API Testing Report

API Endpoint: Homepage

Endpoint Details

- **URL:** http://localhost:5000
- **Method:** GET
- **Description:** This endpoint is used to verify that the backend server is running. It returns a success message indicating that the backend is operational.

Request Parameters

- **None**

Test Cases

1. Successful Response

- **Description:** Verify that the homepage is accessible and returns a success message.
- **Input:** None
- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** ShelfWave backend home.

Test Results

Test Case: Successful Response

Status: Pass

Response Code: 200

Response Body: ShelfWave backend home.

Remarks: The backend is operational

API Endpoint: Get User Details

Endpoint Details

- **URL:** `http://localhost:5000/api/user/getUser`
- **Method:** GET
- **Description:** This endpoint returns the details of the logged-in user. If the user is not logged in, it returns an error message indicating that the user is not authorized. The authentication token is passed as a cookie.

Request Parameters

- **None**

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Retrieval of User Details

- **Description:** Verify that the API returns user details for a logged-in user with a valid authentication token.
- **Input:**
    - **Headers:**
        - Cookie: `token=<valid_token>`
- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** JSON object containing user details

2. User Not Logged In

- **Description:** Verify that the API returns an error when the user is not logged in or the authentication token is missing/invalid.
- **Input:**
    - **Headers:**
        - Cookie: `token=<invalid_or_missing_token>`
- **Expected Response:**
    - **Status Code:** 401 Unauthorized
    - **Body:** JSON object containing an error message

Test Results

Test Case	Status	Response Code	Response Body	Remarks

Successful Retrieval of User Details	Pass	200	{
			"_id": "66a064e6eecd00e7f32a19c9",
			"name": "Patel Arjun",
			"email": "patel689@uwindsor.ca",
			"photo": "link to photo",
			"phone": "1234567890",
			"bio": "Hi I am Arjun"
			}
				User details successfully retrieved

User Not Logged In	Pass	401	{ "message": "Not authorized, please login" }	Appropriate error message shown

API Endpoint: Check Login Status

Endpoint Details

- **URL:** `http://localhost:5000/api/user/loginstatus`
- **Method:** GET
- **Description:** This endpoint checks if the user is logged in. It returns **`true`** if the user is logged in and **`false`** otherwise. The authentication token is passed as a cookie.

Request Parameters

- **None**

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. User is Logged In

- **Description:** Verify that the API returns **`true`** when the user is logged in with a valid authentication token.
- **Input:**
  - **Headers:**
    - Cookie: `token=<valid_token>`
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body: `true`**

2. User is Not Logged In

- **Description:** Verify that the API returns **`false`** when the user is not logged in or the authentication token is missing/invalid.
- **Input:**
  - **Headers:**
    - Cookie: **`token=<invalid_or_missing_token>`**
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body: `false`**

Test Results

Test Case	Status	Response Code	Response Body	Remarks
User is Logged In	Pass	200	true	Correctly indicates user is logged in
User is Not Logged In	Pass	200	false	Correctly indicates user is not logged in

API Endpoint: User Logout

Endpoint Details

- **URL:** `http://localhost:5000/api/user/logout`
- **Method:** GET
- **Description:** This endpoint logs out the user by deleting the authentication token cookie and returns a message indicating that the user logged out successfully.

Request Parameters

- **None**

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Logout

- **Description:** Verify that the API logs out the user and deletes the authentication token cookie.
- **Input:**
    - **Headers:**
        - Cookie: `token=<valid_token>`
- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** JSON object containing a success message

2. Logout Without Token

- **Description:** Verify that the API returns an appropriate message when there is no authentication token.
- **Input:**
    - **Headers:**
        - Cookie: `token=<missing_or_invalid_token>`
- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** JSON object containing a message

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Logout	Pass	200	{ "message": "Successfully Logged Out" }	User successfully logged out
Logout Without Token	Pass	200	{ "message": "Successfully Logged Out" }	Proper message displayed

API Endpoint: User Registration

Endpoint Details

- **URL:** `http://localhost:5000/api/user/register`
- **Method:** POST
- **Description:** This endpoint registers a new user. It takes `email`, `password`, and `name` as form data and returns a success message upon successful registration.

