**Dripyard API documentation**

**1. Authentication (/auth)**

Handles user signup and login. These endpoints are public and do not require an authentication token.

* **POST /auth/sent/login-signup-otp**
  + **Description:** Initiates the login or signup process by generating and emailing a One-Time Password (OTP) to the user.
  + **Method:** sentLoginOtp in AuthController
  + **Request Body:**

JSON

{

"email": "customer@example.com"

}

* + **Success Response (200 OK):**

JSON

{

"message": "OTP sent",

"status": true

}

* **POST /auth/signup**
  + **Description:** Creates a new customer account (or logs in an existing user) by verifying the OTP sent to their email.
  + **Method:** createUserHandler in AuthController
  + **Request Body:**

JSON

{

"fullName": "Test Customer",

"email": "customer@example.com",

"otp": "123456"

}

* + **Success Response (201 Created):** Returns a JWT for session management.

JSON

{

"jwt": "your-jwt-token",

"status": true,

"message": "Register Success",

"role": "ROLE\_CUSTOMER"

}

* **POST /auth/signin**
  + **Description:** Authenticates an existing user (primarily for the admin's password-based login) and returns a JWT.
  + **Method:** signin in AuthController
  + **Request Body:**

JSON

{

"email": "admin@example.com",

"password": "yourpassword"

}

* + **Success Response (200 OK):** Returns a JWT with the appropriate role (ROLE\_ADMIN or ROLE\_CUSTOMER).

**2. Admin Functionalities (/api/admin)**

Endpoints for managing the store. **Authentication:** Requires a ROLE\_ADMIN JWT in the Authorization header.

* **POST /api/admin/products**
  + **Description:** Creates a new product.
  + **Method:** createProduct in AdminController.
* **GET /api/admin/products**
  + **Description:** Retrieves a list of all products in the store.
  + **Method:** getAllProducts in AdminController.
* **DELETE /api/admin/products/{productId}**
  + **Description:** Deletes a product by its ID.
  + **Method:** deleteProduct in AdminController.
* **GET /api/admin/orders**
  + **Description:** Retrieves a list of all customer orders.
  + **Method:** getAllOrders in AdminController.
* **PATCH /api/admin/orders/{orderId}/status**
  + **Description:** Updates the status of an order (e.g., PLACED, SHIPPED, DELIVERED).
  + **Method:** updateOrderStatus in AdminController.
  + **Query Parameter:** status (e.g., "SHIPPED").
* **POST /api/admin/coupons/create**
  + **Description:** Creates a new discount coupon.
  + **Method:** createCoupon in AdminCouponController.
* **DELETE /api/admin/coupons/delete/{id}**
  + **Description:** Deletes a coupon by its ID.
  + **Method:** deleteCoupon in AdminCouponController.
* **GET /api/admin/coupons/all**
  + **Description:** Retrieves a list of all coupons.
  + **Method:** getAllCoupons in AdminCouponController.

**3. Public Product & Store Endpoints**

Endpoints for browsing products and general store information.

* **GET /**
  + **Description:** A simple health-check endpoint to confirm the backend is running.
  + **Method:** home in HomeController.
  + **Authentication:** Public.
* **GET /api/products**
  + **Description:** Retrieves a paginated and filtered list of products.
  + **Method:** findProductByCategoryHandler in ProductController.
  + **Authentication:** Public.
  + **Query Parameters:** category, color, size, minPrice, maxPrice, minDiscount, sort, stock, pageNumber, pageSize.
* **GET /api/products/{productId}**
  + **Description:** Retrieves the details of a single product.
  + **Method:** findProductById in ProductController.
  + **Authentication:** Public.
* **GET /api/products/{productId}/reviews**
  + **Description:** Gets all reviews for a specific product.
  + **Method:** getReviewsByProductId in ReviewController.
  + **Authentication:** Public.
* **GET /api/images/{key}**
  + **Description:** Downloads and displays a product image from the Cloudflare R2 bucket.
  + **Method:** getImage in ImageController.
  + **Authentication:** Public.

**4. Authenticated User Actions**

Endpoints for logged-in customers and admins. **Authentication:** Requires a valid JWT in the Authorization header.

* **GET /api/users/profile**
  + **Description:** Retrieves the profile of the currently authenticated user.
  + **Method:** getUserProfileHandler in UserController.
* **Cart Management (/api/cart)**
  + GET /api/cart: Retrieves the current user's shopping cart.
  + PUT /api/cart/add: Adds an item to the user's cart.
  + DELETE /api/cart/item/{cartItemId}: Deletes an item from the cart.
  + PUT /api/cart/item/{cartItemId}: Updates the quantity of an item in the cart.
* **Order Management (/api/orders)**
  + POST /api/orders/: Creates a new order from the items in the user's cart.
  + GET /api/orders/user: Retrieves the order history for the current user.
  + GET /api/orders/{id}: Retrieves a specific order by its ID.
  + PUT /api/orders/{orderId}/cancel: Cancels an order placed by the user.
* **Wishlist Management (/api/wishlist)**
  + GET /api/wishlist: Retrieves the current user's wishlist.
  + POST /api/wishlist/add-product/{productId}: Adds or removes a product from the user's wishlist.
* **Reviews (/api/reviews)**
  + POST /api/products/{productId}/reviews: Creates a new review for a product.
  + PATCH /api/reviews/{reviewId}: Updates a review written by the user.
  + DELETE /api/reviews/{reviewId}: Deletes a review written by the user.
* **Image Upload (/api/images)**
  + POST /api/images/upload: Uploads a product image to Cloudflare R2. This is intended for admin use but is available to any authenticated user.

**5. Payment Handling (/api/payments)**

* **GET /api/payments/{orderId}/success**
  + **Description:** The callback URL for Razorpay. After a successful payment, Razorpay redirects the user here. The backend confirms the payment and updates the order status.
  + **Method:** paymentSuccessRedirect in PaymentController.
  + **Authentication:** Public.
  + **Query Parameter:** razorpay\_payment\_id.