

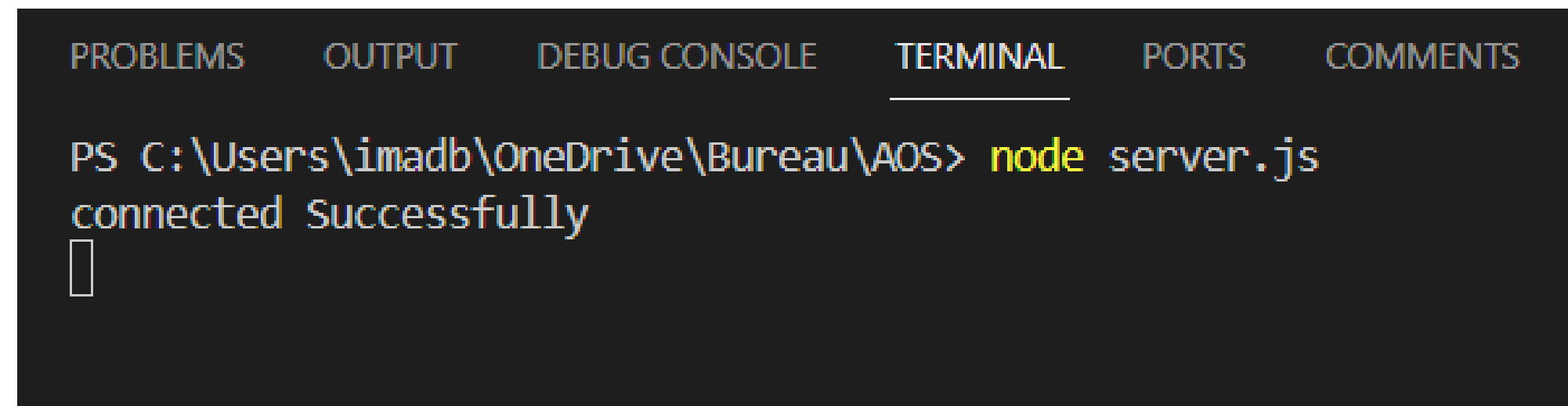
Aos

API Documentation

1 - To run on the Server :

Open the project in Vs code

Run the command : **node server.js**



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS  
PS C:\Users\imadb\OneDrive\Bureau\AOS> node server.js  
connected Successfully  
█
```

Try it out

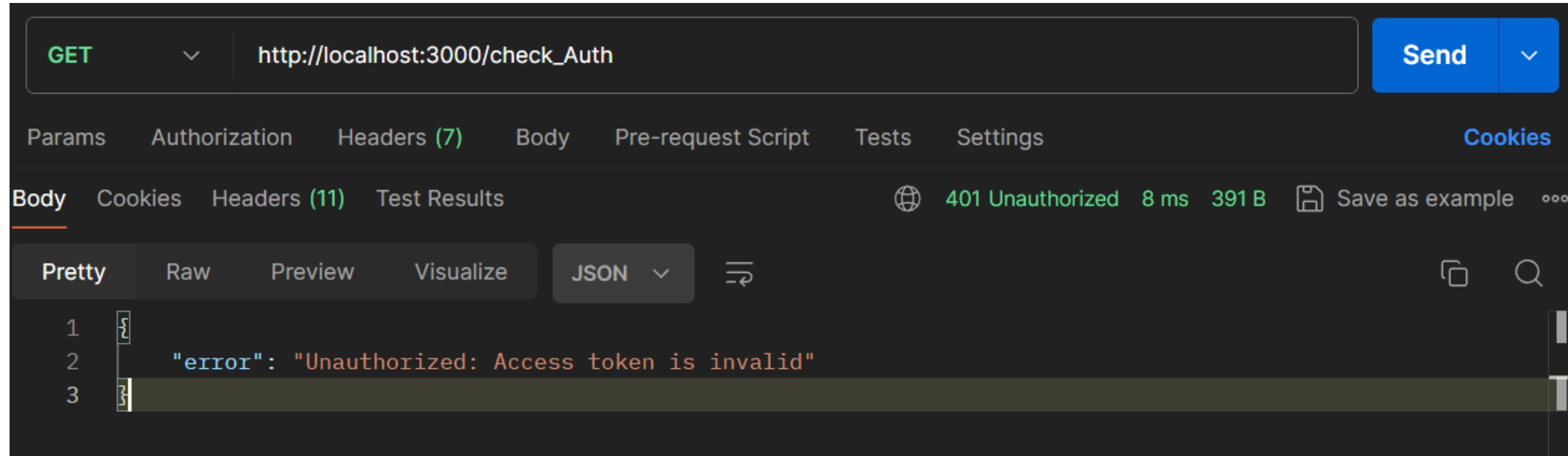
Navigate to :

<http://localhost:3000>



Fist thing First

We should Authenticate ower user
when the Application (Front end and mobile)
first been run Send Request to the Endpoint
GET `http://localhost:3000/check_Auth`



200 => Authorized

401=> unAuthorized

Store the authentication
status to work with it

Register

URL : POST **http://localhost:3000/Register**

Body :

- FirstName (String, required): User's first name.
