**PERSONAL AGENTIC AI CHATBOT**

**Abstract:**

A personal agentic AI chatbot is an advanced system designed to function autonomously, adapt dynamically to user needs, and execute complex tasks with minimal human intervention. It utilizes cutting-edge Generative AI (GenAI) techniques to understand natural language, process vast amounts of data, and provide intelligent, context-aware responses. Unlike traditional rule-based chatbots, which follow predefined scripts, an agentic AI chatbot operates with a higher degree of autonomy, reasoning, and decision-making capabilities. By leveraging state-of-the-art technologies such as LangGraph for managing conversation flows, FastAPI for high-performance backend processing, Streamlit for interactive user interfaces, and models like OpenAI’s GPT, Groq, Meta Llama, and Mistral, the chatbot achieves superior natural language understanding, scalability, and efficiency.  
Performance and scalability are crucial aspects of an agentic chatbot's design. With efficient backend frameworks like FastAPI, the chatbot ensures rapid response times, even when handling concurrent user requests. Distributed computing and load balancing techniques can be employed to scale the system for enterprise use cases, ensuring that the chatbot remains responsive under high traffic conditions. Caching mechanisms further optimize performance by reducing redundant computations and improving efficiency.

The chatbot also incorporates multi-modal capabilities, allowing it to process and generate text, voice, and even images if needed. Speech-to-text (STT) and text-to-speech (TTS) technologies enable voice interactions, making the chatbot more accessible to users who prefer verbal communication. For vision-based AI models, image recognition and generation can be integrated, expanding the chatbot’s applications to domains such as healthcare diagnostics, creative design, and security monitoring..

**INTRODUCTION**

A personal AI chatbot is an intelligent virtual assistant designed to engage in human-like conversations, answer queries, and perform tasks based on user input. Unlike traditional chatbots with predefined responses, modern AI chatbots leverage advanced Generative AI techniques to provide context-aware, dynamic, and interactive responses.

Key Features of a Personal AI Chatbot

Conversational Intelligence – Uses Natural Language Processing (NLP) and Machine Learning (ML) to understand and generate human-like text.

Real-time Assistance – Provides instant responses, retrieves information from the web, and assists with daily tasks.

Personalization – Adapts to user preferences and maintains context for meaningful interactions.

Scalability & Performance – Built with frameworks like FastAPI and LangGraph for efficient request handling and conversation management.

Multi-Model Support – Utilizes OpenAI, Groq, Meta Llama, and Mistral for generating responses with diverse AI capabilities.

Web Search & Integration – Fetches live information and integrates with APIs to enhance knowledge and accuracy.

GUI & Deployment – Interfaces with Streamlit for a user-friendly experience and can be deployed across multiple platforms.

Applications

Personal virtual assistant

Knowledge retrieval and summarization

Coding and debugging support

Real-time decision-making

Customer service automation

Backend Implementation

FastAPI is chosen for its asynchronous capabilities, making the chatbot highly responsive even under heavy loads. Key backend functionalities include.

Handling user requests and routing them to appropriate LLMs.

Managing API calls efficiently to optimize performance and reduce latency.

Implementing caching mechanisms to speed up response times.

Logging and monitoring chatbot interactions for continuous improvement.

Frontend Development

The frontend is built using Streamlit, a Python-based framework ideal for rapid UI prototyping and deployment. The user interface includes:

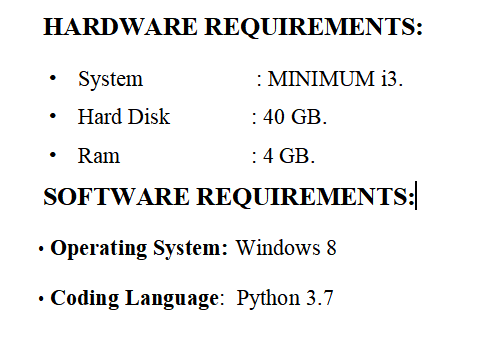
A simple chat-based interface for interacting with the chatbot.

Dynamic response rendering with text, links, and formatted outputs.

User settings for customizing chatbot behavior and preferences.

Session memory to maintain chat history across user interactions.

