Project Documentation: `graph_chatbot`

Overview

The `graph_chatbot` project is a smart, interactive data visualization assistant built using Python. It uses a natural language interface to allow users to ask questions about data and receive visual insights in the form of graphs or charts. This chatbot is capable of interpreting user input, querying data sources, and generating dynamic visualizations.

Setup Instructions

1. **Install Dependencies**
   Ensure you have Python 3.10+ installed. Then install the required libraries: `bash pip install -r requirements.txt`

2. **Run the Application** `bash streamlit run app.py`

Detailed File Descriptions

1. `app.py`

- **Purpose:** Main entry point for the Streamlit app.
- **Functionality:**
  - Loads the chatbot interface.
  - Accepts user questions.
  - Processes queries through the query generation engine.
  - Displays visualizations.
- **Technologies:** Streamlit, pandas, visualization modules.

---

2. `chart1.py`, `chart2.py`, `chart3.py`

These files handle different chart generation scenarios:

- `chart1.py`: Generates bar/line charts.
- `chart2.py`: Deals with comparative charts or category distributions.
- `chart3.py`: Possibly focused on weekly trend comparisons.
- **Common Operations:**
  - Data slicing and filtering.
  - Chart styling with `matplotlib` or `plotly`.
  - Exporting to image formats.

---

3. `config.py`

- **Purpose:** Stores application-wide constants and configurations.
- **May Include:**
  - File paths
  - API keys (LLM, database)
  - Environment flags

---

4. `db_utils.py`

- **Purpose:** Provides utility functions for database interaction.
- **Functions:**
  - Establish connections to SQLite/PostgreSQL.
  - Run SQL queries.
  - Return `pandas` DataFrames from results.

---

5. `schema_utils.py`

- **Purpose:** Extracts schema details from databases or CSVs.
- **Usage:** Helps LLM understand table structure and valid fields for querying.

---

6. `query_generator.py`

- **Purpose:** Core module for generating SQL or pandas queries from user input.
- **Functionality:**
  - Accepts natural language from the chatbot.
  - Leverages LLM to generate a valid data access query.
  - Handles contextual understanding using schema information.

---

7. `nlp_response.py`

- **Purpose:** Handles pre-processing of natural language input.
- **Key Features:**
  - Maps user-friendly terms to actual values in data (e.g., "successful" → "csuccess").
  - Resolves synonyms or ambiguous terms.
  - Acts as a middleware before invoking LLM logic.

---

8. `visualization.py`

- **Purpose:** Central handler for graph creation and rendering.
- **Responsibilities:**
  - Decides which chart file to call.
  - Handles layout and design.

- Manages saving/exporting charts as PNGs for reports.

---

9. `excel_*.py` Files (`excel.py`, `excel2.py`, `excelll.py`, `excel_3.py`)

- **Purpose:** Manage reading and cleaning of Excel and CSV datasets.
- **Features:**
  - Strip whitespace.
  - Standardize column names.
  - Convert to pandas DataFrame for further analysis.

---

10. `requirements.txt`

- **Purpose:** Lists Python packages required for the project.
- **Expected Libraries:**
  - `streamlit`, `pandas`, `plotly`, `matplotlib`, `openai` or `groq`, etc.

---

11. `README.md`

- **Purpose:** Project overview and minimal usage guide.
- **Note:** Recommend expanding this with screenshots, examples, and contribution steps.

---

12. `DATA.csv`, `Graph Down.csv`, etc.

- **Purpose:** Real or mock datasets used for visualization.
- **Usage:** Users ask questions based on this data (e.g., downtime types, transaction stats).

---

13. `data3_backup.sql`

- **Purpose:** SQL backup of data. Can be restored locally for testing DB queries.

Flow of Execution

```
User Question → NLP Preprocessing → Query Generation → Data Retrieval → Chart Selection → Visualization → Output Chart
```

Suggested Improvements

- Add logging in each core module.

- Integrate tests (`pytest`) for all query and chart logic.
- Expand `README.md` with full user guide.
- Use `.env` files for secure config storage.
- Containerize using Docker for production.

Summary

The `graph_chatbot` project is a modular and intelligent system for visualizing tabular data through a chatbot interface. It showcases best practices in: - Modular coding - LLM integration - Data visualization - Natural language processing

This documentation can be used for GitHub projects, internal wikis, or project submission reports.