Documentation for Financial Data Chatbot

Overview

This document provides an overview of a Flask-based chatbot that responds to predefined financial queries using data from two CSV files: final_financial_data.csv and average_growth.csv. The chatbot can answer questions related to total financial metrics and growth rates for specified companies.

File Structure

- **app.py (This script):** Contains the chatbot logic and Flask application.
- **final_financial_data.csv:** CSV file with financial data for companies.
- **average_growth.csv:** CSV file with growth rates for companies.
- **templates/**
  - index.html: HTML file for the user interface.

Setup

Prerequisites

Ensure you have the following installed:

- Python
- Flask
- pandas

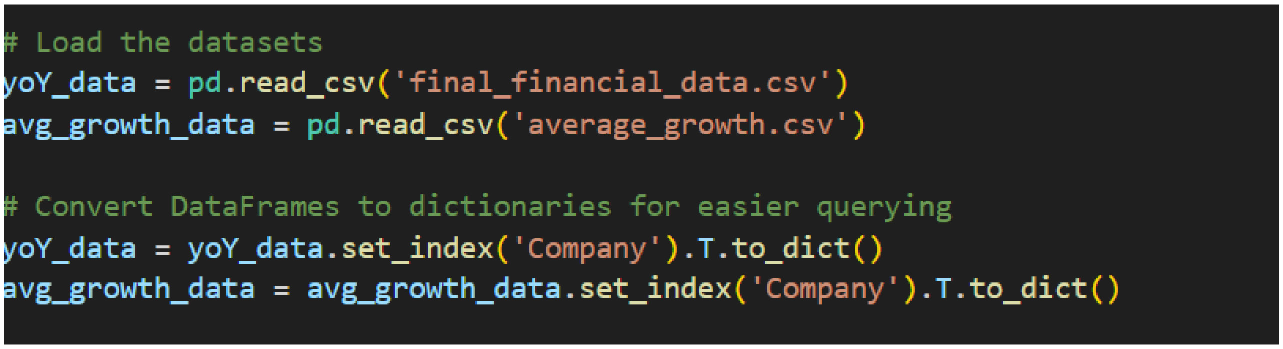
Install the required Python libraries using:

```
pip install pandas
pip install flask
```

File Preparation

- Place final_financial_data.csv and average_growth.csv in the same directory as app.py.
- Ensure final_financial_data.csv includes columns: Company, Total Revenue, Net Income, Total Assets, Total Liabilities, Cash Flow from Operating Activities.
- Ensure average_growth.csv includes columns: Company, Revenue Growth (%), Net Income Growth (%), Assets Growth (%), Liabilities Growth (%), Cash Flow from Operations Growth (%), Average Revenue Growth (%), Average Net Income Growth (%), Average Assets Growth (%), Average Liabilities Growth (%), Average Cash Flow from Operations Growth (%).
- **Code Explanation**
- **Data Loading and Processing**

- The script loads the financial and growth data from CSV files into pandas DataFrames. These DataFrames are then converted to dictionaries for easier querying:



```
# Load the datasets
yoY_data = pd.read_csv('final_financial_data.csv')
avg_growth_data = pd.read_csv('average_growth.csv')

# Convert DataFrames to dictionaries for easier querying
yoY_data = yoY_data.set_index('Company').T.to_dict()
avg_growth_data = avg_growth_data.set_index('Company').T.to_dict()
```

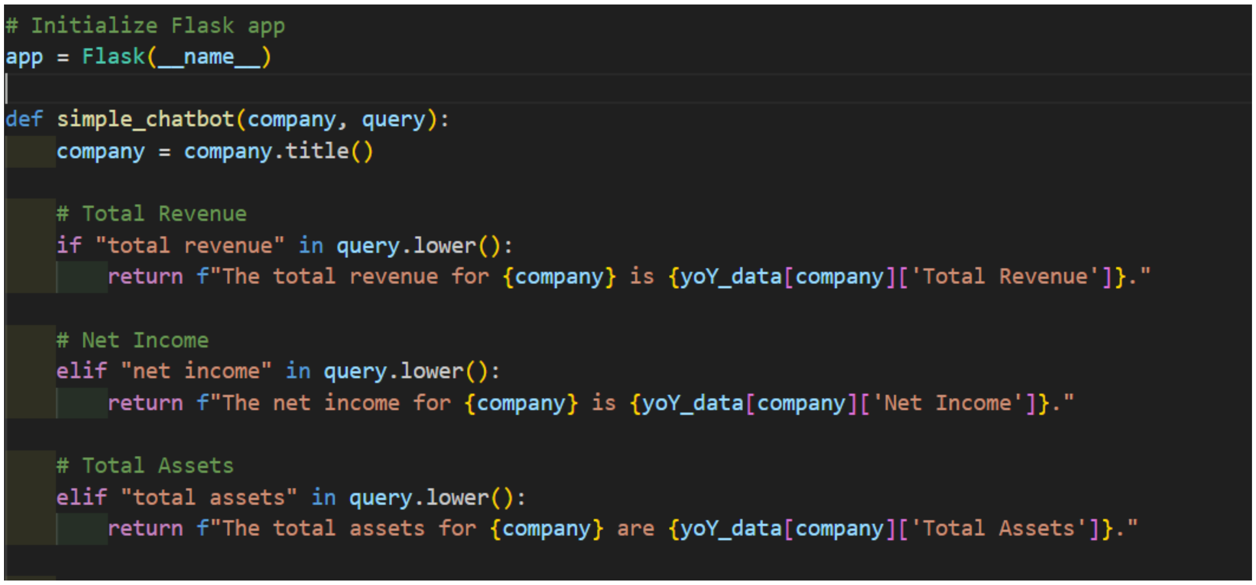
Flask Application

The Flask application initializes and defines routes for user interaction:

- **Route /**: Renders the HTML interface.
- **Route /chat**: Handles POST requests to process user queries.

