GitHub Link:

https://github.com/SY64I/cisc327-library-management-a4-6063

Section 1

Arlen Smith, 20386063, December 1st 2025

Section 2

The tool I used to perform End-to-End testing for this assignment was Playwright. Test cases for end-to-end testing are all contained in a newly added test_e2e.py script located in my project's tests folder. The streams of user flow the test cases all flow into each other, covering adding a book to the database, borrowing the newly added book, returning the borrowed book, and searching for the book using all 3 search methods. Many assertions are used throughout, to ensure that the user is on the correct pages,that data is correctly appearing in input fields when entered, that success messages appear after each action, that the correct book is appearing in the catalog when added or searched, and that the book's availability is correctly updated after a borrow or return.

Section 3

To run either the end-to-end tests provided for the first time, the project's files and external packages must first be installed if the user has not already done so by running the commands below in order:

```
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

After the user has setup the project, to run the tests the user must create two terminals, one to run the application, the other to run the tests. In one terminal, run the following commands:

```
source venv/bin/activate
python3 app.py
```

In the other terminal, run the following commands:

```
source venv/bin/activate
pytest
```

To run the containerized application, run the following command:

```
docker run -p 5000:5000 sy64i/library-app:v1
```

Section 4

A summary of the test cases in test_e2e.py is below:

test_has_title()

Functions as a sanity check to ensure that the website is online and can be connected to, also functions to clear any existing database data before testing.

- Clears the database data, goes to the website's main page, ensures that the title is "Library Management System' to ensure that the user is on the main page.

test_user_add_book()

Test case that walks through a user adding a book into the database with correct data. Below is a step-by-step of what the test case does:

1. Starts by going to the main page and asserts that the website's title is correct and that the user is on the catalog page through the website's URL.
2. The "Add Book" button is clicked, and assertions are made to ensure the website's header says "Add New Book" and to ensure the user is on the add_book page through the website's URL.
3. The title input field is clicked, and the title "UI Testing: Tips and Tricks" is entered.
4. The author input field is clicked, and the author "AJ. Smith" is entered.
5. The ISBN input field is clicked, and the ISBN "4526138443166" is entered.
6. The total copies input field is clicked, and the number of copies is entered to be "7".
7. Assertions to ensure each input field above has the values entered are made.
8. The button "Add Book to Catalog" is clicked. An assertion is made to ensure the user is now back on the catalog page through checking the website's URL.
