)

---

7. Update Product (Seller)

PUT `/seller/update-product/{productId}`

Authentication: Required (Bearer token)

Description: Update an existing product details.

Path Parameters:

Parameter	Type	Required	Description
`productId`	Long	Yes	Product ID to update

Request Body:

```
{
  "name": "Bamboo Coffee Mug - Premium",
  "description": "Premium eco-friendly bamboo and cork coffee mug",
  "price": 22.99,
  "stockQuantity": 150,
  "carbonDetails": {
    "material": "bamboo",
    "manufacturingCarbon": 0.5,
    "shippingCarbon": 0.3,
    "packagingCarbon": 0.2
  }
}
```

Response (200 OK):

```
{
  "message": "Product updated successfully!"
}
```

Tables Modified:

- `products` - Product record updated

---

8. Delete Product (Seller)

DELETE `/seller/delete-product/{productId}`

Authentication: Required (Bearer token)

Description: Delete a product from the catalog.

Path Parameters:

Parameter	Type	Required	Description
`productId`	Long	Yes	Product ID to delete

Response (200 OK):

```
{
  "message": "Product deleted successfully!"
}
```

Tables Modified:

- `products` - Product record deleted
- `order_items` - Related order items affected

---

9. Get Seller's Orders

GET `/seller/my-orders`

Authentication: Required (Bearer token)

Description: Fetch all orders for products sold by the seller.

Response (200 OK):

```
[
  {
    "id": 101,
    "user": {
      "id": 3,
      "username": "customer_abc"
    },
    "items": [
      {
        "id": 201,
        "product": {
          "productId": 5,
          "name": "Bamboo Coffee Mug"
        },
        "quantity": 2,
        "subtotal": 37.98,
        "carbonTotal": 2.0
      }
    ],
    "totalPrice": 37.98,
    "totalCarbon": 2.0,
    "status": "PENDING",
    "createdAt": "2024-11-11T11:00:00"
  }
]
```

Tables Read:

- `orders` - Orders containing seller's products
- `order_items` - Order line items

---

10. Update Order Status (Seller)

PUT `/seller/update-status/{orderId}`

Authentication: Required (Bearer token)

Description: Seller updates the status of an order (e.g., from PENDING to SHIPPED).

Path Parameters:

Parameter	Type	Required	Description
`orderId`	Long	Yes	Order ID

Query Parameters:

Parameter	Type	Required	Description
`status`	Enum	Yes	PENDING, CONFIRMED, SHIPPED, DELIVERED, CANCELLED

Example Request:

```
PUT /seller/update-status/101?status=SHIPPED
```

Response (200 OK):

```
{
  "id": 101,
  "status": "SHIPPED",
  "totalPrice": 37.98,
  "totalCarbon": 2.0,
  "createdAt": "2024-11-11T11:00:00"
}
```

Tables Modified:

- `orders` - Order status updated

Admin Endpoints

11. Get All Users

GET `/admin/all-users`

Authentication: Required (Bearer token + ADMIN role)

Description: Retrieve all registered users on the platform.

Response (200 OK):


```
[
  {
    "id": 1,
    "username": "john_doe",
    "email": "john@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "role": "USER",
    "createdAt": "2024-11-11T10:30:00"
  },
  {
    "id": 2,
    "username": "seller_xyz",
    "email": "seller@example.com",
    "role": "SELLER",
    "createdAt": "2024-11-10T09:15:00"
  }
]
```


Tables Read:

- `users` - All user records

---

12. Get All Sellers

GET `/admin/all-sellers`

Authentication: Required (Bearer token + ADMIN role)

Description: Retrieve all sellers on the platform.

Response (200 OK):


```
[
  {
    "id": 2,
    "username": "seller_xyz",
    "email": "seller@example.com",
    "firstName": "XYZ",
    "role": "SELLER",
    "createdAt": "2024-11-10T09:15:00"
  }
]
```

Tables Read:

- `users` - Filtered by role = 'SELLER'

---

13. Get All Customers

GET `/admin/all-customers`

Authentication: Required (Bearer token + ADMIN role)

Description: Retrieve all customers (USER role) on the platform.

Response (200 OK):

```
[
  {
    "id": 1,
    "username": "john_doe",
    "email": "john@example.com",
    "role": "USER",
    "createdAt": "2024-11-11T10:30:00"
  },
  {
    "id": 3,
    "username": "customer_abc",
    "email": "customer@example.com",
    "role": "USER",
    "createdAt": "2024-11-10T14:45:00"
  }
]
```

Tables Read:

- `users` - Filtered by role = 'USER'

---

14. Get All Admins

GET `/admin/all-admins`

Authentication: Required (Bearer token + ADMIN role)

Description: Retrieve all admin users on the platform.

