Language Learning Chatbot -Documentation

1. Introduction

The Language Learning Chatbot is designed to assist users in learning a new language by engaging them in interactive conversations. The bot utilizes Ollama (Llama3 model) as its language model and LangChain to structure AI-driven responses. It enables users to practice conversations in different real-life scenarios while tracking mistakes for improvement.

2. System Architecture

1.Architecture OverviewUser Input Collection: The bot gathers user details (language to learn, native language, proficiency level, and scenario selection).

2.Scenario Generation: Based on user inputs, the bot generates an interactive learning scenario.

3.Conversation Handling: AI-driven conversation in the target language, with responses adapted to the user's level.

4.Mistake Tracking: Identifies user mistakes and logs them into an SQLite database.

5.Progress Report: At the end of the session, the bot provides a summary of the user's performance and areas for improvement.

3. Technology Stack

Component	**Technology Used**
AI Model	OpenAI's GPT-4
Framework	LangChain
Database	SQLite (for tracking mistakes)
Programming Language	Python
Logging	Python Logging Module

4.Features

A. User Input Collection

The bot collects:

- User's name
- Language to learn
- Native language
- Proficiency level (A1-C2)
- Scenario choice (Restaurant, Airport, Hotel, Market)

B. AI-Powered Conversation

- Generates scenario-specific conversations
- Maintains conversation history
- Provides responses in the target language

C. Mistake Tracking & Feedback

- Tracks user mistakes during conversation
- Saves mistakes into an SQLite database
- Generates a personalized report

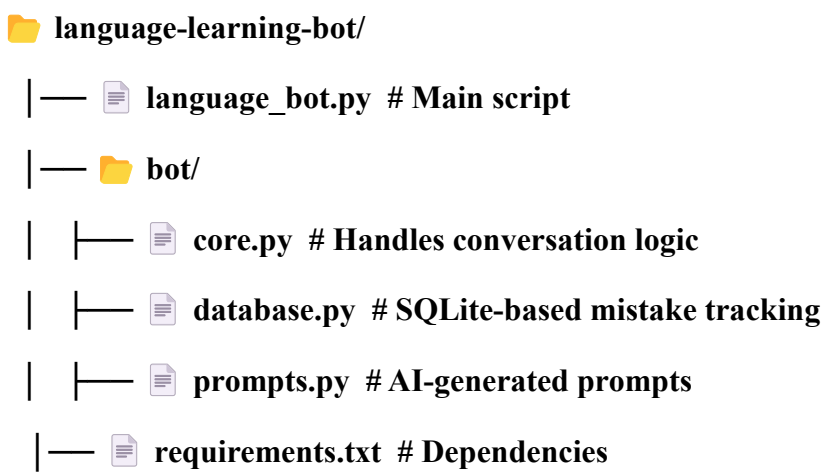
D. Learning Report Generation

- Summarizes conversation history
- Highlights common mistakes and areas for improvement
-

5. How It Works

- User selects a language and scenario.
- Chatbot initiates a conversation using the selected scene.
- User interacts with the chatbot, and responses are processed via LangChain.
- Bot identifies mistakes and logs them in SQLite.
- At the end of the session, a progress report is generated.

6. Code Structure



7. Installation & SetupStep

1: Install Dependencies

pip install -r requirements.txtStep

2: Run the Chatbot

python language_bot.py

8. Future Improvements

- Speech-to-Text: Convert spoken conversations into text.
- Multi-turn Memory: Improve conversation flow.
- More Scenarios: Add diverse real-life situations.

