# Logical and security testing

# Contents

# Vulnerability Report for Flask Application

Steps to run the flask application.

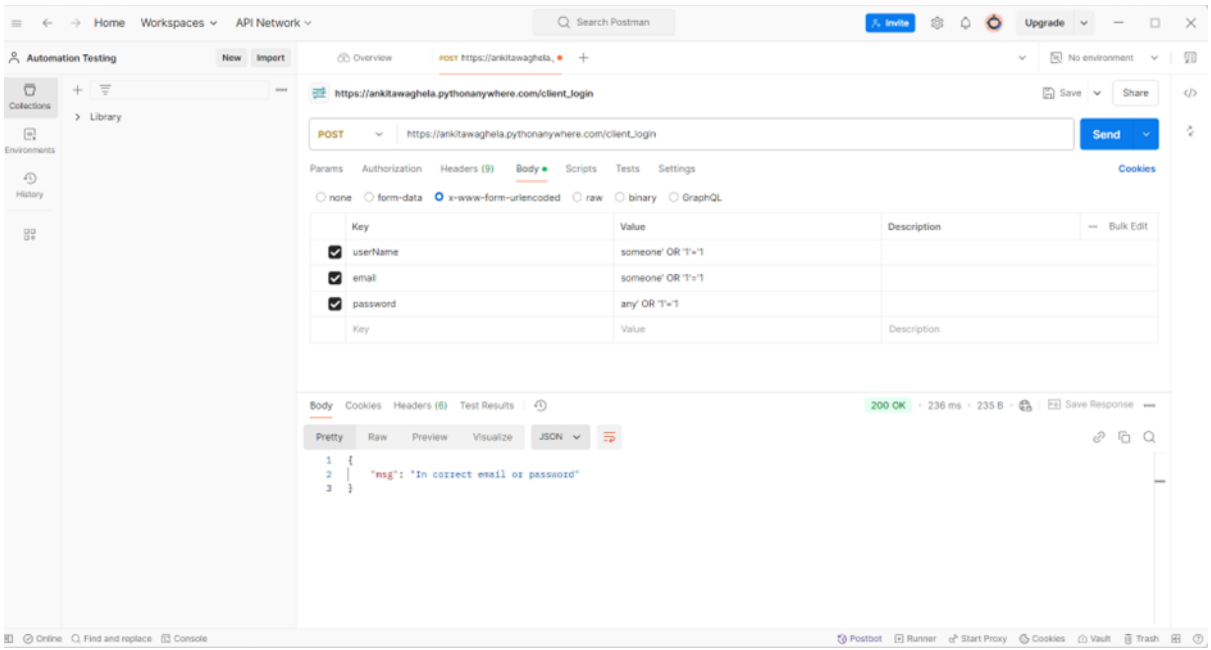
1. Log in to your PythonAnywhere dashboard.
2. Open a new Bash console from the dashboard.
3. Clone the GitHub repository: git clone https://github.com/Galileo0/Soar_Test.git
4. Navigate to the cloned directory: cd Soar_Test
5. Create a virtual environment: mkvirtualenv --python=/usr/bin/python3.8 myenv
6. Install the required dependencies: pip install flask jwt requests
7. Initialize the database: python db_init.py
8. Go to the "Web" tab in your PythonAnywhere dashboard and create a new web app.
9. Choose "Manual configuration" and select Python 3.8.
10. Set the working directory to /home/yourusername/Soar_Test.
11. Modify the WSGI file to include your virtual environment path and import your Flask app:
    a. `import sys`
    b. `path = '/home/yourusername/Soar_Test'`
    c. `if path not in sys.path:`
    d. `sys.path.append(path)`
    e. `from task import app as application`
12. Save the changes and reload your web app.

# 1. SQL Injection in /client_login

Risk Score: Critical

The **/client_login** endpoint is vulnerable to SQL injection attacks due to unsanitized user input in SQL queries.

Soar_Test/task.py - Line 77, Line 78



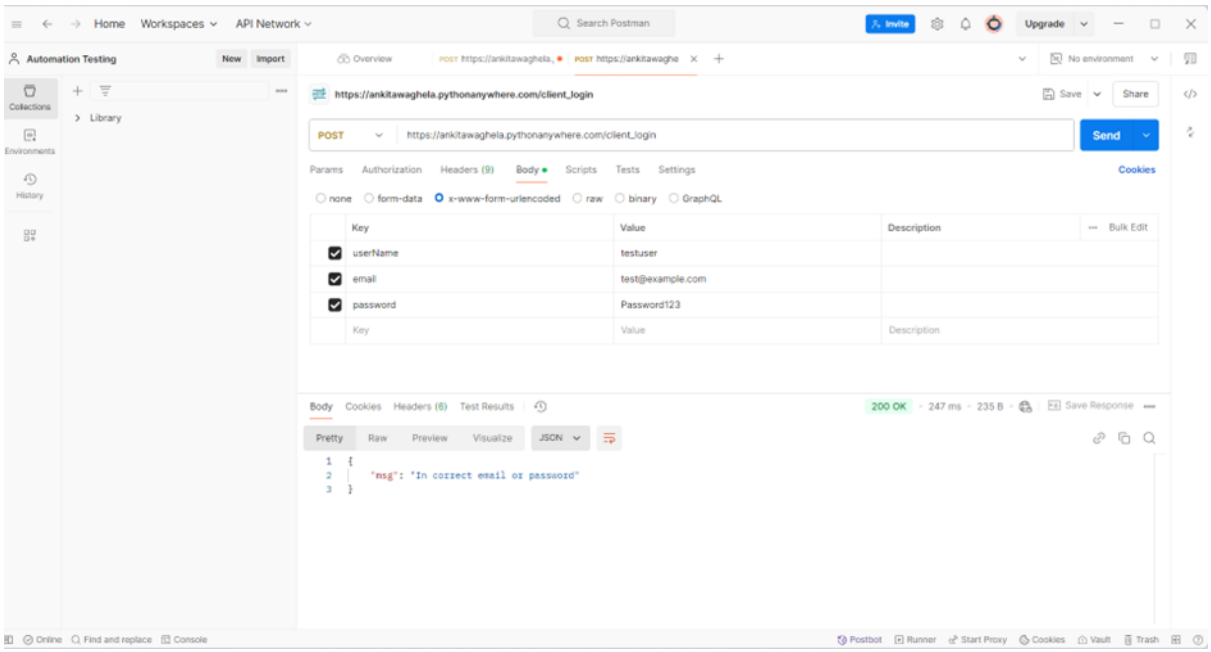

In this vulnerability, the client login endpoint accepts unsanitized user input, making it susceptible to SQL injection attacks. The screenshot shows how an attacker could manipulate the login credentials using SQL injection patterns (OR 1=1) to potentially bypass authentication.

# 2. Insecure Password Storage

Risk Score: High

Passwords are stored in plain text in the database.