Response (200 OK):

```
[
  {
    "id": 100,
    "username": "admin_master",
```

```
    "email": "admin@ecobazaarx.com",
    "role": "ADMIN",
    "createdAt": "2024-11-01T08:00:00"
  }
]
```

Tables Read:

- `users` - Filtered by role = 'ADMIN'

---

15. Change User Role

PUT `/admin/change-role/{username}`

Authentication: Required (Bearer token + ADMIN role)

Description: Change a user's role from one to another.

Path Parameters:

Parameter	Type	Required	Description
`username`	String	Yes	Username of user to update

Query Parameters:

Parameter	Type	Required	Description
`role`	Enum	Yes	USER, SELLER, or ADMIN

Example Request:

```
PUT /admin/change-role/john_doe?role=SELLER
```

Response (200 OK):

```
{
  "message": "Role for user 'john_doe' updated to: SELLER"
}
```

Tables Modified:

- `users` - User role field updated

---

16. Delete User

DELETE `/admin/delete-user/{id}`

Authentication: Required (Bearer token + ADMIN role)

Description: Delete a user account from the system.

Path Parameters:

Parameter	Type	Required	Description
`id`	Long	Yes	User ID to delete

Response (200 OK):

```
{
  "message": "User deleted successfully (ID: 1)"
}
```

Tables Modified:

- `users` - User record deleted
- Related records (products, orders) affected by cascade

---

17. Update User Details

PUT `/admin/update-user/{id}`

Authentication: Required (Bearer token + ADMIN role)

Description: Update user profile information.

Path Parameters:

Parameter	Type	Required	Description
`id`	Long	Yes	User ID to update

Request Body:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "email": "john.smith@example.com",
  "phone": "+91-9876543210"
}
```

Response (200 OK):

```
{
  "id": 1,
```

```
  "username": "john_doe",
  "email": "john.smith@example.com",
  "firstName": "John",
  "lastName": "Smith",
  "phone": "+91-9876543210",
  "role": "USER",
  "createdAt": "2024-11-11T10:30:00"
}
```


Tables Modified:

- `users` - User details updated

---

18. Get All Orders

GET `/admin/all-orders`

Authentication: Required (Bearer token + ADMIN role)

Description: Retrieve all orders placed on the platform.

Response (200 OK):


```
[
  {
    "id": 101,
    "user": {
      "id": 1,
      "username": "john_doe"
    },
    "items": [
      {
        "id": 201,
        "product": {
          "productId": 1,
          "name": "Eco-Friendly Water Bottle"
        },
        "quantity": 2,
        "subtotal": 59.98,
        "carbonTotal": 5.0
      }
    ],
    "totalPrice": 59.98,
    "totalCarbon": 5.0,
    "status": "DELIVERED",
    "createdAt": "2024-11-11T11:00:00"
  }
]
```


Tables Read:

- `orders` - All order records
- `order_items` - All order line items

Checkout & Orders Endpoints

19. Checkout / Create Order

POST `/api/checkout/add-checkout`

Authentication: Required (Bearer token)

Description: Convert user's cart to an order (creates order with cart items).

Request Body: None (uses authenticated user's cart context)

Response (200 OK):


```
{
  "id": 102,
  "user": {
    "id": 1,
    "username": "john_doe"
  },
  "items": [
    {
      "id": 202,
      "product": {
        "productId": 5,
        "name": "Bamboo Coffee Mug"
      },
      "quantity": 1,
      "subtotal": 18.99,
      "carbonTotal": 1.0
    }
  ],
  "totalPrice": 18.99,
  "totalCarbon": 1.0,
  "status": "PENDING",
  "createdAt": "2024-11-11T12:00:00"
}
```


Tables Modified:

- `orders` - New order created
- `order_items` - Order items inserted
- `cart` - Cart items cleared (if applicable)

---

20. Get User's Orders

GET `/api/checkout/my-orders`

Authentication: Required (Bearer token)

Description: Retrieve all orders placed by the authenticated user.

Response (200 OK):


