**Travel-Assist-AI Chatbot (using LLM)**

A chatbot to answer user travel queries and recommend holiday packages based on a dataset, powered by OpenAI APIs (LLM).

**Part 1: Introduction**

**Project Background**

In today's busy and fast-paced life, everyone wants a quick escape in the form of a holiday. However, the overwhelming number of choices and the lack of personalized assistance can make holiday planning daunting.

To address this, we have developed **TravelAssist AI**, a chatbot that combines the power of large language models and rule-based functions to ensure accurate and reliable holiday recommendations.

**Problem Statement**

Given a dataset containing information about holiday packages (Package name, Destination, No. of days, Sightseeing, etc.), build a chatbot that:

* Parses the dataset
* Provides accurate holiday recommendations based on user requirements

**Dataset**

The dataset is taken from Kaggle: *Travel Listings from MakeMyTrip*

* Original dataset: **30k rows**
* For this project: **trimmed to 4 rows** (to overcome OpenAI timeout issues)

**Approach**

**Conversation & Information Gathering**

* Uses LLMs to understand and generate natural responses
* Conversational flow to gather user requirements

**Information Extraction**

* Rule-based functions extract relevant holiday packages that match user needs

**Personalized Recommendation**

* Engages in further dialogue with the user
* Provides accurate package details and answers queries

**Part 2: System Design**

The chatbot contains the following layers:

* Intent Clarity Layer
* Intent Confirmation Layer
* Product Mapping Layer
* Product Information Extraction Layer
* Product Recommendation Layer

**Major Functions**

* initialize\_conversation() → Initializes the system message for the chatbot
* get\_chat\_model\_completions() → Handles conversation flow and returns AI responses
* moderation\_check() → Flags inappropriate messages from user or assistant
* intent\_confirmation\_layer() → Validates if user profile (Destination, Package, Origin, Duration, Budget) is captured
* dictionary\_present() → Ensures chatbot output is a Python dictionary for profile info
* compare\_holiday\_with\_user() → Compares user requirements with dataset, returns top matches
* initialize\_conv\_reco() → Sets up recommendation dialogue

**Part 3: Implementation**

**Intent Clarity & Confirmation Layers**

* Captures user requirements into a structured dictionary
* Uses prompt engineering, chain-of-thought reasoning, and few-shot prompting

**Dictionary & Moderation Checks**

* dictionary\_present() → ensures valid Python dict output
* moderation\_check() → stops offensive/inappropriate chats

**Product Mapping & Information Extraction**

* product\_map\_layer() → extracts key features based on dataset
* extract\_dictionary\_from\_string() → parses final user profile dict
* compare\_holiday\_with\_user() → filters by budget & matches requirements

**Product Recommendation Layer**

* Initializes recommendation conversation
* Generates recommendations in structured format
* Engages user with follow-up Q&A

**Dialogue Management**

* dialogue\_mgmt\_system() → integrates all layers into one flow

**User Interface**

* Built with **Flask framework** for smooth interaction

**Chatbot Functionalities**

* Handles irrelevant requests
* Flags offensive/prohibited content
* Origin city limited to New Delhi / Mumbai (with fallback handling)
* Handles budget below minimum dataset value (6500 INR)
* Gracefully handles no matching packages
* Provides holiday recommendations when user inputs match dataset

**Limitations & Challenges**

* Dataset trimmed to **4 rows** → very limited recommendation scenarios
* Exact value matching only → no alternative suggestions
* Must enter "ok" after info collection to fetch results
* Frequent timeout errors if dataset size increased
* Positive recommendation scenario (example):
  + Destination: Gangtok
  + Duration: 7 nights
  + Package: Deluxe
  + Origin: New Delhi
  + Budget: 36000 INR

**Future Enhancements**

* Expand dataset size (with optimized query handling)
* Fuzzy matching & alternative recommendations
* Richer UI with better conversation management

**Tech Stack**

* Python
* Flask
* OpenAI API (LLMs)