**Sara Chaker**

**IS 601-852**

**Professor Keith Williams**

**05/02/2205**

**Final Reflection Paper**

## **Overview and Project Goals**

This document summarizes my experience working on the User Management System final project for Professor Keith Williams's course at NJIT. Throughout this project, I had the opportunity to practically implement software engineering practices such as feature development, quality assurance (QA), comprehensive testing, and continuous integration/deployment (CI/CD) via GitHub Actions and Docker.

The project's main objectives included:

* Gaining practical development experience through hands-on coding and collaboration.
* Improving quality assurance (QA) by identifying, documenting, and resolving software bugs.
* Enhancing test coverage to cover critical functionalities, edge cases, and ensure robust code.
* Implementing new features while adhering to best practices, writing clean, maintainable code.
* Ensuring deployability through successful CI/CD pipeline setup with Docker and GitHub Actions.

**QA Issues Raised and Resolved (5 Issues)**

As required, I identified and documented five critical QA issues related to the existing functionalities. I carefully detailed each issue clearly in GitHub, including replication steps, expected outcomes, and actual behavior:

1. [Dockerfile libc-bin downgrade error](https://github.com/Sarachaker/user_management/issues/2): Allowed --allow-downgrades for glibc patch.
2. [Profile picture URL validator](https://github.com/Sarachaker/user_management/issues/3): Ensured URLs end with `.jpg .jpeg.
3. [Nickname generation bypass](https://github.com/Sarachaker/user_management/issues/4): Always run the generator when the nickname is missing.
4. [Email verification role logic](https://github.com/Sarachaker/user_management/issues/5): Prevent AUTHENTICATED from re-verifying.
5. [Weak password acceptance](https://github.com/Sarachaker/user_management/issues/6): Added Pydantic validator for strength rules.

Each of these issues was resolved through a structured approach involving

* Clear identification and documentation of problems.
* Writing and executing tests that reproduce issues reliably.
* Implementing robust and maintainable fixes.
* Peer-reviewing the solutions through the GitHub issue workflow.

*Links to all closed issues can be found in the repository's Issues section and issues hyperlink.*

## **Comprehensive Testing (10 New Tests)**

I significantly expanded the project's test coverage by writing 10 additional tests. These tests covered critical functionalities, focusing primarily on user authentication, database interactions, edge cases, and my newly implemented feature:

1. Test profile picture upload with oversized file.
2. Test invalid MinIO bucket name scenarios.
3. Test retrieval of a non-existent profile picture.
4. Test login with invalid credentials.
5. Test token expiration handling.
6. Test user registration with duplicate emails.
7. Test input validation for user profile updates.
8. Test proper cleanup after database session leaks.
9. Test user deletion edge cases.
10. Test unauthorized access to protected routes.

These tests were integrated into GitHub Actions CI/CD workflows, ensuring the project consistently passes all automated checks.

## **Feature Implementation: MinIO Profile Picture Upload**

I selected and implemented the Profile Picture Management feature, utilizing MinIO storage to securely upload and serve user profile images. The implementation process included:

* Designing endpoints in FastAPI to handle file upload, validation, and secure storage.
* Integrating MinIO, ensuring robust error handling and efficient file retrieval.
* Writing comprehensive tests ensuring both positive and negative scenarios were well-handled.
* Documenting usage clearly in the project's codebase for maintainability and future reference.

This feature enhances user interaction significantly, demonstrating a practical real-world application of cloud-based object storage within our system.

## **CI/CD Workflow and Deployability (DockerHub)**

The project is fully automated via GitHub Actions, ensuring reliable builds and deployments. The CI/CD pipeline performs:

* Automated execution of pytest-based test suites.
* Docker image builds upon successful tests.
* Automatic deployment of the Docker image to Docker Hub.

The Dockerized deployment ensures our system is consistently available and easy to deploy across environments.

DockerHub Deployment: Docker Hub Repository

## **Challenges and Learnings**

During the project, I encountered several practical challenges:

* Handling exceptions and error scenarios robustly.
* Debugging Docker container and integration errors.
* The task involves writing meaningful tests that cover edge cases.

Resolving these challenges has significantly strengthened my practical problem-solving skills, debugging techniques, and understanding of professional software development processes. I am now better prepared for industry scenarios, confident in my ability to effectively collaborate, identify issues proactively, write comprehensive tests, and deploy software reliably.

## **Final Thoughts**

Overall, the User Management System project provided an excellent foundation for enhancing my practical software engineering skills. The process—from raising QA issues and writing robust tests to successfully deploying via Docker—has been both challenging and rewarding. I look forward to applying these learned skills in my future software development career.

***Important Links:***

**Github Repository:** <https://github.com/Sarachaker/user_management>

**Dockerhub Repository:** <https://hub.docker.com/repository/docker/sarachaker/user_management/general>

**Deployment:** <https://github.com/Sarachaker/user_management/deployments/production>

**Workflow CI/CD Pipeline:** <https://github.com/Sarachaker/user_management/actions>