FINAL PROJECT REPORT

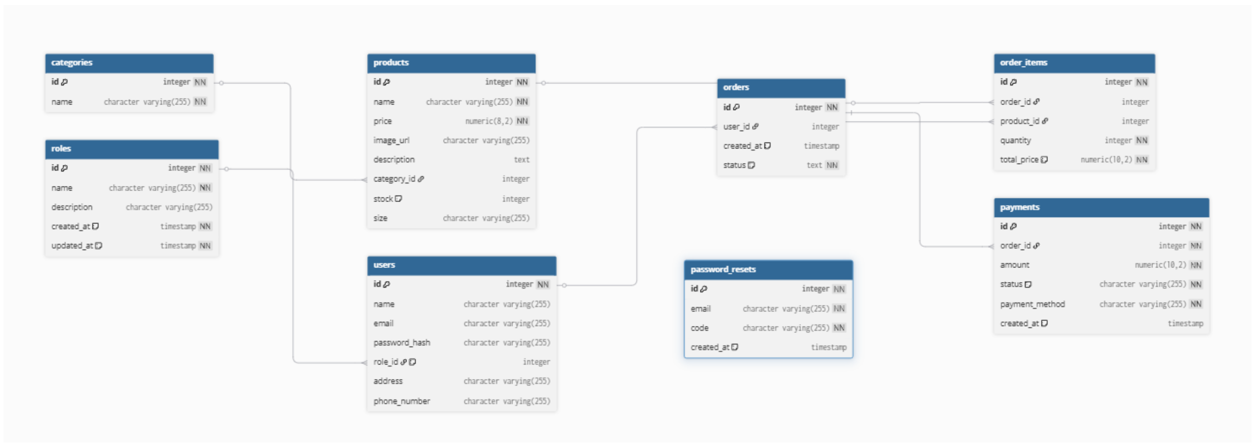
SEMESTER 3, ACADEMIC YEAR: 2024-2025

CT313H: WEB TECHNOLOGIES AND SERVICES

- **Project/Application name: Clothify Store e-Commerce**
- **GitHub links (for both frontend and backend): https://github.com/24-25Sem3-Courses/ct313hm01-project-duongthevi.git**
- **Student ID 1: B2206023**
- **Student Name 1: Dương Thế Vĩ**
- **Student ID 2: B2207579**
- **Student Name 2: Ngô Gia Vinh**
- **Class/Group Number (e.g, CT313HM01): CT313HM01/ Group 15**
- **Link Video Demo: https://youtu.be/3prT8wVfKB4**

I. Introduction

- An e-commerce web application designed for selling apparel products, allowing users to browse, add items to a cart, place orders, and manage their profiles with a responsive and user-friendly interface.
- A list of tables and their structures in the database:



categories
id integer NN
name character varying(255) NN
roles
id integer NN
name character varying(255) NN
description character varying(255)
created_at timestamp NN
updated_at timestamp NN
products
id integer NN
name character varying(255) NN
price numeric(8,2) NN
image_url character varying(255)
description text
category_id integer
stock integer
size character varying(255)
users
id integer NN
name character varying(255)
email character varying(255)
password_hash character varying(255)
role_id integer
address character varying(255)
phone_number character varying(255)
orders
id integer NN
user_id integer
created_at timestamp
status text NN
password_resets
id integer NN
email character varying(255) NN
code character varying(255) NN
created_at timestamp
order_items
id integer NN
order_id integer
product_id integer
quantity integer NN
total_price numeric(10,2) NN
payments
id integer NN
order_id integer NN
amount numeric(10,2) NN
status character varying(255) NN
payment_method character varying(255) NN
created_at timestamp

1. user

Column	Type	Constraint
id	integer	Primary Key, NN
name	character varying(255)	Not Null
email	character varying(255)	Not Null
password_hash	character varying(255)	Not Null
role	character varying(255)	Not Null

2. categories

Column	Type	Constraint
id	integer	Primary Key, NN
name	character varying(255)	Not Null

3. products

Column	Type	Constraint
id	integer	Primary Key, NN
name	character varying(255)	Not Null
price	numeric(8,2)	Not Null
image_url	character varying(255)	(optional)
description	text	(optional)
category_id	integer	Foreign Key → categories(id)
stock	integer	(optional)

4. orders

Column	Type	Constraint
id	integer	Primary Key, NN
user_id	integer	Foreign Key → users(id)
status	character varying(255)	Not Null
created_at	timestamp	Not Null

5. order_items

Column	Type	Constraint
id	integer	Primary Key, NN
order_id	integer	Foreign Key → orders(id)
product_id	integer	Foreign Key → products(id)
quantity	integer	Not Null

6. payments

Column	**Type**	**Constraint**
Id	Interger	Not Null
order_id	integer	Not Null (NN)
amount	numeric(10,2)	Not Null (NN)
status	character varying(255)	Not Null (NN)
payment_method	character varying(255)	Not Null (NN)
created_at	timestamp	Not Null (NN)

7. roles

Column	**Type**	**Constraint**
Id	Interger	Not Null
name	character varying(255)	Not Null (NN)
description	character varying(255)	Optional
created_at	timestamp	Optional
updated_at	timestamp	Optional

8. password_resets

Column	**Type**	**Constraint**
id	integer	Primary Key, Not Null
email	character varying(255)	Not Null
code	character varying(255)	Not Null
created_at	timestamp	Default: CURRENT_TIMESTAMP

- A task assignment sheet for each member if working in groups.

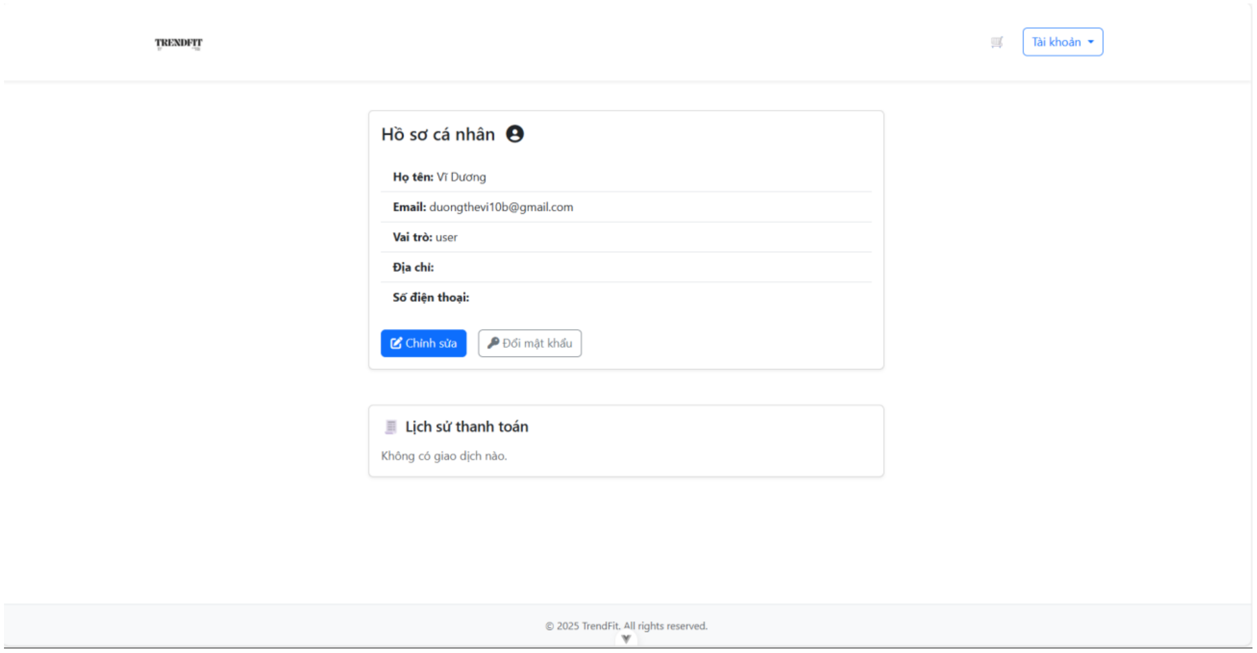
Task	Member	Tech Used
Design database schema & migrations	Dương Thế Vĩ	PostgreSQL, Knex.js, zod
Implement user authentication & roles	Ngô Gia Vinh	Express.js, JWT
Develop RESTful API for products/orders/payment	Ngô Gia Vinh, Dương Thế Vĩ	Express.js, Knex.js, S3
Build SPA user interface	Dương Thế Vĩ, Ngô Gia Vinh	Vue.js, Vue Router, Pinia
Integrate frontend with backend API	Ngô Gia Vinh, Dương Thế Vĩ	Fetch API
Write API documentation	Ngô Gia Vinh	Swagger
Test backend & frontend functionality	Dương Thế Vĩ	Postman and Browser
Deploy SPA and API	Ngô Gia Vinh	Nginx and PM2

Write report	Ngô Gia Vinh, Dương Thế Vĩ	

II. Details of implemented features

1. Feature / Application page 1: User Profile Page

- **Description:** This page allows users to view and update their personal information, such as name, email, and password. It also displays the user's payment history and related order details. The page helps users manage their account and track their purchases.



Hồ sơ cá nhân

Họ tên: Vĩ Dương
Email: duongthevi10b@gmail.com
Vai trò: user
Địa chỉ:
Số điện thoại:

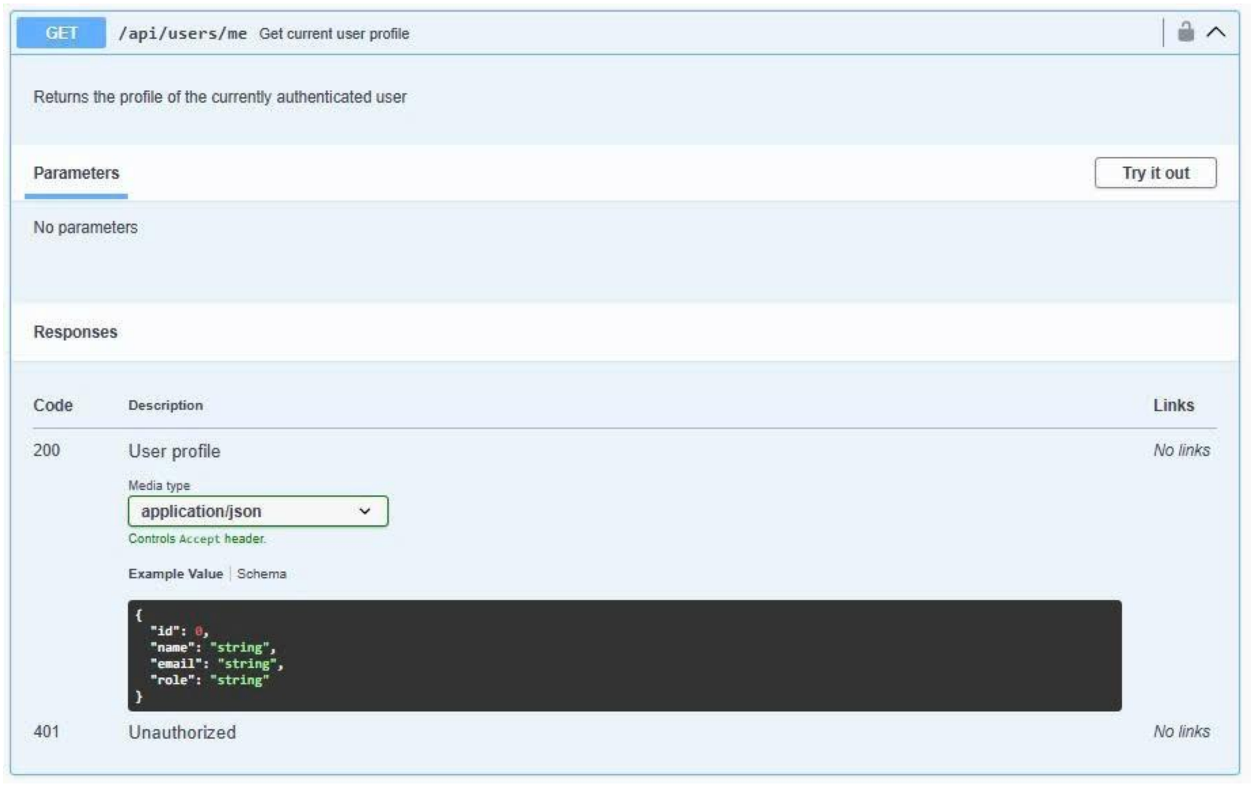
Lịch sử thanh toán

Không có giao dịch nào.

© 2025 TrendFit. All rights reserved.

- **Implementation details:**
  + Libraries used:
    ▪ Frontend:
      • Pinia (state management)
    ▪ Backend:
      • Jsend
      • ApiError
  + Server-side APIs used:

Endpoint	Method	Purpose
/api/users/me	GET	Get current logged-in user's profile
/api/payments/user/:userId	GET	Get paginated payment records for a user
/api/order/:orderId/items	GET	Get all items in an order



GET `/api/users/me` Get current user profile

Returns the profile of the currently authenticated user

Parameters

Try it out

No parameters

Responses

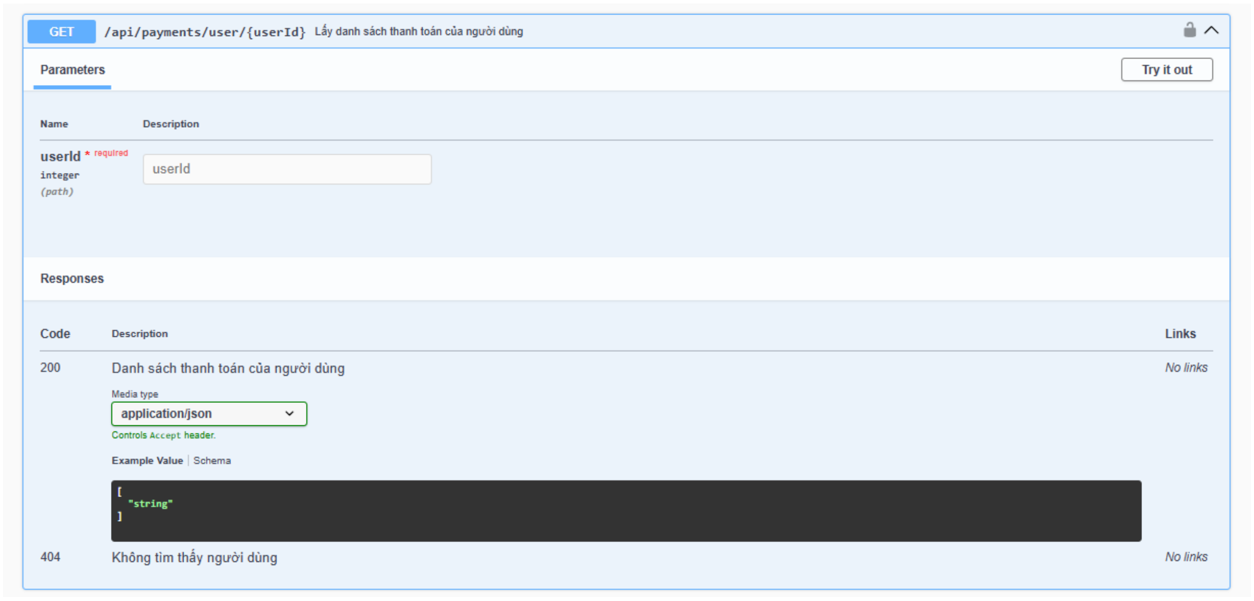
Code	Description	Links
200	User profile	*No links*
401	Unauthorized	*No links*

Media type: application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "email": "string",
  "role": "string"
}
```



GET `/api/payments/user/{userId}` Lấy danh sách thanh toán của người dùng

Parameters

Try it out

Name	Description
userId * required `integer` *(path)*	userId

Responses

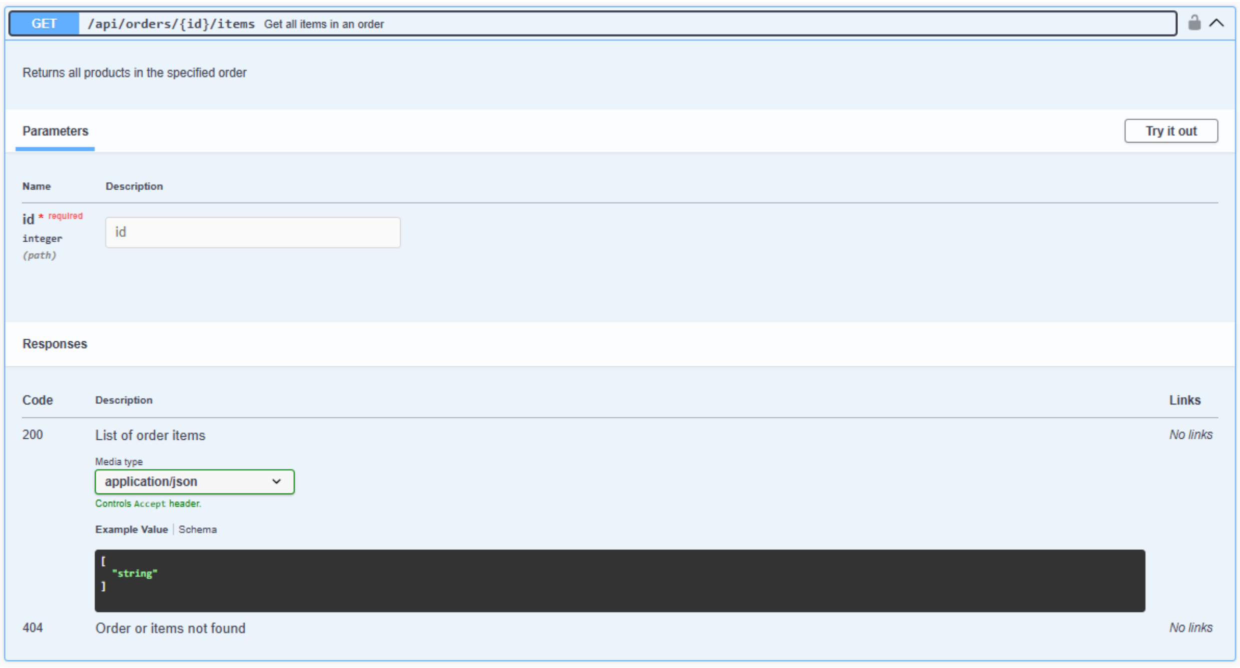
Code	Description	Links
200	Danh sách thanh toán của người dùng	*No links*
404	Không tìm thấy người dùng	*No links*

Media type: application/json

Controls `Accept` header.

Example Value | Schema