**SYSTEM REQUIREMENTS:**



**Existing System:**

**Limitations of Traditional Chatbots**

Traditional chatbot systems operate using predefined rules and scripts, which limit their ability to handle complex or ambiguous queries. These systems suffer from:

* **Lack of Context Awareness**: They cannot remember past interactions or adapt to user preferences.
* **Limited Conversational Abilities**: Rule-based responses lead to unnatural and robotic conversations.
* **High Maintenance Costs**: Constant updates are needed to address new queries and improve responses.
* **Inability to Handle Ambiguity**: Traditional models fail when presented with vague or unexpected inputs.

**Why a Generative AI-Based Chatbot?**

The need for an **intelligent, adaptive, and scalable** chatbot solution has led to the emergence of LLM-powered chatbots. These models generate responses dynamically, ensuring more human-like interactions and improved user satisfaction.

**3. System Under Proposal**

The proposed **Personal AI Chatbot** addresses the shortcomings of traditional systems by integrating **state-of-the-art Generative AI techniques**. The chatbot utilizes **LangGraph for agent orchestration**, **FastAPI for backend processing**, and **Streamlit for frontend interactions**.

**Key Features**

1. **Natural Language Understanding (NLU):**
   * Uses advanced LLMs (OpenAI, Groq, Meta Llama, and Mistral) for high-quality text generation.
   * Supports multiple languages and context tracking.
2. **Dynamic Context Retention:**
   * Maintains conversation history for coherent and personalized interactions.
   * Uses memory-efficient techniques to manage long-term user preferences.
3. **Web Search Integration:**
   * Retrieves real-time information from the internet for up-to-date responses.
   * Summarizes search results for concise answers.
4. **Multi-Model Support:**
   * Selects the best model based on query complexity and response accuracy.
   * Ensures fallback mechanisms for enhanced reliability.
5. **Task Automation & Assistance:**
   * Provides personalized recommendations based on user input.
   * Supports scheduling, reminders, and data retrieval.
6. **Secure & Scalable Architecture:**
   * Built using FastAPI for efficient request handling.
   * Deployed with robust security measures, including authentication and encryption.
7. **User-Friendly Frontend:**
   * Streamlit-based UI for seamless interaction.
   * Responsive design with easy navigation and customization options.

**LITERATURE SURVEY**

1. Guendalina Caldarini, Sardar Jaf, Kenneth McGarry (2022)

Title: A Literature Survey of Recent Advances in Chatbots

Methodology: This study reviews recent advancements in chatbots, focusing on the integration of artificial intelligence and natural language processing techniques.

Result: The authors identify key challenges and limitations in current chatbot applications, providing insights into areas requiring further research.

Limitation: The survey emphasizes the need for continued investigation to overcome existing challenges in chatbot development.

2. Md. Al-Amin et al. (2024)

Title: History of Generative Artificial Intelligence (AI) Chatbots: Past, Present, and Future Development

Methodology: This comprehensive review traces the evolution of chatbot technology from early rule-based systems to modern AI-powered conversational agents.

Result: The paper highlights significant milestones and innovations that have shaped the development of contemporary chatbots.

Limitation: The study underscores the necessity for ongoing innovation to address emerging challenges in chatbot technology.

3. Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, Chaoning Zhang (2024)

Title: A Complete Survey on LLM-based AI Chatbots

Methodology: This survey examines the evolution and deployment of large language model (LLM)-based chatbots across various sectors.

Result: The authors explore diverse applications of LLM-based chatbots and discuss open challenges in the field.

Limitation: The paper addresses issues related to data usage and the potential misuse of generated knowledge by chatbots.

4. Hamed Khosravi et al. (2023)

Title: Chatbots and ChatGPT: A Bibliometric Analysis and Systematic Review of Publications in Web of Science and Scopus Databases

Methodology: This bibliometric analysis investigates scientific literature related to chatbots, with a focus on ChatGPT, by analyzing data from Scopus and Web of Science databases.

Result: The study identifies key areas of interest and significant topics for future investigation in the chatbot field.

Limitation: The authors emphasize the need for more comprehensive studies to understand the broader impact of chatbots.

**SYSTEM STUDY**

Traditional chatbot systems operate using predefined rules and scripts, which limit their ability to handle complex or ambiguous queries. These systems suffer from:

* Lack of Context Awareness: They cannot remember past interactions or adapt to user preferences.
* Limited Conversational Abilities: Rule-based responses lead to unnatural and robotic conversations.
* High Maintenance Costs: Constant updates are needed to address new queries and improve responses.
* Inability to Handle Ambiguity: Traditional models fail when presented with vague or unexpected inputs.

