CISC 327 Assignment 4

Radmehr Vafadar | 20421691 | 2025, November 29th

1 E2E Testing Approach

To build the tests quickly and accurately, I used Playwright Codegen to record my interactions while manually clicking through the site (navigating pages, filling forms, and submitting actions). After generating the initial script, I organized it into two clear test cases: one focused on adding a book to the catalog, and the other focused on borrowing a book as a patron. In the add-book test, the script goes to the catalog page, opens the "Add New Book" form, fills in the title, author, ISBN, and total copies, and submits the form. To confirm the action actually worked, I added an assertion that checks the new ISBN appears in the catalog table, which verifies the book was successfully created and displayed to the user. In the borrow-book test, the script returns to the catalog page, enters a valid Patron ID in the borrow field for the book, and clicks the Borrow button. After that, I assert that a borrow confirmation message appears on the page, which confirms that the borrow action completed successfully and the user received feedback from the system.

2 Execution Instructions

Installation

```
git clone https://github.com/RadmehrVafadar/cisc327-library-management-20421691
cd ./cisc327-library-management-20421691
```

Run Website Independently:

```
Python ./app.py
```

Then open http://127.0.0.1:5000 or whatever your local host is

Run e2e tests:

```
Pytest tests/test_e2e.py
```

3 Test Case Summary

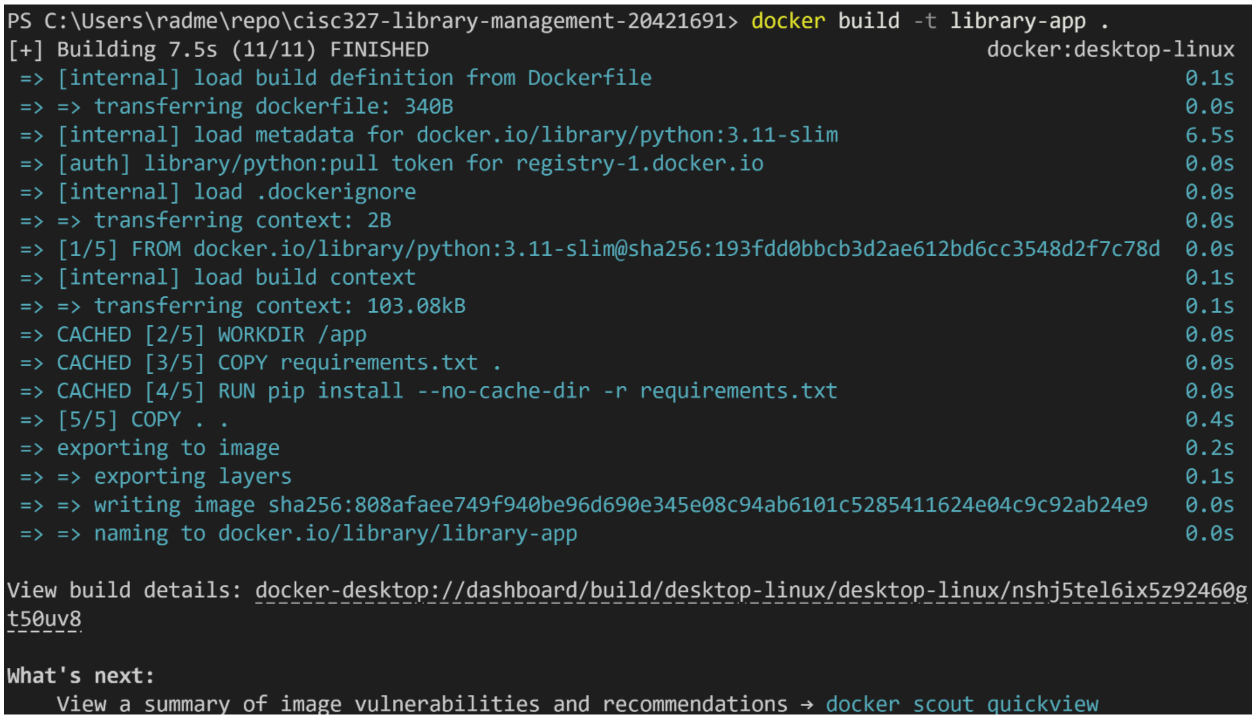
Test Case	Main Actions (User Flow)	Expected Result (What the test checks)
Add New Book to Catalog	Opens /catalog, clicks Add New Book, fills Title, Author, ISBN, Total Copies, then submits the form.	The new book is visible in the catalog table (asserts the ISBN cell appears on the page).
Borrow Book Using Patron ID	Opens /catalog, finds the book row, enters a Patron ID, clicks Borrow.	A borrow confirmation message appears on the page (asserts the success message is visible).

