ST0503 BackEnd Web Development

CA2 Documentation

KENNETH CHEN

P2100072

DAAA/FT/1B/04

Timeline

Description automatically generated

NOTE:

The purchase table does NOT reference productid from the product table, but only contains it. This is because when an admin deletes a product, it is IMPORTANT that we DO NOT delete the order record of that product. Therefore, in the database structure, we do NOT use any ON DELETE constraint.

Likewise, the orders table does NOT reference any information from address and payment, but contains details for the purpose of an accurate record.

Contents

[Frontend Reference 5](#_Toc95039998)

[**URL** 6](#_Toc95039999)

[**Description** 6](#_Toc95040000)

[/ 6](#_Toc95040001)

[Home page 6](#_Toc95040002)

[/login 6](#_Toc95040003)

[Login page 6](#_Toc95040004)

[/register 6](#_Toc95040005)

[Register page 6](#_Toc95040006)

[/search 6](#_Toc95040007)

[Product search/results page 6](#_Toc95040008)

[/product/:productid 6](#_Toc95040009)

[Product page 6](#_Toc95040010)

[/interests 6](#_Toc95040011)

[Customer category interests page 6](#_Toc95040012)

[/admin 6](#_Toc95040013)

[Admin panel 6](#_Toc95040014)

[/checkout 6](#_Toc95040015)

[Checkout page 6](#_Toc95040016)

[/home/partial/:type 6](#_Toc95040017)

[Partial HTML for home page. Returns different HTML depending on the role of the user 6](#_Toc95040018)

[/order/:orderid 6](#_Toc95040019)

[Successful order page 6](#_Toc95040020)

[/orders 6](#_Toc95040021)

[View Order History 6](#_Toc95040022)

[Backend API Summary 7](#_Toc95040023)

[User Endpoints: [Endpoint 1] – [Endpoint 4] 7](#_Toc95040024)

[Product Endpoints: [Endpoint 5] – [Endpoint 16] 7](#_Toc95040025)

[Interest Endpoints: [Endpoint 17] – [Endpoint 19] 7](#_Toc95040026)

[Category Endpoints: [Endpoint 20] – [Endpoint 21] 7](#_Toc95040027)

[Promotion Endpoints: [Endpoint 22] – [Endpoint 25] 7](#_Toc95040028)

[Discount Endpoints: [Endpoint 26] – [Endpoint 28] 7](#_Toc95040029)

[Payment Endpoints: [Endpoint 29] – [Endpoint 33] 8](#_Toc95040030)

[Address Endpoints: [Endpoint 34] – [Endpoint 38] 8](#_Toc95040031)

[Order Endpoint: [Endpoint 39] – [Endpoint 41] 8](#_Toc95040032)

[Backend API Reference 9](#_Toc95040033)

[**User endpoints** 9](#_Toc95040034)

[GET /api/token 9](#_Toc95040035)

[POST /api/login 9](#_Toc95040036)

[POST /api/register 9](#_Toc95040037)

[GET /api/user 9](#_Toc95040038)

[**Product endpoints** 10](#_Toc95040039)

[GET /api/products 10](#_Toc95040040)

[GET /api/product/:productid/image 10](#_Toc95040041)

[GET /api/image 10](#_Toc95040042)

[GET /api/search **ADVANCED FEATURE** 10](#_Toc95040043)

[GET /api/product/:productid 11](#_Toc95040044)

[GET /api/brands 11](#_Toc95040045)

[GET /api/product/:productid/reviews 11](#_Toc95040046)

[POST /api/product/:productid/reviews 11](#_Toc95040047)

[POST /api/product 12](#_Toc95040048)

[PUT /api/product/:productid **ADVANCED FEATURE** 12](#_Toc95040049)

[DELETE /api/product/:productid **ADVANCED FEATURE** 13](#_Toc95040050)

[GET /api/product/:productid/promotion **ADVANCED FEATURE** 13](#_Toc95040051)

[**Interest Endpoints** 13](#_Toc95040052)

[GET /api/interests 13](#_Toc95040053)

[POST /api/interests 13](#_Toc95040054)

[DELETE /api/interests 13](#_Toc95040055)

[**Category endpoints** 14](#_Toc95040056)

[GET /api/category 14](#_Toc95040057)

[POST /api/category 14](#_Toc95040058)

[**Promotion endpoints: ALL ADVANCED FEATURES** 14](#_Toc95040059)

[POST /api/promotion 14](#_Toc95040060)

[GET /api/promotion 14](#_Toc95040061)

[PUT /api/promotion/:promotionid 15](#_Toc95040062)

[DELETE /api/promotion/:promotionid 15](#_Toc95040063)

