MAHARAJA INSTITUTE OF TECHNOLOGY MYSORE

Belawadi, SrirangapatnaTq, Mandya-571477

DEPARTMENT OF CSE (Artificial Intelligence)

Assignment – Project Report

2025-26

Subject Name: Cloud Computing

Subject Code: M23BCS505B

Semester: 5th

Project Title: Creating a Cloud-Enabled Chatbot

Submitted by: Team No:

Sl. No.	Student Name	USN.		CO's Mapping				Total	Scaled to
			CO1	CO2	CO3	CO4	CO5		
1	Lakshmi LK	4MH23CA021							

Verified and Approved by:

Faculty Name: Prof. M J Yogesh

Signature:

Date:

Project GitHub Repository:

Link: https://github.com/LakshmiGowda831/Cloud-Chatbot

1. INTRODUCTION

1.1 Background

Chatbots have become an essential component of digital services, enabling automated interactions for customer support, information retrieval, and task execution. Traditional chatbot systems rely on static rule-based responses, which limit scalability and personalization. With the advancement of cloud computing, AI models, and scalable APIs, modern chatbots can now provide intelligent, real-time, context-aware responses. A cloud-enabled chatbot leverages cloud platforms for model hosting, database operations, user session management, and integration with external APIs. This ensures high availability, reliability, and seamless scalability. The project focuses on building an AI-driven, cloud-hosted chatbot capable of answering user queries, handling conversation flows, and integrating with cloud-based services for storage and deployment.

1.2 Motivation

The motivation behind developing a cloud-enabled chatbot is the rapid digital transformation across industries. Organizations increasingly require scalable virtual assistants that can operate 24/7, respond quickly, and handle large numbers of users concurrently. Cloud platforms like AWS, Azure, and Google Cloud simplify deployment while offering strong security and monitoring tools. This project allowed hands-on experience in AI, NLP, cloud hosting, serverless APIs, and full-stack development.

1.3 Objectives

- To design and develop an intelligent chatbot using NLP and cloud-based services
- To implement cloud hosting for chatbot backend and model APIs
- To integrate a database for storing chat logs and session history
- To build a user-friendly frontend for chatbot interaction
- To support real-time messaging using WebSocket's
- To ensure secure authentication and access control using JWT

1.4 Scope

In-Scope

- **NLP-based chatbot logic**
- **Cloud deployment (AWS/Azure/GCP)**
- **REST API and WebSocket communication**
- **User authentication**

- **Chat history management**

- **Frontend UI (web interface)**

Out-of-Scope

- **Voice-based conversational AI**
- **Integration with IoT devices**
- **Advanced analytics dashboards**
- **Mobile application**
- **Offline chatbot functionality**

2. LITERATURE REVIEW

2.1 Existing Systems

1. Rule-Based Chatbots

Traditional chatbots that rely on predefined rules, patterns, and scripted responses (e.g., ELIZA, AIML). They are simple but lack adaptability and cannot handle dynamic or unseen queries.

2. Retrieval-Based Chatbots

These chatbots select the most appropriate response from a set of predefined replies using similarity matching or ranking algorithms. They do not generate new responses but retrieve the closest match.

3. Generative AI Chatbots

Modern systems such as ChatGPT, Dialog flow CX, and IBM Watson use machine learning, deep learning, and transformer-based models to generate real-time, human-like responses. They understand context and can handle a wide range of queries.

4. Cloud Chatbot Platforms

Cloud-based platforms that provide prebuilt conversational tools:

- **Google Dialog flow:** Simple integration with Google Cloud
- **Microsoft Bot Framework:** Enterprise-level chatbot development
- **Amazon Lex:** AI chatbot engine used by Alexa

These platforms are powerful but often come with higher costs or vendor lock-in, leading many developers to prefer custom cloud-enabled chatbot solutions.

2.2 Key Concepts

Natural Language Processing (NLP)

-Involves tokenization, intent classification, and entity extraction used to interpret user queries.

Cloud Deployment

-Using cloud compute instances, serverless functions, or managed container services to deploy chatbot logic.

REST APIs & WebSockets

-REST for standard operations; WebSockets for real-time bidirectional communication.