Limitations of Existing Chatbots

1. Rule-Based Constraints: Responses are predefined and cannot adapt dynamically.
2. Inconsistent User Experience: Users often receive irrelevant or repetitive answers.
3. Scalability Issues: Struggles with high-volume queries and diverse topics.
4. Dependency on Manual Updates: Requires continuous intervention to remain relevant.
5. Limited Integration Capabilities: Cannot interact effectively with APIs or external data sources.

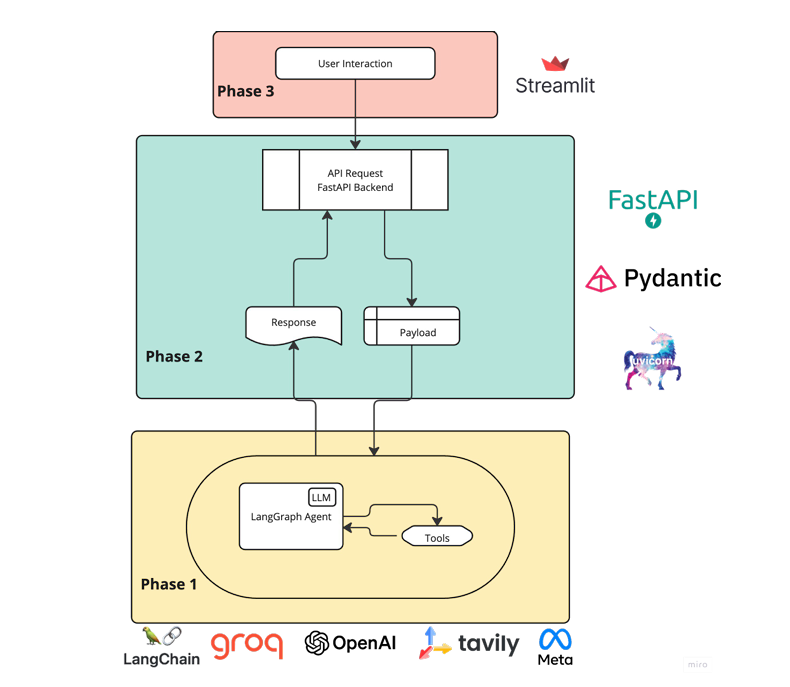
Proposed System:

The proposed Personal AI Chatbot addresses the shortcomings of traditional systems by integrating state-of-the-art Generative AI techniques. The chatbot utilizes LangGraph for agent orchestration, FastAPI for backend processing, and Streamlit for frontend interactions.

Key Features of the Proposed System

1. Enhanced Conversational Intelligence:
   * Supports dynamic, context-aware interactions.
   * Provides natural language understanding (NLU) with deep learning models.
2. Memory Retention & Context Awareness:
   * Maintains previous conversation history.
   * Adapts responses based on user interactions.
3. Real-Time Web Search Integration:
   * Retrieves live data to provide up-to-date responses.
   * Summarizes web search results for better understanding.
4. Multi-Model Integration:
   * Leverages OpenAI, Groq, Meta Llama, and Mistral models for improved accuracy.
   * Ensures fallback mechanisms for enhanced reliability.
5. Automated Task Execution:
   * Assists with scheduling, reminders, and data retrieval.
   * Provides personalized recommendations.
6. Scalability & Security:
   * Designed for high user traffic with FastAPI optimization.
   * Implements robust encryption and authentication measures.
7. User-Friendly UI:
   * Built with Streamlit for intuitive and interactive usage.
   * Provides customization options for enhanced user experience

**SYSTEM DESIGN**



Phase 1: AI Model & Tools

This phase involves LangChain, Groq, OpenAI, and other AI-related tools.

The system interacts with an LLM (Large Language Model) to generate responses based on the input.

The LangChain framework is used to manage prompts and responses.

Phase 2: Backend API

The FastAPI framework is used for handling API requests.

Pydantic helps with data validation and structuring.

The API receives user queries, processes them, and sends requests to the AI model.

The response from the AI model is formatted and sent back.

Phase 3: User Interaction

The Streamlit framework is used for the frontend UI.

Users interact with the chatbot via a web interface.

The frontend sends requests to the backend (FastAPI), and responses are displayed to users.