- LastName (String, required): User's last name.
- Email (String, required): User's email address.
- Password (String, required): User's password.
- Age (Number, optional): User's age.
- Gender (String, required): User's gender (accepted values: 'male' or 'female').
- Telephone (String, required): User's telephone number.
- Address (String, optional): User's address.

Responses :

- 200 OK : Successful registration.
- 400 Bad Request : An error occurred during registration.
- 409 Conflict : Missing or invalid data provided.
- 500 internal Server Error
- 429 Too many Requests (to secure the website from hackers)

After Succesfull Register an email will be snded automatically to the user
Redirect the User to the verification page

VerifyAccount use this endpoint to verify the user email

URL : POST **http://localhost:3000/VerifyAccount**

Responses

- 200 OK: Account verified successfully.
- 400 Bad Request: Missing data or invalid verification code.
- 404 Not Found: Verification token or user not found.
- 409 Conflict: Verification token has expired or account is already verified.
- 500 Internal Server Error: An error occurred during account verification.

Show a button to resend the verification code : sometimes the users do not get their verification code

Send Email Verification

URL POST **http://localhost:3000/ReSend_Verification_Email**

Body :

- `userId`: The ID of the user to whom the verification email will be sent.

Responses :

- 200 OK: The verification email has been successfully sent.
- 401 Unauthorized: The provided token is invalid.
- 404 Not Found: The specified user does not exist.
- 500 Internal Server Error: An error occurred during the process.
- 429 Too many Requests (to secure the website from hackers)

Login

URL : POST **http://localhost:3000/Login**

Body :

- Email (String, required): User's email address.
- Password (String, required): User's password.

Responses :

- 200 OK : Successful login. **user Data will be send as a json response store it to with it latter**
- 401 Incorrect username or pwd
- 400 Bad Request
- 409 Messing or invalid Data
- 500 internal server error
- 429 Too many Requests (to secure the website from hackers)

Logout

URL : POST **http://localhost:3000/Logout**

Responses

- 204 No Content: Successfully logged out.
- 500 Internal Server Error: An error occurred during logout.
- 429 Too many Requests (to secure the website from hackers)

After Successfull Logout :

Redirect the user to the home page and set the Authentication status to false
Show the Home page to the user as a geust

is_email_verified

We can use This endpoint to figure out if the user verified its account or not

If it is not

We can show a button to redirect it to the verification page

GET **`http://localhost:3000/is_email_verified/:userId`**

Description : This endpoint checks the email verification status of the specified user.

body :

- `userId`: The ID of the user whose email verification status will be checked.

Responses

- 200 OK: The email verification status is successfully retrieved.
- 401 Unauthorized: The provided token is invalid.
- 404 Not Found: The specified user does not exist.
- 500 Internal Server Error: An error occurred during the process.
- 429 Too many Requests (to secure the website from hackers)

if the user email is not verified and he has been redirected to the verification page
use this endpoint to send a verification code to him via email:
use the Verify_account endpoint explained before to verify the user

URL POST **`http://localhost:3000/Send_Verification_Email/:userId`**

Description

This endpoint sends a verification email to the specified user for email verification.

Parameters

- `userId`: The ID of the user to whom the verification email will be sent.