9. An assertion to verify that a success message for successfully adding the new book is made.
10. Assertions are made to ensure the book was added and is appearing in the book catalog. This includes verifying the title is "UI Testing: Tips and Tricks", the author is "AJ. Smith", the ISBN is "4526138443166", and that the available copies text displays "7/7 Available".

test_user_borrow_book()

Test case that walks through a user borrowing the newly added book added previously. Below is a step-by-step of what the test case does:

1. Starts by going to the main page and asserts that the website's title is correct and that the user is on the catalog page through the website's URL.

2. Ensures the book added from the previous test exists in the current book catalog. This is done through asserting that the title, ISBN and available copies for the book are “UI Testing: Tips and Tricks”, “4526138443166” and “7/7 Available” respectively.
3. The patron ID field is clicked, and the patron ID “678942” is entered. An assertion is then made to ensure the input field has the correct information.
4. The “Borrow” button in the cell is clicked. Assertions are made to ensure that a success message for successfully adding the new book is made, and to ensure that the book’s available copies have been updated to display “6/7 Available”.

test_user_return_book()

Test case that walks through a user through the process of returning a borrowed book. Below is a step-by-step of what the test case does:

1. Starts by going to the main page and asserts that the website’s title is correct and that the user is on the catalog page through the website’s URL.
2. The “Return Book” button is clicked, and assertions are made to ensure the website’s header says “Return Book” and to ensure the user is on the return page through the website’s URL.
3. The patron ID field is clicked, and the patron ID “678942” is entered.
4. The book ID field is clicked, and the book ID that corresponds to the UI Testing book is entered (the book ID to enter is found through code since the book ID is something the database creates automatically)
5. Assertions to ensure each input field above has the values entered are made.
6. The button “Process Return” is clicked. An assertion to verify that a success message for successfully returning the book is made.
7. The ‘Catalog’ button is clicked, and assertions are made to ensure the website’s header says “Book Catalog” and to ensure the user is on the catalog page through the website’s URL.
8. Assertions are made to ensure that the UI Testing book exists through verifying the book’s title appears, and to ensure that the book’s available copies have been updated to display “7/7 Available”.

test_user_search_book()

Test case that walks through a user searching for a book through all 3 different search methods. Below is a step-by-step of what the test case does:

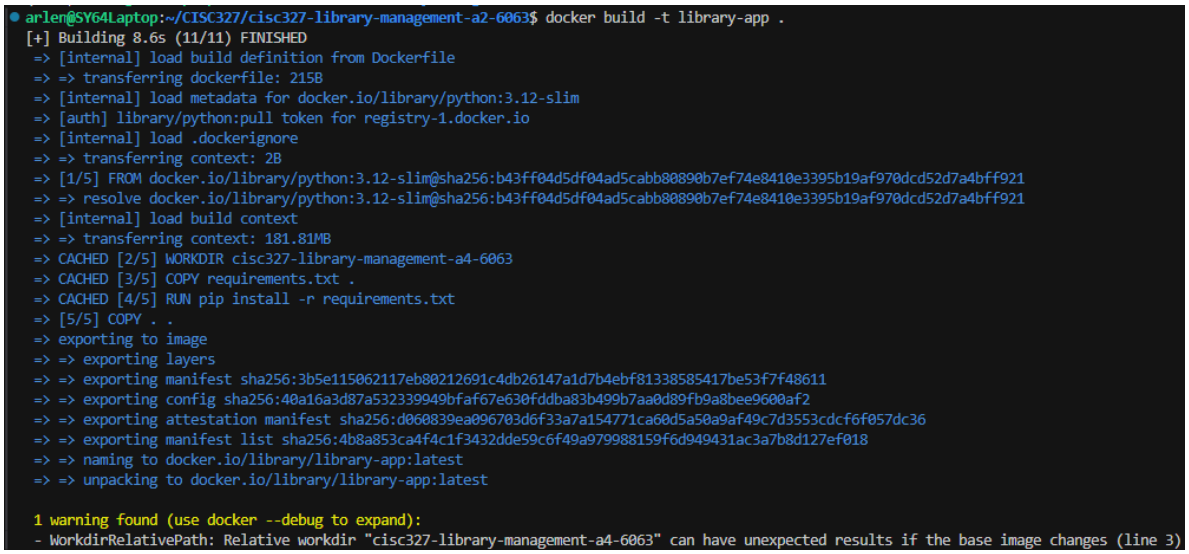
1. Starts by going to the main page and asserts that the website’s title is correct and that the user is on the catalog page through the website’s URL.
2. The “Search” button is clicked, and assertions are made to ensure the website’s header says “Search Books” and to ensure the user is on the search page through the website’s URL.