4 Dockerization Process

How to build docker image:

```
docker build -t library-app .
```

output (screenshot below):



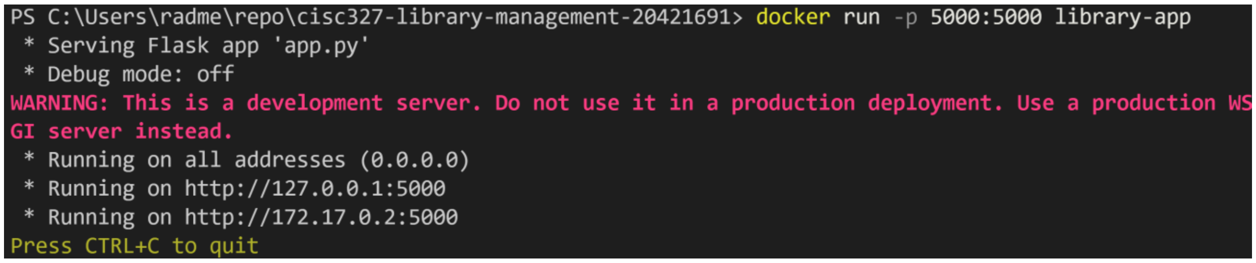
```
PS C:\Users\radme\repo\cisc327-library-management-20421691> docker build -t library-app .
[+] Building 7.5s (11/11) FINISHED                                              docker:desktop-linux
 => [internal] load build definition from Dockerfile                                            0.1s
 => => transferring dockerfile: 340B                                                            0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                             6.5s
 => [auth] library/python:pull token for registry-1.docker.io                                   0.0s
 => [internal] load .dockerignore                                                               0.0s
 => => transferring context: 2B                                                                 0.0s
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:193fdd0bbcb3d2ae612bd6cc3548d2f7c78d  0.0s
 => [internal] load build context                                                               0.1s
 => => transferring context: 103.08kB                                                           0.1s
 => CACHED [2/5] WORKDIR /app                                                                   0.0s
 => CACHED [3/5] COPY requirements.txt .                                                        0.0s
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt                             0.0s
 => [5/5] COPY . .                                                                              0.4s
 => exporting to image                                                                          0.2s
 => => exporting layers                                                                         0.1s
 => => writing image sha256:808afaee749f940be96d690e345e08c94ab6101c5285411624e04c9c92ab24e9   0.0s
 => => naming to docker.io/library/library-app                                                  0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/nshj5tel6ix5z92460g
t50uv8

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

How to build docker container:

```
docker run -p 5000:5000 library-app
```

Then open http://127.0.0.1:5000 or whatever your local host is.

Output (screenshot below):



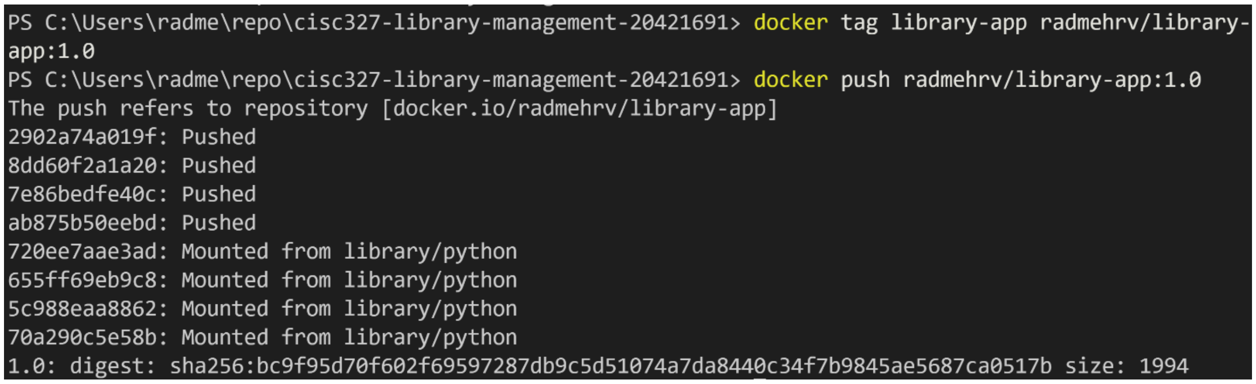
```
PS C:\Users\radme\repo\cisc327-library-management-20421691> docker run -p 5000:5000 library-app
 * Serving Flask app 'app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WS
GI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

5 Ducker Hub Deployment

Tagging the project and pushing it to ducker hub:

```
docker tag library-app radmehrv/library-app:1.0
docker push radmehrv/library-app:1.0
```

Output (screenshot below):

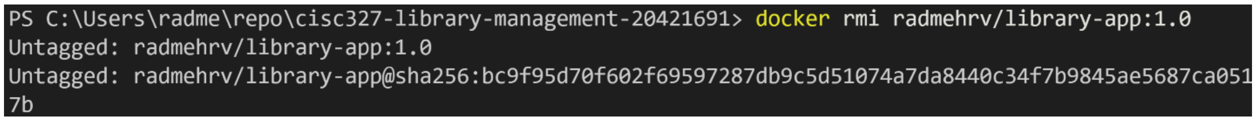


```
PS C:\Users\radme\repo\cisc327-library-management-20421691> docker tag library-app radmehrv/library-
app:1.0
PS C:\Users\radme\repo\cisc327-library-management-20421691> docker push radmehrv/library-app:1.0
The push refers to repository [docker.io/radmehrv/library-app]
2902a74a019f: Pushed
8dd60f2a1a20: Pushed
7e86bedfe40c: Pushed
ab875b50eebd: Pushed
720ee7aae3ad: Mounted from library/python
655ff69eb9c8: Mounted from library/python
5c988eaa8862: Mounted from library/python
70a290c5e58b: Mounted from library/python
1.0: digest: sha256:bc9f95d70f602f69597287db9c5d51074a7da8440c34f7b9845ae5687ca0517b size: 1994
```

Deleting the local image:

```
docker rmi radmehrv/library-app:1.0
```

Output (screenshot below):

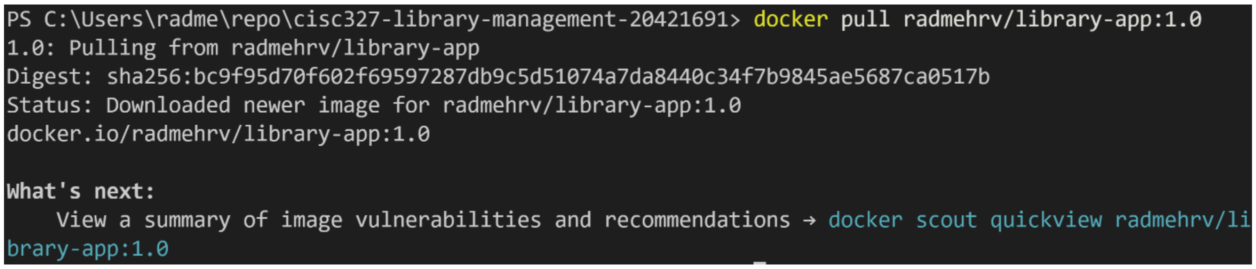


```
PS C:\Users\radme\repo\cisc327-library-management-20421691> docker rmi radmehrv/library-app:1.0
Untagged: radmehrv/library-app:1.0
Untagged: radmehrv/library-app@sha256:bc9f95d70f602f69597287db9c5d51074a7da8440c34f7b9845ae5687ca051
7b
```

