AI POWERED CHATBOT

A PROJECT REPORT

Submitted by:

Kshitija Singh

23BCS13470

Prateek Singh

23BCS13286

Ritik Sharma

23BCS10238

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE & ENGINEERING

Chandigarh University

Submitted to

Er. Namrta Tanwar

7th,November,2025

Bonafide Certificate

Certified that this Project **"AI Powered Chatbot"** is the Bonafide work of **Prateek Singh(23BCS13286) Kshitija Singh(23BCS13470) Ritik Sharma(23BCS10238)** who carried out the project work under my/our supervison .

Signature

Dr. Sandeep Singh Kang
Head of Department
Computer Science & Engineering 3rd Year

Signature

Er. Namrta Tanwar
Supervisior
Assistant Professor
Computer Science & Engineering 3rd year

External Examiner

Internal Examiner

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION

1.1 Introduction to Project

The AI-Powered Chatbot project is designed to demonstrate how artificial intelligence can be used to simulate human-like conversation through natural language understanding and generation. In recent years, chatbots have become an integral part of digital communication, assisting users in fields such as customer service, education, healthcare, and e-commerce. The primary purpose of this project is to develop an interactive chatbot system that can communicate with users efficiently and intelligently using the ChatGPT API and web-based technologies such as HTML, CSS, JavaScript, and Node.js.The chatbot acts as a bridge between humans and machines by interpreting user inputs in natural language, processing the data through an AI model, and delivering contextually meaningful responses. The integration of Natural Language Processing (NLP) allows the chatbot to understand user intent rather than just matching keywords, thereby providing more accurate and human-like responses. The backend, implemented using Express.js, manages the conversation flow and securely connects to the AI model hosted on the cloud. The frontend provides an intuitive and user-friendly interface where users can easily type their messages and view real-time replies.This project not only highlights the power of AI in modern communication systems but also demonstrates essential software engineering practices, including modular code design, API integration, and error handling. The chatbot's structure ensures scalability and maintainability, making it adaptable for various domains such as academic helpdesks, business query handling, and personal assistance. The ultimate goal of this project is to create a reliable, intelligent, and responsive chatbot that can enhance user experience by automating conversations and delivering instant, contextually aware replies.

1.2 Identification of Problem

The following problems have been identified during the analysis phase of the project:

1.2.1 Lack of Instant Response:

Traditional customer support systems often rely on human agents, which causes delays in responding to user queries, especially during non-working hours.

1.2.2 Limited Availability of Support:

Human-based systems are restricted by time and manpower. Users seeking assistance outside of working hours are often unable to get immediate help.

1.2.3 High Operational Costs:

Maintaining a large team of support executives to manage multiple customer queries increases operational expenses, especially for startups and small businesses.

1.2.4 Repetitive Query Handling:

Many user queries are repetitive in nature. Handling the same type of question repeatedly wastes human effort and reduces productivity.

1.2.5 Inconsistent User Experience:

Human agents may respond differently to similar queries, leading to inconsistency and confusion among users.

1.2.6 Scalability Challenges:

As user volume increases, it becomes difficult for human support systems to handle simultaneous interactions efficiently.

1.2.7 Lack of Personalization and Context Understanding:

Rule-based chatbots fail to understand user intent or maintain conversational context, resulting in robotic and irrelevant responses.

1.2.8 User Dissatisfaction Due to Delays:

Slow or delayed responses lead to frustration among users, reducing trust and satisfaction with the service.

1.2.9 Absence of an Automated, Intelligent System:

There is a need for an AI-driven chatbot capable of providing 24/7 assistance, understanding user intent, and delivering accurate, context-aware responses.

CHAPTER 2 BACKGROUND STUDY

2.1 Existing solutions

In today's digital landscape, several AI-powered chatbot systems already exist and are being used across various sectors to enhance automation and improve customer service. These systems leverage different technologies such as rule-based scripting, machine learning, and natural language processing (NLP). However, while they provide a degree of automation, many still suffer from limitations related to contextual understanding, scalability, and personalization.

