## Assignment 3: Web Application with Database Integration and Automated Testing using Selenium

Name: Twinkle Mishra  
Course: Database Automation (PROG8850)

# GitHub Repository

The full project source code is available at: <https://github.com/TwinkleM97/A3_DB_Automation>

# Introduction

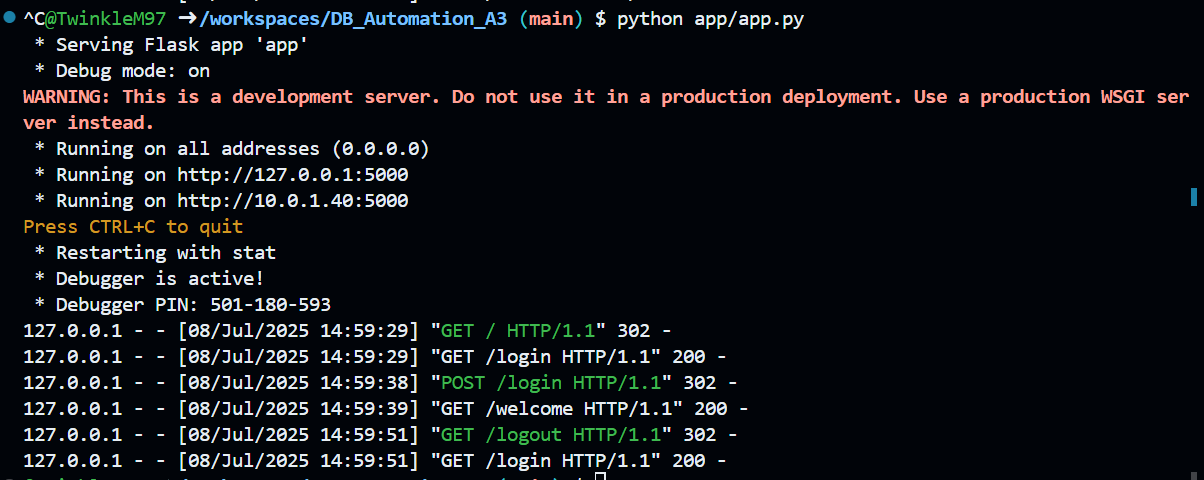
This report outlines the development and testing process of a login-based web application as part of Assignment 3 for the PROG8850 course. The application was built using Flask and integrates with a MySQL database. Automated UI