9.Code

```python
import os
from typing import Dict, List
import logging
from langchain_community.llms import Ollama
from langchain.chains import LLMChain
from langchain.prompts import PromptTemplate

# Configure logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler('language_bot.log'),
        logging.StreamHandler()
    ]
)
logger = logging.getLogger(__name__)

class LanguageLearningBot:
    """Local language learning chatbot using Ollama"""

    def __init__(self):
        """Initialize with local LLM"""
        try:
            self.llm = Ollama(model="llama3")
            self.user_data: Dict[str, str] = {}
            self.conversation_history: List[str] = []
            logger.info("Bot initialized with local LLM")
        except Exception as e:
            logger.error(f"Initialization failed: {str(e)}")
            print("ERROR: Make sure Ollama is running (download from ollama.ai)")
            exit(1)

    def _get_valid_input(self, prompt: str, validator=None, error_msg: str = "Invalid input"):
        """Universal input validator"""
        while True:
            try:
                user_input = input(prompt).strip()
                if not user_input:
                    raise ValueError("Input cannot be empty")
```

```
        while True:
            try:
                choice = self._get_valid_input("Enter number (1-4): ")
                choice_num = int(choice)
                if 1 <= choice_num <= len(scenarios):
                    self.user_data['scene'] = scenarios[choice_num-1]
                    break
                print(f"Please enter between 1-{len(scenarios)}")
            except ValueError:
                print("Numbers only please")

    def generate_scenario(self) -> str:
        """Generate learning scenario"""
        try:
            prompt = PromptTemplate.from_template(
                """Create a {target_lang} learning scenario for {level} level.
                Native language: {native_lang}. Scenario: {scene}.
                Start with an opening line in {target_lang}."""
            )
            chain = LLMChain(llm=self.llm, prompt=prompt)
            return chain.run(**self.user_data)
        except Exception as e:
            logger.error(f"Scenario error: {str(e)}")
            return f"Let's practice {self.user_data['target_lang']} in {self.user_data['scene']}!"

    def start_conversation(self):
        """Main conversation loop"""
        print(f"\n💬 Starting {self.user_data['scene']} scenario")
        print("Type 'quit' to exit\n")

        scenario = self.generate_scenario()
        print(f"\n{scenario}\n")

        conv_prompt = PromptTemplate.from_template(
            """You're a {target_lang} teacher in {scene}.
            Respond naturally in {target_lang} at {level} level.
            Keep responses brief.
            Conversation:
```

```
                    raise ValueError("Input cannot be empty")
                if validator and not validator(user_input):
                    raise ValueError(error_msg)
                return user_input
            except ValueError as e:
                print(f"Error: {str(e)}")

    def collect_user_info(self):
        """Collect and validate user information"""
        print("\n🌎 Welcome to Language Learning Bot!")

        self.user_data['user_id'] = self._get_valid_input("Enter your name: ")

        self.user_data['target_lang'] = self._get_valid_input(
            "Language to learn (e.g., Spanish): ",
            lambda x: len(x) >= 2,
            "Minimum 2 characters required"
        )

        self.user_data['native_lang'] = self._get_valid_input(
            "Your native language (e.g., English): ",
            lambda x: len(x) >= 2,
            "Minimum 2 characters required"
        )

        valid_levels = ['A1', 'A2', 'B1', 'B2', 'C1', 'C2']
        self.user_data['level'] = self._get_valid_input(
            f"Your {self.user_data['target_lang']} level (A1-C2): ",
            lambda x: x.upper() in valid_levels,
            f"Must be one of: {', '.join(valid_levels)}"
        ).upper()

        print("\nChoose scenario:")
        scenarios = ["Restaurant", "Airport", "Hotel", "Market"]
        for i, scene in enumerate(scenarios, 1):
            print(f"{i}. {scene}")

        while True:
```