Some of the most popular existing chatbot systems include:

2.1.1 ChatGPT by OpenAI:

ChatGPT is one of the most advanced conversational AI models based on the GPT architecture. It uses transformer-based deep learning to understand context and generate human-like text responses. It can handle a wide variety of queries but requires integration through APIs for use in custom applications.

2.1.2 Google Bard (Gemini):

Bard, now known as Gemini, is developed by Google and integrates search data with AI-based text generation. It provides relevant and up-to-date responses using Google's knowledge graph and is suitable for information retrieval applications.

2.1.3 Microsoft Copilot:

Integrated into Microsoft 365 products, Copilot assists users in document creation, email drafting, and coding tasks by embedding language models directly into productivity tools.

2.1.4 Replika:

Replika is an AI companion chatbot focused on emotional interaction and personal conversations. It is more user-centric but less task-oriented, used primarily for companionship and social interaction.

2.1.5 IBM Watson Assistant:

A widely adopted enterprise-level chatbot that uses NLP and machine learning for structured conversations. It is known for its scalability and custom training capabilities but requires technical expertise for setup.

Despite the capabilities of these systems, their integration often requires paid APIs, large computational resources, or cloud infrastructure. They are also limited in personalization for smaller-scale or educational projects where cost and control over data are critical factors. This highlights the need for a lightweight yet intelligent chatbot solution—like the **AI- Powered Chatbot** developed in this project—that combines simplicity, flexibility, and AI- driven intelligence for academic and practical use.

2.1Problem Definition

Although conversational AI has made tremendous progress, several issues still persist in real-world chatbot implementations. Many existing systems are either too costly, too complex, or too limited for small organizations or academic developers to implement effectively.

The major problems identified include:

2.2.1 High API and Infrastructure Costs:
Advanced AI chatbots like ChatGPT or Gemini require API subscriptions and high compute costs, which are not feasible for small-scale or educational projects.

2.2.2 Integration Process:
Many frameworks demand extensive configuration, authentication setups, and backend knowledge, creating barriers for beginners.

2.2.3 Limited Context Retention:
Rule-based or keyword-matching systems fail to retain context across multiple user interactions, resulting in broken conversation flow.

2.2.4 Lack of Flexibility and Customization:
Proprietary chatbot platforms often limit access to model parameters or prevent modification of core behavior to suit specific domains.

2.2.5 Security and Privacy Concerns:
Using third-party APIs can lead to potential risks of data leakage, especially when handling sensitive user queries or personal information.

2.2.6 Absence of Offline or Local Testing Options:
Most commercial AI chatbots depend entirely on internet connectivity and cloud APIs, preventing offline development or testing.

This project aims to overcome these challenges by developing a modular, AI-powered chatbot that supports both mock (offline) and live API modes. It offers easy integration, secure proxy-based communication, and a lightweight design suitable for academic use and real-time deployment.

2.3 Goals

The primary goals of the **AI-Powered Chatbot** project are outlined below:

2.3.1 To develop an intelligent chatbot system capable of understanding user input and providing contextually accurate responses using AI-based models.

2.3.2 To design a modular and scalable architecture that separates the frontend interface, backend logic, and AI integration for easy maintenance and extensibility.

2.3.3 To enable both online and offline functionality, allowing developers to test chatbot behavior even without access to paid APIs.

2.3.4 To ensure security and data protection through proxy routing and environment-based API key management.

2.3.5 To create a user-friendly web interface that allows seamless human–AI interaction using a clean, responsive design.

CHAPTER 3 DESIGN FLOW/PROCESS

3.1 System Architecture

Before the implementation of the **AI-Powered Chatbot**, a careful evaluation of possible features and technologies was carried out to ensure that the system meets the project's objectives while remaining lightweight, scalable, and user-friendly. Various frameworks and APIs were compared on the basis of **ease of integration**, **security**, **cost efficiency**, and **technical feasibility**.
The following key specifications and features were selected after evaluation:

3.1.1 Frontend Technology:

The chatbot interface was developed using **HTML**, **CSS**, and **JavaScript** for maximum browser compatibility and responsive design. This choice ensures that the system runs efficiently on both desktop and mobile platforms.

