Report of the AI project

themed :

EASYBOOK : Your guide for all your trips

Made and presented by : -CHEMLAL Amine

-BOUABBOUS Achraf

Supervised By : Pr. ADERGHAL Yasser & Pr. GAMOUH Hamza

Table of Contents

I. Problem definition :

WHY EasyBook ? Travel planning can often be overwhelming due to the abundance of options available online and the lack of personalized guidance. Travelers frequently face challenges such as finding relevant recommendations for accommodations, restaurants, and tourist attractions, as well as seeking immediate answers to their specific questions. This leads to a time-consuming and inefficient planning process.

The **Travel Recommendation App** addresses these issues by providing a centralized, user-friendly platform that delivers:

- **Personalized Travel Recommendations**: Users can search for hotels, restaurants, and tourist attractions based on their destination, desired radius, and minimum ratings, filtered through data from the **Google Places API**.
- **Interactive Conversational Assistance**: A ChatBot powered by **Azure OpenAI GPT** allows users to ask travel-related questions and receive accurate, conversational, and contextually relevant responses.
- **Integrated Map Visualization**: Recommendations are displayed on an interactive map, enhancing the user's ability to visualize and understand the geographical layout of their destination.

This app solves the problems of:

1. **Information Overload**: By filtering recommendations according to user preferences.
2. **Lack of Immediate Assistance**: Through an intelligent ChatBot that provides instant answers.
3. **Disconnected Information**: By integrating recommendation data with visual map representations for better usability.

The **Travel Recommendation App** aims to enhance the travel planning experience by offering intuitive, intelligent, and personalized solutions, empowering users to make informed decisions efficiently. Future enhancements could include multi-lingual support, integration with weather APIs, and booking platforms to offer an end-to-end travel solution.

II. Design and architecture

➢ Application architecture :

The application is divided into three main layers:

1. **Frontend**: Developed with Streamlit, providing a responsive user interface for both travel recommendations and ChatBot interactions.
2. **Backend** :
    - ➢ Integrates with external APIs like **Google Places** for location-based recommendations and **Azure OpenAI GPT** for the Chatbot.
    - ➢ Incorporates **RAG (Retrieval-Augmented Generation)** to enhance the ChatBot with relevant document-based responses.
3. **Database** :
   no database (only API)

➢ Technologies Used :

Languages: Python

Frontend: Streamlit

Backend: Requests library

Database: No Database

APIs:

- ➢ Google Places API (Travel recommendations)
- ➢ Azure OpenAI GPT (ChatBot)

Deployment: Streamlit deployment for local/online hosting.

III. Database and backend integration

➢ Backend :

The Google Places API is utilized to fetch data related to hotels, restaurants, and tourist attractions based on user input. This integration follows a structured approach:

- Input Handling: The app takes user inputs such as the destination, search radius, minimum rating, and type of place (e.g., hotels or restaurants) from the frontend.
- **API Request Construction:**
    - ➢ A request is sent to the Google Places API endpoint (textsearch) with parameters such as the query, radius, API key, and filters.
    - ➢ Example query: https://maps.googleapis.com/maps/api/place/textsearch/json?query=hotels+in+Bali&radius=3000&key=YOUR_API_KEY.
- **Response Handling:**
    - ➢ The API responds with detailed information about places, including their name, rating, address, and photo references.
    - ➢ If photos are available, additional requests are made to the Google Places Photo API to retrieve image URLs for display.
- **Data Visualization:**
    - o The fetched results are displayed to users in a user-friendly manner, including:
        - ➢ A detailed list of recommendations with photos and descriptions.
        - ➢ An interactive map (powered by Folium) where locations are marked with popups showing details of each place.

This backend integration ensures the app delivers real-time, reliable, and comprehensive recommendations tailored to the user's preferences.

2. Azure OpenAI GPT Integration

The conversational capabilities of the app are powered by **Azure OpenAI GPT**, which acts as an intelligent ChatBot assistant for travelers. The integration process is as follows:

- **Input Handling**:
    - Users input travel-related questions or requests, such as "Suggest some 4-star hotels in Tokyo," through a text input field in the app.
- **API Request Construction** :
    - The input is packaged into a JSON payload, which includes:
        - A system message to set the ChatBot's role, e.g., "You are a helpful travel assistant."
        - A user message containing the question or request.
        - Additional parameters, such as max_tokens (to limit response length) and temperature (to control the creativity of responses).
    - The payload is sent to the **Azure OpenAI API** endpoint using an HTTP POST request.
    - Example endpoint: https://your-resource-name.openai.azure.com/openai/deployments/your-deployment-name/chat/completions?api-version=2024-08-01-preview.
- **Response Handling**:
    - The API responds with a structured JSON object containing the ChatBot’s reply.
    - The reply is extracted and displayed in the app, formatted for readability.
- **Error Handling**:
    - The app gracefully manages potential errors, such as network issues or invalid API responses, and informs users accordingly.

