Comp0034-cw2i-TommyLau-bit:

Test code:

```python
import requests
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys


def test_server_running(live_server):
    """Test that the server is running."""
    url = live_server
    response = requests.get(url)
    assert response.status_code == 200, f"Failed to access server at {url}. Status
code: {response.status_code}"

def test_element_value(chrome_driver, live_server):
    """Test the value of an element on the page."""
    chrome_driver.get(live_server)
    element = chrome_driver.find_element(By.ID, "recent-cases-heading")
    element.click()
    highlights_element = chrome_driver.find_element(By.ID, "dashboard-metrics-
heading")
    highlights_value = highlights_element.text
    assert highlights_value == "Dashboard Metrics"

def test_home_page(client):
    response = client.get('/')
    assert response.status_code == 200
    assert b"Recent Cases" in response.data
    assert b"Recent Complaints" in response.data
    assert b"Dashboard Metrics" in response.data


def test_case_detail(client):
    response = client.get('/case/2')
    assert response.status_code == 200
    assert b"Case Details" in response.data
    assert b"Type:" in response.data
    assert b"Status:" in response.data

def test_list_cases(client):
    response = client.get('/cases')
    assert response.status_code == 200
    assert b"FOIA Case" in response.data
```

```python
def test_search_functionality(chrome_driver, live_server):
    """Test the search functionality on the home page."""
    chrome_driver.get(live_server)
    search_input = chrome_driver.find_element(By.ID, "search-input")
    search_input.send_keys("FOIA Case")
    search_input.send_keys(Keys.ENTER)

    # Wait for the search results to load
    WebDriverWait(chrome_driver, 10).until(
        EC.visibility_of_element_located((By.ID, "search-results"))
    )

    search_results_section = chrome_driver.find_element(By.ID, "search-results")
    assert "FOIA Case" in search_results_section.text

def test_navigation_to_case_detail(chrome_driver, live_server):
    """Test navigation to a case detail page from the home page."""
    chrome_driver.get(live_server)
    # Wait for the first 'View Case' link to be clickable
    WebDriverWait(chrome_driver, 10).until(
        EC.element_to_be_clickable((By.CSS_SELECTOR, ".view-case-link"))  # Select
by class
    )

    # Click the first 'View Case' link
    view_case_links = chrome_driver.find_elements(By.CSS_SELECTOR, ".view-case-
link")
    if view_case_links:
        view_case_links[0].click()

    # Ensure the page has navigated to 'Case Details'
    WebDriverWait(chrome_driver, 10).until(
        EC.presence_of_element_located((By.CSS_SELECTOR, "h2"))
    )
    assert "Case Details" in chrome_driver.page_source, "Failed to navigate to Case
Details page."
```

evidence:

```
(.venv) Tommys-MacBook-Pro:comp0034-cw2i-TommyLau-bit tommy$ pytest -v
======================================= test session starts =======================================
platform darwin -- Python 3.8.6, pytest-8.1.1, pluggy-1.4.0 -- /Users/tommy/github-classroom/comp0034-cw2i-TommyLau-bit/.venv/bin/python3
cachedir: .pytest_cache
rootdir: /Users/tommy/github-classroom/comp0034-cw2i-TommyLau-bit
configfile: pyproject.toml
plugins: flask-1.3.0, dash-2.15.0
collected 7 items

tests/test_src_flask.py::test_server_running PASSED                                          [ 14%]
tests/test_src_flask.py::test_element_value PASSED                                           [ 28%]
tests/test_src_flask.py::test_home_page PASSED                                              [ 42%]
tests/test_src_flask.py::test_case_detail PASSED                                            [ 57%]
tests/test_src_flask.py::test_list_cases PASSED                                             [ 71%]
tests/test_src_flask.py::test_search_functionality PASSED                                    [ 85%]
tests/test_src_flask.py::test_navigation_to_case_detail PASSED                               [100%]

======================================== warnings summary =========================================
tests/test_src_flask.py::test_case_detail
  /Users/tommy/github-classroom/comp0034-cw2i-TommyLau-bit/.venv/lib/python3.8/site-packages/flask_sqlalchemy/query.py:30: LegacyAPIWarning:
  The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now ava
ilable as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
    rv = self.get(ident)

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
=================================== 7 passed, 1 warning in 8.47s ===================================
```

Tools and Techniques:

Source code control:

```
(.venv) Tommys-MacBook-Pro:comp0034-cw2i-TommyLau-bit tommy$ git add .
(.venv) Tommys-MacBook-Pro:comp0034-cw2i-TommyLau-bit tommy$ git commit -m "week 8"
[main 192dd8c] week 8
 3 files changed, 78 insertions(+), 3 deletions(-)
 create mode 100644 src/forms.py
 create mode 100644 templates/add_form.html
(.venv) Tommys-MacBook-Pro:comp0034-cw2i-TommyLau-bit tommy$ git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 1.61 KiB | 822.00 KiB/s, done.
Total 7 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/ucl-comp0035/comp0034-cw2i-TommyLau-bit.git
   c329b6a..192dd8c  main -> main
(.venv) Tommys-MacBook-Pro:comp0034-cw2i-TommyLau-bit tommy$
```

TommyLau-bit week 8		192dd8c · 4 minutes ago	4 Commits
migrations	Initial commit		2 days ago
src	week 8		4 minutes ago
templates	week 8		4 minutes ago
tests	Initial commit		2 days ago

URL: https://github.com/ucl-comp0035/comp0034-cw2i-TommyLau-bit.git

Set-up instructions:

COMP0034 Coursework 2 2023/24
COMP0034 Coursework 2 starter repository

MyApp

Introduction

App2 is a dynamic data visualization web application that leverages the REST
API (App1) to provide users with an intuitive interface for analyzing and
interpreting the Metropolitan Police Service's performance in handling Freedom of
Information Requests and Appeals. Our web app offers interactive and informative
dashboards, empowering users to gain valuable insights and make informed decisions
based on the provided data.

Prerequisites
To work with this application, you'll need the following:

1. **GitHub**: You must have access to GitHub for source code control.

2. **Python Environment**: You should use a Python coding environment such as
Visual Studio Code, PyCharm Professional, or a similar tool.

3. **Flask**: This application is built using the Flask Python library to create
the core web application.

4. **SQLite**: SQLite is used as the database for this application.

5. **ChromeDriver**: If you plan to run the Selenium test code, you'll need
ChromeDriver installed and configured.

Installation
To set up and install this application, follow these steps:

1. Clone the GitHub repository:
   ```shell
   git clone 'https://github.com/ucl-comp0035/comp0034-cw2i-TommyLau-bit.git'

2. Change to project directory:
   cd /Users/tommy/github-classroom/comp0034-cw2i-TommyLau-bit

3. Create and activate a virtual environment:
    python -m venv venv
    source venv/bin/activate  # On Windows, use `venv\Scripts\activate`

4. Check pip is the latest versions: pip install --upgrade pip

5. Install dependencies:
    pip install -r requirements.txt
    pip install plotly pandas
    pip install Flask

```
    pip install flask_marshmallow
    pip install Flask-Migrate
    pip install marshmallow-sqlalchemy
    pip install pytest
    pip install -e .
    pip install Flask-WTF
    pip install WTForms-SQLAlchemy
    pip install selenium pytest
    # Install chromedriver too and make sure that it is within the project PATH
    pip install requests
    pip install Flask-Login

6. Run Flask application:
    python app.py or 'flask --app src run --debug'

7. To run tests:
    'pytest' or 'pytest -v'

8. Stop application:
    ctrl + c
```

References

- Use of AI- Format and idea of these two test codes:

```python
def test_search_functionality(chrome_driver, live_server):
    """Test the search functionality on the home page."""
    chrome_driver.get(live_server)
    search_input = chrome_driver.find_element(By.ID, "search-input")
    search_input.send_keys("FOIA Case")
    search_input.send_keys(Keys.ENTER)

    # Wait for the search results to load
    WebDriverWait(chrome_driver, 10).until(
        EC.visibility_of_element_located((By.ID, "search-results"))
    )

    search_results_section = chrome_driver.find_element(By.ID, "search-results")
    assert "FOIA Case" in search_results_section.text

def test_navigation_to_case_detail(chrome_driver, live_server):
    """Test navigation to a case detail page from the home page."""
```

```python
    chrome_driver.get(live_server)
    # Wait for the first 'View Case' link to be clickable
    WebDriverWait(chrome_driver, 10).until(
        EC.element_to_be_clickable((By.CSS_SELECTOR, ".view-case-link"))  # Select
by class
    )

    # Click the first 'View Case' link
    view_case_links = chrome_driver.find_elements(By.CSS_SELECTOR, ".view-case-
link")
    if view_case_links:
        view_case_links[0].click()

    # Ensure the page has navigated to 'Case Details'
    WebDriverWait(chrome_driver, 10).until(
        EC.presence_of_element_located((By.CSS_SELECTOR, "h2"))
    )
    assert "Case Details" in chrome_driver.page_source, "Failed to navigate to Case
Details page."
```