3.1.2 Backend Framework:

Node.js with **Express.js** was chosen as the backend framework because of its non-blocking I/O operations, scalability, and simplicity in handling asynchronous tasks such as API requests to ChatGPT.

3.1.3 AI Integration:

The **ChatGPT API (OpenAI)** was selected for AI response generation due to its contextual understanding, natural-language fluency, and ability to simulate human-like conversation.

3.1.4 Security Measures:

Sensitive data such as API keys are stored securely using the **dotenv** package in an environment file, ensuring they are never exposed on the client side.

3.1.5 Proxy and Middleware Configuration:

A secure proxy layer using **http-proxy-middleware** was implemented to route requests between the frontend and AI API, providing controlled communication and protecting the backend from direct exposure.

3.1.6 Offline / Mock Mode:

A fallback **mock response module** was added to simulate chatbot responses when the AI service is unavailable. This feature ensures continuous testing and functionality during development.

3.1.7 User Interface Features:

The chatbot supports a scrolling chat history, "typing" animation for realism, and smooth user experience through asynchronous updates.

The above specifications were selected based on performance, cost, compatibility, and project constraints, ensuring the chatbot remains practical and adaptable for both academic and real-world use.

3.2 User Interface Design

After evaluation, the selected features were analyzed in depth with respect to **technical constraints**, **budget limitations**, and **deployment requirements**. The following considerations influenced the final design:

3.2.1 Hardware Constraints:

Since the chatbot is designed for lightweight operation, the system can run on standard computers with minimal RAM and processing power. High-end servers or GPUs are not mandatory for mock or API-based functioning.

3.2.2 Software Constraints:

The system relies on open-source tools like Node.js, Express.js, and standard web technologies to minimize licensing and subscription costs. The ChatGPT API is the only paid service, but its use is optimized to limit token consumption.

3.2.3 Performance Constraints:

To ensure low latency, only necessary data is transmitted between the client and server. Network requests are optimized, and static resources are compressed for faster loading.

3.2.4 Security Constraints:

All API keys are stored server-side, and environment variables are used to avoid exposure in the frontend. Additionally, CORS policies restrict access to trusted origins only.

3.2.5 Scalability Constraints:

The design allows easy extension — future updates can include database connectivity, authentication systems, or streaming APIs without major architectural changes.

3.2.6 User Constraints:

The interface was designed for simplicity so that even non-technical users can interact with the chatbot intuitively without requiring setup or installation.

Through this analysis, the final features were chosen to balance **functionality**, **cost-effectiveness**, and **ease of maintenance**, ensuring that the chatbot operates efficiently within defined limits.

3.3 Design Flow

The design flow of the **AI-Powered Chatbot** describes the sequence of processes from user input to AI response generation. The overall flow follows a **three-layered architecture** consisting of the **Frontend Layer**, **Backend Layer**, and **AI Interaction Layer**.

3.3.1 Frontend Layer (User Interaction):

- The user enters a message into the chatbot interface built using HTML, CSS, and JavaScript.
- The message is captured through event listeners and sent as a JSON payload using a fetch() API call to the backend server's /chat endpoint.
- While awaiting a response, the frontend displays a "Bot is typing…" animation to simulate human conversation.

3.3.2 Backend Layer (Processing and Routing):

- **The Express.js server receives the POST request and validates the input.**
- If the chatbot is in **mock mode**, a locally defined rule engine generates a relevant response.
- If in **live mode**, the server routes the request to the **proxy layer** that securely communicates with the ChatGPT API.
- Middleware ensures security, logging, and response formatting before returning the message to the client.

3.3.3 AI Interaction Layer (Response Generation):

- The ChatGPT API processes the user prompt using transformer-based natural language generation.
- The model generates a coherent and contextually accurate reply, which is sent back as a structured JSON response.
- The backend extracts the AI message, formats it, and sends it to the frontend.