[**Discount endpoints: ALL ADVANCED FEATURES** 15](#_Toc95040064)

[GET /api/discount 15](#_Toc95040065)

[DELETE /api/discount 16](#_Toc95040066)

[POST /api/discount 16](#_Toc95040067)

[**Payment endpoints: ALL ADVANCED FEATURES** 17](#_Toc95040068)

[GET /api/payments 17](#_Toc95040069)

[GET /api/payment/:paymentid 17](#_Toc95040070)

[POST /api/payment 17](#_Toc95040071)

[PUT /api/payment/:paymentid 17](#_Toc95040072)

[DELETE /api/payment/:paymentid 18](#_Toc95040073)

[**Address endpoints: ALL ADVANCED FEATURES** 18](#_Toc95040074)

[GET /api/addresses 18](#_Toc95040075)

[GET /api/address/:addressid 18](#_Toc95040076)

[POST /api/address 18](#_Toc95040077)

[PUT /api/address/:addressid 19](#_Toc95040078)

[DELETE /api/address:/:addressid 19](#_Toc95040079)

[**Order endpoints: ALL ADVANCED FEATURES** 19](#_Toc95040080)

[POST /api/order 19](#_Toc95040081)

[GET /api/orders 20](#_Toc95040082)

[GET /api/order/:orderid 20](#_Toc95040083)

# Frontend Reference

Default Port: 3001

|  |  |
| --- | --- |
| **URL** | **Description** |
| / | Home page |
| /login | Login page |
| /register | Register page |
| /search | Product search/results page |
| /product/:productid | Product page |
| /interests | Customer category interests page |
| /admin | Admin panel |
| /checkout | Checkout page |
| /home/partial/:type | Partial HTML for home page, used for account dropdown. Returns different HTML depending on the role of the user |
| /order/:orderid | Successful order page |
| /orders | View Order History |

# Backend API Summary

This is just a summary. See “Backend API Reference” below for more details on individual endpoints

**Basic Requirements**

### User Endpoints: [Endpoint 1] – [Endpoint 4]

* Token information, login, register and user information

### Product Endpoints: [Endpoint 5] – [Endpoint 16]

**ADVANCED FEATURES are in BOLD**

* Add, view**, edit**, **delete,** and search by name, search by brand, **search by categor**y, **search by** **promotion**.
* **Add, view**, **edit product image**, **view image from backend**.
* View brands.
* Add and view reviews.

### Interest Endpoints: [Endpoint 17] – [Endpoint 19]

* Add and view interests

### Category Endpoints: [Endpoint 20] – [Endpoint 21]

* Add and view categories

**ADVANCED FEATURES**

**ALL advanced features in the assignment brief were done.**

### Promotion Endpoints: [Endpoint 22] – [Endpoint 25]

* Add, view, edit and delete promotions

### Discount Endpoints: [Endpoint 26] – [Endpoint 28]

Explanation: Discounts link together promotions and products. A promotion can have many products, and a product can also have many promotions. This link is done in a many-to-many relationship.

* View, add and delete discounts.

### Payment Endpoints: [Endpoint 29] – [Endpoint 33]

* View, add, edit, and delete payment methods

### Address Endpoints: [Endpoint 34] – [Endpoint 38]

* View, add, edit, and delete addresses

### Order Endpoint: [Endpoint 39] – [Endpoint 41]

* Add and view order
* View order history

# Backend API Reference

DEFAULT PORT: 8081

The endpoint number contained in square brackets is identical to the endpoint number in the comments of /backend/controller/app.js. You can easily find the exact endpoint by CTRL-F and searching the endpoint number.

## **User endpoints**

[Endpoint 1]

### GET /api/token

Authorized: Public, Customer, Admin

This endpoint decodes the token passed by Authorization Header and returns the role of that token. If the token is empty, “Public” role is returned instead.

[Endpoint 2]

### POST /api/login

Authorized: Public

Given login information, we return the respective token if the credentials are correct.

[Endpoint 3]

### POST /api/register

Authorized: Public

This endpoint receives information of

<username>, <email>, <contact>, <password>, <profile\_pic\_url>

through the form body and 'registers' the user with these details.

[Endpoint 4]

### GET /api/user

Authorized: Customer, Admin

This endpoint returns information of the currently logged in user. The middleware verifyTokenMiddleware allows us to access the userid through req.userid, which we use to query the user's information. This prevents users from snooping into other users' details.

## **Product endpoints**

[Endpoint 5]

### GET /api/products

Authorized: Public, Customer, Admin

This endpoint returns the information

<productid>, <name>, <description>, <categoryid>, <brand>, <price>

of all products

[Endpoint 6]

### GET /api/product/:productid/image

Authorized: Public, Customer, Admin

Return a product’s image src given the product id.