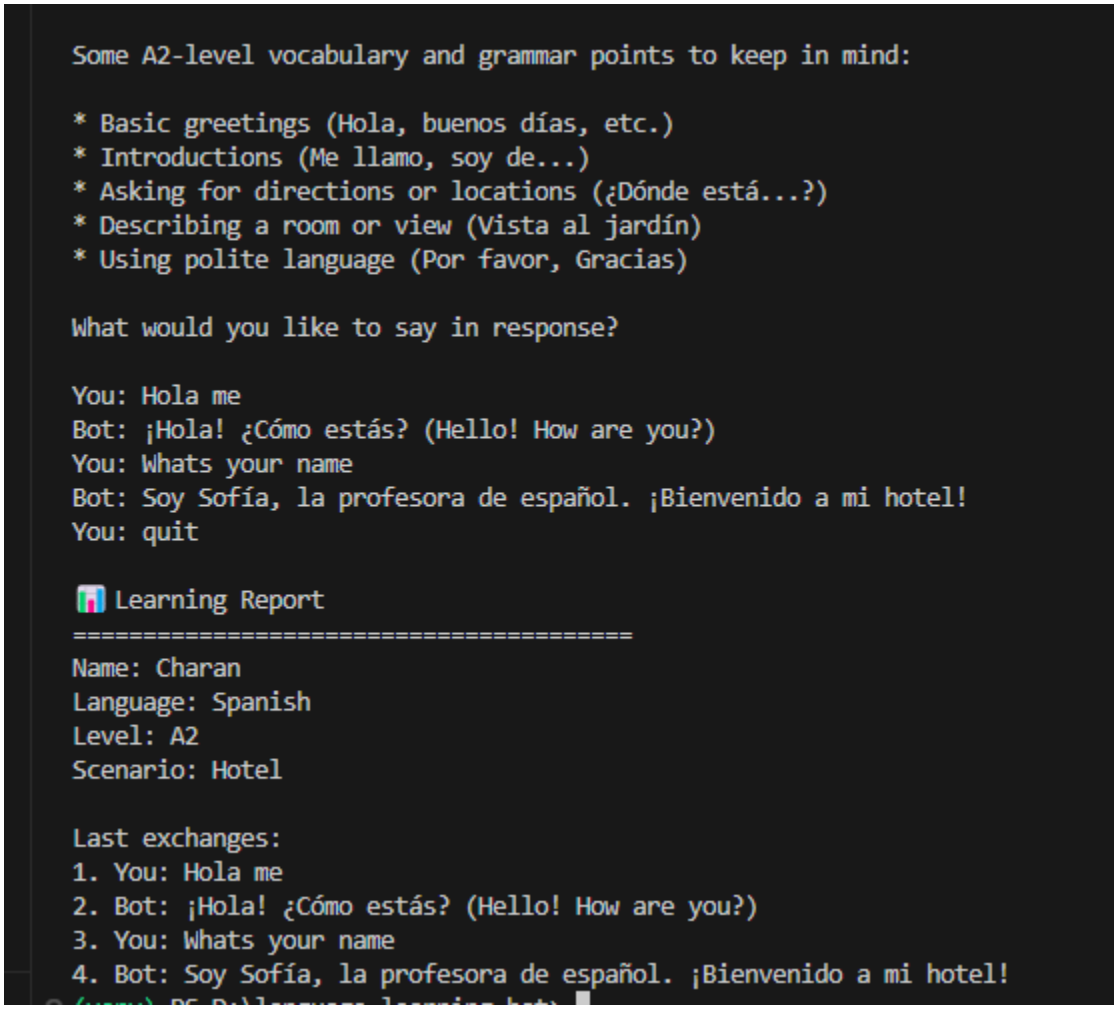
```
            print("Let's try that again")

    def generate_report(self):
        """Generate learning progress report"""
        print("\n📊 Learning Report")
        print("="*40)
        print(f"Name: {self.user_data['user_id']}")
        print(f"Language: {self.user_data['target_lang']}")
        print(f"Level: {self.user_data['level']}")
        print(f"Scenario: {self.user_data['scene']}")

        if self.conversation_history:
            print("\nLast exchanges:")
            for i, msg in enumerate(self.conversation_history[-4:], 1):
                print(f"{i}. {msg}")
        else:
            print("\nNo conversation history yet")
```

```
            Conversation:
            {history}
            Student: {input}
            Teacher:"""
        )

        while True:
            try:
                user_input = input("You: ").strip()
                if user_input.lower() == 'quit':
                    break
                if not user_input:
                    continue

                response = LLMChain(llm=self.llm, prompt=conv_prompt).run(
                    input=user_input,
                    history="\n".join(self.conversation_history[-3:]),
                    **self.user_data
                )
                print(f"Bot: {response}")
                self.conversation_history.append(f"You: {user_input}")
                self.conversation_history.append(f"Bot: {response}")

            except KeyboardInterrupt:
                print("\nEnding conversation...")
                break
            except Exception as e:
                logger.error(f"Conversation error: {str(e)}")
                print("Let's try that again")

    def generate_report(self):
        """Generate learning progress report"""
        print("\n📊 Learning Report")
        print("="*40)
        print(f"Name: {self.user_data['user_id']}")
        print(f"Language: {self.user_data['target_lang']}")
        print(f"Level: {self.user_data['level']}")
        print(f"Scenario: {self.user_data['scene']}")
```

10. OUTPUT:

```
🌍 Welcome to Language Learning Bot!
Enter your name: Charan
Language to learn (e.g., Spanish): Spanish
Your native language (e.g., English): English
Your Spanish level (A1-C2): A2

Choose scenario:
1. Restaurant
2. Airport
3. Hotel
4. Market
Enter number (1-4): 3
```

```
Here's a Spanish learning scenario for A2 level:

**Scenario:** You're on vacation at a hotel in Spain, and you need to check-in. You arrive at the rec

**Opening Line in Spanish:**

"Hola, tengo una reserva para un cuarto individual con vista al jardín, ¿dónde está?"

Translation: "Hello, I have a reservation for a single room with a garden view, where is it?"

Your task is to respond and continue the conversation as if you were really checking-in at the hotel.

Some A2-level vocabulary and grammar points to keep in mind:

* Basic greetings (Hola, buenos días, etc.)
* Introductions (Me llamo, soy de...)
* Asking for directions or locations (¿Dónde está...?)
* Describing a room or view (Vista al jardín)
* Using polite language (Por favor, Gracias)

What would you like to say in response?

You: Hola me
Bot: ¡Hola! ¿Cómo estás? (Hello! How are you?)
You: Whats your name
Bot: Soy Sofía, la profesora de español. ¡Bienvenido a mi hotel!
You: quit

📊 Learning Report
```


Some A2-level vocabulary and grammar points to keep in mind:
* Basic greetings (Hola, buenos días, etc.)
* Introductions (Me llamo, soy de...)
* Asking for directions or locations (¿Dónde está...?)
* Describing a room or view (Vista al jardín)
* Using polite language (Por favor, Gracias)
What would you like to say in response?
You: Hola me
Bot: ¡Hola! ¿Cómo estás? (Hello! How are you?)
You: Whats your name
Bot: Soy Sofía, la profesora de español. ¡Bienvenido a mi hotel!
You: quit
Learning Report
Name: Charan
Language: Spanish
Level: A2
Scenario: Hotel
Last exchanges:
1. You: Hola me
2. Bot: ¡Hola! ¿Cómo estás? (Hello! How are you?)
3. You: Whats your name
4. Bot: Soy Sofía, la profesora de español. ¡Bienvenido a mi hotel!

THANK YOU