LLM Agents for SQL / Pandas Query Generation

Group Members:

Abhishek Sureddy
Akshay Kumar Sureddy
Avinash Reddy Vasipalli
Muhammad Yusuf Hassan
Dhanus Kanth Anand

Objective

- Web-based tool to convert user inputs to relevant SQL/ Pandas queries
- Leverage Large Language Models (LLMs) for code generation

Motivation

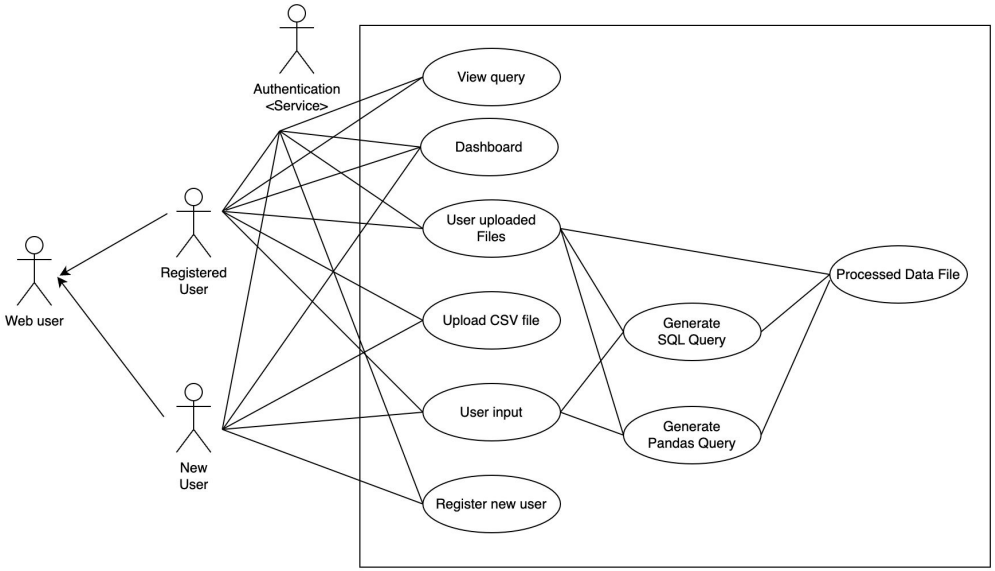
1. **Non-technical users**
   Enable those without coding knowledge to perform complex data analysis tasks easily.

2. **Data analysts**
   Help data analysts streamline their workflow by generating SQL/Pandas queries.

3. **Efficient querying**
   Generate optimized queries to handle large datasets more effectively.

4. **Time-saving tool**
   Reduce the time spent on repetitive query writing, especially for common data operations.

Use Case Diagram



Use Cases

1. User Authentication:

- Users can register and login to the application
- Only registered users can upload/ view/ delete their files

2. Upload Data:

- User uploads an Excel file via the web interface.
- The system checks the file's format (CSV).
- If the format is valid, the system processes the file, else it requests a correct file from the user.

3. Enter Natural Language Query:

- The user types a query or analysis request using everyday language
- The system leverages LLM to understand the user's input and translate it into a technical query format (like SQL or Pandas).

