**Task Name: Demo App UI Automation with Robot Framework & API Testing using Python:**

The following tasks were done during the exercise

* Cloned the project from github repository to local and installed the requirements.txt
* Then executed the run script file in terminal ./run.sh to start the server and used the url [*http://0.0.0.0:8080/*](http://0.0.0.0:8080/)to launch the application in web browser.
* Got a general overview on the application’s functionality
* Identified the test cases and test data needs for the application.
* Developed UI automation using Robot Framework and API automation using python.

1. **Robot Framework Automation:**

**Test cases for Demo app UI automation:**

1. Register as a new user using the Register link and providing the needed details.
2. Login to the application with the registered credentials and verify the user data displayed.
3. Logout of the application.
4. Register again using the same credentials to check the error message.
5. Login using incorrect credentials to check the error message.

**Implementation details:**

1. The main suite containing test cases are available in archana8484/Flasky/Test/UserRegistrationSuite.robot
2. I have created a Resources folder (archana8484/Flasky/Test/Resource) for storing the keyword files.

**RegisterAndLogin.robot –** Contains the keywords applicable for the demo app application

**Common.robot -**  Contains generic keywords that could be used for any application

1. Test data for the application is available in
2. archana8484/Flasky/Test/TestData folder

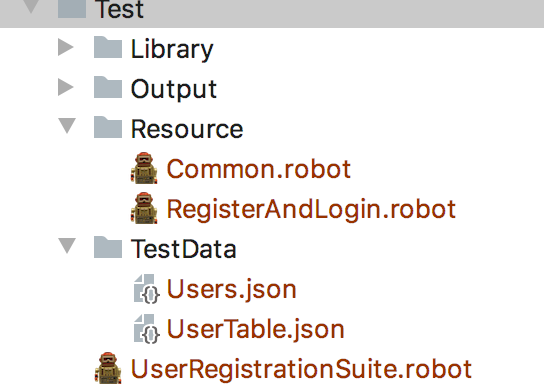
**Users.json** - Has the information of new users to be registered and data for incorrect login.

**UserTable.json** - Expected test results for user view after login

1. Test run Results are available in

archana8484/Flasky/Test/Output folder.

The folder structure looks like this:



**Libraries used:**

Selenium Library

Collections - For working with dictionaries

Operating System – Get file keyword is used to read the contents of the test data json file.