Request Parameters

Parameter	Type	Required	Description
email	string	Yes	User's email address
password	string	Yes	User's password
name	string	Yes	User's name

Test Cases

1. Successful Registration

- **Description:** Verify that a user can register with valid data.
- **Input:**
  - email: valid@example.com
  - password: validpassword
  - name: Valid Name
- **Expected Response:**
  - **Status Code:** 201 Created
  - **Body:** JSON object containing a success message

2. Missing Email

- **Description:** Verify that the API returns an error when the email is missing.
- **Input:**
  - password: validpassword
  - name: Valid Name
- **Expected Response:**
  - **Status Code:** 400 Bad Request
  - **Body:** JSON object containing an error message about the missing email

3. Missing Password

- **Description:** Verify that the API returns an error when the password is missing.
- **Input:**
    - email: valid@example.com
    - name: Valid Name
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing password

4. Missing Name

- **Description:** Verify that the API returns an error when the name is missing.
- **Input:**
    - email: valid@example.com
    - password: validpassword
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing name

5. Invalid Email Format

- **Description:** Verify that the API returns an error when the email format is invalid.
- **Input:**
    - email: invalidemail
    - password: validpassword
    - name: Valid Name
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the invalid email format

H

You are an API tester that tests API using Postman I want you to write report for few endpoints hence generate a nice template for the same including the test cases lets say for example http://localhost:5000/api/user/login takes 2 parameters from form email and password both are required
ChatGPT

Certainly! Here's a structured template for an API testing report using Postman. This template includes sections for the endpoint details, test cases, and test results.

API Testing Report

API Endpoint: User Login

Endpoint Details

- **URL:** `http://localhost:5000/api/user/login`
- **Method:** POST
- **Description:** This endpoint is used for user authentication. It takes `email` and `password` as form data and returns a bearer token upon successful login.

Request Parameters

Parameter	Type	Required	Description
email	string	Yes	User's email address
password	string	Yes	User's password

Test Cases

1. Successful Login

- **Description:** Verify that a user can log in with valid credentials.
- **Input:**
    - email: valid@example.com
    - password: validpassword
- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** JSON object containing `token`

2. Missing Email

- **Description:** Verify that the API returns an error when the email is missing.
- **Input:**

    - password: validpassword
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing email

3. Missing Password

- **Description:** Verify that the API returns an error when the password is missing.
- **Input:**
    - email: valid@example.com
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing password

4. Invalid Email Format

- **Description:** Verify that the API returns an error when the email format is invalid.
- **Input:**
    - email: invalidemail
    - password: validpassword
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the invalid email format

5. Invalid Credentials

- **Description:** Verify that the API returns an error when the credentials are invalid.
- **Input:**
    - email: invalid@example.com
    - password: invalidpassword
- **Expected Response:**
    - **Status Code:** 401 Unauthorized
    - **Body:** JSON object containing an error message about the invalid credentials

Test Results

Test Case	Status	Response Code	Response Body	Remarks

Successful Login	Pass	200	{ "token": "..." }	User successfully logged in
Missing Email	Pass	400	{ "error": "Email required" }	Appropriate error message shown
Missing Password	Pass	400	{ "error": "Password required" }	Appropriate error message shown
Invalid Email Format	Pass	400	{ "error": "Invalid email format" }	Appropriate error message shown
Invalid Credentials	Pass	401	{ "error": "Invalid credentials" }	Appropriate error message shown

Notes

- Ensure that the server is running and accessible at `http://localhost:5000`.
- Use valid credentials for the successful login test.
- Validate response headers and content-type for each response.
- Test results should be documented with screenshots where necessary.

This template can be used as a starting point and modified as per the specific requirements of your project or organization.

4o
Okay lets start with first request its a GET request to homepage which is http://localhost:5000/ no params or body it returns a success message saying that backend is running
ChatGPT
Memory updated

Great! Here's the detailed report for the GET request to the homepage.

API Testing Report

API Endpoint: Homepage

Endpoint Details

- **URL:** `http://localhost:5000/`
- **Method:** GET
- **Description:** This endpoint is used to verify that the backend server is running. It returns a success message indicating that the backend is operational.

Request Parameters

- **None**

Test Cases

1. Successful Response

- **Description:** Verify that the homepage is accessible and returns a success message.
- **Input:** None
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body:** JSON object containing a success message (e.g., `{ "message": "Backend is running" }`)

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Response	Pass	200	{ "message": "Backend is running" }	Backend is operational

Notes

- Ensure that the server is running and accessible at `http://localhost:5000`.
- Validate response headers and content-type for the response.
- Test results should be documented with screenshots where necessary.

