AI-Powered Conversational Chat Platform

A Beginner-Friendly Project Report

Author: O Nithin Sai Balaji
LinkedIn: https://www.linkedin.com/in/o-ns23/
GitHub: https://github.com/O-NS23
Project Repository:
AI-Powered-Conversational-Chat-Platform-Built-with-Streamlit-and-Gemini
Date: December 2025

Executive Summary

This project report describes the development of an AI-powered conversational chatbot built using Google's Gemini API and Streamlit. The application demonstrates how large language models (LLMs) can be integrated into a simple web interface to provide real-time, user-friendly conversational experiences.

As a Computer Science fresher aiming for a career in Data Science and AI/ML, this project highlights:

- Understanding of modern AI and LLM technologies
- Ability to integrate APIs and build end-to-end applications
- Practical problem solving using cloud-based AI services
- Clean coding practices and basic deployment experience

The chatbot accepts user questions, generates intelligent responses using the Gemini model, and maintains chat history within the session, all inside a minimal and intuitive web interface.

1. Project Overview

1.1 What is This Project?

This project is an interactive AI chatbot application that allows users to have real-time conversations with an AI model. Instead of static responses, the chatbot generates contextual and natural language replies based on the user's input.

Key characteristics:

- **Conversational:** Users can ask questions in plain English.
- **Real-time:** Responses are generated within a few seconds.
- **Session-based memory:** The chat history is preserved during the session.
- **Beginner-friendly UI:** The interface is simple and easy to use.
- **Cloud-powered:** Uses Google's Gemini model via an API.

1.2 Motivation and Problem Statement

AI and machine learning are in high demand, and many students learn theory or build isolated models without deploying them as real applications. This creates a gap between academic projects and industry expectations.

The main problem addressed is: *How can a fresher demonstrate practical AI skills by building a simple, deployable AI application that non-technical users can interact with?*

1.3 Objectives

The project was designed with the following learning objectives:

- Integrate a pre-trained AI model (Gemini) into a Python application.
- Build a basic but clean web interface using Streamlit.
- Implement session state to maintain chat history.
- Handle user input validation and simple error cases.
- Deploy and share the application through the cloud.

2. Technical Architecture

2.1 System Architecture

The chatbot follows a simple three-layer architecture:

Frontend (Presentation Layer)

- Technology: Streamlit
- Responsibilities: Display the UI, accept user input, show chat history
- Components: Text input box, buttons, chat display area

Application Logic (Business Layer)

- Technology: Python
- Responsibilities: Input validation, API calls, chat state management
- Components: Functions for generating responses and managing session state

AI Backend (Service Layer)

- Technology: Google Gemini API
- Responsibilities: Natural language understanding and response generation
- Components: Model configuration and API communication

2.2 Technology Stack

Component	Technology	Purpose
Frontend Framework	Streamlit	Build an interactive web interface without HTML/CSS/JS.

Programming Language	Python 3.x	Implement application logic and API integration.
AI Model	Google Gemini 2.5 Flash	Generate natural language responses.
API SDK	`google-generativeai`	Python client to access Gemini API.
State Management	Streamlit Session State	Maintain chat history during a session.
Environment / Deployment	Google Colab + Tunneling (Cloudflare/ngrok)	Run and share the app over the internet.
Version Control	Git and GitHub	Track changes and host project code.

2.3 Chatbot Workflow

The overall workflow can be described in six main steps:

Step 1: User Input

- The user types a question in the text input field.
- The user clicks the "Get Answer" button.

Step 2: Input Validation

- The application checks if the input is empty.
- If empty, a warning message is shown: "Please type something first".
- If valid, processing continues.

Step 3: API Call

- The question is sent to the Gemini model using the API.
- The model processes the text and generates a response.

Step 4: Response Handling

- The response text is extracted from the API output.
- Both the question and the response are appended to the chat history in session state.

Step 5: Display Chat History

- The conversation is displayed in order.
- User messages and AI messages are visually separated.

Step 6: Reset Option

- The user can click "End Here" to clear the chat.
- This resets the session chat history and starts a new conversation.

