**Project Documentation**

**Overview**

The Malawi Projects Chatbot is a RAG FastAPI application designed to provide information about infrastructure projects in Malawi. Users can interact with the chatbot through natural language queries, and the application responds with relevant project data extracted from a SQLite database. The application supports language detection and translation, handles complex queries with filters, and provides pagination for results.

**Process Flow**

**1. User Interaction**

* **User Input**: The user sends a message or query to the chatbot, possibly in English or another language.
* **Chat History**: The application maintains a chat history to handle pagination and context for "show more" requests.

**2. Language Detection and Translation**

* **detect\_and\_translate() Function**:
  + **Purpose**: Detects the language of the user's input and translates it to English if necessary.
  + **Process**:
    - Uses langdetect to identify the language.
    - If the language is not English, uses deep\_translator to translate the input to English.

**3. Query Parsing**

* **parse\_query\_intent() Function**:
  + **Purpose**: Parses the user's query to extract filters for searching the database.
  + **Filters Include**:
    - **Sector**: Education, Health, Transport, etc.
    - **Completion Status**: Completed, In Progress, Not Started.
    - **Region**: Northern Region, Central Region, Southern Region.
    - **Project Type**: Construction, Rehabilitation, etc.
    - **Budget Sorting**: Highest or lowest budget projects.
  + **Process**:
    - Matches keywords in the query to predefined categories.
    - Constructs a filters dictionary with the extracted criteria.

**4. Data Retrieval**

* **get\_project\_data() Function**:
  + **Purpose**: Queries the SQLite database (malawi\_projects1.db) to retrieve project data based on the filters.
  + **Process**:
    - Connects to the database.
    - Builds a SQL query with conditions based on the filters.
    - Executes the query and fetches results into a Pandas DataFrame.
    - Handles data cleaning and type conversion for numeric fields.

**5. Response Formatting**

* **format\_project\_details() Function**:
  + **Purpose**: Formats individual project details into a readable string.
  + **Includes**:
    - Project name, budget, location, sector, status, type, and completion percentage.
* **Response Construction**:
  + Builds a response message including:
    - A header summarizing the query results.
    - A count of total matching projects.
    - Total budget and average completion percentage.
    - Details of the first three projects.
    - Information about remaining projects and instructions to "show more" if applicable.

**6. Pagination and 'Show More' Functionality**

* **/chat/more Endpoint**:
  + **Purpose**: Handles requests to show additional projects beyond the initial set.
  + **Process**:
    - Retrieves the last search filters and offset from the chat history.
    - Fetches the next batch of projects using the same filters.
    - Updates the chat history with the new offset.

**7. Suggested Questions**

* **generate\_suggestions() Function**:
  + **Purpose**: Provides context-aware suggested questions to guide the user.
  + **Based On**:
    - The current filters and available data.
    - Encourages exploration of different sectors, regions, or statuses.

**8. Language Translation of Responses**

* If the original user input was in a language other than English, the final response and suggested questions are translated back into that language.

**Setup and Installation Guide**

**Prerequisites**

* **Python 3.7 or higher**: Ensure Python is installed on your machine.
* **SQLite**: The application uses a SQLite database (malawi\_projects1.db). Make sure you have this file and it's placed in the correct directory.

**Required Python Packages**

* **FastAPI**: The web framework used for building the API.
* **Uvicorn**: ASGI server for running the application.
* **Pandas**: For data manipulation and analysis.
* **langdetect**: For language detection.
* **deep-translator**: For language translation.
* **Jinja2**: For HTML templating (if using templates).
* **aiofiles**: For asynchronous file operations (required by Starlette).
* **python-multipart**: For handling form data (required by FastAPI).

**Installation Steps**

1. **Clone the Repository or Obtain the Source Code**

bash

Copy code

git clone <repository\_url>

Or download and extract the source code into a directory on your machine.