Next, you can perform the test in Postman and document the results accordingly. Let me know if you need any further assistance with this or other endpoints!

4o
Next endpoint is GET User, it returns details of the logged in user and if the user is not logged in then it returns an error saying not authorized and authentication token is passed as a cookie for logged in user
ChatGPT
Memory updated

Here's the detailed report for the GET request to retrieve user details.

API Testing Report

API Endpoint: Get User Details

Endpoint Details

- **URL:** `http://localhost:5000/api/user`
- **Method:** GET
- **Description:** This endpoint returns the details of the logged-in user. If the user is not logged in, it returns an error message indicating that the user is not authorized. The authentication token is passed as a cookie.

Request Parameters

- **None**

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Retrieval of User Details

- **Description:** Verify that the API returns user details for a logged-in user with a valid authentication token.
- **Input:**
    - **Headers:**
        - Cookie: `token=<valid_token>`

- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** JSON object containing user details (e.g., `{ "id": 1, "name": "John Doe", "email": "johndoe@example.com" }`)

2. User Not Logged In

- **Description:** Verify that the API returns an error when the user is not logged in or the authentication token is missing/invalid.
- **Input:**
    - **Headers:**
        - Cookie: `token=<invalid_or_missing_token>`
- **Expected Response:**
    - **Status Code:** 401 Unauthorized
    - **Body:** JSON object containing an error message (e.g., `{ "error": "Not authorized" }`)

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Retrieval of User Details	Pass	200	{ "id": 1, "name": "John Doe", "email": "johndoe@example.com" }	User details successfully retrieved
User Not Logged In	Pass	401	{ "error": "Not authorized" }	Appropriate error message shown

Notes

- Ensure that the server is running and accessible at `http://localhost:5000`.
- Use a valid authentication token for the successful retrieval test.
- Validate response headers and content-type for each response.
- Test results should be documented with screenshots where necessary.

Next, you can perform the test in Postman and document the results accordingly. Let me know if you need any further assistance with this or other endpoints!

4o
Next endpoint is http://localhost:5000/api/user/loginstatus which returns true or false depending on the user has logged in or not it also passes a cookie with authentication token
ChatGPT
Memory updated

Here's the detailed report for the GET request to check the login status.

API Testing Report

API Endpoint: Check Login Status

Endpoint Details

- **URL:** `http://localhost:5000/api/user/loginstatus`
- **Method:** GET
- **Description:** This endpoint checks if the user is logged in. It returns `true` if the user is logged in and `false` otherwise. The authentication token is passed as a cookie.

Request Parameters

- **None**

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. User is Logged In

- **Description:** Verify that the API returns `true` when the user is logged in with a valid authentication token.
- **Input:**
  - **Headers:**
    - Cookie: `token=<valid_token>`

- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** `true`

2. User is Not Logged In

- **Description:** Verify that the API returns `false` when the user is not logged in or the authentication token is missing/invalid.
- **Input:**
    - **Headers:**
        - Cookie: `token=<invalid_or_missing_token>`
- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** `false`

Test Results

Test Case	Status	Response Code	Response Body	Remarks
User is Logged In	Pass	200	true	Correctly indicates user is logged in
User is Not Logged In	Pass	200	false	Correctly indicates user is not logged in

Notes

- Ensure that the server is running and accessible at `http://localhost:5000`.
- Use a valid authentication token for the "User is Logged In" test.
- Validate response headers and content-type for each response.
- Test results should be documented with screenshots where necessary.

Next, you can perform the test in Postman and document the results accordingly. Let me know if you need any further assistance with this or other endpoints!

4o

Next endpoint is http://localhost:5000/api/user/logout it deletes the cookie with authentication token of a logged in user and returns a message saying that the user logged out successfully

ChatGPT
Memory updated

Here's the detailed report for the GET request to log out.

API Testing Report

API Endpoint: User Logout

Endpoint Details

- **URL:** `http://localhost:5000/api/user/logout`
- **Method:** GET
- **Description:** This endpoint logs out the user by deleting the authentication token cookie and returns a message indicating that the user logged out successfully.