Responses :

- 200 OK: The verification email is successfully sent.
- 401 Unauthorized: The provided token is invalid.
- 404 Not Found: The specified user does not exist.
- 500 Internal Server Error: An error occurred during the process.
- 429 Too many Requests (to secure the website from hackers)

User Profile

Get User Profile **for the Owner of the Account**

- Endpoint: GET **http://localhost:3000/Users/:userId/Profile**
- Description: Retrieves the profile of a user with the specified user ID. This endpoint is accessible only to account owner.
- Parameters:
 - userId
- Response:
 - 200 OK: User profile retrieved successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Retrieve information of a specific user **Public View**

- Request Type: **GET**
- Endpoint: **GET http://localhost:3000/Users/:userId**
- Description: Retrieve information of a specific user.
- Parameters:
 - userId: ID of the user.
- Response: User object containing user information.
- Status Code:
 - 200 OK: Successful retrieval.
 - 404 Not Found: User not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Update profile information of a specific user

- Request Type: **PUT**
- Endpoint: **PUT http://localhost:3000/Users/:userId**
- Parameters:
 - userId: ID of the user.
- Body: (Optional) the target fields to be updated
 - FirstName
 - LastName
 - Age
 - Gender
 - Telephone
- Access: Requires authentication.
- Response: Success message upon profile update.
- Status Code:
 - 200 OK: Profile updated successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Delete profile of a specific user

- Request Type: **DELETE**
- Endpoint: **DELETE** **http://localhost:3000/Users/:userId**
- Description: Delete profile of a specific user.
- Parameters:
 - userId: ID of the user.
- Access: Requires authentication.
- Response: Success message upon profile deletion.
- Status Code:
 - 200 OK: Profile deleted successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Products

GET All Products

- Endpoint: GET **http://localhost:3000/Products**
- Description: Retrieves all products with pagination support.
- Parameters: None
- Query Parameters:
 - page (optional): Page number for pagination (default is 1).
 - limit (optional): Number of products to retrieve per page (default is 20).
- Authorization: Public.
- Request Body: None
- Response:
 - 200 OK: Returns an array of products along with pagination details (total pages).
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

GET Product by ID

- Endpoint: GET **http://localhost:3000/Products/:productId**
- Description: Retrieves a product by its ID.
- Parameters:
 - productId: ID of the product.
- Authorization: Public.
- Request Body: None
- Response:
 - 200 OK: Returns the product details.
 - 404 Not Found: Product not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

GET Search Product

- Endpoint: GET **http://localhost:3000/Products/search/:search**
- Description: Searches for products matching a search query with pagination support.
- Parameters:
 - search: Search query.
- Query Parameters:
 - page (optional): Page number for pagination (default is 1).
 - limit (optional): Number of products to retrieve per page (default is 20).
- Authorization: Public.
- Request Body: None
- Response:
 - 200 OK: Returns an array of products matching the search query along with pagination details (total pages).
 - 404 Not Found: No products found matching the search query.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

GET All Categories

- Endpoint: GET **http://localhost:3000/Products/categories**
- Description: Retrieves all product categories.
- Parameters: None
- Authorization: Public.
- Request Body: None
- Response:
 - 200 OK: Returns an array of product categories.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

GET Products by Category

- Endpoint: GET **http://localhost:3000/Products/categories/:category**
- Description: Retrieves products belonging to a specific category with pagination support.
- Parameters:
 - category: Name of the category.
- Query Parameters:
 - page (optional): Page number for pagination (default is 1).
 - limit (optional): Number of products to retrieve per page (default is 20).
- Authorization: Public.
- Request Body: None
- Response:
 - 200 OK: Returns an array of products belonging to the specified category along with pagination details (total pages).
 - 404 Not Found: Could not find products with that category.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Filter Products

- Endpoint: GET **http://localhost:3000/Products/filter**
- Description: Filters products based on specified criteria with pagination support.
- Query Parameters:
 - category (optional): Name of the category to filter by.
 - price (optional): Maximum price of products to filter by.
 - rate (optional): Minimum rating of products to filter by.
 - sortBy (optional): Sorting criteria for the filtered products (mostViewed, mostRated, mostBought).