```
[
  "string"
]
```


GET /api/orders/{id}/items Get all items in an order

Returns all products in the specified order

Parameters

Name	Description
id * required integer (path)	id

Responses

Code	Description	Links
200	List of order items Media type: application/json Controls Accept header. Example Value | Schema	No links
404	Order or items not found	No links

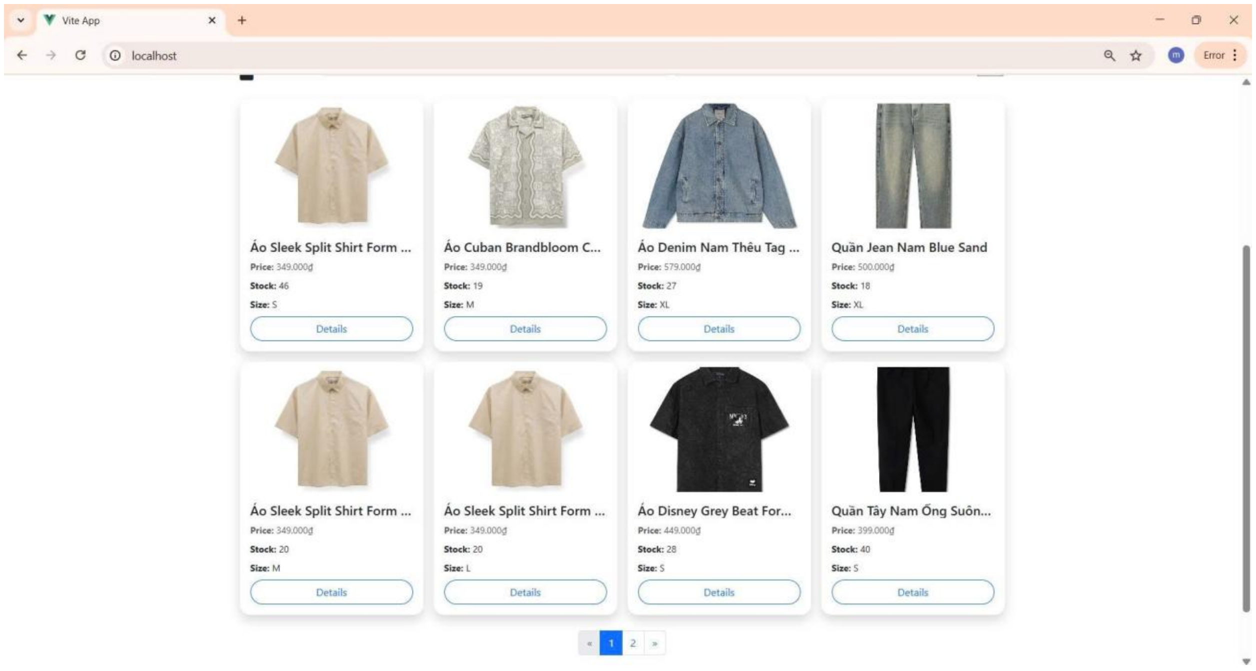
```
[
  "string"
]
```

+ Data read/stored:
  - Tables:
    - users (read/update user info)
    - payments (read payment history)
    - order_items (read order details)
    - products
+ Client-side states needed:

State Variable	Description
user	Computed object of the authenticated user's data
showResetModal	Boolean for showing the password reset modal
payments	List of payment records for the user
page	Current pagination page number
limit	Items per page (default: 5)
totalPages	Total available pages for payment history
loadingPayments	Boolean to show loading state while fetching payments

2. Feature / Application page 2: Product Listing Page

- **Description:** Displays a paginated list of products for users to browse. Users can filter by category and perform keyword-based search on product name or description.



Áo Sleek Split Shirt Form ...
Price: 349.000₫
Stock: 46
Size: S

Áo Cuban Brandbloom C...
Price: 349.000₫
Stock: 19
Size: M

Áo Denim Nam Thêu Tag ...
Price: 579.000₫
Stock: 27
Size: XL

Quần Jean Nam Blue Sand
Price: 500.000₫
Stock: 18
Size: XL

Áo Sleek Split Shirt Form ...
Price: 349.000₫
Stock: 20
Size: M

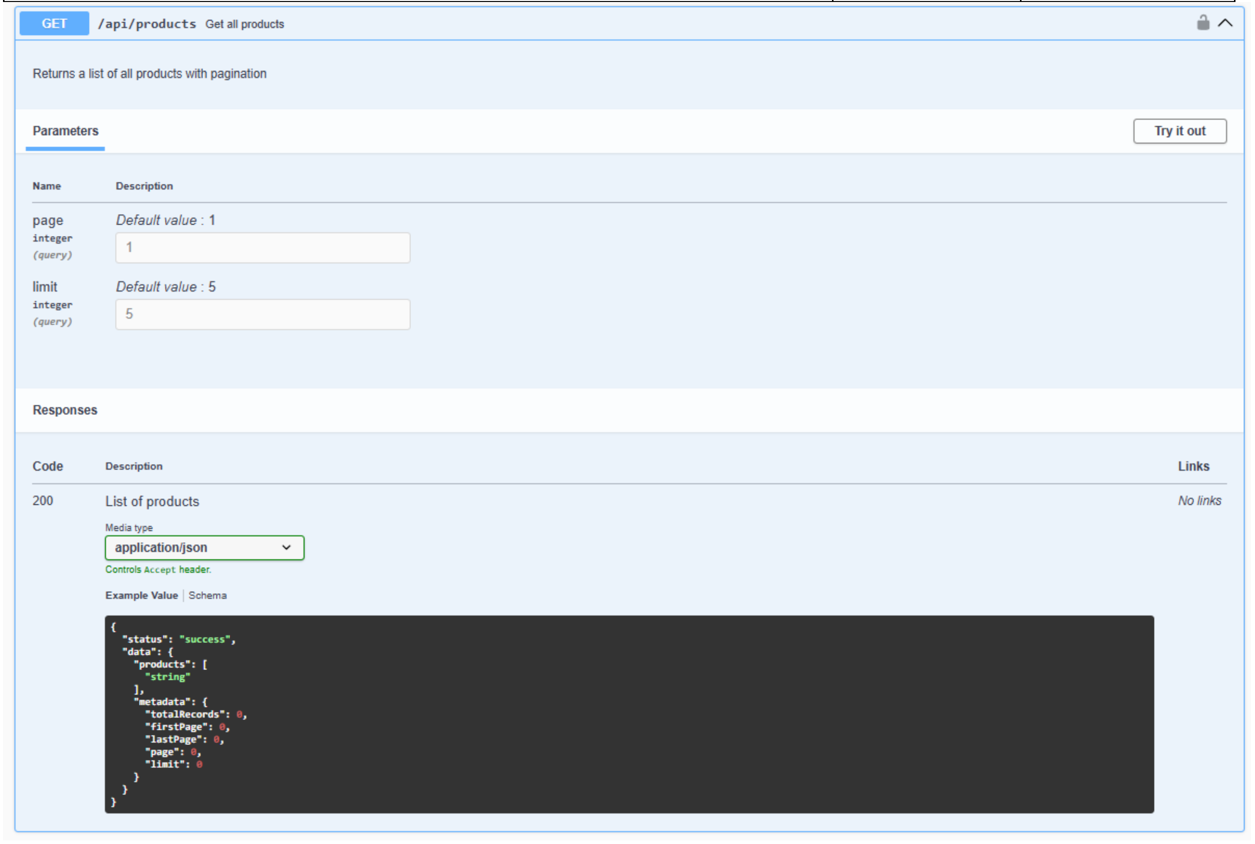
Áo Sleek Split Shirt Form ...
Price: 349.000₫
Stock: 20
Size: L

Áo Disney Grey Beat For...
Price: 449.000₫
Stock: 28
Size: S

Quần Tây Nam Ống Suôn...
Price: 399.000₫
Stock: 40
Size: S

- **Implementation details:**
  + Libraries used:
    ▪ Frontend:
      • Pinia (state management)
    ▪ Backend:
      • Jsend
      • ApiError
  + Server-side APIs used:

API Endpoint	Method	Purpose
/api/products?page=1&limit=10	GET	Get paginated list of all products
/api/products/search?search=keyword&page=1&limit=10	GET	Search products by keyword
/api/products/category/:categoryId?page=1&limit=10	GET	Get products filtered by category
/api/categories	GET	Get all categories for the dropdown filter



`/api/products` Get all products

Returns a list of all products with pagination

Parameters

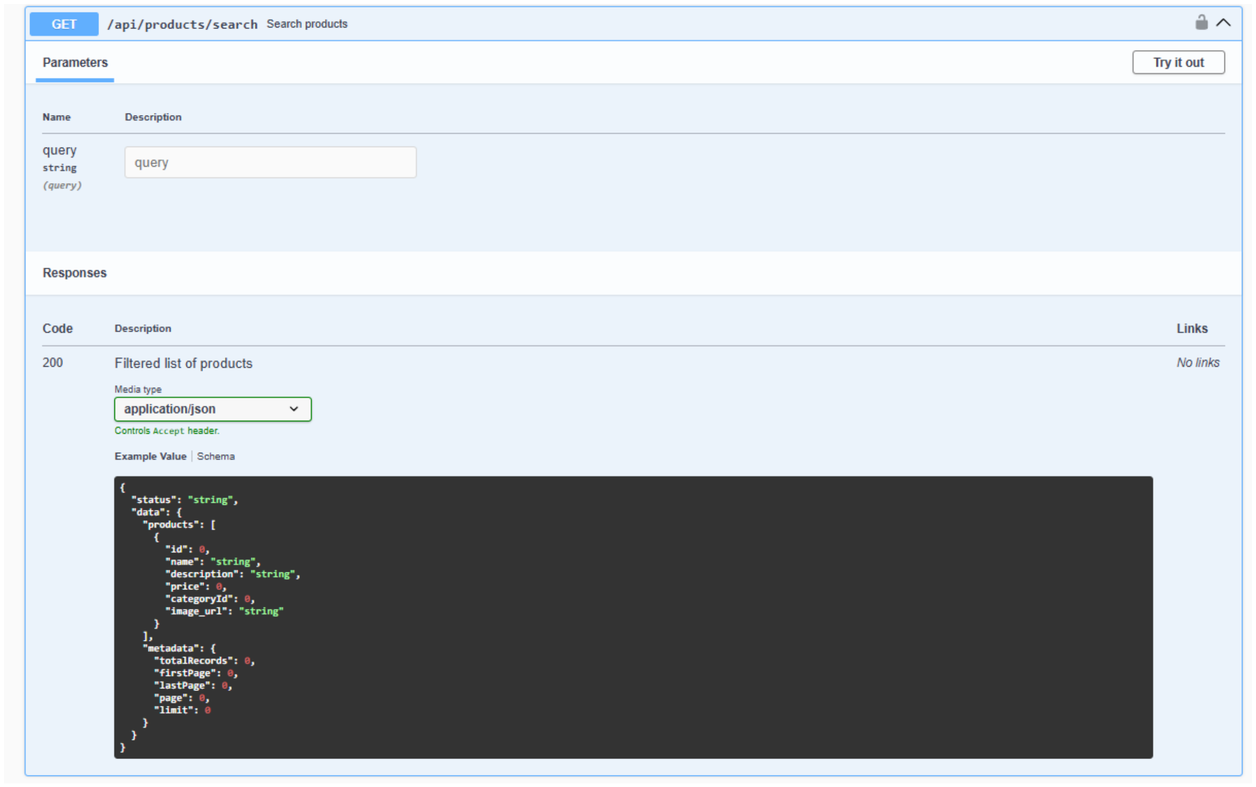
Name	Description
page integer (query)	Default value : 1 1
limit integer (query)	Default value : 5 5

Responses

Code	Description	Links
200	List of products Media type: application/json Controls `Accept` header.	No links

Example Value | Schema

```
{
  "status": "success",
  "data": {
    "products": [
      "string"
    ],
    "metadata": {
      "totalRecords": 0,
      "firstPage": 0,
      "lastPage": 0,
      "page": 0,
      "limit": 0
    }
  }
}
```


GET /api/products/search Search products

Parameters

Try it out

Name	Description
query string *(query)*	query

Responses

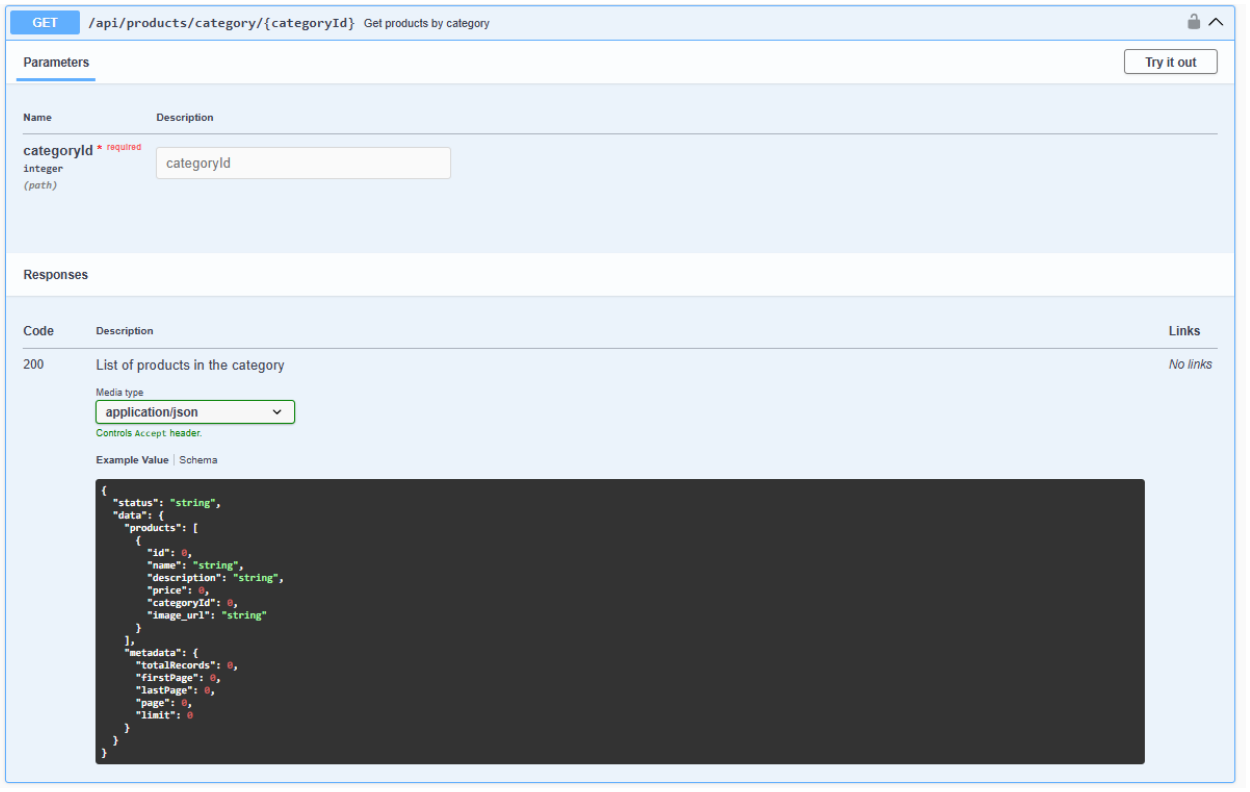
Code	Description	Links
200	Filtered list of products	*No links*

Media type: application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "status": "string",
  "data": {
    "products": [
      {
        "id": 0,
        "name": "string",
        "description": "string",
        "price": 0,
        "categoryId": 0,
        "image_url": "string"
      }
    ],
    "metadata": {
      "totalRecords": 0,
      "firstPage": 0,
      "lastPage": 0,
      "page": 0,
      "limit": 0
    }
  }
}
```


GET /api/products/category/{categoryId} Get products by category

Parameters

Try it out

Name	Description
categoryId * required integer *(path)*	categoryId

Responses

Code	Description	Links
200	List of products in the category	*No links*

Media type: application/json

Controls `Accept` header.

Example Value | Schema

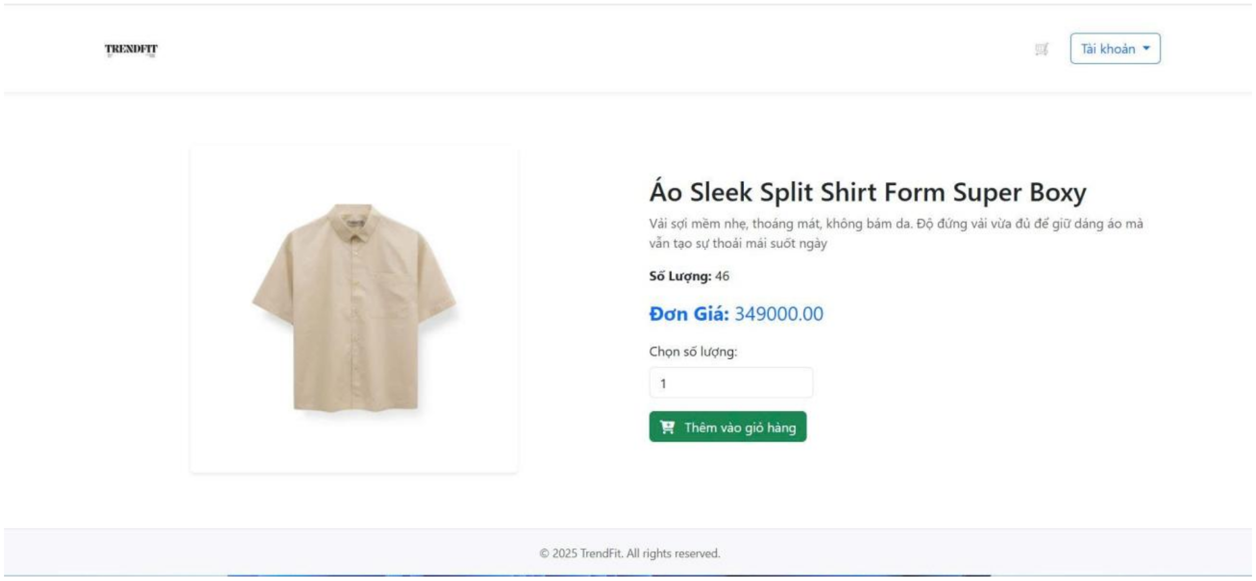
```
{
  "status": "string",
  "data": {
    "products": [
      {
        "id": 0,
        "name": "string",
        "description": "string",
        "price": 0,
        "categoryId": 0,
        "image_url": "string"
      }
    ],
    "metadata": {
      "totalRecords": 0,
      "firstPage": 0,
      "lastPage": 0,
      "page": 0,
      "limit": 0
    }
  }
}
```

+ Data read/stored:
    - Tables:
        - Products
        - categories
+ Client-side states needed:

Variable	Description
products	Current list of products displayed (default or category-based)
searchQuery	Keyword used for searching
searchResult	Result returned from keyword search
hasSearchResults	Boolean to toggle between normal results vs. search result
categories	List of all available product categories
selectedCategory	Category currently selected
currentPage	Current page index (pagination)
totalPages	Total pages for products list
searchCurrentPage	Page index for search results
searchTotalPages	Total pages for search results
isLoading	Loading spinner flag during API fetch
isError / error.message	Error state if API call fails

3. **Feature / Application page 3: Product Detail Page**

- **Description:** Shows detailed information about a selected product, including image, description, price, stock, and allows adding to cart.



Áo Sleek Split Shirt Form Super Boxy

Vải sợi mềm nhẹ, thoáng mát, không bám da. Độ đứng vải vừa đủ để giữ dáng áo mà vẫn tạo sự thoải mái suốt ngày

Số Lượng: 46

Đơn Giá: 349000.00

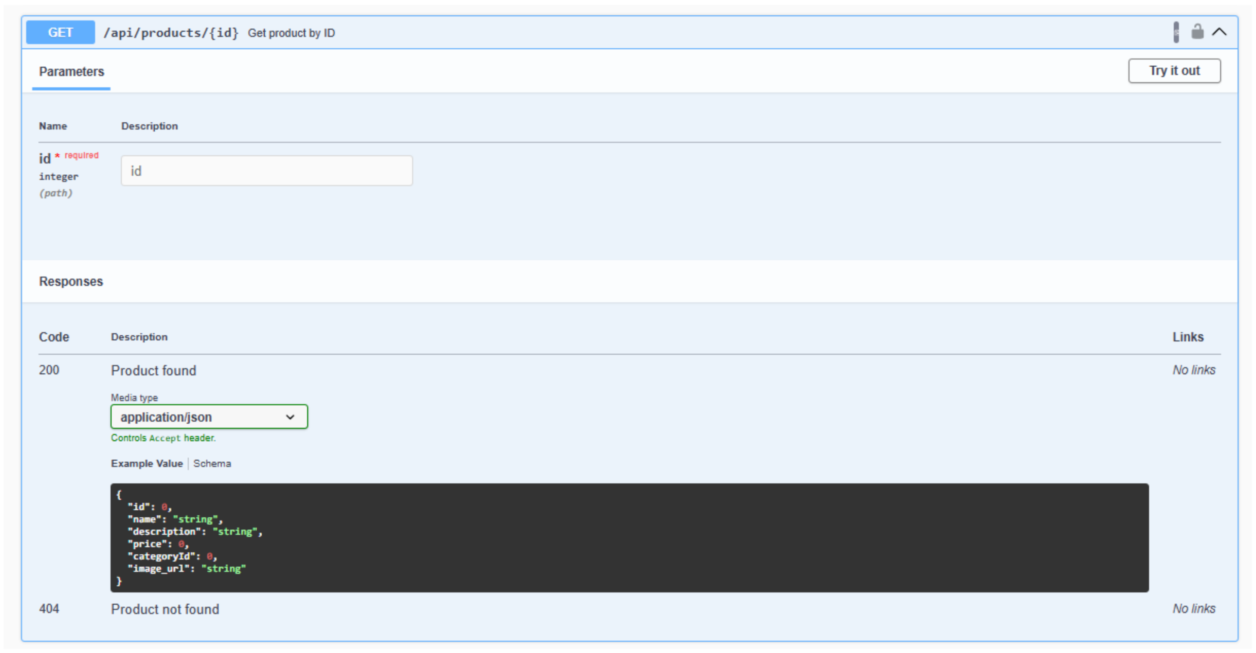
Chọn số lượng:

1

© 2025 TrendFit. All rights reserved.