Soar_Test/task.py - Line 20, Line 28



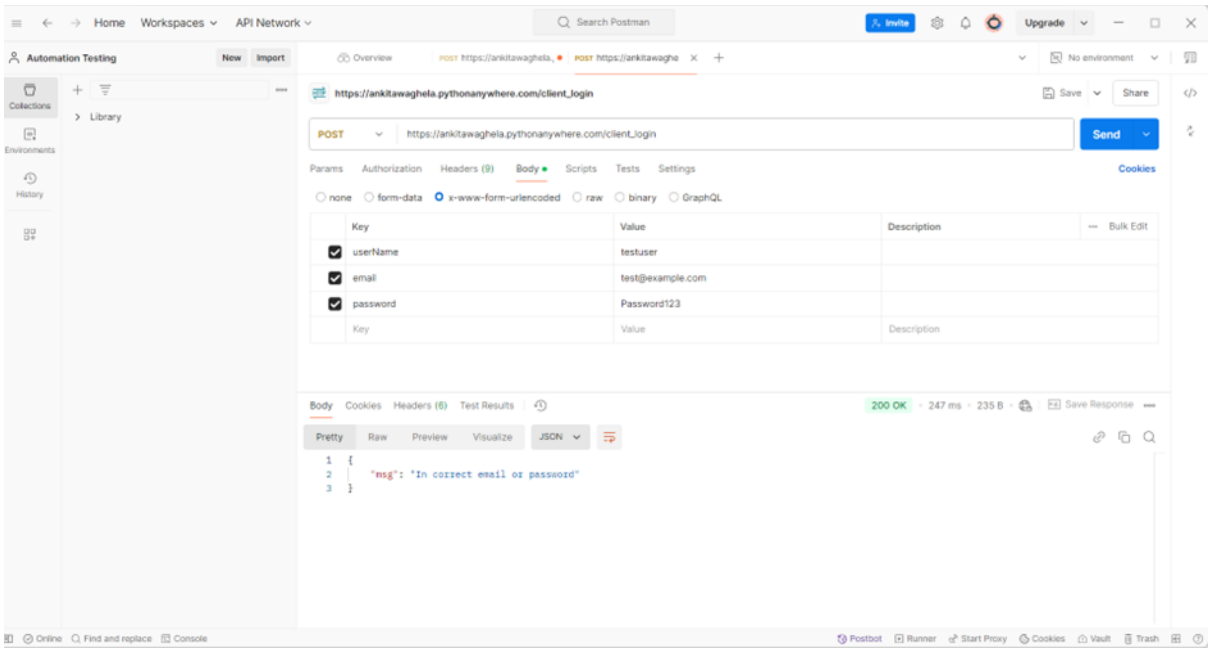

The application stores passwords as plain text in the database, this poses a significant security risk as compromised database access would expose all user passwords directly.

# 3. Weak JWT Secret

Risk Score: High

The JWT secret used for token generation is hardcoded and weak ('123456').

Soar_Test/task.py - Line 20



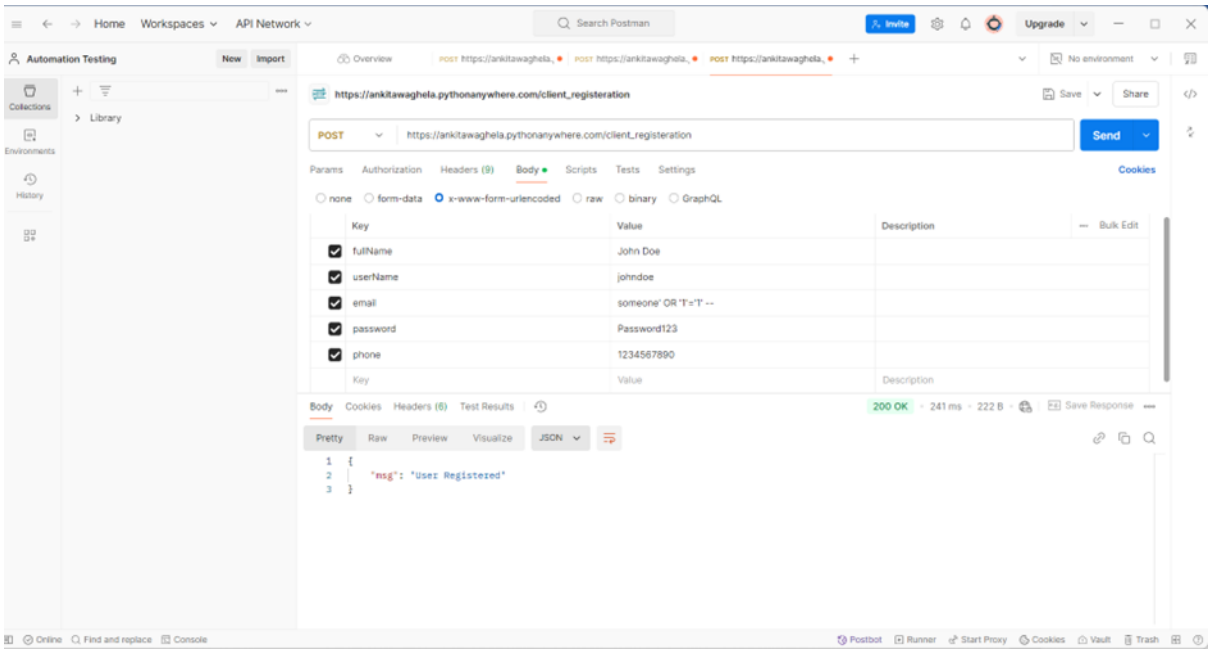

The application uses a predictable and weak JWT secret('123456') for token generation.