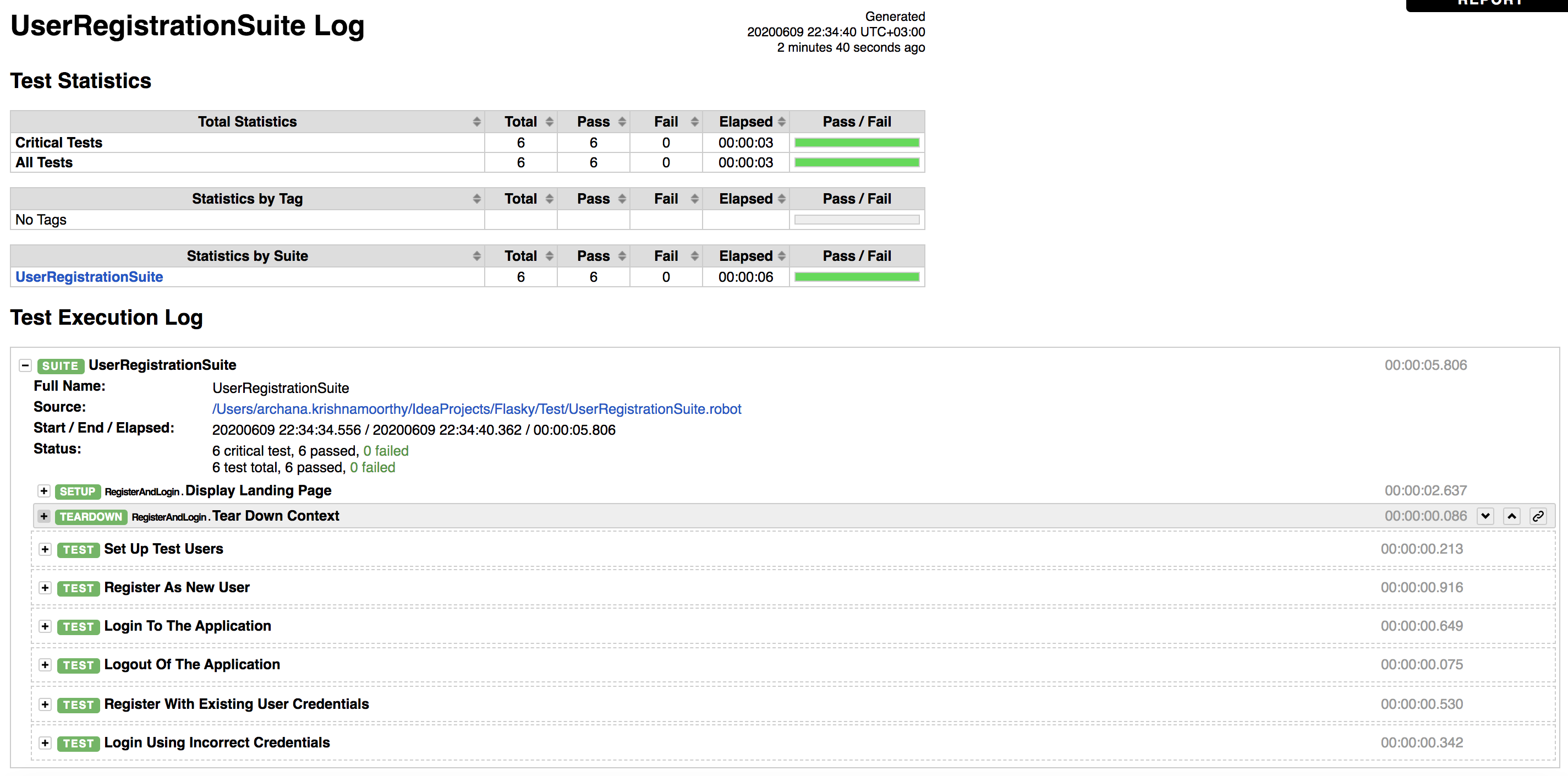
**Running the test:**

I used Intellij Idea IDE for developing robot framework automation. To run the tests, execute the below command in terminal from the correct directory.

**robot -d Output UserRegistrationSuite.robot**

**Test Results:**

Test Results can be found in archana8484/Flasky/Test/Output folder.



1. **API Automation using Python**

**Test cases for API Automation with Python:**

1. Check and retrieve all the users registered to the demo app application.
2. Verify that token for a specific user can be obtained from the API /api/auth/token
3. Check that the user can review his personal information upon getting the token and sending a request to /api/users/{username}.
4. Check that the user can update his personal information upon getting the token and sending a PUT request to /api/users/{username}

**Implementation details:**

1. Python program developed for this API automation is available in archana8484/Flasky/Test/Library/**GetUpdateUsers.py**
2. The libraries imported for this program:

requests

json

1. The main functions created are as below:

* **get\_all\_users():**

Sends a GET request to the /api/users to fetch all the users.

* **get\_user\_info():**

This function calls the auth\_user() function that communicates with the /api/auth/token API to receive authentication token for a given userid/password.

Using the authentication token and userid , the get\_user\_info() function sends a GET request to the API /api/users/{username} to fetch the user’s personal information.

* **update\_user():**

This function calls the auth\_user() function that communicates with the /api/auth/token API to receive authentication token for a given userid/password.

Then the update\_user() function sends a PUT request to the API /api/users/{username} with the header(token + content-type) and json payload(info to be updated) to update the information for a user.

* **main():**

The main() function allows the user to choose any of the above operations and carries out operation based on the selection.

**Running python script:**

/usr/local/bin/python3.7 GetUpdateUsers.py

The command needs to be executed from terminal and from correct directory. Since I had multiple python versions installed, specified the version to be used to execute the file.

Logs are as attached:



1. **Unit Tests for the python code:**

Unit Tests for the python code done in GetUpdateUsers.py are written utilizing the pytest framework.

1. Unit tests are available in archana8484/Flasky/Test/Library/**TestGetUsers.py**
2. Test functions created:

* **test\_get\_users()**

Asserts that the get\_all\_users() function works fine and returns correct result by providing correct expected test results.

* **test\_get\_users\_fail():**

Using negative expected test results (incorrect results), the function fails on comparing between actual results of get\_all\_user() function and expected incorrect results.

* **test\_get\_user\_info():**

Asserts that the get\_user\_info\_internal function works fine and returns correct result by providing correct expected test results.

* **test\_get\_user\_info\_fail()**

Using negative expected test results (incorrect results), the function fails on comparing between actual results of get\_user\_info\_internal function and expected incorrect results.

This is my first experience writing unit tests, so based on the going through a lot of videos and google ☺ , tried to write unit tests for a couple of functions within this given time.

**Logs:**



1. **Issues/bugs found during testing:**

**Issue 1:**

**Description:**

Authenticated as one user (user1), can view information of another user(user2).

**Steps to reproduce:**

1. Send a get request as user1 to the /api/auth/token and obtain a token.
2. Using the token obtained for user1, access the API /api/users/**user2**
3. Access personal information of user2.

