# API Bug report

1) PATCH /users – Missing token returns 403 forbidden instead of 401

- **Issue ID:** 1
- **Title:** PATCH /users returns 403 for missing token instead of 401 unauthorized
- **Precondition**: User exists in the database, but the request is sent without an Authorization header.
- **Steps to reproduce:** Send PATCH /users with no Authorization header, (body { name: "NoAuthUser" }).

  **Expected Result:** should return`401 Unauthorized` with message: "jwt must be provided".

  **Actual Result:** returned 403 Forbidden; body logs show { message: "jwt must be provided" }.

2) PATCH /users – Invalid token returns 403 instead of 401

- **Issue ID:** 2
- **Title:** PATCH /users returns 403 for invalid token instead of 401
- **Precondition:** An Invalid or malformed JWT token is prepared
- **Steps to reproduce:** Send PATCH /users with Authorization: Bearer invalidToken123.

  **Expected Result:** should return 401 Unauthorized with message: "invalid signature" (or equivalent).

  **Actual Result:** returned 403 Forbidden; body shows { message: "jwt malformed" }.

3) PATCH /users – Email can be updated to an existing user's email

- **Issue ID:** 3
- **Title:** PATCH /users | API lets a user update their email to one that's already taken by another account
- **Preconditions:** Two users (A and B) exist with different emails.
- **Steps to reproduce:**

  1. Log in as User A.
  2. Send PATCH /users to update A's email to B's email

  **Expected Result:** should return 400 Bad Request with a message like "Email already in use".

**Actual Result**: it returned 200 OK with "User updated with success!".

4) POST /auth – Empty email returns 401 unauthorized

- **Issue ID:** 4
- **Title:** Missing email triggers an authentication error instead of a validation error
- **Preconditions:** Valid user exists in the database
- **Steps:** Send POST /auth with body { password: "P@ssw0rd123" }.

  **Expected Result:** should return`400 Bad Request` with message "Email and password are required".

  **Actual Result**: It returned 401 Unauthorized with the message "Incorrect email or password."

5) POST /auth – Empty password returns 401 (should be validation error)

- **Issue ID:** 5
- **Title:** Missing password results in 401 Unauthorized instead of 400 Bad Request
- **Preconditions:** Valid user exists in the database
- **Steps:** Send POST /auth with body { email: <valid or random> }.

  **Expected Result:** It should return 400 Bad Request with the message "Email and password are required".

  **Actual Result:** It returned 401 Unauthorized.

6) GET /users – Missing token returns 403 instead of 401

- **Issue ID:** 6
- **Title:** GET /users returns 403 for missing token instead of 401
- **Preconditions:** Valid user exists in the database
- **Steps:** Send 'GET /users' with no Authorization header.

  Expected Result: It should return '401 Unauthorized'.

  Actual Result: it returned '403 Forbidden'.

7) GET /users – Invalid token returns 403 instead of 401

- **Issue ID:** 7
- **Title:** Invalid tokens return 403 Forbidden instead of 401 Unauthorized
- **Precondition:** An Invalid or malformed JWT token is prepared

- **Steps:** Send GET /users with Authorization: Bearer not.a.valid.token.

  **Expected Result:** it should return`401 Unauthorized`.

  **Actual Result:** it returned`403 Forbidden`.

8) DELETE /users – Missing token returns 403 instead of 401

- **Issue ID:** 9
- **Title:** `DELETE /users` returns 403 for missing token instead of 401
- **Steps:** Send `DELETE /users` with **no Authorization** header.

  **Expected:** it should return `401 Unauthorized`.

  **Actual:** the request returned `403 Forbidden`.

9) DELETE /users – Invalid token returns 403 instead of 401

- **Issue ID:** 10
- **Title:** `DELETE /users` returns **403** for invalid token instead of 401 unauthorized
- **Steps:** Send `DELETE /users` with `Authorization: Bearer not.a.valid.token`.

  **Expected Result:** the request should return`401 Unauthorized`.

  **Actual Result:** it returned `403 Forbidden`.

10) POST /users – Successful registration does not return a token

- **Issue ID:** 11
- **Title:** `POST /users` success response missing `token` property
- **Steps:** Register a new user with a valid body.

  **Expected Result:** The response body should contain the token when the registration is successful

  **Actual Result:** The registration is done successfully with body{ message: "User registered with success" }` but no `token`.

11) POST /users – Creation without password succeeds (should fail)

- **Issue ID:** 12
- **Title:** `POST /users` returns 200 OK when password is missing instead of 400 Bad Request
- **Steps:** Send `POST /users` with a missing `password`.

  **Expected Result:** It should return `400 Bad Request` with a validation message.

**Actual Result:** it returned `200 OK` (user created).

12) POST /users – Creation without email succeeds (should fail)

- **Issue ID:** 13
- **Title:** `POST /users` returns 200 OK when email is missing instead of 400 Bad Request
- **Steps:** Send `POST /users` with missing `email`.

  **Expected Result:** it should return`400 Bad Request` with validation message.

  **Actual Result:** it returned `200 OK`.

13) POST /users – Duplicate registration returns 401 instead of 400

- **Issue ID:** 14
- **Title:** Registering a user with an email that's already registered returns **401**
- **Steps:**

  1. Send `POST /users` with a new user → success.

  2. Send `POST /users` again with the same email.

  **Expected Result:** it should return `400 Bad Request` with `"User already registered"` and no `token`.

  **Actual Result:** it returned `401 Unauthorized`.