**Introduction**

The practical course, software development using Python, is one of the most valuable experiences I have ever had in strengthening different aspects of modern software engineering. Before this course, I had a basic knowledge of Python and no knowledge about containerization and CI/CD pipelines. This report reflects my learning process, particularly about pytest, docker, and GitHub Actions, a project created using FastAPI and PostgreSQL.

**Project Overview**

The project's main aim was to develop an efficient user management system with strong features. The system was developed based on FastAPI, a modern web framework for building APIs fast in Python, and PostgreSQL, a strong relational database. Key features of the project included:

***User Registration and Authentication:*** Integrated strong registration and authorization procedures such as the use of email confirmation and the use of roles.

***Password Management:*** Ensured that the process of data registration, updating, and forgotten password retrieval incorporated stringer validation of passwords (Olsen, 1992).

***Profile Management:*** Provided a feature for users to modify the information in their profile and provided managers or admins the ability to promote any user to a professional account, while making sure no one could edit their profile unless they had a verified email.

***Nickname Management:*** Solved the problem of nickname conflicts since the program creates unique nicknames for users.

***Automated Testing:*** Before deploying the changes made in this project, pytest and coverage tools were used to guarantee both coverage and function in tests.

***Containerization and CI/CD:*** Built a docker image out of the application to make deployment easier and also integrated Github actions for CI/CD testing (Houerbi, et al., 2024).

**Methodology**

To finish this project, top-level best practices were utilized of structured/iterative software development with essential technologies for the engine including FastAPI as the backend and PostgreSQL for data management. Docker provided consistent containing that was done to implement the environment while GitHub Actions provided an efficient means of conducting CI/CD pipeline to test and deploy new changes (Rautiainen, 2023). Starting base functionalities of user registration, user authentication, password management, and profile management were developed in a sequential manner giving great importance to security measures including email confirmation, role, and password validation. Improvements in the user’s profile enabled the user to update their profile. In this case, you could only update your profile if you were a verified user, thus reducing cases of data vandalism. Testing through pytest and coverage was used to incorporate checks and logs to help establish debugging in practical scenarios (Ali, Riganelli & Mariani, 2024).

**Result**

The system provides secure user management with additional options for email confirmation, travel lists, password complexity for staff, and a nickname generator for users that gives a promise for a highly functional range of security and usability. Containerization through Docker and continuous integration and continuous delivery by using GitHub Actions made the process of deployment easier and more effective. Testing through the pytest improved reliability which resulted from the scalability of the system that suits real applications through the coverage test.

**Discussion**

This project showed how important it is to draw the theory into operational applications. Being utterly unfamiliar with such things as Docker and GitHub Actions, the steep learning curve was justified as these tools have dramatically increased productivity (Houerbi, et al., 2024). The implementation of FastAPI and PostgreSQL proved to provide a solid ground for developing a stats application with good scalability and security. It can free up time earlier in the development process for testing and automation that pays great dividends later on in the development.

Sure, problems like nickname collisions and role management had called for inventive approaches within which realization showed the value of logic and planning. Furthermore, managing the users’ security by adding improved password checks and email validation proved also useful for the project. CI/CD pipeline integration minimized the role of hand operations and exposed best practices required in modern software deployment.

**Conclusion**

This project was very beneficial to me in terms of developing new knowledge and improving my Python development, testing, deployment, and debugging abilities. With FastAPI, PostgreSQL, Docker, pytest, and GitHub Actions, it was possible to create a secure, high-scalable, feature-filled application. The problems in concrete implementation met and solved in the course of the project extended my knowledge of the actual practice of creating the software.

**References:**

Ali, Q., Riganelli, O., & Mariani, L. (2024). Testing in the Evolving World of DL Systems: Insights from Python GitHub Projects. *arXiv preprint arXiv:2405.19976*.

Olsen, D. (1992). *User interface management systems: models and algorithms*. Morgan Kaufmann Publishers Inc..

Rautiainen, O. (2023). GitHub Enterprise and Migration of CI/CD Pipelines from Azure DevOps to GitHub.

Houerbi, A., Siala, C., Tucker, A., Rzig, D. E., & Hassan, F. (2024). Empirical Analysis on CI/CD Pipeline Evolution in Machine Learning Projects. *arXiv preprint arXiv:2403.12199*.