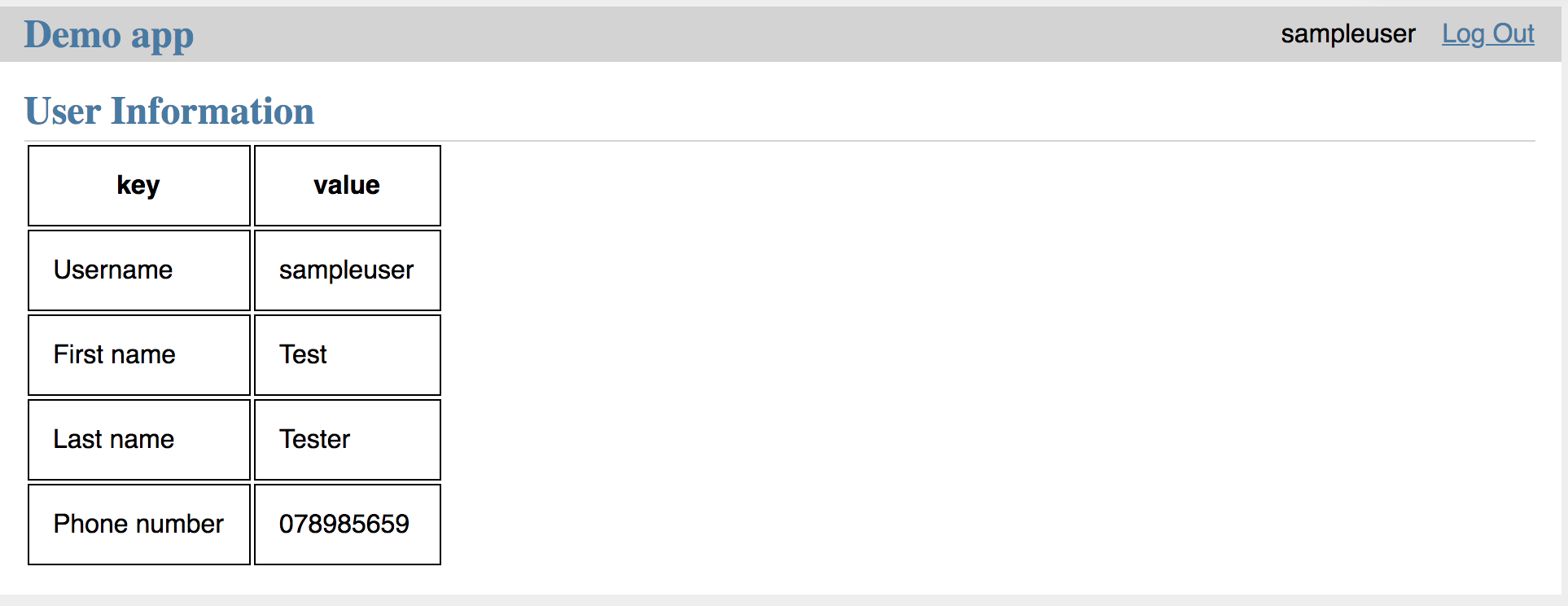
**Screenshots:**

Authenticated as testuser1 and harcoded the api in the program GetUpdateUsers to api/users/sampleuser



Upon running the program, results are fetched for sampleuser with testuser1 token.