- **Implementation details:**

+ Libraries used:
    - Frontend:
        - Pinia (state management)
        - Vite (build tool)
        - Prettier (code formatting)
    - Backend:
        - Zod (data validation)
        - AWS SDK (file upload to S3)
        - JSend
        - ApiError
+ Server-side APIs used:
    - GET /api/products/:id
        - Purpose: Get product details by product id
    - POST /api/orders
        - Purpose: Create order

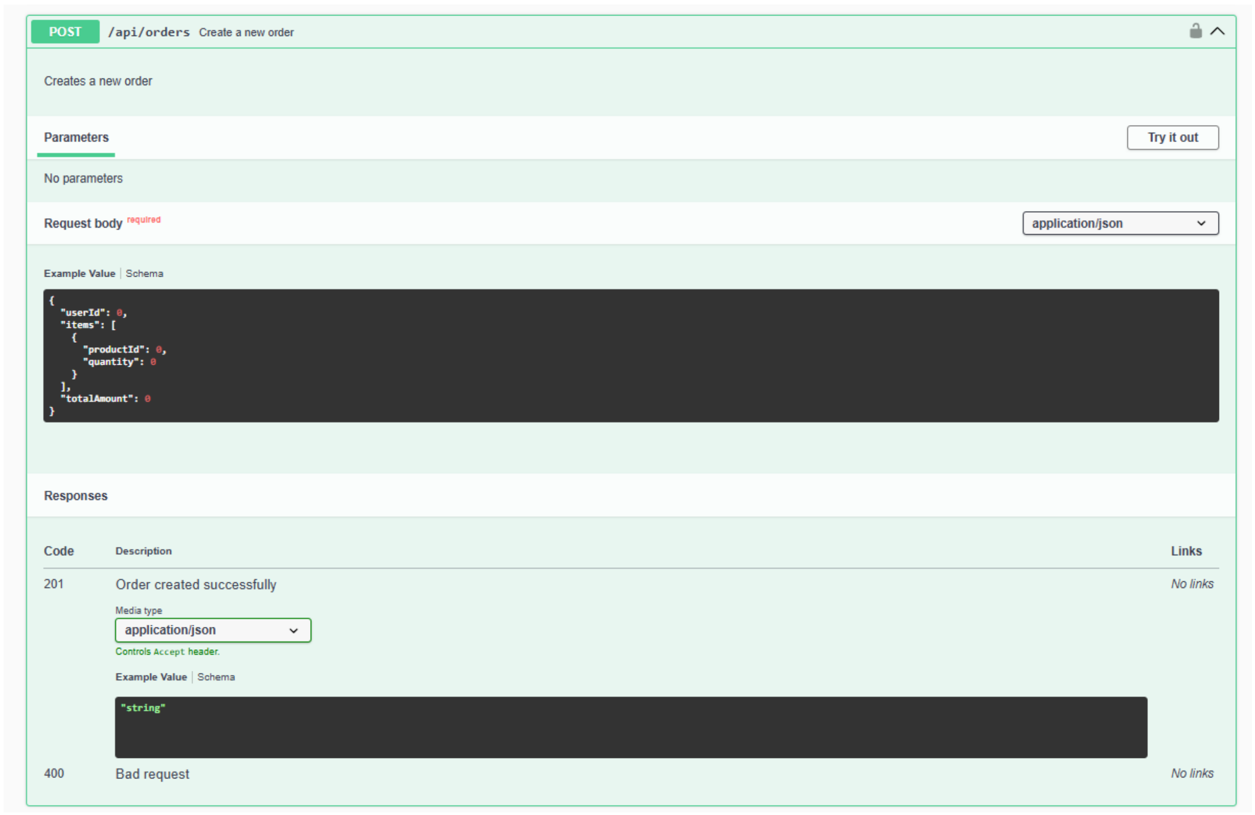


`/api/products/{id}` Get product by ID

Name	Description
id * required integer *(path)*	id

Code	Description	Links
200	Product found Media type: application/json Controls `Accept` header.	*No links*
404	Product not found	*No links*

```
{
  "id": 0,
  "name": "string",
  "description": "string",
  "price": 0,
  "categoryId": 0,
  "image_url": "string"
}
```



/api/orders Create a new order

Creates a new order

No parameters

Request body required

application/json

```
{
  "userId": 0,
  "items": [
    {
      "productId": 0,
      "quantity": 0
    }
  ],
  "totalAmount": 0
}
```

Code	Description	Links
201	Order created successfully Media type: application/json Controls Accept header. Example Value | Schema "string"	No links
400	Bad request	No links

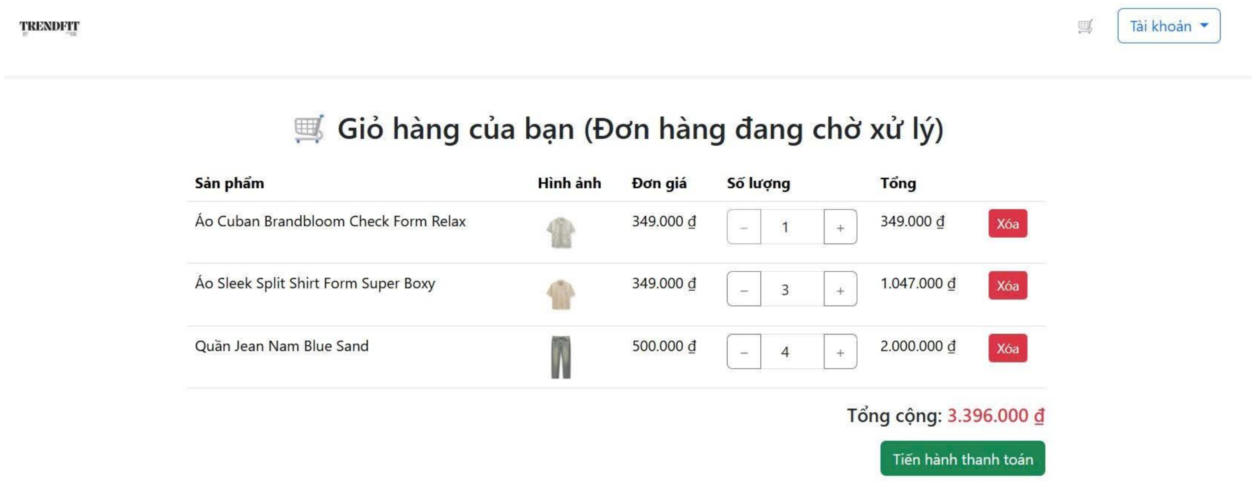
+ Data read/stored:
    - Tables:
        - Products
        - Orders
        - Order_items
+ Client-side states needed:

State Variable	Description
quantity	Quantity of the product the user wants to add
product	The selected product (via API or store fallback)
userRole	The role of the logged-in user
isLoading, isError, error	States from useQuery when fetching product

4. Feature / Application page 4: Cart Page

- **Description**: Displays a list of items in the user's cart (pending order). Users can:
    + View details of each product in the cart.
    + Increase or decrease the quantity of each item.

+ Remove items from the cart.
+ View total cart value.
+ Proceed to payment via a modal popup.



🛒 Giỏ hàng của bạn (Đơn hàng đang chờ xử lý)

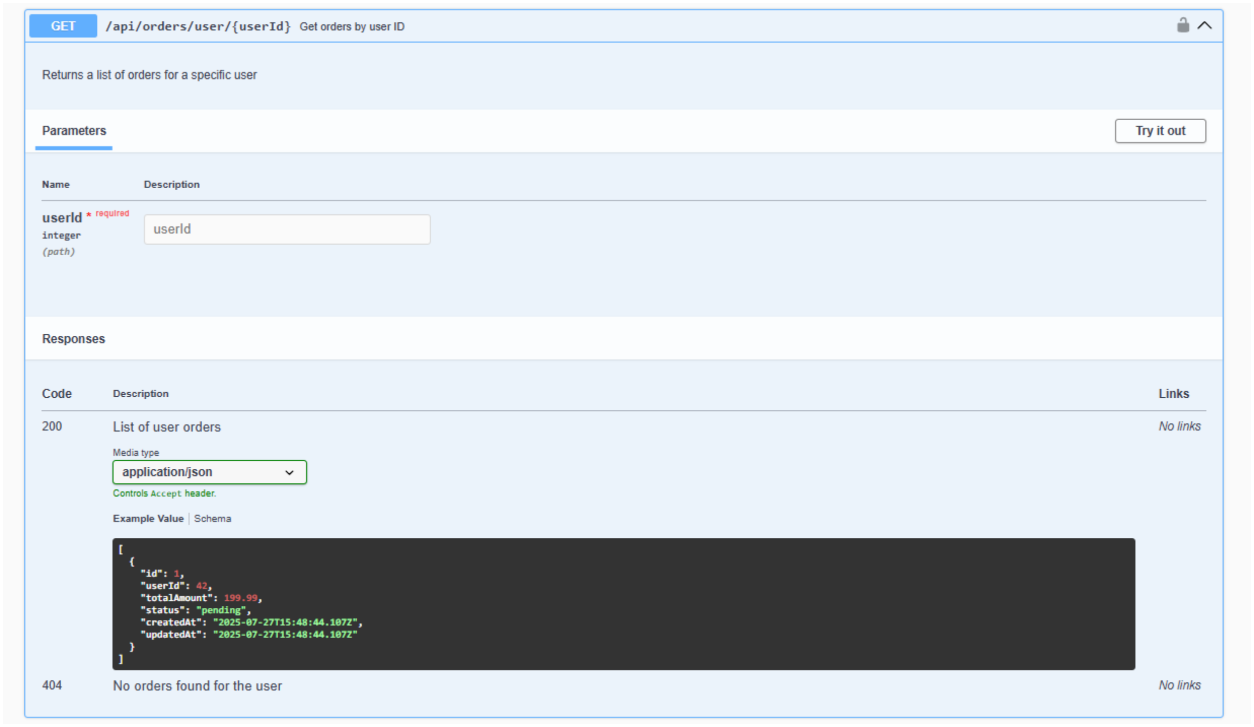
Sản phẩm	Hình ảnh	Đơn giá	Số lượng	Tổng	
Áo Cuban Brandbloom Check Form Relax		349.000 ₫	1	349.000 ₫	Xóa
Áo Sleek Split Shirt Form Super Boxy		349.000 ₫	3	1.047.000 ₫	Xóa
Quần Jean Nam Blue Sand		500.000 ₫	4	2.000.000 ₫	Xóa

Tổng cộng: 3.396.000 ₫

Tiến hành thanh toán

- **Implementation details:**
  + Libraries used:
    - Frontend:
      - Pinia (state management)
    - Backend:
      - JSend
      - APIError
      - Zod
  + Server-side APIs used:

Endpoint	Method	Purpose
/api/orders/user/:userId	GET	Get all orders for a user
/api/order/:orderId/items	GET	Get all items for a specific order
/api/order/order-items/:itemId	PUT	Update quantity of an order item
/api/order/order-items/:itemId	DELETE	Remove an item from the cart
/api/payments	POST	Trigger payment for selected order(s)



GET `/api/orders/user/{userId}` Get orders by user ID

Returns a list of orders for a specific user

Parameters

Try it out

Name	Description
userId * required integer *(path)*	userId

Responses

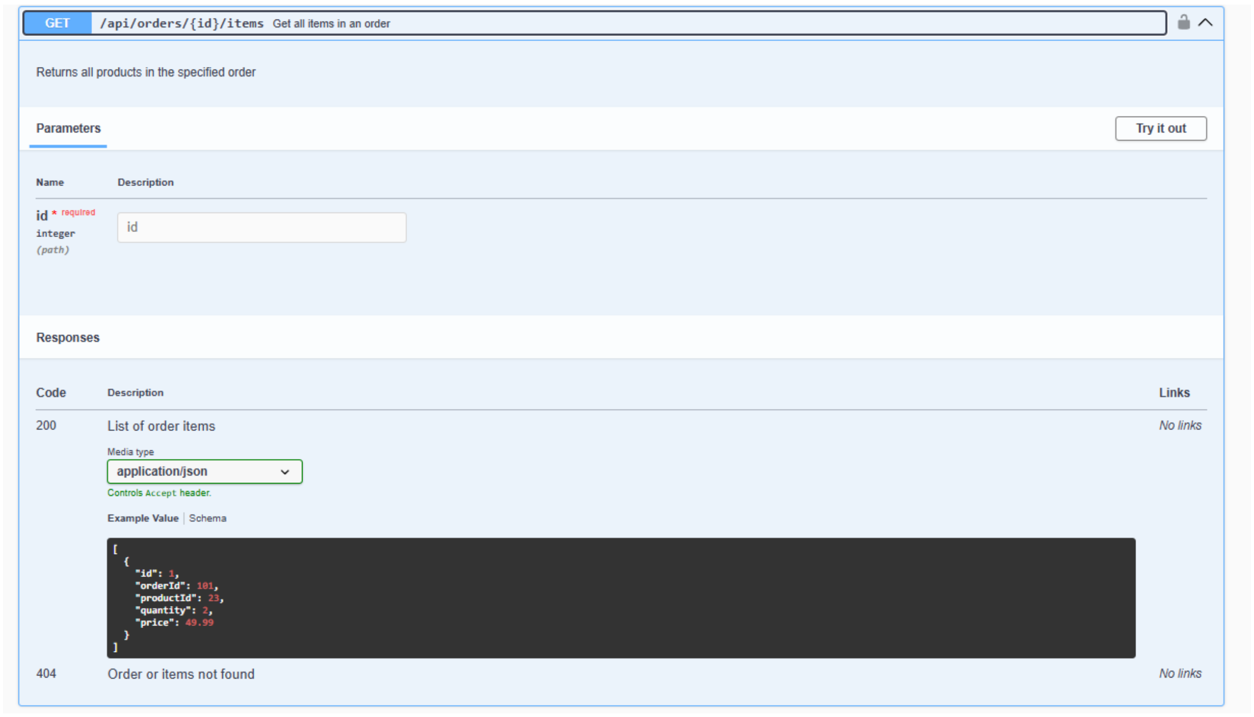
Code	Description	Links
200	List of user orders	*No links*
404	No orders found for the user	*No links*

Media type: application/json

Controls `Accept` header.

Example Value | Schema

```
[
  {
    "id": 1,
    "userId": 42,
    "totalAmount": 199.99,
    "status": "pending",
    "createdAt": "2025-07-27T15:48:44.107Z",
    "updatedAt": "2025-07-27T15:48:44.107Z"
  }
]
```



GET `/api/orders/{id}/items` Get all items in an order

Returns all products in the specified order

Parameters

Try it out

Name	Description
id * required integer *(path)*	id

Responses

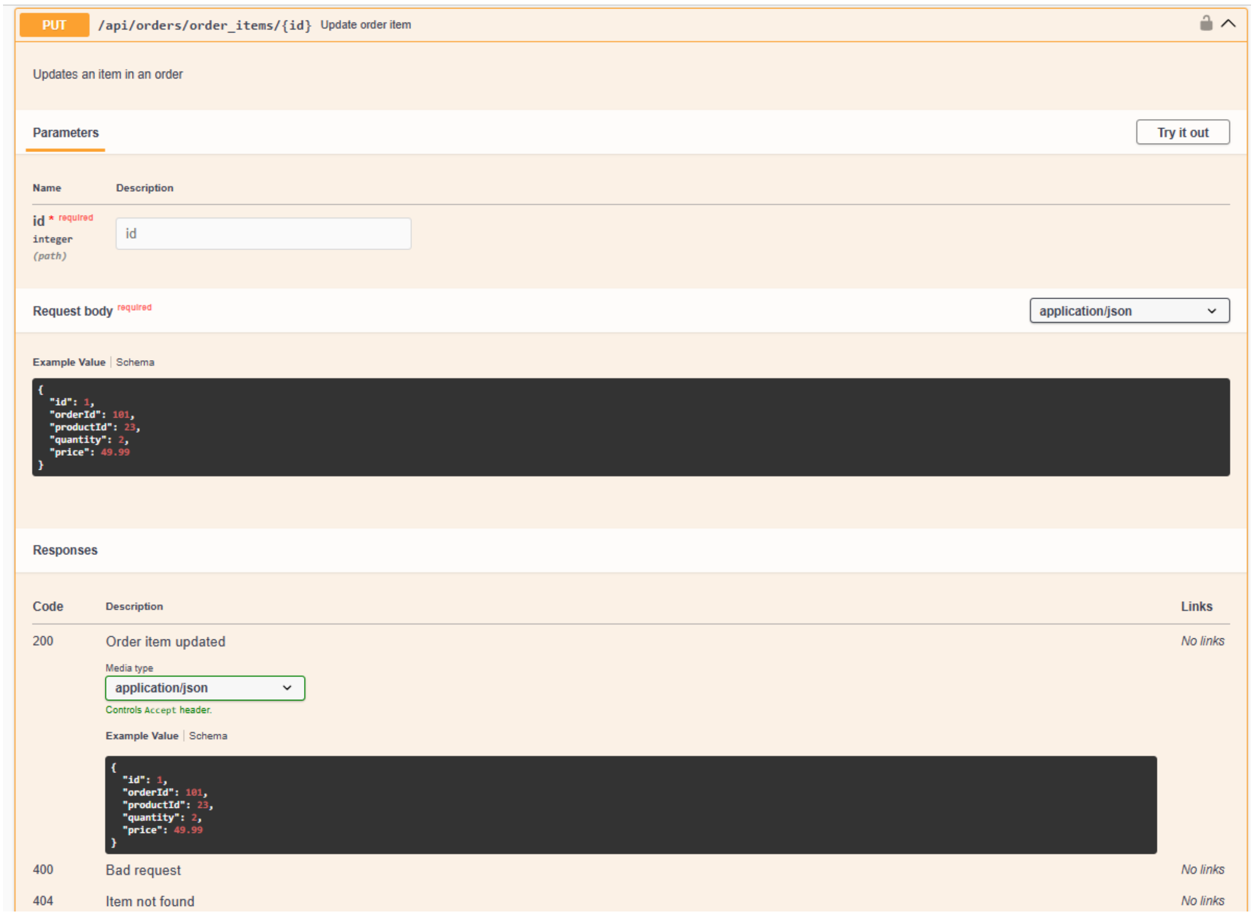
Code	Description	Links
200	List of order items	*No links*
404	Order or items not found	*No links*

Media type: application/json

Controls `Accept` header.

Example Value | Schema

```
[
  {
    "id": 1,
    "orderId": 101,
    "productId": 23,
    "quantity": 2,
    "price": 49.99
  }
]
```



PUT `/api/orders/order_items/{id}` Update order item

Updates an item in an order

Parameters — Try it out

Name	Description
id * required integer (path)	id

Request body required — application/json

Example Value | Schema

```
{
  "id": 1,
  "orderId": 101,
  "productId": 23,
  "quantity": 2,
  "price": 49.99
}
```

Responses

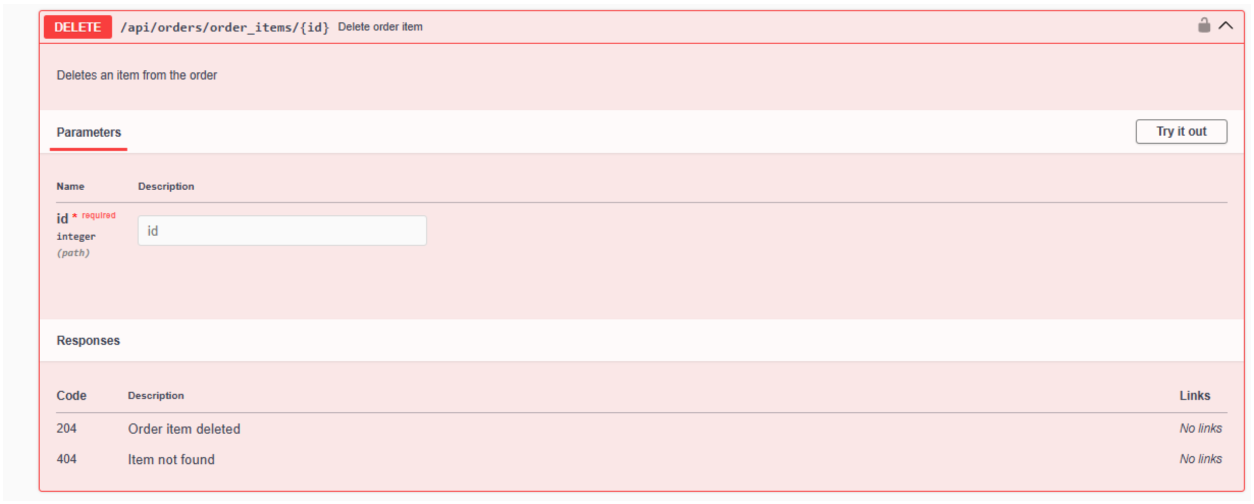
Code	Description	Links
200	Order item updated	No links
400	Bad request	No links
404	Item not found	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "id": 1,
  "orderId": 101,
  "productId": 23,
  "quantity": 2,
  "price": 49.99
}
```



DELETE `/api/orders/order_items/{id}` Delete order item

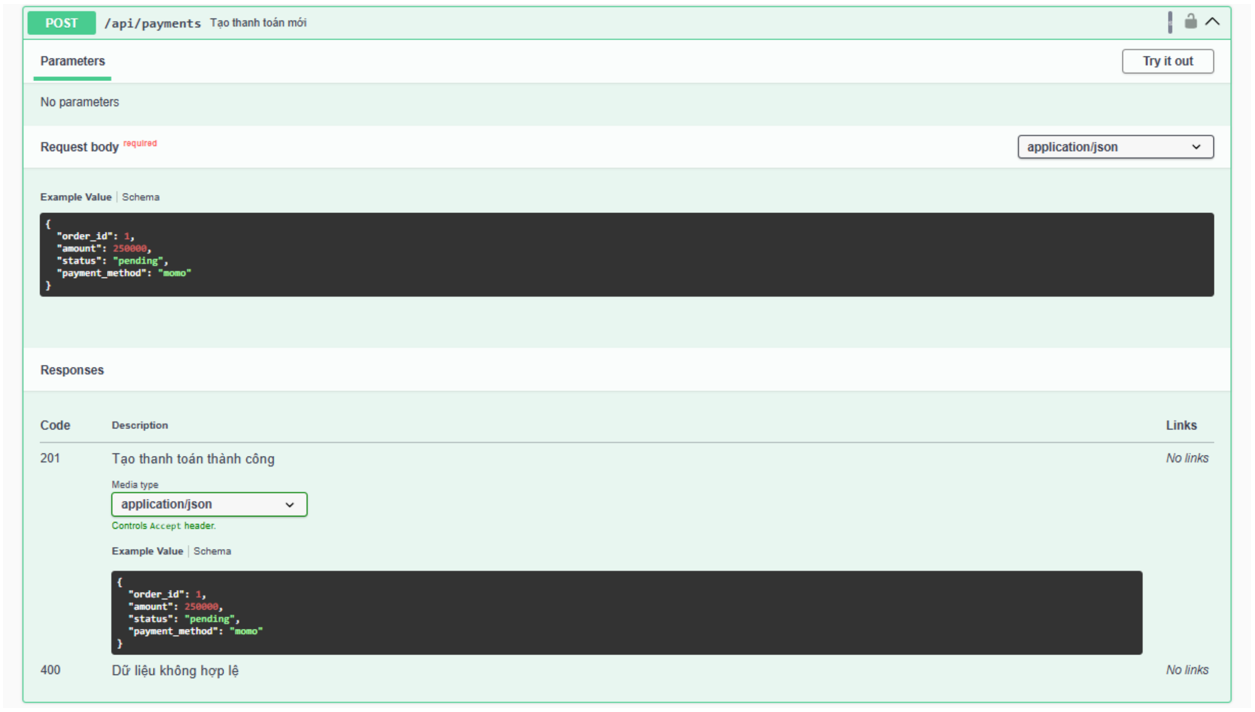
Deletes an item from the order

Parameters — Try it out

Name	Description
id * required integer (path)	id

Responses

Code	Description	Links
204	Order item deleted	No links
404	Item not found	No links


POST /api/payments Tạo thanh toán mới

Parameters

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "order_id": 1,
  "amount": 250000,
  "status": "pending",
  "payment_method": "momo"
}
```

