Financial Chatbot Documentation

Overview:

The Financial Chatbot is a web-based application developed using Flask, HTML, CSS, and JavaScript. The chatbot provides insights into the financial performance of companies based on predefined data. Users can select specific questions, companies, and years from dropdown menus to receive detailed responses.

This financial chatbot leverages data from the 10-K filings of Apple, Microsoft, and Tesla to provide quick insights into key financial metrics. It uses rule-based logic to respond to predefined queries about total revenue, net income changes, total assets, total liabilities, and cash flow from operating activities and many more.

Project Structure:

```
project-directory
├── app.py                # Main Flask application
├── financial_data.csv    # CSV file containing financial data
├── templates
│   └── index.html        # HTML file for the frontend
├── static
    ├── styles.css        # CSS file for styling
    └── script.js         # JavaScript file for frontend logic
```

File Descriptions:

1. app.py
    - The backend of the application, built with Flask. It serves the HTML page and handles the API requests made by the frontend. It also processes the data and returns the appropriate responses based on the user's selections.
2. financial_data.csv:
    - Contains financial data for various companies, including metrics like total revenue, net income, total assets, total liabilities, operating expenses, and growth percentages.
3. templates/index.html:
    - The HTML file that provides the user interface, including dropdown menus for selecting the query, company, and year, and a button to submit the query.
4. static/styles.css:
    - Contains the CSS for styling the frontend, ensuring a clean and visually appealing design.
5. static/script.js:
    - The JavaScript file handles user interactions, captures the dropdown selections, and sends the request to the Flask backend. It also processes the response and updates the UI accordingly.

Key Features:

1. User Interaction:
    - Users can select a question, company, and year from dropdown menus.
    - Upon submission, the chatbot processes the input and displays the corresponding financial information.
2. Data Processing:
    - The chatbot uses financial_data.csv to fetch relevant data based on user inputs. It handles requests for different financial metrics and returns the appropriate responses.

How to Run:

1. Set Up Environment:
    - Ensure you have Python and necessary packages installed. You may use a virtual environment for package management.
    - Install Flask: pip install Flask
    - Install pandas: pip install pandas
2. Start the Application:
    - Navigate to the project directory and run the Flask application using:

      ```
      python app.py
      ```

    - The application will start a local server, typically available at http://127.0.0.1:5000/.
3. Access the Application:
    - Open a web browser and navigate to http://127.0.0.1:5000/ to interact with the chatbot.

Technical Details:

1. Dropdown Functionality:
    - The dropdown menus in index.html allow users to specify their query parameters. The selection is sent to the Flask backend using a POST request.
2. Data Handling:
    - The financial_data.csv file is read into a pandas DataFrame when the Flask app initializes. The chatbot logic uses this DataFrame to retrieve relevant financial data based on user queries.
3. Response Generation:
    - The backend processes the user query, extracts the relevant data, and constructs a response. The response is sent back to the frontend and displayed to the user.

Limitations and Future Improvements:

1. Predefined Queries:
    - The current implementation supports only predefined queries. Future improvements could include natural language processing (NLP) capabilities to handle a broader range of user inputs.
2. Data Expansion:
    - The chatbot can be extended to include more financial metrics, additional companies, and a broader dataset for deeper insights.
3. Enhanced UI/UX:
    - Future updates could enhance the user interface with more interactive elements, such as charts and graphs, to visualize data.

Conclusion:

This Financial Chatbot provides a user-friendly interface for accessing financial data. The combination of Flask, HTML, CSS, and JavaScript creates an interactive and responsive experience, making complex financial information accessible and engaging. This project serves as a foundation for more advanced chatbot development, integrating real-time data and more sophisticated AI features.