Recruitment Application

The objective of the below exercise is to create an application with the following services

- external-recruitment-service
- Internal- recruitment -service

Develop the application in Python/Flask with the below things in mind

- Follow all the clean code principles
- Write automated Unit Test cases using requests package & pytest
- Implement Logging & Authentication wherever applicable
- Wherever there are questions make a reasonable assumption on the requirements and comment in the source code as to what assumption you are making.

<u>External-recruitment-service</u>

Below are the endpoints to be created in external-recruitment-service

1. Endpoint: GET **/profile-for-skillset** with options to pass the required skillsets as query parameter or as JSON data. The end point should return the profiles that matches the required skillset with % match

2. Endpoint: POST **/post-requirement** Input JSON below

   {
   “candidateId”: 1,
   “fullname”: “Joel Oram”,
   “skillsets”: [“java”, “spring-boot”, “docker”, “kubernetes”]
   }

   Expected Result: Store the given input in a database for later reference. Respond with appropriate messages keeping in mind the best practices of HTTP.

3. Endpoint: POST **/match-requirement** Input JSON below
   {
   “requirementId”: 1,
   “position”: “developer”,
   “requiredSkillsets”: [“java”, “kubernetes”, “docker”, “spring-boot”]
   }
   Expected Result: Match the required Skillsets with the available skillsets in the data store / database and return candidates who match the skillsets.

Response:
[
{
“candidateId”: 1,
“fullname”: “Joel Oram”,
“skillsets”: [“java”, “spring-boot”, “docker”, “kubernetes”]
}
]

4. Endpoint: PUT **/update-profile** with option to pass the details to be updated as JSON data. The end point should update the profile and give appropriate responses
5. Endpoint: DELETE /remove-profile to mark the status of the profiles as not to be considered. Marked should not be used in further matches and a the system should not allow such a profile to be reposted.

<u>Internal-recruitment-service</u>

Below are the 2 endpoints to be created in **internal-recruitment-service**

1. Endpoint: POST **/post-requirement** Input JSON below
{
“candidateId”: 1,
“fullname”: “Joel Oram”,
“skillsets”: [“java”, “spring-boot”, “docker”, “kubernetes”]
}

Expected Result: Store the given input in a database for later reference. Respond with appropriate messages keeping in mind the best practices of HTTP.

2. Endpoint: POST **/match-requirement-optional** Input JSON below
{
“requirementId”: 1,
“position”: “developer”,
“requiredSkillsets”: [“java”, “spring-boot”],
“optionalSkillsets”: [“docker”, “Kubernetes”]
}

Expected Result:

Match the required Skillsets and optional Skillsets with the available skillsets in the data store / database within the internal recruitment service and return back if such a candidate exists.

If there are no candidates in the first check then match required SkillSets alone with the available skillsets (by leaving out the Optional Skillsets) and return back if such a candidate exists
If both the match operations above do not return a candidate then call the external-hiring-service to get the candidates that match both the required Skillsets and optional Skillsets and return the candidates with matching skillsets.

• Respond with the list of candidates as below.

```
[
{
“candidateId”: 1,
“fullname”: “Joel Oram”,
“skillsets”: [ “java”, “spring-boot”, “docker”, “kubernetes” ]
}
]
```