Request Parameters

- **None**

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Logout

- **Description:** Verify that the API logs out the user and deletes the authentication token cookie.
- **Input:**
  - **Headers:**
    - Cookie: `token=<valid_token>`
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body:** JSON object containing a success message (e.g., `{ "message": "User logged out successfully" }`)

2. Logout Without Token

- **Description:** Verify that the API returns an appropriate message when there is no authentication token.
- **Input:**
  - **Headers:**
    - Cookie: `token=<missing_or_invalid_token>`
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body:** JSON object containing a message (e.g., `{ "message": "User logged out successfully" }`)

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Logout	Pass	200	{ "message": "User logged out successfully" }	User successfully logged out
Logout Without Token	Pass	200	{ "message": "User logged out successfully" }	Proper message displayed

Notes

- Ensure that the server is running and accessible at `http://localhost:5000`.
- Use a valid authentication token for the successful logout test.
- Validate response headers and content-type for each response.
- Test results should be documented with screenshots where necessary.

Next, you can perform the test in Postman and document the results accordingly. Let me know if you need any further assistance with this or other endpoints!

4o
Next endpoint is http://localhost:5000/api/user/register The request is POST It takes 3 parameters from body using form email password and name all 3 fields are compulsory
ChatGPT
Memory updated

Here's the detailed report for the POST request to register a new user.

API Testing Report

API Endpoint: User Registration

Endpoint Details

- **URL:** `http://localhost:5000/api/user/register`
- **Method:** POST
- **Description:** This endpoint registers a new user. It takes `email`, `password`, and `name` as form data and returns a success message upon successful registration.

Request Parameters

Parameter	Type	Required	Description
email	string	Yes	User's email address
password	string	Yes	User's password
name	string	Yes	User's name

Test Cases

1. Successful Registration

- **Description:** Verify that a user can register with valid data.
- **Input:**
    - email: valid@example.com
    - password: validpassword
    - name: Valid Name
- **Expected Response:**
    - **Status Code:** 201 Created
    - **Body:** JSON object containing a success message (e.g., `{ "message": "User registered successfully" }`)

2. Missing Email

- **Description:** Verify that the API returns an error when the email is missing.
- **Input:**
    - password: validpassword
    - name: Valid Name
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing email

3. Missing Password

- **Description:** Verify that the API returns an error when the password is missing.
- **Input:**
    - email: valid@example.com
    - name: Valid Name
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing password

4. Missing Name

- **Description:** Verify that the API returns an error when the name is missing.
- **Input:**
    - email: valid@example.com
    - password: validpassword
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing name

5. Invalid Email Format

- **Description:** Verify that the API returns an error when the email format is invalid.
- **Input:**
    - email: invalidemail
    - password: validpassword
    - name: Valid Name
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the invalid email format

6. Already Registered User

- **Description:** Verify that the API returns an error when the registered user tries to register again
- **Input:**
    - email: valid@example.com
    - password: validpassword
    - name: Valid Name
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about email being existing one and suggesting to sign in

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Registration	Pass	201	{ "message": "User registered successfully" }	User successfully registered
Missing Email	Pass	400	{ "error": "Email is required" }	Appropriate error message shown
Missing Password	Pass	400	{ "error": "Password is required" }	Appropriate error message shown
Missing Name	Pass	400	{ "error": "Name is required" }	Appropriate error message shown
Invalid Email Format	Pass	400	{ "error": "Invalid email format" }	Appropriate error message shown
Already registered user	Pass	400	{“error”: “Email exists please sign in” }	Appropriate error message shown

API Endpoint: User Login

Endpoint Details

- **URL:** `http://localhost:5000/api/user/login`
- **Method:** POST
- **Description:** This endpoint is used for user authentication. It takes `email` and `password` as form data and returns a bearer token upon successful login.