testing was conducted using Selenium to validate user input functionality and data persistence. All components were containerized using Docker for consistency and reproducibility.

# Part 1: Web Application Development

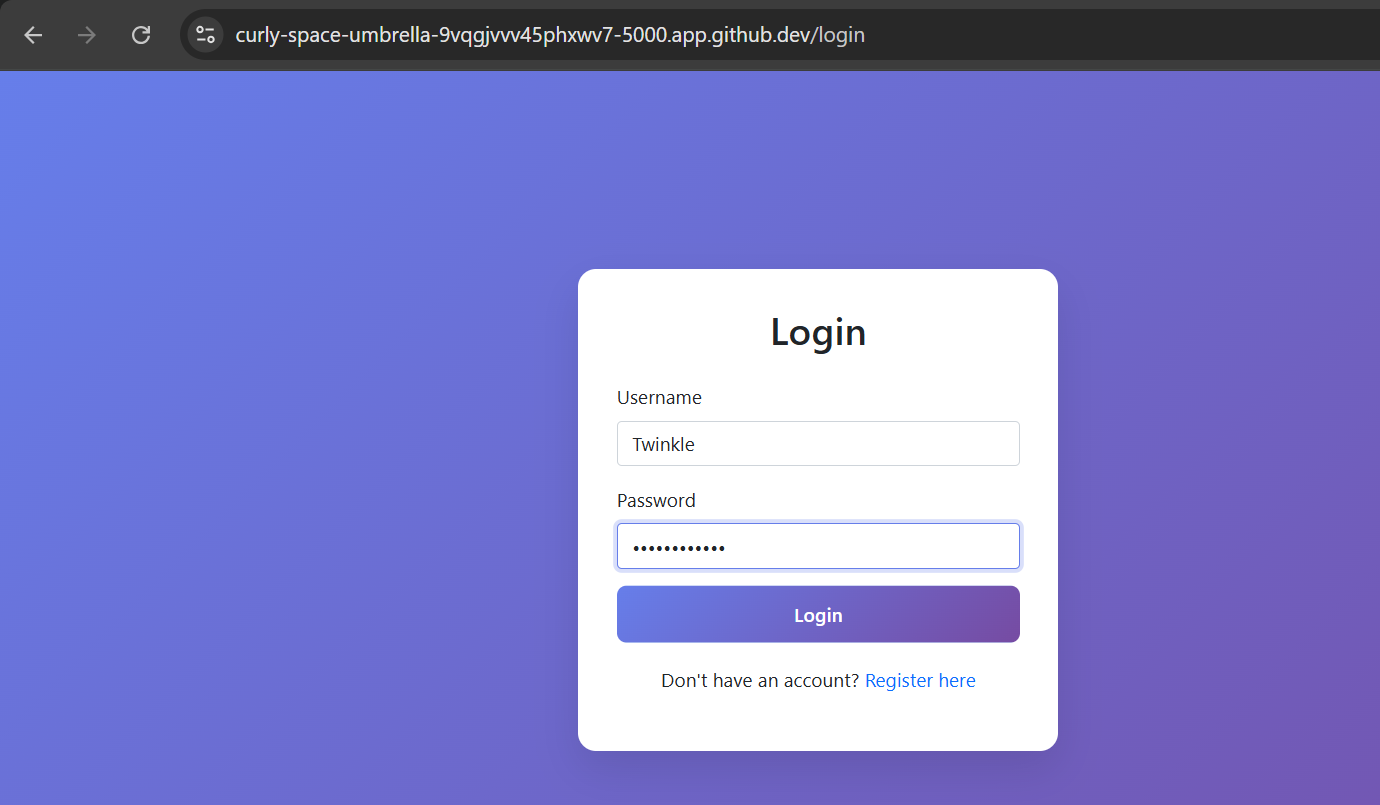
The application was developed using Python's Flask framework. It includes two main routes: `**/register**` for new user registration and `**/login**` for authentication. Upon submission, user credentials are captured and stored in a MySQL database via SQLAlchemy. HTML templates were rendered using Jinja2.

