### **User Management System Final Project Reflection**

**Student Name**: Varun

### **Project Overview**

The **User Management System** project was a thrilling opportunity to immerse myself in real-world software development practices. It provided hands-on experience in Quality Assurance (QA), Test Coverage, Feature Implementation, and Collaboration within a professional-grade codebase. My chosen feature was **Profile Picture Upload with Minio**, and I successfully integrated it while following industry best practices.

This project was a culmination of the concepts and techniques I learned throughout the course, blending technical and teamwork skills into one comprehensive coding adventure.

### **1. Quality Assurance (QA): Detecting and Fixing Bugs**

As part of the QA process, I identified and resolved **5 critical issues**, ensuring the robustness and reliability of the system. Each issue was documented and tracked using GitHub issues. Below are the details of the bugs and their resolutions:

1. **Duplicate Email and Nickname in Bulk User Creation**
   * **Issue**: Duplicate constraints caused failures during bulk user creation.
   * **Resolution**: Added database-level constraints and improved the validation logic for user creation.
   * **GitHub Issue**: [Issue #14](https://github.com/killuazoldyck7/user_management/issues/14)
2. **Role Downgrade During Email Verification**
   * **Issue**: Verified users were unintentionally downgraded to anonymous roles.
   * **Resolution**: Adjusted the email verification logic to preserve roles for authenticated users.
   * **GitHub Issue**: [Issue #10](https://github.com/killuazoldyck7/user_management/issues/10)
3. **Pagination Validation for list\_users Endpoint**
   * **Issue**: Invalid pagination parameters caused API crashes.
   * **Resolution**: Added validation for skip and limit parameters in the API.
   * **GitHub Issue**: [Issue #8](https://github.com/killuazoldyck7/user_management/issues/8)
4. **Missing User-ID in Email Verification Tokens**
   * **Issue**: Verification tokens lacked user IDs, causing failed verifications.
   * **Resolution**: Updated token generation to include user IDs.
   * **GitHub Issue**: [Issue #6](https://github.com/killuazoldyck7/user_management/issues/6)
5. **Dependency Conflicts in Python Setup**
   * **Issue**: Dependency mismatches caused build failures.
   * **Resolution**: Resolved conflicts and pinned dependency versions in requirements.txt.
   * **GitHub Issue**: [Issue #3](https://github.com/killuazoldyck7/user_management/issues/3)

### **2. Test Coverage Improvement**

To ensure comprehensive coverage, I reviewed the test suite and added **10 new test cases**. These tests focused on edge cases, error scenarios, and critical functionalities:

**Highlights of New Test Cases**:

* Validation for duplicate emails and nicknames in bulk user creation.
* Image format and size validation for profile picture uploads.
* Role-based access control tests for endpoints.
* Enhanced user account update tests to handle invalid inputs and duplicate data.

This effort significantly increased the reliability of the system by identifying and preventing potential edge-case failures.

### **3. Feature Implementation: Profile Picture Upload with Minio**

The **Profile Picture Upload with Minio** was my chosen feature for this project. It introduced personalized profile pictures for users, enhancing their experience.

#### **Feature Highlights**

* **New API Endpoint**: /upload/{user\_id} allows users to upload profile pictures.
* **Minio Integration**: Used Minio to securely store images and generate unique URLs.
* **Image Resizing**: Leveraged PIL to resize images to 200x200 pixels.
* **Validation**: Restricted uploads to valid image formats (JPEG, PNG) and file sizes under 2MB.
* **Profile Update**: Updated the user profile with the Minio-hosted picture URL.

#### **Technologies Used**

* **Backend Framework**: FastAPI
* **Object Storage**: Minio
* **Image Processing**: PIL (Python Imaging Library)
* **Database**: PostgreSQL with SQLAlchemy

**GitHub Feature Issue**: [Feature #20](https://github.com/killuazoldyck7/user_management/issues/20)

### **4. Challenges Faced and Key Learnings**

#### **Challenges**

1. **Database Constraints**: Resolving duplicate nickname and email issues required understanding database-level constraints and handling errors gracefully.
2. **Minio Integration**: Learning to integrate Minio with FastAPI while maintaining efficiency and scalability was a key challenge.
3. **Comprehensive Testing**: Writing edge-case tests for concurrent operations and image uploads required meticulous planning and effort.

#### **Key Learnings**

* **RESTful API Design**: Gained expertise in designing scalable APIs with FastAPI.
* **Async Database Operations**: Learned to handle SQLAlchemy's asynchronous operations effectively.
* **Testing Best Practices**: Developed skills in writing robust tests that simulate real-world usage scenarios.
* **Collaboration**: Improved my ability to document and resolve issues collaboratively via GitHub.

### **5. Future Enhancements**

While the project is feature-complete, there is scope for future enhancements:

1. **Profile Picture Management**: Add features to delete or replace existing profile pictures.
2. **Dynamic Image Cropping**: Allow users to crop and adjust images during uploads.
3. **Role-Based Notifications**: Notify admins of new profile picture uploads or verification requests.
4. **Audit Logging**: Implement logging to track changes to user profiles for compliance purposes.
5. **Enhanced Bulk Operations**: Improve the bulk user creation process to handle millions of records with transactional integrity.

### **6. Learnings Throughout the Course**

* Throughout this course, I developed a strong foundation in building and managing scalable web systems. I gained hands-on experience in implementing **RESTful APIs** using **FastAPI**, integrating cloud storage with **Minio**, and optimizing database operations with **SQLAlchemy**. I explored **design patterns** like the **Command Pattern** to improve code structure, implemented **multi-processing** for performance optimization, and enhanced **test coverage** using **pytest**. Mastering **Docker** for containerization and **GitHub Actions** for CI/CD automation enabled me to ensure smooth deployments. Additionally, I strengthened my **QA skills** by identifying critical bugs, validating fixes, and adhering to best development practices for robust, production-ready systems.

### **7. Deployment and Submission**

* **GitHub Repository**: [User Managemen-Final Project](https://github.com/killuazoldyck7/user_management)-Varun
* **DockerHub Repository**: [Docker Image](https://hub.docker.com/r/varunrahul/user_management)
* **Deployment Status**: Successfully deployed with CI/CD pipelines.

### **8. Conclusion**

This project was a rewarding journey that bridged the gap between classroom learning and real-world software development. From fixing bugs and improving test coverage to implementing new features, I gained valuable insights into modern software engineering practices. It has prepared me to excel in professional development environments and contribute meaningfully to team projects.