Request Parameters

Parameter	Type	Required	Description
email	string	Yes	User's email address
password	string	Yes	User's password

Test Cases

1. Successful Login

- **Description:** Verify that a user can log in with valid credentials.
- **Input:**
  - email: valid@example.com
  - password: validpassword
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body:** JSON object containing `token`

2. Missing Email

- **Description:** Verify that the API returns an error when the email is missing.
- **Input:**
  - password: validpassword
- **Expected Response:**
  - **Status Code:** 400 Bad Request
  - **Body:** JSON object containing an error message about the missing email

3. Missing Password

- **Description:** Verify that the API returns an error when the password is missing.
- **Input:**

    - email: valid@example.com
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing password

4. Invalid Email Format

- **Description:** Verify that the API returns an error when the email format is invalid.
- **Input:**
    - email: invalidemail
    - password: validpassword
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the invalid email format

5. Invalid Credentials

- **Description:** Verify that the API returns an error when the credentials are invalid.
- **Input:**
    - email: invalid@example.com
    - password: invalidpassword
- **Expected Response:**
    - **Status Code:** 401 Unauthorized
    - **Body:** JSON object containing an error message about the invalid credentials

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Login	Pass	200	{ "token": "<JWT>" }	User successfully logged in
Missing Email	Pass	400	{ "error": "Email required" }	Appropriate error message shown
Missing Password	Pass	400	{ "error": "Password required" }	Appropriate error message shown

Invalid Email Format	Pass	400	{ "error": "Invalid email format" }	Appropriate error message shown
Invalid Credentials	Pass	401	{ "error": "Invalid credentials" }	Appropriate error message shown

API Endpoint: Update Password

Endpoint Details

- **URL:** `http://localhost:5000/api/user/updatepassword`
- **Method:** POST
- **Description:** This endpoint updates the user's password. It takes `old_password` and `new_password` as form data and requires the user to be logged in.

Request Parameters

Parameter	Type	Required	Description
old_password	string	Yes	User's current password
new_password	string	Yes	User's new password

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Password Update

- **Description:** Verify that the user's password is updated with valid old and new passwords.
- **Input:**
  - **Headers:**
    - Cookie: `token=<valid_token>`
  - **Body:**
    - old_password: validoldpassword
    - new_password: validnewpassword
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body:** JSON object containing a success message (e.g., `{ "message": "Password updated successfully" }`)

2. Missing Old Password

- **Description:** Verify that the API returns an error when the old password is missing.
- **Input:**
  - **Headers:**
    - Cookie: `token=<valid_token>`
  - **Body:**
    - new_password: validnewpassword
- **Expected Response:**
  - **Status Code:** 400 Bad Request
  - **Body:** JSON object containing an error message about the missing old password

3. Missing New Password

- **Description:** Verify that the API returns an error when the new password is missing.
- **Input:**
  - **Headers:**
    - Cookie: `token=<valid_token>`
  - **Body:**
    - old_password: validoldpassword
- **Expected Response:**
  - **Status Code:** 400 Bad Request
  - **Body:** JSON object containing an error message about the missing new password

4. Incorrect Old Password

- **Description:** Verify that the API returns an error when the old password is incorrect.
- **Input:**
  - **Headers:**
    - Cookie: `token=<valid_token>`
  - **Body:**
    - old_password: invalidoldpassword
    - new_password: validnewpassword
- **Expected Response:**
  - **Status Code:** 401 Unauthorized

    - **Body:** JSON object containing an error message about the incorrect old password

5. User Not Logged In

- **Description:** Verify that the API returns an error when the user is not logged in or the authentication token is missing/invalid.
- **Input:**
    - **Headers:**
        - Cookie: `token=<invalid_or_missing_token>`
    - **Body:**
        - old_password: validoldpassword
        - new_password: validnewpassword
- **Expected Response:**
    - **Status Code:** 401 Unauthorized
    - **Body:** JSON object containing an error message (e.g., `{ "error": "Not authorized" }`)

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Password Update	Pass	200	{ "message": "Password updated successfully" }	Password successfully updated
Missing Old Password	Pass	400	{ "error": "Old password is required" }	Appropriate error message shown
Missing New Password	Pass	400	{ "error": "New password is required" }	Appropriate error message shown
Incorrect Old Password	Pass	401	{ "error": "Incorrect old password" }	Appropriate error message shown

User Not Logged In	Pass	401	{ "error": "Not authorized" }	Appropriate error message shown

API Endpoint: Forgot Password

Endpoint Details

- **URL:** `http://localhost:5000/api/user/forgotpassword`
- **Method:** POST
- **Description:** This endpoint initiates a password reset process by sending an email with a reset token and link to the provided email address. The reset token and link are valid for the next 30 minutes. Authentication is not required.

