

## CISC/CMPE 327 – Assignment 4

**Name:** Aiden Ramezani

**Student ID:** 20259781

**GitHub link:** <https://github.com/Aidenrmz/cisc327-library-management-a4-9781>

### 1. End-to-End (E2E) Testing Approach

For browser-based end-to-end testing, **Playwright with Pytest** was used to automate real user interactions with the Flask Library Management System. Playwright was selected because it supports real browser automation, reliable UI selectors, and seamless integration with Pytest.

The E2E tests simulate realistic user behavior rather than isolated unit tests. The Flask application is automatically launched in the background using a Pytest fixture before the tests run and is cleanly terminated afterward. This ensures that all tests run against a real, live instance of the web application.

#### Tested User Flows

Two primary user workflows were automated:

1. Adding a new book and verifying it appears in the catalog
2. Borrowing a book and verifying a successful confirmation message

Each test includes clear UI-based assertions to confirm that:

- Books are successfully added,
- Newly added books appear in the catalog,
- Borrow operations complete correctly,
- Success messages are displayed to the user.

```
$ pytest tests/test_e2e.py
=====
platform win32 -- Python 3.12.10, pytest-7.4.2, pluggy-1.6.0
rootdir: C:\Users\arian\OneDrive\Desktop\CISC327-CMPE327-F25-main
plugins: base-url-2.1.0, cov-7.0.0, mock-3.15.1, playwright-0.7.2
collected 2 items

tests\test_e2e.py .. [100%]

=====
2 passed in 4.26s =====
((venv))
```

Figure 1 – Successful End-to-End Test Execution

### 2. Execution Instructions

#### 2.1 Environment Setup

```
python -m venv venv
```

```
source venv/Scripts/activate # Windows Git Bash
```

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

```
playwright install
```

## 2.2 Run E2E Tests

```
pytest tests/test_e2e.py
```

## 2.3 Build Docker Image

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

## 2.4 Run Application in Docker

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

Access the application at:

<http://localhost:5000>

## 3. Test Case Summary

Test ID	Description	Actions	Expected Result
TC-E2E-01	Add new book and verify catalog	Navigate to Add Book → Enter book details → Submit → Navigate to Catalog	The new book appears in the catalog list
TC-E2E-02	Borrow a book and verify confirmation	Navigate to Borrow Page → Select book → Enter Patron ID → Submit	Confirmation message displayed indicating successful borrowing

## 4. Dockerization Process

The application was containerized using a **Dockerfile** built on top of a lightweight Python base image (python:3.11-slim). The Dockerfile performs the following steps:

1. Sets the working directory inside the container.
2. Copies and installs Python dependencies from requirements.txt.
3. Copies the Flask application source code into the container.
4. Sets required environment variables for Flask.
5. Exposes port **5000**.
6. Launches the application using:

```
flask run --host=0.0.0.0 --port=5000
```