Re-pulling it from Docker Hub:

```
docker pull radmehrv/library-app:1.0
```

Output (screenshot below):

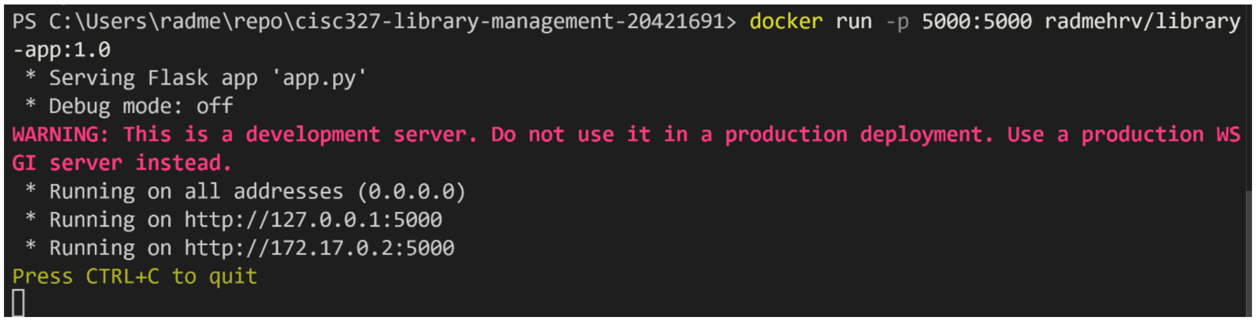


```
PS C:\Users\radme\repo\cisc327-library-management-20421691> docker pull radmehrv/library-app:1.0
1.0: Pulling from radmehrv/library-app
Digest: sha256:bc9f95d70f602f69597287db9c5d51074a7da8440c34f7b9845ae5687ca0517b
Status: Downloaded newer image for radmehrv/library-app:1.0
docker.io/radmehrv/library-app:1.0

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview radmehrv/li
brary-app:1.0
```

