BUG-001

Title: POST /api/v1/users does not return token upon successful registration (contrary to API documentation)

Severity: High

Environment: Localhost

Steps to Reproduce:

1. Send POST request to /api/v1/users with a valid body:

{

"email": "user949439 @gmail.com",

"password": "user123",

"name": "user"

}

2. Observe the response.

Expected Result:

- Status 200
- Response body contains:

{

"message": "User registered with success",

"token": "<JWT_TOKEN>"

}

Actual Result:

- Status is correct (200).
- Response body **does not** contain token property, only:

{

"message": "User registered with success"

}

CREATE USER

- REQUEST

```
POST /api/v1/users
```

```
{
  "name": "user",
  "email": "user@gmail.com",
  "password": "user123"
}
```

- RESPONSE

```
{
  "message": "User registered with success",
  "token": "eyJhbGciOiJI..."
}
```

BUG-002 — Auth returns token when email missing

Title: POST /api/v1/auth issues token and returns 200 when email field is missing.
Severity: Critical / High (security & auth flow).
Environment: Local dev (http://localhost:3000)
Attachment: screenshot (provided).

Steps to reproduce

1. Send POST to /api/v1/auth with JSON body containing only password:

- '{"password":"user123"}'

2. Observe response status and body.

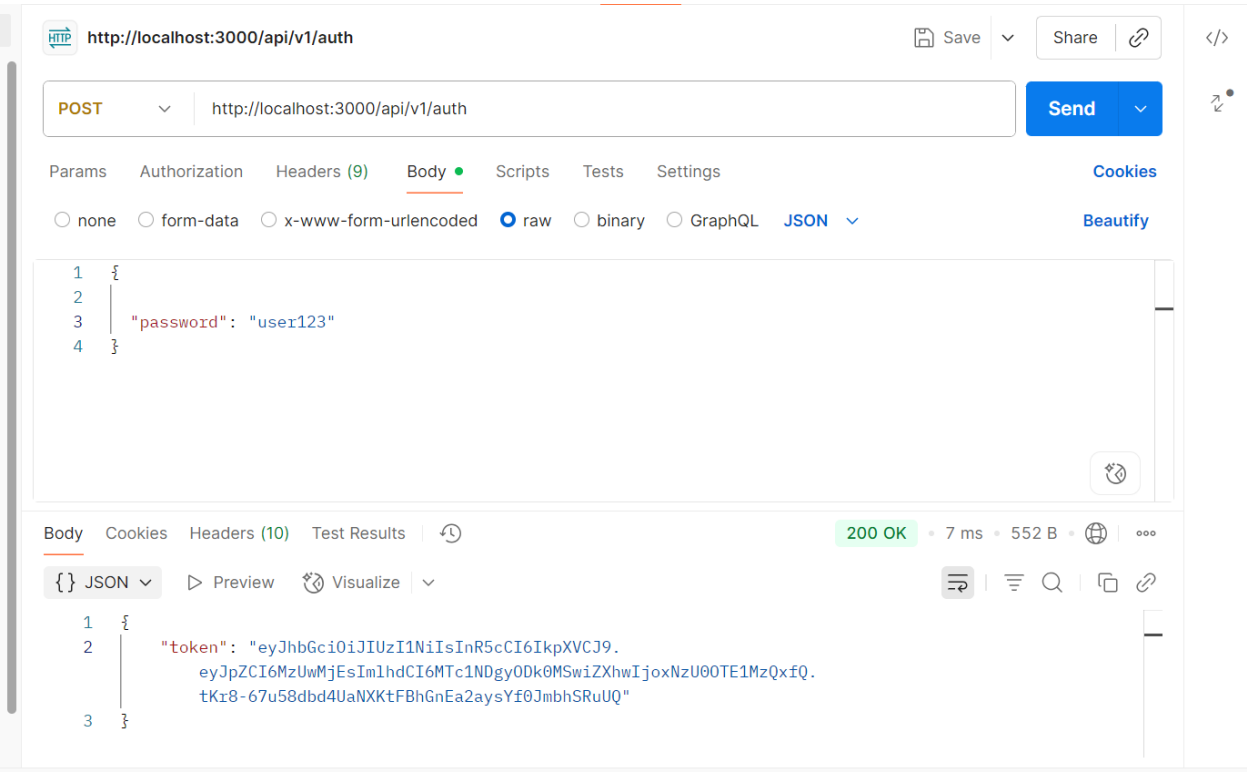
Actual result

- Response status: 200 OK.

- Response body contains a token (JWT).