**SOFTWARE ENVIRONMENT**

A personal agentive chatbot operates in a dynamic software environment that integrates multiple technologies to provide a seamless and interactive user experience. The chatbot's architecture typically consists of a natural language processing (NLP) module, a dialogue management system, and a knowledge base. The NLP module, powered by machine learning algorithms and libraries like NLTK, spaCy, or Stanford CoreNLP, analyzes user inputs and extracts meaningful information. The dialogue management system, built using frameworks like Rasa, Dialogflow, or Botpress, generates responses and manages the conversation flow. The knowledge base, which can be a graph database like Neo4j or a traditional relational database like MySQL, stores and retrieves relevant information to inform the chatbot's responses.

The chatbot leverages various technologies, including machine learning, deep learning, and cloud services, to provide advanced functionality. Machine learning libraries like TensorFlow, PyTorch, or Scikit-learn enable intent recognition, sentiment analysis, and entity extraction. Deep learning techniques like recurrent neural networks (RNNs) and transformers facilitate advanced NLP tasks. Cloud platforms like AWS, Google Cloud, or Microsoft Azure provide scalability and reliability. The chatbot also integrates with external services like weather APIs, payment gateways, or social media platforms to offer comprehensive assistance.

To ensure security and reliability, the chatbot employs data encryption methods like SSL/TLS or AES, authentication protocols like OAuth or JWT, and access control mechanisms like role-based access control (RBAC) or attribute-based access control (ABAC). The development process follows agile methodologies like Scrum or Kanban, DevOps practices like continuous integration (CI) and continuous deployment (CD), and thorough testing, including unit testing, integration testing, and user acceptance testing (UAT).

The chatbot can be deployed on cloud platforms like AWS, Google Cloud, or Microsoft Azure, on-premises for organizations requiring strict data control and security, or in a hybrid environment that combines cloud and on-premises deployment. This flexible deployment approach enables the chatbot to adapt to various organizational requirements and infrastructure constraints.

In terms of benefits, the chatbot offers numerous advantages, including 24/7 availability, personalized support, and automated task completion. The chatbot can also help organizations reduce support costs, improve customer satisfaction, and increase operational efficiency. Furthermore, the chatbot can provide valuable insights into customer behavior and preferences, enabling organizations to make data-driven decisions and improve their overall business strategy.

To develop a chatbot, organizations can follow a structured approach that involves defining the chatbot's purpose and scope, designing the conversation flow and dialogue, developing the chatbot using various tools and technologies, testing and deploying the chatbot, and continuously monitoring and improving its performance. This approach enables organizations to create a chatbot that meets their specific needs and provides a high-quality user experience.

In conclusion, a personal agentive chatbot is a powerful tool that can provide numerous benefits to organizations and individuals. By leveraging advanced technologies like NLP, machine learning, and cloud services, the chatbot can offer personalized support, automate tasks, and provide valuable insights into customer behavior and preferences.

**INPUT AND OUTPUT DESIGN**

**INPUT DESIGN**

The input design for a personal agentive chatbot refers to the process of defining how users will interact with the chatbot. This includes the types of input the chatbot will accept, the format of the input, and the rules for processing the input.

Types of Input

1. Text Input: Users can type messages to the chatbot using a keyboard or mobile device.

2. Voice Input: Users can speak to the chatbot using voice assistants like Siri, Alexa, or Google Assistant.

3. Button Input: Users can click on buttons or icons to provide input to the chatbot.

Input Format

1. Natural Language: Users can input text or voice in natural language, and the chatbot will use NLP to understand the intent.

2. Structured Input: Users can input data in a structured format, such as filling out a form or selecting from a dropdown menu.

Input Processing

1. Tokenization: The chatbot breaks down user input into individual tokens or words.

2. Part-of-Speech Tagging: The chatbot identifies the part of speech (noun, verb, adjective, etc.) for each token.

3. Named Entity Recognition: The chatbot identifies specific entities (names, locations, organizations, etc.) in the user input.

DESIGN OF OUTPUT   
The output design for a personal agentive chatbot refers to the process of defining how the chatbot will respond to user input. This includes the types of output the chatbot will generate, the format of the output, and the rules for generating the output.

Types of Output

1. Text Output: The chatbot generates text responses to user input.

2. Voice Output: The chatbot generates voice responses to user input using text-to-speech synthesis.

3. Visual Output: The chatbot generates visual responses, such as images, videos, or animations.

Output Format

1. Natural Language: The chatbot generates responses in natural language, using a conversational tone and style.

2. Structured Output: The chatbot generates responses in a structured format, such as a table or a list.

Output Generation

1. Template-Based Generation: The chatbot uses pre-defined templates to generate responses.

2. Machine Learning-Based Generation: The chatbot uses machine learning algorithms to generate responses based on patterns in the data.