1. **Navigate to the Project Directory**

bash

Copy code

cd <project\_directory>

1. **Create a Virtual Environment (Optional but Recommended)**

bash

Copy code

python -m venv venv

1. **Activate the Virtual Environment**
   * **On Windows**:

bash

Copy code

venv\Scripts\activate

* + **On macOS/Linux**:

bash

Copy code

source venv/bin/activate

1. **Install the Required Packages**

bash

Copy code

pip install fastapi uvicorn pandas langdetect deep-translator Jinja2 aiofiles python-multipart

Alternatively, if a requirements.txt file is provided:

bash

Copy code

pip install -r requirements.txt

1. **Ensure the SQLite Database is in Place**
   * Place the malawi\_projects1.db file in the same directory as the main.py file or adjust the path in the code if it's located elsewhere.
2. **Check the Database Connection**
   * Run the verify\_database() function to ensure the application can connect to the database.

python

Copy code

python -c "from main import verify\_database; verify\_database()"

* + Look for logs indicating the number of projects found in the database.

**Running the Application**

1. **Start the Uvicorn Server**

bash

Copy code

uvicorn main:app --reload --host 127.0.0.1 --port 8080

* + The --reload flag enables auto-reload on code changes.
  + Adjust the --host and --port as needed.

1. **Access the Application**
   * Open your web browser and navigate to:

arduino

Copy code

http://127.0.0.1:8080

* + If you have a chat.html template in the templates directory, it will render the chat interface.

1. **Using the API Documentation**
   * Swagger UI is available at:

arduino

Copy code

http://127.0.0.1:8080/docs

* + You can test the endpoints directly from the API documentation.

**Testing the Application**

* **Example Queries**:
  + "What are the current infrastructure projects?"
  + "Show me completed health projects in the Southern Region."
  + "I want to see the highest budget education projects."
  + "Show more" (after an initial query to get additional results).
* **Expected Behavior**:
  + The application should respond with a summary of matching projects, including project details and suggested follow-up questions.
  + If more results are available, it should inform the user and handle "show more" requests appropriately.
  + Language detection and translation should work seamlessly for non-English queries.

**Logging and Debugging**

* **Logging**:
  + The application uses Python's logging module to output informational messages and errors.
  + Logs are printed to the console, indicating the flow of data and any issues encountered.
* **Error Handling**:
  + The application handles exceptions gracefully, informing the user if an error occurs and suggesting alternative queries.

**Project Structure**

* **main.py**: The main application file containing all the endpoint definitions and core functions.
* **templates/**: Directory containing HTML templates (e.g., chat.html) if used.
* **static/**: Directory for static files like CSS and JavaScript, if applicable.
* **malawi\_projects1.db**: The SQLite database file containing project data.

**Extending the Application**

* **Adding More Filters**:
  + Extend the parse\_query\_intent() function to include more filters like districts, funding sources, or specific project codes.
* **Enhancing the Database**:
  + Update the database schema or data as needed to include additional information.
* **Improving the Frontend**:
  + Develop a more interactive frontend using frameworks like React or Vue.js.
  + Integrate the API endpoints with the frontend to provide a seamless user experience.

**Troubleshooting**

* **Common Issues**:
  + **Module Not Found Errors**: Ensure all required packages are installed in your environment.
  + **Database Connection Errors**: Verify that the malawi\_projects1.db file is in the correct location and that the database schema matches the queries.
  + **CORS Issues**: If accessing the API from a different domain, ensure that CORS settings in the middleware are configured appropriately.
* **Getting Help**:
  + Use the logs to identify where errors are occurring.
  + Check the function definitions and ensure that all necessary imports are included.
  + Ensure that the correct versions of packages are installed (Python 3.7 or higher is recommended).

**Final Notes**

* **Environment Management**: Using a virtual environment helps keep dependencies organized and avoids conflicts with other Python projects.
* **Code Maintenance**: Keep the codebase updated, especially if dependencies receive updates that might affect functionality.
* **Security Considerations**:
  + Be cautious with exposing the database or sensitive data.
  + Sanitize user inputs to prevent SQL injection, even though parameterized queries are used.