# Shop Organics API Reference Documentation

## Introduction

The Shop Organics app allows users to browse, shop, and manage organic products. This document provides an overview of the available API endpoints used for CRUD (Create, Read, Update, Delete) operations in the system. The APIs are built using RESTful principles and are designed to interact with the app's backend.

## Base URL

All API requests are made to the base URL:

```
https://api.shoporganics.com/v1
```

## Auth

The APIs require an Authorization Token in the header for secure access. You can get the token by logging into the app and using the Login API.

Example:

```
Authorization: Bearer <YOUR_AUTH_TOKEN>
```

## API Endpoints Overview

### 1. Product Management

- Get All Products
- Get Single Product
- Create New Product
- Update Product
- Delete Product

### *Get All Products*

**Endpoint**: /products

**Method**: GET

**Description**: Fetch all available products.

**Example Request:**

**Request:**

```
GET /products
Authorization: Bearer <YOUR_AUTH_TOKEN>
```

**Example Response:**

**Response:**

```
{
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": "Organic Apple",
      "price": 3.5,
      "category": "Fruits",
      "description": "Fresh organic apples."
    },
    {
      "id": 2,
      "name": "Organic Banana",
      "price": 2.0,
      "category": "Fruits",
      "description": "Fresh organic bananas."
    }
  ]
}
```

### *Get Single Product*

**Endpoint**: `/products/{product_id}`

**Method**: `GET`

**Description**: Fetch a specific product by its ID.

**Example Request:**

```
GET /products/1
Authorization: Bearer <YOUR_AUTH_TOKEN>
```

**Example Response:**

```
{
  "status": "success",
  "data": {
    "id": 1,
    "name": "Organic Apple",
    "price": 3.5,
    "category": "Fruits",
    "description": "Fresh organic apples."
  }
}
```

### *Create New Product*

**Endpoint**: `/products`

**Method**: `POST`

**Description**: Create a new product.

**Example Request:**

```
POST /products
Authorization: Bearer <YOUR_AUTH_TOKEN>
Content-Type: application/json
```

**Example Request Body:**

```
{
  "name": "Organic Mango",
  "price": 4.0,
  "category": "Fruits",
  "description": "Fresh organic mangoes."
}
```

**Example Response:**

```
{
  "status": "success",
  "message": "Product created successfully",
  "data": {
    "id": 3,
    "name": "Organic Mango",
    "price": 4.0,
    "category": "Fruits",
    "description": "Fresh organic mangoes."
  }
}
```

### *Update Product*

**Endpoint**: `/products/{product_id}`

**Method**: `PUT`

**Description**: Update the details of an existing product.

**Example Request:**

```
PUT /products/1
Authorization: Bearer <YOUR_AUTH_TOKEN>
Content-Type: application/json
```

**Example Request Body:**

```
{
  "name": "Organic Apple",
  "price": 3.8,
  "category": "Fruits",
  "description": "Fresh organic apples with improved flavor."
}
```

**Example Response:**

```
{
  "status": "success",
  "message": "Product updated successfully",
  "data": {
    "id": 1,
    "name": "Organic Apple",
    "price": 3.8,
    "category": "Fruits",
    "description": "Fresh organic apples with improved flavor."
  }
}
```

### *Delete Product*

**Endpoint**: `/products/{product_id}`

**Method**: `DELETE`

**Description**: Delete a product by its ID.

**Example Request:**

```
DELETE /products/1
Authorization: Bearer <YOUR_AUTH_TOKEN>
```

**Example Response:**

```
{
  "status": "success",
  "message": "Product deleted successfully"
}
```

## 2. Product Reviews

- Add Product Review
- Get Product Review

### *Add Product Review*

**Endpoint**: /addreview

**Method**: POST

**Description**:  Add a review for a product.

**Example Request:**

Request:

```
POST /addreview
Authorization: Bearer <YOUR_AUTH_TOKEN>
Content-Type: application/json
```

**Example Request Body:**

Body:

```
{
  "product_id": 1,
  "rating": 5,
  "comment": "Great quality!"
}
```

**Example Response:**

Response:

```
{
  "status": "success",
  "message": "Review added successfully"
}
```

### *Get Product Review*

**Endpoint**: /productreview

**Method**: GET

**Description**: Fetch reviews for a specific product.

**Example Request:**

Request:

```
GET /productreview?product_id=1
Authorization: Bearer <YOUR_AUTH_TOKEN>
```

