Trevor White (20348221)
Sophia Pagazani (20333850)
Aniss Hamouda (20348807)
CISC 327
25/11/2024

Project Assignment 5

Integration Test

This document provides an overview of the integration tests written for our medicine management system. First, we explain how to run them, then we provide a short explanation and justification for each, along with proof of execution. This discussion is organized into two sections: tests implemented prior to this assignment and others added to fill some logical gaps.

1. How to Run Tests

The first step is to update your virtual environment with the required testing libraries. Additionally, one of our tests uses the `playwright` tool, so you must also install its browsers. Note that you may see an error after the browsers are installed regarding permissions; this can be ignored. For these two steps, run the following:

```
pip install -r requirements.txt
playwright install
```

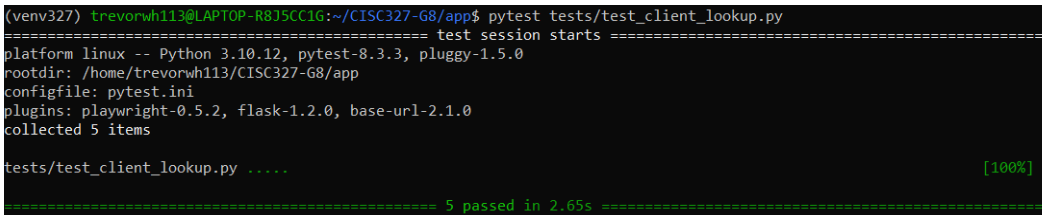
To run the tests, you can simply run the `pytest` command inside the repository. To target specific integration tests, add the path as a parameter. Otherwise, this will execute all the tests except `verify_navigation.py`, which relies on a tool that does not support many platforms. It can only be run if using a Unix/Linux release and through explicit invocation. For instance, if you are in the `tests/` directory, these two commands cover everything:

```
pytest
pytest verify_navigation.py
```

2. Existing Integration Tests

2.1 `test_client_lookup.py`

This test was originally created to test the integration of the frontend of the client search page with the output from the client filtering functionality, ensuring the search parameter is correctly passed and that only the targeted entries are displayed. This case now also verifies the integration of the database, checking that the page successfully fetches the client data.

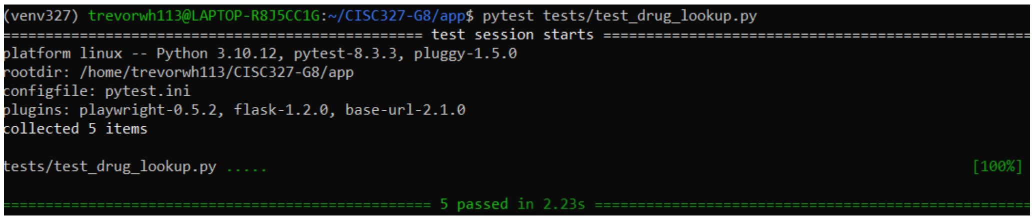


```
(venv327) trevorwh113@LAPTOP-R8J5CC1G:~/CISC327-G8/app$ pytest tests/test_client_lookup.py
================================================ test session starts =================================================
platform linux -- Python 3.10.12, pytest-8.3.3, pluggy-1.5.0
rootdir: /home/trevorwh113/CISC327-G8/app
configfile: pytest.ini
plugins: playwright-0.5.2, flask-1.2.0, base-url-2.1.0
collected 5 items

tests/test_client_lookup.py .....                                                                              [100%]

================================================= 5 passed in 2.65s ==================================================
```

Figure 1: Passing output of `test_client_lookup.py`

2.2 `test_drug_lookup.py`

Much like the client lookup test (2.1), this test case was intended to verify that the HTML page accurately integrates with the drug filtering, sending a search parameter and then displaying the identified entries. Additionally, it tests that the drug database is successfully queried by the drug filtering functionality.