3. The search input field is clicked, and the search prompt “ui testing” is entered.
4. Assertions are made to ensure the input field has the correct information and that the search type box has the search type set to “title” (which should be the website’s default)

5. The “Search” button is clicked. An assertion is made to ensure that search results text displays the search prompt “ui testing” and that the search type is “(title)”. Another assertion is made to ensure a book with the title “UI Testing: Tips and Tricks” has appeared.
6. The search input field is clicked, and the search prompt “a” is entered.
7. The search type box option “author” is selected.
8. Assertions are made to ensure the input field has the correct information and that the search type box has the search type set to “author”.
9. The “Search” button is clicked. An assertion is made to ensure that search results text displays the search prompt “a” and that the search type is “(author)”. Another assertion is made to ensure a book with the author “AJ. Smith” has appeared.
10. The search input field is clicked, and the search prompt “4526138443166” is entered.
11. The search type box option “isbn” is selected.
12. Assertions are made to ensure the input field has the correct information and that the search type box has the search type set to “isbn”.
13. The “Search” button is clicked. An assertion is made to ensure that search results text displays the search prompt “4526138443166” and that the search type is “(isbn)”. Another assertion is made to ensure a book with the ISBN “4526138443166” has appeared.
14. Since this is the last test case, the database is cleared to ensure the test data is reset.

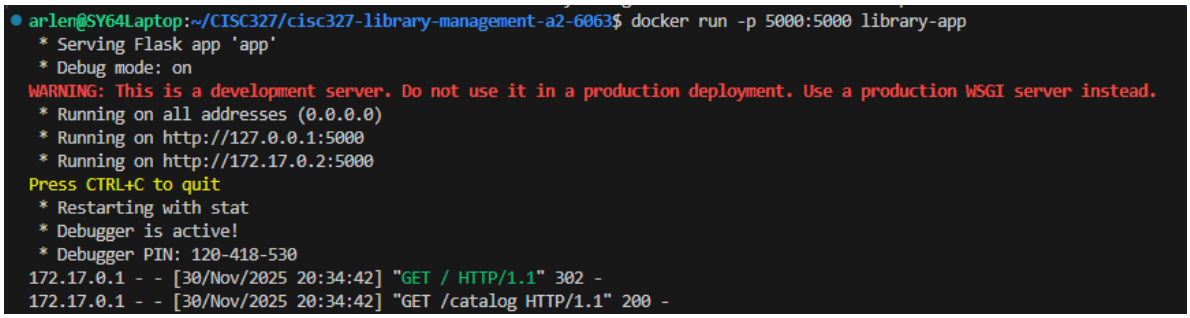
Section 5

To dockerize the app, the Dockerfile uses Python version 3.12, swaps to the main directory, copies the requirements text and installs the requirements. Then it copies the rest of the app files, exposes the Dockerfile to port 5000, then specifies the default python commands to run the app.

To build the docker container, the commands from the instructions were ran, and lead to the results below:

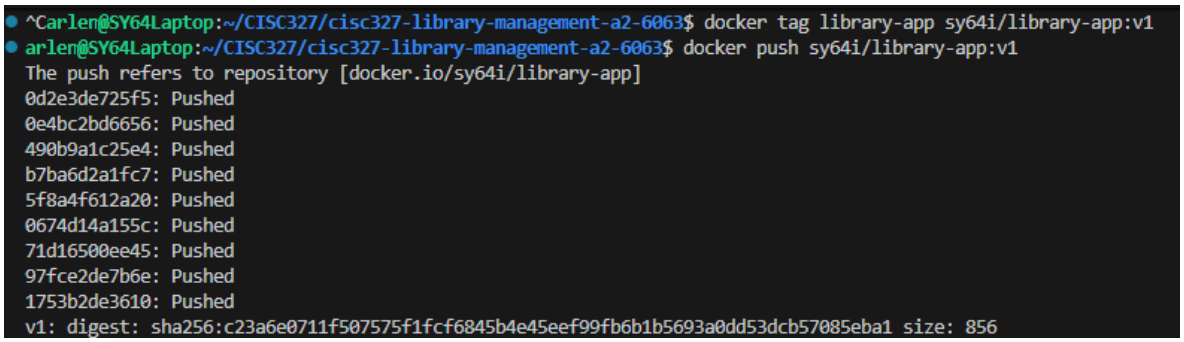


Then, the build was ran, and lead to the following output:

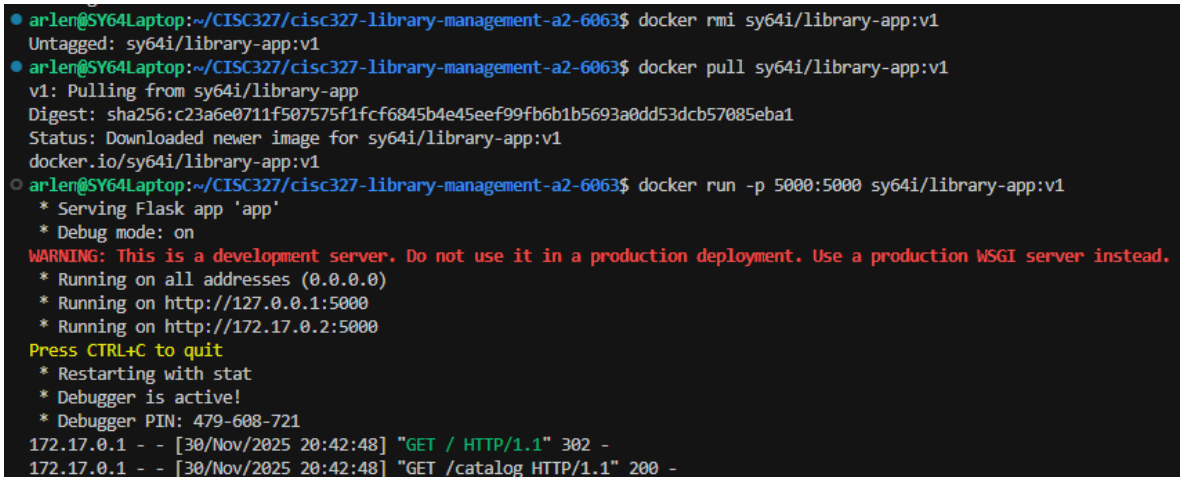


Section 6

Below is a screenshot of the container being successfully pushed:



Below is a screenshot of the container being successfully deleted, pulled then ran:



Section 7

The main challenges that arose during this assignment was figuring out how to use Playwright and how to create the Docker container when I first started. I have previously never used either Playwright or Docker before, so it took some time for me to figure out how to use them. Once I understood how they worked however, the assignment was relatively smooth. The only other issue I ran into was not being able to run tests due to my website not running, but I eventually solved that through running tests by having one terminal run the website and another run the tests as I specified in the instructions in Section 3.