# 4. Lack of Input Validation in /client_registeration

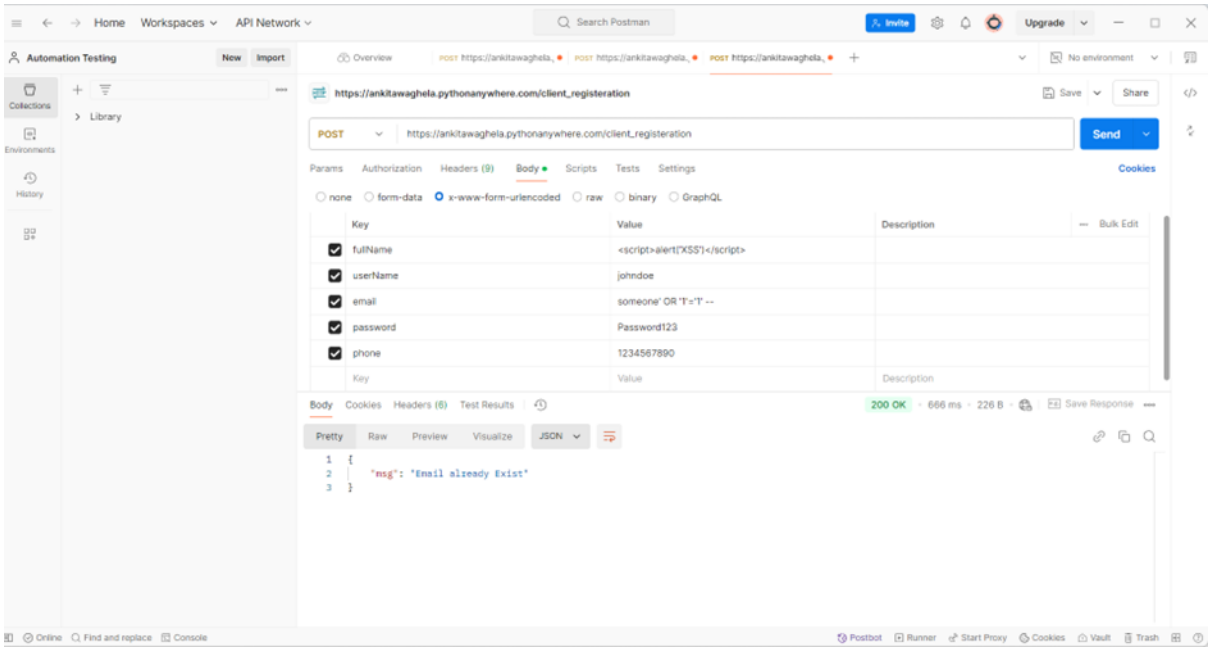
Risk Score: Medium

The **/client_registeration** endpoint lacks proper input validation.

Soar_Test/task.py - Line 62




First test: Successfully registers a user with basic information



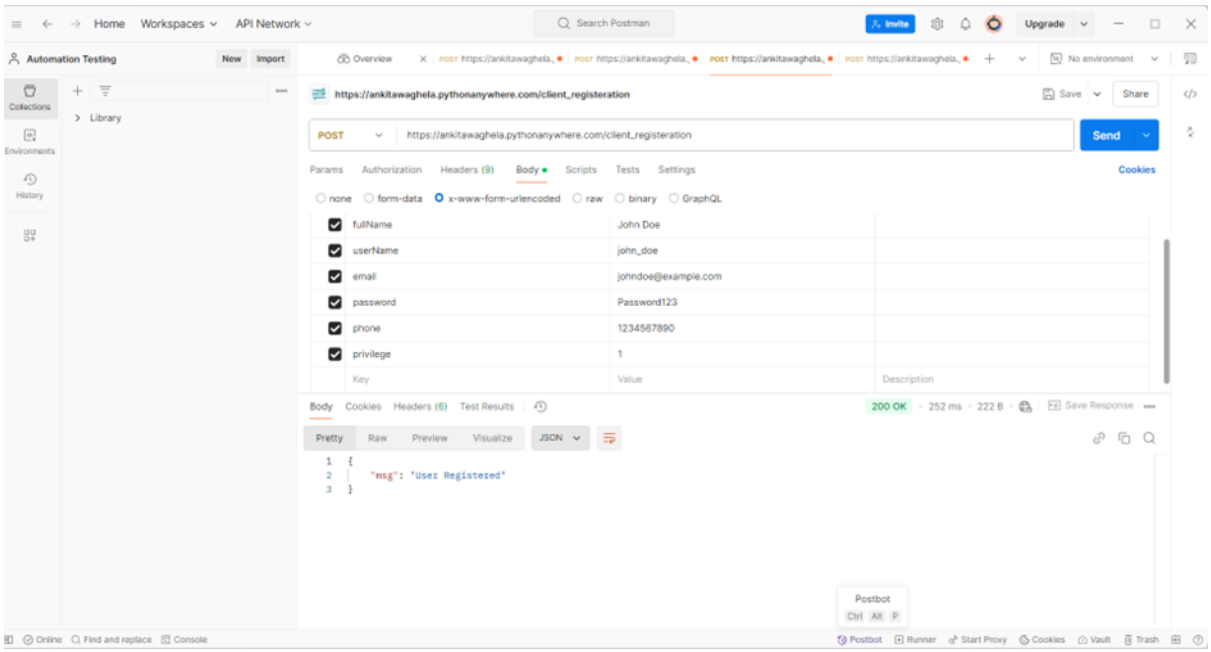

Shows the endpoint accepting potentially malicious script tags in the fullName field, indicating lack of input sanitization.

# 5. Privilege Escalation Vulnerability

Risk Score: High

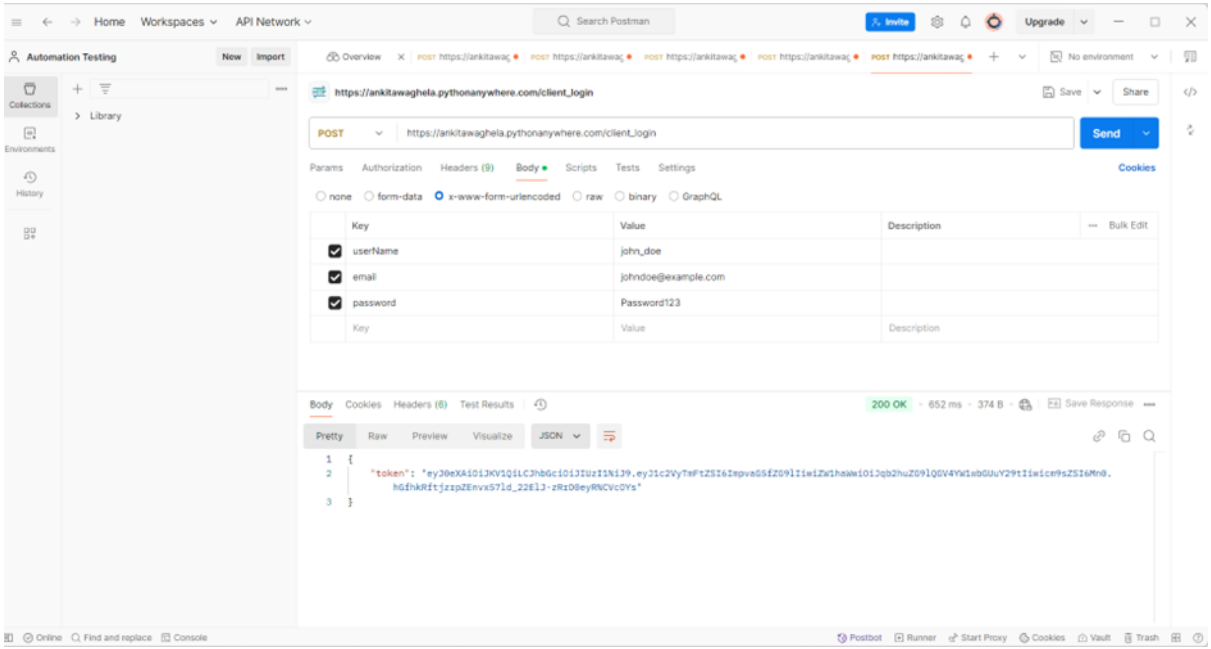
The **privillage** field can be set during user registration without checks.