3. Features and Functionality

3.1 Core Features

The application includes the following core features:

- Real-time AI responses using the Gemini model.
- Automatic chat history storage in session state.
- Input validation to prevent empty queries.
- Simple and clean user interface built with Streamlit.
- Basic error handling for smoother user experience.
- Reset button to clear conversation and start fresh.

3.2 User Interface Elements

Text Input Field

- Accepts any text-based question from the user.
- Acts as the main interaction point.

Buttons

- "Get Answer": Sends the input to the AI model.
- "End Here": Clears the chat history and resets the session.

Chat Display

- Shows the full conversation between the user and the AI.
- Differentiates between user messages and AI replies for readability.

3.3 User Journey

A typical user journey:

1. Open the chatbot web page.
2. Type a question such as "How to post projects on LinkedIn?".
3. Click "Get Answer" and wait for the AI response.
4. Read the detailed answer in the chat area.
5. Ask follow-up questions if needed.
6. Click "End Here" to clear and begin a new conversation.

4. Code Structure and Implementation

4.1 Configuration and Setup

The main components are implemented in a single Streamlit application file:

```
import streamlit as st
```

```
import google.generativeai as genai

# Configure Gemini API
genai.configure(api_key="YOUR_GOOGLE_API_KEY")
model = genai.GenerativeModel("gemini-2.5-flash")

# Streamlit page configuration
st.set_page_config(page_title="Smart Chat ")
st.title("AI Chat App ")
```

This part:

- Imports Streamlit and the Gemini SDK.
- Configures the API key and selects the model.
- Sets the page title and header for the UI.

4.2 Session State for Chat History

```
if "chat" not in st.session_state:
    st.session_state.chat = []
```

This ensures:

- Chat history is stored in $\texttt{st.session}_{s}tate$.
- `The list is initialized only once per session.`

4.3 Handling User Input and API Calls

```
question = st.text_input("Ask your question ")

if st.button("Get Answer"):
    if question.strip() == "":
        st.warning("Please type something first ")
    else:
        with st.spinner("Thinking... "):
            reply = model.generate_content(question).text
            st.session_state.chat.append(("You", question))
            st.session_state.chat.append(("AI", reply))
```

Responsibilities:

- Collect user input using `text_input`.
- Validate that the input is not empty.
- Call the Gemini model and get the reply.
- Append both question and response to the chat history.

4.4 Displaying Chat History and Resetting

```
st.write("### Chat History")
for sender, msg in st.session_state.chat:
    if sender == "You":
        st.markdown(f" **You:** {msg}")
    else:
        st.markdown(f" **AI:** {msg}")

if st.button("End Here"):
    st.session_state.chat = []
    st.success("Chat Ended ")
```

This code:

- Iterates over the stored conversation.
- Displays user and AI messages with different prefixes.
- Provides a reset button to clear the chat.

5. Project Development and Results

5.1 Development Phases

The project was built in four simple phases:

Phase 1: Planning

- Studied Gemini API documentation.
- Chose Streamlit for the user interface.
- Defined scope: simple one-page chatbot with history.

Phase 2: Implementation

- Implemented API integration in a Python environment.
- Built the Streamlit interface.
- Connected UI elements to the backend logic.

Phase 3: Testing

- Tested different types of user queries.
- Checked input validation and error handling.
- Verified that chat history worked as expected.

Phase 4: Deployment and Documentation

- Ran the app in Google Colab and exposed it via tunneling.
- Pushed the code to GitHub.
- Prepared a beginner-friendly project report.

5.2 Example Output

One sample interaction used to verify correctness:

User Question: "How to post projects on LinkedIn?"

AI Response: A detailed, step-by-step explanation covering:

- Adding projects in the LinkedIn "Projects" section.
- Sharing projects via posts with links and screenshots.
- Writing an article to explain the project in depth.

This shows that the chatbot can return helpful, structured answers suitable for real users.