The SQLite database is initialized automatically on application startup to ensure the container runs correctly without requiring any external database services or volume mounts.

```
#8 9.677 Downloading python_slugify-8.0.4-py2.py3-none-any.whl (10 kB)
#8 9.695 Downloading werkzeug-3.1.4-py3-none-any.whl (224 kB)
#8 9.712          225.0/225.0 kB 23.2 MB/s eta 0:00:00
#8 9.728 Downloading iniconfig-2.3.0-py3-none-any.whl (7.5 kB)
#8 9.745 Downloading packaging-25.0-py3-none-any.whl (66 kB)
#8 9.751          66.5/66.5 kB 81.9 MB/s eta 0:00:00
#8 9.766 Downloading markupsafe-3.0.3-cp311-cp311-manylinux2014_x86_64_manylinux_2_17_x86_64_manylinux_2_28_x86_64.whl
#8 9.784 Downloading requests-2.32.5-py3-none-any.whl (64 kB)
#8 9.788          64.7/64.7 kB 97.2 MB/s eta 0:00:00
#8 9.805 Downloading text_unidecode-1.3-py2.py3-none-any.whl (78 kB)
#8 9.816          78.2/78.2 kB 21.0 MB/s eta 0:00:00
#8 9.834 Downloading typing_extensions-4.15.0-py3-none-any.whl (44 kB)
#8 9.839          44.6/44.6 kB 43.7 MB/s eta 0:00:00
#8 9.866 Downloading certifi-2025.11.12-py3-none-any.whl (159 kB)
#8 9.876          159.4/159.4 kB 40.3 MB/s eta 0:00:00
#8 9.895 Downloading charset_normalizer-3.4.4-cp311-cp311-manylinux2014_x86_64_manylinux_2_17_x86_64_manylinux_2_28_x8
#8 9.905          151.6/151.6 kB 39.0 MB/s eta 0:00:00
#8 9.925 Downloading idna-3.11-py3-none-any.whl (71 kB)
#8 9.936          71.0/71.0 kB 25.2 MB/s eta 0:00:00
#8 9.950 Downloading urllib3-2.5.0-py3-none-any.whl (129 kB)
#8 9.961          129.8/129.8 kB 19.5 MB/s eta 0:00:00
#8 10.47 Installing collected packages: text-unidecode, urllib3, typing-extensions, python-slugify, pluggy, packaging,
#8 MarkupSafe, itsdangerous, iniconfig, idna, greenlet, click, charset_normalizer, certifi, blinker, Werkzeug, requests,
#8 pytest, pyee, Jinja2, pytest-base-url, playwright, Flask, pytest-playwright
#8 13.74 Successfully installed Flask-2.3.3 Jinja2-3.1.6 MarkupSafe-3.0.3 Werkzeug-3.1.4 blinker-1.9.0 certifi-2025.11
#8 .12 charset_normalizer-3.4.4 click-8.3.1 greenlet-3.2.4 idna-3.11 iniconfig-2.3.0 itsdangerous-2.2.0 packaging-25.0 pl
#8 aywright-1.56.0 pluggy-1.6.0 pyee-13.0.0 pytest-7.4.2 pytest-base-url-2.1.0 pytest-playwright-0.7.2 python-slugify-8.0
#8 .4 requests-2.32.5 text-unidecode-1.3 typing-extensions-4.15.0 urllib3-2.5.0
#8 13.74 WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the s
#8 ystem package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
#8 13.88
#8 13.88 [notice] A new release of pip is available: 24.0 -> 25.3
#8 13.88 [notice] To update, run: pip install --upgrade pip
#8 DONE 14.3s

#9 [5/5] COPY . .
#9 DONE 1.2s

#10 exporting to image
#10 exporting layers
#10 exporting layers 1.6s done
#10 writing image sha256:cbe00e95b29addd24035995d467634f5473d1a597de1c4bc645b775ee4577456 done
#10 naming to docker.io/library/library-app 0.0s done
#10 DONE 1.6s
```

**Figure 2 – Docker Build Output**

The terminal window shows the following command sequence:

```

arian@Aiden-Laptop MINGW64 ~/OneDrive/Desktop/CISC327-CMPE327-F25-main (main)
$ source ./venv/Scripts/activate
(venv)
arian@Aiden-Laptop MINGW64 ~/OneDrive/Desktop/CISC327-CMPE327-F25-main (main)
$ 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 WSGI server instead.
./venv/Scripts/activate
(venv)
arian@Aiden-Laptop MINGW64 ~/OneDrive/Desktop/CISC327-CMPE327-F25-main (main)
$ 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 WSGI 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
172.17.0.1 - [01/Dec/2025 01:51:27] "GET / HTTP/1.1" 302 -
172.17.0.1 - [01/Dec/2025 01:51:27] "GET /catalog HTTP/1.1" 200 -
172.17.0.1 - [01/Dec/2025 01:51:28] "GET /favicon.ico HTTP/1.1" 404 -
[ ]

```

The browser window displays the "Library Management System" application. The header includes the title, project name (CISC 327 - Software Quality Assurance Project), and user (Aiden Ramezani). The main menu bar has links for Catalog, Add Book, Return Book, and Search. Below the menu is a section titled "Book Catalog" with a sub-section "Book Catalog". It says "Browse all available books in our library collection." A table lists one book entry:

ID	Title	Author	ISBN	Availability	Actions
1	E2E Borrowable Book 1764553286	E2E Borrow Author	9784553286626	1/2 Available	<input type="text" value="Patron ID (6 digit)"/> <input type="button" value="Borrow"/>

[+ Add New Book](#)

Figure 3 – Docker Run and Application in Browser

## 5. Docker Hub Deployment

The Docker image was successfully pushed to Docker Hub, removed locally, pulled from the registry, and executed again to verify cloud deployment functionality.

