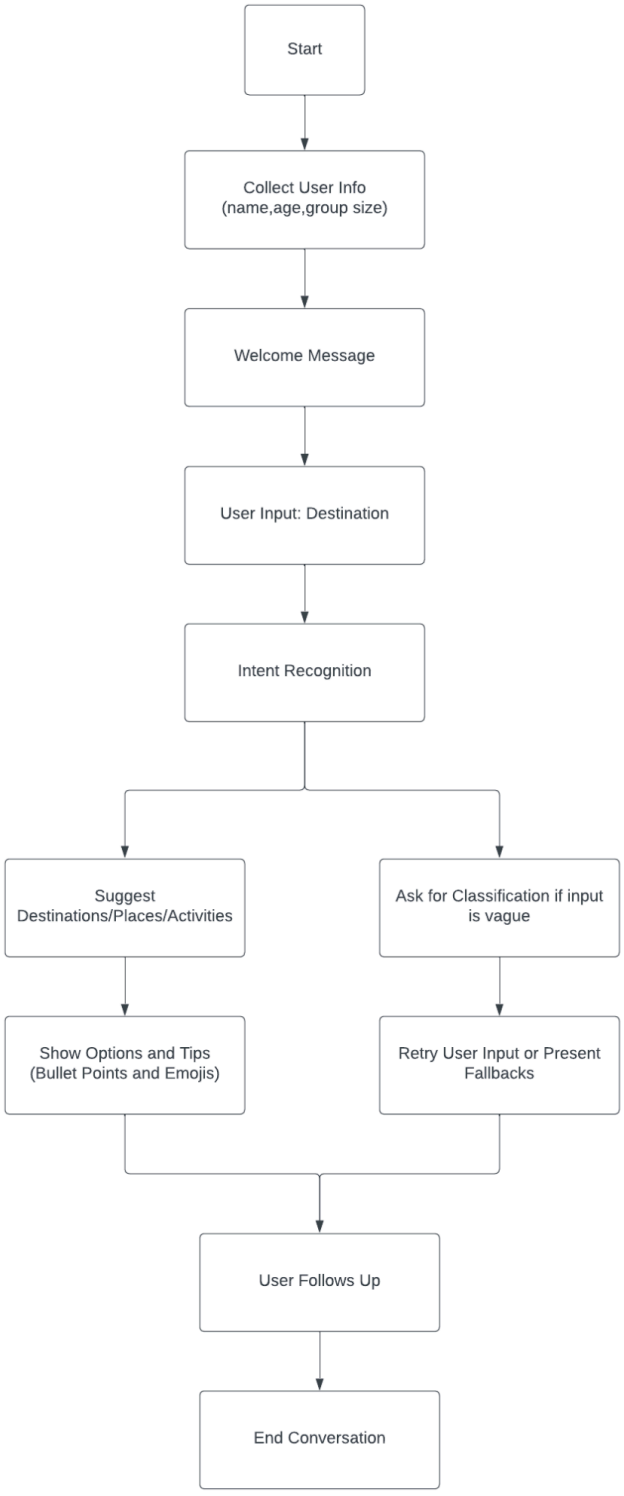
Flow Diagram & Conversation Logic

Project: Travel Planner Chatbot (LangChain + Streamlit + FastAPI)

UML-Based Flow Diagram (Page 2)

Attached on the next page is a simplified UML-style conversation flow diagram that outlines how the chatbot interacts with the user and handles various scenarios.


Start
Collect User Info
(name,age,group size)
Welcome Message
User Input: Destination
Intent Recognition
Suggest
Destinations/Places/Activities
Ask for Classification if input
is vague
Show Options and Tips
(Bullet Points and Emojis)
Retry User Input or Present
Fallbacks
User Follows Up
End Conversation

Conversation Flow Explanation

This bot is implemented using a **LangChain LLMChain** backend with memory and a **Streamlit frontend**. Here's how each component and step works:

---

1. Start

- User opens the app and is prompted to enter:
    - Name
    - Age
    - Group size

This personal information is stored and passed to the backend for personalized recommendations.

---

2. Welcome Message

- Once the user enters their name, the bot displays a friendly, enthusiastic welcome message and prompts the user to share their travel destination or intent.

---

3. User Input: Destination / Intent

- The user types in something like:

  “I want to go to Mexico for Spring Break.”

---

4. Intent Recognition

- The LLM understands whether the user is:
    - Asking for destination ideas
    - Interested in food, parties, nature, etc.
    - Talking about an event, festival, or activity
    - Or asking something off-topic (like programming)

---

5. Conditional Logic

- Basic conditional logic is built in to **adapt responses** based on:
    - Age (e.g., if user is 19, it may suggest youth hostels, college-friendly destinations)
    - Group size (e.g., group activities or solo travel advice)
    - Keywords (e.g., "drinking", "coding", "weather" → custom responses or re-routing)

---

6. Suggest Destinations / Tips

- The bot responds with:
    - Suggested destinations or cities
    - Activities, restaurants, events
    - Local tips, emojis, and bullet points for clarity and polish

---

7. Ask for Clarification

- If the user says something vague (e.g., "somewhere fun"), the bot prompts:

  “Do you prefer beaches, cities, or mountains?”

This loop helps gather enough information for relevant suggestions.

---

8. Fallback / Retry

- If the intent is completely out of scope (e.g., “Write me a Java program”), the bot responds politely:

  “I’m focused on travel planning right now! Could you share where you’d like to go next?”

This ensures the user stays in a valid conversation loop.

---

9. End Conversation

- If the user types "bye", "thanks", or "that's all", the bot politely ends the conversation with a farewell message like:

  "Have a wonderful trip, Sathvik! 🌍✈️ See you next time."

Error Handling & User Experience

- The app includes:
  - **Default values** to prevent crashes
  - **Fallback messages** for unrelated queries
  - A memory chain to maintain chat history
  - Clear user guidance via the left-hand sidebar
  - Instructions for what to do next

Summary

This TravelBot combines conversational flow, LLM-powered memory, conditional logic, personalization, and a polished user interface. It satisfies all functional assignment criteria and demonstrates thoughtful user-centric design.