Expected result

- Response status: 400 Bad Request (or at least not 200) when required field(s) are missing.
- Response body should **not** contain token. Example:



BUG: 003
Title: Plain-text password returned in API response after PATCH request to /api/v1/users

Severity: Critical (Security Vulnerability)
Priority: High

Description:
When updating user details via the PATCH /api/v1/users endpoint, the API response includes the user's password in **plain text**. This is a major security flaw — passwords should never be returned in API responses in any form (especially unencrypted).

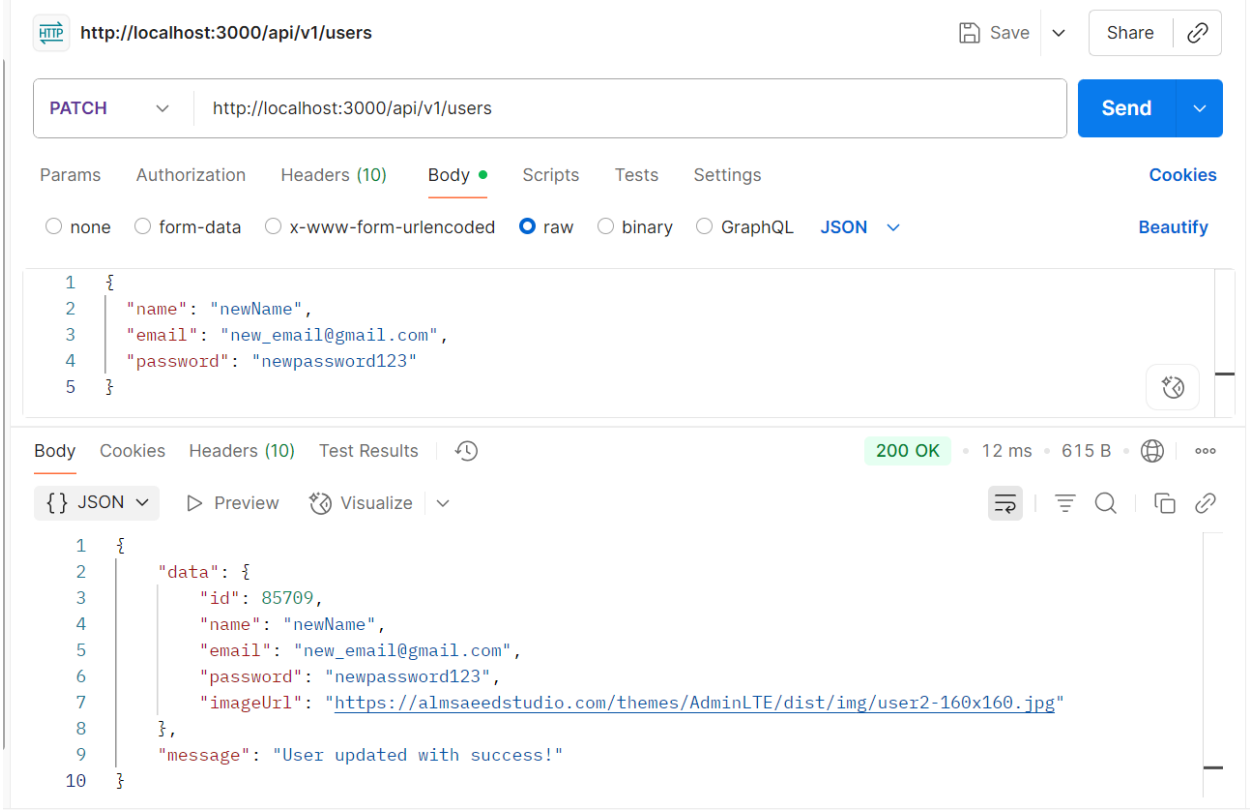
Steps to Reproduce:

1. Open Postman (or any API testing client).

2. Send a PATCH request to /api/v1/users with valid authentication and updated user data.
3. Check the API response.

Actual Result:

- API response body contains the password field in plain text.



Expected Result:

- API should **never** return passwords in any form.
- Password field must be excluded from API responses entirely, even if hashed.

```
PATCH /api/v1/users
```

```
const token = 'eyJhbGciOiJI...';
req.setRequestHeader('Authorization', token);
```

```
{
  "name": "newName",
  "email": "new_email@gmail.com",
  "password": "newpassword123"
}
```

- RESPONSE

```
{
  "message": "User updated with success"
}
```

BUG: 004
Title: User deletion requires undocumented admin privileges – DELETE /api/v1/users
Severity: Major (Functionality / Documentation Gap)
Priority: High