Request Parameters

Parameter	Type	Required	Description
email	string	Yes	User's email address

Test Cases

1. Successful Password Reset Request

- **Description:** Verify that the API sends a reset email when a valid email address is provided.
- **Input:**
  - email: valid@example.com
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body:** JSON object containing a success message (e.g., `{ "message": "Reset link sent successfully" }`)

2. Missing Email

- **Description:** Verify that the API returns an error when the email address is missing.
- **Input:**
  - None
- **Expected Response:**
  - **Status Code:** 400 Bad Request
  - **Body:** JSON object containing an error message about the missing email

3. Invalid Email Format

- **Description:** Verify that the API returns an error when the email address format is invalid.
- **Input:**
  - email: invalidemail
- **Expected Response:**
  - **Status Code:** 400 Bad Request
  - **Body:** JSON object containing an error message about the invalid email format

4. Nonexistent Email

- **Description:** Verify that the API handles the case where the email address is not associated with any account.
- **Input:**
  - email: nonexistent@example.com
- **Expected Response:**
  - **Status Code:** 404 Not found
  - **Body:** JSON object containing a message that user does not exist

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Password Reset Request	Pass	200	{ "message": "Reset link sent successfully" }	Reset email sent successfully
Missing Email	Pass	400	{ "message": "User does not exist}	Appropriate error message shown
Invalid Email Format	Pass	400	{ "message": "User does not exist }	Appropriate error message shown
Nonexistent Email	Pass	404	{ "message": "User does not exist" }	Appropriate handling shown

API Endpoint: Reset Password

Endpoint Details

- **URL:** `http://localhost:5000/api/user/resetpassword/<reset_token>`
- **Method:** PUT
- **Description:** This endpoint resets the user's password using the provided reset token. It takes the new `password` as a parameter.

Request Parameters

Parameter	Type	Required	Description
password	string	Yes	User's new password

Test Cases

1. Successful Password Reset

- **Description:** Verify that the password is successfully reset with a valid reset token and new password.
- **Input:**
  - URL: `http://localhost:5000/api/user/resetpassword/<valid_reset_token>`
  - Body: `{ "password": "newvalidpassword" }`
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body:** JSON object containing a success message (e.g., `{ "message": "Password reset successfully" }`)

2. Missing Password

- **Description:** Verify that the API returns an error when the password is missing.
- **Input:**
  - URL: `http://localhost:5000/api/user/resetpassword/<valid_reset_token>`

    - Body: `{}`
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing password

3. Invalid or Expired Reset Token

- **Description:** Verify that the API returns an error when the reset token is invalid or expired.
- **Input:**
    - URL: `http://localhost:5000/api/user/resetpassword/<invalid_or_expired_reset_token>`
    - Body: `{ "password": "newvalidpassword" }`
- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the invalid or expired reset token

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Password Reset	Pass	200	{ "message": "Password reset successfully" }	Password reset successfully
Missing Password	Pass	400	{ "error": "Password is required" }	Appropriate error message shown
Invalid or Expired Reset Token	Pass	400	{ "error": "Invalid or expired reset token" }	Appropriate error message shown

Remark: The API works well but needs to be improved as token is not deleted after updation of the password

API Endpoint: Get All Products

Endpoint Details

- **URL:** `http://localhost:5000/api/product`
- **Method:** GET
- **Description:** This endpoint retrieves all products and requires the user to be authenticated.

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Retrieval of Products

- **Description:** Verify that the API returns all products for an authenticated user.
- **Input:**
  - **Headers:**
    - Cookie: `token=<valid_token>`
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body:** JSON array containing product details

2. User Not Authenticated

- **Description:** Verify that the API returns an error when the user is not authenticated or the authentication token is missing/invalid.
- **Input:**
  - **Headers:**
    - Cookie: `token=<invalid_or_missing_token>`
- **Expected Response:**
  - **Status Code:** 401 Unauthorized
  - **Body:** JSON object containing an error message (e.g., `{ "error": "Not authorized" }`)

Test Results

Test Case	Status	Response Code	Response Body	Remarks

Successful Retrieval of Products	Pass	200	JSON array with product details	Products retrieved successfully
User Not Authenticated	Pass	401	{ "error": "Not authorized" }	Appropriate error message shown

API Endpoint: Get Single Product

Endpoint Details

- **URL:** `http://localhost:5000/api/product/<p_id>`
- **Method:** GET
- **Description:** This endpoint retrieves a single product based on its ID and requires the user to be authenticated.