3. Hybrid Generation: The chatbot uses a combination of template-based and machine learning-based generation to produce responses.

Best Practices for Input and Output Design

1. Keep it Simple: Use simple and intuitive input and output formats to make it easy for users to interact with the chatbot.

2. Be Consistent: Use consistent input and output formats throughout the chatbot toavoid confusing users.

3. Use Feedback Mechanisms: Use feedback mechanisms, such as confirmation messages or error messages, to help users understand the chatbot's responses.

4. Test and Iterate: Test the input and output designs with real users and iterate on the design based on user feedback.

**SAMPLE CODE**

# if you dont use pipenv uncomment the following:

from dotenv import load\_dotenv

load\_dotenv()

#Step1: Setup API Keys for Groq, OpenAI and Tavily

import os

GROQ\_API\_KEY=os.environ.get("GROQ\_API\_KEY")

TAVILY\_API\_KEY=os.environ.get("TAVILY\_API\_KEY")

OPENAI\_API\_KEY=os.environ.get("OPENAI\_API\_KEY")

#Step2: Setup LLM & Tools

from langchain\_groq import ChatGroq

from langchain\_openai import ChatOpenAI

from langchain\_community.tools.tavily\_search import TavilySearchResults

openai\_llm=ChatOpenAI(model="gpt-4o-mini")

groq\_llm=ChatGroq(model="llama-3.3-70b-versatile")

search\_tool=TavilySearchResults(max\_results=2)

#Step3: Setup AI Agent with Search tool functionality

from langgraph.prebuilt import create\_react\_agent

from langchain\_core.messages.ai import AIMessage

system\_prompt="Act as an AI chatbot who is smart and friendly"

def get\_response\_from\_ai\_agent(llm\_id, query, allow\_search, system\_prompt, provider):

    if provider=="Groq":

        llm=ChatGroq(model=llm\_id)

    elif provider=="OpenAI":

        llm=ChatOpenAI(model=llm\_id)

    tools=[TavilySearchResults(max\_results=2)] if allow\_search else []

    agent=create\_react\_agent(

        model=llm,

        tools=tools,

        state\_modifier=system\_prompt

    )

    state={"messages": query}

    response=agent.invoke(state)

    messages=response.get("messages")

    ai\_messages=[message.content for message in messages if isinstance(message, AIMessage)]

    return ai\_messages[-1]

# if you dont use pipenv uncomment the following:

from dotenv import load\_dotenv

load\_dotenv()

#Step1: Setup Pydantic Model (Schema Validation)

from pydantic import BaseModel

from typing import List

class RequestState(BaseModel):

    model\_name: str

    model\_provider: str

    system\_prompt: str

    messages: List[str]

    allow\_search: bool

#Step2: Setup AI Agent from FrontEnd Request

from fastapi import FastAPI

from ai\_agent import get\_response\_from\_ai\_agent

ALLOWED\_MODEL\_NAMES=["llama3-70b-8192", "mixtral-8x7b-32768", "llama-3.3-70b-versatile", "gpt-4o-mini"]

app=FastAPI(title="LangGraph AI Agent")

@app.post("/chat")

def chat\_endpoint(request: RequestState):

    """

    API Endpoint to interact with the Chatbot using LangGraph and search tools.

    It dynamically selects the model specified in the request

    """

    if request.model\_name not in ALLOWED\_MODEL\_NAMES:

        return {"error": "Invalid model name. Kindly select a valid AI model"}

    llm\_id = request.model\_name

    query = request.messages

    allow\_search = request.allow\_search

    system\_prompt = request.system\_prompt

    provider = request.model\_provider

    # Create AI Agent and get response from it!

    response=get\_response\_from\_ai\_agent(llm\_id, query, allow\_search, system\_prompt, provider)

    return response

#Step3: Run app & Explore Swagger UI Docs

if \_\_name\_\_ == "\_\_main\_\_":

    import uvicorn

    uvicorn.run(app, host="127.0.0.1", port=9999)

# if you dont use pipenv uncomment the following:

from dotenv import load\_dotenv

load\_dotenv()

#Step1: Setup UI with streamlit (model provider, model, system prompt, web\_search, query)

import streamlit as st

st.set\_page\_config(page\_title="LangGraph Agent UI", layout="centered")

st.title("AI Chatbot Agents")

st.write("Create and Interact with the AI Agents!")

system\_prompt=st.text\_area("Define your AI Agent: ", height=70, placeholder="Type your system prompt here...")