Use Cases

4. Generate and Execute Query:

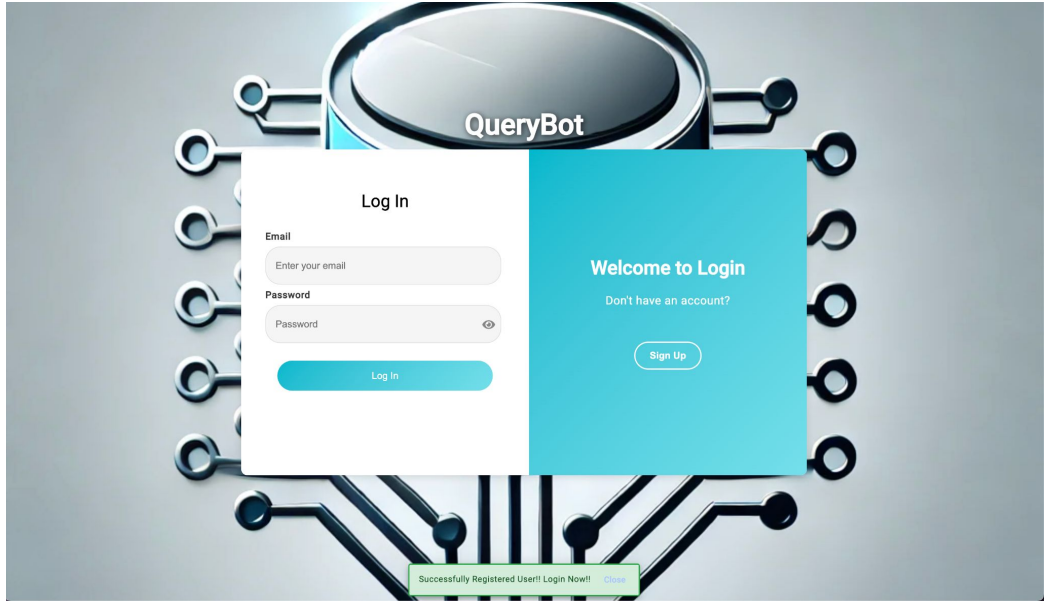
- We prompt the LLM with user input and a few rows from the data.
- LLM generates the SQL/Pandas query and check for vulnerabilities and executes it on the uploaded data if it is safe.

5. Display/ Export Results:

- The user can view and copy (to clipboard) the LLM-generated query.
- The processed data will be displayed onto the screen so that the user can see the results.
- The user can export and download the processed data in csv