The src attribute contains the image’s data encoded in base64. E.g. this endpoint would return  
data:<mimetype>;base64,<base64\_data>. Images are displayed to the client using its base64 data.

[Endpoint 7]

### GET /api/image

Authorized: Public, Customer, Admin

Return an image’s src that is stored in backend, given the name of the image.

The src attribute contains the image’s data encoded in base64. E.g. this endpoint would return data:<mimetype>;base64,<base64\_data>. Images are displayed to the client using its base64 data.

[Endpoint 8]

### GET /api/search **ADVANCED FEATURE**

Authorized: Public, Customer, Admin

This endpoint searches a product given the queries

<query>

- The search term that searches across product name, product brand and product description

<categoryids>

- A string that contains comma separated categoryids <- ADVANCED EFATURE

<promotionids>

- A string that contains comma separated promotionids <- ADVANCED FEATURE

<brands>

- A string that contains comma separated brand names

[Endpoint 9]

### GET /api/product/:productid

Authorized: Public, Customer, Admin

Given a productid, this endpoint returns even more detailed information of that product. To see exactly what information is returned, the specific SQL query can be found in `/model/product.js`

[Endpoint 10]

### GET /api/brands

Authorized: Public, Customer, Admin

Return all brand names.

[Endpoint 11]

### GET /api/product/:productid/reviews

Authorized: Public, Customer, Admin

Return all reviews information of a product. To see exactly what information is returned, the specific SQL query can be found in `/model/product.js`.

[Endpoint 12]

### POST /api/product/:productid/reviews

Authorized: Customer

This endpoint posts a review on a product, by taking in the following:

> parameter

<productid>

- The id of that product

> body

<rating>

- The number value of rating. Decimals are allowed

<review\_content>

- The text in the review itself

[Endpoint 13]

### POST /api/product

Authorized: Admin

This endpoint adds a new product, taking in:

> body

<product\_name>

- name of product

<product\_price>

- raw price of product

<product\_category>

- category of product

<product\_brand>

- brand of product

<product\_desc>

- description of product

<image>

- image of the product

This endpoint uses multer to handle the form data.

[Endpoint 14: ADVANCED FEATURE]

### PUT /api/product/:productid **ADVANCED FEATURE**

Authorized: Admin

This endpoint edits a product given the same information as Endpoint 13.

[Endpoint 15: ADVANCED FEATURE]

### DELETE /api/product/:productid **ADVANCED FEATURE**

Authorized: Admin

Given a productid, delete the image and database records of that productid if it exists. NOTE this endpoint is omnipotent

[Endpoint 16: ADVANCED FEATURE]

### GET /api/product/:productid/promotion **ADVANCED FEATURE**

Authorized: Public, Customer, Admin

Get all promotions of a product, given the productid in parameters.

## **Interest Endpoints**

[Endpoint 17]

### GET /api/interests

Authorized: Customer

This endpoint returns the categories(interests) a customer is interested in

[Endpoint 18]

### POST /api/interests

Authorized: Customer

This endpoint takes in a comma separated string of categoryids, and inserts them into the user's interests. A 422 error is thrown when a categoryid is already inserted.

[Endpoint 19]

### DELETE /api/interests

Authorized: Customer

Given a comma-separated string of categoryids, delete the respective categoryids from the user's interest if they exist. Note this is omnipotent.

## **Category endpoints**

[Endpoint 20]

### GET /api/category

Authorized: Public, Customer, Admin

This endpoint returns the categoryid, category name and description of all categories

[Endpoint 21]

### POST /api/category

Authorized: Admin

This endpoint allows admins to add a new category.

## **Promotion endpoints: ALL ADVANCED FEATURES**

[Endpoint 22: ADVANCED FEATURE]

### POST /api/promotion

Authorized: Admin

Create a promotion given

<promotion\_name>

- Name of promotion

<promotion\_start>

- Format: YYYY/MM/DD

<promotion\_end>

- Format: YYYY/MM/DD

[Endpoint 23: ADVANCED FEATURE]

### GET /api/promotion

Authorized: Public, Customer, Admin

This endpoint returns the information

<promotionid>, <promotionname>, <start>, <end >, <created\_at >

of all existing promotions

[Endpoint 24: ADVANCED FEATURE]

### PUT /api/promotion/:promotionid

Authorized: Admin

Update a promotion, given

> Parameters

<promotionid>

- Id of promotion

> Body

<promotion\_name>

- Name of promotion

<promotion\_start>

- Format: YYYY/MM/DD

<promotion\_end>

- Format: YYYY/MM/DD

[Endpoint 25: ADVANCED FEATURE]

### DELETE /api/promotion/:promotionid

Authorized: Admin

Given promotionid, delete that promotion if it exists.