 - location (optional): Location of the products.
 - page (optional): Page number for pagination (default is 1).
 - limit (optional): Number of products to retrieve per page (default is 20).
- Authorization: Public.
- Request Body: None
- Response:
 - 200 OK: Returns an array of filtered products along with pagination details (total pages).
 - 400 Bad Request: Invalid sortBy parameter.
 - 404 Not Found: Could not find products matching the specified criteria.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Request Example : GET http://localhost:3000/products/filter?

category=electronics&price=500&rate=4&sortBy=mostViewed&page=1&limit=10

Rate Store

Rate Product

- Endpoint: POST **http://localhost:3000/Users/:userId/RateProduct/:productId**
- Description: Allows a user to rate a product.
- Parameters:
 - userId: ID of the user who is rating the product.
 - productId: ID of the product being rated.
- Request Body:
 - rate: The rating given by the user (must be a number between 1 and 5).
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product rated successfully.
 - 400 Bad Request: User already rated this product.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data or invalid rate format.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Delete Rate Product

- Endpoint: DELETE **http://localhost:3000/Users/:userId/RateProduct/:productId**
- Description: Allows a user to delete their rating for a product.
- Parameters:
 - userId: ID of the user who rated the product.
 - productId: ID of the product.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product rating deleted successfully.
 - 400 Bad Request: User didn't rate this product.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Edit Rate Product

- Endpoint: PUT **http://localhost:3000/Users/:userId/RateProduct/:productId**
- Description: Allows a user to edit their rating for a product.
- Parameters:
 - userId: ID of the user who rated the product.
 - productId: ID of the product.
- Request Body:
 - rate: The new rating given by the user (must be a number between 1 and 5).
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product rating edited successfully.
 - 400 Bad Request: User didn't rate this product or invalid rate format.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Get Product User Rate

- Endpoint: GET **http://localhost:3000/Users/:userId/RateProduct/:productId**
- Description: Retrieves the rating given by a user for a specific product.
- Parameters:
 - `userId`: ID of the user who rated the product.
 - `productId`: ID of the product.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Returns the rating given by the user for the product.
 - 400 Bad Request: User didn't rate this product.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Comment to a product

Comment Product

- Endpoint: POST **http://localhost:3000/Users/:userId/CommentProduct/:productId**
- Description: Allows a user to comment on a product.
- Parameters:
 - userId: ID of the user who is commenting on the product.
 - productId: ID of the product being commented on.
- Request Body:
 - Comment: The comment provided by the user.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product commented successfully.
 - 400 Bad Request: User already commented on this product.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Delete Comment Product

- Endpoint: DELETE **http://localhost:3000/Users/:userId/CommentProduct/:productId**
- Description: Allows a user to delete their comment on a product.
- Parameters:
 - userId: ID of the user who commented on the product.
 - productId: ID of the product.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product comment deleted successfully.
 - 400 Bad Request: User didn't comment on this product.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Get Product User Comment

- Endpoint: GET **http://localhost:3000/Users/:userId/CommentProduct/:productId**
- Description: Retrieves comments made by a user on a specific product.
- Parameters:
 - userId: ID of the user who commented on the product.
 - productId: ID of the product.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Returns the comments made by the user on the product.
 - 400 Bad Request: User didn't comment on this product.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Edit Comment

- Endpoint: PUT **http://localhost:3000/Users/:userId/CommentProduct/:productId**
- Description: Allows a user to edit their comment on a product.
- Parameters:
 - userId: ID of the user who commented on the product.
 - productId: ID of the product.
- Request Body:
 - Comment: The updated comment provided by the user.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product comment edited successfully.
 - 400 Bad Request: User didn't comment on this product or unauthorized edit attempt.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Interste To product

Add to Interested Products

- Endpoint: POST **http://localhost:3000/Users/:userId/Interested/:productId**
- Description: Allows a user to add a product to their list of interested products.
- Parameters:
 - userId: ID of the user who wants to add the product to their list of interested products.
 - productId: ID of the product being added to the user's list of interested products.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product added to interested list successfully.
 - 400 Bad Request: Product already in the user's interested list.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Delete from Interested Products

- Endpoint: DELETE **http://localhost:3000/Users/:userId/Interested/:productId**
- Description: Allows a user to remove a product from their list of interested products.