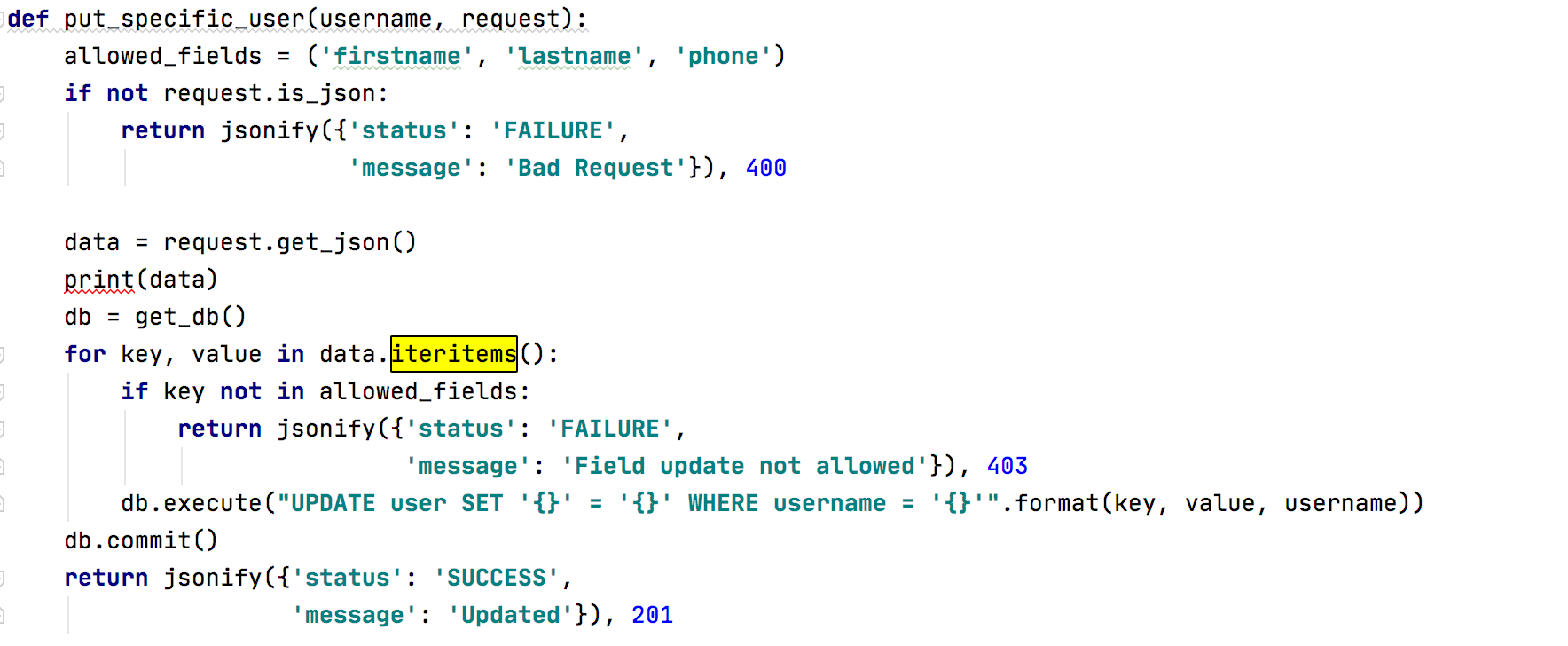




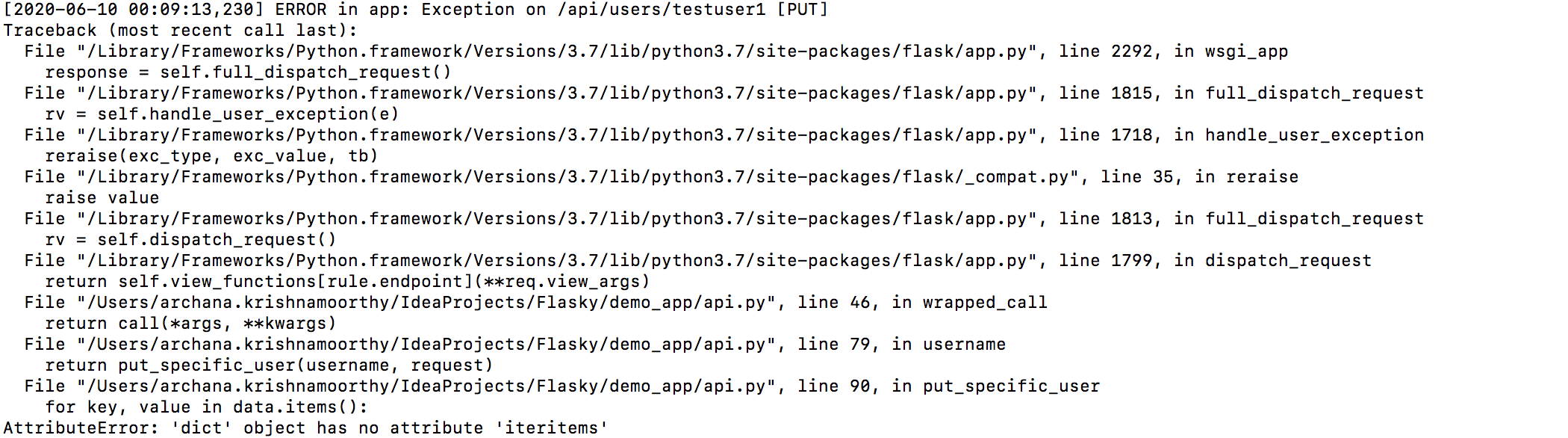
**Issue 2:**

**Description:**

Python 3.7 does not anymore work with **iteritems.** The api.py has the following code in PUT request.

****

Upon trying to PUT a request to api/users/testuser1, following exception occurs.



**Temporary solution:**

Locally, modified the code as **for key,value in data,items()** and the PUT request then goes through.

**Improvement suggestions to the App:**

1. The app must be hosted on a secure service eventhough it’s a demo app.

2. Log in page could be improved further to retrieve forgotten passwords and strong password rules could be enforced.