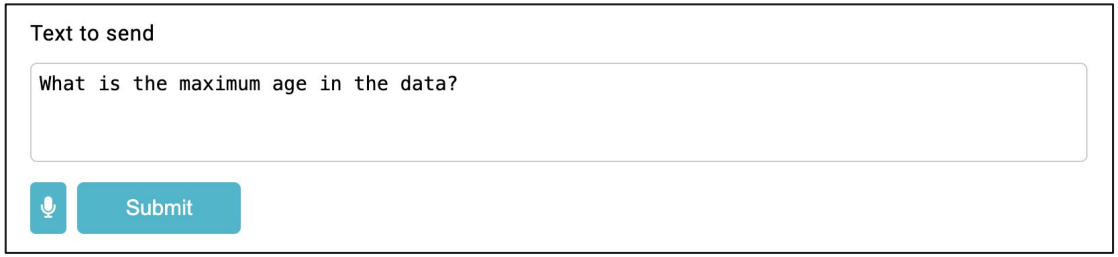
High-Level Features

1. **User Authentication:** Users can register and login to the application



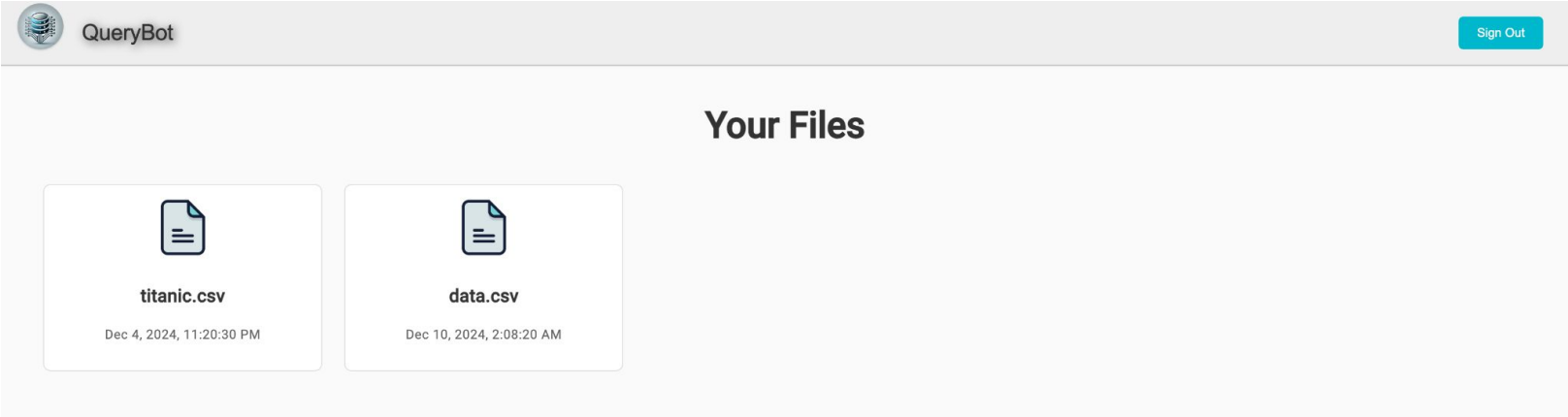
High-Level Features

2. Natural Language Interface: Accepts queries natural language queries in English, through text and voice based inputs



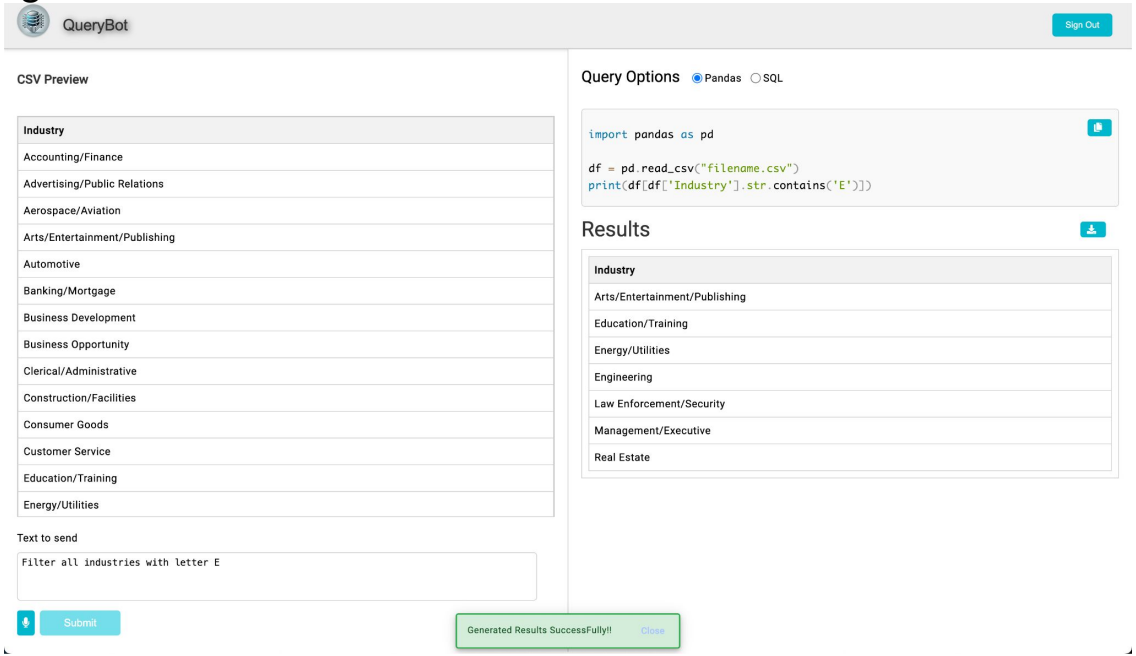
High-Level Features

3. **User Spaces:** User can view all their uploaded CSV files, upload new files, and delete any existing files.



High-Level Features

4. **Query Generation:** Converts user inputs into SQL or Pandas queries on the given CSV file using LLM Agents



High-Level Features

5. **Data Processing:** Returns downloadable results in CSV format

Results

Industry
Arts/Entertainment/Publishing
Education/Training
Energy/Utilities
Engineering
Law Enforcement/Security
Management/Executive
Real Estate