Soar_Test/task.py - Line 62




This vulnerability allows unauthorized privilege elevation during user registration. The Postman screenshot demonstrates, registration Endpoint: /client_registration accepts a privilege field without proper validation

- User registration with basic information (John Doe)
- Privilege field set to "1" without authorization checks
- Successful 200 OK response received




- JWT tokens are accepted without validating signatures
- Attackers could forge tokens to impersonate users

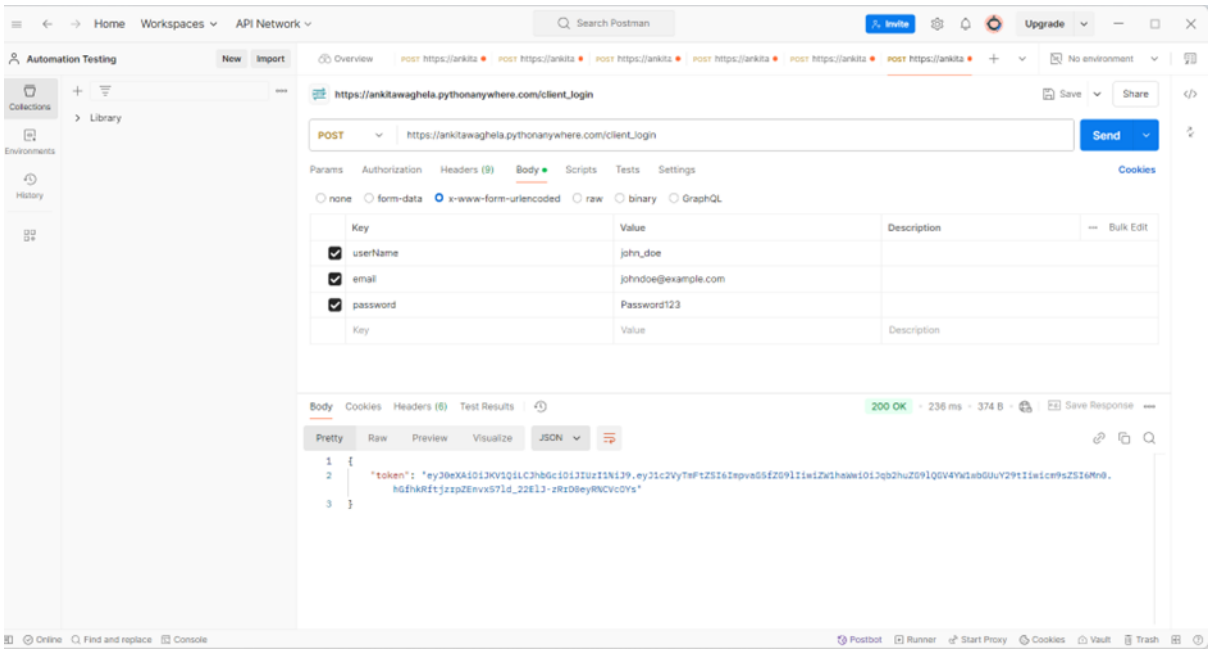
- Compromises entire authentication system

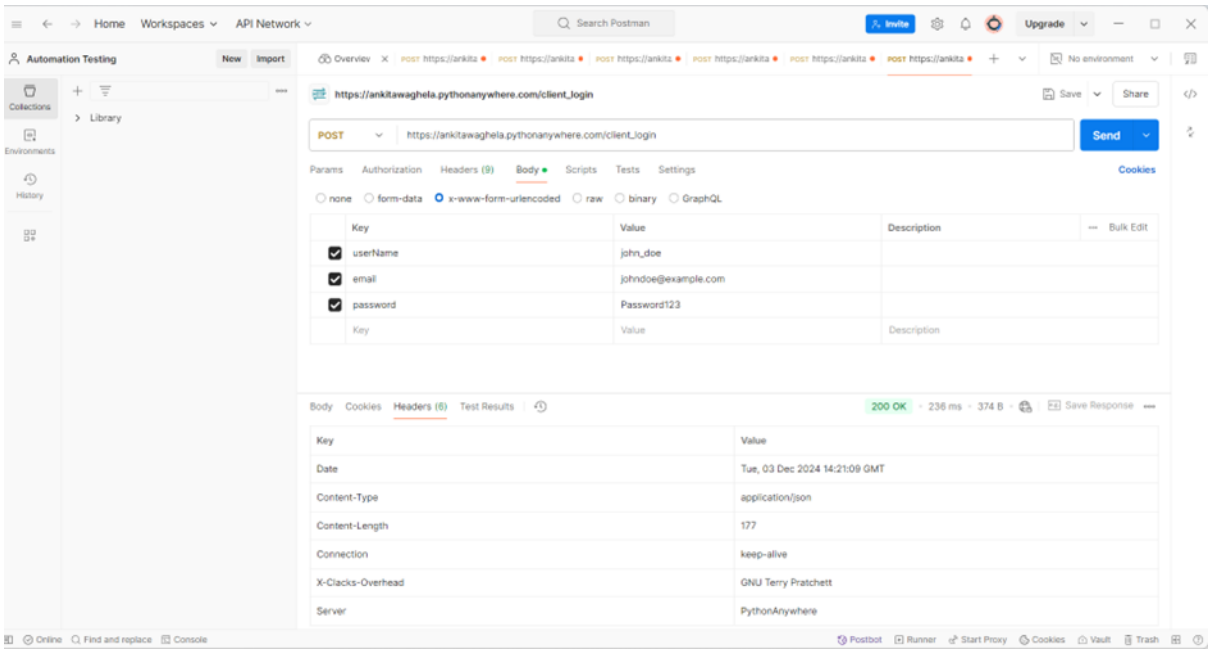
# 6. Insecure JWT Verification

Risk Score: High

The **decodeNoneJwt** function doesn't verify the JWT signature.

