HappyFox Interview Task

Functional Requirements

1. Email Fetching

FR1.1 Authentication

- **Objective**: Securely authenticate with the Gmail API using OAuth 2.0.
- **Details**:
  - Implement OAuth 2.0 flow to obtain access and refresh tokens from Gmail.
  - Store tokens securely and use them to authenticate API requests.
  - Handle token refresh process when access tokens expire.
- **Importance**: Ensures secure access to the user's email data while complying with Gmail's security standards.

FR1.2 Fetching Emails

- **Objective**: Retrieve a specific number of emails from the user's Gmail account.
- **Details**:
  - Fetch emails using Gmail API endpoints.
  - Allow configurable parameters such as the number of emails (as of now).
  - Process API responses to extract relevant email data.
- **Importance**: Provides the core functionality of accessing email data for processing, catering to different user requirements.

FR1.3 Database Storage

- **Objective**: Store fetched email data in a PostgreSQL database.
- **Details**:

    - Design an appropriate schema to store email attributes such as sender, recipient, subject, and content.
    - Use SQLAlchemy for Object-Relational Mapping (ORM) to interact with the database in Python.
    - Ensure efficient data insertion and retrieval.
- **Importance**: Enables persistent storage and efficient access to email data, which is crucial for processing and analysis.

FR1.4 Database Migrations

- **Objective**: Manage and apply database schema changes using Alembic.
- **Details**:
    - Set up Alembic for tracking database schema changes.
    - Write migration scripts for any changes in the database schema.
    - Apply migrations to update the database schema without data loss.
- **Importance**: Facilitates the evolution of the database structure alongside application development, ensuring data integrity.

FR1.5 Dockerized Database

- **Objective**: Utilize Docker Compose for running and managing the PostgreSQL database.
- **Details**:
    - Create a `docker-compose.yml` file defining the PostgreSQL service.
    - Configure database settings such as ports, volumes, and environment variables in Docker Compose.
    - Use Docker Compose to launch and manage the database environment consistently across different setups.
- **Importance**: Simplifies database setup and maintenance, ensuring a consistent and isolated environment for development and production.

2. Email Processing Based on Rules

FR2.1 Rule Parsing

- **Objective**: Interpret and process rules defined in a JSON file.
- **Details**:
  - Develop functionality to read rules from a JSON file, where each rule consists of conditions and corresponding actions.
  - Ensure robust parsing to handle various data formats and potential errors in the JSON file.
- **Importance**: Enables the application to be flexible and adaptable to different rule sets, making it customizable for various email processing needs.

FR2.2 Condition Support

- **Objective**: Evaluate emails based on specified conditions.
- **Details**:
  - Support conditions based on email attributes such as "From", "To", "Subject", and "Date Received".
  - Implement logic to assess these conditions against the attributes of each fetched email.
- **Importance**: Provides the fundamental mechanism for filtering and selecting emails based on user-defined criteria.

FR2.3 Predicate Support

- **Objective**: Implement various predicates for string and date fields in email attributes.
- **Details**:
  - Support string predicates like "Contains", "Does not Contain", "Equals", "Does not equal", and date predicates like "Less than", "Greater than".
  - Ensure accurate and efficient evaluation of these predicates against email attributes.
- **Importance**: Enhances the flexibility and precision of rule conditions, allowing for more targeted email processing.

FR2.4 Rule Evaluation

- **Objective**: Apply rules to emails using 'All' or 'Any' predicates.
- **Details**:
  - Implement logic to determine if an email satisfies all ('All') or at least one ('Any') of the conditions specified in a rule.
  - Process each email against the set of defined rules based on this logic.
- **Importance**: Dictates how rules are applied, offering versatility in handling complex email processing scenarios.

FR2.5 Action Execution

- **Objective**: Perform specified actions on emails that match rule conditions.
- **Details**:
  - Define actions such as "Mark as read", "Mark as unread", and "Move Message" to be applied to emails.
  - Execute these actions via the Gmail API based on the outcome of rule evaluation.
- **Importance**: Represents the operational outcome of the rule processing, directly impacting the user's email account based on specified criteria.

FR2.6 Logging and Error Handling

- **Objective**: Maintain comprehensive logs for system operations and handle errors effectively.
- **Details**:
  - Implement logging for key events and decisions in the email processing workflow.
  - Develop error handling mechanisms to manage and respond to exceptions, especially those involving external API interactions.
- **Importance**: Ensures transparency in the system's operation and resilience against failures, aiding in troubleshooting and reliability.

Non-Functional Requirements

1. **Performance**
    - Process and fetch emails efficiently, maintaining high responsiveness even under heavy loads.
2. **Reliability**
    - Ensure high system uptime and consistent performance across various operational scenarios.
3. **Usability**
    - Provide clear documentation and user-friendly interfaces for setup, configuration, and operation.
4. **Scalability**
    - Design the system to accommodate increasing loads and data volumes with minimal adjustments.
5. **Security**
    - Protect email data and authentication credentials, adhering to industry-standard security practices.
6. **Maintainability**
    - Follow best coding practices to facilitate easy maintenance, updates, and scalability of the application.
7. **Compatibility**
    - Ensure compatibility with current and future technology stacks, including Python versions, operating systems, and Gmail API updates.