API Feature requirements

1. Users can view all items when entering the website
    - GET /products

2. Items are displayed properly based on the selected department and category
    - /products/inCategory/{category_id}
    - /products/inDepartment/{department_id}

3. Users can search items through search box
    - /products/search

4. Support paging if we have too many items
    - All products APIs supporting pages

5. Users can see item details by selecting a specific item
    - /products/{product_id}

6. Users can add items to their shopping carts
    - /shoppingcart/add

7. Users can register/login using website custom forms, or social login libraries
    - /customers/login
    - /customers

8. Users can update personal profiles with shipping addresses and other info
    - /customer
    - /customers/address
    - /customers/creditCard

9. Users can checkout with Stripe payment gateways. This requirement is **mandatory**, you must use the related APIs for the payment.
    - /stripe/charge

10. Users will get confirmations over emails about their orders
    - /orders

11. Clear unused shopping cart frequently.
    - /shoppingcart/empty/{cart_id}

API Details

POST /customer/login

Input validation for customer login

- email => required and email
- password => required

Cases:

- If required filed will not pass, then it will return this error “USR_02 - The field(s) are/is required.” Field is replaced with name.
- If wrong email passed, then this value will through “USR_03 – The email is invalid.”
- If email or password in incorrect then this error will through “USR_01 - Email or Password is invalid.”

Output:

Return required object along token and expiry time.

POST /customer

Input validation for register customer

- email => required and email
- name => required
- password => required

Cases:

- If required filed will not pass, then it will return this error “USR_02 - The field(s) are/is required.” Field is replaced with name.
- If wrong email passed, then this value will through “USR_03 – The email is invalid.”
- If email found, then this error will through “USR_04 – The email already exists.”

Output:

Return required object along token and expiry time.

PUT /customer

Input validation for profile update

- name => required and string
- email => required and email
- day_phone => phoneNumber
- eve_phone => phoneNumber
- mob_phone => phoneNumber
- USER-KEY => token

Cases:

- If token is empty, then this value will through “AUT_01 - Authorization code is empty.”
- If token is wrong, then this value will through “AUT_02 - Access Unauthorized.”
- If required filed not pass, then it will return this error “USR_02 - The field is required.” Filed replaced with name.
- If wrong email passed, then this error will return “USR_03 - The Email is invalid.”.
- If wrong phone passed, then this error will return “USR_06 - this is an invalid phone number.”

Output:

Return object after update.

PUT /customer/address

Input validation for address update

- address_1 => required
- city => required
- region => required
- country => required
- shipping_region_id => required
- USER-KEY => token

Cases:

- If token is empty, then this value will through “AUT_01 - Authorization code is empty.”
- If token is wrong, then this value will through “AUT_02 - Access Unauthorized.”
- If required filed not pass, then it will return this error “USR_02 - The field is required.” Filed replaced with name.
- If wrong shipping_region_id passed, then this error will return “USR_09 - The Shipping Region ID is not number.”

Output:

Return object after update.

PUT /customer/creditCard

Following validation implemented for update customer credit card

- credit_card => required and creditCard
- USER-KEY => token

Cases:

- If token is empty, then this value will through “AUT_01 - Authorization code is empty.”
- If token is wrong, then this value will through “AUT_02 - Access Unauthorized.”
- If credit_card will not pass, then it will return this error “USR_02 - The credit card is required.”
- If wrong credit card passed, then this value will through “USR_08 - this is an invalid Credit Card.”

Output:

Return object after update.

GET /products

Input validation for all products

- page => integer
- limit => integer
- description_length => string

Cases:

- By default, limit is 20 from page 1 without any word truncate.
- If page number will not pass, then records will show from first page else as per page number. If input is not number, then error will show.
- If limit will be empty, then 20 records will show else records will show as per limit. If input is not number, then error will show.
- If description length will pass, then description text will trim as per that else full description will show. If input is not number, then error will show.

Output:

Return products total and list of rows.