This integration ensures that users can receive contextually relevant and accurate answers to their travel-related queries, enhancing the interactivity and usefulness of the app.

Technical Implementation

Both integrations leverage the requests library in Python for API calls and data handling. The structure is modular, with separate Python scripts (google_places.py and azure_chatbot.py) to encapsulate API-specific logic. This modularity makes the backend maintainable and extensible for future integrations.

By combining these powerful APIs, the backend provides a robust foundation for delivering personalized recommendations and real-time conversational support, significantly enriching the travel planning experience for users.

IV. Frontend Development

The frontend of the **Travel Recommendation App** is designed with **Streamlit**, chosen for its simplicity, interactivity, and rapid deployment capabilities. The user interface prioritizes usability and aesthetics to deliver a seamless experience for travelers. Below is a detailed breakdown of the frontend development process and its features.

Key Features and Functionalities

1. **Sidebar Navigation**:
    - **Purpose**: The sidebar provides an organized layout for input options, ensuring easy access and usability.
    - **Features**:
        - **Destination Input**: Users can specify their desired location (e.g., "Paris, France").
        - **Filters**:
            - Minimum rating (slider ranging from 0.0 to 5.0).
            - Search radius (numeric input between 100m and 50,000m).
        - **Search Type Selection**:
            - Options include hotels, restaurants, and tourist attractions.

        - **Mode Selection**:
            - Toggle between the "Rechercher" (search) mode and "ChatBot" for conversational assistance.

2. **Dynamic Map Integration**:
    - **Purpose**: Visualize recommended locations on an interactive map for better understanding and navigation.
    - **Features**:
        - **Map Centering**: Dynamically adjusts to the first result of the search query.
        - **Markers**: Each result is marked with:
            - Name
            - Address
            - Rating
        - **Pop-ups**: Display detailed information when a marker is clicked.
        - **Folium Library**: Handles map rendering and interaction.
3. **Results Display Section**:
    - **Purpose**: Provide detailed recommendations below the map in a user-friendly format.
    - **Features**:
        - **Listing Results**:
            - ➢ Displays the name, address, and rating of each location.
            - ➢ Includes photos (if available) fetched from the Google Places API.
        - **Responsive Design**:
            - ➢ Adjusts layout for desktop and mobile views.
            - ➢ Uses a clean separator (st.markdown("---")) between results for better readability.

4. **ChatBot Interface**:
    - **Purpose**: Offer conversational support to users for travel-related queries.
    - **Features**:
        - ➢ A text input field to ask questions.
        - ➢ Dynamic response display with contextually relevant answers from the Azure OpenAI ChatBot.
        - ➢ Error handling for invalid queries or API issues.
5. **Responsive Styling**:
    - o **Custom CSS**:
        - ➢ Sidebar and buttons styled for a modern look and feel.
        - ➢ Input fields are enhanced with rounded corners and vibrant borders.
        - ➢ Button hover effects for better interactivity.
    - o **Streamlit Layout Features**:
        - ➢ Wide layout configuration (st.set_page_config) ensures content is well-spread and easy to navigate.
        - ➢ Typography and spacing tailored for readability.

Frontend Implementation Details

1. **Technologies Used**:
    - o **Streamlit**: Main framework for UI development.
    - o **Folium**: Map rendering and integration with Google Places API.
    - o **HTML and CSS**: For custom styling to improve aesthetics and user experience.

2. **User Workflow**:
    - o Users input their preferences (destination, filters) in the sidebar.
    - o Search results are dynamically displayed both on the map and in a list format.
    - o Users can interact with the ChatBot for additional queries.

3. **Error Handling**:
    - o Clear error messages guide users in cases of invalid inputs or API failures.
    - o Graceful degradation ensures core functionality remains unaffected.

Future Enhancements

To further improve the frontend experience, the following features could be added:

- **Dark Mode**: Allow users to toggle between light and dark themes.
- **Multi-Lingual Support**: Enable users to switch between languages dynamically.
- **Dynamic Widgets**: Add interactive widgets for personalized itineraries or budget calculations.
- **Integration with Booking Platforms**: Allow users to directly book hotels or restaurants from the app.

By focusing on simplicity, elegance, and interactivity, the frontend ensures that users can effortlessly navigate the app while enjoying a visually appealing experience. It serves as a bridge between complex backend functionalities and the user, making the travel planning process smooth and enjoyable.