NOTE this endpoint is intended to be omnipotent

## **Discount endpoints: ALL ADVANCED FEATURES**

[Endpoint 26: ADVANCED FEATURE]

### GET /api/discount

Authorized: Public, Customer, Admin

Return discount information

<product\_discount>

* Discount applied when promotion is active

<productid>

* Product id

<name>

* Product name

<promotionid>

* Id of promotion that activates the product\_discount

<promotionname>

* Name of promotion

[Endpoint 27: ADVANCED FEATURE]

### DELETE /api/discount

Authorized: Admin

Given the productid and promotionid through the form body, we delete the discount that links the two together. NOTE this endpoint is omnipotent. This can be done as both productid and promotionid combined MUST be a unique key, according to the database structure.

[Endpoint 28: ADVANCED FEATURE]

### POST /api/discount

Authorized: Admin

Create a discount, given:

<productid>

- Id of product

<promotionid>

- Promotion of product

<discount\_value>

- Percent of discount, taking values between 0-100.

For instance, 25 here would indicate that product

receives 25% off when the promotion goes live.

## **Payment endpoints: ALL ADVANCED FEATURES**

[Endpoint 29: ADVANCED FEATURE]

### GET /api/payments

Authorized: Customer

This endpoint returns information on all payment methods of the currently logged-in user.

[Endpoint 30: ADVANCED FEATURE]

### GET /api/payment/:paymentid

Authorized: Customer, Admin

Given a paymentid, return detailed information of that payment id. An error is thrown if a customer tries to view another user’s payment details. Admin can see any user’s payment detail.

[Endpoint 31: ADVANCED FEATURE]

### POST /api/payment

Authorized: Customer

Add a payment method given:

<card\_number>

- Credit/Debit card number

<cvv>

- Security number

<card\_name>

- The name on the card

<card\_expiration>

- Expiry date of card. Format: YYYY/MM

[Endpoint 32: ADVANCED FEATURE]

### PUT /api/payment/:paymentid

Authorized: Customer

Update a payment method given

> Parameter

<paymentid>

- Id of payment method

> Body

<card\_number>

- Credit/Debit card number

<cvv>

- Security number

<card\_name>

- The name on the card

<card\_expiration>

- Expiry date of card. Format: YYYY/MM

[Endpoint 33: ADVANCED FEATURE]

### DELETE /api/payment/:paymentid

Authorized: Customer

Given paymentid, delete the payment method if it exists. NOTE this endpoint is omnipotent. An error is thrown when a user tries to delete a payment method that is not theirs.

## **Address endpoints: ALL ADVANCED FEATURES**

[Endpoint 34: ADVANCED FEATURE]

### GET /api/addresses

Authorized: Customer

Get basic information of all addresses a user has

[Endpoint 35: ADVANCED FEATURE]

### GET /api/address/:addressid

Authorized: Customer, Admin

This endpoint returns detailed information of an address given the addressid. An error is thrown when a customer tries to view another user’s address details. Admins can view any user’s address details.

[Endpoint 36: ADVANCED FEATURE]

### POST /api/address

Authorized: Customer

Create an address for the logged-in customer, given

<address\_name>

- Name of the receiver/person who lives at the address

<address>

- Address detail itself. No format.

<postal\_code>

- A six-digit number

[Endpoint 37: ADVANCED FEATURE]

### PUT /api/address/:addressid

Authorized: Customer

Update an address given information similar to [Endpoint 36]

[Endpoint 38: ADVANCED FEATURE]

### DELETE /api/address:/:addressid

Authorized: Customer

Given addressid, delete the address associated with it if it exists. NOTE: this endpoint is omnipotent. An error is thrown when a customer tries to delete another user’s address.

## **Order endpoints: ALL ADVANCED FEATURES**

[Endpoint 39: ADVANCED FEATURE]

### POST /api/order

Authorized: Customer

Create an order given:

<paymentid>

- Id of the payment method

<addressid>

- Id of the address

<cart>

- String containing cart information

<total>

- The total amount paid for the items

[Endpoint 40: ADVANCED FEATURE]

### GET /api/orders

Authorized: Customer

Return information of user’s order history

[Endpoint 41: ADVANCED FEATURE]

### GET /api/order/:orderid

Authorized: Public, Customer, Admin

Get the list of items bought, from an order. This does not contain any sensitive information, and thus all roles, Public, Customer and Admin have the permission to view this.