Buddy Bot

Buddy Bot is developed using **Python**, **OpenAI API**, and **LangChain** to provide an interactive user experience. It incorporates **intent recognition**, **email validation**, **natural language processing (NLP)**, and **fallback mechanisms** to handle unexpected inputs effectively.

The chatbot's logic is designed to guide users through a series of conversational states and handle their inputs with predefined responses and fallbacks.

Here's a brief breakdown of the flow and error handling:

1. **Initial Input (Name & Email Collection)**:
    - The chatbot starts by asking for the user's name and email address.
    - **Email Validation**: The chatbot validates the email using a strict regex pattern to ensure the input is a properly formatted email address (e.g., user@domain.com). If the email is invalid, the chatbot prompts the user to provide a valid email.
2. **Handling Various User Inputs**: Once the user's name and email are collected, the chatbot enters the main interaction loop, where it handles different types of user inputs:
    - **Intent Recognition**: The chatbot recognizes the user's intent based on specific keywords or phrases like info, task, joke and unknown. For example:
        - **"How are you?"**: The chatbot randomly selects a response from a list of predefined responses.
        - **"Tell me a joke"**: The chatbot provides a random joke from a list of jokes.
        - **"Capital of <Country>"**: The chatbot looks up the capital city of a country (if available) and provides an answer.
        - **Exit**: If the user types "bye" or "goodbye," the chatbot ends the conversation.

Each of these intents is recognized by matching specific keywords in the user's input.

3. **Fallback and Error Handling**:
    - If the chatbot doesn't recognize the user's input (e.g., when the user asks something outside the predefined intents), it falls back to a set of suggestions like asking for a world capital, telling a joke, or offering help. This ensures the chatbot remains useful even if the user's request is unclear.
    - **Email Error Handling**: If the user inputs an invalid email (e.g., "abhi"), the chatbot prompts them to provide a valid email.

- **General Error Handling**: For any input that doesn't match a predefined intent, the chatbot responds with fallback suggestions, asking the user to rephrase or offering alternative topics of conversation.

Explanation of Intents Used:

1. **"How are you?"**:
   - This intent is triggered when the user asks the chatbot how it is doing. The chatbot responds with a random mood-based response from a predefined list.
2. **"Tell me a joke"**:
   - This intent is triggered when the user asks the chatbot to tell them a joke. The chatbot selects a joke randomly from a list and shares it with the user.
3. **"Capital of <Country>"**:
   - This intent is triggered when the user asks for the capital of a specific country (e.g., "What is the capital of France?"). The chatbot looks up the capital in a predefined dictionary of country capitals and returns the answer.
4. **Exit**:
   - This intent is activated when the user types "bye" or "goodbye." The chatbot ends the conversation with a farewell message.
5. **Fallback**:
   - This intent is triggered when the chatbot doesn't understand the user's input. It falls back to suggesting alternative actions, such as asking for a world capital or telling a joke.
6. **Email Collection & Validation**:
   - This intent is triggered when the user needs to provide or confirm their email address. The chatbot validates the email format using a regex and prompts the user to enter a valid email if the initial input is incorrect.

Summary of Error Handling and Fallbacks:

- **Email validation errors**: If an invalid email is provided (e.g., "aseem"), the chatbot will respond by asking the user to enter a valid email address.
- **Unrecognized user inputs**: If the chatbot cannot identify the user's intent, it will respond with fallback suggestions, guiding the user to ask about world capitals, jokes, or seek help.
- **General conversation errors**: The chatbot gracefully handles unexpected inputs by offering relevant options for the user to continue interacting.

This ensures the chatbot provides a smooth and informative experience, even when encountering errors or ambiguous inputs.

UML DIAGRAM