MODEL\_NAMES\_GROQ = ["llama-3.3-70b-versatile", "mixtral-8x7b-32768"]

MODEL\_NAMES\_OPENAI = ["gpt-4o-mini"]

provider=st.radio("Select Provider:", ("Groq", "OpenAI"))

if provider == "Groq":

    selected\_model = st.selectbox("Select Groq Model:", MODEL\_NAMES\_GROQ)

elif provider == "OpenAI":

    selected\_model = st.selectbox("Select OpenAI Model:", MODEL\_NAMES\_OPENAI)

allow\_web\_search=st.checkbox("Allow Web Search")

user\_query=st.text\_area("Enter your query: ", height=150, placeholder="Ask Anything!")

API\_URL="http://127.0.0.1:9999/chat"

if st.button("Ask Agent!"):

    if user\_query.strip():

        #Step2: Connect with backend via URL

        import requests

        payload={

            "model\_name": selected\_model,

            "model\_provider": provider,

            "system\_prompt": system\_prompt,

            "messages": [user\_query],

            "allow\_search": allow\_web\_search

        }

        response=requests.post(API\_URL, json=payload)

        if response.status\_code == 200:

            response\_data = response.json()

            if "error" in response\_data:

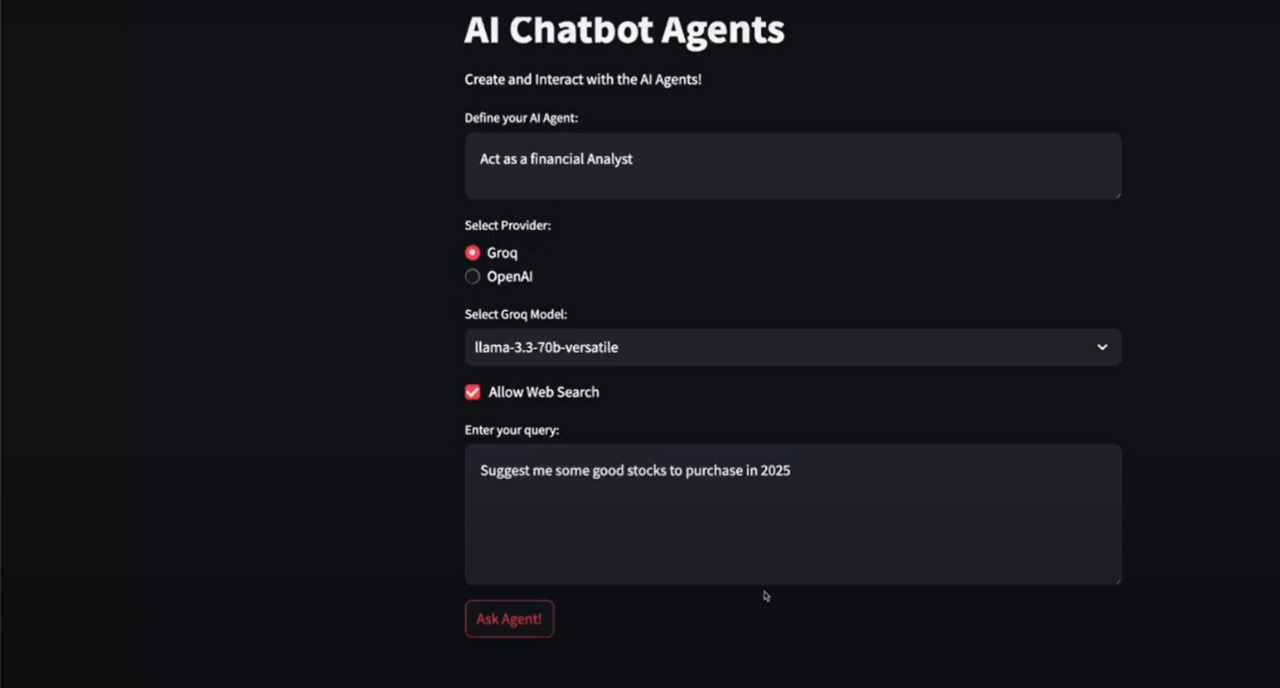
                st.error(response\_data["error"])

            else:

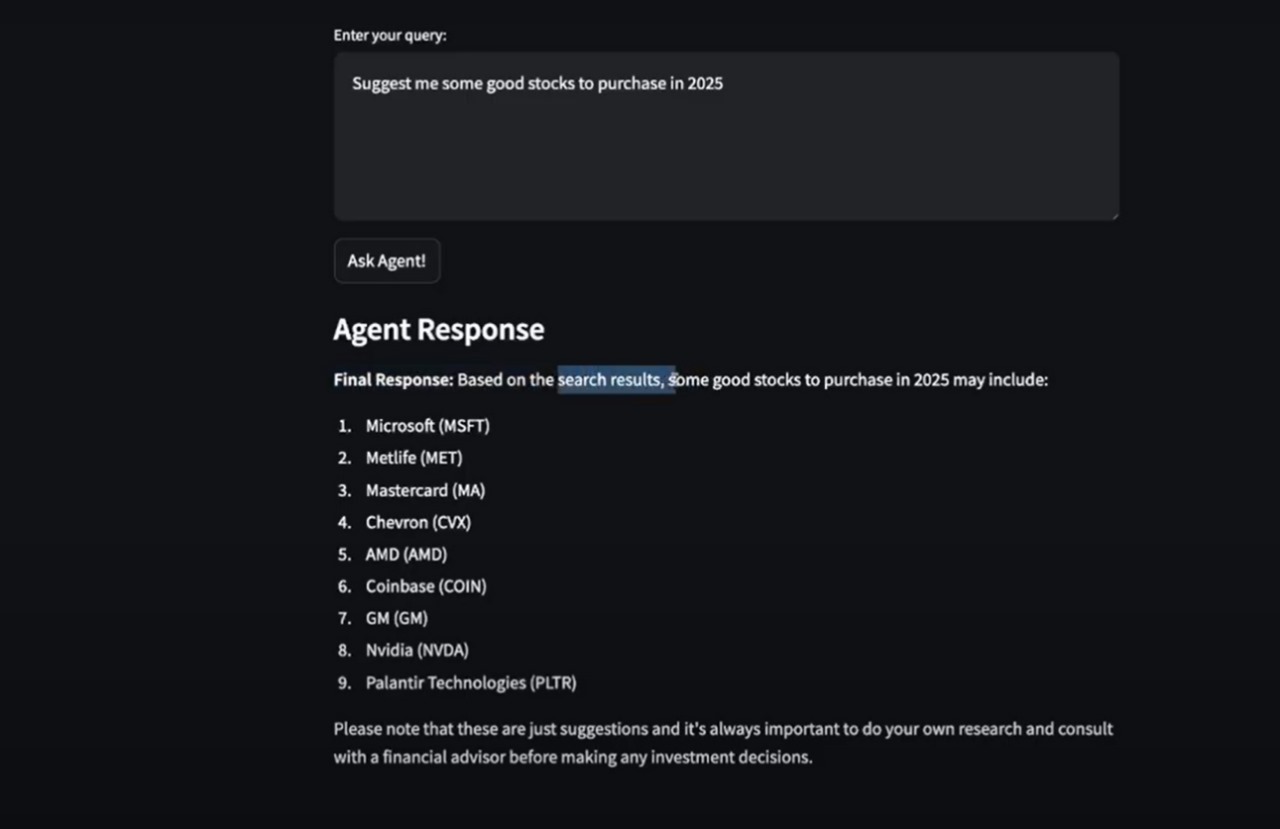
                st.subheader("Agent Response")

                st.markdown(f"\*\*Final Response:\*\* {response\_data}")

**SCREEN SHOTS**



Users can define their AI agent's role, such as acting as a financial analyst, and choose between different AI providers like Groq and OpenAI. In this case, the selected model is LLaMA 3.3-70B (versatile) from Groq, with an option to enable web search for real-time data retrieval. The interface allows users to input queries, such as requesting stock recommendations for 2025, and submit them via the "Ask Agent!" button.



The image displays the response of an AI chatbot to a stock investment query. The user asked the chatbot to suggest good stocks to purchase in 2025. The AI, utilizing web search, provided a list of nine stock recommendations: Microsoft (MSFT), Metlife (MET), Mastercard (MA), Chevron (CVX), AMD (AMD), Coinbase (COIN), General Motors (GM), Nvidia (NVDA), and Palantir Technologies (PLTR). The response also includes a disclaimer advising users to conduct their own research and consult a financial advisor before making investment decisions.