Project Flow

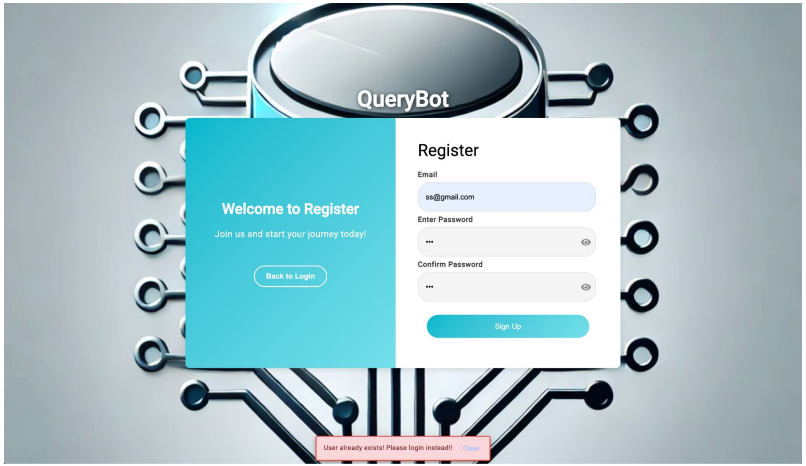


Figure 1: SignUp Page

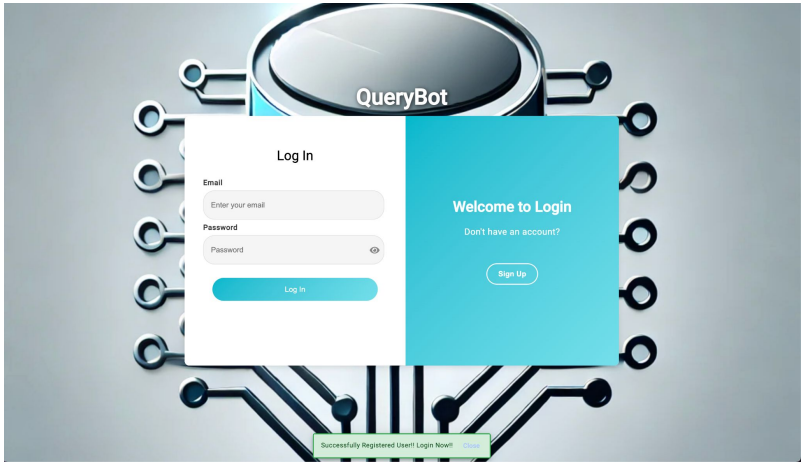


Figure 2: SignUp Successful

Project Flow

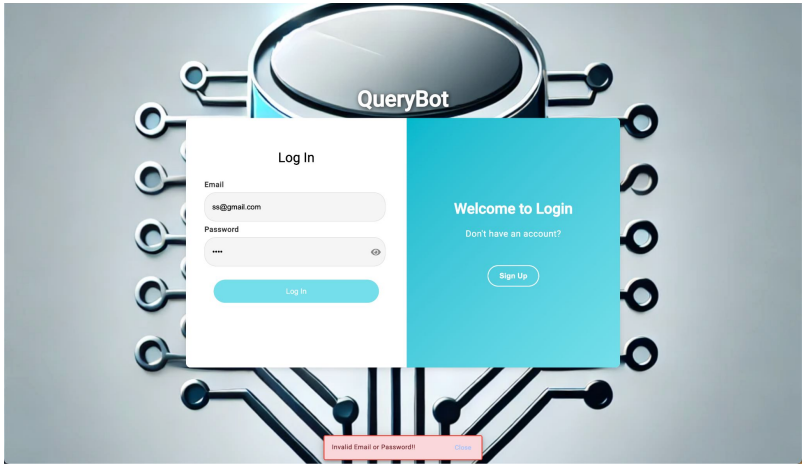


Figure 3: Login

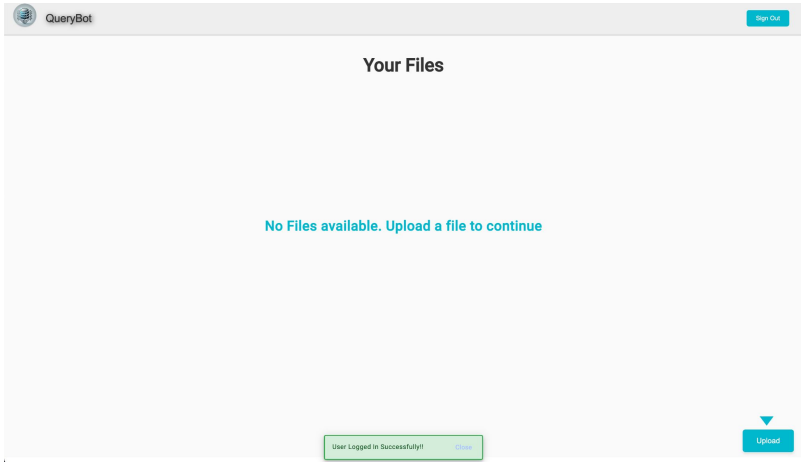