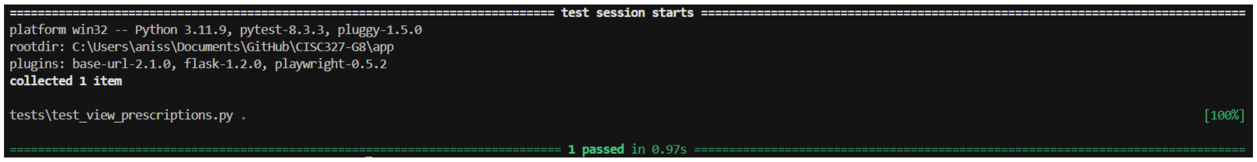


```
(venv327) trevorwh113@LAPTOP-R8J5CC1G:~/CISC327-G8/app$ pytest tests/test_drug_lookup.py
============================================= test session starts ==============================================
platform linux -- Python 3.10.12, pytest-8.3.3, pluggy-1.5.0
rootdir: /home/trevorwh113/CISC327-G8/app
configfile: pytest.ini
plugins: playwright-0.5.2, flask-1.2.0, base-url-2.1.0
collected 5 items

tests/test_drug_lookup.py .....                                                                          [100%]

============================================== 5 passed in 2.23s ===============================================
```

Figure 2: Passing output of `test_drug_lookup.py`

2.3 `test_view_prescription.py`

This is one of our smaller integration tests. It simply ensures that there is connectivity between the client page front end and the database in the backend. This is done by loading the client page and checking that the right prescriptions are present.

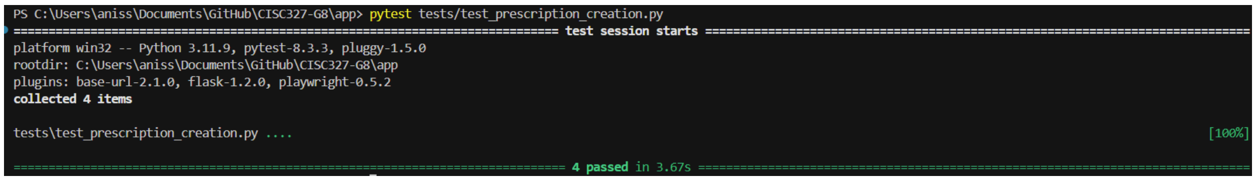


```
=========================================================================== test session starts ===========================================================================
platform win32 -- Python 3.11.9, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\aniss\Documents\GitHub\CISC327-G8\app
plugins: base-url-2.1.0, flask-1.2.0, playwright-0.5.2
collected 1 item

tests\test_view_prescriptions.py .                                                                                                                                  [100%]

============================================================================ 1 passed in 0.97s ============================================================================
```

Figure 3: Passing output of `test_view_prescription.py`

2.4 `test_prescription_creation.py`

This test was intended to ensure that the prescription creation page functioned properly. It tests that the page will load correctly and that the page will not go anywhere if the entered data is invalid. It also ensures that if the entered data is valid, submitting the form brings you to the client page, and that the client page shows the new prescription.



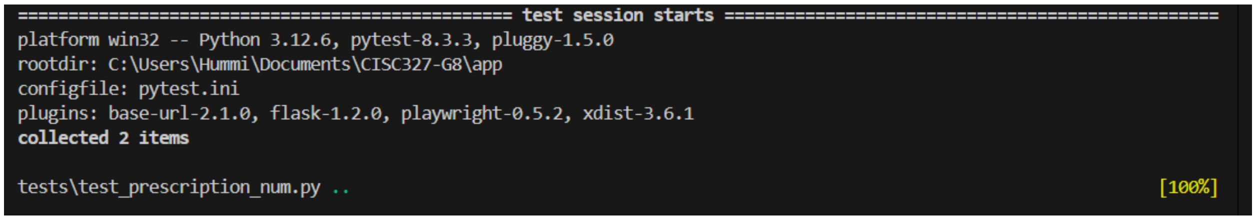
```
PS C:\Users\aniss\Documents\GitHub\CISC327-G8\app> pytest tests/test_prescription_creation.py
=========================================================================== test session starts ===========================================================================
platform win32 -- Python 3.11.9, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\aniss\Documents\GitHub\CISC327-G8\app
plugins: base-url-2.1.0, flask-1.2.0, playwright-0.5.2
collected 4 items

tests\test_prescription_creation.py ....                                                                                                                            [100%]

============================================================================ 4 passed in 3.67s ============================================================================
```

Figure 4: Passing output of `test_prescription_creation.py`

3. New Integration Tests

3.1 `test_prescription_num.py`

To verify that the active prescription number on the client search page was accurately being incremented and decremented, we added this integration test. It first checks if the client exists on the search page and the number of active clients it has, then pushes the changed active prescription lists and old prescription lists to /update-lists (which updates the database through a script). After that has been processed, it then checks again whether the client still exists and has an appropriately updated number.