### Docker Hub Command Sequence

```

docker build -t library-app .

docker tag library-app AIDENRMZ/library-app:v1

docker push AIDENRMZ/library-app:v1

docker rmi AIDENRMZ/library-app:v1

docker pull AIDENRMZ/library-app:v1

docker run -p 5000:5000 AIDENRMZ/library-app:v1

```

```
arian@Aiden-Laptop MINGW64 ~/OneDrive/desktop/CISC327-CMPE327-F25-main (main)
$ docker push aidenrmz/library-app:v1
The push refers to repository [docker.io/aidenrmz/library-app]
22e39856bc11: Pushed
1654ca450129: Pushed
34f125505920: Pushed
a38e001efb49: Pushed
720ee7aae3ad: Mounted from library/python
655ff69eb9c8: Mounted from library/python
5c988eaa8862: Mounted from library/python
70a290c5e58b: Mounted from library/python
v1: digest: sha256:9a8766a4fc598824e15a4c3cf1d51f3811ed0ee9bbbd7fff04205be0ec042711
((venv) )
```

Figure 4 – Successful Docker Push to Docker Hub

```
arian@Aiden-Laptop MINGW64 ~/OneDrive/desktop/CISC327-CMPE327-F25-main (main)
$ docker rmi aidenrmz/library-app:v1
Untagged: aidenrmz/library-app:v1
Untagged: aidenrmz/library-app@sha256:9a8766a4fc598824e15a4c3cf1d51f3811ed0ee9bbbd7fff04205be0ec042711
((venv) )
arian@Aiden-Laptop MINGW64 ~/OneDrive/desktop/CISC327-CMPE327-F25-main (main)
$ docker rmi aidenrmz/library-app:v1
Error response from daemon: No such image: aidenrmz/library-app:v1
((venv) )
```

Figure 5 – Docker Image Removed Locally (docker rmi)

```
arian@Aiden-Laptop MINGW64 ~/OneDrive/desktop/CISC327-CMPE327-F25-main (main)
$ docker pull aidenrmz/library-app:v1
v1: Pulling from aidenrmz/library-app
Digest: sha256:9a8766a4fc598824e15a4c3cf1d51f3811ed0ee9bbbd7fff04205be0ec042711
Status: Downloaded newer image for aidenrmz/library-app:v1
docker.io/aidenrmz/library-app:v1

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview aidenrmz/library-app:v1
((venv) )
```

Figure 6 – Docker Pull from Docker Hub

The terminal window shows the following command being run:

```

arian@Aiden-Laptop MINGW64 ~/OneDrive/Desktop/CISC327-CMPE327-F25-main (main)
$ source ./venv/Scripts/activate
(venv)
arian@Aiden-Laptop MINGW64 ~/OneDrive/Desktop/CISC327-CMPE327-F25-main (main)
$ 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 WSGI server instead.
./venv/Scripts/activate
(venv)
arian@Aiden-Laptop MINGW64 ~/OneDrive/Desktop/CISC327-CMPE327-F25-main (main)
$ 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 WSGI 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
172.17.0.1 - [01/Dec/2025 01:51:27] "GET / HTTP/1.1" 302 -
172.17.0.1 - [01/Dec/2025 01:51:27] "GET /catalog HTTP/1.1" 200 -
172.17.0.1 - [01/Dec/2025 01:51:28] "GET /favicon.ico HTTP/1.1" 404 -

```

The browser window displays the "Library Management System" interface with the following details:

- Header:** 127.0.0.1:5000/catalog, Library Management System, CISC 327 - Software Quality Assurance Project, Aiden Ramezani.
- Navigation Bar:** Catalog, Add Book, Return Book, Search.
- Section:** Book Catalog. Sub-section: Catalog.
- Text:** Browse all available books in our library collection.
- Table:** Shows a single book entry:
 

ID	Title	Author	ISBN	Availability	Actions
1	E2E Borrowable Book	E2E Borrow Author	9784553286626	1/2 Available	Patron ID (6 digit) Borrow
- Buttons:** Add New Book.

Figure 7 – Docker Run from Pulled Image

## 6. Challenges and Reflections

One of the main challenges encountered during this assignment was managing the automatic startup and shutdown of the Flask server for the E2E test environment. Unlike unit tests, browser-based tests require a fully running application, which required careful synchronization inside Pytest.

Another challenge involved ensuring that the SQLite database initialized automatically inside the Docker container without relying on persistent volumes or external authentication. This was resolved by initializing the database schema safely at application startup.

This assignment demonstrated the practical value of:

- End-to-end testing for verifying complete system behavior,
- Docker for consistent runtime environments,
- Cloud-based container deployment using Docker Hub.

Together, these tools improve software reliability, portability, and deployment reproducibility.