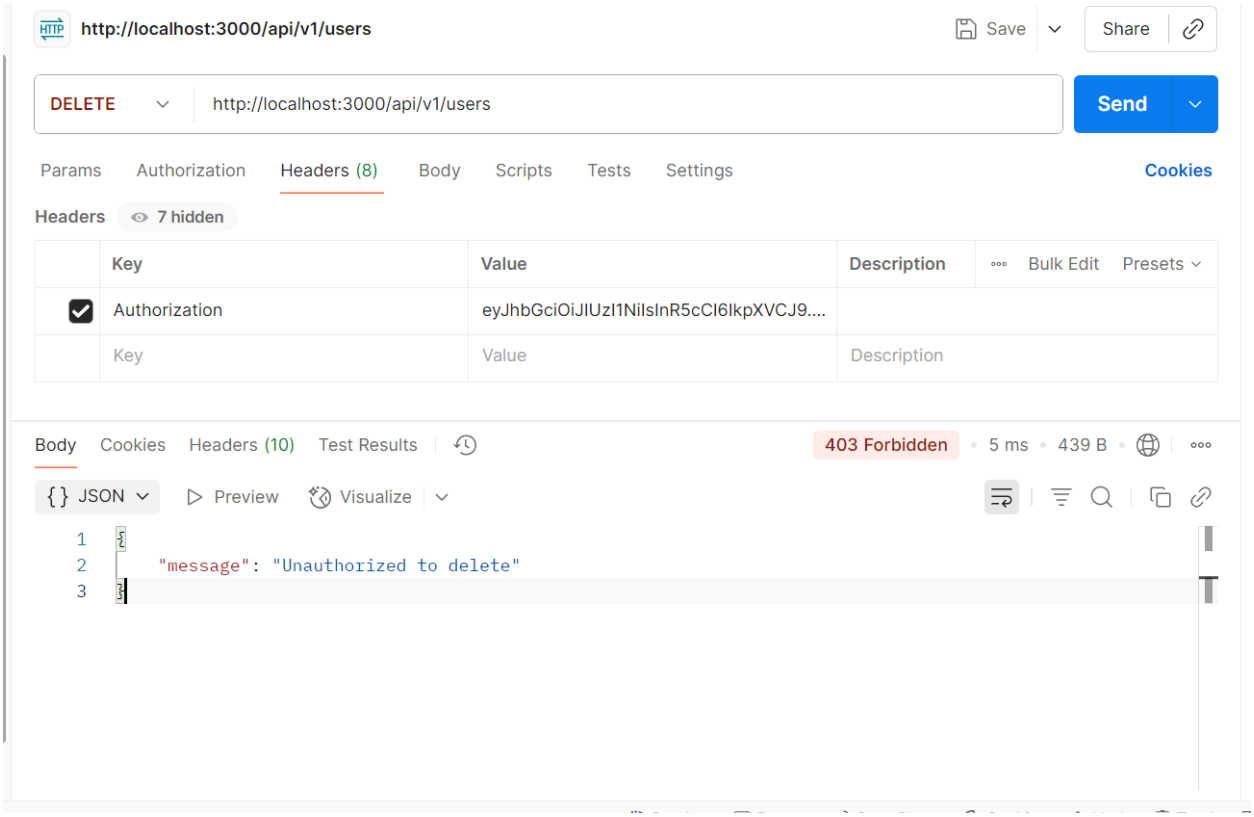
Description:
When sending a DELETE request to /api/v1/users with a valid token (from a regular authenticated user), the API does not delete the user as expected.
The API documentation states that providing a valid token and sending the request should result in successful deletion. However, actual behavior suggests that admin privileges may be required — this requirement is not documented.

Steps to Reproduce:

1. Open Postman (or any API testing client).
2. Authenticate as a regular user and retrieve a valid token.
3. Send a DELETE /api/v1/users request with the token in the Authorization header.
4. Check the API response.

Actual Result:

- Request is rejected or does not delete the user.
- No clear message about needing admin privileges.



Expected Result:

- If admin privileges are required, the documentation must clearly state this.
- If admin privileges are not required, deletion should work with a valid authenticated user token as per the documentation.

DELETE USER BY TOKEN

- REQUEST

```
DELETE /api/v1/users
```

```
const token = 'eyJhbGciOiJI...';
req.setRequestHeader('Authorization', token);
```

- RESPONSE

```
{
  "message": "User deleted with success"
}
```

BUG: 005

Title:
API allows user creation with missing or empty required fields after delete operation

Severity:

High

Description:
When performing a DELETE request first to remove a user (either specific or all users) and then calling the POST /api/v1/users endpoint with an empty body or with missing one or more of the required fields (name, email, password), the API still creates the user successfully and returns status **200** with the message "User registered with success".