Responses

Code	Description	Links
201	Tạo thanh toán thành công	No links
400	Dữ liệu không hợp lệ	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

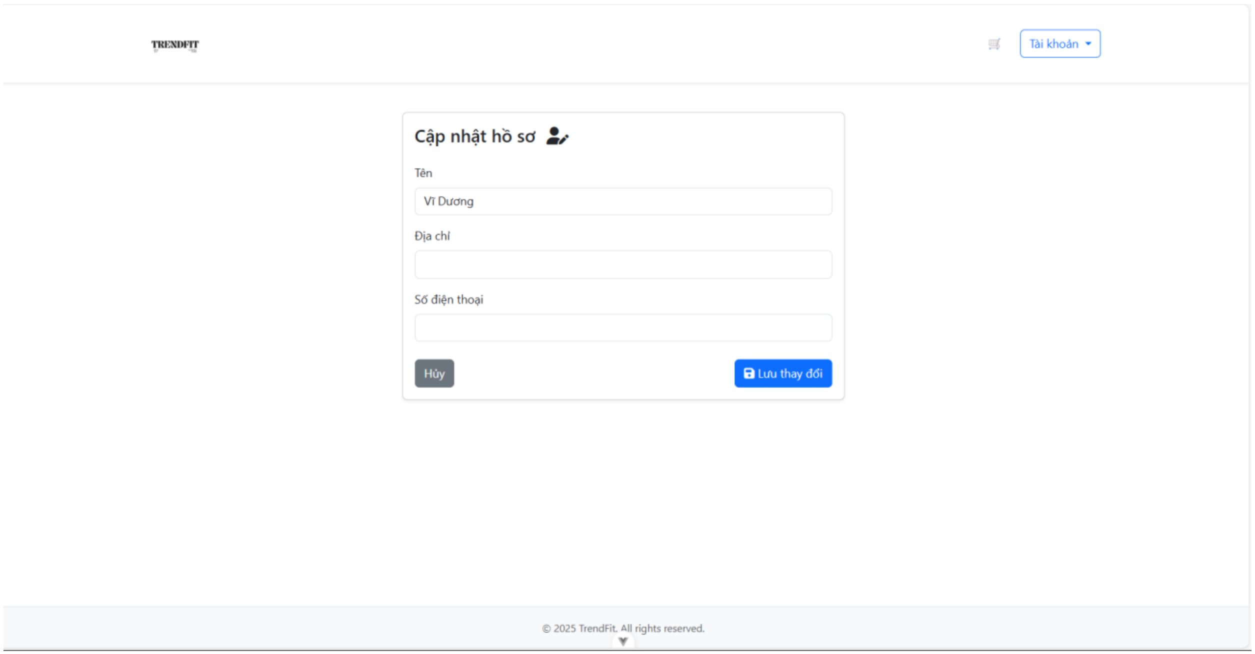
```
{
  "order_id": 1,
  "amount": 250000,
  "status": "pending",
  "payment_method": "momo"
}
```

- Data read/stored:
  - Tables:
    - Products
    - Orders
    - Order_items
- Client-side states needed:

Variable	Description
selectedOrderItems	Items in the active cart/order
totalAmount	Total calculated cost of the current cart
showPaymentModal	Boolean to control modal visibility
notificationMessage, notificationType	For displaying success or error messages
loadingCart (optional)	Show loading spinner while fetching data

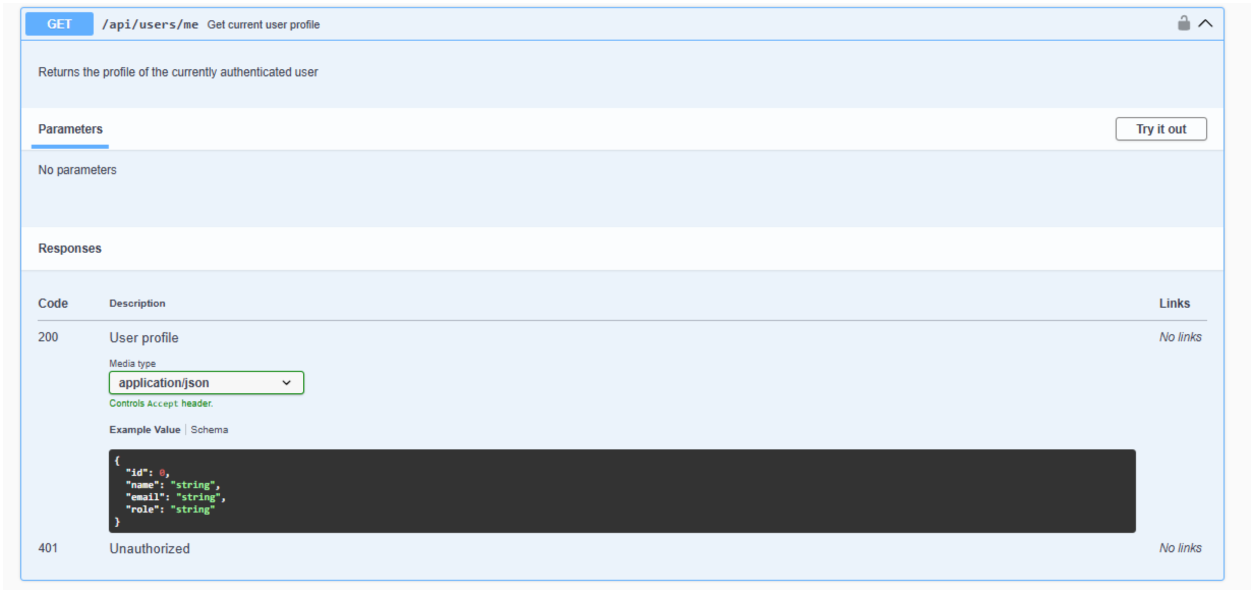
5. Feature / Application page 5: User Profile Update

- **Description**: This page allows authenticated users to update their personal information, such as name, address, and phone number. It fetches the current user data, pre-fills the form, and submits the updated profile to the backend.



- **Implementation details:**
  + Libraries used:
    - Frontend:
      - Pinia (state management)
    - Backend:
      - Zod
      - ApiError
      - JSend
  + Server-side APIs used:

Endpoint	Method	Purpose
/api/users/me	GET	Fetch current user’s profile
/api/users/me	PATCH	Update current user’s profile



GET /api/users/me Get current user profile

Returns the profile of the currently authenticated user

Parameters

No parameters

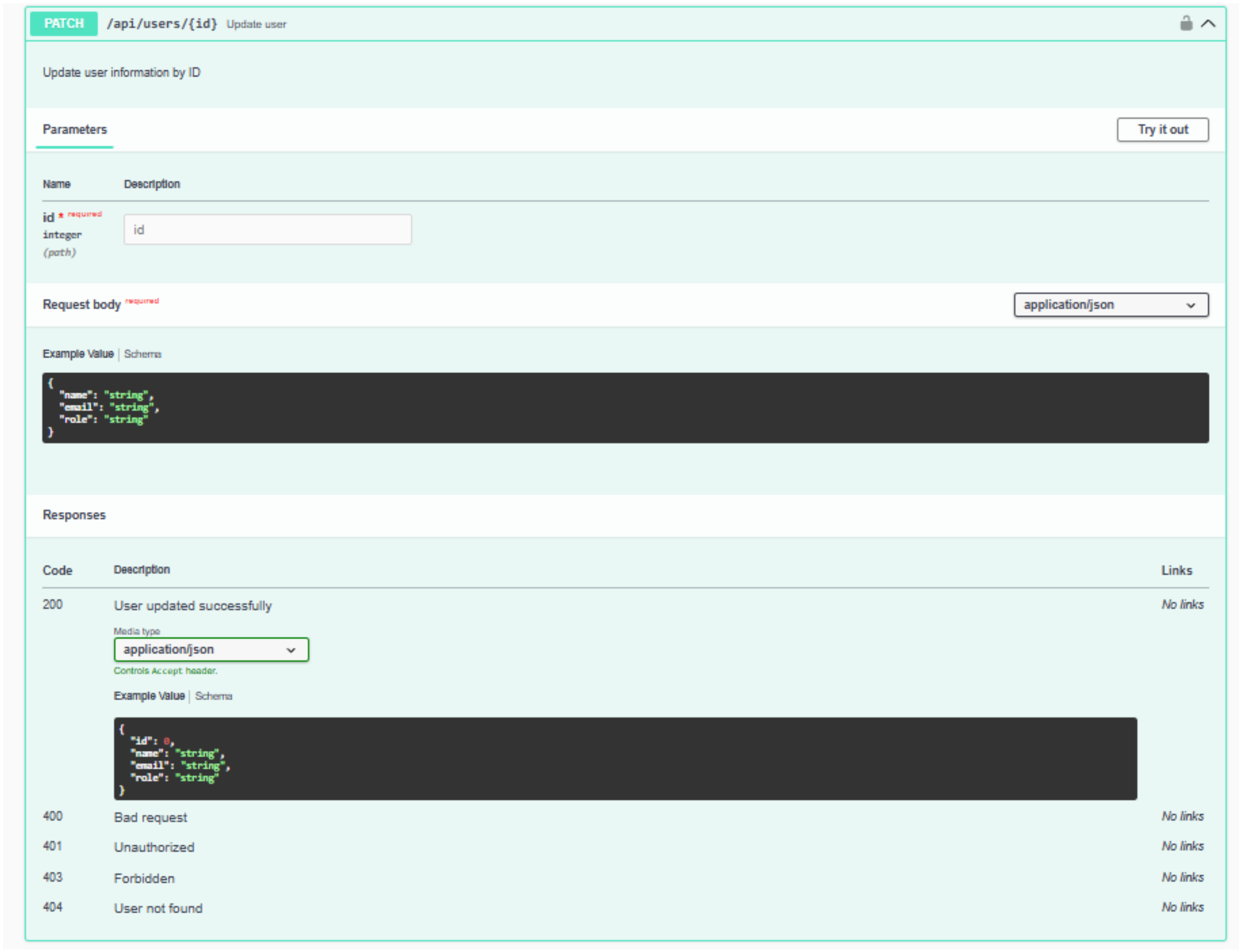
Responses

Code	Description	Links
200	User profile	No links
401	Unauthorized	No links

Media type: application/json — Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "email": "string",
  "role": "string"
}
```



PATCH /api/users/{id} Update user

Update user information by ID

Parameters

Name	Description
id * required integer (path)	id

Request body required — application/json

Example Value | Schema

```
{
  "name": "string",
  "email": "string",
  "role": "string"
}
```

Responses

Code	Description	Links
200	User updated successfully	No links
400	Bad request	No links
401	Unauthorized	No links
403	Forbidden	No links
404	User not found	No links

Media type: application/json — Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "email": "string",
  "role": "string"
}
```

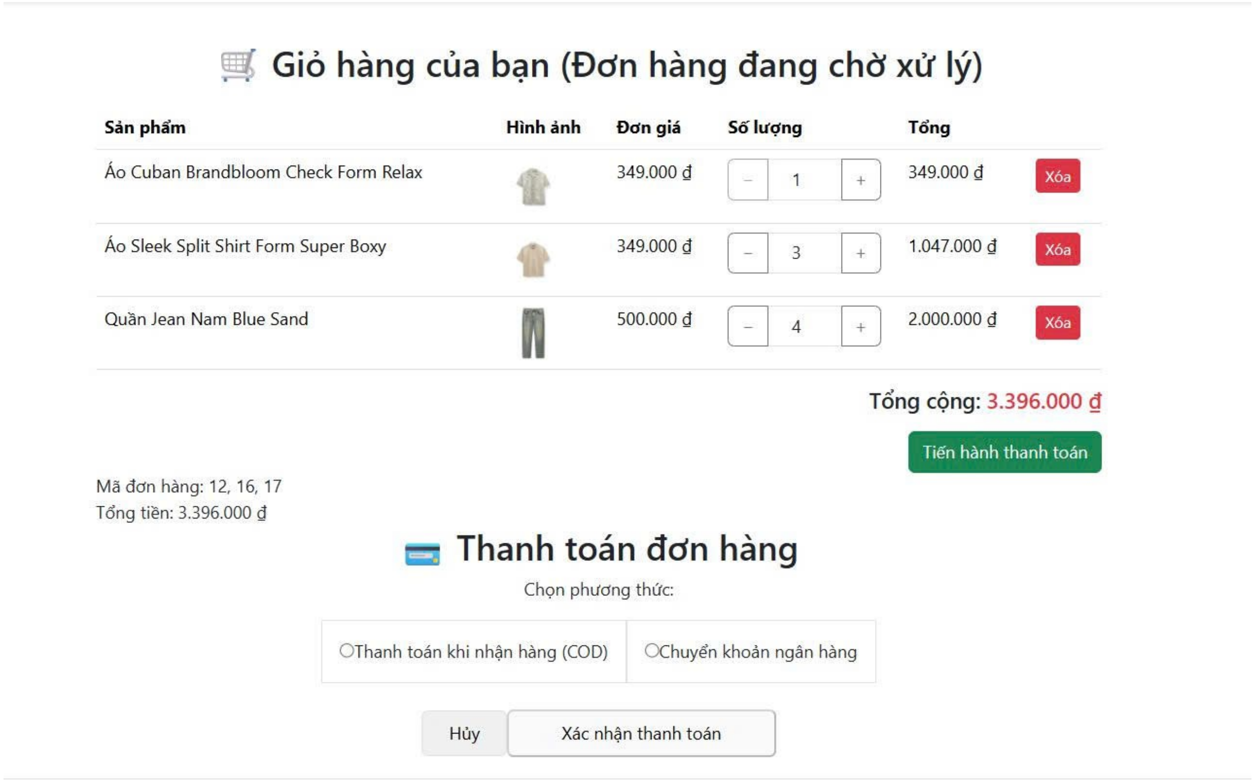
+ Data read/stored:
  - Tables:
    - users

+ Client-side states needed:

Variable	Description
name	User's full name (editable input)
address	User's address (editable input)
phone	User's phone number (editable input)
message	Notification message for success or error
submitting	Boolean indicating form submission/loading status
isLoading	Indicates if user data is being fetched
isError	Indicates if fetching user data failed
error	Error details if applicable

6. **Feature / Application page 6: Checkout Page**

- **Description**: A modal popup that allows users to complete payment for one or more orders in their cart. Users can:
    + View total cost and order IDs being paid.
    + Choose a payment method (e.g., Cash on Delivery or Bank Transfer).
    + Confirm or cancel the payment process.
    + Automatically close the modal if the cart becomes empty during the session.



🛒 Giỏ hàng của bạn (Đơn hàng đang chờ xử lý)

Sản phẩm	Hình ảnh	Đơn giá	Số lượng	Tổng	
Áo Cuban Brandbloom Check Form Relax		349.000 ₫	− 1 +	349.000 ₫	Xóa
Áo Sleek Split Shirt Form Super Boxy		349.000 ₫	− 3 +	1.047.000 ₫	Xóa
Quần Jean Nam Blue Sand		500.000 ₫	− 4 +	2.000.000 ₫	Xóa

Tổng cộng: 3.396.000 ₫

Tiến hành thanh toán

Mã đơn hàng: 12, 16, 17
Tổng tiền: 3.396.000 ₫

💳 Thanh toán đơn hàng

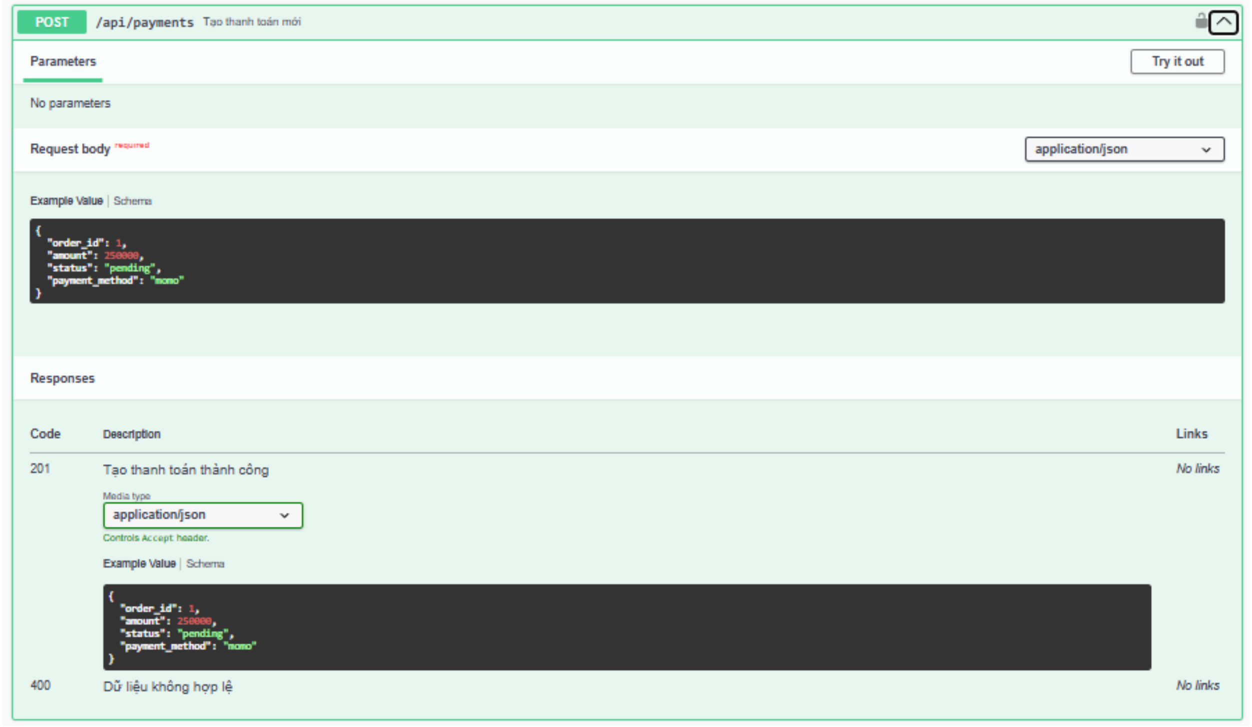
Chọn phương thức:

○Thanh toán khi nhận hàng (COD) ○Chuyển khoản ngân hàng

Hủy Xác nhận thanh toán

- **Implementation details:**
  + Libraries used:
    - Frontend:
      - Pinia (state management)
    - Backend:
      - Zod
      - ApiError
      - JSend
  + Server-side APIs used:

Endpoint	Method	Purpose
/api/payments	POST	Create new payment



```
{
  "order_id": 1,
  "amount": 250000,
  "status": "pending",
  "payment_method": "momo"
}
```

Code	Description	Links
201	Tạo thanh toán thành công	No links
400	Dữ liệu không hợp lệ	No links

```
{
  "order_id": 1,
  "amount": 250000,
  "status": "pending",
  "payment_method": "momo"
}
```

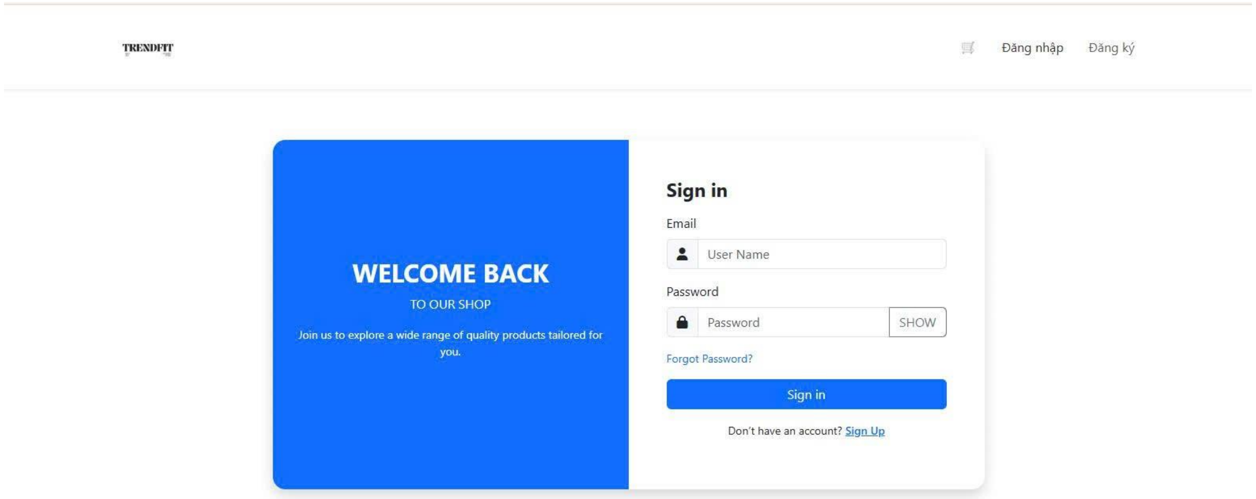
  + Data read/stored:
    - Tables:
      - Orders
      - payments
  + Client-side states needed:

Variable	Description
name	User's full name (editable input)
address	User's address (editable input)
phone	User's phone number (editable input)

message	Notification message for success or error
submitting	Boolean indicating form submission/loading status
isLoading	Indicates if user data is being fetched
isError	Indicates if fetching user data failed
error	Error details if applicable

7. **Feature / Application page 7: Login Page**

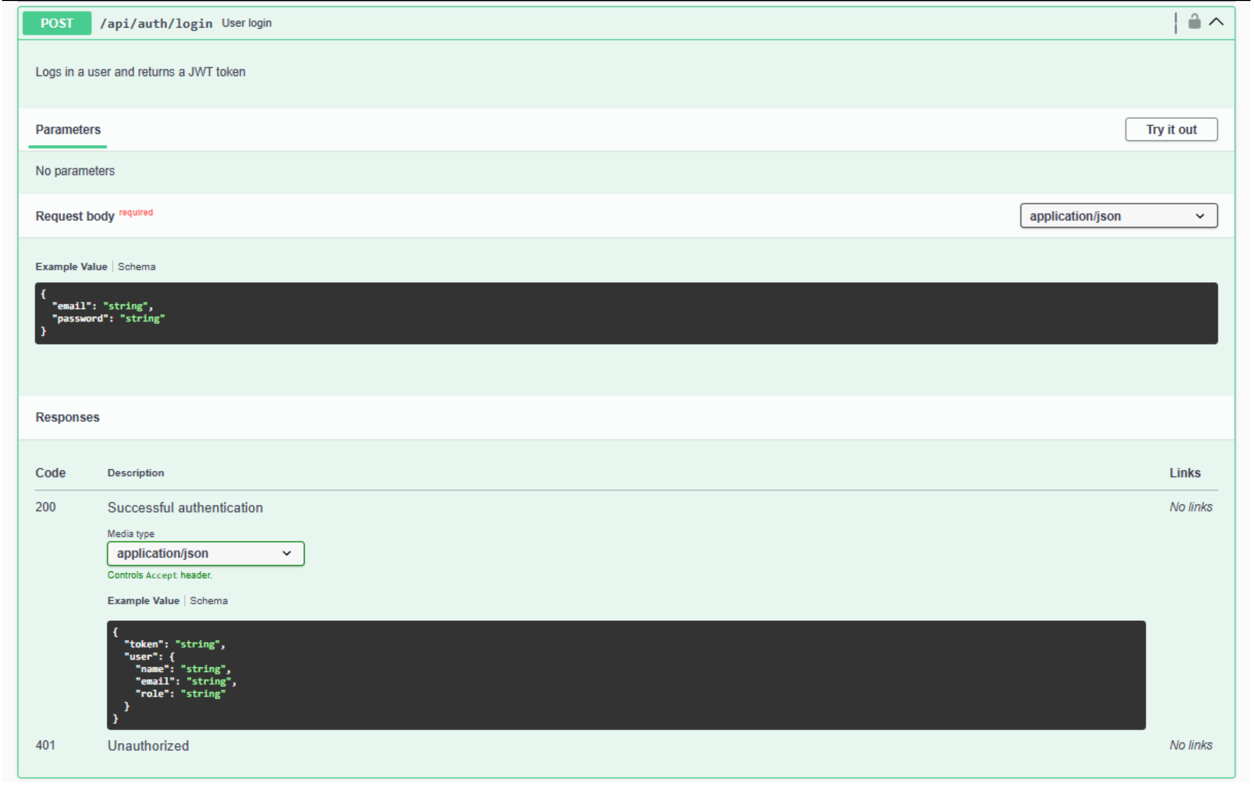
- **Description**: The login page allows users to access the system by entering their email and password. Upon successful authentication, the user is redirected based on their role
    + Admin users are redirected to the Admin Dashboard
    + Regular users are redirected to the main Homepage.
    + If credentials are invalid, a user-friendly error message is displayed.



- **Implementation details:**
    + Libraries used:
        ▪ Frontend:
            • Pinia (state management)
        ▪ Backend:
            • JWT
            • JSend
            • ApiError
    + Server-side APIs used:

Endpoint	Method	Purpose

/api/auth/login	GET	Authenticate user credentials


POST /api/auth/login User login

Logs in a user and returns a JWT token

Parameters

No parameters

Request body required application/json

Example Value | Schema

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

Responses

Code	Description	Links
200	Successful authentication Media type application/json Controls Accept header.	No links
401	Unauthorized	No links

Example Value | Schema

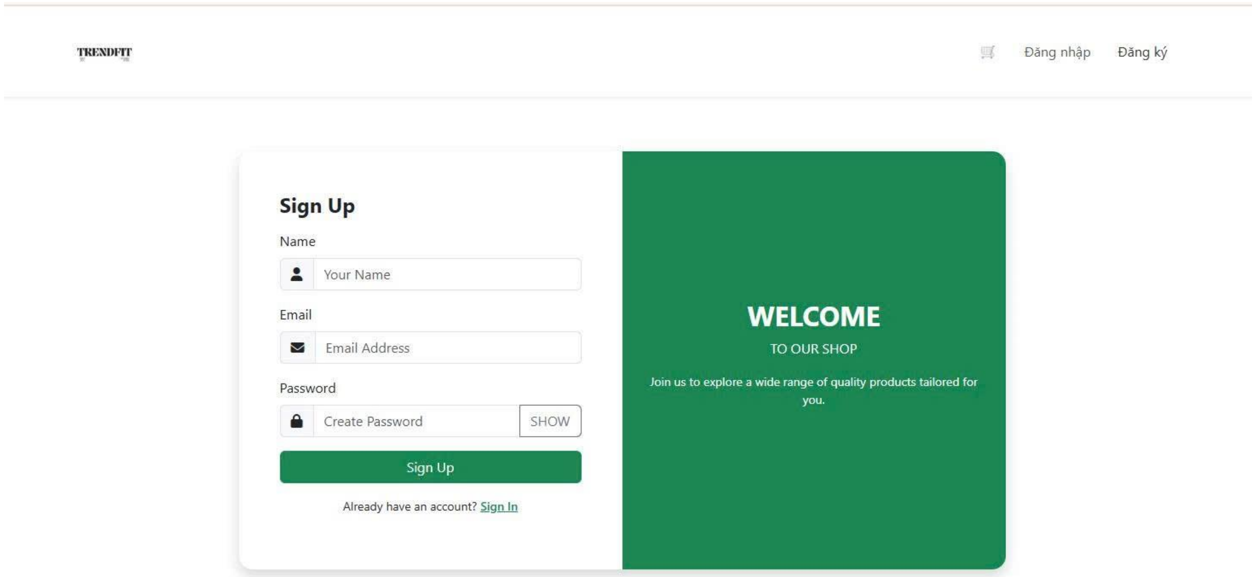
```
{
  "token": "string",
  "user": {
    "name": "string",
    "email": "string",
    "role": "string"
  }
}
```

+ Data read/stored:
    - Tables:
        - users
+ Client-side states needed:

Variable	Description
form	Object containing email and password
loading	Boolean to indicate login is in progress
error	String showing the error message if login fails
showPassword	Boolean toggle for password field visibility
router	Used to navigate to Home or Admin dashboard after login
Variable	Description
form	Object containing email and password
loading	Boolean to indicate login is in progress

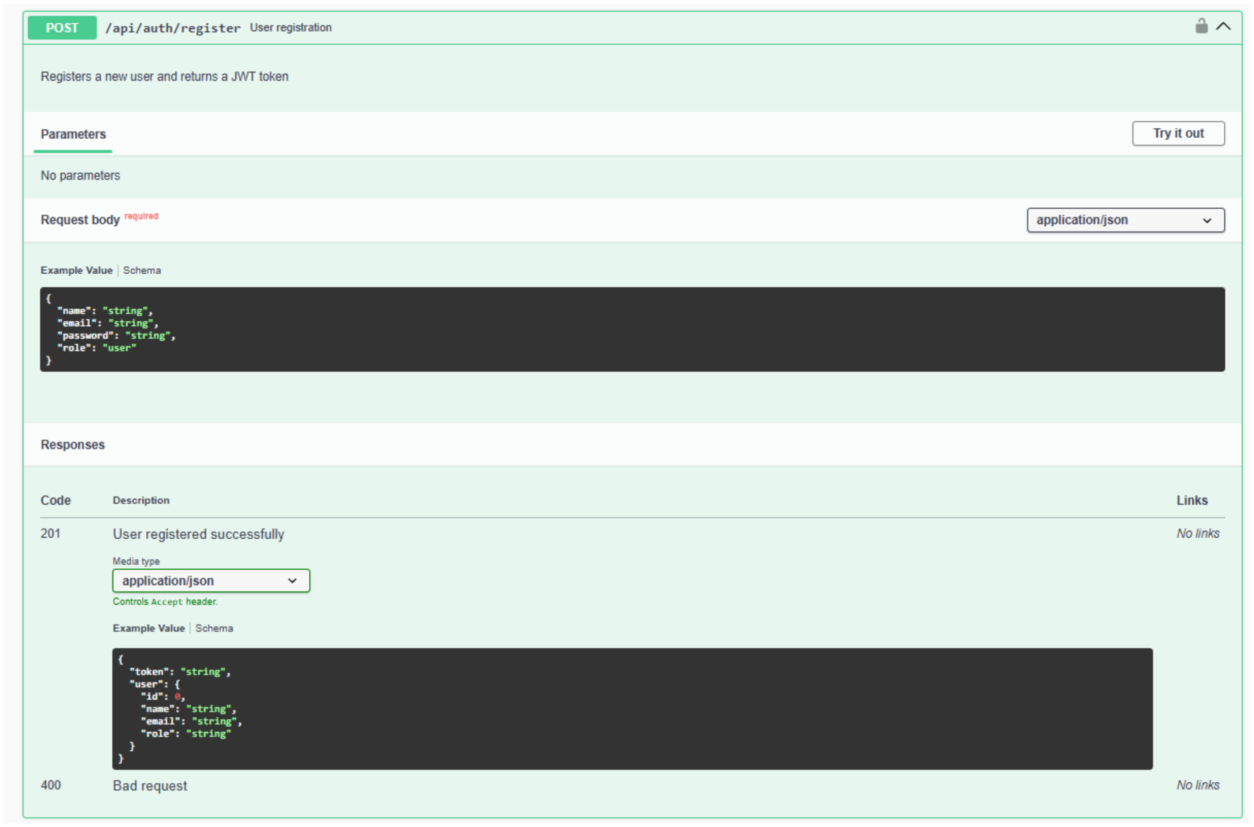
8. Feature / Application page 8: Sign up Page

- **Description**: A registration interface where new users can sign up by providing their name, email, and password. The system creates a new user account with a default role (user) and redirects them to the homepage upon successful registration
    + Input fields with icons for better UX
    + Password visibility toggle
    + Loading spinner and error handling
    + Link to sign-in page for existing users
    + Default role assignment (user)



- **Implementation details:**
    + Libraries used:
        - Frontend:
            - Pinia (state management)
        - Backend:
            - Bcrypt (for password hashing)
            - JWT
            - JSend
            - ApiError
    + Server-side APIs used:

Endpoint	Method	Purpose
/api/auth/register	POST	Create a new user account


POST /api/auth/register User registration

Registers a new user and returns a JWT token

Parameters

No parameters

Request body required — application/json

Example Value | Schema

```
{
  "name": "string",
  "email": "string",
  "password": "string",
  "role": "user"
}
```

Responses

Code	Description	Links
201	User registered successfully Media type: application/json Controls Accept header. Example Value | Schema	No links
400	Bad request	No links

```
{
  "token": "string",
  "user": {
    "id": 0,
    "name": "string",
    "email": "string",
    "role": "string"
  }
}
```

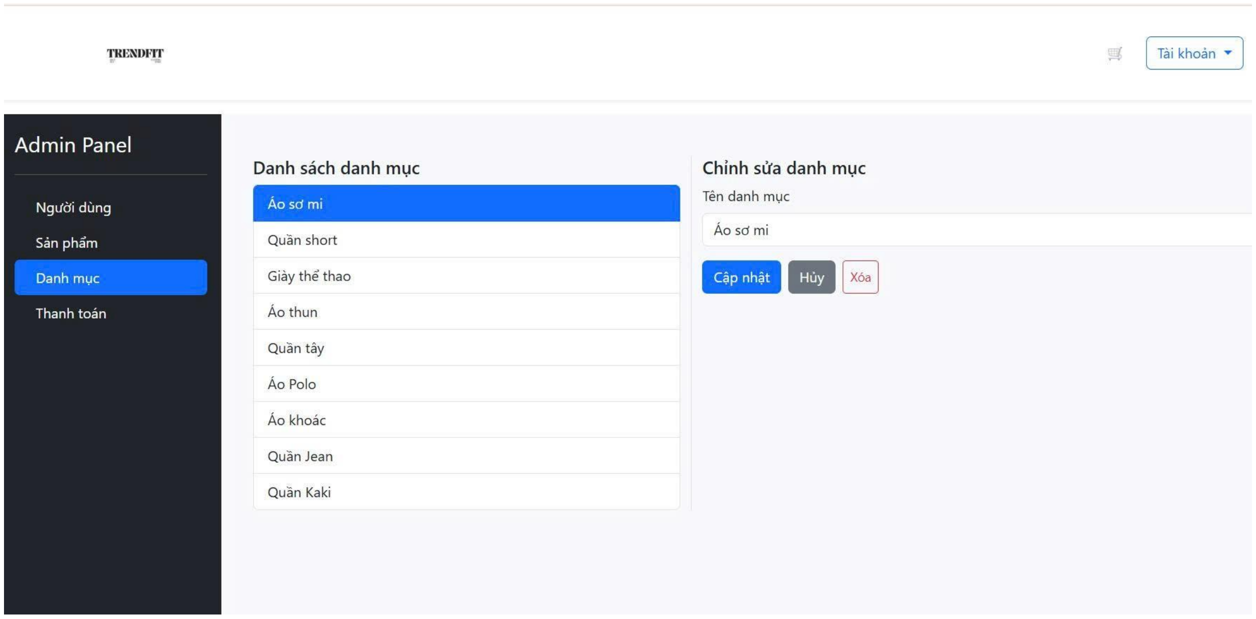
+ Data read/stored:
  - Tables:
    - users
+ Client-side states needed:

Variable	Description
form	Object storing name, email, password, role
loading	Boolean to indicate registration process is active
error	String showing error message if registration fails
showPassword	Boolean to toggle password visibility
router	Used to navigate to Home after successful registration
Variable	Description
form	Object storing name, email, password, role
loading	Boolean to indicate registration process is active

9. **Feature / Application page 9: Category Management Page**

- **Description**: A management interface for handling product categories. This page enables the admin to create, update, select, or delete categories in real-time. It syncs data with the backend and reflects updates instantly using Vue Query caching and store state management.

- Displays list of existing categories
- Allows adding a new category
- Allows selecting and editing a category
- Allows deleting a selected category
- Uses global store for managing selected state



Danh sách danh mục

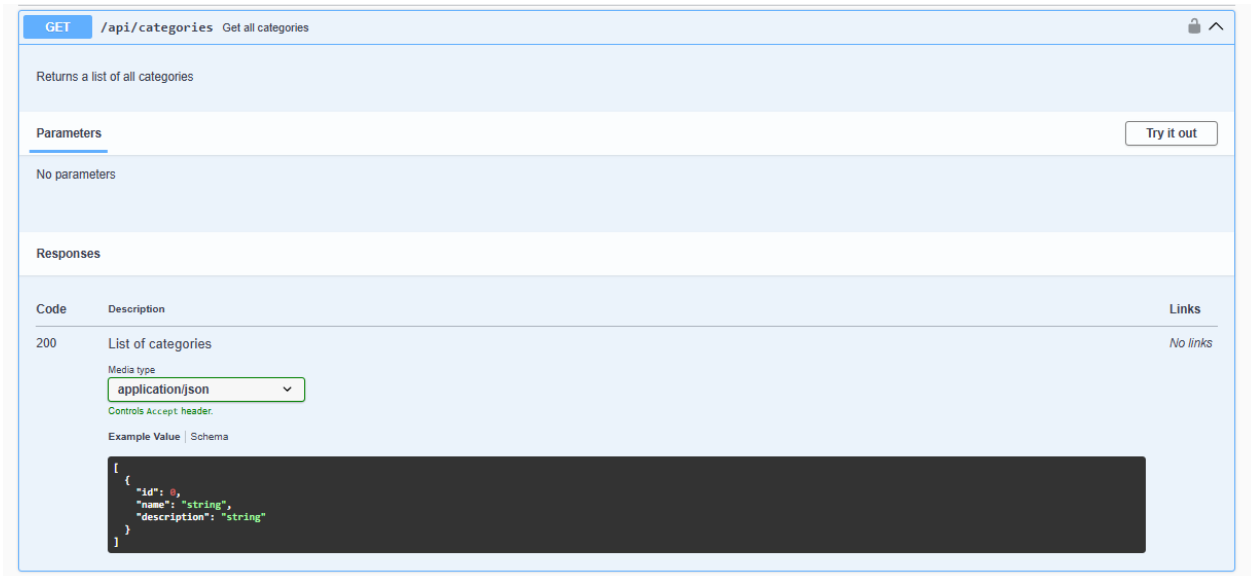
- Áo sơ mi
- Quần short
- Giày thể thao
- Áo thun
- Quần tây
- Áo Polo
- Áo khoác
- Quần Jean
- Quần Kaki

Chỉnh sửa danh mục

Tên danh mục: Áo sơ mi

- **Implementation details:**
  - Libraries used:
    - Frontend:
      - Pinia (state management)
    - Backend:
      - JSend
      - ApiError
      - Zod
  - Server-side APIs used:

Endpoint	**Method**	**Purpose**
/api/categories	GET	Retrieve all categories
/api/categories	POST	Create a new category
/api/categories/:id	PATCH	Update a specific category
/api/categories/:id	DELETE	Delete a specific category



GET /api/categories Get all categories

Returns a list of all categories

Parameters

No parameters

Responses

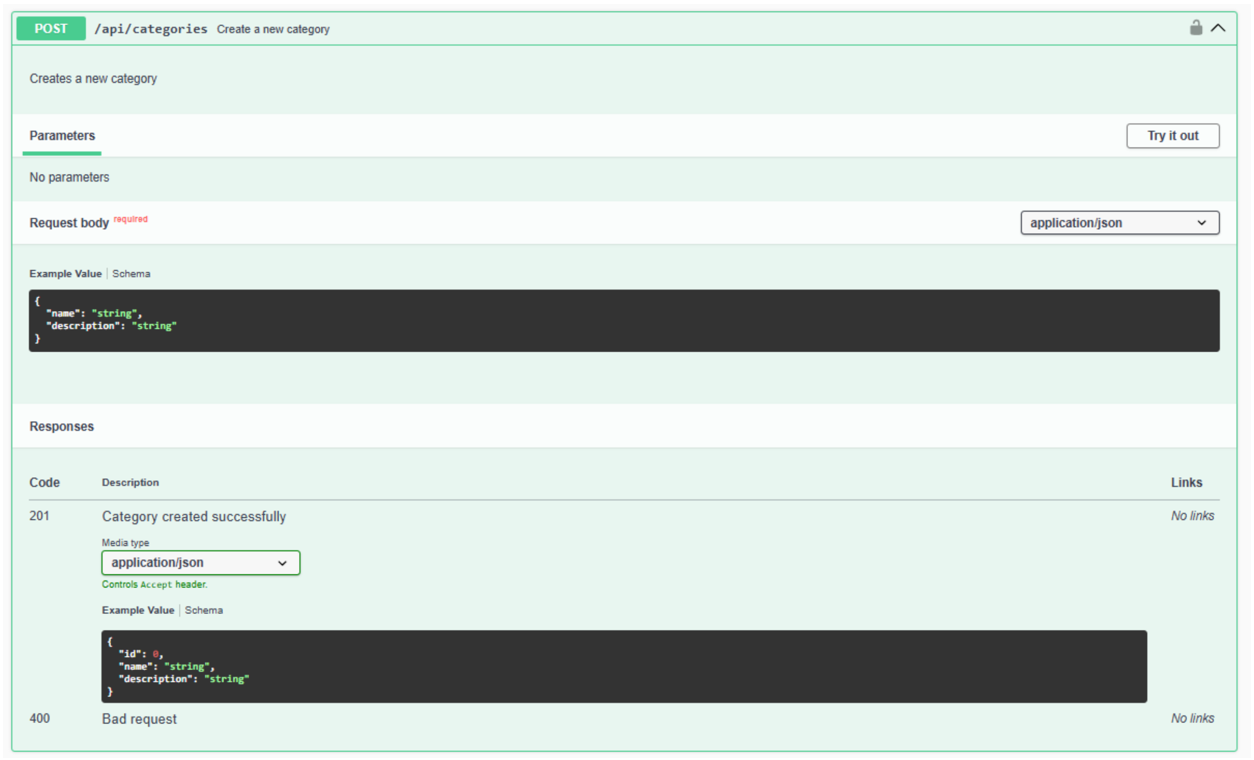
Code	Description	Links
200	List of categories	*No links*

Media type: application/json

Controls `Accept` header.

