Aycheek Technical Test

Overview

Develop a small, functional application to manage administrative documents for expatriates. This task is designed to evaluate your understanding of backend, frontend, and Docker basics in a realistic scenario.

Core Requirements

Backend (Spring Boot)

1. Essential REST APIs
    - **GET** `/api/documents` → Retrieve all user documents.
    - **POST** `/api/documents` → Upload a new document.
    - **GET** `/api/document-types` → Fetch document types.

2. Minimal Data Model

```
DocumentType {
  id: Long
  name: String // e.g., "Residence Permit", "Work Visa"
  validityPeriod: Integer // in months
}

Document {
  id: Long
  type: DocumentType
  issuedDate: Date
  expiryDate: Date
  status: String // VALID, EXPIRED
}
```

Key Notes:

- *Preload data for DocumentType (e.g., Residence Permit, Work Visa).*
- *Automatically calculate the status field based on expiryDate.*

Frontend (Angular)

1. Basic Views
    - Document List: Display uploaded documents in a table with:
        - Columns: Document Type, Issued Date, Expiry Date, and Status.
        - Status color coding: Green for valid, Red for expired.
    - Document Upload Form:
        - Input fields: Document Type (dropdown), Issued Date, Expiry Date.
2. Features
    - Simple, responsive design.
    - Status-based color coding (Green = Valid, Red = Expired).

Docker Setup

1. Create a Dockerfile for each service
    - Backend (Spring Boot):
        - Define the base image (e.g., openjdk:17).
        - Copy the compiled JAR file into the container.
        - Set the container's entry point to run the application.
    - Frontend (Angular):
        - Use an image like node:lts to build the application.
        - Use a web server image (e.g., nginx) to serve the static files.

2. Provide a basic docker-compose.yml to
    - Run a PostgreSQL database.
    - Start the Spring Boot backend.

Initial Data

```
INSERT INTO document_types (name, validity_period) VALUES
('Residence Permit', 24),
('Work Visa', 12);
```

Evaluation Criteria

1. **Backend**
    - REST API functionality.
    - Clear and logical data model.
    - Basic error handling.

2. **Frontend**
    - Clean and responsive UI.
    - Functional document upload and display.

3. **Docker**
    - Working setup for backend and database.

4. **Documentation**
    - Brief README with setup instructions.

Submission Requirements

- Backend with APIs.
- Frontend with minimal UI.
- Database script to initialize tables and data.
- Docker configuration to start backend and database.
- README:
    - Setup instructions.
    - Short explanation of your approach.

Additional Notes

- **Focus on simplicity and functionality.**
- **Partial submissions are acceptable and appreciated if time runs out. You can document incomplete parts in the README.**