

# Logical and security testing

## Contents

<b>Vulnerability Report for Flask Application</b> .....	3
1. SQL Injection in /client_login.....	4
2. Insecure Password Storage .....	5
3. Weak JWT Secret .....	6
4. Lack of Input Validation in /client_registration .....	7
5. Privilege Escalation Vulnerability .....	8
6. Insecure JWT Verification.....	10
7. Information Disclosure in Error Messages .....	11
8. Lack of Rate Limiting .....	13
9. Insufficient Password Policy.....	15
10. Lack of HTTPS Enforcement.....	16

# Vulnerability Report for Flask Application

Steps to run the flask application.

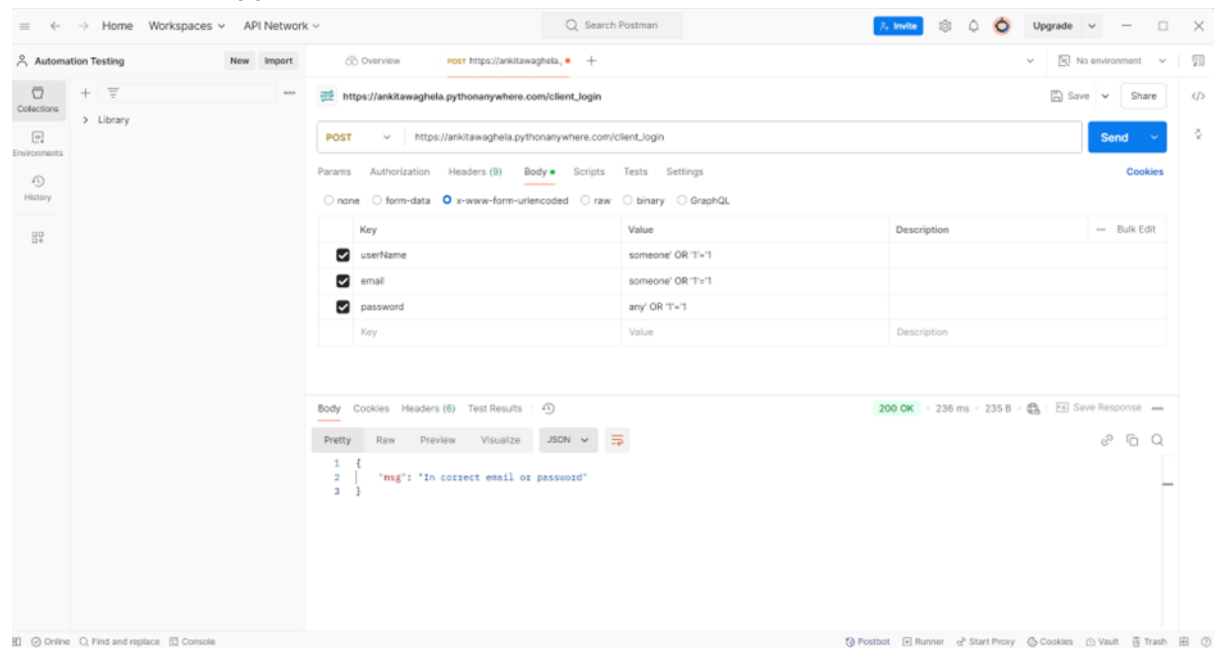
1. Log in to your PythonAnywhere dashboard.
2. Open a new Bash console from the dashboard.
3. Clone the GitHub repository: `git clone https://github.com/Galileo0/Soar_Test.git`
4. Navigate to the cloned directory: `cd Soar_Test`
5. Create a virtual environment: `mkvirtualenv --python=/usr/bin/python3.8 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



