PYTHON Developer Assessment Exam

1. Test Overview

Goal:
Build a *minimal viable prototype* (MVP) for a basic "Seeker-Provider" matching application. Candidates should showcase their proficiency in back-end, front-end, database modeling, basic matching logic, security considerations, and documentation.

Scope of the Mini-Project:

- **User Registration & Authentication**
    - Two roles: Seeker and Provider
    - Basic session management or token-based authentication (JWT or similar)
- **Profile Management**
    - Seeker: For instance, "industry preference," "location," or "credit rating."
    - Provider: e.g. "industry focus," "services offered," "location."
- **Matching Functionality**
    - A simple matching approach that suggests a set of Providers for a Seeker (and vice versa) based on common fields (e.g. location, industry).
- **Basic Front-End (or API + minimal front-end)**
    - A simple UI or Postman scripts demonstrating the create/update flow for each user type.
- **Deployment**
    - Dockerfile or instructions for spinning up the app locally (or on a sandbox environment).

Timeframe:
1–5 days to complete this assignment.

---

2. Test Instructions to the Candidate

1. **Repository Setup**

- Create a public or private repository (GitHub, GitLab, etc.).
- Initialize with a clear **README** that explains setup steps, dependencies, and usage.

2. **Technical Stack**
    - Back-End: Preferred frameworks (Node.js/Express, **Python**/Flask, .NET, etc.).
    - Front-End: Minimal single-page application (**React**, Vue, Angular) or a simple HTML/JS approach.
    - Database: Use a lightweight DB (SQLite, PostgreSQL, MongoDB—candidate's choice). Provide instructions for schema creation or migrations.
    - Matching Logic: Simple algorithm or function that ranks or filters Providers for a given Seeker.
    - Authentication & Security: Token-based or session-based login. Must demonstrate minimal password handling and role-based checks.
3. **Functional Requirements**
    - Sign-Up & Login:
        - Seeker: Email, password, industry preference, location, etc.
        - Provider: Email, password, service focus, location, etc.
    - Edit Profile: Either user type can update relevant fields.
    - View Matches (Seeker → Providers): Display or return a list of relevant Providers.
    - (Optional) Analytics or Logging: A minimal stats page or simple logs demonstrating user actions, to show how they'd track usage or activity.
4. **Deployment & Delivery:**
    - Docker: A simple Dockerfile and possibly a docker-compose.yml so the app can be run in one or two commands.
    - README: Must include precise instructions (e.g., "Run docker-compose up" or "npm install && npm start" for local dev).
    - Bonus: If they can deploy to a free Heroku instance, Netlify, or other, they can provide a URL for easy testing.
5. **Documentation:**

- Outline system architecture (1–2 paragraphs).
- Provide basic flow diagrams (optional but encouraged).
- Summarize known limitations or how you'd enhance the MVP if given more time.

Steps to Submission

- Create a quick Loom video (3–5 minutes) where you will demo the usage.
- Please provide a **GitHub link** or a **ZIP file** containing the code, along with a **README file** that includes setup instructions. *(Optional, but recommended for code quality assessment.)*