TESTING

Title: Multi Factor Authentication using MERN.

Introduction: In today's digital landscape, security has become a paramount concern for both users and developers. With the increasing frequency of cyberattacks and data breaches, traditional authentication methods relying solely on usernames and passwords are no longer sufficient to safeguard sensitive information. Multi-Factor Authentication (MFA) has emerged as a crucial security measure, providing an additional layer of protection by requiring users to verify their identity through multiple factors.

Postman Testing: Postman testing involves simulating HTTP requests (e.g., GET, POST, PUT) to APIs, allowing developers to verify the functionality, performance, and security of their endpoints. It is commonly used for testing RESTful APIs by sending requests with specific parameters, headers, and payloads, and analysing the server's response for accuracy and correctness**.**

Approach:

- User login with MFA
- Generation and Validation of one-time-passwords (OTPs)
- Integration of OTP-based multi-factor authentication into the login process

Testing: In our MFA MERN project, the postman testing includes only POST method.

It includes two operations:

1. 'auth/login'
2. 'auth/verify-otp'

Testing login operation:

Steps for testing:

Step 1: Set up the request in postman

- Method: `POST`
- URL: ` http://localhost:3003/auth/login`

Step 2: Define the Request Body

- The request body will typically include user credentials like email and password.
- **Body Type:** Choose `raw` and select `Json` format.
- **Example JSON Body:**

```
{
  "email" : "user@gmail.com",
  "password" : "userpassword"
}
```

Step 3: Set Headers

- Ensure the content type is set to JSON.
- Header:
  `content-Type`: `application/json`

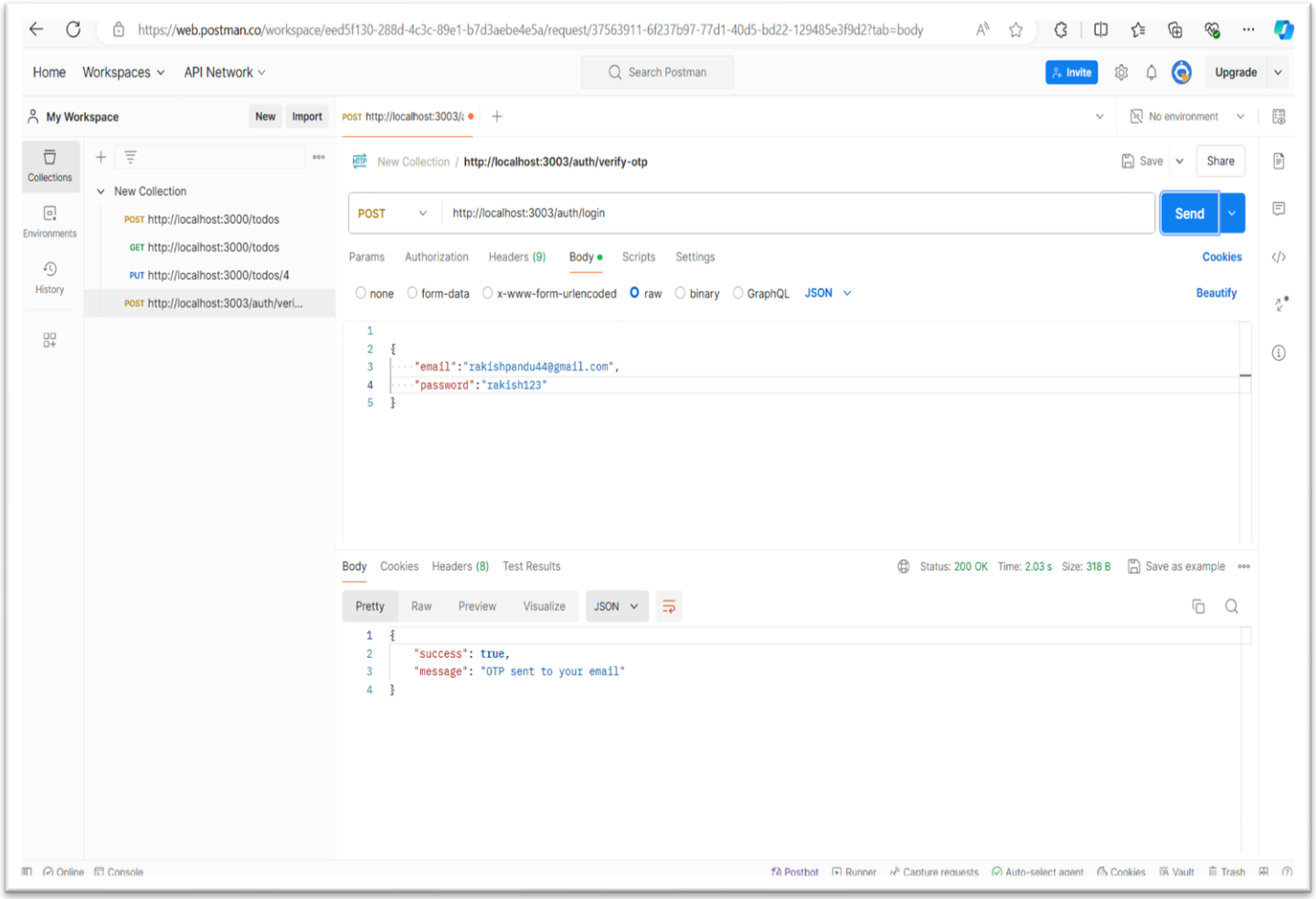
Step 4: Send the Request

- Click the `send` button in postman to send the login request.