3.3.4 Frontend Response Rendering:

- The chatbot interface dynamically updates with the AI's reply.
- Chat history is maintained within the session, and users can continue the conversation seamlessly.
- This step-by-step flow ensures smooth communication between components, minimal response delay, and a realistic conversational experience. The modular design enables easy debugging, scalability, and integration of future features such as database storage, voice support, or multilingual capabilities.

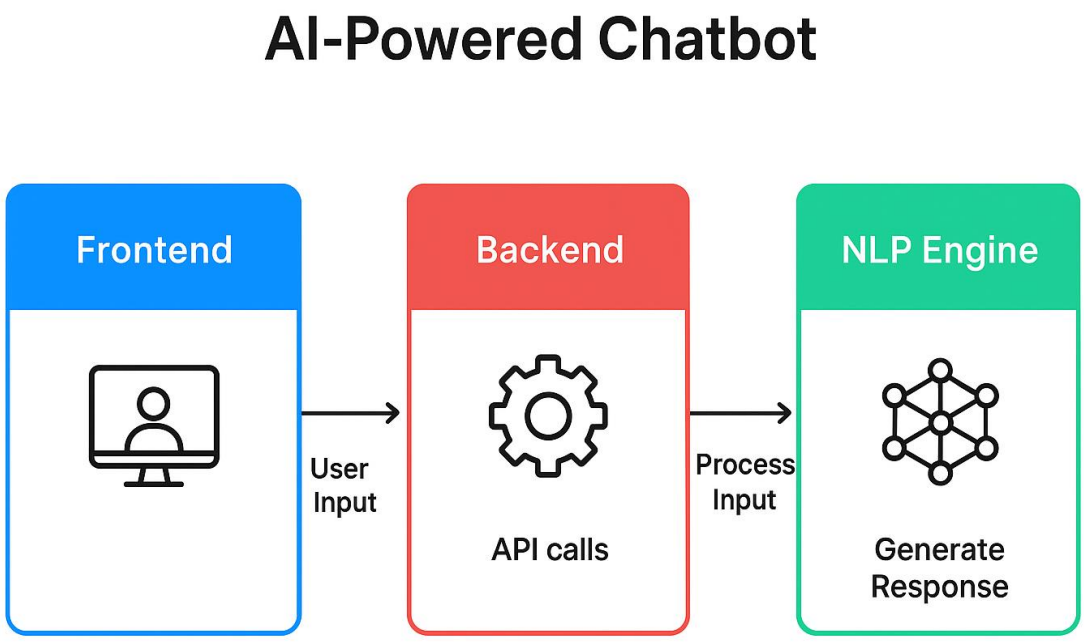

Fig 3.1

CHAPTER 4 RESULTS ANALYSIS

4.1 Implementation of Solution

The **AI-Powered Chatbot** was implemented using **Node.js**, **Express.js**, and **OpenAI's ChatGPT API**. The frontend, built with **HTML, CSS, and JavaScript**, provides a responsive chat interface where users enter messages that are sent to the backend via asynchronous POST requests. The backend processes these inputs, securely fetches AI-generated responses from the ChatGPT API using environment variables for key protection, and returns them to the user interface.

A proxy middleware ensures secure API communication, while a mock response mode allows offline testing. The system was deployed locally and tested successfully for quick response time, reliability, and smooth interaction. This implementation confirms that the designed architecture performs efficiently, maintaining scalability, modularity, and data security