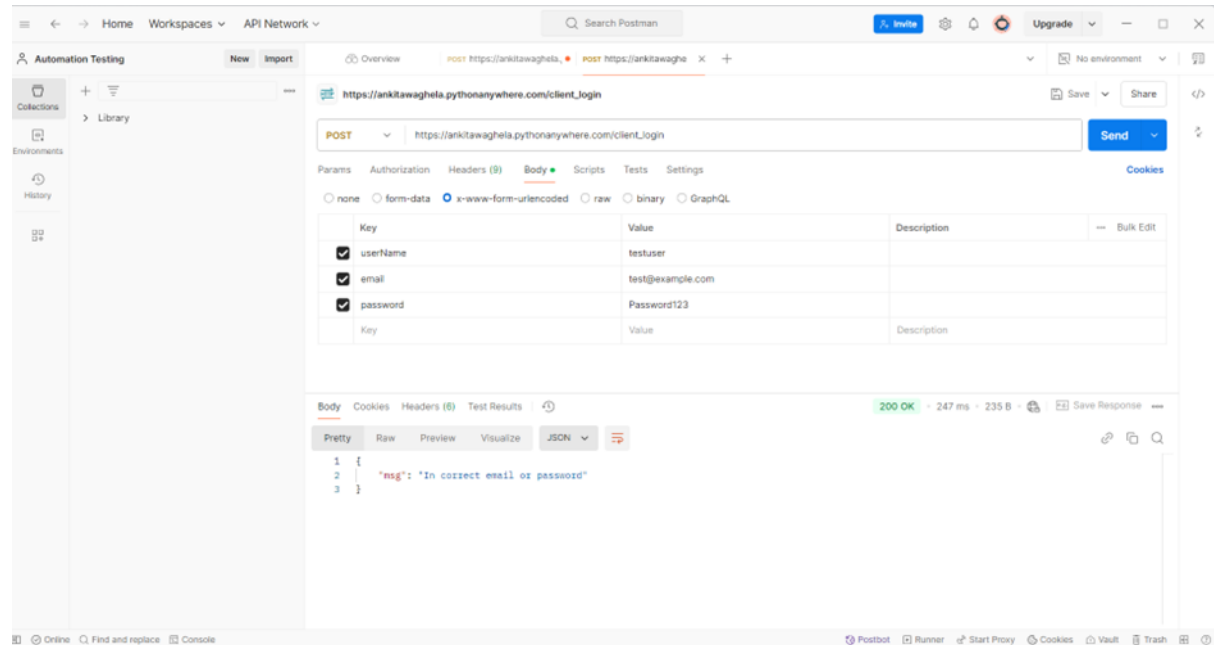
In this vulnerability, the client login endpoint accepts unsanitized user input, making it susceptible to SQL injection attacks. The screenshot shows how an attacker could manipulate the login credentials using SQL injection patterns (OR 1=1) to potentially bypass authentication.

## 2. Insecure Password Storage

Risk Score: High

Passwords are stored in plain text in the database.

[Soar\\_Test/task.py](#) - Line 20, Line 28



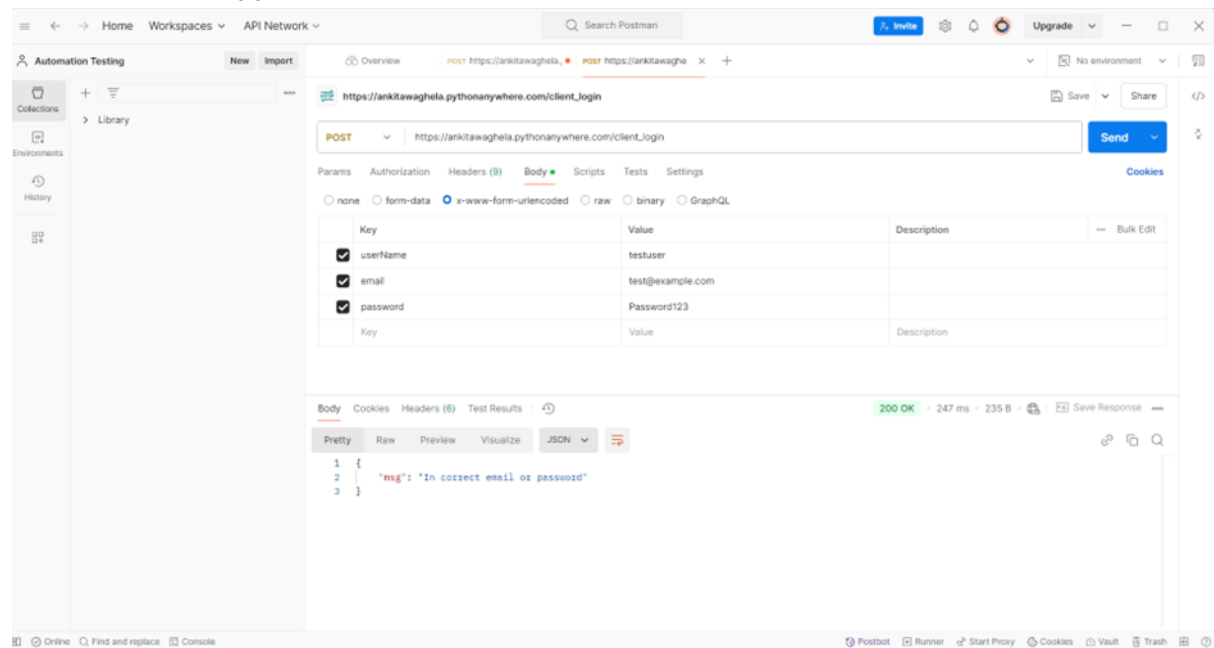
The application stores passwords as plain text in the database, this poses a significant security risk as compromised database access would expose all user passwords directly.