This behavior is incorrect because the API should validate the request body and reject any creation attempt that does not include all mandatory fields.

Steps to Reproduce:

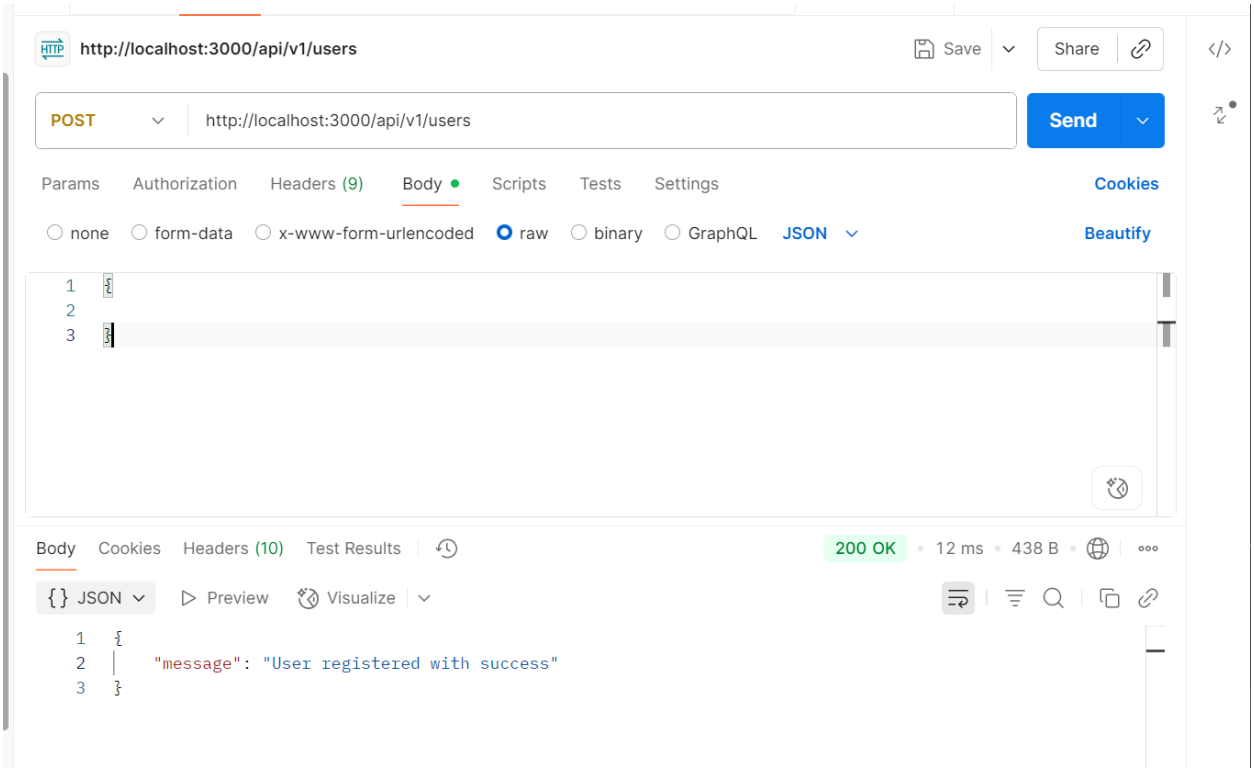
1. Send DELETE /api/v1/users (or /api/v1/all-users) to remove the user(s).
2. Send POST /api/v1/users with one of the following bodies:
    - {}

- { "name": "test", "email": "test@gmail.com" } (missing password)
- { "name": "test", "password": "pass123" } (missing email)
- { "email": "test@gmail.com", "password": "pass123" } (missing name)

3. Observe the response.

Expected Result:
The API should return **400 Bad Request** or **422 Unprocessable Entity** with an appropriate error message indicating the missing required fields.

Actual Result:
The API returns **200 OK** with "User registered with success" and creates the user despite missing mandatory fields.



BUG: 006

Title:

Incorrect status code and message when creating a user with missing fields (without prior delete)

Severity:

High

Description:

In the normal flow (without performing a delete beforehand), when calling POST /api/v1/users with missing one or more required fields (name, email, password), the API responds with 401 Unauthorized and the message "User already exists".

This behavior is incorrect because:

The status code 401 is intended for authentication errors, not for validation failures.

The "User already exists" message is misleading since the user does not exist — the request is simply missing required fields.

Steps to Reproduce:

Without performing any delete operation, call POST /api/v1/users with one of the following:

{ "name": "test", "email": "test@gmail.com" } (missing password)

{ "name": "test", "password": "pass123" } (missing email)

{ "email": "test@gmail.com", "password": "pass123" } (missing name)

Observe the response.