**Screenshots of the running Flask application:**

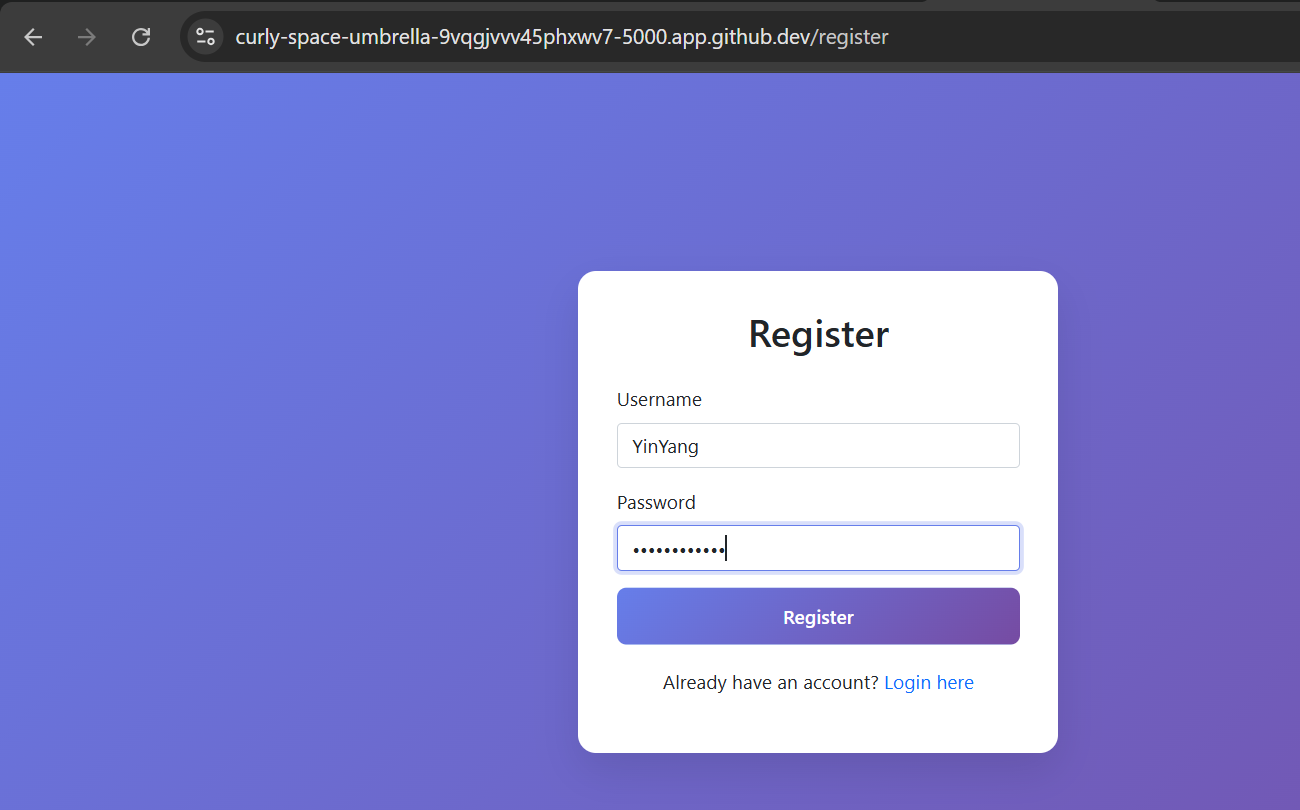
1. **flask app running**



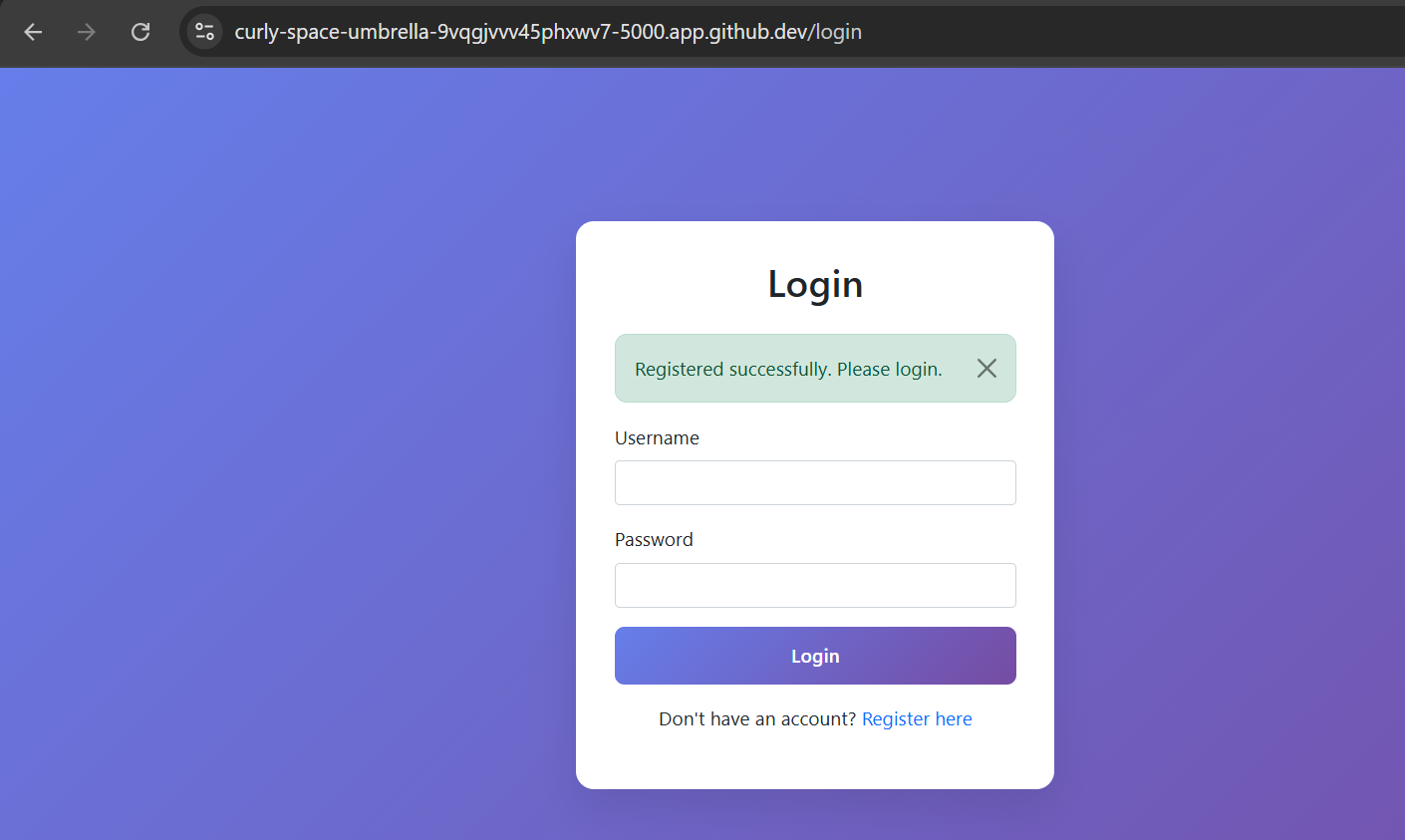
1. **Login page**



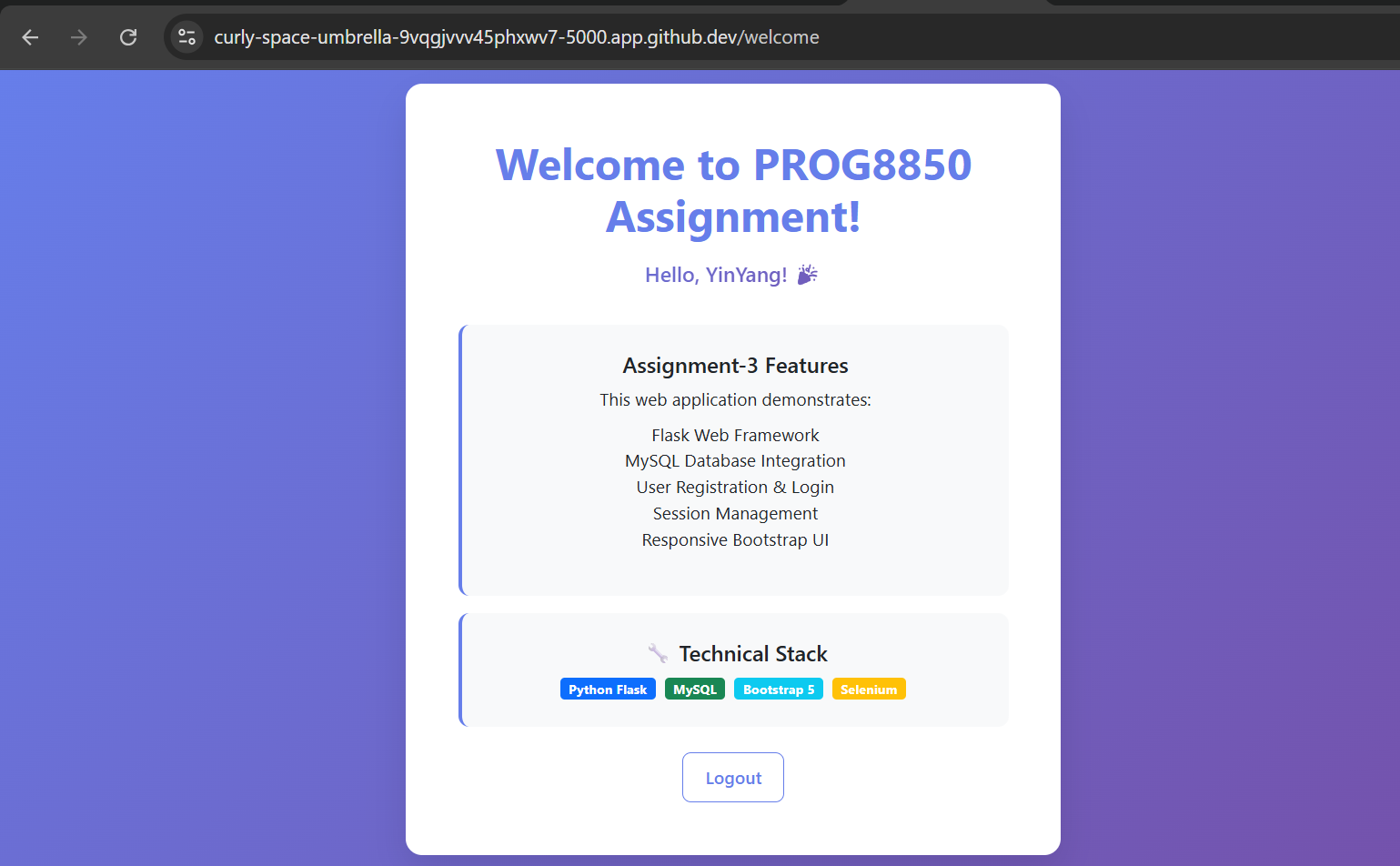