### 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



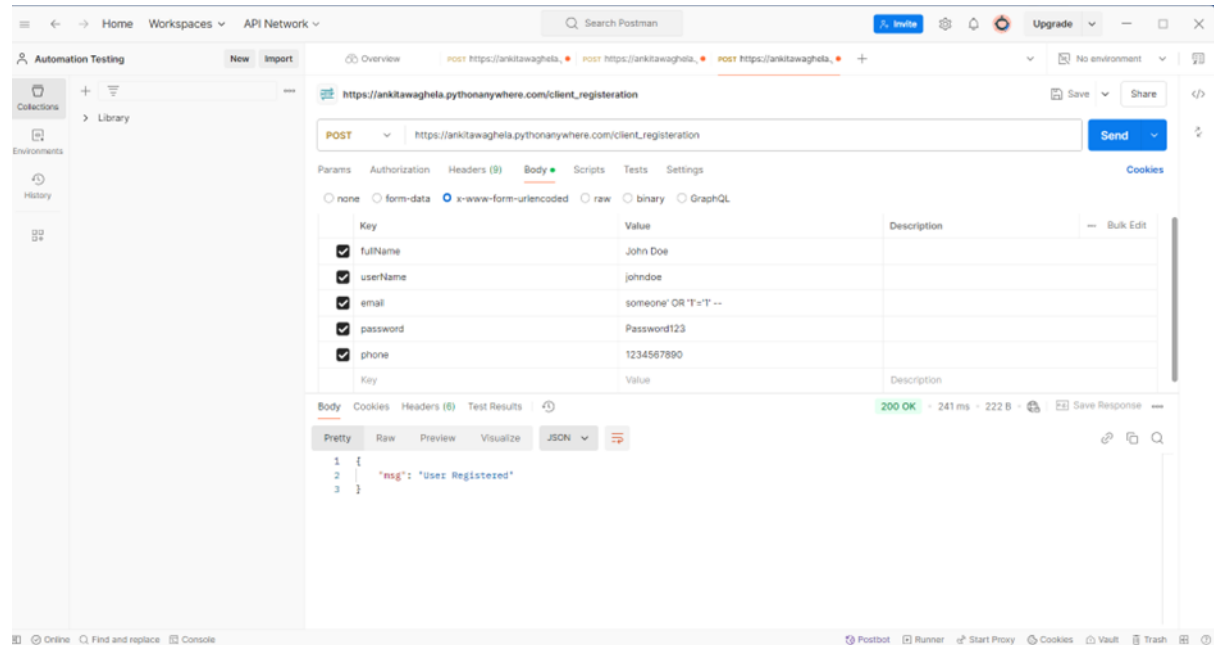
The application uses a predictable and weak JWT secret('123456') for token generation.

