Full Stack Engineer Technical Proficiency Test Script

DO NOT WRITE ON THIS SCRIPT

KHÔNG VIẾT LÊN ĐỀ NÀY

(To be returned)

1. Test Summary

Secure email authentication practices are a critical component of an organization's information security posture on the internet. Firms often verify their domain names to ensure their email authentication records (such as DMARC, SPF, and DKIM) in DNS and other sources are correctly configured and valid. You can learn more about these email security protocols here: https://www.cloudflare.com/learning/email-security/dmarc-dkim-spf/

For this technical test, you are required to develop a simple web application that enables clients to log in and test their domain names for common email security configuration issues. A good example of a web application that performs DMARC tests is: https://mxtoolbox.com/dmarc.aspx

2. Hard Requirements

The web application **MUST** comprise a NodeJS backend, a ReactJS frontend, and a Python script. All source code should be organized and stored in a GitHub repository that is accessible to us for review.

PYTHON SCRIPT:

The Python script will be responsible for performing domain record validations.

- **Functionality:**
  - Validate the following email authentication records in DNS: DMARC, SPF, and DKIM.
  - For each check, return a `pass` / `fail` status along with any specific issues found.
- **Integration:**
  - The Python script will be invoked by the NodeJS backend when a client requests a domain check from the ReactJS frontend.
- **Sample:**
  - *(Note: A sample Python script with commonly used libraries is assumed to be provided separately by the test giver.)*

NODEJS BACKEND:

The backend application will serve as the API layer for the frontend and interact with the Python script.

- **Request Handling:**
  - Handle requests from the ReactJS frontend for:
    - Login (user authentication)
    - Domain Check: Forward the domain request to the Python script and return its response to the frontend.
- **API Documentation:**
  - Provide a Swagger / OpenAPI specification endpoint accessible at `/api/docs` for clear API documentation.
- **Data Persistence:**
  - For simplicity, user/password records and a history of completed domain checks can be stored in a simple JSON file.

REACTJS FRONTEND:

The frontend will provide the user interface for interacting with the application.

- **Login Page:**
  - Implement a page for user authentication using a username and password.
- **Dashboard Page:**
  - **Domain List:** Display a list of previously saved domain names.
  - **Add New Domain:** Provide functionality to add a new domain name for testing.
  - **Domain Details:** Show detailed results for each domain, including:
    - The domain tests performed.
    - The results of each test (e.g., DMARC: Pass, SPF: Fail - reason, DKIM: Pass).

3. Bonus Requirements

Meeting these requirements will demonstrate a higher level of proficiency and adherence to best practices.

- **Database:**
  - Utilize a PostgreSQL database for data persistence instead of a local JSON file.
- **Containerization:**
  - Include a Dockerfile for containerizing the application components.
- **CI/CD:**
  - Implement a Continuous Integration/Continuous Deployment (CI/CD) pipeline using GitHub Actions or a GitHub Runner.

4. Submission Guidelines:

- Please ensure all code is committed to a GitHub repository and that the repository is accessible for review.
- Include clear instructions on how to set up, run, and test the application in the repository's README file.