Take-Home Assignment: File Upload

Objective

Develop a backend application that allows authenticated users to upload files of any size (hardware resources permitting).

Project Requirements

1. Backend Application

- **File Upload:**
  - Implement an API endpoint that allows users to upload files of any size (hardware resources permitting).
  - Efficiently handle large file uploads, make sure to prevent memory issues.
- **User Authentication:**
  - For simplicity, you can hard-code a set of valid usernames and passwords.
- **Dockerization:**
  - Containerize the backend application using Docker.
  - Ensure the Docker image is functional and can be built using provided instructions.
- **Testing (Mandatory):**
  - Write test cases covering critical backend functionalities.
  - Focus on testing file upload endpoint.

2. Client Application

- **Interaction with API:**
  - Build a simple client application interact with the backend API.
- **File Upload Interface:**
  - Provide an interface for users to select and upload files.
  - Display a list of uploaded files

- **Dockerization (Optional):**
    - If time permits, containerize the client application using Docker.

3. Orchestration with Docker Compose (bonus and only if time allows)

- **Docker Compose Setup:**
    - Create a `docker-compose.yml` file to orchestrate the backend (and client, if containerized).
    - Ensure services can be started together with a single command.

4. Testing

- **Backend Testing (Mandatory):**
    - Include test cases for critical backend functionalities.
    - Provide instructions on how to run the tests.
- **Client Testing (Optional):**
    - If time permits, include basic tests for client-side functionality.

5. Documentation and Instructions

- **Setup Instructions:**
    - Provide clear, step-by-step instructions on how to build and run the project using Docker Compose.
    - Include any prerequisites and environment variables needed.
- **Usage Guide:**
    - Explain how to authenticate and upload files.

Technical Specifications

- **Programming Languages & Frameworks:**
    - **Backend:**
    - Use a language and framework you're comfortable with, such as:
        - **Python with FastAPI**
        - **Go with Gin, Beego or FastHTTP**
        - **Node with express**
    - **Client:**
    - Use a simple web-based client with HTML/CSS/JavaScript - any front-end framework is allowed

- **File Storage:**
    - Store uploaded files in a directory within the Docker container, using Docker volumes for persistence.

Deliverables

1. **Source Code:**
    - Complete source code for the backend and client applications.
    - Code should be clean, well-organized, and follow best practices.
    - Include comments and documentation within the code where necessary.
2. **Docker Configuration:**
    - `Dockerfile` for the backend application.
    - `Dockerfile` for the client application (optional if time permits).
    - `docker-compose.yml` file to orchestrate the services.
3. **Documentation:**
    - **Setup Instructions:**
        - Detailed steps on how to build and run the application using Docker Compose.
        - Instructions on configuring environment variables and any necessary setup.
    - **Usage Guide:**
        - Instructions on how to use the app.
    - **Assumptions and Decisions:**
        - Document any assumptions or simplifications made to fit the time constraints.
4. **Test Cases (Backend Required):**
    - Include test scripts for backend functionalities.
    - Provide instructions on how to run the tests.

Evaluation Criteria

- **Functionality:**
    - The application meets the specified requirements.
    - Users can authenticate and upload files of any size.
    - Users are notified when new files are uploaded.
- **Code Quality:**
    - Code is clean, readable, and well-structured.
    - Proper use of design patterns and best practices.

    - Maintainability of the code is important.

- **Dockerization:**
    - Backend application is properly containerized.
    - Docker Compose setup works as described in the documentation.

- **Testing:**
    - Test cases cover critical backend functionalities.
    - Tests are easy to run and understand.

- **Documentation:**
    - Clear and comprehensive instructions are provided.
    - Easy for another developer to set up and use the application.

- **Scalability:**

Build the system in a way to add future features like:

- Real time notifications.
- User authentication.
- Add dark theme / light theme on the ui

- **Future maintenance:**

  In the second stage of this test, we will expect you to integrate one of the solutions mention on the previous point of scalability, make sure you come prepared for implementing the 3 of them if required.

Submission Instructions

- **GitHub Repository:**
    - Host your code in a **private GitHub repository**.
    - Add `@angvp` as a collaborator to the repository.
    - Ensure all necessary files are included and the repository is well-organized.

Notes

- **Time Management:**
    - Focus on implementing core functionalities first.
    - Use libraries or frameworks that aid in rapid development.
    - It's acceptable to make reasonable simplifications to meet the time constraints.

- **Assumptions:**
    - You may hard-code a set of valid user credentials for authentication.
    - Efficiently handle large files to prevent memory issues.
- **Testing:**
    - Prioritize writing test cases for the backend application.
    - Focus on testing file upload functionality, and file metadata storage.
- **Simplifications to Fit Time Constraints:**
    - Security as this won't be ever a production service please make sure to not spend much time here :)
    - Excluded file download functionality.
    - Simplified notification mechanism.
    - Optional client-side testing and Dockerization.

Bonus Points

- **Client Dockerization:**
    - Containerize the client application if time permits.
- **Implement a basic real time notification system:**
    - Once the file is uploaded let the other connected users that a new file was added.
- **Enhance security**
    - Add a basic auth system
    - Only allow uploads certain formats (.iso, .img)

Good Luck!

We look forward to reviewing your implementation. If you have any questions or need clarifications, please feel free to reach out.