## 4. Lack of Input Validation in /client\_registration

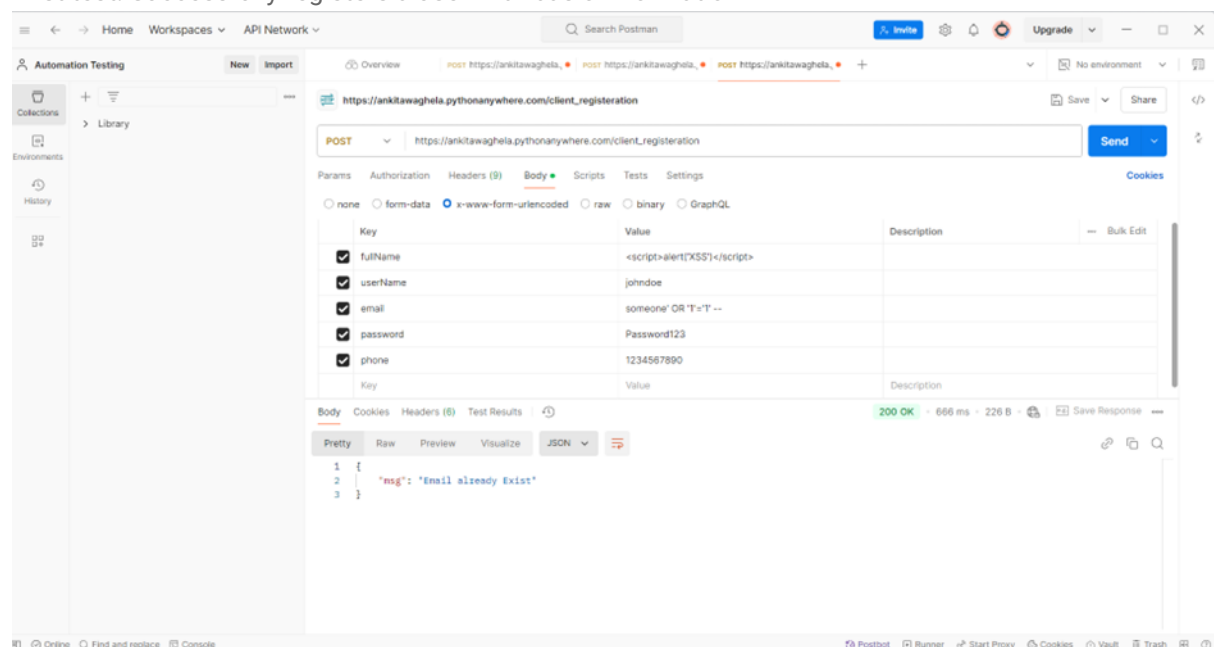
Risk Score: Medium

The **/client\_registration** endpoint lacks proper input validation.

[Soar\\_Test/task.py](#) - Line 62

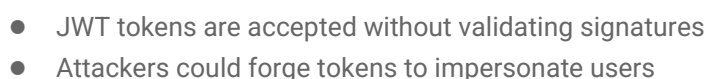
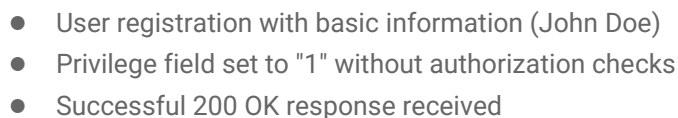


