Throughout this course, I gained extensive hands-on experience developing and deploying web applications using Python and the FastAPI framework. Before taking this course, I had limited exposure to web development with asynchronous frameworks. Now, I feel confident building full-stack FastAPI applications, understanding how to structure routes, schemas, services, and database models cleanly. I also learned how to configure applications to run efficiently both locally and in production environments, which included managing settings, error handling, and server configurations. Additionally, I gained experience integrating with RESTful APIs, which taught me how to consume external services and build my own API endpoints that other systems could call. This experience helped me develop a deeper understanding of backend architecture and how different components of a modern application interact.

Another major skill I developed was using Git for version control. The course emphasized professional Git practices such as writing meaningful commit messages, working on feature branches, and collaborating through pull requests. I now understand how critical version control is for working on larger projects, especially when collaborating with other developers or maintaining clean project histories. I also learned how to containerize applications using Docker. This included writing Dockerfiles, creating docker-compose configurations to run multi-service setups (like FastAPI, PostgreSQL, Kafka, and Redis together), and understanding networking between containers. Managing full-stack workflows using Docker gave me insight into real-world deployment practices and the importance of reproducible environments.

Another important topic we covered was implementing user authorization and authentication. I worked with OAuth2, JWT tokens, and role-based access controls to ensure users could securely log in and interact with web applications. This experience, combined with learning about the web security model (e.g., how to protect against common vulnerabilities like XSS or CSRF), helped me better appreciate the importance of secure software design. Testing was another major focus. I learned to write unit tests and integration tests using pytest, and how to incorporate testing into continuous integration workflows on GitHub Actions. Building a habit of writing testable code and maintaining high code quality helped me understand how professional developers ensure reliability and maintainability over time.

Overall, this course gave me a complete understanding of modern Python web development, from backend APIs to databases to Dockerized deployments. More importantly, it trained me to work in a professional environment by following industry standards, using proper version control, writing clean and tested code, and developing secure and maintainable systems. I now feel much more prepared to build production-quality applications and contribute effectively to software engineering teams.

5 QA Issues

1. https://github.com/amschultz21/IS601-Final/issues/1
2. https://github.com/amschultz21/IS601-Final/issues/2
3. https://github.com/amschultz21/IS601-Final/issues/3
4. https://github.com/amschultz21/IS601-Final/issues/4
5. https://github.com/amschultz21/IS601-Final/issues/5

11 New Test

1. https://github.com/amschultz21/IS601-Final/blob/main/tests/conftest.py
2. https://github.com/amschultz21/IS601-Final/blob/main/tests/test_api/test_users_api.py
3. https://github.com/amschultz21/IS601-Final/blob/main/tests/test_services/test_user_service.py

3 Features

1. Profile Picture Upload with Minio
2. Event Driven Email Notifications with Celery and Kafka
3. User Search and Filtering

Docker Hub Repository

GitHub Repository