- Parameters:
 - userId: ID of the user who wants to remove the product from their list of interested products.
 - productId: ID of the product being removed from the user's list of interested products.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product removed from interested list successfully.
 - 400 Bad Request: Product not found in the user's interested list.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Add to Not Interested Products

- Endpoint: POST **http://localhost:3000/Users/:userId/NotInterested/:productId**
- Description: Allows a user to add a product to their list of not interested products.
- Parameters:
 - userId: ID of the user who wants to add the product to their list of not interested products.
 - productId: ID of the product being added to the user's list of not interested products.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product added to not interested list successfully.
 - 400 Bad Request: Product already in the user's not interested list.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Delete from Not Interested Products

- Endpoint: DELETE **http://localhost:3000/Users/:userId/NotInterested/:productId**
- Description: Allows a user to remove a product from their list of not interested products.
- Parameters:
 - userId: ID of the user who wants to remove the product from their list of not interested products.
 - productId: ID of the product being removed from the user's list of not interested products.
- Authorization: Required (user must be authenticated).
- Response:
 - 200 OK: Product removed from not interested list successfully.
 - 400 Bad Request: Product not found in the user's not interested list.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Basket

Add a product to the basket of a specific user

- Request Type: POST
- Endpoint: POST **http://localhost:3000/Users/:userId/basket/:productId**
- Description: Add a product to the basket of a specific user.
- Parameters:
 - userId: ID of the user.
 - productId: ID of the product to add to the basket.
- Access: Requires authentication.
- Response: Success message upon adding the product to the basket.
- Status Code:
 - 200 OK: Product added to basket successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Remove a product from the basket of a specific user

- Request Type: **DELETE**
- Endpoint: **DELETE** **`http://localhost:3000/Users/:userId/basket/:productId`**
- Description: Remove a product from the basket of a specific user.
- Parameters:
 - `userId`: ID of the user.
 - `productId`: ID of the product to remove from the basket.
- Access: Requires authentication.
- Response: Success message upon removing the product from the basket.
- Status Code:
 - 200 OK: Product deleted from basket successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Retrieve the basket of a specific user

- Request Type: **GET**
- Endpoint: **GET http://localhost:3000/Users/:userId/basket**
- Description: Retrieve the basket of a specific user.
- Parameters:
 - userId: ID of the user.
- Access: Requires authentication.
- Response: Array of products in the user's basket.
- Status Code:
 - 200 OK: Successful retrieval.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Favorite

Add a product to favorites of a specific user

- Request Type: POST
- Endpoint: POST **http://localhost:3000/Users/:userId/favorite/:productId**
- Description: Add a product to favorites of a specific user.
- Parameters:
 - userId: ID of the user.
 - productId: ID of the product to add to favorites.
- Access: Requires authentication.
- Response: Success message upon adding the product to favorites.
- Status Code:
 - 200 OK: Product added to favorites successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Remove a product from favorites of a specific user

- Request Type: **DELETE**
- Endpoint: **DELETE** **http://localhost:3000/Users/:userId/favorite/:productId**
- Description: Remove a product from favorites of a specific user.
- Parameters:
 - userId: ID of the user.
 - productId: ID of the product to remove from favorites.
- Access: Requires authentication.
- Response: Success message upon removing the product from favorites.
- Status Code:
 - 200 OK: Product deleted from favorites successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or product not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Retrieve the favorites of a specific user

- Request Type: GET
- Endpoint: GET **http://localhost:3000/Users/:userId/favorite**
- Description: Retrieve the favorites of a specific user.
- Parameters:
 - userId: ID of the user.
- Access: Requires authentication.
- Response: Array of products in the user's favorites.
- Status Code:
 - 200 OK: Successful retrieval.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Stores

Create a store for a specific user

- Request Type: POST
- Endpoint: POST **http://localhost:3000/Users/:userId/createStore**
- Description: Create a store for a specific user.
- Parameters:
 - userId: ID of the user.
- Body:
 - Email: Email of the store owner.
 - Password: Password for the store.
 - StoreName: Name of the store.
 - Store_Description: Description of the store.
 - Telephone: Telephone number of the store owner.