First test: Successfully registers a user with basic information



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

[Soar\\_Test/task.py](#) - Line 62





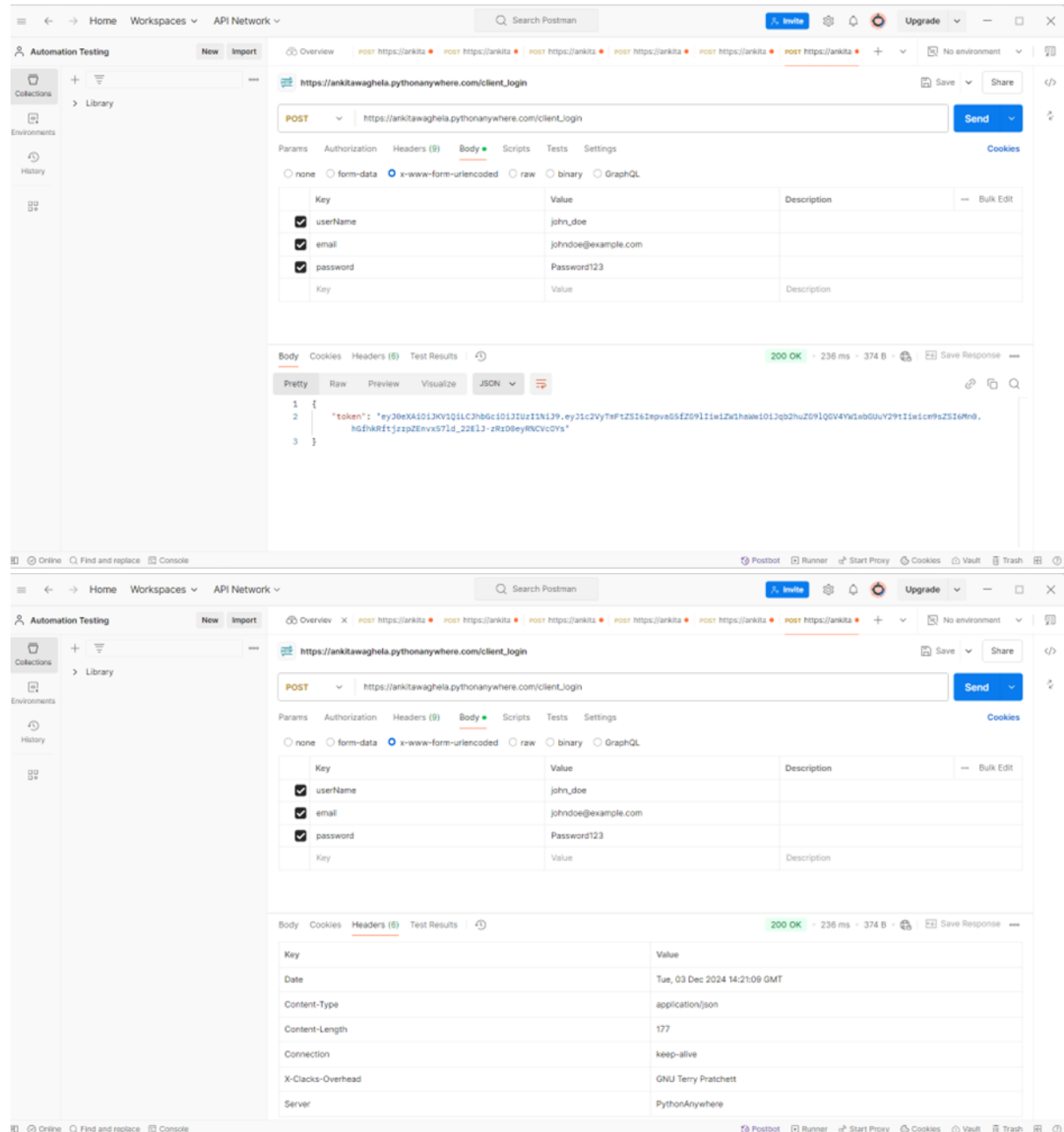
- Compromises entire authentication system

## 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 image displays two screenshots of the Postman application, illustrating the results of an API test for the endpoint `https://ankitawaghela.pythonanywhere.com/client_registration`.

**Top Screenshot (Error Message):**

- Method:** POST
- URL:** `https://ankitawaghela.pythonanywhere.com/client_registration`
- Body (x-www-form-urlencoded):**

Key	Value	Description
fullName	John Doe	
userName		
email	john.doe1@example.com	
password	Password123	
phone	1234567890	
- Response:** 200 OK - 712 ms - 219 B
- Body (JSON):**

