ARCTIC SHORES
Data-Driven People Insights

Python/Web Programming Exercise

Introduction

This programming exercise is designed to take 60-120 minutes. The aim is to demonstrate programming approach as a priority over finishing it!

Either email back a zipped project folder containing your source files, or a link to a git repository containing your work.

The Task

You will create a small Python 3 web app backed by a sqlite database accessed via an ORM. Please create a Django app if you are familiar with Django, however if you wish to use a different web framework which you're more familiar with (e.g. Flask, FastApi, web2py) and ORM (e.g. SQLAlchemy, Peewee, PonyORM) then that's also fine :)

Bootstrap and setup your code structure according to best practice for your chosen web framework before proceeding.

STEP 1

Create two database ORM models / classes:

a) **Candidate**: for storing candidate details. Fields are:

- Name: there is no restriction on the format of the name.
- Candidate Reference: this is a sequence of 8 letters and digits.

b) **Score**: for storing test scores for candidates. Fields are:

- Score: must be a float between 0-100 inclusive.

Note that a single candidate can have multiple scores as each candidate is allowed to take the test multiple times.

STEP 2

Write the code to generate the database tables in an sqlite db corresponding to the models created STEP 1.

STEP 3

Write a python utility module containing two functions which will:

a) Read a CSV file such as the attached (candidates.csv) into to the system to add candidates and scores to the DB. Please note that ome candidates completed the test more than once so have several scores. And a couple of entries have invalid data and these need handling/ignoring.

b) Read the attached JSON file (candidates.json) and write out a CSV file like the attached (candidates.csv) with candidates ordered by score.

STEP 4

If you have time...

Create a view which will display an HTML page at path "/candidates" with an html table containing:

a) The list of all candidates in the DB sorted alphabetically by name.

b) Beside each name, print all the candidate's scores in ascending order.

c) Highlight the candidate with the highest score.

N.B. The page should look neat, but please don't spend time making it look pretty!