

# Beer's AI Chatbot Documentation (Enhanced Version)

## Overview

Beer's AI Chatbot is a web-based conversational interface built with **Streamlit**, integrating **text and voice input**, **text-to-speech**, and a **locally hosted AI model (Gemma:7b)** via **Ollama**. This chatbot features a modern neon-themed UI with Lottie animations, responsive design, and robust error handling. It is ideal for developers and enthusiasts interested in deploying local AI chat systems.

---

## Table of Contents

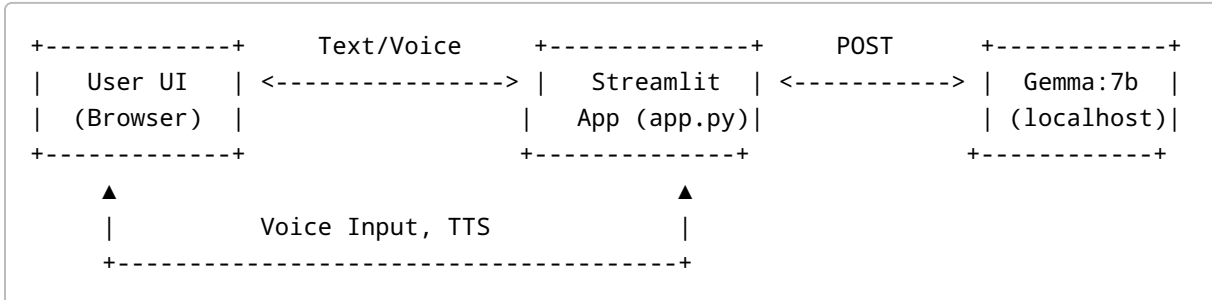
1. [Features](#)
  2. [Architecture](#)
  3. [Technologies Used](#)
  4. [Code Structure](#)
  5. [Dependencies](#)
  6. [Environment Configuration](#)
  7. [Setup Instructions](#)
  8. [Usage](#)
  9. [Key Components](#)
  10. [Styling](#)
  11. [Session State Management](#)
  12. [Error Handling](#)
  13. [Testing](#)
  14. [Security & Optimization Tips](#)
  15. [Limitations](#)
  16. [Future Improvements](#)
  17. [Version & Changelog](#)
  18. [Contributing](#)
  19. [License](#)
- 

## Features

- **Text Input & Output**
- **Voice Input via Microphone**
- **Text-to-Speech (TTS)** using `pyttsx3`
- **AI-Powered Responses** from Gemma:7b running locally
- **Special Commands** like "show date", "tell time"
- **Responsive UI** with dark neon theme and Lottie animations
- **Session-based Chat History**

- Graceful Error Handling

## Architecture



## Technologies Used

- Python 3.8+
- Streamlit
- requests
- speechrecognition
- pyttsx3
- streamlit-lottie
- python-dotenv
- CSS (via HTML injection)

## Code Structure

- `app.py` : Main application file
- `requirements.txt` : Python dependencies
- `launch.sh` : Shell script to start app
- `.env` : Environment configuration (MODEL\_URL, VOICE)

## Dependencies

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

**requirements.txt:**

```
streamlit
requests
```

```
speechrecognition
pyttsx3
streamlit-lottie
python-dotenv
```

---

## Environment Configuration

Create a `.env` file:

```
MODEL_URL=http://localhost:11434/api/generate
VOICE=Alex
```

Load it in `app.py`:

```
from dotenv import load_dotenv
load_dotenv()
```

---

## Setup Instructions

```
git clone <repository-url>
cd <repository-directory>
python -m venv venv
source venv/bin/activate # or venv\Scripts\activate on Windows
pip install -r requirements.txt
ollama run gemma:7b
streamlit run app.py
```

---

## Usage

1. Open browser: `http://localhost:8501`
  2. Type or speak a message
  3. Use 🗣️ to hear responses
  4. Try commands: `show date`, `tell time`, etc.
-

## Key Components

- **Lottie Animation:** Fetch JSON via `load_lottieurl`
  - **Speech Recognition:** Google API via `speech_recognition`
  - **TTS:** `pyttsx3` engine, macOS default: Alex
  - **AI Model Integration:** `POST` request to `localhost:11434`
- 

## Styling

- Dark theme with `#030508` background
  - Neon cyan borders ( `#00FFEA` )
  - Hover effects, scrollbars, and chat bubble animations
- 

## Session State Management

```
st.session_state['chat_history'] = [("user", "Hello")] # or []
st.session_state['is_speaking'] = False
st.session_state['current_reply'] = ""
```

---

## Error Handling

- `WaitTimeoutError`, `UnknownValueError`, `RequestError` for voice
  - HTTP request exceptions
  - TTS fallback handling
- 

## Testing

```
streamlit run app.py
```

Manual testing:

- Validate TTS works
  - Validate voice transcription
  - Validate AI response and command parsing
-

## Security & Optimization Tips

- Never expose `localhost:11434` publicly without a proxy
  - Use `streamlit run --server.headless true` for deployment
  - Add a reverse proxy (e.g., NGINX with SSL)
- 

## Limitations

- macOS-only voice (custom config needed for Linux/Windows)
  - Internet required for speech recognition
  - English only
  - Session data only (no DB persistence)
- 

## Future Improvements

- Cross-platform voice configuration
  - Persistent chat history (e.g., SQLite, JSON)
  - Multi-language voice/text support
  - Model selector UI
  - Chat analytics
  - Docker support
- 

## Version & Changelog

**Version:** v1.0.0 **Changelog:**

- Initial release with full text/voice/AI integration
  - Neon UI with Lottie animations
  - Special command support
- 

## Contributing

1. Fork the repo
  2. Create branch `git checkout -b feature-name`
  3. Commit & push
  4. Create a Pull Request
- 
-