Chatbot Using Google Generative AI

Dr.R. Prema	**V.Sriram Charan ,V.K.S.Karthik ,K.Rahul**
AP/CSE	**B.E(CSE)**
SCSVMV University	**SCSVMV University**
Kanchipuram	**Kanchipuram**

Introduction:

A chatbot is a software application designed to mimic human conversation, typically through text or voice interactions. It uses artificial intelligence (AI) to understand and respond to user inputs in real-time. Chatbots serve various purposes, such as providing customer support, retrieving information, or assisting with tasks. They can operate based on predefined rules or leverage machine learning to enhance their conversational abilities over time. These virtual assistants are commonly integrated into websites, messaging apps, and mobile platforms, helping businesses streamline operations and improve user engagement by offering quick, efficient, and interactive solutions to meet users' needs.

Types of Chatbots

1. Rule-Based Chatbots
2. AI-Powered Chatbots
3. Hybrid Chatbots

Features of the chatbot is to have advances in AI and NLP are expected to make chatbots more intuitive, capable of holding more human-like conversations, and seamlessly integrating with various devices and applications.

- **Rule-Based Chatbots**:
  - Operate on predefined rules and scripts.
  - Limited in scope, only capable of answering specific queries.
  - Example: Automated customer service bots for FAQs.

 AI-Powered Chatbots:

- Use machine learning and NLP to understand user inputs.
- Continuously learn and improve over time.
- Can handle complex queries and engage in contextual conversations.
- Example: Virtual assistants like Siri, Google Assistant, and GPT-based bots.

 Hybrid Chatbots:

- Combine rule-based and AI-powered approaches.
- Provide structured responses while improving with AI-driven learning.

Objectives:

- To develop a conversational AI chatbot that simulates human conversation.
- To utilize Google's pre-trained generative model for generating high-quality responses.
- To ensure efficient, non-blocking user interactions using asynchronous programming (asyncio).

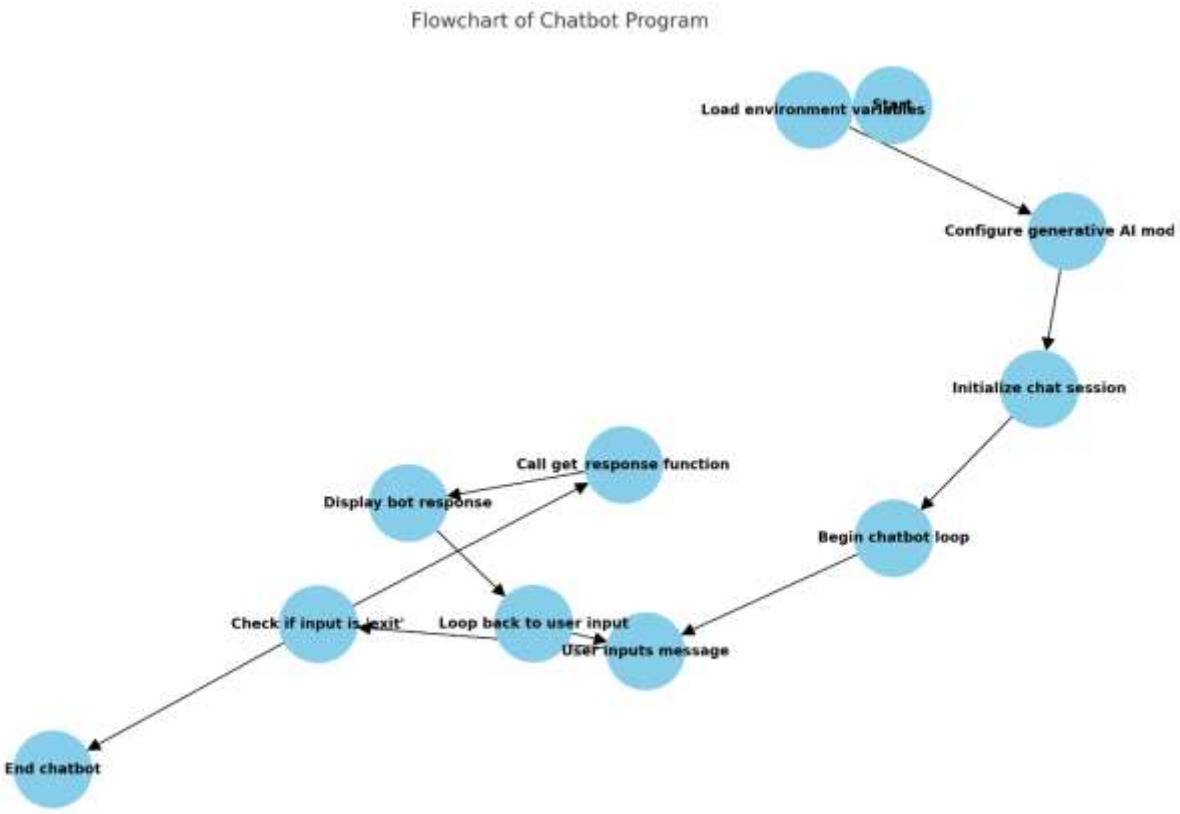
Tools and Technologies:

Technology	Purpose
Python	Core programming language used for development.
Google Generative AI	Provides the generative model (gemini-1.5-flash) for conversation.
dotenv	Loads environment variables (e.g., API keys) securely.
asynscio	Enables asynchronous programming for non-blocking interaction.

Libraries Used In The Chatbot:

- google.generativeai
- dotenv
- os
- asyncio

Flow Chart:



Implementation:

*Save this file with any name you want with .py at the end of the name.

```
import google.generativeai as genai
from dotenv import load_dotenv
import os
import asyncio


# Load environment variables
load_dotenv()
api_key = os.getenv("GOOGLE_API_KEY")
```

```
# Configure the generative AI model
genai.configure(api_key=api_key)
flash = genai.GenerativeModel(
    'gemini-1.5-flash',
    generation_config=genai.GenerationConfig(max_output_tokens=20)
    )
chat = flash.start_chat(history=[])

# Change get_response to a synchronous function
def get_response(chat, user_input):
    response = chat.send_message(user_input)
    return response.text

async def chatbot():
    print("Chatbot is ready! Type 'exit' to quit.")

    while True:
        user_input = input("You: ")
        if user_input.lower() == "exit":
            print("Goodbye!")
            break

        # Use asyncio.to_thread to run get_response in a thread
        response = await asyncio.to_thread(get_response, chat, user_input)
        print("Bot:", response)

if __name__ == "__main__":
```

```
asyncio.run(chatbot())
```

*Save this as .env in the same folder as the above.

```
GOOGLE_API_KEY=AIzaSyAl5HaMxMVUCCwQlLq7r44Kndk-96Yibnc
```

Advantages:

- 24/7 Availability
- Cost-Effective
- Quick Response Times
- Integration with Other Systems

Disadvantages:

- Limited Understanding
- Lack of Personalization
- Dependence on Pre-Programmed Responses
- Inability to Handle Emotional Contexts

Applications:

- E-commerce
- Education
- HR and Recruitment
- Banking and Finance

Conclusion:

This project demonstrates the implementation of a responsive, real-time chatbot using Google Generative AI. By combining modern AI models with asynchronous programming, the chatbot achieves efficient and secure user interactions. With future improvements, the system can be scaled and adapted to various domains, such as customer support, education, and entertainment.

References:

1. Google Generative AI Documentation
2. Python asyncio Documentation
3. dotenv Python Package Documentation

Done by,

V.Sriram Charan(11239A098)

V.K.S.Karthik(11239A097)

K.Rahul(11239M005)