Case Study for Position of Principal Software Engineer

Background: National Community Investment Fund (NCIF) is developing an AI-based platform that integrates data from 21 diverse datasets, including public/private structured and unstructured data. The platform's goal is to democratize data and enable advanced analytics for stakeholders like investors, financial institutions, and community organizations. By leveraging AI/ML capabilities, we aim to create impactful insights that drive decision-making in underserved and underrepresented communities.

Goal: The candidates will demonstrate their ability to design and implement a backend API to integrate data, handle user authentication, and provide secure and efficient data retrieval using natural language programming (NLP). For the purpose of this Case Study, we are sharing an extract of 3 databases (see below).

Task

1. **Initial thoughts on Platform Architecture:** Provide a schematic and supporting 1-page proposal on structuring a scalable backend architecture to (a) support NCIF's multi-source data integration, (b) adhering to AWS best practices (c) ability to handle structured and unstructured data (d) user authentication and (e) use of Large Language Models.

   Note: our data is currently housed in AWS S3 database.

2. **Data Integration:** Load a provided dataset into an SQLite database. The dataset includes the following for the State of IL:

   a. EPA Air Quality Data
   b. FFIEC Summary of Deposits (SOD)
   c. NCUA Credit Union Data.

   We propose using two Master Keys for connecting databases - Census Tract for geospatial connections and FDIC certificate numbers (CERT) for operational connections.

3. **Business Case Queries and Visualization**
   a. Compute the density of bank and credit union branches by census tract
   b. Identify census tracts with PM2.5 > 10 and with more than 5 branches of banks and credit unions. Is there a correlation between the variables? Show this in a table and chart form.

4. **API Creation:** Demonstrate how you will use API and SQL queries to:

    a. Aggregates data from multiple sources (SOD, NCUA, and EPA) to compute the density of branches per Census Tract, categorized by air quality levels. Can this handle unstructured data?

    b. Supports dynamic querying. Users should be able to specify conditions like "Census Tracts with PM2.5 above 15 and more than 5 branches." The system should parse the query, access relevant datasets, and return results dynamically.

    c. Uses machine learning to predict the likelihood of increased air pollution (PM2.5 levels) in Census Tracts based on branch density and historical pollution data. Train a simple model using provided historical data from EPA.

    d. Add an NLP interface where users can input natural language queries (e.g., "Show me all tracts with above-average air pollution and a bank branch") and retrieve results.

Deliverables

- **Presentation**: A 3-5 slide presentation summarizing the approach, API structure, and results for the queries.

- **Code Repository**: A repository with:
    - Source code.
    - README file outlining the setup process, assumptions made, and steps followed.
    - Any additional documentation or insights from their approach.