Example Value | Schema

```
[
  {
    "id": 0,
    "name": "string",
    "description": "string"
  }
]
```



POST /api/categories Create a new category

Creates a new category

Parameters

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "name": "string",
  "description": "string"
}
```

Responses

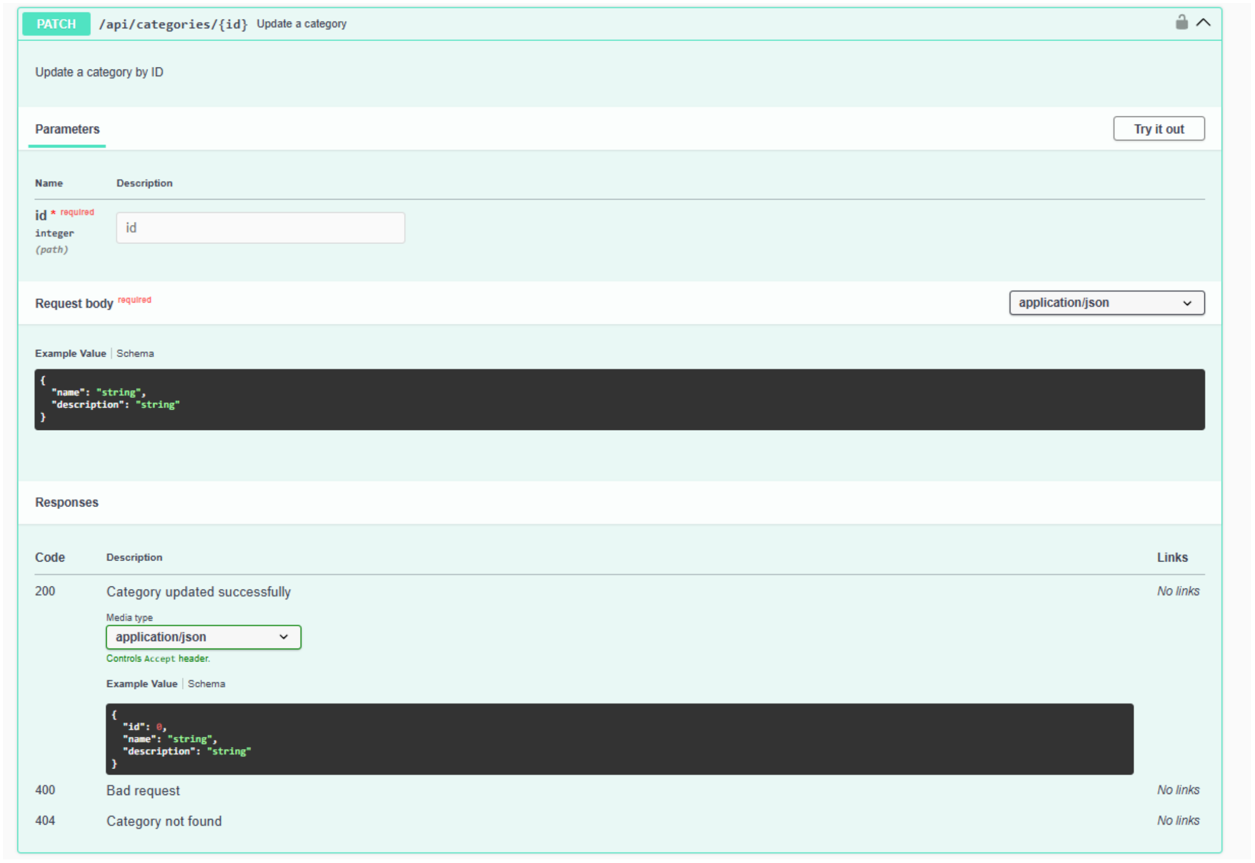
Code	Description	Links
201	Category created successfully	*No links*
400	Bad request	*No links*

Media type: application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "description": "string"
}
```


PATCH /api/categories/{id} Update a category

Update a category by ID

Parameters

Try it out

Name	Description
id * required integer (path)	id

Request body required

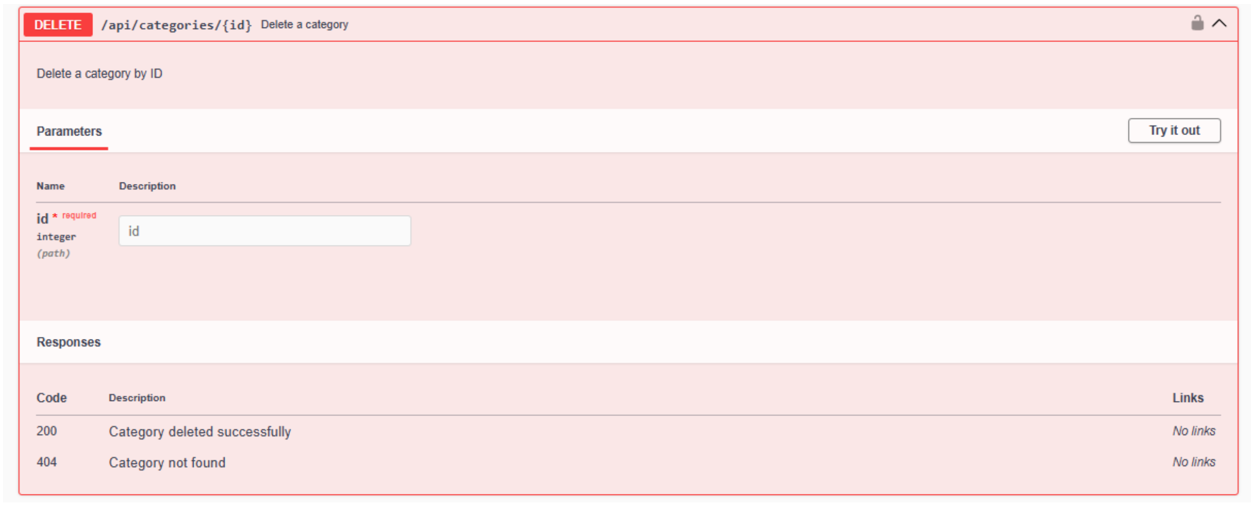
application/json

Example Value | Schema

```
{
  "name": "string",
  "description": "string"
}
```

Responses

Code	Description	Links
200	Category updated successfully Media type: application/json Controls Accept header. Example Value | Schema `{ "id": 0, "name": "string", "description": "string" }`	No links
400	Bad request	No links
404	Category not found	No links


DELETE /api/categories/{id} Delete a category

Delete a category by ID

Parameters

Try it out

Name	Description
id * required integer (path)	id

Responses

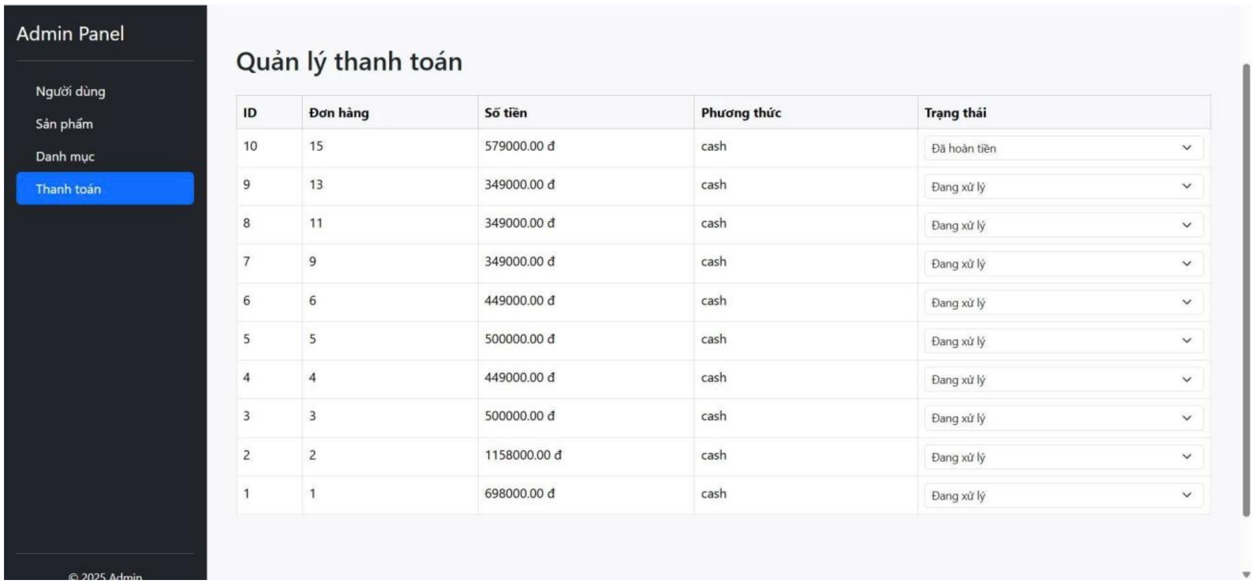
Code	Description	Links
200	Category deleted successfully	No links
404	Category not found	No links

+ Data read/stored:
    - Tables:
        - categories
+ Client-side states needed:

Variable	Description
form	Reactive object for category form (id, name)
categories	List of all categories (from store)
selectedCategory	Currently selected category (from store)
isLoading / isError	Booleans for loading/error state during fetching
creating / updating	Booleans to show loading state during creation or update
isSubmitting	Computed value combining creating and updating

10. Feature / Application page 10: Payment Management Page

- **Description**: This is an admin page designed for managing payment records associated with customer orders. Admin users can:
    + View all payment entries including ID, amount, method, and status
    + Dynamically update the status of a payment (e.g., from pending to paid)
    + Get immediate feedback on loading state or API errors
    + Automatically sync changes with backend through Vue Query cache invalidation

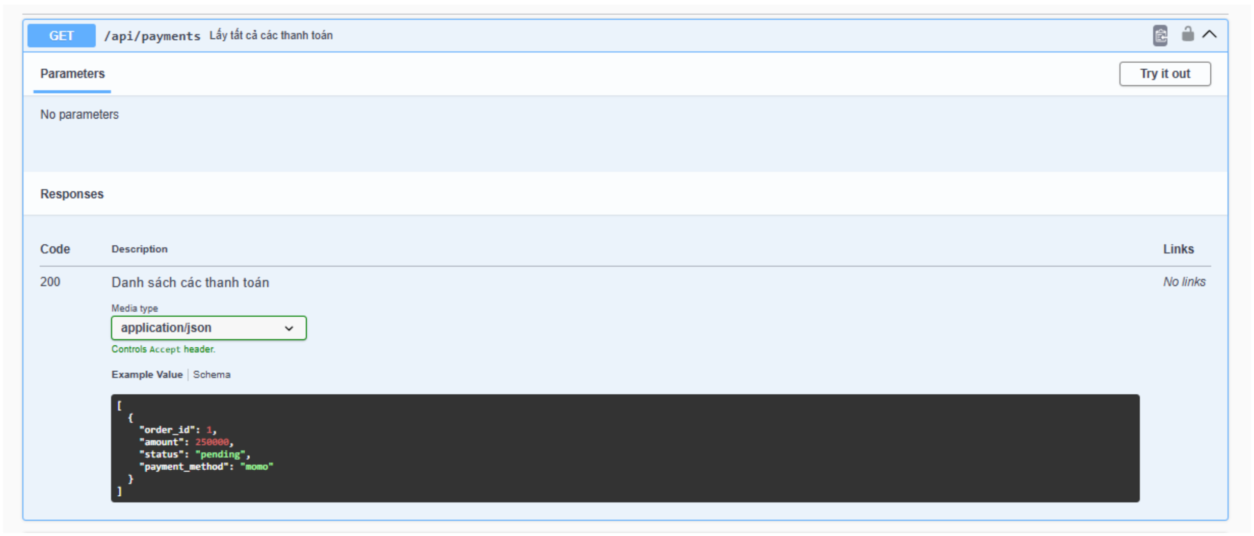


Quản lý thanh toán

ID	Đơn hàng	Số tiền	Phương thức	Trạng thái
10	15	579000.00 đ	cash	Đã hoàn tiền
9	13	349000.00 đ	cash	Đang xử lý
8	11	349000.00 đ	cash	Đang xử lý
7	9	349000.00 đ	cash	Đang xử lý
6	6	449000.00 đ	cash	Đang xử lý
5	5	500000.00 đ	cash	Đang xử lý
4	4	449000.00 đ	cash	Đang xử lý
3	3	500000.00 đ	cash	Đang xử lý
2	2	1158000.00 đ	cash	Đang xử lý
1	1	698000.00 đ	cash	Đang xử lý

- **Implementation details:**
  - Libraries used:
    - Frontend:
      - Pinia (state management)
    - Backend:
      - Zod
      - ApiError
      - JSend
  - Server-side APIs used:

Endpoint	**Method**	**Purpose**
/api/payments	GET	Retrieve list of all payments
/api/payments/:id	PUT	Update payment status