V. Challenges faced and solutions

Challenges :

1. **Integration of Multiple APIs**:
   - Managing two distinct APIs (Google Places API and Azure OpenAI) required careful handling of requests and responses to ensure smooth performance and accurate data display.
2. **Rate Limits and API Constraints**:
   - Both APIs impose limits on the number of requests per minute, which required implementing fallback mechanisms to handle rate limit errors gracefully.
3. **Frontend Design and Responsiveness**:
   - Creating a modern, visually appealing, and responsive user interface using Streamlit, while balancing functionality and aesthetics.
4. **Error Handling and User Feedback**:
   - Ensuring meaningful error messages were displayed when APIs failed or when user inputs were invalid.
5. **Real-Time Performance**:
   - Delivering near-instant results for users, particularly when processing multiple API calls, was a challenge in maintaining performance.

Solutions

1. **API Integration**:
   - Modularized the API requests to separate concerns:
     - ➢ Google Places API for location-based recommendations.
     - ➢ Azure OpenAI API for conversational features.
   - Centralized response handling ensured clear error messages and smooth user experience.
2. **Rate Limit Management**:
   - Added error handling for rate limit scenarios, displaying user-friendly messages suggesting retries after a certain period.

3. **Frontend Improvements**:
    - Used Streamlit's wide layout and custom CSS for a professional and visually appealing design.
    - Ensured responsiveness across devices using containers and streamlined layouts for consistent user experience.
4. **Enhanced Error Handling**:
    - Wrapped API calls in try-except blocks to capture and display clear messages for connection issues, invalid responses, or missing data.
    - Provided default fallback messages when specific data (e.g., photos or ratings) was unavailable from APIs.
5. **Real-Time Optimization**:
    - Leveraged efficient request handling and caching of certain data during a session to minimize redundant API calls and reduce latency.

VI. User guide

Features

1. Search for Recommendations

- **Input Fields** (on the sidebar):
    - **Destination**: Enter the name of the city or location (e.g., "Bali, Indonesia").
    - **Minimum Rating**: Use the slider to filter results by rating (0 to 5).
    - **Search Radius**: Specify the radius in meters (100 to 50,000).
    - **Type of Place**: Choose between:
        - Hotels 🏨
        - Restaurants 🍴
        - Tourist Attractions ☆
- **Search Button**:
    - Click **Rechercher** 🔍 to fetch recommendations.
- **Output**:

  - An interactive map displaying recommended places.
  - Detailed information about each place, including:
    - Name
    - Address
    - Rating
    - Photo (if available).

2. Azure ChatBot

- **How to Use**:
  - Enter a question in the chatbot input field located in the sidebar (e.g., "What are the best travel tips for Bali?").
  - Click **Enter** to send your query.
- **Output**:
  - The chatbot will provide an intelligent response based on your question.

Example Usage

Searching for Hotels in Paris:

1. Enter "Paris, France" in the destination field.
2. Set the minimum rating to 4.5.
3. Set the radius to 2,000 meters.
4. Select "Hotels" and click **Rechercher** .
5. View the map with pins for recommended hotels.
6. Scroll down to see detailed information about each hotel.

Asking the ChatBot:

1. In the chatbot input field, type, "What are some safety tips for tourists in Paris?"
2. The chatbot will provide helpful advice based on your query.

Error Handling

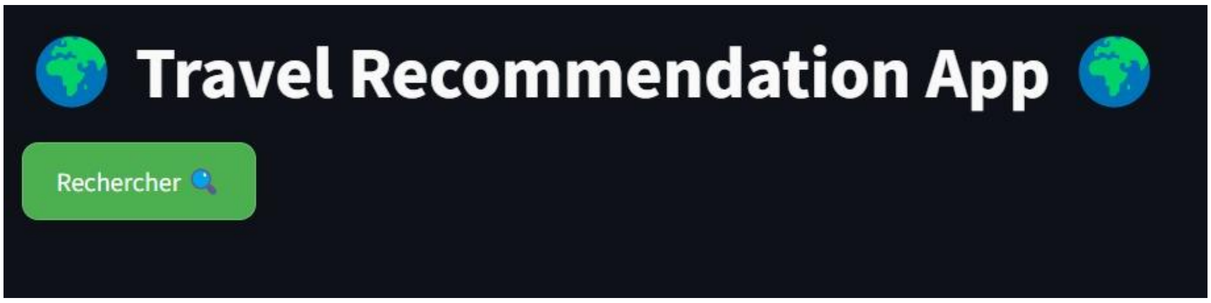
1. **Invalid Inputs**:
   - Ensure the destination is spelled correctly.
   - If no results are found, try expanding the search radius or lowering the minimum rating.