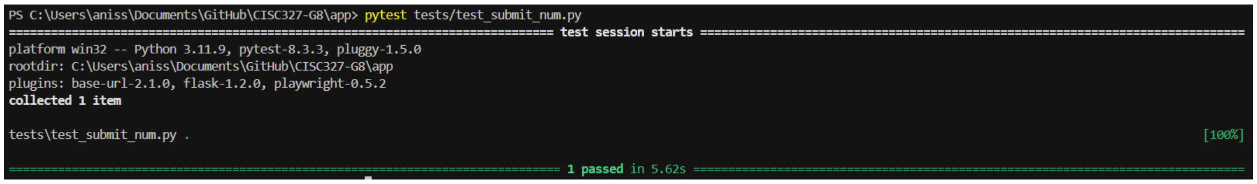


```
================================================ test session starts =================================================
platform win32 -- Python 3.12.6, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\Hummi\Documents\CISC327-G8\app
configfile: pytest.ini
plugins: base-url-2.1.0, flask-1.2.0, playwright-0.5.2, xdist-3.6.1
collected 2 items

tests\test_prescription_num.py ..                                                                               [100%]
```

Figure 5: Passing output of `test_prescription_num.py`

3.2 `test_submit_num.py`

This test case was added to verify that submitting a client prescription is integrated with the client search page. It checks the client search page to check how many prescriptions that client has, then goes to the prescription creation page to add a new prescription to that client. Once it is done, it reloads the client search page to ensure the number of active prescriptions has increased. We added this to ensure there was connectivity throughout the website.

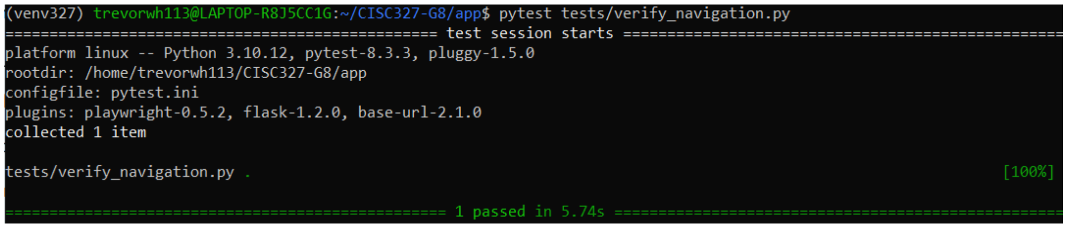


```
PS C:\Users\aniss\Documents\GitHub\CISC327-G8\app> pytest tests/test_submit_num.py
======================================================================= test session starts ========================================================================
platform win32 -- Python 3.11.9, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\aniss\Documents\GitHub\CISC327-G8\app
plugins: base-url-2.1.0, flask-1.2.0, playwright-0.5.2
collected 1 item

tests\test_submit_num.py .                                                                                                                                   [100%]

======================================================================== 1 passed in 5.62s =========================================================================
```

Figure 6: Passing output of `test_submit_num.py`

3.3 `verify_navigation.py`

This final test case was added to verify the integration of the home page and the various navigation mechanisms. Perhaps better considered an end-to-end test, it simulates pressing the buttons to navigate between each page, starting at the home page. We added this case because, although all the frontends are tested individually, there was no system in place to confirm that navigation was functioning as intended.



```
(venv327) trevorwh113@LAPTOP-R8J5CC1G:~/CISC327-G8/app$ pytest tests/verify_navigation.py
============================================= test session starts ==============================================
platform linux -- Python 3.10.12, pytest-8.3.3, pluggy-1.5.0
rootdir: /home/trevorwh113/CISC327-G8/app
configfile: pytest.ini
plugins: playwright-0.5.2, flask-1.2.0, base-url-2.1.0
collected 1 item

tests/verify_navigation.py .                                                                              [100%]

============================================== 1 passed in 5.74s ===============================================
```

Figure 7: Passing output of `verify_navigation.py`