Project Proposal: NLP-to-SQL Translator

1. Project Overview

The **NLP-to-SQL Translator** aims to simplify the process of querying an SQL database by allowing users to speak or write queries in natural language. The system will leverage a pre-trained language model to understand natural language inputs, generate appropriate SQL queries, and execute them on a connected database to retrieve desired information. This project will demonstrate how users can interact with their databases without needing in-depth SQL knowledge, allowing for more intuitive data retrieval.

2. High-Level Software Architecture

- **User Interface (UI) Layer:**
  - **Frontend:** A simple web interface or command-line tool where users can input natural language queries.
  - **Backend Interface:** Communicates with the LLM and the database. Receives user input, passes it to the LLM, and returns results to the UI.
- **Language Processing Layer:**
  - **LLM Integration (OpenAI API or custom model):** This layer takes the user's natural language input and translates it into SQL. The language model (like GPT) should be fine-tuned to understand specific domain language and the database schema.
- **Database Interaction Layer:**
  - **Database Schema Representation:** This layer holds the schema of the SQL database, which is provided to the LLM as context to assist in query translation.
  - **SQL Execution Engine:** This module receives the generated SQL query, connects to the database, and executes the query to retrieve results.
- **Result Processing and Output:**
  - After the SQL query is executed, the results are processed and formatted for easy viewing on the frontend.

---

3. Prototype Development Steps

Step 1: Setup a Simple SQL Database

- Create a simple SQL database manually that can be run locally, such as using SQLite. Example schema:

```
CREATE TABLE Employees (
    EmployeeID INTEGER PRIMARY KEY,
    Name TEXT,
    Department TEXT,
    Salary INTEGER
);

CREATE TABLE Departments (
    DepartmentID INTEGER PRIMARY KEY,
    DepartmentName TEXT
);
```

- Insert sample data:

```
INSERT INTO Employees (Name, Department, Salary) VALUES
('Alice', 'HR', 5000);
INSERT INTO Employees (Name, Department, Salary) VALUES ('Bob',
'Engineering', 7000);

INSERT INTO Departments (DepartmentName) VALUES ('HR');
INSERT INTO Departments (DepartmentName) VALUES ('Engineering');
```

Step 2: Connect to OpenAI API (for NLP Processing)

- Set up the OpenAI API integration, which takes the user's input and the schema as context. An API call may look like this:

```
import openai

# Define the database schema context
schema = """
Employees table: EmployeeID, Name, Department, Salary
Departments table: DepartmentID, DepartmentName
"""

# User input: Natural language query
user_query = "Show me the names of employees in the HR
department"

# OpenAI API call
response = openai.Completion.create(
    model="text-davinci-003",
    prompt=f"Given the database schema: {schema}, translate the
user's request: '{user_query}' into an SQL query.",
    max_tokens=100
)

sql_query = response.choices[0].text.strip()
print(sql_query)  # SQL query generated by the LLM
```

Example expected output:

```
SELECT Name FROM Employees WHERE Department = 'HR';
```

Step 3: Execute the SQL Query on the Local Database

- Use a local database connection, such as SQLite in Python, to execute the generated SQL:

```
import sqlite3

# Connect to the database
conn = sqlite3.connect('example.db')
cursor = conn.cursor()

# Execute the SQL query from the LLM
cursor.execute(sql_query)
results = cursor.fetchall()

# Process the results
for row in results:
    print(row)

conn.close()
```

Step 4: Display the Results to the User

- If this is a web-based interface, the results can be formatted and returned in a simple HTML table or as text output in a terminal for CLI-based applications.

Summary

This proposal outlines the process of creating an NLP-to-SQL Translator. The high-level architecture includes a UI for user interaction, an LLM for natural language processing, and a database layer for executing SQL queries. The prototype steps involve creating a local SQL database, integrating with the OpenAI API to generate SQL queries from natural language inputs, and running those queries to retrieve data from the database.