## Introduction

The goal of this project is to assess your skills in generating embeddings from text data and using Milvus, a high-performance vector search engine, to detect duplicate entries. The task involves working with job postings data containing job descriptions. Furthermore, we would like to see your ability to present the results using Docker or Docker Compose, emphasizing reproducibility.

**Note: Please include a 2 min video to demo your work**

## Part 1- Data Preprocessing

Start by a brief exploratory data analysis (EDA). Understand the structure and types of data available, and any possible anomalies or outliers. Clean the data, handling any missing values, incorrect entries, or anomalies you identify.

You will be provided with a dataset of job postings. Each entry in the data will contain:

Job ID, Job Title, Job Description, Company Name, Location, Posting Date, Source etc.

## Part 2 - Generating Embeddings

1. Determine an appropriate model to generate embeddings for the job descriptions. It can be pre-trained models like Sentence Transformers, GloVe, FastText, Word2Vec, Doc2Vec, CBOW, skip-gram, or any other approach you think would be best suited.
2. Justify your choice: Explain why you chose this method and its potential advantages for this task and its potential advantages in the context of duplicate detection.
3. Generate embeddings for each job description in the dataset.

## Part 3: Implementing Milvus for Duplicate Detection

1. Set up a Milvus instance.
2. Insert the embeddings generated from Part 2 into Milvus.
3. Implement a method to search for potential duplicates within the Milvus system.
4. Evaluate the results. How effective is the method in detecting duplicates?
5. How did you decide on the threshold for determining duplicates in Milvus? Which metrics are you using?
6. Can you think of other use-cases for the embeddings you generated, beyond duplicate detection?

## Part 4: Docker/Docker Compose Integration

1. Containerization:
   1. Create a Dockerfile to encapsulate your environment, dependencies, and code.
   2. Ensure data is accessible to the container, either by mounting a volume or including it in the image (not recommended for large datasets).
2. Docker Compose: If you have multiple services (e.g., a database, a backend service, etc.):
   1. Define each service in a docker-compose.yml file.
   2. Ensure proper networking between services.
3. Instructions: In your README:
   1. Detailed steps to build and run the Docker image/container.
   2. Explain any required environment variables or configurations.
4. How would you handle real-time data streaming into this system?