JWT Authentication

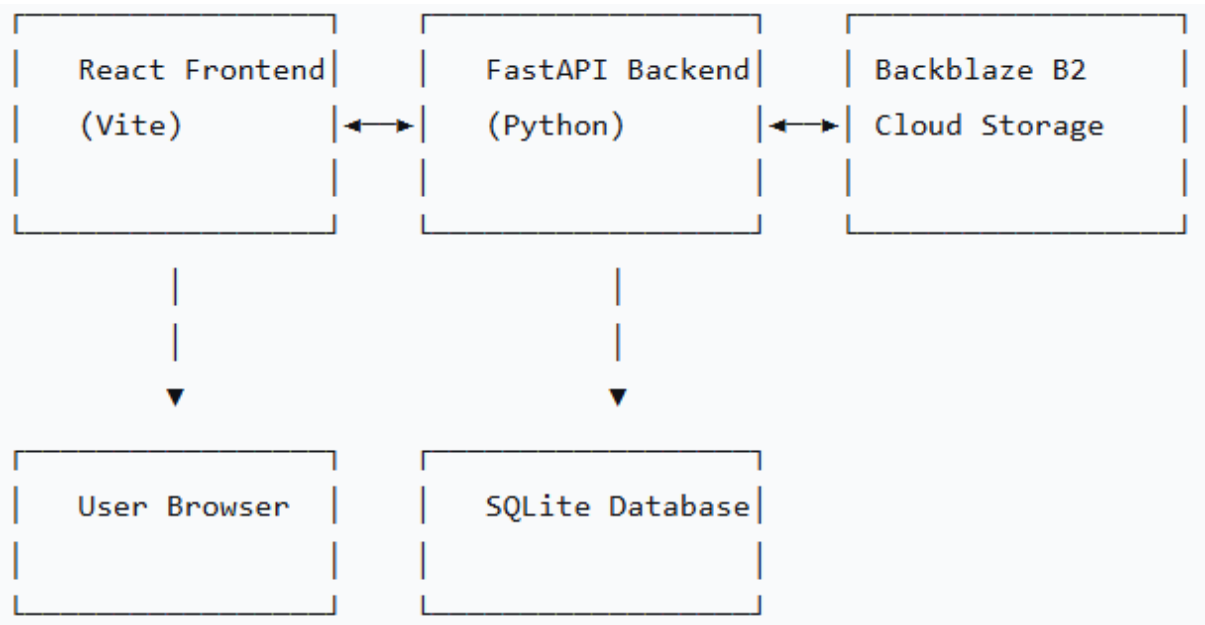
-Ensures secure access to chatbot APIs and user-specific session tracking.

Database Storage

-Dynamic Storage

3. SYSTEM DESIGN

3.1 System Architecture Diagram



3.2 Architecture Explanation

Frontend (HTML,CSS)

Handles UI, chat bubbles, message sending, and session storage.

Backend (python,FastAPI)

Processes user requests, manages authentication, calls NLP models, and handles database operations.

Cloud NLP Engine

Ollama and Hugging Face

Database

Stores:

- **User profiles**
- **Chat history**
- **Session tokens**
- **Analytics logs**

Cloud Hosting

App deployed using:

- **Locally**

3. IMPLEMENTATION DETAILS

Backend

- Python FastAPI
- NLP Models (Ollama,Hugging Face, OpenAI API)
- JWT Authentication

Frontend

- HTML
- Tailwind CSS
- WebSocket APIs

Cloud

- Cloud Storage(Dynamic Storage)

3.2 Dataset Description

- Training datasets from Hugging Face or custom intent-based datasets
- User queries and chatbot responses
- Session metadata

3.3 Step-by-Step Implementation

Module 1: Server Access

- Accessing localhost server

Module 2: Chat Interface

- Chat UI with message threads
- Typing animation
- Real-time WebSocket support

Module 3: NLP Engine

- Intent classification
- Entity extraction
- Response generation using models or APIs

