Project Title	**Agentic AI-Based Travel Planning Assistant Using LangChain**
Skills take away From This Project	**Python Programming, LLM Integration, Agentic AI (Langchain), Prompt engineering, API Integration, Streamlit**
Domain	**Travel/Tourism**

Problem Statement:

Planning a trip requires choosing flights, hotels, and attractions while considering time, budget, weather, distance, and personal preferences.
Travelers often switch between multiple websites, compare inconsistent information, and manually build itineraries that may be inefficient, unrealistic, or incomplete.

There is a need for an **intelligent, automated system** that can handle real-time information, reason like a travel expert, and provide optimized itineraries tailored to user preferences.

Business Use Cases

Travel agencies, hotel platforms, airline aggregators, and tourism companies are shifting to AI-driven self-service travel planning tools. An AI Travel Agent can:

- Reduce customer support workload
- Provide personalized recommendations
- Automate itinerary design

- Improve customer satisfaction
- Save users time and money

Companies like **MakeMyTrip**, **Booking.com**, **ClearTrip**, and **Ixigo** are adopting conversational and agentic AI to enhance user experience.

---

3. Project Objectives

✓ Primary Objectives

1. Build an agentic AI system using LangChain that autonomously creates trip itineraries.
2. Integrate tools for:
    - Flight search (JSON dataset)
    - Hotel suggestions (JSON dataset)
    - Places/POIs search (JSON dataset)
    - Real-time weather (Open-Meteo API – free)
3. Enable multi-step reasoning and decision-making (ReAct / ToolCalling agents).
4. Generate structured itineraries with:
    - Day-wise plan
    - Accommodation
    - Weather expectations
    - Budget estimation

✓ Secondary Objectives

5. Implement filtering, ranking, and optimization (cheapest flight, highest-rated hotel, etc.).
6. Ensure the system can justify decisions (“Why we selected this?”).
7. Provide outputs in clean JSON + human-readable format.
8. Build a simple interface (CLI or Streamlit).

4. Project Approach

Step 1 — Data Setup

Use provided datasets: Refer data sources section in this document below.

- flights.json
- hotels.json
- places.json

Call real API:

- Weather: Open-Meteo (No API key required)

Step 2 — Build Tools

Create LangChain tools:

1. Flight Search Tool
    - Reads flights.json
    i. Filters by source → destination

        ii. Suggests cheapest / fastest flight
    - Hotel Recommendation Tool
        i. Reads hotels.json
        ii. Filters by city, rating, price
    - Places Discovery Tool
        i. Reads places.json
        ii. Recommends attractions based on type & rating
    - Weather Lookup Tool
        i. Calls free Open-Meteo API
        ii. Provides forecast for travel dates
    - Budget Estimation Tool
        i. Sums flight + hotel + per-day local expenses

Step 3 — Create the Agent

Build a LangChain ReAct or OpenAI ToolCalling Agent.

Agent Responsibilities:

- Understand user's travel query
- Decide which tools to call
- Retrieve data
- Analyze results
- Construct 3–7 day itinerary
- Estimate cost
- Produce final answer in structured format

Step 4 — Generate Final Output

The agent should provide:

1. Trip Summary
2. Flight Option Selected
3. Hotel Recommendation
4. Day-wise Itinerary
5. Weather for Each Day
6. Budget Breakdown
7. Reasoning (optional)

---

5. Expected Results

Final system should generate outputs like this: Example:

```
Your 3-Day Trip to Goa (Feb 12–14)

Flight Selected:

- IndiGo (₹4800) – Departs Delhi at 14:00

Hotel Booked:

- Sea View Resort (₹3200/night, 4-star)

Weather:

- Day 1: Sunny (31°C)

- Day 2: Partly Cloudy

- Day 3: Light Breeze

Itinerary:
```

```
Day 1: Baga Beach, Candolim Market

Day 2: Basilica of Bom Jesus, Old Goa Heritage Walk

Day 3: Water Sports at Calangute

Estimated Total Budget:

- Flight: ₹4800

- Hotel: ₹6400

- Food & Travel: ₹2500

-------------------------------------

Total Cost: ₹13,700
```

6. Data Sources

✓ Flights, Hotels, Places (Large JSON Files)

Sample json files for `flights.json,hotels.json,places.json` are already available, refer the drive link -

https://drive.google.com/drive/folders/18TK-2VfwoFRA515Cbl9yfv9dwW74HbX0?usp=sharing

**** You can use the given data or collect more data from relevant sources(from free apis or external sites through scraping).**

✓ Weather Data (Live & Free)

API: Open-Meteo - Works without API key.

URL:

`https://api.open-meteo.com/v1/forecast?latitude=15.2993&longitude=74.1240&daily=temperature_2m_max&timezone=auto`

Expected Outcome:

- **Clearly explaining the problem, business use case, and what the system should solve.**
- **Student correct use of the provided JSON datasets and integration of the free weather API or any other APIs.**
- **Agentic Workflow & LangChain Implementation- build meaningful tools and an agent that can autonomously decide and generate itineraries.**
- **Code Quality & Project Structure- Ensure Code is clean, readable, modular, and logically organized with proper documentation.**
- **Final Output Quality & Presentation- Trip itinerary is clear, structured, and complete with flights, hotels, weather, and recommendations.**

Project Guidelines:

1. **Coding Standards**
   - Use meaningful names: Variables, functions, and database tables should have descriptive names.
   - Follow PEP 8 (for Python): Maintain consistent formatting with proper indentation and spacing.
   - Modularize your code: Break your code into functions or classes to enhance readability and reusability.
   - Error handling: Implement try-except blocks for handling API errors and SQL exceptions.
   - Document your code: Include docstrings and comments to explain logic and functions.
2. **SQL Database Practices**
   - Normalize tables: Avoid redundancy and ensure efficient data storage.
   - Use indexes: Optimize query performance with appropriate indexing.
   - Follow naming conventions: Use consistent and descriptive names for tables and fields.

3. **Streamlit Application Development**
    - Interactive features: Ensure the UI is responsive, with interactive widgets for filters.
    - Minimalist design: Keep the layout simple for a smooth user experience.
4. **General Best Practices**
    - Test frequently: Regularly test each component (e.g., API requests, SQL queries, Streamlit app) during development.
    - Documentation: Provide a README file with setup instructions, project objectives.

Reference:

If you don't know how to approach the project, kindly refer to the project orientation recording provided in this table.

Streamlit Doc	https://docs.streamlit.io/library/api-reference
LangChain Documentation	https://docs.langchain.com/
Streamlit recording (English)	Special session for STREAMLIT(11/08/2024)
Project Live Evaluation Metrics	Project Live Evaluation
Capstone Explanation Guideline	Capstone Explanation Guideline
GitHub Reference	How to Use GitHub.pptx

Timeline:

The project must be completed and submitted within **15 days from the assigned date.**