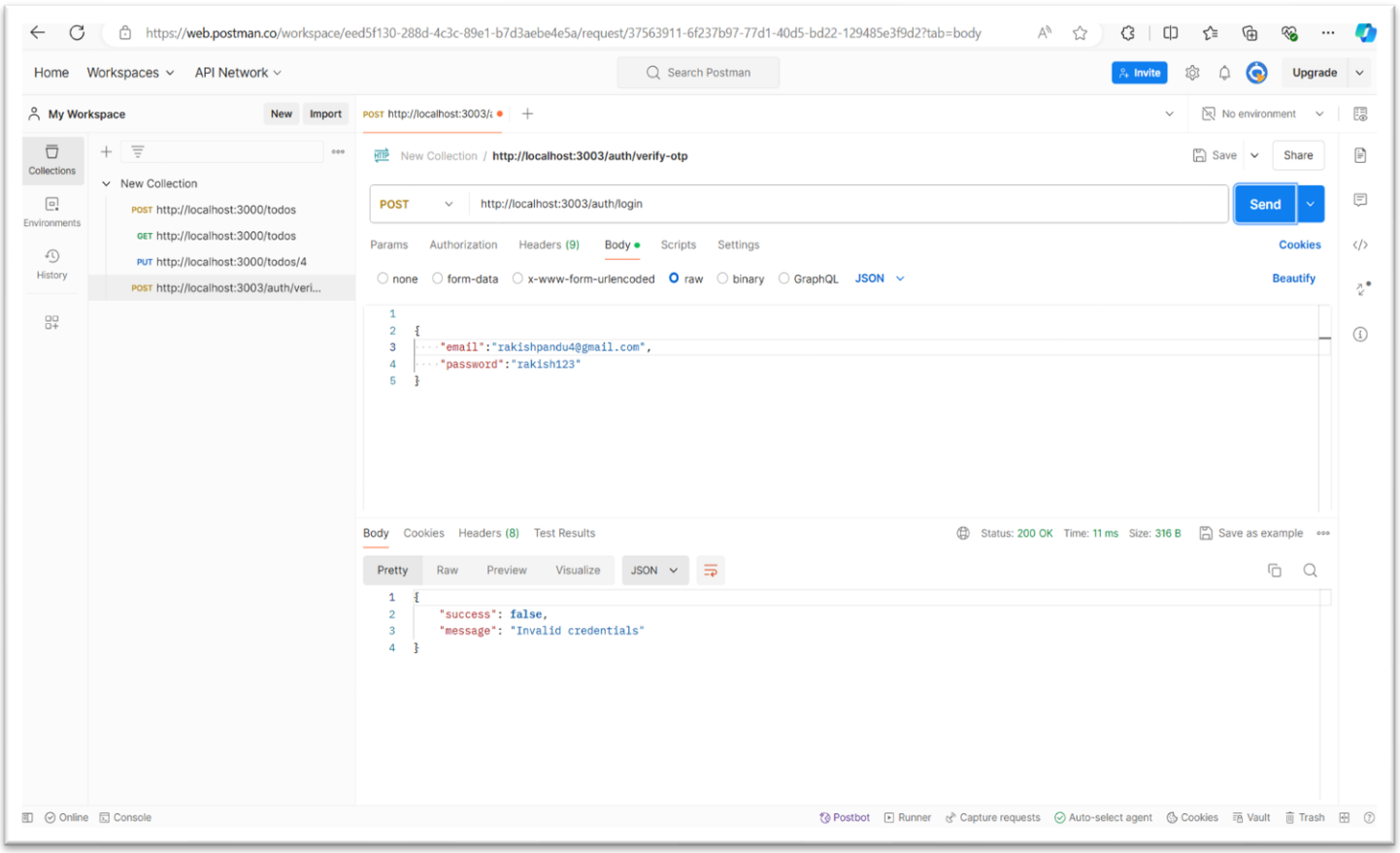
Step 5: Analyse the Response

- Success Response (200) : If the login is successful, you will typically receive a response containing the status of the login details.

```
{
    "success": true,
    "message": "OTP sent to your email"
}
```



- Error Response: If login fails i.e., incorrect credentials, you will get an error response.