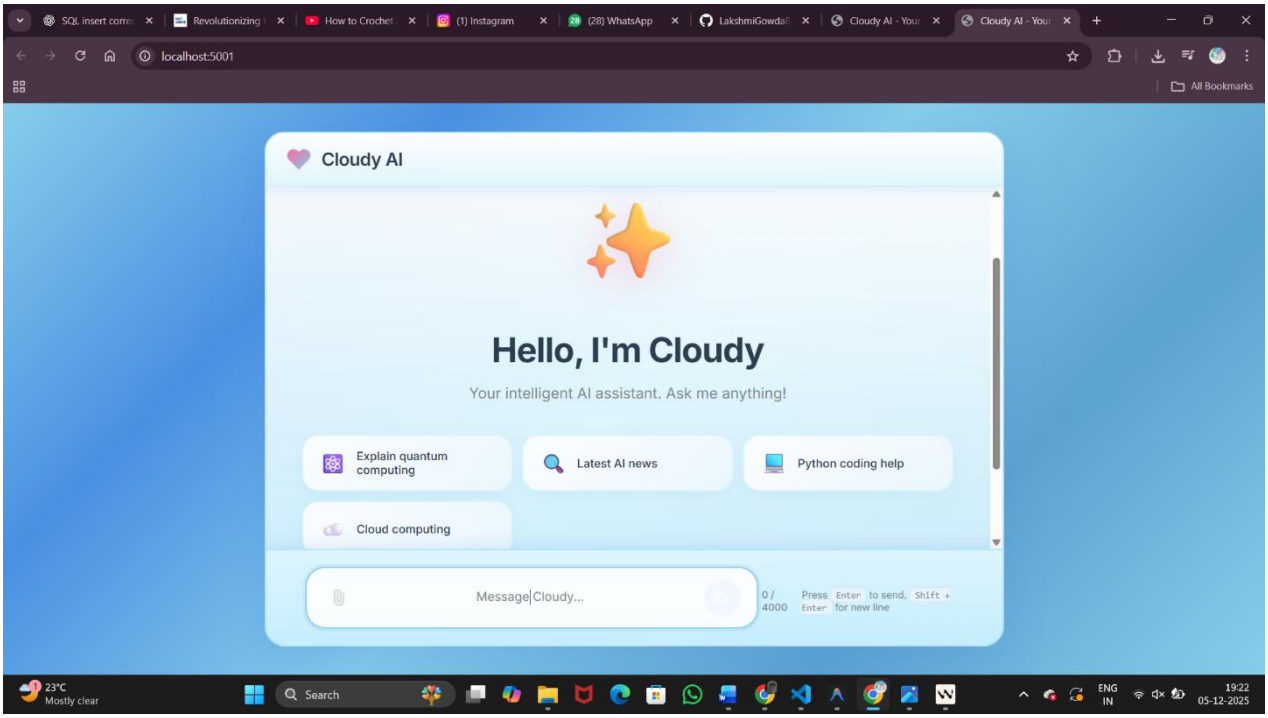
Module 4: Storage

- Store conversations in Dynamic storage Manner

4. RESULTS AND ANALYSIS

4.1 Output Screenshots

FIG 1: Cloudy AI Interface – Initial dashboard with featured actions and chat window.