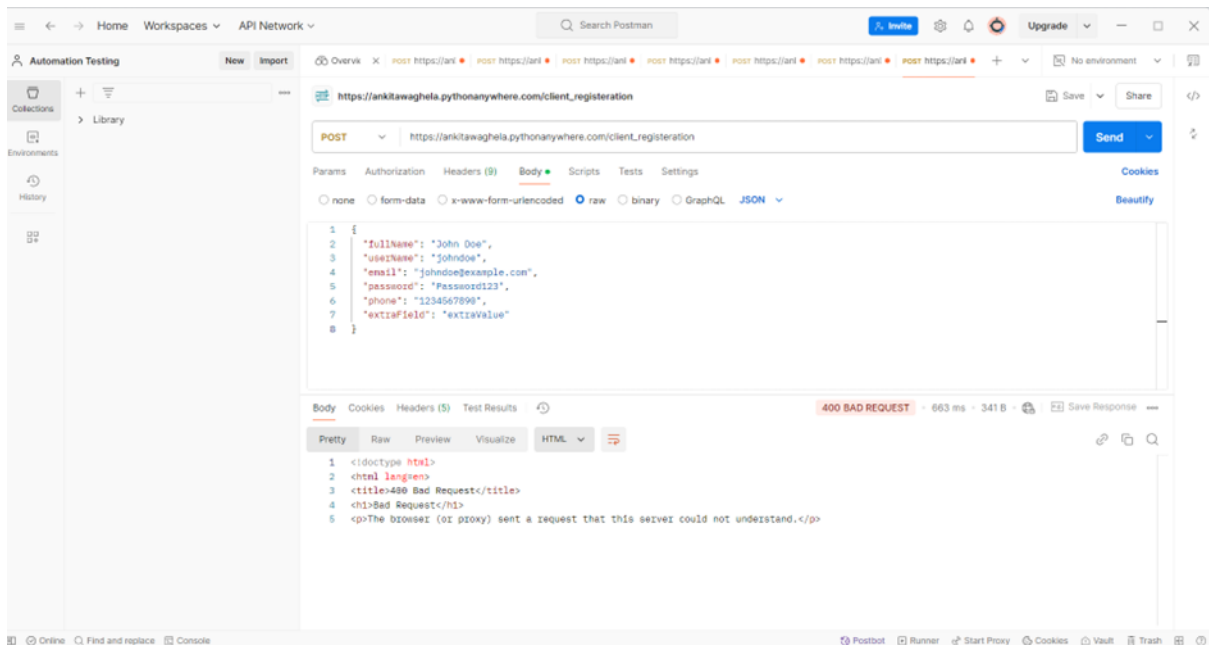
```
1 {
2   "msg": "Invalid Data"
3 }
```

**Bottom Screenshot (Success Message):**

- Method:** POST
- URL:** `https://ankitawaghela.pythonanywhere.com/client_registration`
- Body (x-www-form-urlencoded):**

Key	Value	Description
fullName	John Doe	
userName	john.doe	
email	john.doe1@example.com	
password	Password123	
phone	1234567890	
- Response:** 200 OK - 252 ms - 222 B
- Body (JSON):**

```
1 {
2   "msg": "User Registered"
3 }
```



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

The top screenshot shows the Postman interface with a POST request to `https://ankitawaghela.pythonanywhere.com/client_login`. The request body is a JSON object with the following data:

Key	Value	Description
<input checked="" type="checkbox"/> username	john_doe	
<input checked="" type="checkbox"/> email	john_doe@example.com	
<input checked="" type="checkbox"/> password	Password12	

The response is a 200 OK status with a body of:

```
{  "msg": "In correct email or password"}
```

The bottom screenshot shows the 'Run' configuration for the 'client\_login' collection. The 'Run order' is set to 'POST client\_login'. The 'Run configuration' section includes the following options:

- Choose how to run your collection:**
  - ☒ Run manually: Run this collection in the Collection Runner.
  - ☐ Schedule runs: Periodically run collection at a specified time on the Postman Cloud.
  - ☐ Automate runs via CLI: Configure CLI command to run on your build pipeline.
- Run configuration:**
  - Iterations: 20
  - Delay: 0 ms
  - Data file: Select File
  - ☐ Persist responses for a session
  - ☐ Turn off logs during run
  - Advanced settings

The 'Run Library' button is visible at the bottom right.

The screenshot displays the Postman Automation Testing interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. The left sidebar shows 'Collections', 'Environments', and 'History'. The main area is titled 'Library - Run results' and shows a table of test results for a 'POST client\_login' test. The test was run 10 times, all passing with a 200 OK status. The interface includes a 'Run Again' button and a 'View Summary' link. The table shows the following data for each iteration:

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	10	3s 843ms	0	275 ms

Below the table, the results for each iteration are shown, including the URL 'https://arkitawaghela.pythonanywhere.com/client\_login' and response details like '200 OK', '223 ms', and '235 B'.

- 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

The image displays two screenshots of the Postman API client interface, illustrating a successful registration request with a weak password.

**Top Screenshot:** Shows a POST request to `https://ankitawaghela.pythonanywhere.com/client_registration` with the following body data:

Key	Value	Description
fullName	John Doe	
userName	john doe1	
email	john doe1@example.com	
password	123456	
phone	1234567890	

The response is a 200 OK status with a body of `{ "msg": "User Registered" }`.