Figure 4: Dashboard

Project Flow

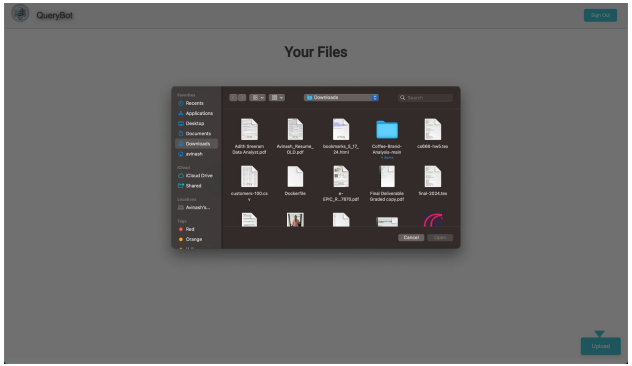


Figure 5: Upload

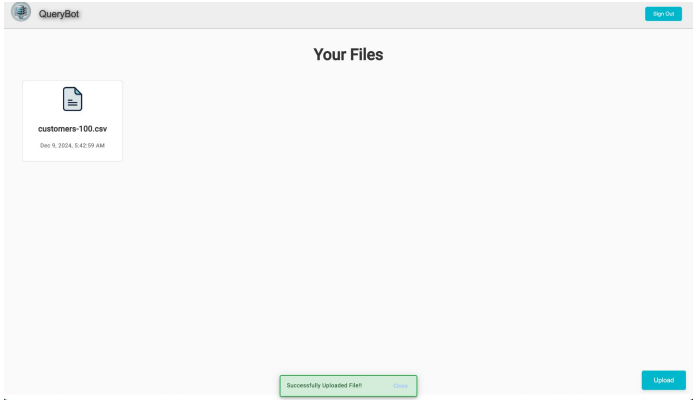


Figure 6: Upload Successful

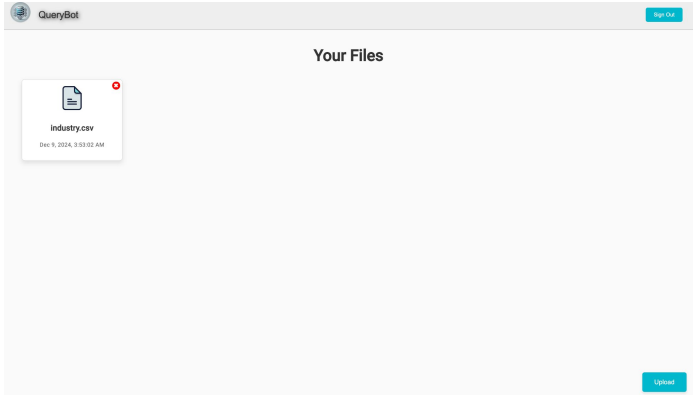


Figure 7: Upload Tile

Project Flow

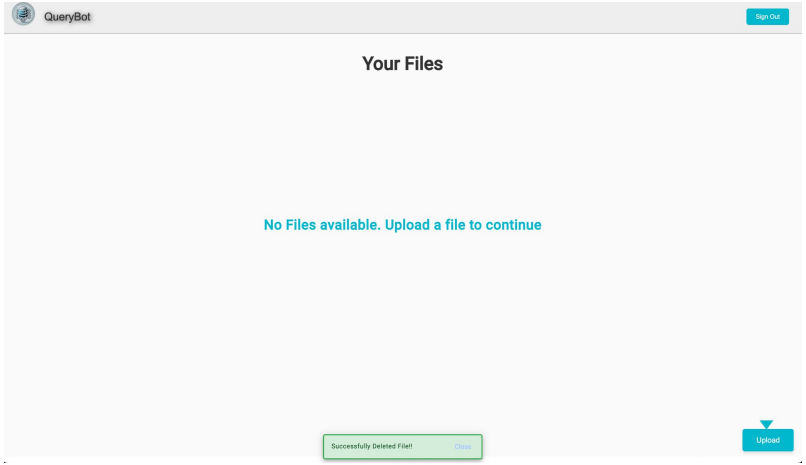


