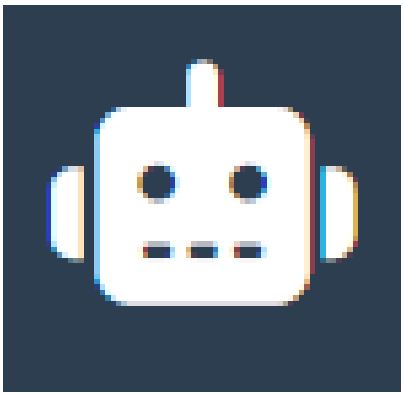
Resume Chatbot Project Report



Author: Christos Polimatidis
GitHub Repo: Christos' repo link

1. Introduction

I built **Chris Assistant**, a resume chatbot that simulates an interview by answering questions about my professional background. The goal was to create a clean, interactive web interface with persistent conversation history—all while keeping the backend simple but effective.

Features:

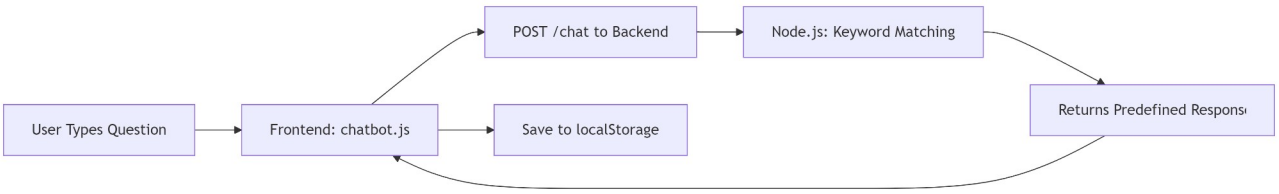
- **Conversation Memory**: Saves chat history locally (even after refresh).
- **Sample Questions**: One-click prompts for common interview questions.
- **Emoji-Powered Responses**: Friendly, engaging answers with relevant icons.
- **Typing Indicators**: Simulates "thinking" before replying.

Tech Stack:

- **Frontend**: HTML/CSS + Bootstrap (with custom animations)
- **Backend**: Node.js/Express (lightweight API)
- **Storage**: **localStorage** for conversation history

2. How It Works

System Architecture



Key Components

Frontend (chatbot.html/css/js)

- **UI**: Clean chat interface with:
- Sidebar showing conversation history
- Sample question buttons (e.g., "What are your skills?")
- Animated typing indicators (`...`)

Conversation Persistence:

- Uses `localStorage` via `history.js` to save/load chats.
- Each session tracks timestamps and message previews.
- **Backend (server.js)**
- **Keyword Matching**:
- Processes user input with regex to handle variations (e.g., "skill" → "skills").

Example response dictionary:

json

{
'skills|skill|expertise': "My skills include JavaScript, Python...",
'projects|worked on': "I built a weather app...",
'default': "Ask me about my skills or projects!"
}

•**CORS Enabled**: Allows frontend-backend communication during development.

3. Clever Details

Whole-Word Matching:

- Not just **includes()**—uses regex (**\bword\b**) to avoid false matches (e.g., "ski" ≠ "skills").

User Experience Touches:

- **Animated Status**: Pulsing "Online" badge with CSS.
- **Message Effects**: Smooth fade-in animations for new messages.
- **Error Handling**: Fallback response if the backend fails.

Scalable Design:

- Adding new Q&A pairs is as easy as extending the **resumeResponses** object.

4. Challenges & Solutions

Challenges	Solutions
Partial keyword matches	Used regex word boundaries
Persistant chat history	Implemented localStorage with timestamps
Backend-frontend disconnect	Added CORS middleware (app.use(cors()))