Soar_Test/task.py - Line 28




A POST request to /client_login endpoint

Login credentials for user "john_doe"

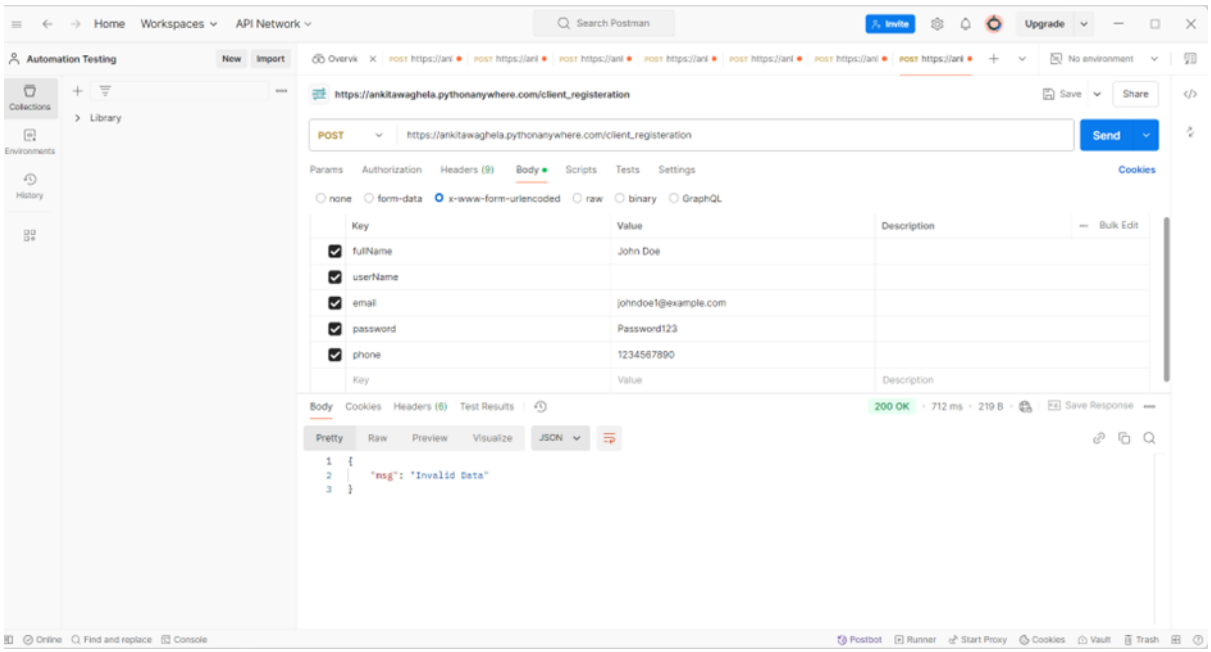
A successful login response with a JWT token

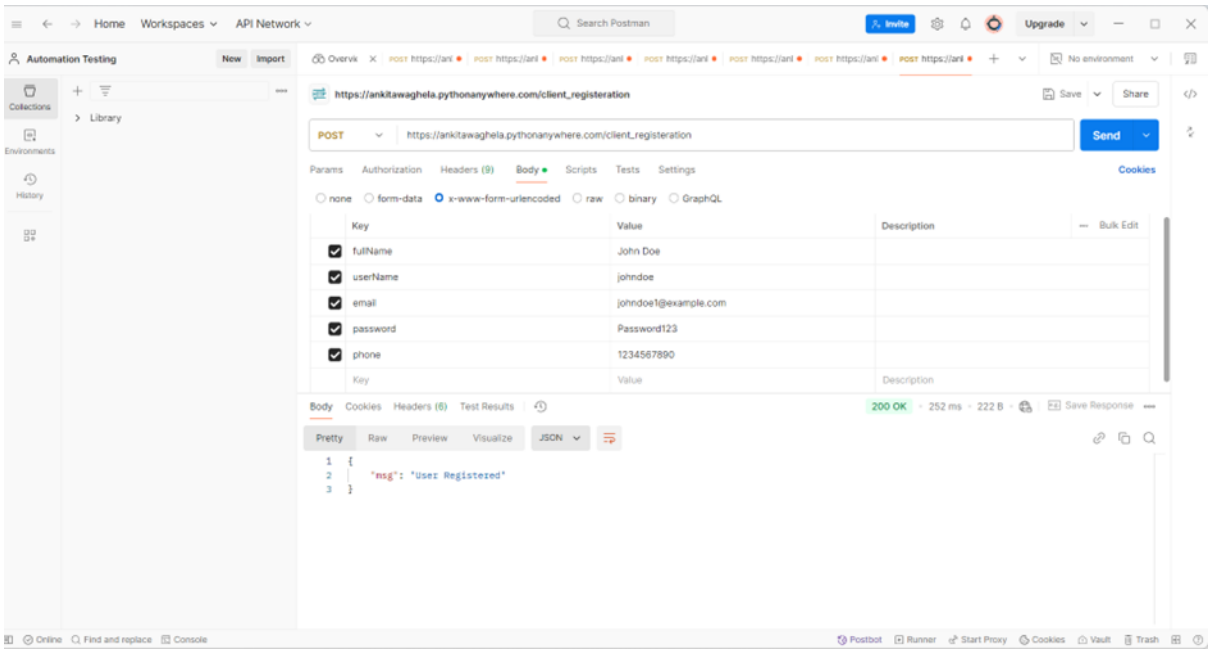
# 7. Information Disclosure in Error Messages

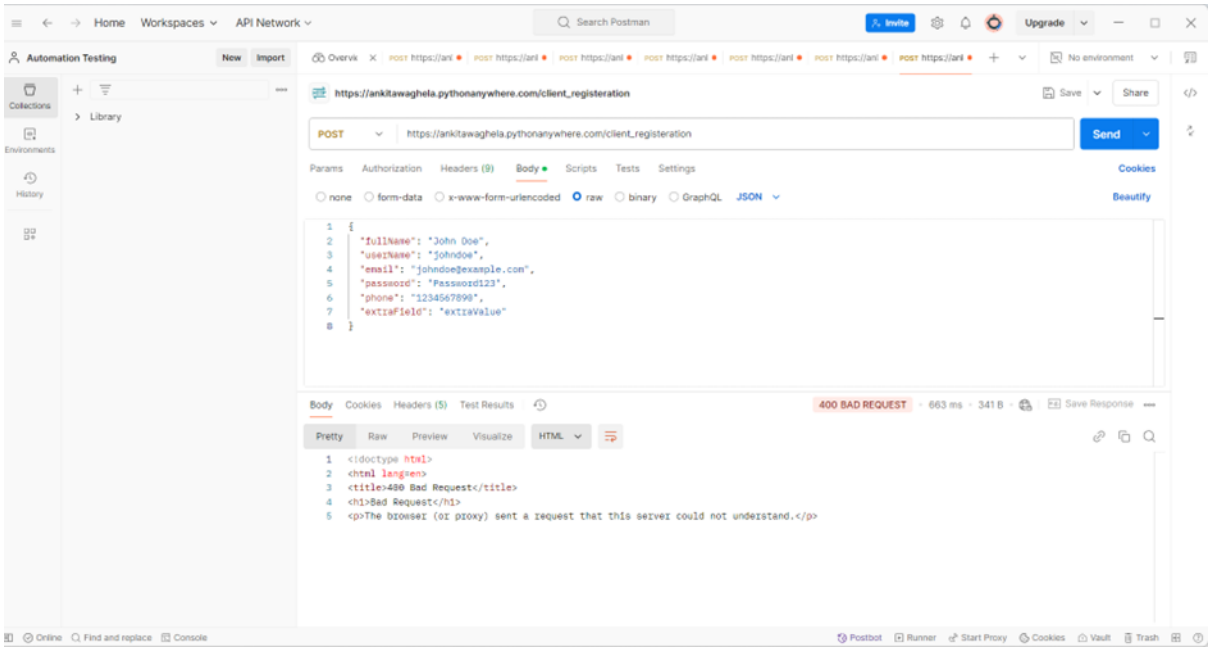
Risk Score: Medium

Detailed error messages could provide useful information to attackers.

