Ansaro
HIRE SMARTER

# Backend Coding Challenge

As a workforce prediction company, we think it's fitting that our backend coding challenge focus on workforce data. Your task is to build a web application that accepts, stores, and summarizes job applicant data per the specification below.

Successfully completing the challenges requires configuring a database to store 3 objects, creating 3 API endpoints that interact with that database, writing a script to populate the database with dummy data, and sharing your code with us.

You are welcome to use whatever languages, frameworks, databases, etc. you want to complete this challenge. Please don't hesitate to reach out to sam.stone@ansaro.ai if you have any questions!

## ***Objects***

Each database object should have the fields and field types as specified below:

Object 1: **Job**

- Title: string
- Location: string
- Description: string

Object 2: **Applicant**

- Job: link to **Job** object
- First_name: string
- Last_name: string
- Email: string or email (depending on database format)
- Status: string, restricted to the choices below:
    - Under review
    - Rejected
    - Offer pending
    - Offer rejected
    - Offer accepted – yet to start
    - Started

Object 3: **Experience**

- Applicant: link to **Applicant** object

- Category: string, restricted to choices below
    - Education
    - Work
- Start: datetime
- End: datetime
- Description: string
- Institution: string

***Endpoints***

Each API endpoint should have the URLs, HTTP methods, and return types as specified below:

Endpoint 1: **api/jobs/{id}/**

- Accepted methods: GET, POST, PATCH
- This endpoint can be used to create, view, or update a **Job.** We will leave the specifics of what each call should return to you.
- In addition to handling the HTTP methods above, your endpoint should take the additional URL argument “summary” for GET requests (**api/jobs/{id}/summary/**) and return data formatted like the example JSON response below:

```
{
        “applicant_count”: 8,
        “applicants_under_review”: 5,
        “applicants_offer_pending”: 2,
        “most_common_institution”: “Ansaro”,
        “average_work_experience_count”: 3.7,
        “average_years_of_work_experience”: 4.8,
}
```

Endpoint 2: **api/applicant/{id}/**

- Accepted methods: GET, POST, PATCH
- This endpoint can be used to create, view, or update an **Applicant.**
- Your GET response should be formatted as follows – note that this response displays data from the **Experience** objects associated with the **Applicant** object:

```
{
        “Job”: “Sales Rep”,
        “First_name”: “Mike”,
        “Last_name”: “Liu”,
        “Email”: “mike.liu@test.com”,
        "Status": "Under review",
        "Experiences": [
                {
                        "Category ": "Education",
                        "Start": "September 1, 2000",
                        "End: "June 1, 2004",
                        "Institution": "University of California”,
                        “Description”: “I attended UC Berkeley, where I majored in CS and
                                        minored in underwater basket-weaving.”
                }, {
                        "Category ": "Work",
                        "Start": "August 1, 2004",
                        "End": "June 30, 2010",
                        "Institution": "Hooli”,
                        “Description”: “I worked as an engineer on the Hooli search team.”
                },
        ]
}
```

Endpoint 3: **api/experience/{id}/**

- Accepted methods: POST
- This endpoint can be used to create a new **Experience;** it has no return type other than an HTTP code (200, 400, etc.)

## *Script*

Please write a script that creates dummy data for 3 jobs, 100 applicants, and 1-10 experiences per applicant (applicants should have different numbers of experiences.)

## *Submission*

Please upload your code to Github and tell us where to find it. You should include a **Readme** that instructs us on how to configure and run the application locally (including how to run the script that generates the dummy data.)

If your API has a browsable interface, we will use that; otherwise we’ll test it using Postman or Curl.