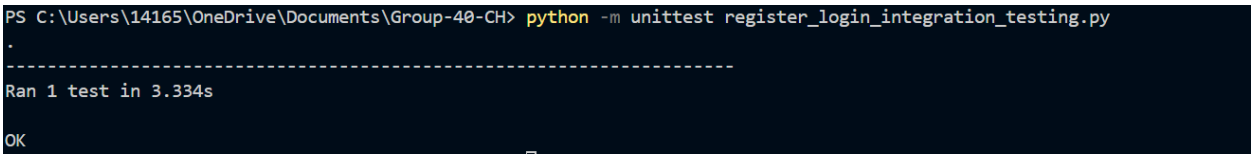
To execute the integration test scripts:

- make sure you are in the root directory of the folder after creating the virtual environment. Run: python -m unittest test-script-name.py

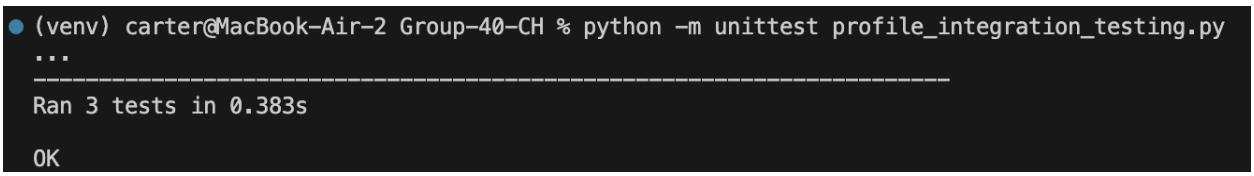
1. Register_login_integration_testing

This test verifies that a new user can successfully register and then log in with the chosen registration details. It works by sending a POST request to /register with valid user details and then asserting that the response redirects to the /login page. It then sends a POST request to the /login page with the same user details and asserts that the response redirects to the /route which is our homepage (flightSearch)



2. profile_integration_testing.py

This test ensures that a logged-in user can successfully update their profile details. It works by sending a POST request to the /profile/{email} route with valid user details (including first name, last name, email, and password). It also checks for error handling, ensuring that the user is shown appropriate error messages when passwords do not match or when required fields are left blank.



3. seat_selection_integration_testing.py

This test verifies that a logged-in user can successfully select seats for a flight. It simulates a user who is logged in (by setting the session with the user's email) and sends a POST request to the /save-seat-selection route with the seat choices. The test checks whether the seat selection is correctly saved in the database for the logged-in user and ensures that the user is redirected to the profile management page after a successful selection. Additionally, the test validates that if the user is not logged in, they are redirected to the login page when trying to access the seat selection page.

```
(venv) carter@MacBook-Air-2 Group-40-CH % python -m unittest seat_selection_integration_testing.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.437s

OK
```

4. login_booking_integration_testing.py
   This script sets up a test client and clears the MongoDB collections before inserting a test user and some booking records. It then tests the integration between the login functionality and the booking history retrieval by using a POST request to the /login route to simulate a user login, sending the test user's email and password as form data. The response is checked to ensure the login was successful by verifying the status code and the presence of the "flightsearch" keyword in the response data. After logging in, the script uses a GET request to access the /booking-history route, checking the response status to ensure the booking history page is successfully retrieved. It then verifies that the booking history content Is correctly displayed by checking for specific booking details in the response data. This approach ensures that authenticated users can successfully view their booking history, demonstrating the integration of the login process with the booking history retrieval.

```
(venv) PS C:\Users\teren\OneDrive - Queen's University\Year Three\CISC 327\Group-40-CH> python -m unittest login_booking_integration_testing.py
.
----------------------------------------------------------------------
Ran 1 test in 0.185s

OK
```