GET /api/payments Lấy tất cả các thanh toán

Parameters

Try it out

No parameters

Responses

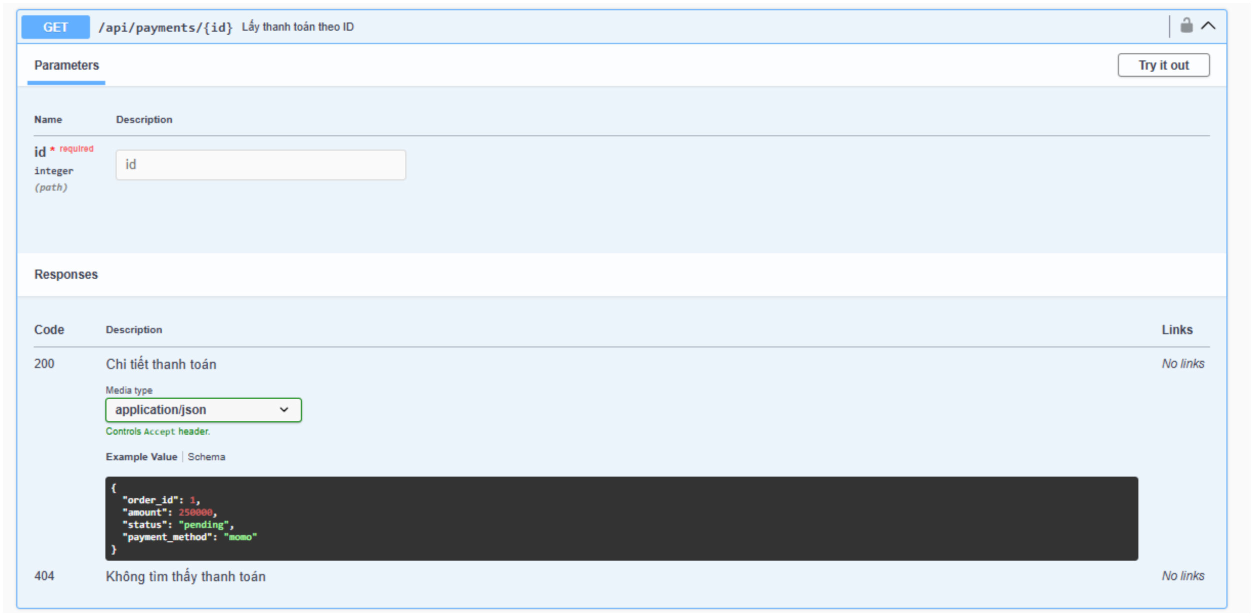
Code	Description	Links
200	Danh sách các thanh toán	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
[
  {
    "order_id": 1,
    "amount": 250000,
    "status": "pending",
    "payment_method": "momo"
  }
]
```



/api/payments/{id} Lấy thanh toán theo ID

Name	Description
id * required integer (path)	id

Responses

Code	Description	Links
200	Chi tiết thanh toán	No links
404	Không tìm thấy thanh toán	No links

Controls Accept header.

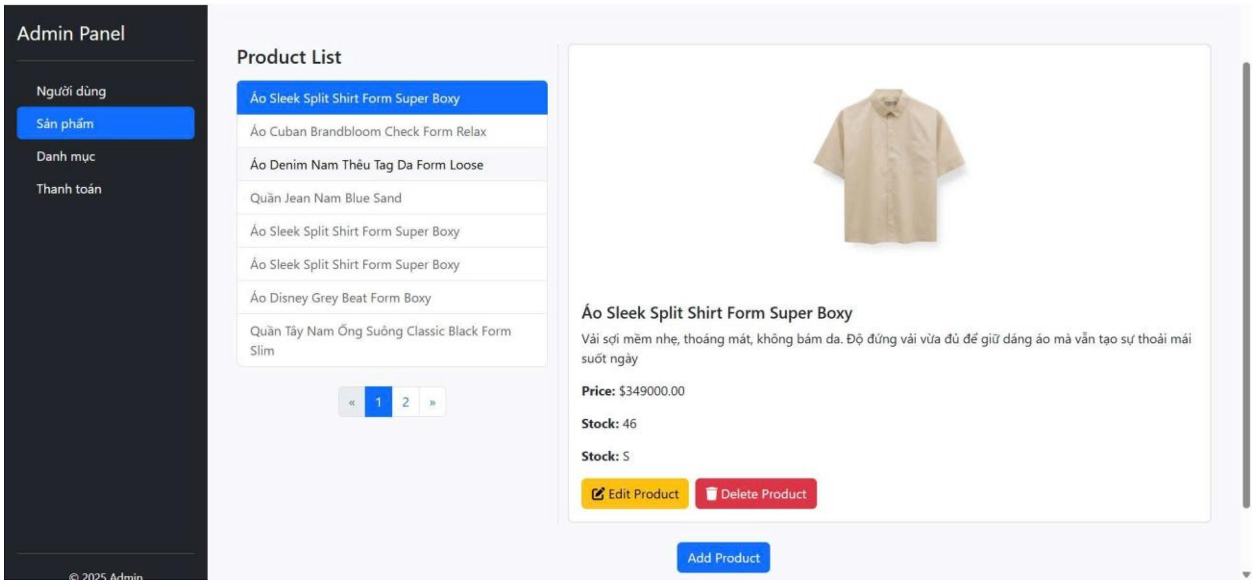
```
{
  "order_id": 1,
  "amount": 250000,
  "status": "pending",
  "payment_method": "momo"
}
```

+ Data read/stored:
  - Tables:
    - Payments
    - orders
+ Client-side states needed:

Variable	**Description**
allPayments	Array of all payment records (from Pinia store)
isLoading	Indicates whether payment list is being fetched
error	Error object returned if the fetch fails
updatePaymentStatus	Vue Query mutation for updating a specific payment
onStatusChange(payment)	Triggered when user changes payment status via dropdown

11. Feature / Application page 11: Products Management Page

- **Description**: This is an admin-facing page that allows authorized users to manage the entire product catalog. Users can:
  + View a paginated list of all available products
  + Select a product to view detailed information
  + Create new products via a modal popup
  + Edit existing product details (name, price, image, stock, etc.)
  + Delete products with confirmation
  + Upload product images and assign category relationships

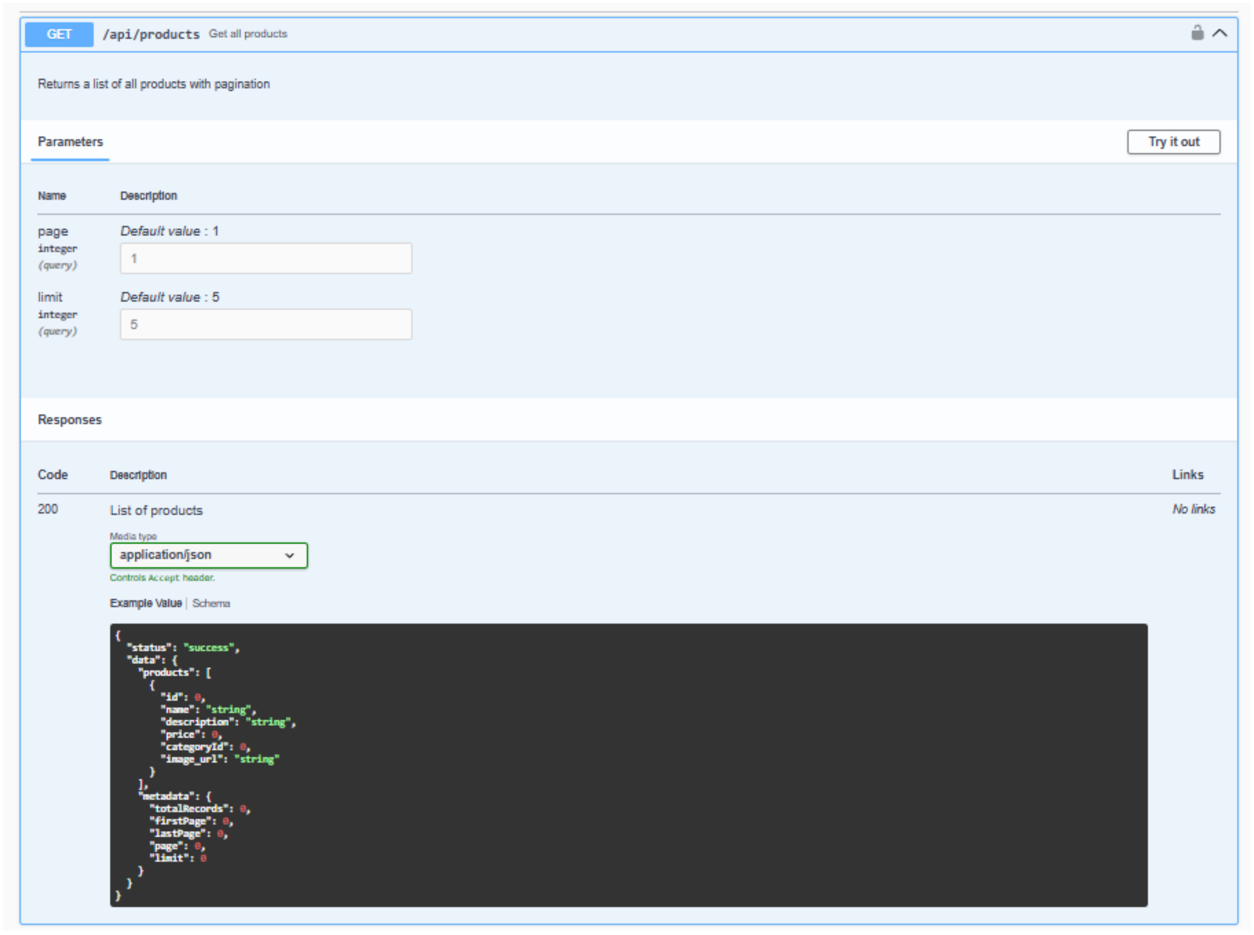


Áo Sleek Split Shirt Form Super Boxy
Áo Cuban Brandbloom Check Form Relax
Áo Denim Nam Thêu Tag Da Form Loose
Quần Jean Nam Blue Sand
Áo Sleek Split Shirt Form Super Boxy
Áo Sleek Split Shirt Form Super Boxy
Áo Disney Grey Beat Form Boxy
Quần Tây Nam Ống Suông Classic Black Form Slim

Áo Sleek Split Shirt Form Super Boxy
Vải sợi mềm nhẹ, thoáng mát, không bám da. Độ đứng vải vừa đủ để giữ dáng áo mà vẫn tạo sự thoải mái suốt ngày
Price: $349000.00
Stock: 46
Stock: S

- **Implementation details:**
  + Libraries used:
    - Frontend:
      - Pinia (state management)
    - Backend:
      - AWS SDK (file upload to S3)
      - JSend
      - ApiError
      - Zod
  + Server-side APIs used:

Endpoint	**Method**	**Purpose**
/api/products?page=1	GET	Fetch paginated product list
/api/products	POST	Create a new product
/api/products/:id	PATCH	Update an existing product
/api/products/:id	DELETE	Delete a product
/api/categories	GET	Get list of product categories



GET /api/products Get all products

Returns a list of all products with pagination

Parameters

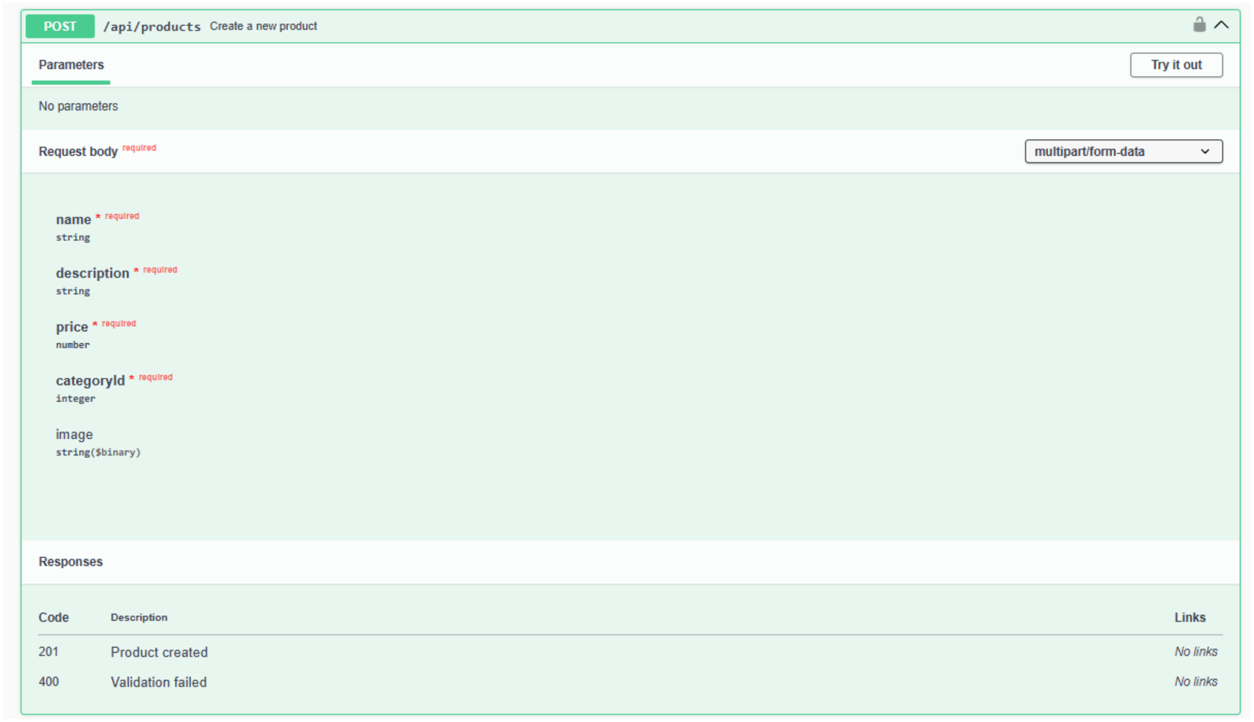
Try it out

Name	Description
page integer (query)	*Default value* : 1 1
limit integer (query)	*Default value* : 5 5

Responses

Code	Description	Links
200	List of products Media type: application/json Controls Accept header. Example Value | Schema	*No links*

```json
{
  "status": "success",
  "data": {
    "products": [
      {
        "id": 0,
        "name": "string",
        "description": "string",
        "price": 0,
        "categoryId": 0,
        "image_url": "string"
      }
    ],
    "metadata": {
      "totalRecords": 0,
      "firstPage": 0,
      "lastPage": 0,
      "page": 0,
      "limit": 0
    }
  }
}
```



POST /api/products Create a new product

Parameters

Try it out

No parameters

Request body required

multipart/form-data

name * required
string

description * required
string

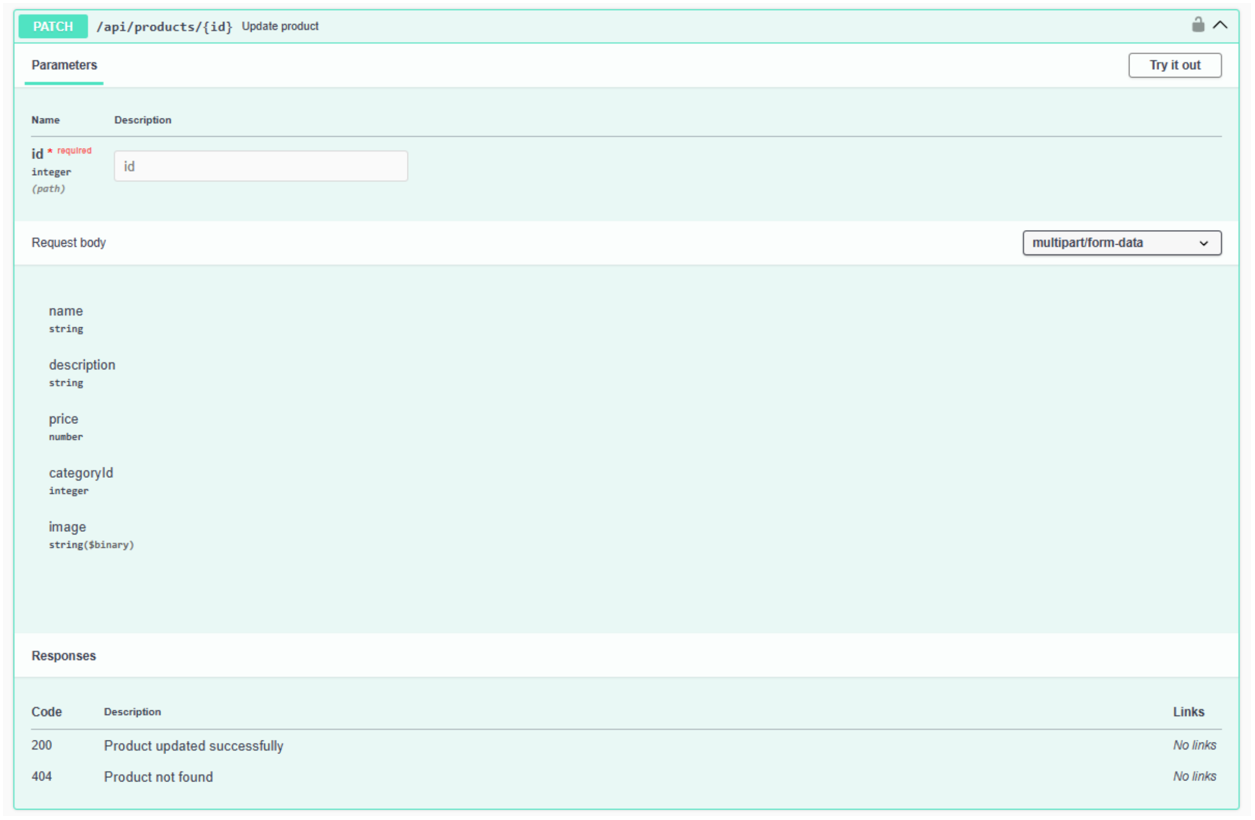
price * required
number

categoryId * required
integer

image
string($binary)

Responses

Code	Description	Links
201	Product created	*No links*
400	Validation failed	*No links*


PATCH /api/products/{id} Update product

Parameters | Try it out

Name	Description
id * required integer (path)	id

Request body — multipart/form-data

name
string

description
string

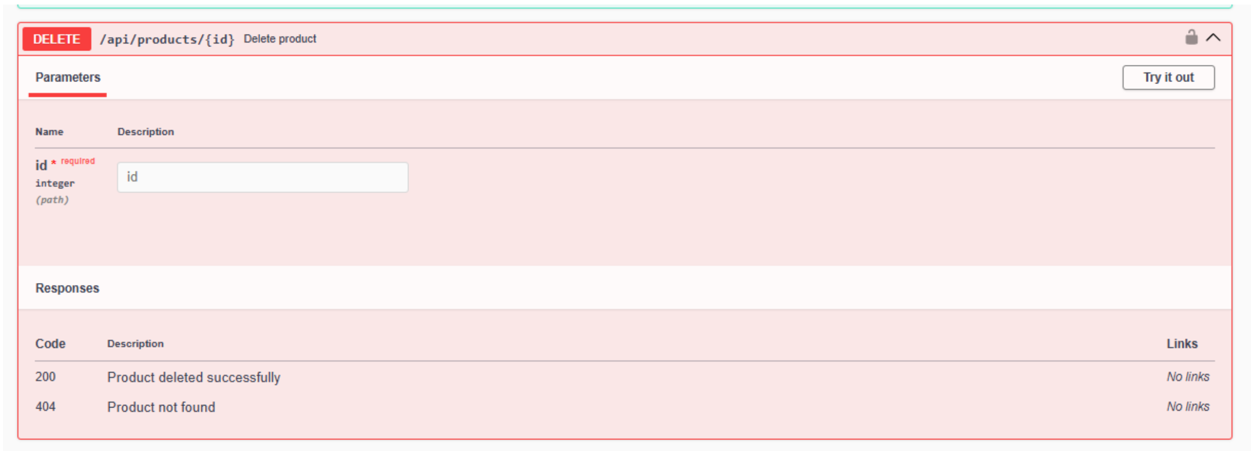
price
number

categoryId
integer

image
string($binary)

Responses

Code	Description	Links
200	Product updated successfully	No links
404	Product not found	No links


DELETE /api/products/{id} Delete product

Parameters | Try it out

Name	Description
id * required integer (path)	id

Responses

Code	Description	Links
200	Product deleted successfully	No links
404	Product not found	No links

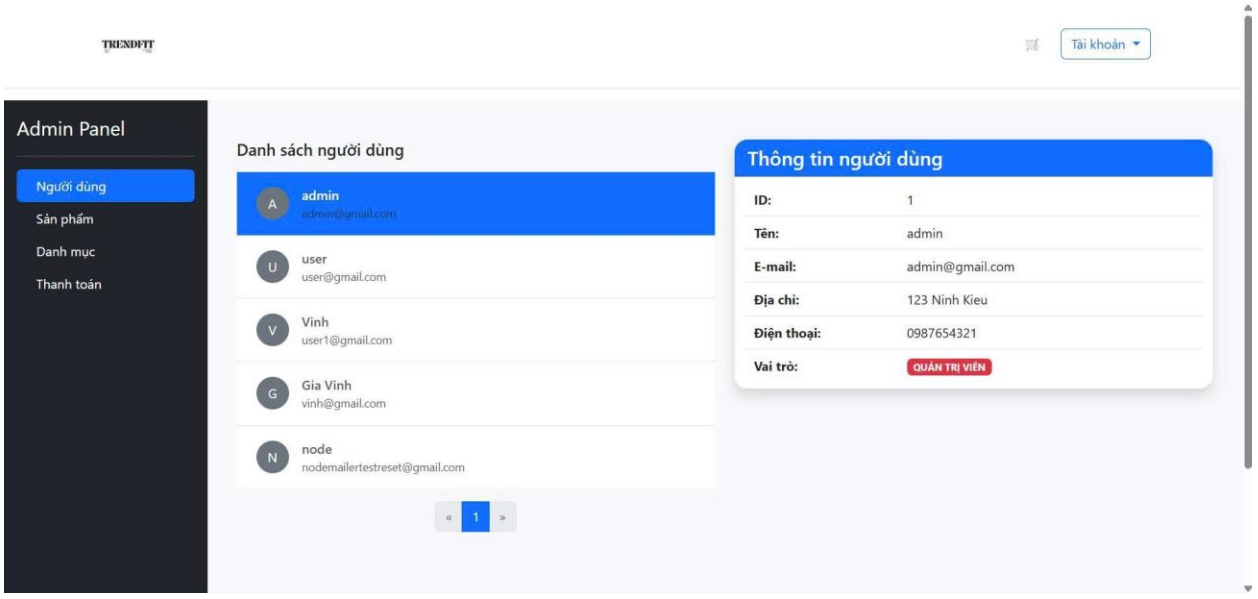
+ Data read/stored:
    - Tables:
        - Products
        - categories
+ Client-side states needed:

Variable	**Description**
products	List of products on the current page
selectedProduct	Product currently selected by the admin
newProduct	Form state for creating a new product

editedProduct	Form state for editing an existing product
imageFile	File object for the updated image (edit mode)
totalPages	Total number of product pages (used in pagination)
currentPage	Current page extracted from the route query
showEditModal	Boolean controlling visibility of edit modal
showCreateModal	Boolean controlling visibility of create modal

12. Feature / Application page 12: Users Management Page

- **Description**: This page enables administrators to manage registered users of the application. The layout consists of two main sections: a user list with pagination on the left, and a detailed user information panel on the right. Admins can:
  - View a paginated list of users
  - Select a user to view full profile details
  - Update user information (such as name, email)
  - Persist changes to the backend using API calls
  - View real-time updates reflected in the store



Danh sách người dùng

admin	admin@gmail.com
user	user@gmail.com
Vinh	user1@gmail.com
Gia Vinh	vinh@gmail.com
node	nodemailertestreset@gmail.com

Thông tin người dùng

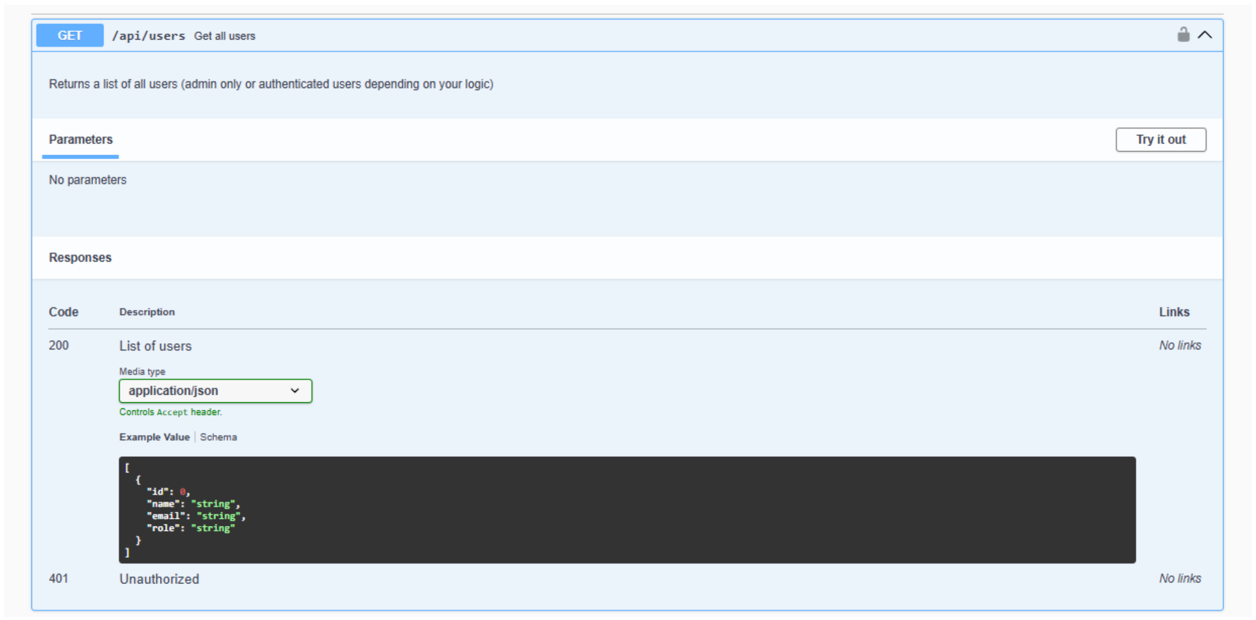
ID:	1
Tên:	admin
E-mail:	admin@gmail.com
Địa chỉ:	123 Ninh Kieu
Điện thoại:	0987654321
Vai trò:	QUẢN TRỊ VIÊN

- **Implementation details:**
  - Libraries used:
    - Frontend:
      - Pinia (state management)
    - Backend:
      - JSend

- ApiError

+ Server-side APIs used:

Endpoint	**Method**	**Purpose**
/api/users?page=1	GET	Retrieve paginated user list
/api/users/:id	PUT	Update user profile data



GET `/api/users` Get all users

Returns a list of all users (admin only or authenticated users depending on your logic)

Parameters

No parameters

Responses

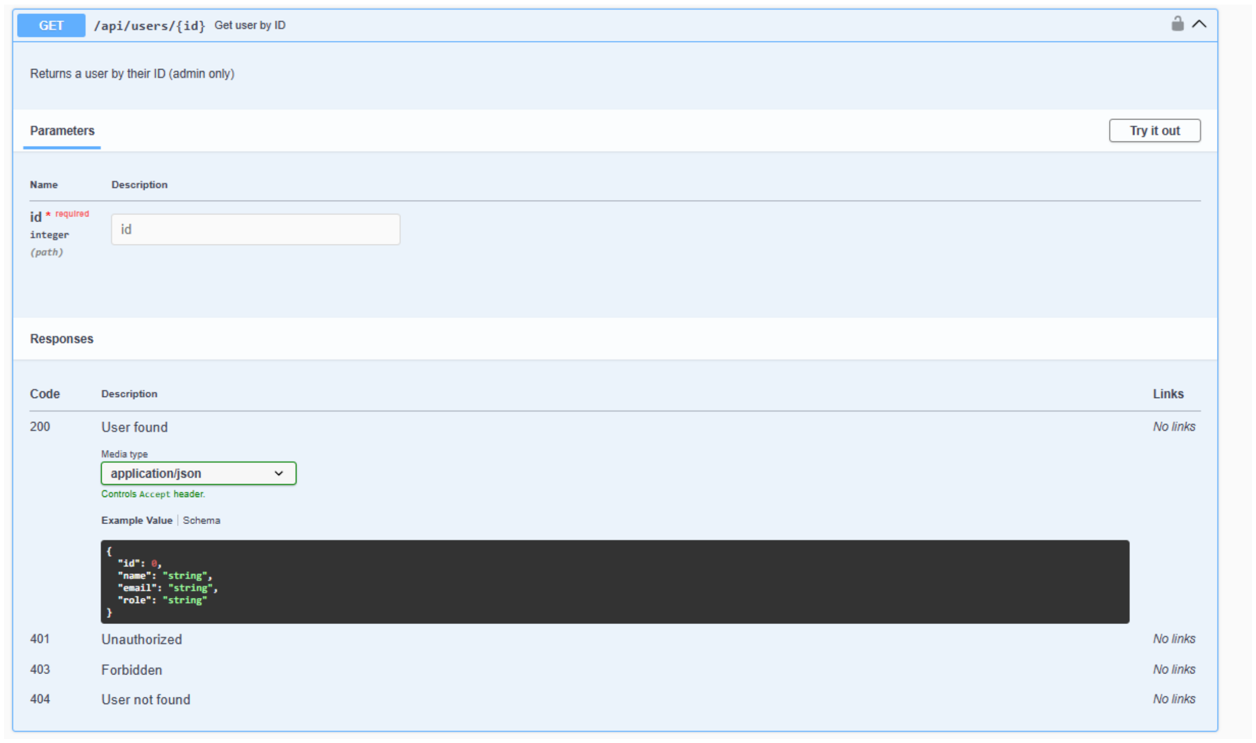
Code	Description	Links
200	List of users	No links
401	Unauthorized	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
[
  {
    "id": 0,
    "name": "string",
    "email": "string",
    "role": "string"
  }
]
```