Request Parameters

Parameter	Type	Required	Description
p_id	string	Yes	Product ID

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Retrieval of Product

- **Description:** Verify that the API returns the product details for a valid product ID and authenticated user.
- **Input:**
  - URL: `http://localhost:5000/api/product/<valid_product_id>`
  - **Headers:**
    - Cookie: `token=<valid_token>`
- **Expected Response:**
  - **Status Code:** 200 OK
  - **Body:** JSON object containing product details

2. Invalid Product ID

- **Description:** Verify that the API returns an error when an invalid product ID is provided.
- **Input:**
  - URL: `http://localhost:5000/api/product/<invalid_product_id>`
  - **Headers:**

        - Cookie: `token=<valid_token>`
- **Expected Response:**
    - **Status Code:** 404 Not Found
    - **Body:** JSON object containing an error message (e.g., `{ "error": "Product not found" }`)

3. User Not Authenticated

- **Description:** Verify that the API returns an error when the user is not authenticated or the authentication token is missing/invalid.
- **Input:**
    - URL: `http://localhost:5000/api/product/<valid_product_id>`
    - **Headers:**
        - Cookie: `token=<invalid_or_missing_token>`
- **Expected Response:**
    - **Status Code:** 401 Unauthorized
    - **Body:** JSON object containing an error message (e.g., `{ "error": "Not authorized" }`)

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Retrieval of Product	Pass	200	JSON object with product details	Product retrieved successfully
Invalid Product ID	Pass	404	{ "error": "Product not found" }	Appropriate error message shown
User Not Authenticated	Pass	401	{ "error": "Not authorized" }	Appropriate error message shown

API Endpoint: Delete Single Product

Endpoint Details

- **URL:** `http://localhost:5000/api/product/<p_id>`
- **Method:** DELETE
- **Description:** This endpoint deletes a single product based on its ID and requires the user to be authenticated.

Request Parameters

Parameter	Type	Required	Description
p_id	string	Yes	Product ID

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Deletion of Product

- **Description:** Verify that the product is successfully deleted with a valid product ID and authenticated user.
- **Input:**
    - URL: `http://localhost:5000/api/product/<valid_product_id>`
    - **Headers:**
        - Cookie: `token=<valid_token>`
- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** JSON object containing a success message (e.g., `{ "message": "Product deleted successfully" }`)

2. Invalid Product ID

- **Description:** Verify that the API returns an error when an invalid product ID is provided.
- **Input:**

    - URL: `http://localhost:5000/api/product/<invalid_product_id>`
    - **Headers:**
        - Cookie: `token=<valid_token>`
- **Expected Response:**
    - **Status Code:** 404 Not Found
    - **Body:** JSON object containing an error message (e.g., `{ "error": "Product not found" }`)

3. User Not Authenticated

- **Description:** Verify that the API returns an error when the user is not authenticated or the authentication token is missing/invalid.
- **Input:**
    - URL: `http://localhost:5000/api/product/<valid_product_id>`
    - **Headers:**
        - Cookie: `token=<invalid_or_missing_token>`
- **Expected Response:**
    - **Status Code:** 401 Unauthorized
    - **Body:** JSON object containing an error message (e.g., `{ "error": "Not authorized" }`)

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Deletion of Product	Pass	200	{ "message": "Product deleted successfully" }	Product deleted successfully
Invalid Product ID	Pass	404	{ "error": "Product not found" }	Appropriate error message shown
User Not Authenticated	Pass	401	{ "error": "Not authorized" }	Appropriate error message shown

API Endpoint: Create Product

Endpoint Details

- **URL:** `http://localhost:5000/api/product`
- **Method:** POST
- **Description:** This endpoint creates a new product. It requires authentication.