1. **Register page**



1. **Register success**



1. **Login success welcome page**

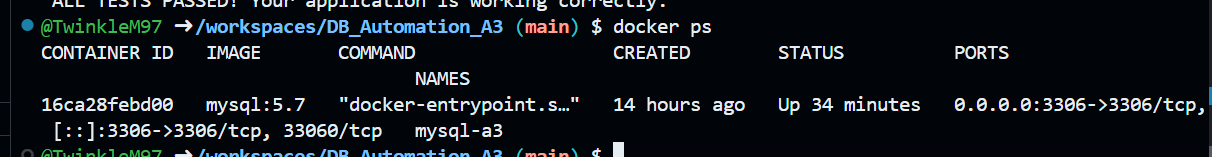


# Part 2: Database Setup

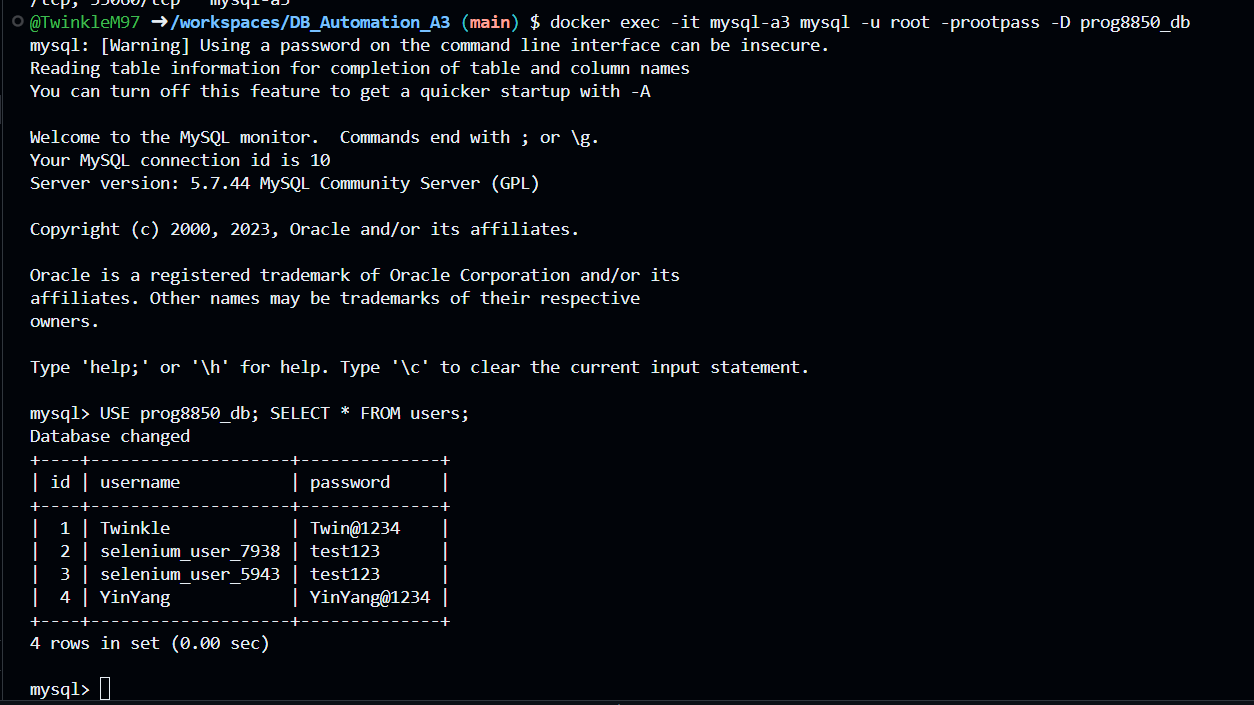
The database was configured using Docker Compose. A MySQL container was launched and initialized with a `users` table, which stores an ID, username, and password. The database is automatically bootstrapped using an `**init.sql**` script.

Relevant screenshots:

1. **mysql container running**



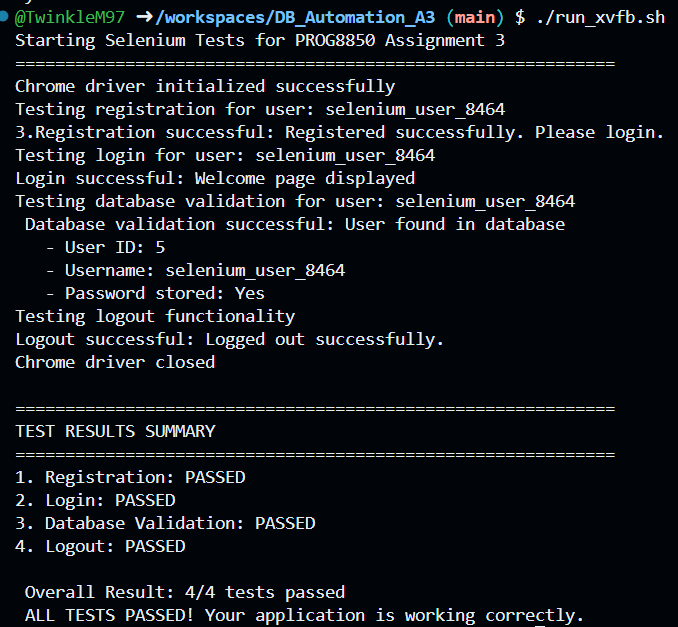
1. **user inserted in database**



# Part 3: Selenium Integration

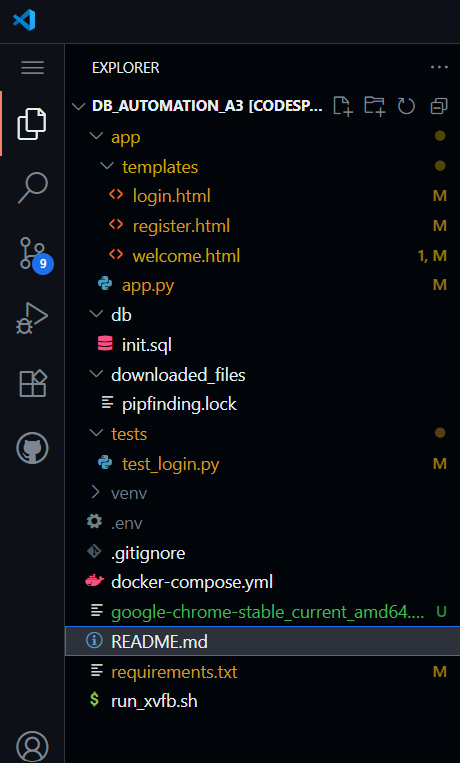
A Selenium script was created to automate the login and registration flows. Using headless Chrome in a Docker container, the script mimics a real user interacting with the interface, submits credentials, and checks for successful database insertions. Screenshots were taken to verify that all tests passed.

1. **Selenium test execution with all tests passing.**

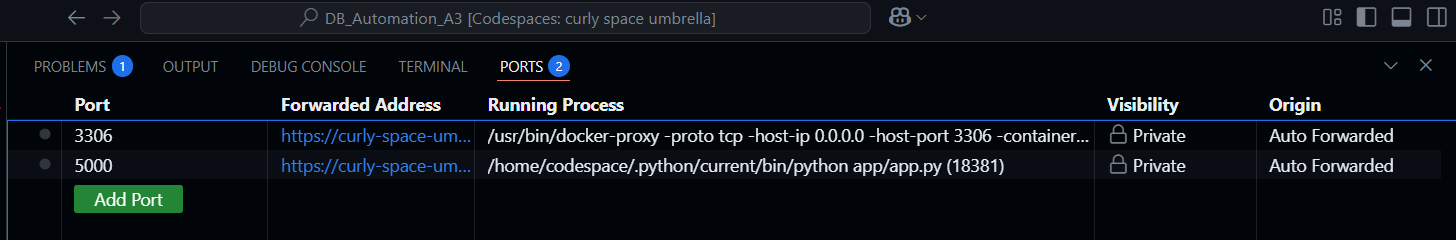


# Part 4: Project Structure and Execution

The entire project is structured with Docker Compose to orchestrate Flask, MySQL, and the testing environments. Environment variables and dependencies are isolated for maintainability. Screenshots of the file structure and forwarded port URL in GitHub Codespaces are shown below.



1. **Port forwarding url**



# Conclusion

This project successfully demonstrates the automation of a database-integrated login web application and its testing via Selenium. All components were built with reproducibility, clarity, and modularity in mind using Docker and GitHub. This approach ensures consistent deployment and testing environments.

**References**

Flask. (n.d.). *Flask documentation*. Pallets Projects. <https://flask.palletsprojects.com/>

MySQL. (n.d.). *MySQL 8.0 reference manual*. Oracle. <https://dev.mysql.com/doc/>

Selenium. (n.d.). *Selenium with Python documentation*. SeleniumHQ. <https://selenium-python.readthedocs.io/>

Docker. (n.d.). *Docker documentation*. Docker Inc. <https://docs.docker.com/>

GitHub. (n.d.). *GitHub Actions documentation*. GitHub. <https://docs.github.com/en/actions>