Figure 8: Delete File

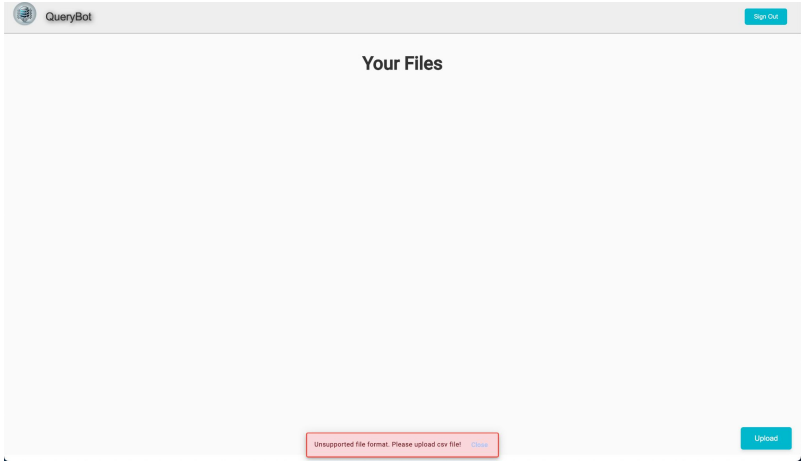


Figure 9: Unsuccessful Upload

Project Flow

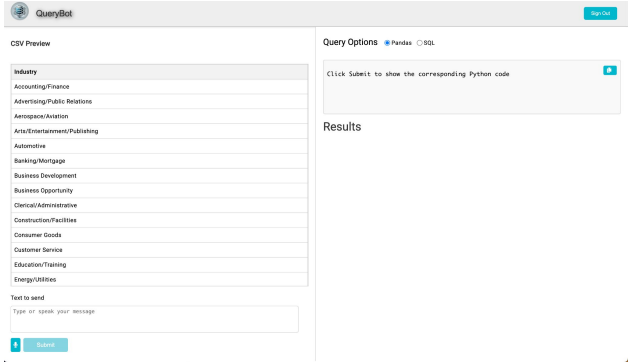


Figure 10: File Upload Page

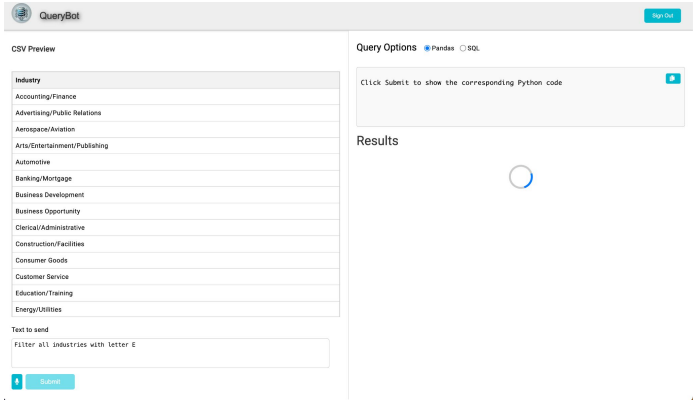


Figure 11: LLM Calling

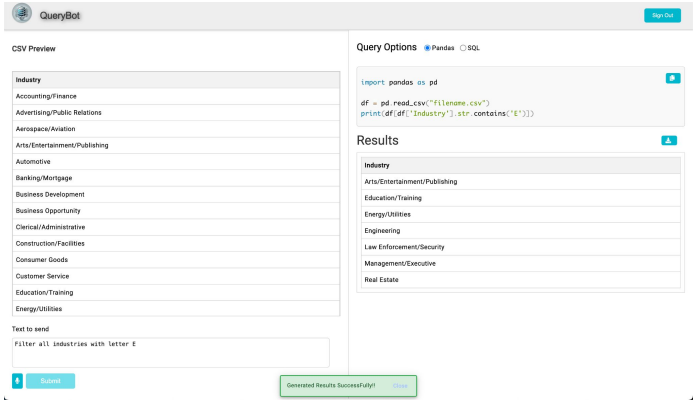


