Acronis

Interview Programming Assignment
Candidate: Adaickalavan Meiyappan

Instructions to Candidate

- Candidate is free to use any programming language, though **Python** or **GoLang** is preferred.
- Candidate is free to use any library or framework of their choice
- Candidate's submission is expected to have the following:
    - **Requirements.txt** file listing all dependencies
    - **Readme** file:
        - Detailing the setup procedure to run the code
        - Steps needed to actually run and test the code
        - Any assumptions that candidate has made w.r.t solving the assignment
    - **Secrets** file listing any secrets like passwords, API keys etc. that the candidate has used.
- It is recommended that candidate's code (i.e. code written by the candidate) **not be mixed** with any boilerplate, auto generated or framework generated code. This ensures that candidate's code can be judged independently without any "noise" from other sources.
- Candidate is encouraged to submit code via **GitHub**
- This is a pure **backend** assignment, candidate is not expected to develop any UI/UX/Frontend elements

Acronis

Interview Programming Assignment
Candidate: Adaickalavan Meiyappan

Programming Assignment

<u>Problem to solve:</u>

1. A file ***words.txt*** will be given containing 50-60 English words
2. **Programmatically** read the file and remove any duplicate words
3. **Programmatically** call a **third party REST API** to get the English definition for each word:
    1. If multiple definitions are returned by the API, select the **first** definition and discard the rest.
    2. Any **<u>free</u>** third party dictionary service can be used, e.g. https://www.wordsapi.com/ (2500 free requests per day)
4. Store the words along with their definitions in a **key-value/NoSql** data store. E.g. MongoDB.
5. Expose an **endpoint** to retrieve the definition as **JSON** over an **HTTP request**. Candidate is free to structure the request and response as per their preference. An example is as follows:
    1. Request → /definition/?word=Hello
    2. Response → 200, { definition: “The definition of the word Hello” }

Bonus:

1. Containerize and run the entire application using **Docker**
2. Deploy the container in any **cloud** service and make the endpoint accessible publically
3. Test cases

How will candidate’s output be judged?

1. Working code satisfying **all** the given requirements. If some requirement is not met, candidate is to justify why so.
2. Code **reliability**. E.g.:
    1. Handling of corner cases
    2. Bug and error free
3. Code **hygiene**. E.g.:
    1. Structure
    2. Readability
    3. Comments
4. Code **sophistication**. E.g.:
    1. Multi-threading
    2. Usage of proper data structures and algorithms

-------------------------------------- END --