Chatbot Logic

The simple_chatbot function processes user queries and provides responses based on the financial and growth data:



```
# Initialize Flask app
app = Flask(__name__)

def simple_chatbot(company, query):
    company = company.title()

    # Total Revenue
    if "total revenue" in query.lower():
        return f"The total revenue for {company} is {yoY_data[company]['Total Revenue']}."

    # Net Income
    elif "net income" in query.lower():
        return f"The net income for {company} is {yoY_data[company]['Net Income']}."

    # Total Assets
    elif "total assets" in query.lower():
        return f"The total assets for {company} are {yoY_data[company]['Total Assets']}."
```

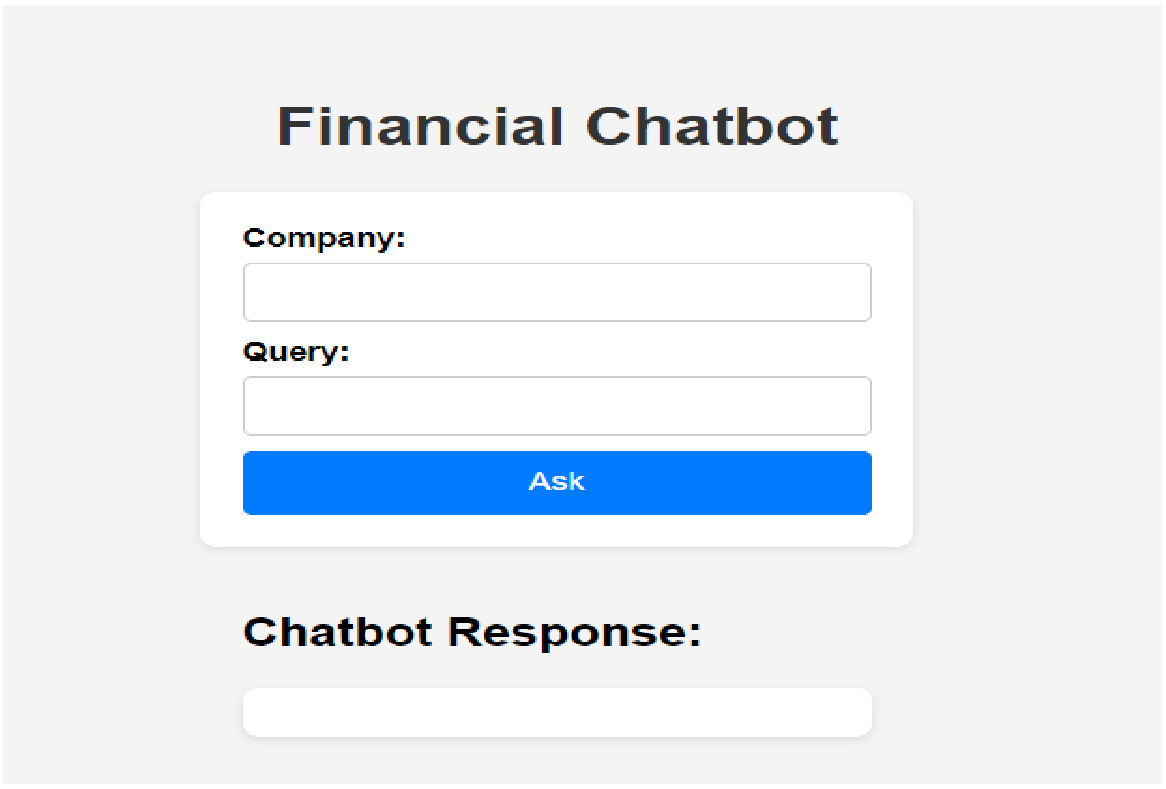
Running the Application

The Flask application runs in debug mode for development purposes:

```python
# Run the Flask app
if __name__ == "__main__":
    app.run(debug=True)
```

How to Use

1. **Start the Flask Server**: Run the Flask application with:
2. **Access the Chatbot Interface**: Open a web browser and navigate to `http://127.0.0.1:5000/` to access the chatbot interface.
3. **Interact with the Chatbot**: Enter a company name and a query into the input fields and submit the form. The chatbot will provide a response based on the predefined queries.



Example Queries

- **Total Revenue**: "What is the total revenue?"
- **Net Income**: "How has net income changed?"
- **Revenue Growth**: "What is the revenue growth rate?"

Limitations

- The chatbot only responds to predefined queries related to financial data and growth rates.
- It requires exact column names and proper formatting in the CSV files.

Future Enhancements

- **Dynamic Data Handling**: Implement functionality to handle dynamic or unstructured queries.
- **Data Visualization**: Add charts or graphs to visualize financial trends and metrics.
- **Natural Language Processing (NLP)**: Integrate NLP techniques to improve query understanding and flexibility.

Conclusion

This Flask-based chatbot provides a basic interface for querying financial data and growth rates. It serves as a starting point for more advanced chatbot development in financial analysis.