```
[
  {
    "id": 101,
    "user": {
      "id": 1,
      "username": "john_doe"
    },
    "items": [
      {
        "id": 201,
        "product": {
          "productId": 1,
          "name": "Eco-Friendly Water Bottle"
        },
        "quantity": 2,
        "subtotal": 59.98,
        "carbonTotal": 5.0
      }
    ],
    "totalPrice": 59.98,
    "totalCarbon": 5.0,
    "status": "DELIVERED",
    "createdAt": "2024-11-10T11:00:00"
  },
  {
    "id": 102,
    "items": [...],
    "status": "PENDING",
    "createdAt": "2024-11-11T12:00:00"
  }
]
```


Tables Read:

- `orders` - User's orders (filtered by user_id)
- `order_items` - Order line items

---

21. Update Order Status

PUT `/api/checkout/status/{orderId}`

Authentication: Required (Bearer token)

Description: Update the status of an order.

Path Parameters:

Parameter	Type	Required	Description
`orderId`	Long	Yes	Order ID

Query Parameters:

Parameter	Type	Required	Description
`status`	Enum	Yes	PENDING, CONFIRMED, SHIPPED, DELIVERED, CANCELLED

Example Request:

```
PUT /api/checkout/status/101?status=DELIVERED
```

Response (200 OK):

```
{
  "id": 101,
  "totalPrice": 59.98,
  "totalCarbon": 5.0,
  "status": "DELIVERED",
  "createdAt": "2024-11-10T11:00:00"
}
```

Tables Modified:

- `orders` - Order status updated

Database Tables

1. **users** Table

Stores user account information and authentication details.

```
CREATE TABLE users (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,
  username VARCHAR(50) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  phone VARCHAR(20),
  role VARCHAR(20) DEFAULT 'USER' NOT NULL,
```

```
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL
);
```

Fields:

Field	Type	Constraints	Description
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique user identifier
username	VARCHAR(50)	UNIQUE, NOT NULL	User's login username
password	VARCHAR(255)	NOT NULL	BCrypt hashed password
email	VARCHAR(100)	UNIQUE, NOT NULL	User's email address
first_name	VARCHAR(50)	-	First name
last_name	VARCHAR(50)	-	Last name
phone	VARCHAR(20)	-	Contact phone number
role	VARCHAR(20)	DEFAULT 'USER'	USER, SELLER, or ADMIN
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Account creation timestamp

2. **products** Table

Stores product catalog with carbon footprint data.

```
CREATE TABLE products (
  product_id BIGINT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(100) NOT NULL,
  description TEXT,
  category VARCHAR(50),
  price DECIMAL(10, 2) NOT NULL,
  stock_quantity INT DEFAULT 0,
  image_url VARCHAR(255),
  posted_by BIGINT NOT NULL,
  carbon_footprint DECIMAL(10, 3) DEFAULT 0,
  carbon_explanation TEXT,
  carbon_points INT DEFAULT 0,
  carbon_details_material VARCHAR(100),
  carbon_details_manufacturing_carbon FLOAT,
  carbon_details_shipping_carbon FLOAT,
  carbon_details_packaging_carbon FLOAT,
  carbon_details_recycle_percentage INT,
  carbon_details_country VARCHAR(50),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (posted_by) REFERENCES users(id) ON DELETE CASCADE
);
```

Fields:

Field	Type	Constraints	Description
`product_id`	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique product identifier
`name`	VARCHAR(100)	NOT NULL	Product name
`description`	TEXT	-	Detailed product description
`category`	VARCHAR(50)	-	Product category
`price`	DECIMAL(10, 2)	NOT NULL	Product price
`stock_quantity`	INT	DEFAULT 0	Available stock
`image_url`	VARCHAR(255)	-	Product image URL
`posted_by`	BIGINT	NOT NULL, FOREIGN KEY	Seller user ID
`carbon_footprint`	DECIMAL(10, 3)	DEFAULT 0	CO_2 equivalent in kg
`carbon_explanation`	TEXT	-	Explanation of carbon calculation
`carbon_points`	INT	DEFAULT 0	Eco-points awarded
`carbon_details_material`	VARCHAR(100)	-	Material composition
`carbon_details_manufacturing_carbon`	FLOAT	-	Manufacturing CO_2 (kg)
`carbon_details_shipping_carbon`	FLOAT	-	Shipping CO_2 (kg)
`carbon_details_packaging_carbon`	FLOAT	-	Packaging CO_2 (kg)
`carbon_details_recycle_percentage`	INT	-	Recyclable percentage
`carbon_details_country`	VARCHAR(50)	-	Manufacturing country

Field	Type	Constraints	Description
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Product creation timestamp
updated_at	TIMESTAMP	ON UPDATE	Last update timestamp

3. **orders** Table

Stores customer orders and their status.

