**MY JOURNEY (by Yash Shah)**

This course has been a good experience in understanding real-world software development workflows, version control, testing methodologies, containerization, and cloud deployment. From the outset, the curriculum was designed to simulate the pace and expectations of a professional development environment. I’m deeply grateful to Professor Keith Williams for creating such an engaging, hands-on course. His guidance, enthusiasm for technology, and practical demonstrations helped bridge the gap between textbook theory and real-world systems.

One of the highlights of the course was the strong emphasis on industry practices. From mastering git and GitHub workflows to writing clean, modular, and testable code, I feel far more prepared for collaborative development environments. The use of GitHub Issues, Pull Requests, automated testing, and DockerHub deployments gave me a robust foundation in DevOps principles.

The biggest learning curve was understanding test-driven development and writing meaningful QA tests. Initially, I found it difficult to anticipate edge cases and write isolated, reliable tests. Over time, through feedback and repetition, I was able to design and execute over **10 new test cases** that enhanced system reliability. I also resolved **5 quality assurance (QA) issues**, ranging from input validation to improving API reliability, all of which are closed and linked below.

**Final Project**

This FastAPI and PostgreSQL-based project focuses on user management with features like registration, authentication, email verification, role-based access control, and strong password validation. I implemented profile management enhancements, including professional status upgrades by managers/admins and validation of profile links. Key challenges included generating unique nicknames, enforcing secure passwords, and managing user roles effectively. I ensured only verified users could update profiles and wrote automated tests using pytest and coverage for reliability. Dockerization and CI/CD with GitHub Actions streamlined deployment. This project sharpened my skills in backend development, testing, and real-world problem-solving.

The final project was both challenging and rewarding. I also built a **User Profile Picture Upload Feature using MinIO Object Storage**, which required me to implement a secure file upload endpoint, integrate MinIO as a distributed object storage solution, and deploy the application via Docker. This feature tested everything I had learned — from designing RESTful endpoints and validating file uploads, to configuring cloud storage environments, writing end-to-end tests for upload workflows, and successfully deploying the full application stack to DockerHub.

Here are the key components of my final submission:

**QA Issues (6)**

* Profile picture validation (<https://github.com/yashah9/IS601-Final/issues/1> )
* Email verification/validation (<https://github.com/yashah9/IS601-Final/issues/3>)
* Password validation (<https://github.com/yashah9/IS601-Final/issues/6> )
* Reset password with constraints (<https://github.com/yashah9/IS601-Final/issues/16> )
* User role management (<https://github.com/yashah9/IS601-Final/issues/8> )
* Nickname generation (<https://github.com/yashah9/IS601-Final/issues/18>)

**New Test Cases (10)**

* <https://github.com/yashah9/IS601-Final/blob/main/tests/test_minio.py>
* <https://github.com/yashah9/IS601-Final/blob/file-upload-testcase/tests/test_api/test_users_api.py>

**New Feature Implementation**

* Minio Feature (<https://github.com/yashah9/IS601-Final/issues/11>)

To implement the profile picture upload feature, I integrated **MinIO** as a secure and scalable object storage solution. I used the official minio SDK to interact with the MinIO server, leveraging environment variables (MINIO\_ROOT\_USER, MINIO\_ROOT\_PASSWORD) and application settings for client initialization. The configuration values — such as the endpoint, SSL toggle, and default bucket name — were loaded from a centralized settings.config module to maintain security and flexibility. A key part of the implementation was the ensure\_bucket\_exists(bucket\_name) function, which programmatically verifies and creates the required bucket if it does not already exist, handling exceptions using S3Error. For file uploads, the upload\_profile\_picture(file\_data, file\_name) function validates allowed image formats (.jpg, .jpeg, .png, .gif) before uploading the file using put\_object. Upon successful upload, it returns a direct file URL, while unsupported file types or upload errors raise meaningful exceptions. This setup not only ensured reliability and performance but also followed best practices in cloud-based file storage and error handling.

**🐳 Docker Deployment**

* DockerHub Repo:  
  <https://hub.docker.com/r/yashshah0910/is601_final_api>

Through this course, I developed skills that will stay with me throughout my software engineering career. I now understand the value of writing testable code, maintaining clean project organization, and deploying applications confidently. Beyond the technical fundamentals, Professor Williams also introduced us to innovative AI tools such as **Windsurf** for real-time backend prototyping and **Cursor** as an AI-augmented coding assistant. These demonstrations expanded my understanding of how AI can accelerate development, improve code quality, and support complex debugging workflows. By incorporating these tools into our learning process, he emphasized that staying current with emerging technologies is just as important as mastering foundational concepts. I’m grateful for the opportunity to apply what I’ve learned in a project that’s functional, test-covered, and successfully deployed to production.