Running it from Docker Hub:

```
docker run -p 5000:5000 library-app
```

Output (screenshot below):



```
PS C:\Users\radme\repo\cisc327-library-management-20421691> docker run -p 5000:5000 radmehrv/library
-app:1.0
 * Serving Flask app 'app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WS
GI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

6 CHALLENGES AND REFLECTIONS

One challenge was the time it takes to wire E2E tests by hand, especially choosing selectors, filling out each interaction step-by-step in code, and repeatedly running the tests to confirm the flow is correct. Using Playwright Codegen helped a lot here because it generated reliable starter code from real browser actions, and I could focus my effort on organizing the script into clean test cases and adding meaningful assertions to verify the results. (see below image for reference)

Another issue I ran into was test data stability across reruns. Since the add-book test uses a fixed ISBN, rerunning the suite against a database that already contains that ISBN can cause the “add book” step to fail due to uniqueness constraints. This made it clear that E2E tests are easiest to maintain when they either (1) generate unique data per run, or (2) reset/clean up the test database between runs, so each run starts from a predictable state.

Overall, this assignment showed me the value of E2E testing as a “whole system” check. Compared to unit tests, these tests confirm that routing, templates, form submissions, and user-facing success messages all work together the way a real user experiences them, which is exactly what the assignment is asking for.

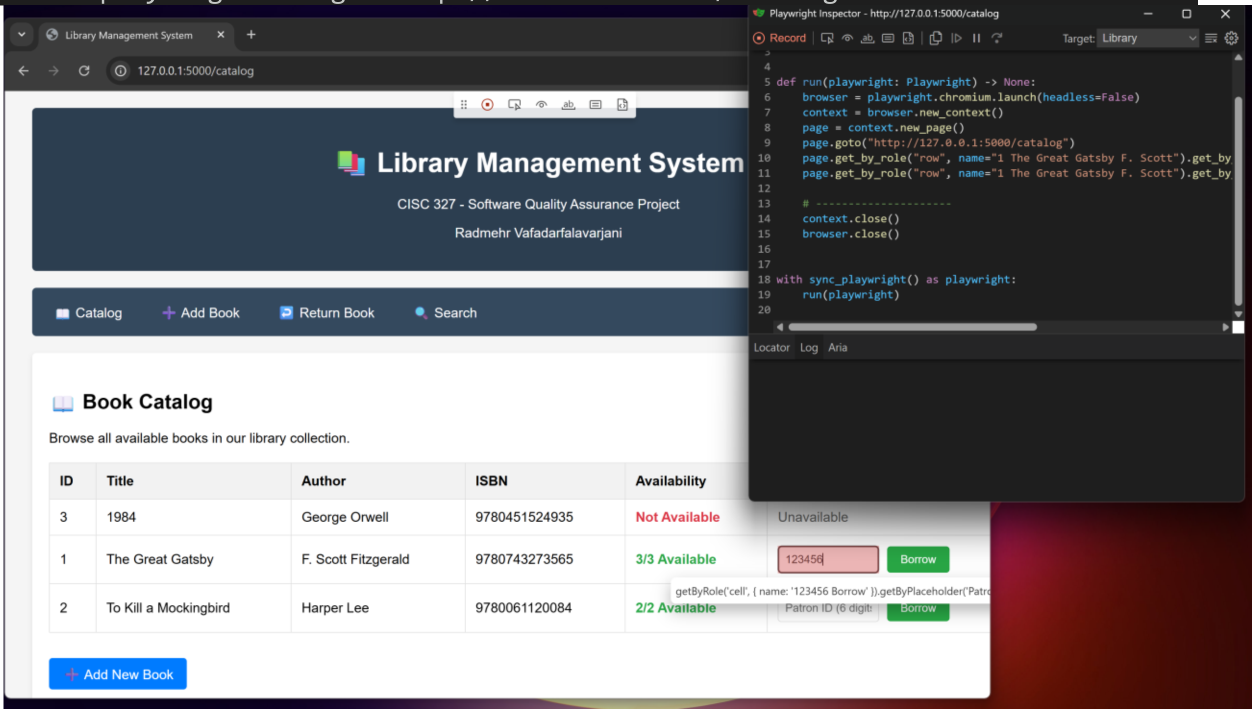
BONUS: Command for playwright codegen

First run the local site (write on separate cmd so you can still use it):

```
python .\app.py
```

Run Codegen (might have to change the local host link to whatever it is set for you)

```
python -m playwright codegen http://127.0.0.1:5000/catalog
```



Library Management System
127.0.0.1:5000/catalog
Playwright Inspector - http://127.0.0.1:5000/catalog
Record
Target: Library
Locator
Log
Aria

```
def run(playwright: Playwright) -> None:
    browser = playwright.chromium.launch(headless=False)
    context = browser.new_context()
    page = context.new_page()
    page.goto("http://127.0.0.1:5000/catalog")
    page.get_by_role("row", name="1 The Great Gatsby F. Scott").get_by
    page.get_by_role("row", name="1 The Great Gatsby F. Scott").get_by

    # --------------------
    context.close()
    browser.close()


with sync_playwright() as playwright:
    run(playwright)
```

Library Management System

CISC 327 - Software Quality Assurance Project

Radmehr Vafadarfalavarjani

Catalog | Add Book | Return Book | Search

Book Catalog

Browse all available books in our library collection.

ID	Title	Author	ISBN	Availability	
3	1984	George Orwell	9780451524935	Not Available	Unavailable
1	The Great Gatsby	F. Scott Fitzgerald	9780743273565	3/3 Available	123456 Borrow
2	To Kill a Mockingbird	Harper Lee	9780061120084	2/2 Available	Patron ID (6 digit Borrow

getByRole('cell', { name: '123456 Borrow' }).getByPlaceholder('Patro

Add New Book