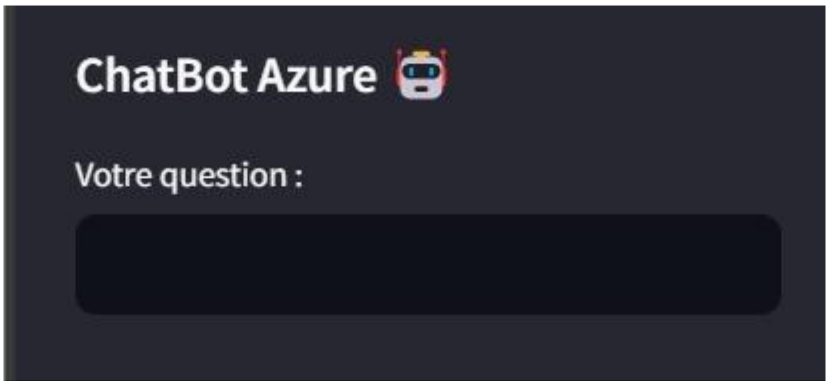
2. **API Errors**:
   - If the app encounters issues with the APIs, it will display a user-friendly error message.
   - Example: "Erreur API Google Places : Invalid API key."

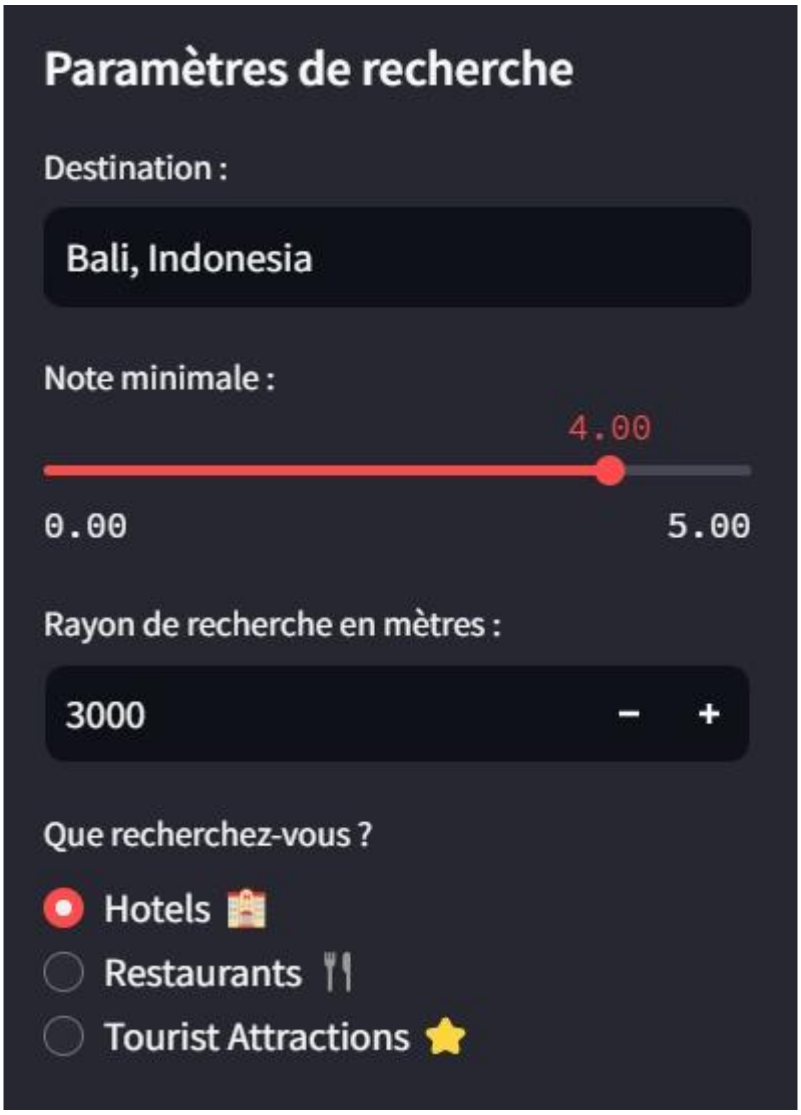
Future Features (Optional for Developers)

- **Add User Profiles**: Save favorite places or past searches.
- **Offline Mode**: Cache results for offline access.
- **More Customization**: Add filters for price range, open hours, etc

Screens :







Paramètres de recherche

Destination : Bali, Indonesia

Note minimale : 4.00 (0.00 – 5.00)

Rayon de recherche en mètres : 3000

Que recherchez-vous ?

[✓ Hotels 🏨]
Restaurants 🍴
Tourist Attractions ⭐

Repository link :

VII. Conclusion

The **Travel Recommendation App** successfully integrates Google Places, Azure OpenAI, and RAG to provide a robust travel assistant. Future improvements include advanced personalization, support for multiple languages, and integration with booking platforms.