**Documentation: Financial Chatbot Prototype**

**Overview**

This project is a simple Flask-based web application that functions as an AI-powered financial chatbot. It is designed to analyze financial data from corporate 10-K filings, specifically for Microsoft, Tesla, and Apple, and respond to predefined financial queries about the companies' performance. The chatbot prototype provides a starting point for developing more advanced AI-driven financial chatbots.

**Folder Structure**

graphql

Copy code

└── app.py # Main Flask application

└── templates

└── index.html # HTML template for the chatbot interface

└── static

└── styles.css # Optional CSS file for styling the chatbot (optional, can be added later)

**Installation Instructions**

1. **Clone the repository**: If you haven't yet, clone the repository to your local machine or create the necessary files in your project directory.
2. **Install dependencies**: You need to have Python 3 installed. Install the necessary libraries using pip:

bash

Copy code

pip install flask

1. **Run the Flask app**: In your project directory, run the following command to start the Flask server:

bash

Copy code

python app.py

1. **Open the application**: Once the server is running, open a browser and visit http://127.0.0.1:5000 to interact with the chatbot.

**How It Works**

1. **Main Flask Application (app.py)**:
   * The app.py file contains the Flask routes and the logic to handle user inputs and return responses.
   * The chatbot logic is based on predefined queries related to financial data, such as:
     + "What is the total revenue?"
     + "How has net income changed over the last year?"
   * The chatbot uses if-else statements to match user input to predefined queries and provides responses based on financial data.
2. **HTML Template (index.html)**:
   * The index.html file provides a simple form for users to input queries.
   * When the user submits a query, it sends the input to the backend (Flask) and displays the chatbot’s response on the webpage.
   * The form uses a POST request to send data to the Flask route, which processes the query and returns the appropriate response.
3. **CSS Styling (Optional)**:
   * Optionally, you can include a styles.css file in the static folder to style the chatbot’s interface.
   * Example styling could involve centering the chatbot on the page, adjusting font sizes, and customizing colors to enhance the user experience.

**Predefined Queries and Responses**

* **What is the total revenue?**
  + Returns the total revenue for the selected company.
* **How has net income changed over the last year?**
  + Compares net income between two fiscal years and returns the change.
* **How much are total assets and liabilities?**
  + Returns the total assets and liabilities for the selected company.
* **How is the cash flow from operating activities?**
  + Provides details about cash flow from operations.

**Example Flow**

1. The user visits the chatbot interface at http://127.0.0.1:5000.
2. The user inputs a predefined query (e.g., "What is the total revenue?").
3. The input is sent to the backend via the POST method.
4. The chatbot matches the query with predefined responses and displays the corresponding financial information.

**Limitations**

* The chatbot only responds to predefined queries. It does not have advanced natural language processing (NLP) to handle free-form queries.
* The financial data provided in the responses must be manually updated in the script (or loaded from an external data source).

**Future Enhancements**

* **Data Integration**: Integrating dynamic financial data sources (e.g., APIs or databases) to provide real-time updates.
* **Advanced NLP**: Incorporating natural language processing (NLP) for more flexible user interactions and understanding free-form queries.
* **Scalability**: Allowing the chatbot to analyze and respond to queries for multiple companies simultaneously.

**Conclusion**

This prototype provides a basic chatbot framework capable of answering predefined financial queries using Flask, Python, and HTML. It highlights the fundamental concepts of chatbot development, such as handling user input, mapping responses, and creating an interactive interface. The chatbot leverages simple logic for matching user queries to predefined responses based on financial data extracted from 10-K reports, offering a user-friendly interface for financial analysis.

While this version is a simplified prototype, it serves as a foundational step toward developing more sophisticated AI-powered chatbots. Future enhancements could include integrating Natural Language Processing (NLP) for more dynamic and context-aware interactions, connecting to databases for real-time data retrieval, and incorporating machine learning models for advanced financial analysis and decision support.

In summary, this project showcases the potential of AI in transforming financial analysis, making complex data accessible in an interactive, conversational format, with possibilities for further scaling and refinement.