WELLBOT

Project Statement:

Intoday's fast-paced digital era, individuals seek accurate and accessible wellness information without medical complexity. Many face challenges understanding professional terminology or accessing reliable online health guidance. The WellnessBot project addresses this gap by developing a bilingual, AI-powered wellness chatbot that simplifies health communication. It provides symptom checking, first- aid guidance, and wellness tips in English and Hindi, enabling users from diverse backgrounds to access foundational wellness knowledge easily. The chatbot integrates Flask, Rasa NLU, and React/Streamlit interfaces, ensuring interactive, context-aware, and multilingual wellness support. It emphasizes that while the chatbot offers wellness advice, professional medical consultation is essential for diagnosis and treatment.

Outcomes

- A fully functional AI-driven wellness chatbot integrated with Flask and Rasa NLU.
- Multilingual (English and Hindi) communication through translation API.
- User authentication and profile management for personalized chatbot interactions.
- Knowledge base integration for symptom-based responses and first-aid tips.
- Dual frontends using React (Web) and Streamlit (App) for seamless usability.
- SQLite database for storing chat history and user profiles.
- Robust backend API supporting authentication, chat logic, and multilingual functio nality.

Modules Implemented

Milestone 1 (Weeks 1–2)

Module1:WebFrontend(React/Vite)

Objective: Design a modern, responsive web interface for user authentication, profile handling, and chat functionality.

Components:

- AuthPage.jsx: User login interface
- ProfilePage.jsx: User profile management
- App.jsx: Application router for page navigation

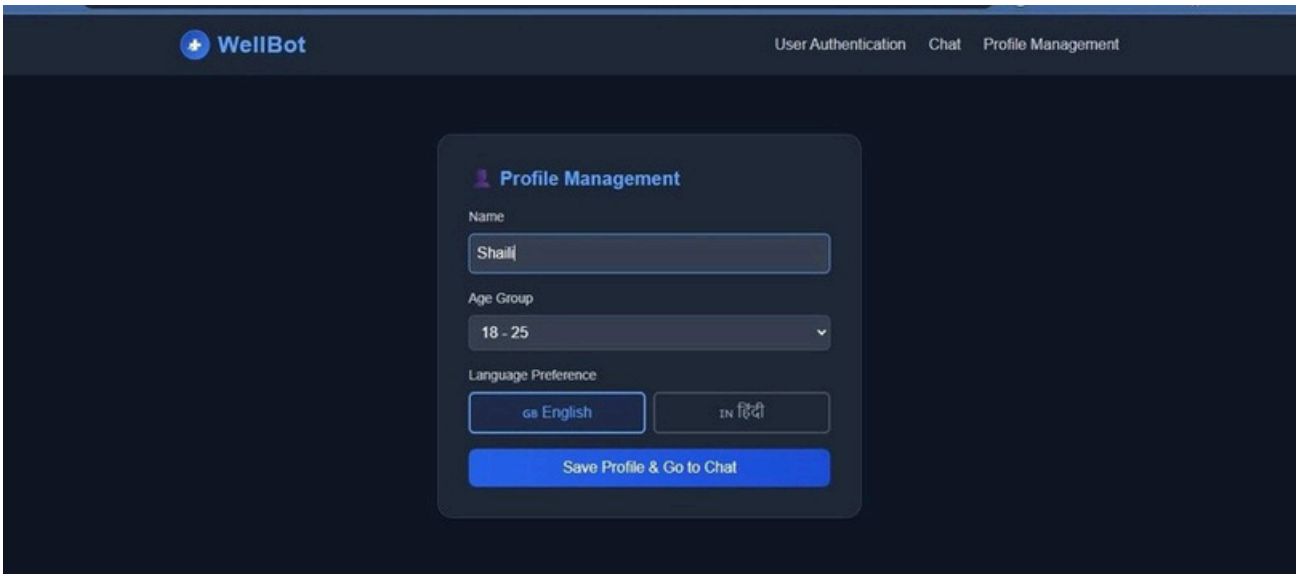
2. Features:

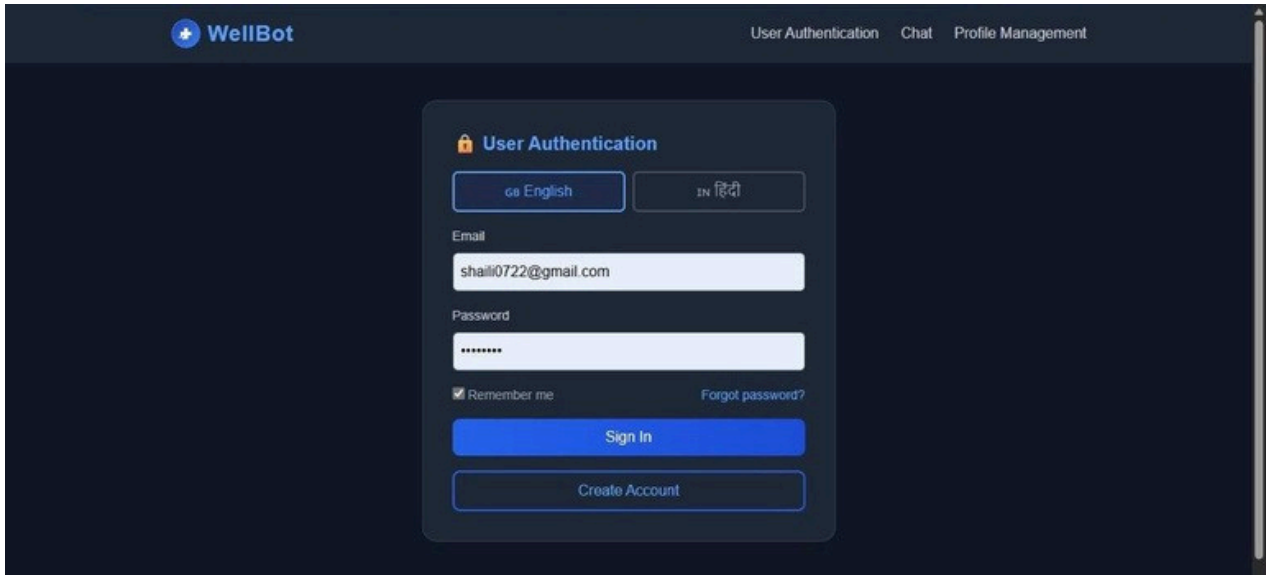
- Modern UI: Built with React and Tailwind CSS
- Routing: Navigation managed using React Router
- API Integration: Axios used for backend communication
- Responsive Design: Optimized for both desktop and mobile devices

Technology Used:

- React
- Vite
- Tailwind CSS
- Axios

Screenshots :





Milestone 2 (Weeks 2-5)

Module 2: Backend API

Objective:Develop a robust Flask-based backend API to handle authentication, chat processing, and knowledge base retrieval using Rasa NLU.

Key Components Implemented

1. Flask Application Setup

- **Framework:** Flask with CORS for cross-origin requests
- **Port**: localhost:8000
- **Debug Mode**: Enabled during development

2. User AuthenticationSystem

- **Token-Based Authentication**: UUIDs used for secure session handling
- **In-Memory User Store**: For managing user credentials and sessions

Endpoints Implemented:

- POST /auth/login – Login with email and password
- GET /auth/validate – Validate session tokens
- GET /profile – Retrieve user profile data
- PUT /profile – Update user preferences (e.g., language

3. Chat System

- Natural Language Understanding (NLU): Integrated Rasa NLU for intent recognition and entity extraction.
- Knowledge Base: kb.json file containing structured wellness data (symptoms, first aid, and fitness tips).

- Fallback Mechanism: Rasa Response Selector handles complex or ambiguous queries.
- Chat History: Integrated SQLite database to store message history with timestamps for each user session.

4. Knowledge Base Management

- JSON structure organized under tags such as symptom, wellness, and first_aid.
- Multilingual responses (English/Hindi) supported using translation API.
- Intent Matching: Based on keyword and direct symptom classification.

Files Implemented:

- backend/app.py – Flask backend
- backend/kb.json – Knowledge base file
- backend/chat_history.db – SQLite database

5. Database Integration

- SQLite used for lightweight local data storage.
- Stores user messages, bot responses, and timestamps for persistence.

Technologies Used (Backend)

- Flask
- Flask-CORS
- Rasa NLU
- SQLite,
- Python-dotenv

Rasa NLU Integration

Objective:
Integrate Rasa NLU for intent detection, entity extraction, and response generation using the main configuration and training files.

1. Intent Recognition

Rasa identifies user intents such as:

- symptom_query
- ask_first_aid
- ask_wellness_tip
- greeting
- fallback

Example:

User input: “I have a fever.”

→ Intent: symptom_query

→ Action: utter_symptom_response → Returns related wellness advice.

2. TrainingDataFiles

File	Description
data/nlu.yml	Containstrainingexamples for user intents like like “I have a cough” or “What to do for fever?”
domain.yml	Definesintents, responses, entities, and actions used in the chatbot
config.yml	Defines the Rasa NLU pipeline for intent recognition and entity extraction

3. Configuration (config.yml)

Defines the Rasa NLU pipeline and machine learning components used to understand user messages.

Purpose :

Allows Rasa to process natural language text, classify user intents, and generate accurate responses.