Soar_Test/task.py - Line 142






The Postman screenshots demonstrate verbose error handling that could expose sensitive information:

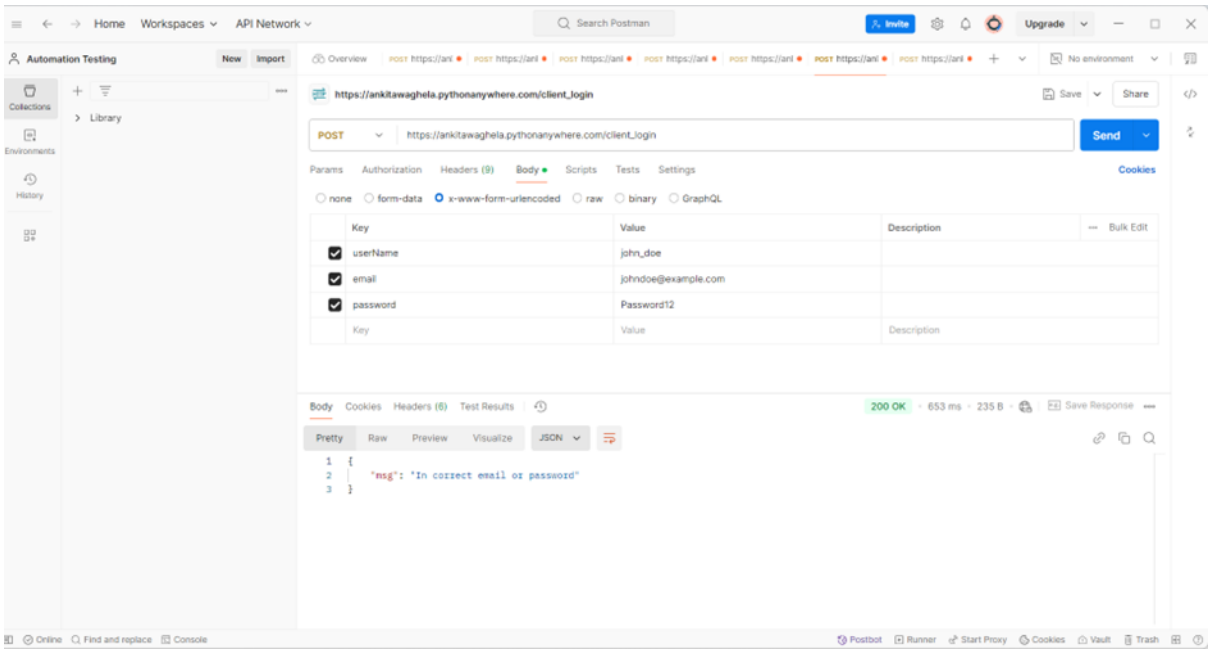
- Test Case 1 - Invalid Data:
  - POST request to /client_registration endpoint
  - User registration attempt with standard information
  - Response shows detailed error message: "Invalid Data"
  - Response time: 771 ms with 200 OK status
- Test Case 2 - Successful Registration:
  - Same endpoint with corrected data
  - Full registration details including:Full Name: John Doe
  - Response shows "User Registered" message
  - Response time: 252 ms with 200 OK status