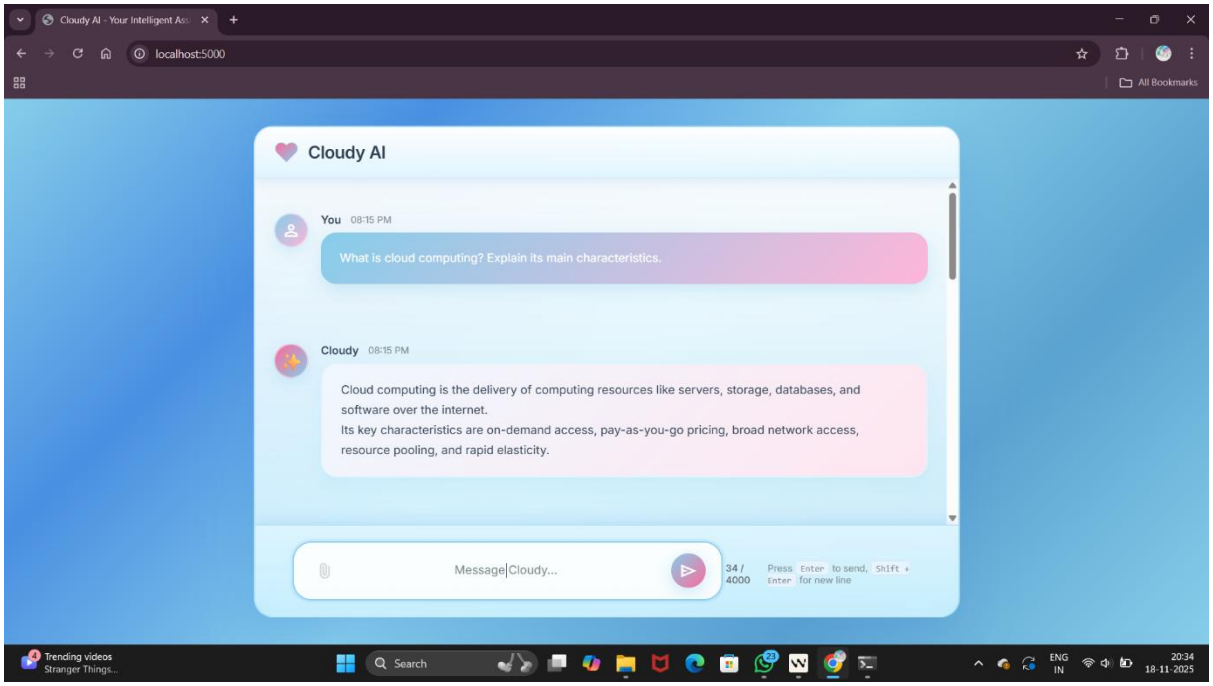


FIG 2: Chat Interface View – Showing user query and AI-generated response inside the Cloudy AI conversation window.

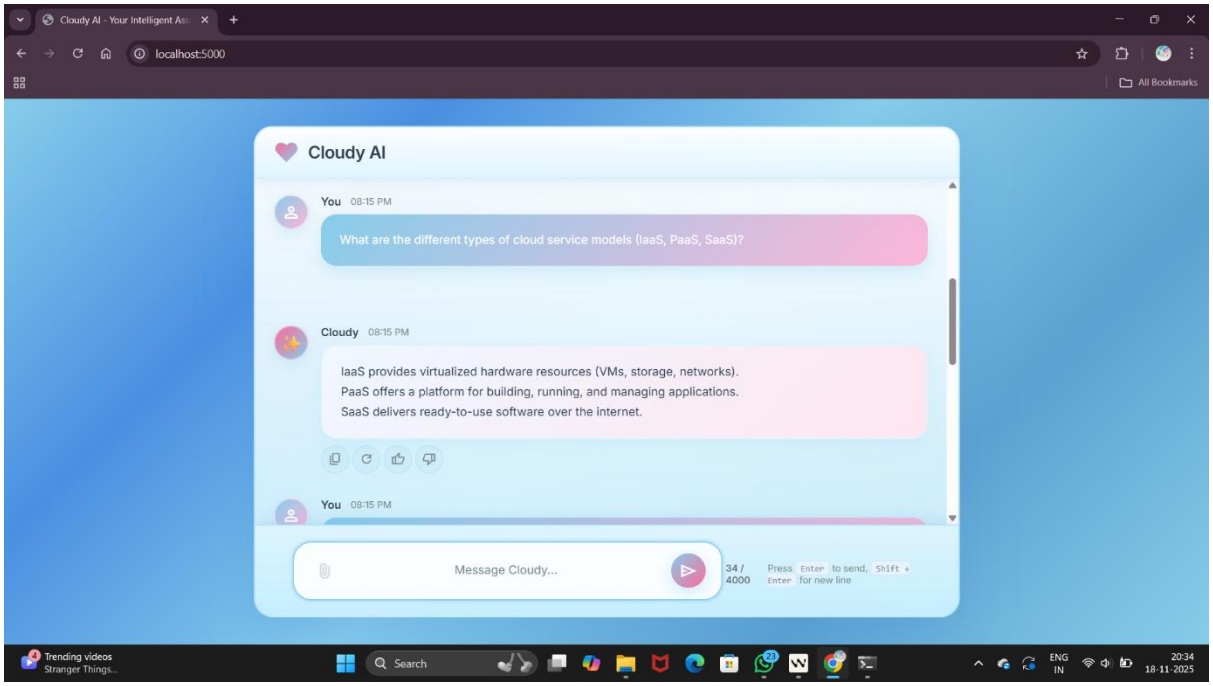


FIG 3: Chat Response View – Displaying Cloudy AI's explanation of cloud service models (IaaS, PaaS, SaaS) in the conversation interface.