Figure 12: Successful Result

Project Flow

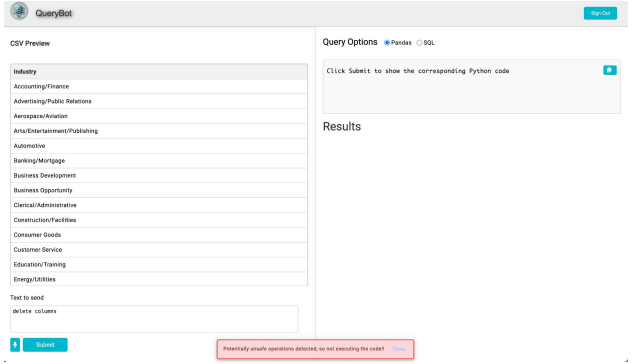


Figure 13: Unsafe Operation

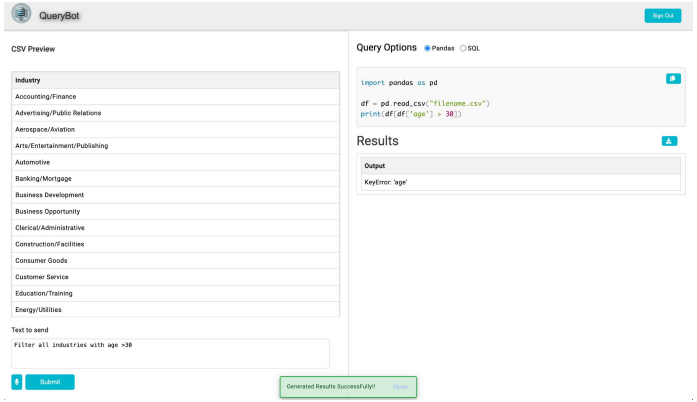


Figure 14: KeyError

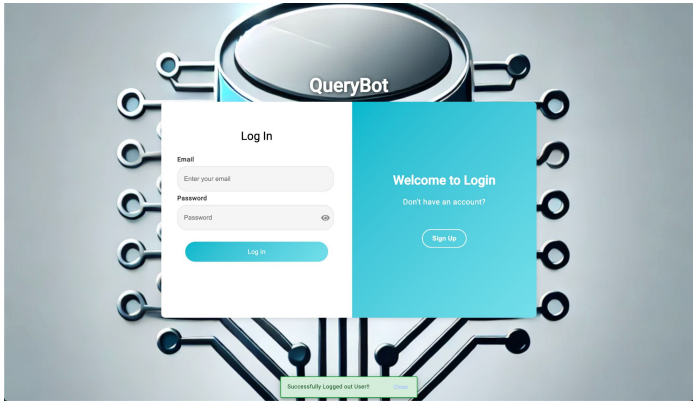
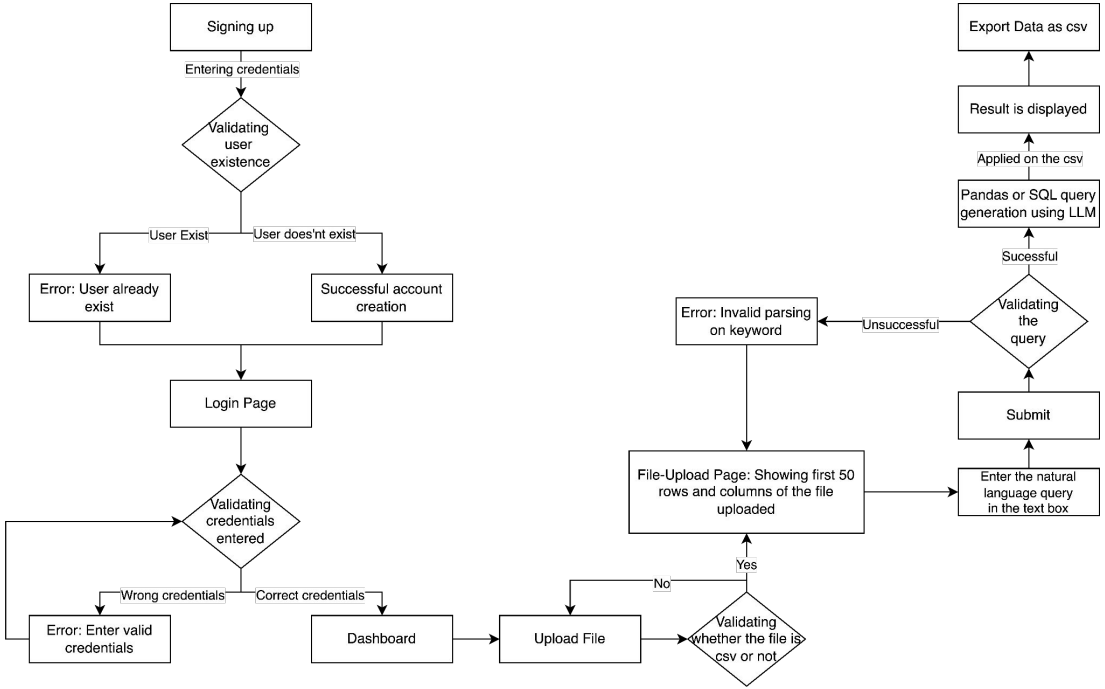


