Software Requirements Specification (SRS)

Project Title: NoCode Chatbot Builder for Websites

Name: Yash Aneja
Registration Number: 23BCE1918

1. Introduction

1.1 Purpose

The purpose of this project is to create a NoCode platform that allows organizations and individuals to create and deploy AI-powered chatbots on their websites without writing any code. Users can define the chatbot's role, upload their website documentation, and instantly generate embeddings to power contextual responses. The system will then provide a unique JavaScript snippet that can be embedded in the user's website to integrate the chatbot seamlessly.

1.2 Scope

The system will provide:

- Role-based chatbot creation.
- Document upload for embedding generation.
- Integration with an AI language model (Gemini) for intelligent conversation.
- Automatic generation of a unique JS embed code for deployment.
- Real-time chat interface preview with drag-and-drop customization.
- Secure storage of embeddings and user data.
- Authentication and user account management.

1.3 Definitions, Acronyms, and Abbreviations

LLM: Large Language Model

JS: JavaScript

UI: User Interface

NoCode: Development approach without programming knowledge

RAG: Retrieval-Augmented Generation

2. Overall Description

2.1 Product Perspective

The NoCode Chatbot Builder will be a standalone web application where users can create, customize, and deploy AI chatbots on their websites. It integrates with:

- Supabase for authentication, database, and file storage.
- Vector database for storing embeddings.
- Gemini LLM API for chatbot responses.

2.2 Product Functions

- User registration, login, and profile management.

- Chatbot role definition and settings.
- Documentation upload (PDF, DOCX, TXT, etc.).
- Automatic vector embedding generation.
- JS embed code generation for chatbot deployment.
- Live chatbot preview with drag-and-drop customization.
- User dashboard to manage multiple chatbots.

2.3 User Classes and Characteristics

Website Owners: Want a chatbot without coding.

Developers: Need quick chatbot setup for clients.

Organizations: Require internal documentation-based assistants.

Returning Users: Manage multiple chatbots from the dashboard.

2.4 Operating Environment

- Web browsers (desktop and mobile).
- Cloud infrastructure hosting AI models, embeddings, and storage.
- Vite + modern JS frameworks for frontend.

2.5 Design and Implementation Constraints

- Must integrate with Supabase for authentication and database.
- Must comply with privacy and security regulations (GDPR, CCPA).
- Must handle large document uploads efficiently.
- Chatbot must load fast and not slow down the host website.

2.6 Assumptions and Dependencies

- Users have basic website editing access to insert JS code.
- Gemini API is available and stable.
- Vector embedding storage remains accessible and performant.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Account Management

- The system shall allow users to register, log in, and manage their profile.
- The system shall support OAuth and email-password authentication.

- The system shall allow users to define chatbot roles and behaviors.
- The system shall allow uploading website documentation for training.

3.1.3 Embedding Generation

- The system shall automatically generate and store embeddings per user.
- The system shall associate embeddings with the respective chatbot instance.

3.1.4 JS Code Generation

- The system shall generate a unique JavaScript snippet for each chatbot.
- The embed code shall load the chatbot on the target website without conflicts.

3.1.5 Live Preview & Customization

- The system shall provide a real-time chatbot preview.
- The preview shall support drag-and-drop repositioning and styling options.

- The system shall encrypt sensitive data.
- The system shall ensure embeddings and documents are stored securely.

3.2 Non-Functional Requirements

3.2.1 Performance

- The chatbot must load within 2 seconds on any supported browser.

3.2.2 Usability

- The platform must provide an intuitive drag-and-drop customization interface.

3.2.3 Reliability

- The system must maintain 99.5% uptime.
- AI response accuracy must remain consistent over time.

- The system must allow easy updates to embedding logic and UI components.

3.2.5 Compliance

- The system must follow GDPR, CCPA, and other relevant privacy laws.

4. External Interface Requirements

4.1 User Interfaces

- Responsive web UI for chatbot creation and customization.
- Minimalistic, professional, and animated UI with smooth transitions.
- Support for 3D objects and unique icons (non-classic).

4.2 Hardware Interfaces

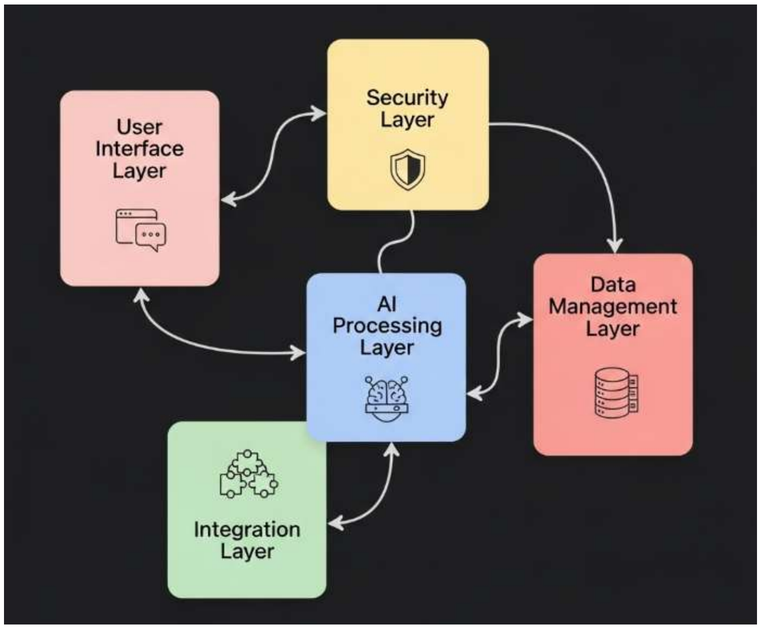
- Standard desktop and mobile devices.

4.3 Software Interfaces

- Supabase API for authentication and database operations.
- Gemini API for chatbot responses.
- Vector database API for storing and retrieving embeddings.

4.4 Communication Interfaces

- Secure communication via HTTPS and encrypted data transfer protocols.


Security Layer
User Interface Layer
Data Management Layer
AI Processing Layer
Integration Layer