GET /products/search

Input validation for product search

- query_string => required
- page => integer
- limit => integer
- description_length => string

Cases:

- if query_string is empty then it will through error “PRO_01 - The query string is required.”
- By default, limit is 20 from page 1 without any word truncate.
- If page number will not pass, then records will show from first page else as per page number. If input is not number, then error will show.
- If limit will be empty, then 20 records will show else records will show as per limit. If input is not number, then error will show.
- If description length will pass, then description text will trim as per that else full description will show. If input is not number, then error will show.

Output:

Return products total and list of rows.

GET /products/details/{product_id}

Input validation for this product detail

- product_id => required and integer

Cases:

- If product_id will not pass, then it will return this error “PRO_01 - The product ID is required.”
- If wrong product_id passed, then this error will return “PRO_02 - The product ID is not number.”

Output:

Return a product object.

GET /products/inCategory/{category_id}

Input validation for product by category id

- category_id => required and integer
- page => integer
- limit => integer
- description_length => string

Cases:

- If category_id will not pass, then it will return this error “CAT_02 - The category ID is required.”
- If wrong category_id passed, then this error will return “CAT_03 - The category ID is not number.”
- By default, limit is 20 from page 1 without any word truncate.
- If page number will not pass, then records will show from first page else as per page number. If input is not number, then error will show.
- If limit will be empty, then 20 records will show else records will show as per limit. If input is not number, then error will show.
- If description length will pass, then description text will trim as per that else full description will show. If input is not number, then error will show.

Output:

Return total and a list of products.

GET /products/inDepartment/{department_id}

Input validation for product by department id

- department_id => required and integer
- page => integer
- limit => integer
- description_length => string

Cases:

- If department_id will not pass, then it will return this error “DEP_01 - The department ID is required.”
- If wrong department_id passed, then this error will return “DEP_03 - The department ID is not number.”
- By default, limit is 20 from page 1 without any word truncate.
- If page number will not pass, then records will show from first page else as per page number. If input is not number, then error will show.
- If limit will be empty, then 20 records will show else records will show as per limit. If input is not number, then error will show.
- If description length will pass, then description text will trim as per that else full description will show. If input is not number, then error will show.

Output: Return total and product list.

POST /orders

Input validation for create order and confirmation email

- cart_id => required and integer
- shipping_id => required and integer
- tax_id => required and integer
- USER-KEY => token

Cases:

- If token is empty, then this value will through “AUT_01 - Authorization code is empty.”
- If token is wrong, then this value will through “AUT_02 - Access Unauthorized.”
- If required filed not pass, then it will show error “OR_01 - The field is required.” Filed replaced with name.
- If wrong cart_id/shipping_id/tax_id passed, then this error will return “OR_02 - The filed is not number.” Filed replaced with name.
- If cart_id not found, then this error will return “OR_03 - Cart ID doesn't exist”.
- Send confirmation email to customer against order.

Output:

Return order id.

POST /stripe/charge

Input validation for stripe charge

- stripeToken => required //tok_visa
- order_id => required and integer // 1
- description => required // any text
- amount => required and integer // minimum 100

Cases:

- If required filed not pass, then it will show error “STR_01 - The field is required.” Filed replaced with name.
- If wrong cart_id/amount passed, then this error will return “STR_02 - The filed is not number.”. Filed replaced with name.
- Request for stripe checkout return error if anything goes wrong.

Output:

Return stripe object.

POST /shoppingcart/add

Input validation for create a shopping cart

- cart_id => required and integer
- product_id => required and integer
- attributes => required

Cases:

- If cart_id not pass, then it will show error “SC_01 - The cart ID is required.”
- If required filed not pass, then it will show error “PRO_01 - The product ID is required.”.
- If wrong cart_id passed, then this error will return “SC_02 - The cart ID is not number.”
- If wrong product_id passed, then this error will return “PRO_02 - The product ID is not number.”
- If record doesn’t add, then SQL error will show.

Output:

Return an array of products in the cart.

DELETE /shoppingcart/empty/{cart_id}

Input validation for clear shopping cart

- cart_id => required and integer

Cases:

- If cart_id will not pass, then it will show error “SC_01 - The cart ID is required.”.
- If wrong cart_id passed, then this error will return “SC_02 - The cart ID is not number.”
- If record doesn’t add, then SQL error will show.

Output:

Return empty.