Expected Result:

The API should return 400 Bad Request or 422 Unprocessable Entity.

The message should indicate the specific missing field(s), e.g., "Password is required" or "Email is required".

Actual Result:

The API returns 401 Unauthorized.

The message incorrectly states "User already exists".

BUG: 007

Title: Missing Documentation for Default Profile Image When No Image Is Provided in User Creation API

Severity: Medium

Description:
When creating a user via the **Create User API** without providing a profile image, the system automatically assigns a default image. However, this behavior is not documented in the API documentation. This may lead to confusion for API consumers who expect the image field to be null or omitted when no image is provided.

Steps to Reproduce:

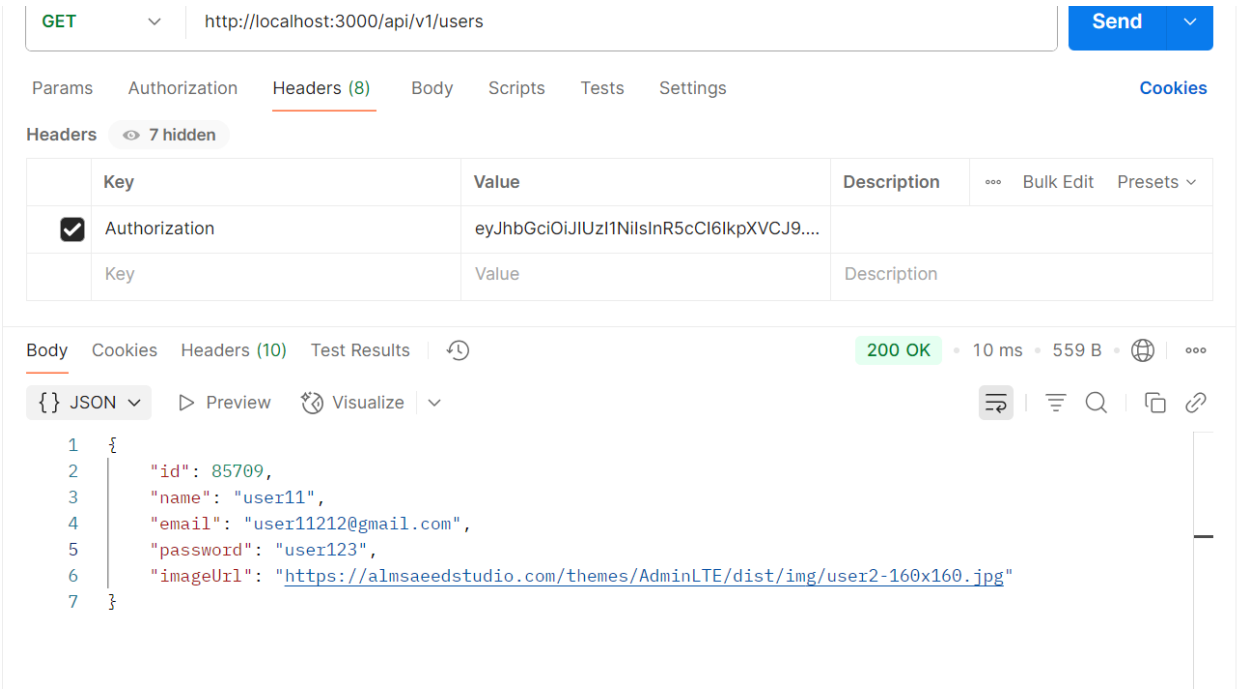
1. Send a **POST** request to /api/v1/users without including a profile image in the request body.
2. Retrieve the created user via a **GET** request to /api/v1/users/{id}.

Expected Result:

- The API documentation should clearly state that if no image is provided, the system assigns a default image and returns its link in the image field.

Actual Result:

- The API assigns a default image automatically and returns its URL, but this is **not documented** in the API specifications.


GET
http://localhost:3000/api/v1/users
Send
Params
Authorization
Headers (8)
Body
Scripts
Tests
Settings
Cookies
Headers
7 hidden
Key
Value
Description
Bulk Edit
Presets
Authorization
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9....
Key
Value
Description
Body
Cookies
Headers (10)
Test Results
200 OK
10 ms
559 B
JSON
Preview
Visualize
{
"id": 85709,
"name": "user11",
"email": "user11212@gmail.com",
"password": "user123",
"imageUrl": "https://almsaeedstudio.com/themes/AdminLTE/dist/img/user2-160x160.jpg"
}