Testing OTP Verification:

Step 1: Set Up the Request in Postman

- Method: `POST`
- URL: `http://localhost:3003/auth/verify-otp`

Step 2: Define the Request Body

Since this is an OTP verification endpoint, the request body will typically include data such as :

- `email`: The user's email that the OTP was sent to.
- `otp`: The One-Time Password entered by the user.
- **Body Type:** Choose `raw` and select `Json` format.
- **Example JSON Body:**

{

"email" : "user@gmail.com",

"otp" : "123456"

}

Step 3: Set Headers

- Ensure the content type is set to JSON.
- Header:
  `content-Type`: `application/json`

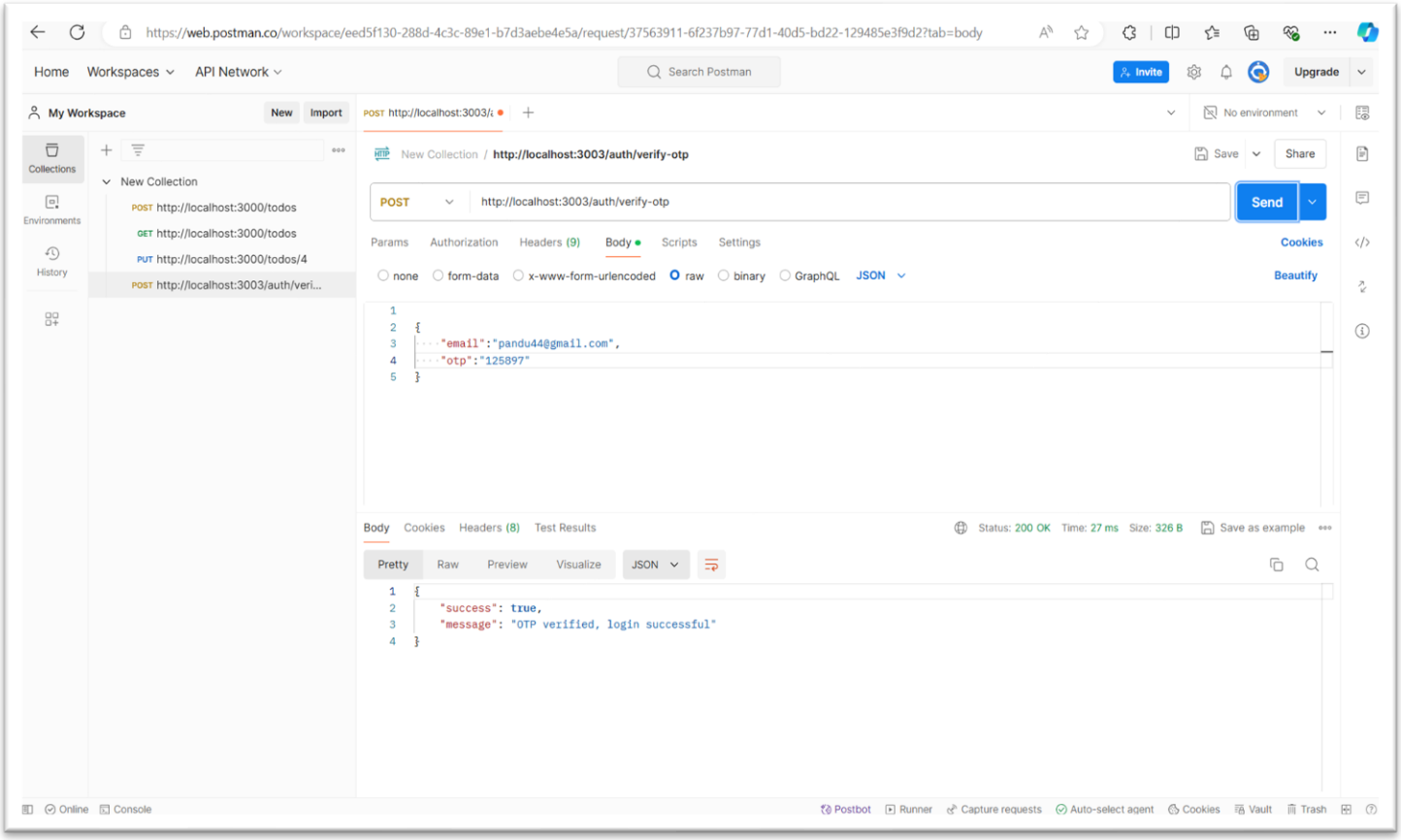
Step 4: Send the Request

- Click the `send` button in postman to send the login request.

Step 5: Analyse the Response

- Success Response (200) : Typically, it would return a success message indicating that the OTP was verified.

```
{
    "success": true,
    "message": "OTP verified, login successful"
}
```



- Error response: If the OTP is incorrect or expired, it might return an error message.

```
{
    "success": false,
    "message": "Invalid OTP"
}
```