4. Dialogue Flow

Example:
Input: "I feel dizzy."
→ Intent: symptom_query
→ Action: utter_symptom_response
→ Bot: "Make sure you stay hydrated and rest. If symptoms persist, consult a doctor."
This ensures consistent, context-aware, and natural dialogue flow.

5. Integration with Knowledge Base

Implemented in response_generator.py.

- Connects Rasa with knowledge_base.json to dynamically fetch wellness data.
- Retrieves relevant first-aid steps, symptom responses, and wellness tips.
- If a match is not found, Rasa's Response Selector provides a fallback response.

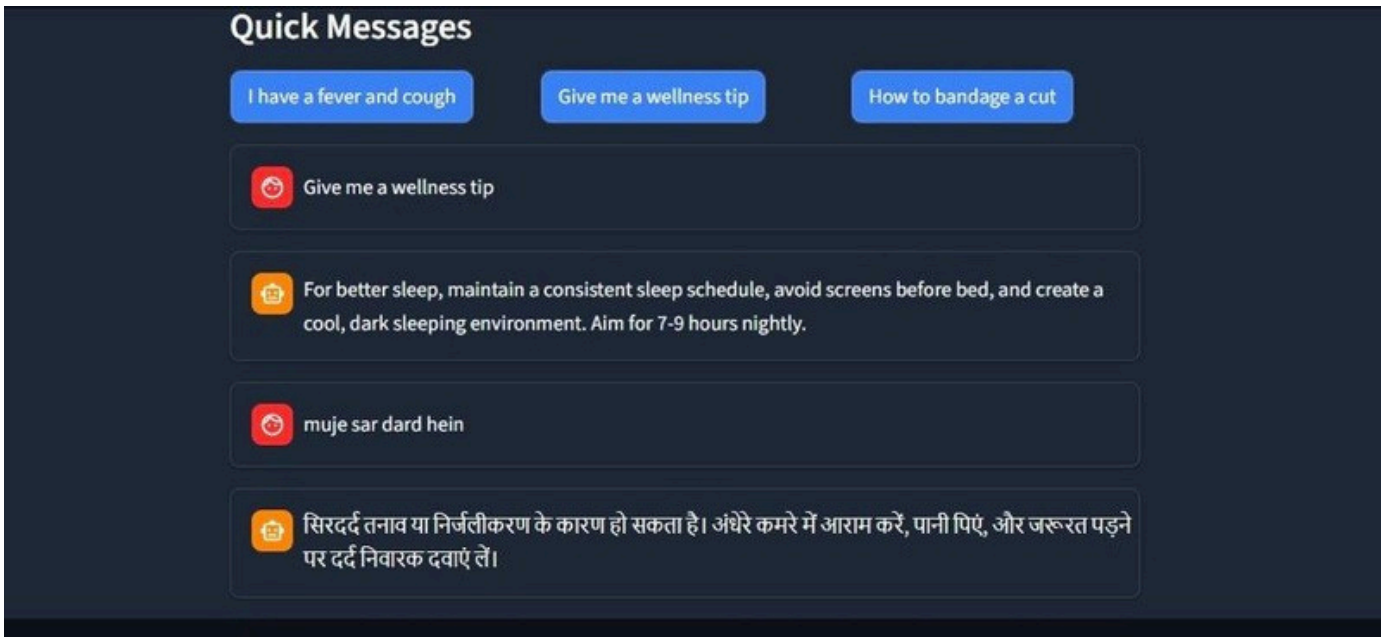
6. Multilingual Support

Implemented using Deep Translator (GoogleTranslator).
All responses are automatically translated into Hindi based on the user's profile language preference.
Ensures bilingual accessibility throughout the user experience.

7. Database Integration

Chat history is stored in chat_history.db for every session.
Each entry logs:

- User message
- Detected intent
- Bot response
- Timestamp



Frontend Integration

Streamlit Frontend *(frontend/streamlit_chat.py)*

Objective:
Develop a lightweight, user-friendly interface for interacting with the chatbot API.

Features:

- Logininterface with user session storage
- Real-time chat with conversation history
- Quick message buttons for common queries
- Language toggle (English/Hindi)
- Persistent login using Streamlit's session state

Technology Used:

- Streamlit
- Requests,
- Deep Translator

Expected Results

- **Chatbot** responds accurately to health-related and symptom-based queries in both **English and Hindi.**
- Provides **reliable first-aid and wellness guidance** using the integrated knowledge base.
- **Rasa NLU** ensures correct intent recognition and appropriate response generation.
- **Flask backend** securely manages authentication, user sessions, and chat flow.
- **Multilingual translation** delivers context-aware responses according to user language preference.
- **Smooth, responsive UI** through React and Streamlit interfaces ensures user-friendly digital health support.
- **SQLite database** stores chat history and user profiles for continuity and personalization.
- Enables r**eal-time, bilingual wellness communication** for accessible and trustworthy health assistance.