**Bottom Screenshot:** Shows the same POST request, but with the password field set to `pass123` instead of `123456`. The response is also a 200 OK status with a body of `{ "msg": "User Registered" }`.

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

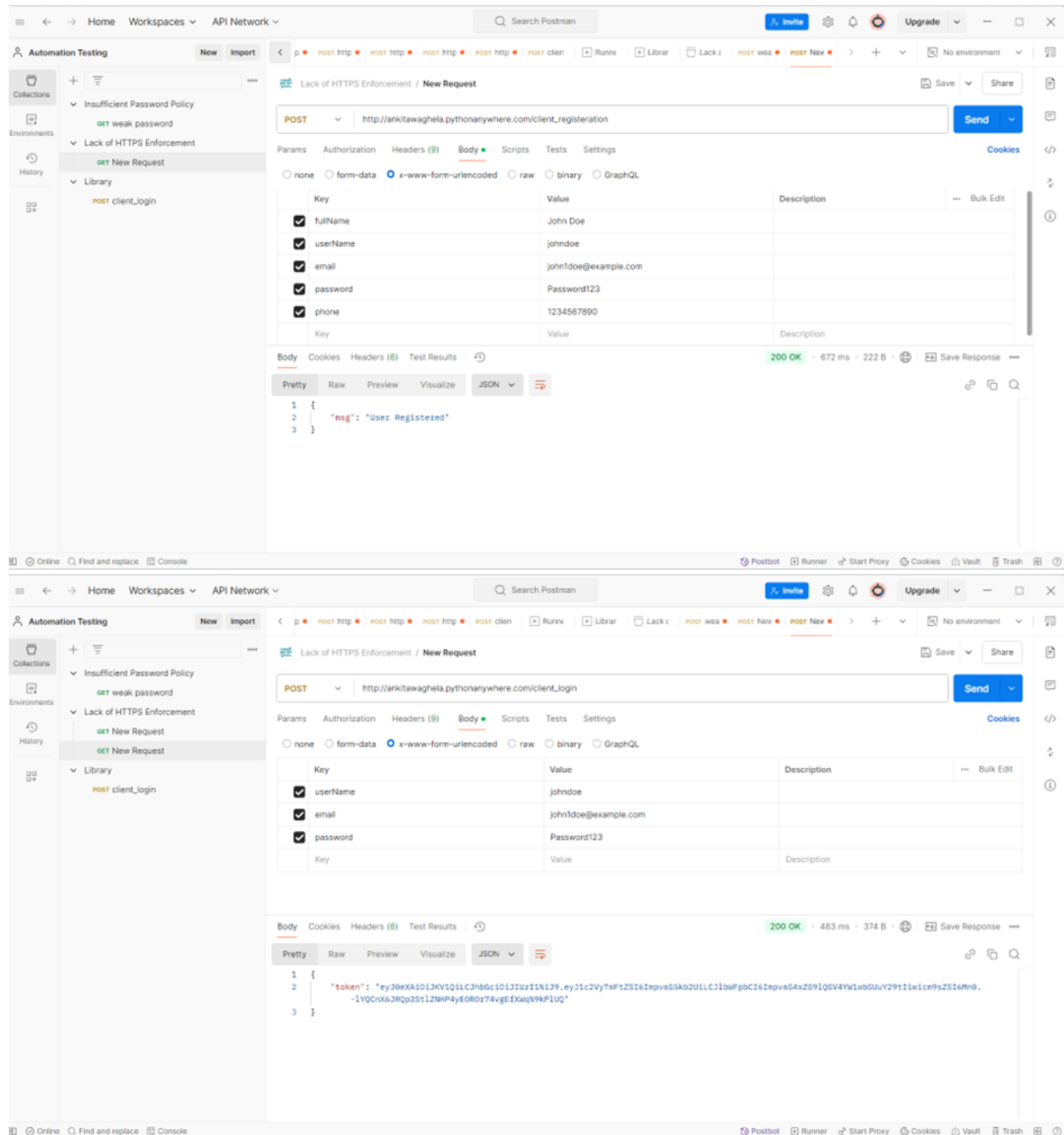
## 10. Lack of HTTPS Enforcement

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