**Example Response:**

Response:

```
{
  "status": "success",
  "reviews": [
    {
      "rating": 5,
      "comment": "Great quality!"
    }
  ]
}
```

## 3. Cart Management

- Add to Cart

### *Add to Cart*

**Endpoint**: /addtocart

**Method**: POST

**Description**: Add a product to the shopping cart.

**Example Request:**

Request:

```
POST /addtocart
Authorization: Bearer <YOUR_AUTH_TOKEN>
Content-Type: application/json
```

**Example Request Body:**

Body:

```
{
  "product_id": 1,
  "quantity": 2
}
```

**Example Response:**

Response:

```
{
  "status": "success",
  "message": "Product added to cart"
}
```

## 4. User Authentication

- Login
- Logout

### *Login*

**Endpoint**: /login

**Method**: POST

**Description**: Authenticate a user and return an access token.

**Example Request:**

Request:

```
POST /login
Content-Type: application/json
```

**Example Request Body:**

Body:

```
{
  "email": "user@example.com",
  "password": "securepassword"
}
```

**Example Response:**

Response:

```
{
  "status": "success",
  "token": "your_auth_token"
}
```

### *Logout*

**Endpoint**: /logout

**Method**: POST

**Description**: Logout the user and invalidate the token.

**Example Requst:**

**Request:**

```
POST /logout
Authorization: Bearer <YOUR_AUTH_TOKEN>
```

**Example Response:**

**Response:**

```
{
  "status": "success",
  "message": "User logged out successfully"
}
```

## 5. Order Management

- My Orders

### *My Orders*

**Endpoint**: /myorders

**Method**: GET

**Description**: Fetch a list of orders placed by the user.

**Example Request:**

**Request:**

```
GET /myorders
Authorization: Bearer <YOUR_AUTH_TOKEN>
```

**Example Response:**

Response:


```
{
  "status": "success",
  "orders": [
    {
      "order_id": 123,
      "items": [
        {
          "product_name": "Organic Apple",
          "quantity": 2,
          "price": 7.0
        }
      ],
      "total": 7.0
    }
  ]
}
```


## 6. Store Management

. Store

### *Store Details*

**Endpoint**: /store

**Method**: GET

**Description**: Fetch store details.

**Example Request:**

Request:

```
GET /store
Authorization: Bearer <YOUR_AUTH_TOKEN>
```

**Example Response:**

Response:

```
{
  "status": "success",
  "store": {
    "name": "Shop Organics",
    "location": "123 Green St, Springfield"
  }
}
```

## Error Handling

When interacting with the API, you may encounter various errors. Below are common HTTP status codes and their meanings:

- **400 Bad Request**: The request was malformed or missing required parameters
- **401 Unauthorized**: Authentication token is missing or invalid
- **403 Forbidden**: You do not have permission to access this resource
- **404 Not Found**: The requested resource was not found
- **500 Internal Server Error**: An unexpected error occurred on the server

**Example Error Response:**

```
{
  "status": "error",
  "message": "Product not found",
  "error": "404 Not Found"
}
```

## Testing APIs with Swagger

Swagger provides an easy-to-use interface for testing your APIs. You can use it to interact with the API endpoints directly, sending requests and viewing responses.

- **Access Swagger UI**: Visit Swagger UI
- **Authenticate**: Before sending any requests, make sure to authenticate by using your Bearer token.
- **Select Endpoint**: Choose an endpoint from the list.
- **Send Request**: Input the necessary parameters, then click on the 'Try it out' button.
- **View Response**: Swagger will display the response with status, data, and any error messages.

## Common Troubleshooting

- **Invalid Authorization TokenError:** 401 Unauthorized
  - **Solution**: Ensure you are passing a valid bearer token in the header.
- **Bad Request FormatError:** 400 Bad Request
  - **Solution**: Ensure your JSON request body is correctly formatted.
- **Resource Not FoundError**: 404 Not Found
  - **Solution**: Ensure the resource exists and the ID is correct.
- **Server ErrorError**: 500 Internal Server Error
  - **Solution**: Contact the backend team with the error message for assistance.

## Conclusion

This document follows RESTful best practices with proper HTTP method usage for interacting with Shop Organics API. Authenticate before making requests, handle errors gracefully, and use Swagger for testing and exploration. For complete implementation details, refer to the Shop Organics Implementation.