**Reflection on Learnings and Experience Working on the FastAPI Project**

Throughout the course of this project, I have gained a deeper understanding of various aspects of FastAPI, including dependency injection, role-based access control, and integration with external services. This journey has been both challenging and rewarding, offering numerous learning opportunities and the satisfaction of overcoming complex issues.

**Learnings and Key Takeaways**

**Dependency Management:**

* **Understanding Dependencies:** Efficient management of dependencies was crucial. Using dependencies.py, we defined critical dependencies like get\_settings, get\_email\_service, get\_db, get\_current\_user, and require\_role. This ensured our application components were loosely coupled and easily testable.

**Role-Based Access Control:**

* **Implementing Security:** We used the require\_role dependency to manage role-based access control, allowing only authorized users to perform specific actions. This enhanced our application's security and integrity.

**File Upload Handling:**

* **Validating and Processing Uploads:** Handling file uploads, especially profile pictures, required robust validation mechanisms. We implemented checks for allowed file types (JPEG, PNG, GIF) and leveraged FastAPI’s dependency injection for secure file uploads.

**Mocking and Testing:**

* **Ensuring Test Reliability:** One significant challenge was failing test cases related to file uploads. By mocking the MinIO client and requests to all external services, we isolated our tests from external dependencies, resulting in more reliable and consistent test outcomes.

**CI/CD Pipeline Configuration:**

* **Streamlining Deployments:** Configuring the CI/CD pipeline in production.yml was critical. Simplifying the Docker Compose configuration and removing unnecessary services like MinIO streamlined our deployment process and reduced potential points of failure.

**Implemented Feature**

**🌄 Profile Picture Upload with Minio**

* Detail: Implemented the /upload-profile-picture endpoint with role-based access control, file type validation, and dependency injection for secure and efficient file uploads.
* **GitHub Repository:** [**https://github.com/syedahmad2878/finalterm**](https://github.com/syedahmad2878/finalterm)
* **DockerHub Repository Name:** [**syedahmad2878/finalterm**](https://hub.docker.com/search?q=syedahmad2878%2Ffinalterm)

**Experience Working on the Project**

Working on this project has been invaluable, reinforcing my skills in backend development and system design. Integrating FastAPI with services like PostgreSQL and MinIO provided practical insights into building scalable and maintainable applications. Below are the issues addressed, new tests added, and a key feature implemented:

**Closed QA Issues**

1. **File Type Validation for Profile Picture Uploads:**
   * [**Issue Link**](https://github.com/syedahmad2878/finalterm/issues/7) **[**[**File Type Validation for Profile Picture Uploads**](https://github.com/syedahmad2878/finalterm/issues/7)**]**
   * Resolution: Implemented MIME type validation for uploaded files**.**
2. **Failed External Service Integrations in Tests:**
   * [**Issue Link**](https://github.com/syedahmad2878/finalterm/issues/6) **[**[**Failed External Service Integrations in Tests**](https://github.com/syedahmad2878/finalterm/issues/6)**]**
   * Resolution: Mocked MinIO client and requests to external services.
3. **Inconsistent API Responses Between Environments:**
   * [**Issue Link**](https://github.com/syedahmad2878/finalterm/issues/5) **[**[**Inconsistent API Responses Between Environments**](https://github.com/syedahmad2878/finalterm/issues/5)**]**
   * Resolution: Ensured consistent API response structures across environments.
4. **Database Connection Errors in CI/CD Pipeline:**
   * [**Issue Link**](https://github.com/syedahmad2878/finalterm/issues/4) **[**[**Database Connection Errors in CI/CD Pipeline**](https://github.com/syedahmad2878/finalterm/issues/4)**]**
   * Resolution: Refined database connection handling in CI/CD pipeline.
5. **Concurrent File Uploads Cause Server Crash:**
   * [**Issue Link**](https://github.com/syedahmad2878/finalterm/issues/3) **[**[**Concurrent File Uploads Cause Server Crash**](https://github.com/syedahmad2878/finalterm/issues/3)**]**
   * Resolution: Enhanced concurrency handling for file uploads.

**Added Test Cases:**

1. Test for JPEG Profile Picture Upload
2. Test for PNG Profile Picture Upload
3. Test for GIF Profile Picture Upload
4. Test for Invalid File Type Upload
5. Test for Empty File Upload
6. Test for Large File Upload
7. Test for Missing File Upload
8. Test for Missing Authorization
9. Test for Invalid User ID
10. Test for Insufficient Permissions

### Conclusion

This project has been an enriching experience, providing practical exposure to FastAPI’s powerful features and best practices in backend development. The challenges faced and solutions implemented have significantly enhanced my technical skills and understanding of building robust, secure, and scalable applications. I look forward to applying these learnings in future projects and continuing to explore the capabilities of FastAPI.

**DockerHub Repository:** syedahmad2878/finalterm

**GitHub Repository:** <https://github.com/syedahmad2878/finalterm>