5.3 Basic Performance Metrics

Metric	**Result (Observed)**
Average response time	Around 2–4 seconds per query
Model used	Gemini 2.5 Flash
Input validation	Prevents empty queries from being sent
Chat history	Correctly maintains order of questions and answers
UI responsiveness	Interface remains responsive during calls

6. Key Learnings

6.1 Technical Skills Gained

API Integration

- Learned how to configure and call a cloud-based AI API.
- Understood how to parse responses and handle errors.

Streamlit and State Management

- Used Streamlit to build a functional UI with minimal code.
- Applied session state to preserve data across interactions.

Python Best Practices

- Used clear variable names and simple structure.
- Handled edge cases such as empty inputs.

Deployment Basics

- Ran the app in a cloud notebook environment.
- Learned how to expose local apps using tunneling tools.

6.2 Professional Growth

Beyond coding, this project helped in:

- Practicing end-to-end development from idea to deployment.
- Writing clear documentation for recruiters and beginners.
- Creating a portfolio-ready AI project demonstrating practical skills.

7. How to Run the Project

7.1 Prerequisites

- Python 3.8 or higher installed.
- A Google account with access to the Gemini API.
- Basic familiarity with running Python scripts or Streamlit apps.

7.2 Setup Steps

Step 1: Clone the Repository

```
git clone https://github.com/O-NS23/AI-Powered-Conversational-Chat-Platform-Built-with-Strea
cd AI-Powered-Conversational-Chat-Platform-Built-with-Streamlit-and-Gemini
```

Step 2: Create and Activate a Virtual Environment

```
python -m venv venv
# Windows:
venv\Scripts\activate
# macOS / Linux:
source venv/bin/activate
```

Step 3: Install Dependencies

```
pip install -r requirements.txt
```

Step 4: Configure the Gemini API Key

- Generate an API key from Google AI Studio.
- Set it in the code or as an environment variable.

Step 5: Run the Application

```
streamlit run app.py
```

Then open `http://localhost:8501` in your browser to start chatting with the AI.

8. Skills Demonstrated and Conclusion

8.1 Skills Demonstrated

This project demonstrates:

- Python programming for web applications.
- Integration of AI/LLM models into real applications.
- Basic full-stack development using Streamlit.
- State management and simple UX design.
- Use of GitHub for version control and portfolio building.

8.2 Conclusion

The AI-powered conversational chat platform shows how a fresher can build a complete, working AI application by combining an LLM API with a lightweight web framework. It moves beyond theory and simple notebooks into a user-facing tool that anyone can open and use.

For recruiters, this project demonstrates practical skills in AI integration, web development, and clear documentation. For beginners, it serves as an example of how to start building simple but meaningful AI projects that can be deployed and shared.

AI Chat App 🤓

Ask your question 👇

'How to post the projects in linkedin?'

Get Answer | End Here

Chat History

👱 **You:** 'How to post the projects in linkedin?'

🤓 **AI:** Posting projects on LinkedIn is an excellent way to showcase your skills, experience, and the tangible results of your work. There are several ways to do this, ranging from a dedicated "Projects" section to general posts and articles.

Here's a comprehensive guide:

---

1. Using the Dedicated "Projects" Section (Most Recommended)

This is the official and most structured way to highlight your projects on your LinkedIn profile.

Steps:

1. **Go to your LinkedIn Profile:** Click on your profile picture, then "View Profile."
2. **Add a New Section:**
   - Scroll down to where you see sections like "Experience," "Education," "Skills," etc.
   - Look for the **"Add profile section"** button (or the `+` icon next to "Add section").
   - Click on **"Recommended"** > **"Add projects."** (If you already have a Projects section, you'll see a `+` icon within it to add a new project).
3. **Fill in the Project Details:**
   - **Project name:** Give your project a clear, concise, and descriptive title.
   - **Associated with:** This is crucial. Link the project to a specific **Experience** (e.g., a job role where you did this project) or **Education** entry (e.g., a university course). This provides context.
   - **Start date / End date:** Specify the timeline.
   - **Project URL (Highly Recommended!):** This is where you link to the actual project! This could be:
     - A live website/demo