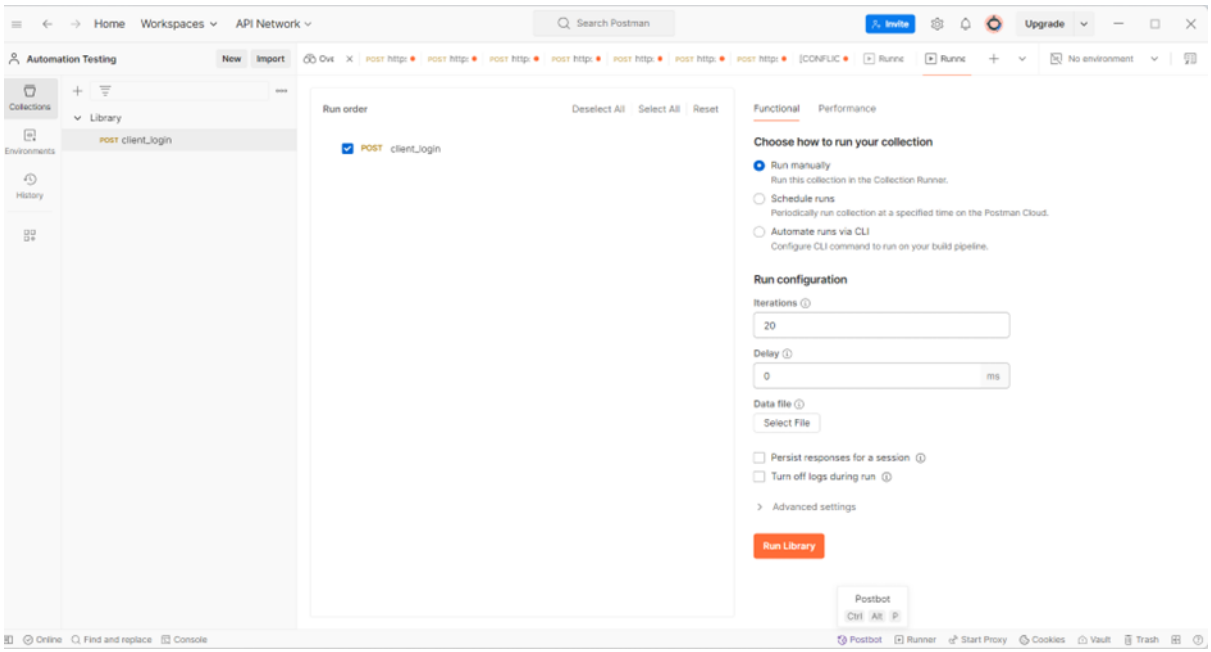
# 8. Lack of Rate Limiting

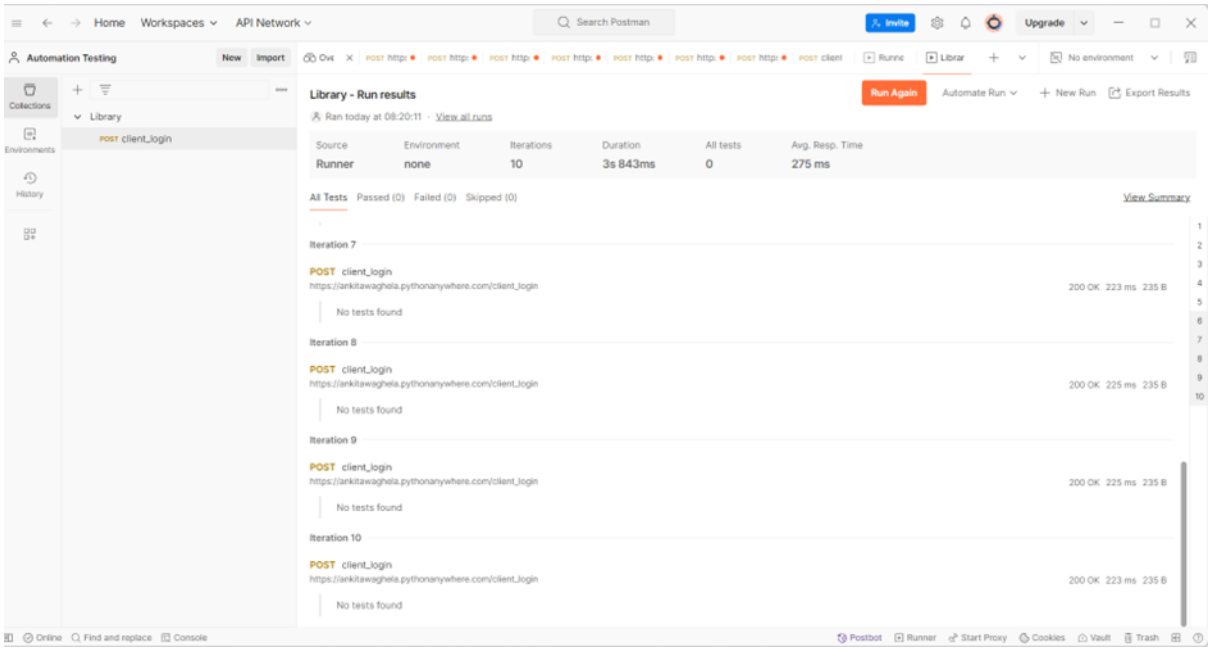
Risk Score: Medium

No rate limiting on login or registration endpoints.

Soar_Test/task.py - Line 72





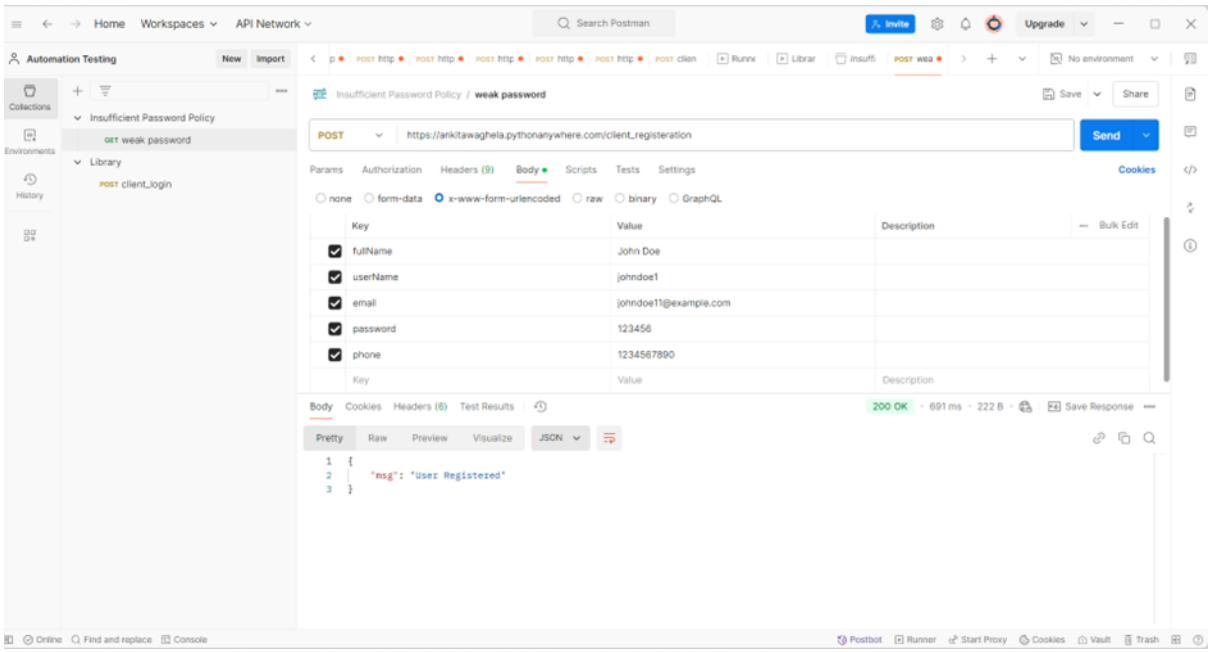

- Multiple rapid login attempts to /client_login endpoint
- Run configuration set to 20 iterations with 0 delay
- No restrictions on number of attempts