```
CREATE TABLE orders (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,
  user_id BIGINT NOT NULL,
  total_price DECIMAL(10, 2),
  total_carbon DECIMAL(10, 3),
  status VARCHAR(20),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Fields:

Field	Type	Constraints	Description
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique order identifier
user_id	BIGINT	NOT NULL, FOREIGN KEY	Customer user ID
total_price	DECIMAL(10, 2)	-	Total order amount
total_carbon	DECIMAL(10, 3)	-	Total carbon footprint (kg CO_2e)
status	VARCHAR(20)	-	PENDING, CONFIRMED, SHIPPED, DELIVERED, CANCELLED
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP	Order creation timestamp

4. **order_items** Table

Stores individual line items within an order.

```
CREATE TABLE order_items (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,
  order_id BIGINT NOT NULL,
  product_id BIGINT NOT NULL,
  quantity INT NOT NULL,
  subtotal DECIMAL(10, 2),
  carbon_total DECIMAL(10, 3),
  FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE,
  FOREIGN KEY (product_id) REFERENCES products(product_id) ON DELETE CASCADE
);
```

Fields:

Field	Type	Constraints	Description
`id`	BIGINT	PRIMARY KEY, AUTO_INCREMENT	Unique order item identifier
`order_id`	BIGINT	NOT NULL, FOREIGN KEY	Associated order ID
`product_id`	BIGINT	NOT NULL, FOREIGN KEY	Product ID
`quantity`	INT	NOT NULL	Quantity ordered
`subtotal`	DECIMAL(10, 2)	-	Line item total (price × quantity)
`carbon_total`	DECIMAL(10, 3)	-	Carbon footprint for this item

Authentication

JWT Token Format

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

```
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqb2huX2RvZSIsImlhdCI6MTczMjQxODAwM
H0.SIGNATURE
```

Role-Based Access Control (RBAC)

Endpoint	Required Role	Description
`/register`, `/login`	None	Public endpoints
`/all-products`, `/api/products`	None	Public endpoints
`/seller/*`	SELLER	Seller-specific operations
`/admin/*`	ADMIN	Administrator operations
`/api/checkout/*`	USER/CUSTOMER	User orders (any authenticated user)

Error Responses

401 Unauthorized

```
{
  "success": false,
  "message": "Invalid credentials"
}
```

403 Forbidden

```
{
  "error": "Access denied. Required role: ADMIN"
}
```

400 Bad Request

```
{
  "error": "Missing or invalid request parameters",
  "details": "Carbon details are required to estimate the carbon footprint."
}
```

404 Not Found

```
{
  "error": "Resource not found",
  "message": "User not found with username: john_doe"
}
```

500 Internal Server Error

```
{
  "error": "Internal server error",
  "message": "An unexpected error occurred"
}
```

Example Complete Workflow

1. Register a New Customer

```
curl -X POST http://localhost:8080/register \
  -H "Content-Type: application/json" \
  -d '{
    "username": "eco_customer",
    "password": "SecurePass@2024",
    "email": "customer@example.com",
    "firstName": "Eco",
    "lastName": "Customer",
    "role": "USER"
  }'
```

2. Login

```
curl -X POST http://localhost:8080/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "eco_customer",
    "password": "SecurePass@2024"
  }'
```

3. Get All Products

```
curl http://localhost:8080/all-products
```

4. Filter Products by Category and Max Carbon

```
curl "http://localhost:8080/api/products?category=Kitchen&maxCarbon=5"
```

5. Place Order (Checkout)

```
curl -X POST http://localhost:8080/api/checkout/add-checkout \
  -H "Authorization: Bearer YOUR_JWT_TOKEN"
```

6. View My Orders

```
curl http://localhost:8080/api/checkout/my-orders \
  -H "Authorization: Bearer YOUR_JWT_TOKEN"
```

Notes

- **Password Hashing:** All passwords are hashed using BCrypt (strength 12) before storage.
- **JWT Expiration:** Token expiration is configured in the backend security config.
- **CORS:** Enabled for frontend-backend communication.
- **Timestamps:** All datetime fields use UTC and are stored as LocalDateTime.
- **Carbon Footprint:** Calculated based on material, manufacturing, shipping, and packaging details.
- **Cascade Deletes:** Deleting a user cascades to products and orders they created.

Last Updated: November 11, 2024