This project is an AI-powered recruiting outreach assistant that helps recruiters generate email sequences based on user inputs. It consists of a React frontend, a Flask backend with WebSockets, LangChain for AI processing, and a database (SQLite/PostgreSQL) to store chat history.

Frontend:(react + tailwind css)

Chat Interface (Left side)

- A **c**hatbot UI (like ChatGPT) where recruiters provide input.
- Uses WebSockets for real-time communication.

Dynamic Workspace (Right side)

- A rich text editor (like a document editor) where AI-generated sequences appear.
- Users can edit manually and ask AI to adjust.
- Should sync edits with the AI in real time.

Backend (Flask)

LLM Processing

- Uses Langchain or OpenAI Assistants to generate and refine recruiting sequences.
  Real-time AI Updates: Use OpenAI streaming for a ChatGPT-like experience.

Database (PostgreSQL)

- Stores user details plus preferences, recruiting email sequences, and interactions.
  Context Memory: Store user-specific context/history in a database.

WebSockets (Flask-SocketIO)

- Enables real time updates between chat, AI, and editor.

User Flow

1. A user interacts with the ChatBar
   a. AI asks follow-up questions (e.g., “What’s the job role? What’s the tone of the message?”)
   b. AI follows up(What would you like to say?)

    c. A user inputs Message Details
    d. AI asks for more specifics (eg. "Could you provide more details on your recruiting needs?)
    e. A user provides final input (e.g., "Looking for 3+ years of experience and relocation flexibility").

2. Backend Processes Input Using OpenAI & LangChain
    a. The front end sends the concatenated inputs to the backend.
    b. LangChain wraps the input with a structured prompt for OpenAI
    c. OpenAI or claude returns a 3-step email sequence.
3. Generated Email Sequence is Stored in the Database
    a. The backend stores the generated sequence along with the session ID.
    b. This allows the AI to retrieve past conversations if needed.
    c. Allows AI to remember previous sessions
4. AI Sends the Sequence to the Frontend
    a. The backend sends the email sequence to the front end via WebSockets.
    b. The sequence is rendered in the Workspace (editable text area on the right).
5. User Edits or Asks AI to Modify the Sequence
    a. he user can edit the sequence manually in the workspace.
    b. OR, the user can type "I want to tweak the subject line" in ChatBar.
    c. AI receives editing instructions and modifies the sequence accordingly.
    d. The updated sequence replaces the old sequence in the workspace.

Technical Architecture

```
 [User Browser] <--WebSocket--> [Flask Backend] <---> [LangChain/GPT-4]

|                                                 |

|                                                 |

[React UI] <--------------------------------->   [SQLite DB]
```

Frontend Stack

  React + TypeScript

Socket.io-client for real-time communication

Component-based architecture with ChatBar and Workspace components

Backend Stack

Flask Flask-SocketIO

LangChain with OpenAI integration SQLite3 for database

```
SYSTEM_PROMPT = """You are Helix, an AI recruiting assistant that helps generate
recruitment outreach sequences.
You should first understand the:
1. Role being recruited for
2. Target candidate profile
3. Company culture and values
4. Unique value proposition

When generating sequences:
1. Be professional and personalized
2. Use {{first_name}} for personalization
3. Highlight relevant job requirements and opportunities
4. Emphasize company culture and growth potential
5. Include clear call-to-actions for next steps"""
```