- Access: Requires authentication.
- Response: Success message upon store creation.
- Status Code:
 - 200 OK: Store created successfully.
 - 401 Unauthorized: Invalid token.
 - 409 Conflict: Missing data or invalid input.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Retrieve All Stores

Retrieve all stores with pagination support.

- Request Type: **GET**
- Endpoint: **GET http://localhost:3000/Stores/**
- Description: Retrieve all stores with pagination support.

Parameters:

- page (optional): Page number for pagination. Defaults to 1 if not provided.
- limit (optional): Number of stores per page. Defaults to 20 if not provided.

Access: Public.

Response:

- 200 OK: Successful retrieval. Returns an object containing totalPages and an array of stores.
- 500 Internal Server Error: An error occurred on the server side.
- 429 Too many Requests (to secure the website from hackers)

Example : GET http://localhost:3000/Stores/?page=1&limit=10

Get Store Details

- Endpoint: GET **http://localhost:3000/Stores/:storeId**
- Description: Retrieves details of a specific store.
- Parameters:
 - storeId: ID of the store.
- Authorization: No authentication required.
- Request Body: None
- Response:
 - 200 OK: Returns the details of the store.
 - 404 Not Found: Store not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)
- Additional Functionality:
 - If the request contains a userId in the request body, the function checks if the user has already visited the store. If not, it records the visit and increments the store's visit count.
 - You should Send the userId in the body if it is authenticated

Follow a store for a specific user

- Request Type: POST
- Endpoint: POST **http://localhost:3000/Users/:userId/Follow/:storeId**
- Description: Follow a store for a specific user.
- Parameters:
 - userId: ID of the user.
 - storeId: ID of the store to follow.
- Access: Requires user authentication.
- Response: Success message upon following the store.
- Status Code:
 - 200 OK: Store followed successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or store not found.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Get Store Products

- Endpoint: GET **http://localhost:3000/Stores/:storeId/Products**
- Description: Retrieves products belonging to a specific store.
- Parameters:
 - storeId: ID of the store.
- Query Parameters:
 - page (optional): Page number for pagination (default is 1).
 - limit (optional): Number of products to retrieve per page (default is 20).
- Authorization: Public.
- Request Body: None
- Response:
 - 200 OK: Returns an array of products belonging to the store along with pagination details (total pages).
 - 404 Not Found: Store not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Request Example :GET http://localhost:3000/Stores/123456789/Products?page=1&limit=10

Get Store Followers

- Endpoint: GET **http://localhost:3000/Stores/:storeId/Followers**
- Description: Retrieves followers of a specific store.
- Parameters:
 - storeId: ID of the store.
- Query Parameters:
 - page (optional): Page number for pagination (default is 1).
 - limit (optional): Number of followers to retrieve per page (default is 20).
- Authorization: Public.
- Request Body: None.
- Response:
 - 200 OK: Returns an array of followers belonging to the store along with pagination details (total pages).
 - 404 Not Found: Store not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Example Request : GET http://localhost:3000/Stores/12345/Followers?page=1&limit=10

Rate Store

Rate a Store

- Endpoint: POST **http://localhost:3000/Users/:userId/RateStore/:storeId**
- Description: Allows a user to rate a store.
- Request Body:
 - rate: The rating given to the store (a number between 1 and 5).
- Parameters:
 - userId: ID of the user.
 - storeId: ID of the store being rated.
- Authorization: Requires authentication.
- Response:
 - 200 OK: Store rated successfully.
 - 400 Bad Request: User already rated this store.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or store not found.
 - 409 Conflict: Rate must be a number between 1 and 5.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Delete Store Rating

- Endpoint: DELETE **http://localhost:3000/Users/:userId/RateStore/:storeId**
- Description: Allows a user to delete their rating for a store.
- Parameters:
 - userId: ID of the user.
 - storeId: ID of the store.
- Authorization: Requires authentication.
- Response:
 - 200 OK: Store rating deleted successfully.
 - 400 Bad Request: User didn't rate this store.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or store not found.
 - 500 internal Server Error
 - 429 Too many Requests (to secure the website from hackers)

Edit Store Rating

- Endpoint: PUT **http://localhost:3000/Users/:userId/RateStore/:storeId**
- Description: Allows a user to edit their rating for a store.