4.2 Result Explanation

The cloud-enabled chatbot efficiently processes natural language queries and generates clear, context-aware responses in real time. It uses dynamic, in-memory session handling instead of a database, allowing the system to remain fast, lightweight, and privacy-friendly. Since no data is permanently stored, each session resets automatically after use, simplifying the overall architecture. Cloud deployment ensures high uptime, low latency, and the ability to handle multiple users simultaneously. The NLP engine accurately interprets user intent, and the frontend interacts smoothly with the backend through REST APIs or WebSockets. This dynamic approach avoids delays from database operations, improving response speed. Error handling keeps the system stable and reliable throughout interactions. Overall, the results show that the chatbot achieves strong performance, quick processing, cloud scalability, and responsive conversation flow without needing any persistent storage.

4.3 Advantages

1. **Faster Response Time:**
   Without database read/write operations, the chatbot processes queries much faster.
2. **Lightweight Architecture:**
   Dynamic in-memory storage keeps the system simple and easy to maintain.
3. **High Privacy:**
   Since no data is permanently stored, user conversations are not saved, increasing privacy.
4. **Reduced System Overhead:**
   No need for a database server reduces resource usage and costs.
5. **Scalable Cloud Performance:**
   Cloud deployment allows handling many users at once with minimal latency.
6. **Easy to Deploy and Manage:**
   The absence of a database layer simplifies setup, updates, and scaling.

4.4 Limitations

1. **No Conversation History:**
   Since data is not stored, users cannot view previous chats or continue old conversations.
2. **Limited Personalization:**
   The chatbot cannot remember user preferences or past interactions across sessions.
3. **Short-Lived Context:**
   Dynamic in-memory storage resets after each session, causing loss of long-term context.
4. **No Analytics or Insights:**

Without stored chat logs, the system cannot generate reports or analyze user behavior.

5. **Not Suitable for Large Applications:**

   Big platforms require persistent storage, which this system does not support.

6. **Session Loss on Refresh:**

   If the page refreshes or connection drops, all chat context is immediately lost.

4.5 Future Enhancements

1. **Add Database Integration:**

   Introduce persistent storage to save chat history, user profiles, and analytics data.

2. **Implement User Authentication:**

   Allow users to log in and continue previous conversations across devices.

3. **Advanced NLP Models:**

   Integrate more powerful transformer models for deeper context understanding.

4. **Voice Input and Speech Output:**

   Add speech-to-text and text-to-speech for a voice-enabled chatbot experience.

5. **Multilingual Support:**

   Enable the chatbot to understand and respond in multiple languages.

6. **Sentiment Analysis:**

   Detect user emotions and adjust responses accordingly.

5. Repository URL

https://github.com/LakshmiGowda831/Cloud-Chatbot

6. CONCLUSION

The Cloud-Enabled Chatbot project successfully demonstrates how modern cloud technologies and NLP models can be combined to create an efficient, intelligent, and scalable conversational system. By using dynamic in-memory storage instead of a database, the chatbot maintains fast processing, simplified architecture, and enhanced privacy for users. Cloud deployment ensures high availability, low latency, and the ability to handle multiple users simultaneously, while the NLP engine provides accurate, context-aware responses. The system meets its objectives of delivering real-time interactions, secure API communication, and responsive user experience through a streamlined frontend and backend structure. Overall, the project showcases the practical application of cloud computing and AI in building lightweight yet powerful chatbot solutions, and it provides a strong foundation for future enhancements such as database support, multilingual processing, voice interaction, and advanced analytics.