GET `/api/users/{id}` Get user by ID

Returns a user by their ID (admin only)

Parameters

Name	Description
id * required integer (path)	id

Responses

Code	Description	Links
200	User found	No links
401	Unauthorized	No links
403	Forbidden	No links
404	User not found	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

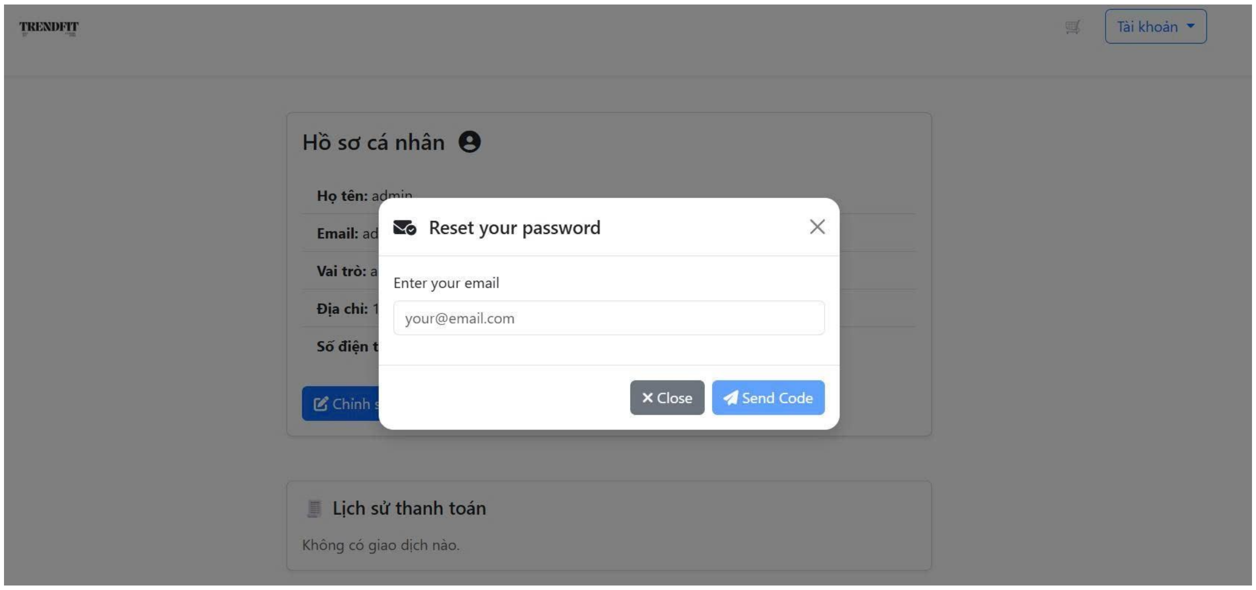
```
{
  "id": 0,
  "name": "string",
  "email": "string",
  "role": "string"
}
```

+ Data read/stored:
    - Tables:
        - users
+ Client-side states needed:

Variable	**Description**
users	Paginated list of user objects
selectedUser	The user currently selected in the list
isLoading	Indicates whether data is being fetched
isError	Flags if an error occurred during API call
errorMessage	Holds the error message to display
currentPage	Current page number (parsed from route.query.page)
totalPages	Total number of user pages (received from backend metadata)

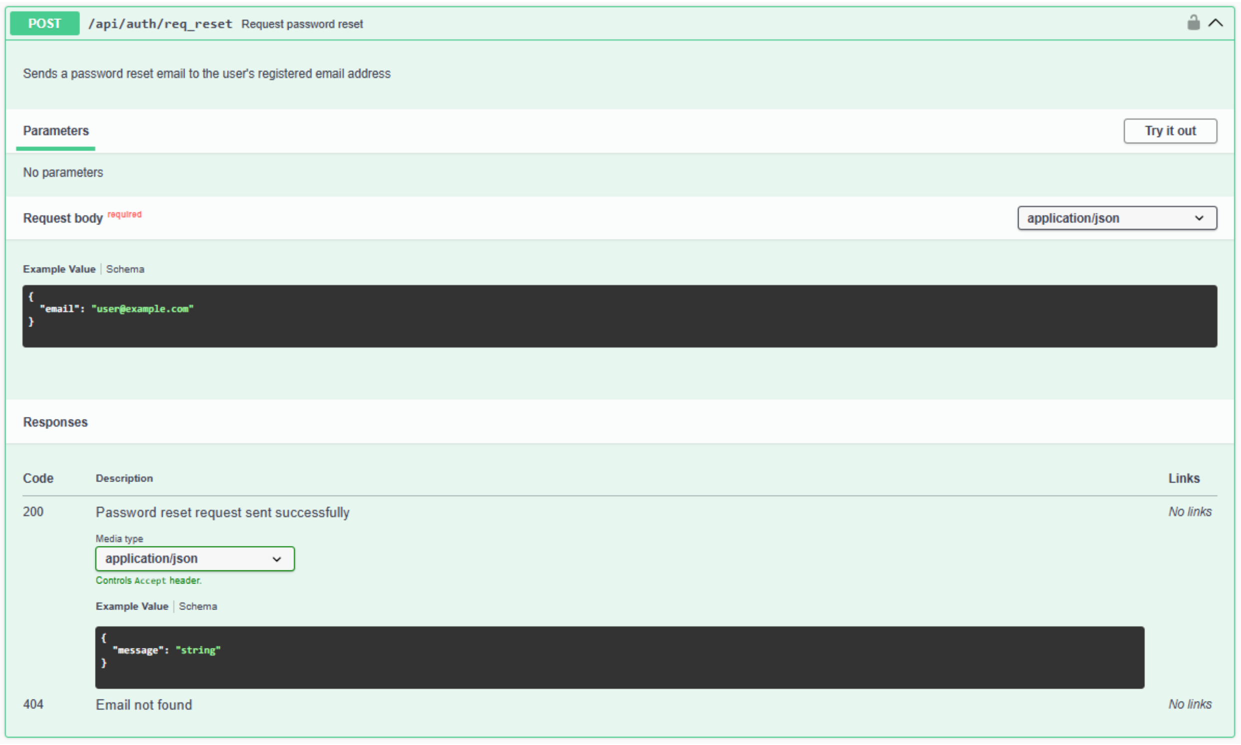
13. **Feature / Application page 13: Password Reset**

- **Description**: A modal popup that allows users to initiate a password reset request by submitting their email address. Once submitted, a reset code is sent to the email. Users can:
    + Input their email address
    + Submit a password reset request
    + View success or error messages dynamically
    + Cancel and close the modal
    + Automatically be redirected to the verification step if needed



- **Implementation details:**
  + Libraries used:
    - Frontend:
      - Pinia (state management)
    - Backend:
      - Nodemailer (send mail)
  + Server-side APIs used:

Endpoint	Method	Purpose
/api/auth/req_reset	POST	Sends a password reset code to email


POST /api/auth/req_reset Request password reset

Sends a password reset email to the user's registered email address

Parameters

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "email": "user@example.com"
}
```

Responses

Code	Description	Links
200	Password reset request sent successfully	No links
404	Email not found	No links

Media type: application/json

Controls Accept header.

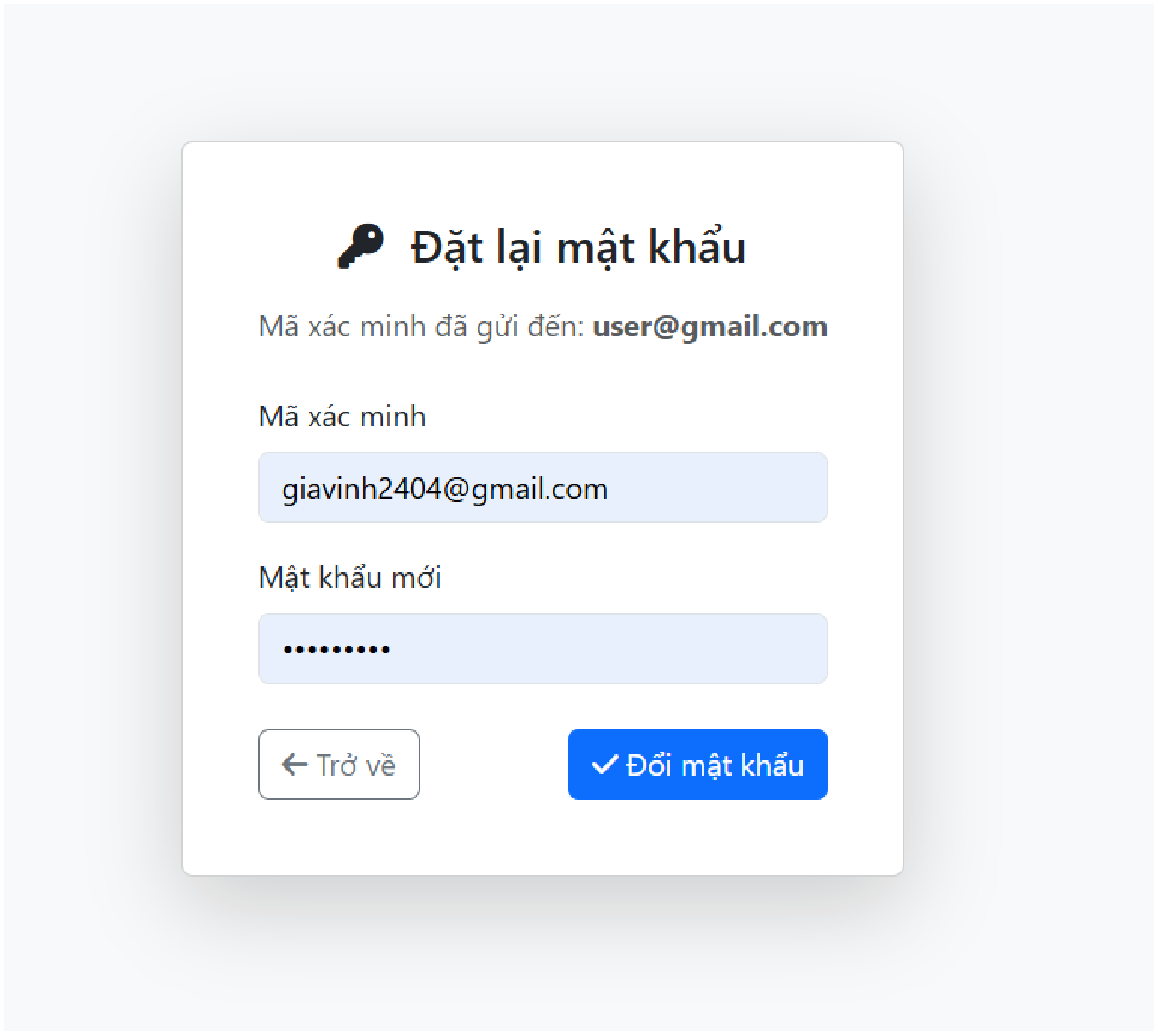
Example Value | Schema

```
{
  "message": "string"
}
```

+ Data read/stored:
  - Tables:
    - Users
    - Reset_password
+ Client-side states needed:

Variable	Description
email	Email address entered by the user
success	Indicates whether the reset email has been sent successfully
loading	Boolean flag showing whether the request is in progress
errorMessage	Message displayed if the reset request fails
show (prop)	Controls modal visibility from parent component
resetStore.step	Current password reset step (used to route user after closing modal)

14. Feature / Application Page 14: Password Reset Confirmation:


Đặt lại mật khẩu

Mã xác minh đã gửi đến: **user@gmail.com**

Mã xác minh

giavinh2404@gmail.com

Mật khẩu mới

•••••••••

- **Description:** A confirmation screen or modal that allows users to complete the password reset process after receiving a reset code via email. User can:
    - Enter their email address
    - Enter the reset code
    - Enter a new password
    - Submit a password reset request
    - View dynamic success or error messages
- **Implementation Details:**
    - Frontend: use ‘ref’
    - Backend: brcypt (for password hasing), Jsend (for consistent response format), ApiError (for standardized error handling)
    - Client-side State Variables:

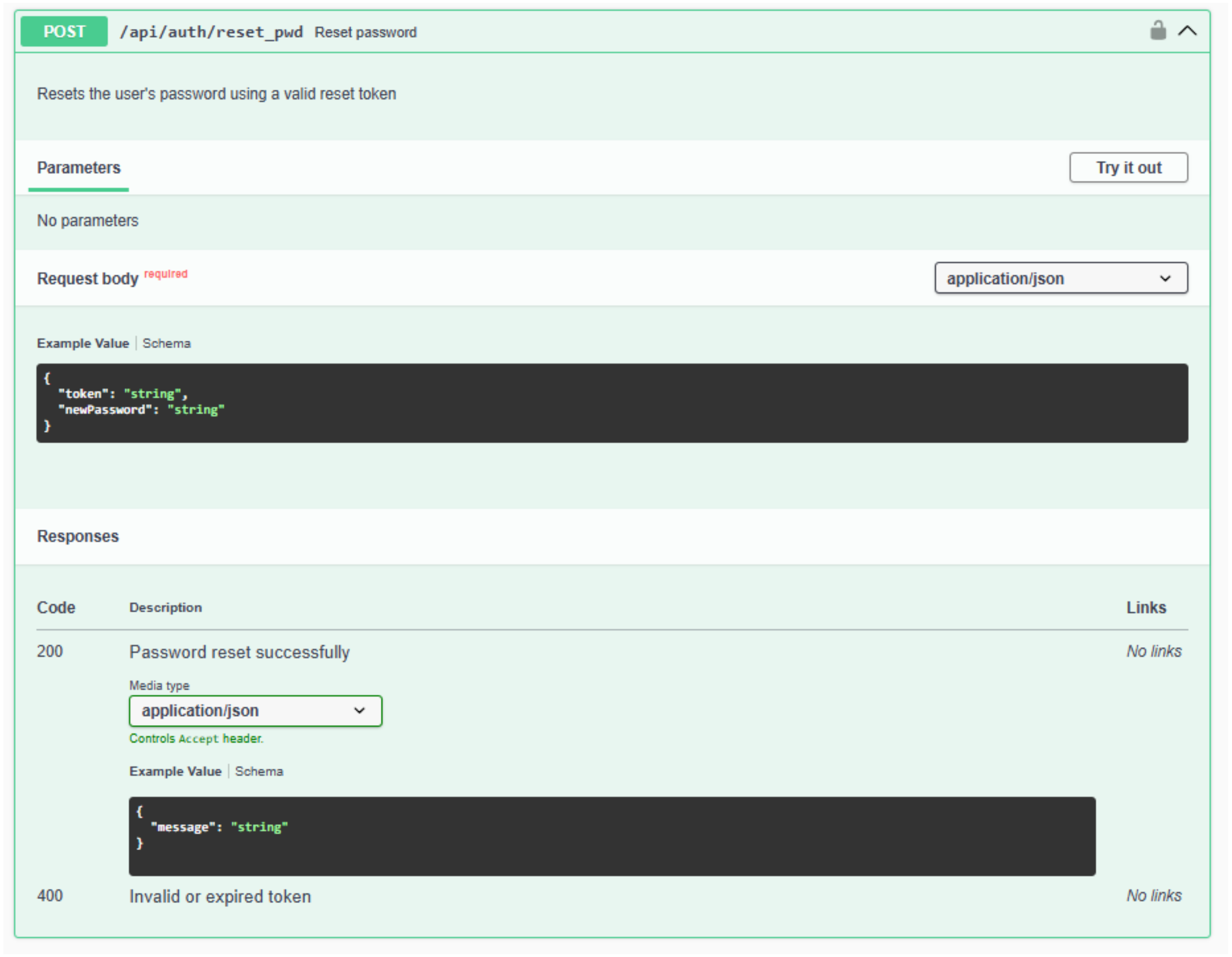
Variable	**Description**

email	Email address entered by the user
code	Reset code sent to the user's email
newPassword	New password to be set
success	Boolean indicating if password reset was successful
loading	Indicates whether the request is in progress
errorMessage	Displays error messages if the request fails
resetStore.step	Used to manage routing or step progression after reset is done

- Database Tables: users and password_reset

- **API Endpoint**

Endpoint	Method	Purpose
/api/auth/reset_pwd	POST	Verifies reset code and updates password



POST /api/auth/reset_pwd Reset password

Resets the user's password using a valid reset token

Parameters

Try it out

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "token": "string",
  "newPassword": "string"
}
```

Responses

Code	Description	Links
200	Password reset successfully Media type: application/json Controls `Accept` header. Example Value | Schema `{ "message": "string" }`	*No links*
400	Invalid or expired token	*No links*