- Request Body:
 - rate: The new rating given to the store (a number between 1 and 5).
- Parameters:
 - userId: ID of the user.
 - storeId: ID of the store.
- Authorization: Requires authentication.
- Response:
 - 200 OK: Store rating edited successfully.
 - 400 Bad Request: User didn't rate this store or user's rate is missing.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or store not found.
 - 409 Conflict: Rate must be a number between 1 and 5.
 - 500 internal Server Error
 - 429 Too many Requests (to secure the website from hackers)

Get User's Rating for a Store

- Endpoint: GET **http://localhost:3000/Users/:userId/RateStore/:storeId**
- Description: Retrieves the user's rating for a specific store.
- Parameters:
 - userId: ID of the user.
 - storeId: ID of the store.
- Authorization: Requires authentication.
- Response:
 - 200 OK: Returns the user's rating for the store.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: User or store not found.
 - 500 internal Server Error
 - 429 Too many Requests (to secure the website from hackers)

Store Owner

Get Store Profile Only for the Owner of the Store

- Endpoint: GET **http://localhost:3000/Stores/Profile/:storeId**
- Description: Retrieves the profile information of the store with the specified ID.
- Parameters:
 - storeId: ID of the store for which the profile information is requested.
- Authorization: Required (only accessible to Store Owner).
- Response:
 - 200 OK: Profile information of the store retrieved successfully.
 - 401 Unauthorized: Invalid token or unauthorized access.
 - 404 Not Found: Store not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Create Product

- Endpoint: POST **http://localhost:3000/Stores/:storeId/Products/Create**
- Description: Creates a new product under the specified store.
- Parameters:
 - storeId: ID of the store under which the product will be created.
- Authorization: Required (only accessible to S).
- Request Body:
 - Title: Title of the product.
 - Description: Description of the product.
 - Category: Category of the product.
 - Price: Price of the product.
- Response:
 - 200 OK: Product created successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: Store not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Delete Store

- Endpoint: DELETE **http://localhost:3000/Stores/:storeId**
- Description: Deletes the store with the specified ID.
- Parameters:
 - storeId: ID of the store to be deleted.
- Authorization: Required (only accessible to Store Owner).
- Response:
 - 200 OK: Store deleted successfully.
 - 401 Unauthorized: Invalid token.
 - 404 Not Found: Store not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Delete Product

- Endpoint: DELETE **http://localhost:3000/Stores/:storeId/Products/:productId**
- Description: Deletes the product with the specified ID under the specified store.
- Parameters:
 - storeId: ID of the store containing the product.
 - productId: ID of the product to be deleted.
- Authorization: Required (only accessible to Store Owner).
- Response:
 - 200 OK: Product deleted successfully.
 - 401 Unauthorized: Invalid token or unauthorized.
 - 404 Not Found: Store or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Edit Store

- Endpoint: PUT **http://localhost:3000/Stores/:storeId**
- Description: Updates the details of the store with the specified ID.
- Parameters:
 - storeId: ID of the store to be updated.
- Authorization: Required (only accessible to administrators).
- Request Body (Optional) Fields to be updated:
 - StoreName: New name for the store.
 - Store_Description: New description for the store.
 - Telephone: New telephone number for the store.
- Response:
 - 200 OK: Store updated successfully.
 - 401 Unauthorized: Invalid token or unauthorized.
 - 404 Not Found: Store not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)

Edit Product

- Endpoint: PUT **http://localhost:3000/Stores/:storeId/Products/:productId**
- Description: Updates the details of the product with the specified ID under the specified store.
- Parameters:
 - storeId: ID of the store containing the product.
 - productId: ID of the product to be updated.
- Authorization: Required (only accessible to administrators).
- Request Body (Optional) Fields to be updated:
 - Title: New title for the product.
 - Description: New description for the product.
 - Price: New price for the product.
- Response:
 - 200 OK: Product updated successfully.
 - 401 Unauthorized: Invalid token or unauthorized.
 - 404 Not Found: Store or product not found.
 - 409 Conflict: Missing data.
 - 500 Internal Server Error: Server encountered an unexpected condition.
 - 429 Too many Requests (to secure the website from hackers)