        - A GitHub repository
        - A Behance/Dribbble portfolio link
        - A Google Drive link to a report/presentation
        - A YouTube video explaining the project
    - **Description:** This is where you tell the story of your project. Focus on:
        - **Problem:** What challenge were you trying to solve?
        - **Task/Role:** What was your specific role and responsibilities?
        - **Actions:** What steps did you take? What technologies/tools did you use?
        - **Results/Impact:** What was the outcome? **Quantify it whenever possible!** (e.g., "Increased conversion by 15%," "Reduced processing time by 20 hours/month").
        - Use the STAR method (Situation, Task, Action, Result) for a powerful description.
    - **Skills:** Add relevant skills that you utilized during the project. These act as keywords for recruiters.
    - **Team members:** If you collaborated with others, you can search for and add their LinkedIn profiles. They'll receive a notification and can accept the association.
    - **Media:** You can upload images, videos, documents, or presentations directly to showcase your work visually. Screenshots of interfaces, diagrams, or project wireframes are very effective.
4. **Save:** Once you're done, click "Save."

---

2. Posting Your Projects as Updates/Posts

This is great for sharing new projects, milestones, or getting immediate visibility.

1. **Start a post:** From your LinkedIn homepage, click "Start a post."
2. **Write a compelling caption:**
    - Announce your project clearly.
    - Briefly explain what it is and its purpose/impact.
    - Ask a question to encourage engagement (e.g., "What do you think?").
    - Include a strong call to action (e.g., "Check out the live demo here!").
3. **Include a link:** Paste the project URL (GitHub, live site, portfolio, etc.) into the post. LinkedIn will usually generate a rich preview.
4. **Add visual media: This is critical!** Upload screenshots, a short video walkthrough, a GIF, or a relevant image. Posts with visuals get significantly more engagement.
5. **Use relevant hashtags:** E.g., #webdevelopment #datascience #uxdesign #projectshowcase #portfolio.
6. **Tag relevant people/companies:** If applicable.
7. **Post:** Click "Post."
8. **Engage:** Respond to comments and questions.

---

3. Writing a LinkedIn Article About Your Project

For more in-depth explanations, case studies, or technical deep dives.

1. **Go to your LinkedIn homepage:** Click "Write article" (usually next to "Start a post").
2. **Create a detailed article:**
    - **Title:** Catchy and descriptive.
    - **Cover image:** Professional and relevant.
    - **Body:**
        - Explain the problem, your approach, challenges faced, solutions implemented, and the final results.
        - Break it down with headings, bullet points, and images/videos.
        - Discuss the technologies used and lessons learned.
        - This is an opportunity to demonstrate your thought process and problem-solving skills.
    - **Call to action:** Link to the live project, your portfolio, or encourage discussion in the comments.
3. **Publish:** Click "Publish." Your article will appear on your profile and can be shared like a post.

---

4. Attaching Media to Other Sections

You can add project visuals or links to your profile's **Summary** or within **Experience** entries.

- **Summary:** Under your profile picture, click the pencil icon to edit your summary. You can add media (documents, photos, sites, videos) that link to your portfolio or specific project highlights.
- **Experience:** Edit an existing job role. You'll see an option to "Add media" where you can link to projects you completed during that role.

Tips for Making Your Project Stand Out:

- **Quantify your achievements:** Numbers speak louder than words ("Increased user engagement by 20%," "Processed 500GB of data").
- **Use the STAR Method:** For descriptions, outline the **S**ituation, **T**ask, **A**ction, and **R**esult.
- **Visuals are paramount:** Screenshots, diagrams, mockups, short video demos. Show, don't just tell.
- **Provide direct links:** Always link to the live project, GitHub repo, or detailed case study.
- **Highlight your role:** Clearly define what *you* did, especially in team projects.

- **Showcase relevant skills:** Use keywords that recruiters search for.
- **Proofread carefully:** Errors can detract from your professionalism.
- **Engage with your audience:** Respond to comments and questions on your posts/articles.
- **Tailor to your goals:** If you're looking for a specific type of role, emphasize projects that align with those requirements.

By effectively utilizing these methods, you can transform your LinkedIn profile into a powerful portfolio that clearly demonstrates your capabilities to potential employers and collaborators.