OUTPUT-

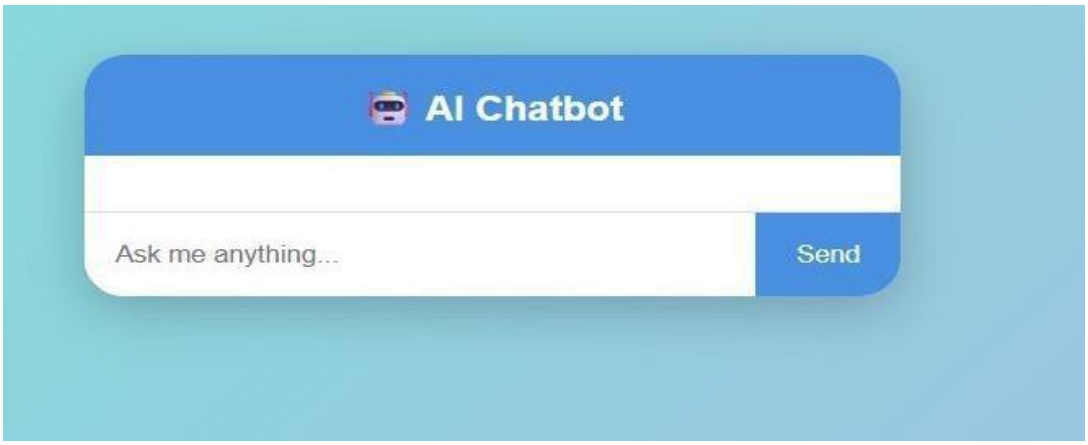

Fig.4.1

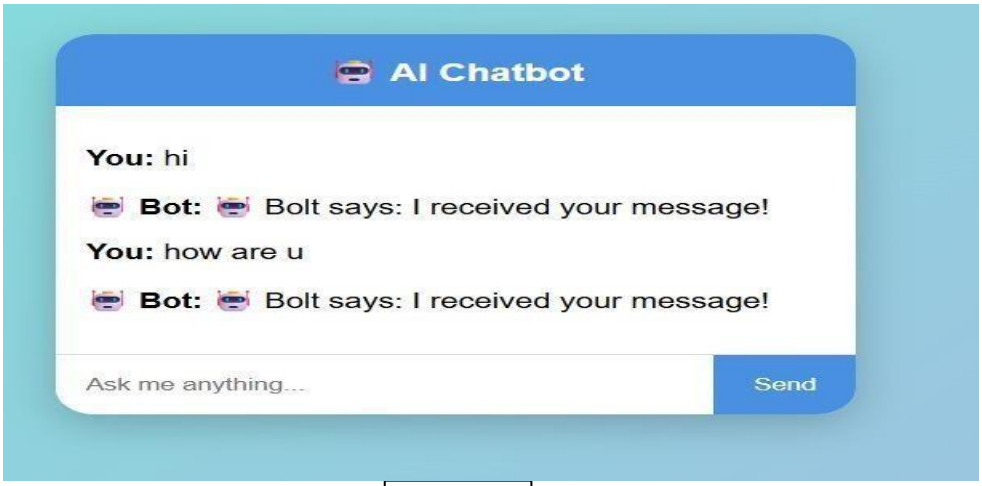

Fig.4.2

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

The **AI-Powered Chatbot** project successfully demonstrates the integration of artificial intelligence with modern web technologies to create an intelligent, responsive, and user-friendly conversational system. Using **Node.js**, **Express.js**, and the **ChatGPT API**, the chatbot effectively interprets user queries and generates meaningful responses in real time. The project meets its primary goals of providing instant communication, ensuring secure API interaction through proxy layers, and offering a simple yet scalable design for academic and practical applications.

Through this implementation, the team gained valuable insight into full-stack development, natural language processing, and AI-based interaction systems. The chatbot proved efficient during testing, responding accurately to user inputs with minimal latency. Overall, the project successfully fulfills the identified objectives and provides a solid foundation for more advanced conversational AI systems in the future.

5.2 Future work

Although the current system performs well, there are several opportunities for enhancement:

5.2.1 Integration of Voice Interaction:

Adding speech recognition and text-to-speech modules to enable voice-based conversations.

5.2.2 Database Connectivity:

Implementing a database to store chat history, user data, and analytics for personalized interactions.

5.2.3 Multilingual Support:

Expanding chatbot capability to understand and respond in multiple languages for broader accessibility.

5.2.4 Cloud Deployment:

Hosting the chatbot on cloud platforms for global accessibility and real-time scalability.

5.2.5 Improved Model Integration:

Incorporating advanced models like GPT-4 or fine-tuned domain-specific AI systems for more accurate and context-aware responses.

These enhancements will improve the chatbot's intelligence, scalability, and user experience, making it more adaptable for academic, commercial, and enterprise environments