Figure 15: Sign Out

Flow of Execution



Technologies Used

We used the following technologies to build and optimize our project:

1. **Frontend:** Angular for developing an interactive and user-friendly UI
2. **Backend:** Flask, Integrating LLMs and LangChain for Natural Language processing and query generation
3. **DataBase:** AWS DynamoDB for scalable cloud data storage

Unit Testing

We write unit tests using the `Pytest` library

1. **Testing API Models:**
   - We write unit tests for the `User` and `UserFile` API models
   - We create objects using dummy attributes like `name`, `user_id`, etc
   - Object is inserted into the table using the `put()` method
   - We then use the `get()` method to retrieve the data and verify that it is correct

```
user = User(user_id='123', name='John Doe', username='johndoe', email='john.doe@example.com', hashed_password='password')
user.put()  # Insert into USER_TABLE
assert User.get('123') == {'user_id': '123', 'name': 'John Doe', 'username': 'johndoe', 'hashed_password': 'password', 'email
```

`User` model testing

```
user_files = UserFiles(user_id='123', files=['file1', 'file2'])
user_files.put()  # Insert into USER_FILES_TABLE
assert UserFiles.get('123') == {'user_id': '123', 'files': ['file1', 'file2']}
```

`UserFiles` model testing

Unit Testing

2. Testing the LLM Agents:

- We run checks for a range of queries on a sample CSV file and match outputs to the ground truth answers. Below are some samples for the Titanic dataset:

```
Filter the rows where age is 21
How many people are there with age = 21?
What was the maximum fare?
Which cabin had the fare of 76.2917?
How many people were named John?
What was the name of passenger in cabin D56?
How many male people survived?
```

- We also have testcases to check inputs to the agent function calls are valid, i.e. valid dataframe and query string.

Unit Testing

3. Testing Query Validators:

- We write tests to check the functionality of the query validator function.
- The query validator functions make sure to filter out any keywords that if executed, could cause harm to the system like delete or alter data, execute other commands, etc.
- Below are some samples:

```
assert sql_input_query_validator("SELECT * FROM table") == True
assert sql_input_query_validator("DELETE FROM table") == False
assert sql_input_query_validator("DROP TABLE table") == False
```

SQL query validation sample testcases

```
assert pandas_input_query_validator("df") == True
assert pandas_input_query_validator("rm -rf /") == False
assert pandas_input_query_validator("import os") == False
```

Pandas query validation sample testcases

Demo Video:
https://drive.google.com/file/d/1WT693CTNA-ejxAjdLeu0GIlnYCjIxB4Y/view?usp=sharing

Report:
https://github.com/Avi-2362/LLM-agents-for-SQL-Pandas-query-generation/blob/main/report.pdf

Thank You!!