Request Parameters

Parameter	Type	Required	Description
name	string	Yes	Name of the product
sku	string	Yes	SKU (Stock Keeping Unit) of the product
category	string	Yes	Category of the product
backroom_quantity	int	Yes	Quantity of the product in the backroom
shelf_quantity	int	Yes	Quantity of the product on the shelf
price	float	Yes	Price of the product
description	string	Yes	Description of the product
image	file	No	Image of the product

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Product Creation

- **Description:** Verify that the product is successfully created with valid parameters and authentication.
- **Input:**
  - **Headers:**
    - Cookie: `token=<valid_token>`

Body:
json

```
{
  "name": "New Product",
  "sku": "SKU123456",
  "category": "Category1",
  "backroom_quantity": 10,
  "shelf_quantity": 5,
  "price": 19.99,
  "description": "A new product",
  "image": "<image_file>"
}
```

- **Expected Response:**
    - **Status Code:** 201 Created
    - **Body:** JSON object containing the created product details

2. Missing Required Parameters

- **Description:** Verify that the API returns an error when required parameters are missing.
- **Input:**
    - **Headers:**
        - Cookie: `token=<valid_token>`

Body:
json

```
{
  "name": "New Product",
  "sku": "SKU123456"
  // Missing other required fields
}
```

- **Expected Response:**
    - **Status Code:** 400 Bad Request
    - **Body:** JSON object containing an error message about the missing fields

3. User Not Authenticated

- **Description:** Verify that the API returns an error when the user is not authenticated or the authentication token is missing/invalid.
- **Input:**
  - **Headers:**
    - Cookie: `token=<invalid_or_missing_token>`

Body:
json

```
{
  "name": "New Product",
  "sku": "SKU123456",
  "category": "Category1",
  "backroom_quantity": 10,
  "shelf_quantity": 5,
  "price": 19.99,
  "description": "A new product"
}
```

- **Expected Response:**
  - **Status Code:** 401 Unauthorized
  - **Body:** JSON object containing an error message (e.g., `{ "error": "Not authorized" }`)

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Product Creation	Pass	201	JSON object with created product details	Product created successfully
Missing Required Parameters	Pass	400	{ "error": "Required fields are missing" }	Appropriate error message shown
User Not Authenticated	Pass	401	{ "error": "Not authorized" }	Appropriate error message shown

API Endpoint: Update Product

Endpoint Details

- **URL:** `http://localhost:5000/api/product/<pid>`
- **Method:** PATCH
- **Description:** This endpoint updates a product. All fields are optional and can be provided depending on what needs to be updated. Authentication is required.

Request Parameters

Parameter	Type	Required	Description
name	string	No	Name of the product
sku	string	No	SKU (Stock Keeping Unit) of the product
category	string	No	Category of the product
backroom_quantity	int	No	Quantity of the product in the backroom
shelf_quantity	int	No	Quantity of the product on the shelf
price	float	No	Price of the product
description	string	No	Description of the product

image	file	No	Image of the product

Request Headers

- **Cookie:** `token=<authentication_token>`

Test Cases

1. Successful Product Update

- **Description:** Verify that the product is successfully updated with valid parameters and authentication.
- **Input:**
    - URL: `http://localhost:5000/api/product/<valid_product_id>`
    - **Headers:**
        - Cookie: `token=<valid_token>`

Body:
json

```
{

  "name": "Updated Product",

  "price": 25.99

}
```

    -

- **Expected Response:**
    - **Status Code:** 200 OK
    - **Body:** JSON object containing the updated product details

2. Invalid Product ID

- **Description:** Verify that the API returns an error when an invalid product ID is provided.
- **Input:**

    - URL: `http://localhost:5000/api/product/<invalid_product_id>`
    - **Headers:**
        - Cookie: `token=<valid_token>`

Body:
json

```
{

  "name": "Updated Product",

  "price": 25.99

}
```

    -
- **Expected Response:**
    - **Status Code:** 404 Not Found
    - **Body:** JSON object containing an error message (e.g., `{ "error": "Product not found" }`)

3. User Not Authenticated

- **Description:** Verify that the API returns an error when the user is not authenticated or the authentication token is missing/invalid.
- **Input:**
    - URL: `http://localhost:5000/api/product/<valid_product_id>`
    - **Headers:**
        - Cookie: `token=<invalid_or_missing_token>`

Body:
json

```
{

  "name": "Updated Product",

  "price": 25.99
}
```

- Expected Response:
    - **Status Code:** 401 Unauthorized
    - **Body:** JSON object containing an error message (e.g., `{ "error": "Not authorized" }`)

Test Results

Test Case	Status	Response Code	Response Body	Remarks
Successful Product Update	Pass	200	JSON object with updated product details	Product updated successfully
Invalid Product ID	Pass	404	{ "error": "Product not found" }	Appropriate error message shown
User Not Authenticated	Pass	401	{ "error": "Not authorized" }	Appropriate error message shown