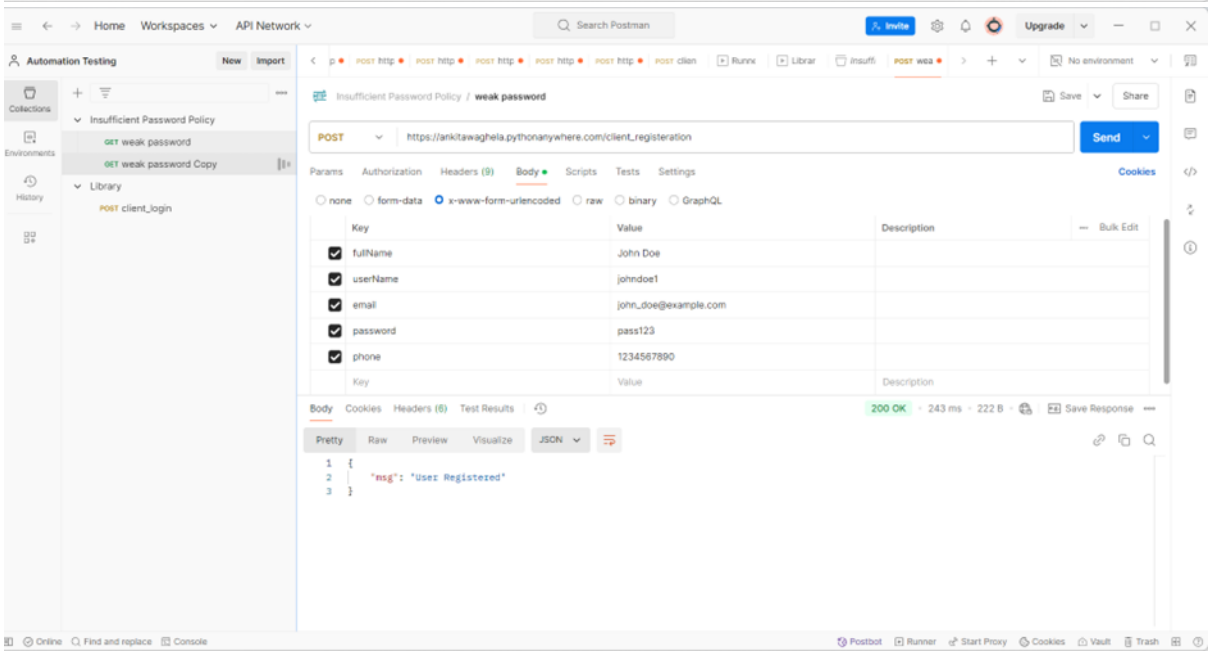
# 9. Insufficient Password Policy

Risk Score: Medium

No password complexity requirements enforced during registration.

Soar_Test/task.py - Line 28




- Registration endpoint accepting weak passwords
- Password "123456" successfully accepted
- No complexity requirements enforced

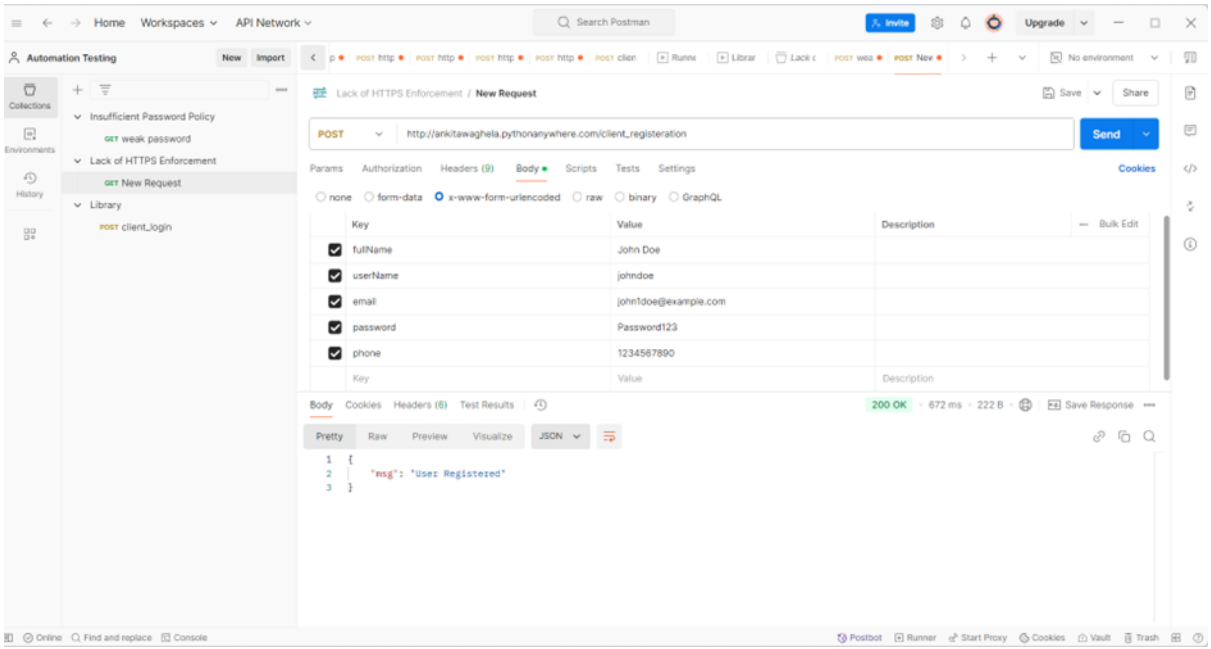
# 10. Lack of HTTPS Enforcement

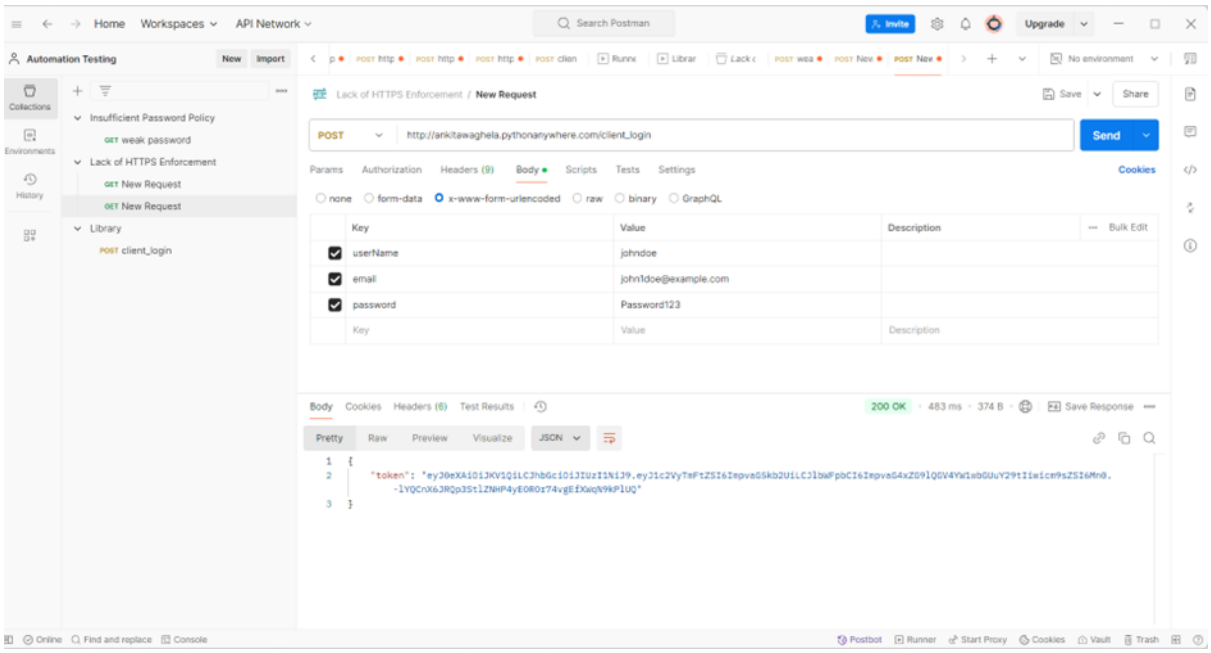
Risk Score: Medium

The application doesn't enforce HTTPS.

You can use this content to create a Word document, adding any necessary formatting or additional details as needed.

Soar_Test/task.py - Line 28




HTTP requests accepted without SSL/TLS

Registration endpoint using unsecured HTTP

Login endpoint accepting plain HTTP connections