**CONCLUSION**

A personal agentive AI chatbot is a revolutionary technology that has the potential to transform the way we interact with machines. By leveraging advanced natural language processing (NLP), machine learning, and cognitive architectures, a personal agentive AI chatbot can provide personalized support, automate tasks, and offer expert advice.

The key benefits of a personal agentive AI chatbot include:

Enhanced user experience through personalized and human-like interactions

- Increased efficiency and productivity through automation of routine tasks

- Improved decision-making through expert advice and recommendations

- Enhanced customer satisfaction through 24/7 support and quick resolution of issuesHowever, the development of a personal agentive AI chatbot also poses several challenges, including:- Ensuring the chatbot's understanding of natural language and context- Managing the chatbot's knowledge base and keeping it up-to-date

- Ensuring the chatbot's security and protecting user data

- Addressing ethical concerns and ensuring the chatbot's transparency and accountabilityTo overcome these challenges, it is essential to adopt a multidisciplinary approach that combines expertise in AI, NLP, cognitive science, and human-computer interaction. Additionally, it is crucial to prioritize user-centered design, testing, and iteration to ensure that the chatbot meets user needs and expectations.

In conclusion, a personal agentive AI chatbot has the potential to revolutionize human-computer interaction and transform the way we live and work. While there are challenges to be addressed, the benefits of this technology make it an exciting and worthwhile area of research and development.

**REFERENCES**

[1]A. Chan, R. Salganik, A. Markelius, C. Pang, N. Rajkumar, D. Krashenin-nikov, L. Langosco, Z. He, Y. Duan, M. Carroll et al., “Harms fromincreasingly agentic algorithmic systems,” in Proceedings of the 2023ACM Conference on Fairness, Accountability, and Transparency, 2023,pp. 651–66

[2]A. Paul, C. L. Yu, E. A. Susanto, N. W. L. Lau, and G. I. Meadows,“Agentpeertalk: Empowering students through agentic-ai-driven discern-ment of bullying and joking in peer interactions in schools,” arXiv preprintarXiv:2408.01459, 2024.

[3]Y. Shavit, S. Agarwal, M. Brundage, S. Adler, C. O’Keefe, R. Campbell,T. Lee, P. Mishkin, T. Eloundou, A. Hickey et al., “Practices for governingagentic ai systems,” Research Paper, OpenAI, December, 2023.

[4]S. Guenat, P. Purnell, Z. G. Davies, M. Nawrath, L. C. Stringer, G. R. Babu,M. Balasubramanian, E. E. Ballantyne, B. K. Bylappa, B. Chen et al.,“Meeting sustainable development goals via robotics and autonomoussystems,” Nature communications, vol. 13, no. 1, p. 3559, 2022.

[5]E. H. B. Maia, L. C. Assis, T. A. De Oliveira, A. M. Da Silva, and A. G.Taranto, “Structure-based virtual screening: from classical to artiﬁcialintelligence,” Frontiers in chemistry, vol. 8, p. 343, 2020.

[6]S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao,“React: synergizing reasoning and acting in language models (2022),”arXiv preprint arXiv:2210.03629, 2023.

[7]R. Yang, L. Song, Y. Li, S. Zhao, Y. Ge, X. Li, and Y. Shan, “Gpt4tools:Teaching large language model to use tools via self-instruction,” Advancesin Neural Information Processing Systems, vol. 36, 2024.

[8]S. Feuerriegel, J. Hartmann, C. Janiesch, and P. Zschech, “Generative ai,”Business & Information Systems Engineering, vol. 66, no. 1, pp. 111–126,2024.

[9]T. Chong, T. Yu, D. I. Keeling, and K. de Ruyter, “Ai-chatbots on theservices frontline addressing the challenges and opportunities of agency,”Journal of Retailing and Consumer Services, vol. 63, p. 102735, 2021.

[10]L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, “Review of imageclassiﬁcation algorithms based on convolutional neural networks,” RemoteSensing, vol. 13, no. 22, p. 4712, 2021.

[11]A. Brisset, R. Gill, and R. Gannon, “The search for a native language:Translation and cultural identity,” in The translation studies reader.Routledge, 2021, pp. 289–319.

[12]H. Ko, S. Lee, Y. Park, and A. Choi, “A survey of recommendation systems:recommendation models, techniques, and application ﬁelds,” Electronics,vol. 11, no. 1, p. 141, 2022.