

WellBot Global wellness assistant chatbot : Project Documentation

.



By : Under the guidance of:

Omkar Bondge Dr.K.Santhiya

Kakarlapudi Deepika

Shaili Boddula

Nirmal Kumar

# Abstract

In the digital era, where access to authentic and timely health information is essential, users often struggle to distinguish credible guidance from misleading or incomplete sources available online. This challenge becomes more serious when individuals seek quick, understandable, and non-diagnostic wellness advice for everyday ailments. The **Wellness Guide Bot** project aims to bridge this gap by offering a structured, safe, and reliable conversational platform designed to deliver general health and wellness information. The system is built to assist users with first-aid advice, symptom explanations, and preventive care information while ensuring adherence to ethical guidelines and avoiding medical diagnosis.

To achieve this, the project integrates state-of-the-art technologies including the **Rasa open-source conversational AI framework** and a **Flask-based web application**. The bot is engineered with a multilingual Natural Language Understanding (NLU) engine capable of recognizing intents and extracting complex entities such as symptom severity, duration, and affected body location. Additionally, the system incorporates secure **JWT-based user authentication**, user profile personalization, and bilingual support in **English and Hindi**, ensuring accessibility to a wider demographic. The backend further interacts with a curated **SQLite knowledge base**, ensuring that all responses originate from verified and structured health information modeled after reputable sources such as WHO and CDC guidelines.

Beyond user interaction, the Wellness Guide Bot features a comprehensive **admin dashboard** that enables system administrators to manage the knowledge base, monitor user queries, analyze usage trends, and review feedback through an integrated rating and comment system. This dashboard enhances the maintainability and scalability of the system, ensuring continuous improvement and content accuracy. This report presents a complete overview of the software development lifecycle including requirement analysis, architectural design, database modeling, NLU training, system integration, testing methodologies, and deployment strategies. The result is a robust and user-centric wellness chatbot that addresses the growing need for accessible, multilingual, and trustworthy digital health support.

# Table of Contents

1. **Introduction**
   * 1.1 Background
   * 1.2 Problem Statement
   * 1.3 Project Objectives
   * 1.4 Scope of the Project
2. **Literature Review**
   * 2.1 Evolution of Chatbots in Healthcare
   * 2.2 Comparison with Existing Solutions
   * 2.3 The Role of Multilingual AI
3. **System Analysis**
   * 3.1 Feasibility Study
   * 3.2 Functional Requirements
   * 3.3 Non-Functional Requirements
   * 3.4 Hardware & Software Requirements
4. **System Design**
   * 4.1 System Architecture
   * 4.2 Database Design (ER Diagram)
   * 4.3 User Interface Design
5. **Implementation Details**
   * 5.1 Module 1: User Authentication (Flask)
   * 5.2 Module 2: Conversational Core (Rasa)
   * 5.3 Module 3: Advanced NLU & Multilingualism
   * 5.4 Module 4: Admin Dashboard & Feedback
6. **Testing & Validation**
   * 6.1 Unit Testing
   * 6.2 Integration Testing
   * 6.3 User Acceptance Testing (UAT)
7. **Conclusion & Future Scope**
8. **References**

# 1. Introduction

## 1.1 Background

The intersection of Artificial Intelligence (AI) and healthcare—often termed "HealthTech"—has seen explosive growth. Chatbots are becoming the first line of defense in healthcare triage, offering users instant answers to common questions. However, many existing solutions are either too simplistic (rule-based buttons), too expensive (proprietary medical AI), or lack support for regional languages like Hindi.

## 1.2 Problem Statement

Users seeking health advice often face three main hurdles:

1. **Information Overload:** Search engines provide millions of conflicting results.
2. **Language Barriers:** Most high-quality health info is in English, alienating Hindi speakers.
3. **Lack of Trust:** It is difficult to distinguish between verified medical advice and internet myths.

## 1.3 Project Objectives

The primary objective is to develop a **"Wellness Guide Bot"** that acts as a reliable, non-diagnostic health assistant.

* To implement a secure login system to track user interactions.
* To allow users to query symptoms, first-aid, and wellness tips in natural language.
* To support both English and Hindi input and output seamlessly.
* To provide an admin panel for dynamic content updates without changing code.

## 1.4 Scope of the Project

The project covers the development of a web-based application. It includes the backend server (Flask), the AI engine (Rasa), and the database (SQLite).

* **In Scope:** Symptom checking for common ailments (cold, flu), first aid advice, wellness tips, user profiles, admin analytics.
* **Out of Scope:** Medical diagnosis, prescription generation, emergency response services, integration with live doctors.

# 2. Literature Review

## 2.1 Evolution of Chatbots in Healthcare

Early chatbots like ELIZA (1966) were simple pattern matchers. Modern bots use Transformer architectures (like BERT) to understand context. In healthcare, bots like Babylon Health and Ada have set the standard, but they often require expensive subscriptions and are closed-source.

## 2.2 Comparison with Existing Solutions

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | **Standard Web Search** | **Babylon/Ada Health** | **Wellness Guide Bot** |
| **Cost** | Free | High Subscription | **Free / Open Source** |
| **Privacy** | User tracked by ads | Data stored on cloud | **Local/Private Hosting** |
| **Language** | Varies | Mostly English | **Bilingual (En/Hi)** |
| **Customization** | None | None | **Full Admin Control** |

## 2.3 The Role of Multilingual AI

In diverse nations like India, English-only applications exclude a vast demographic. By integrating Hindi NLU (Natural Language Understanding), this project bridges the digital divide, ensuring equitable access to health information.

# 3. System Analysis

## 3.1 Feasibility Study

* **Technical Feasibility:** The project uses Python, Flask, and Rasa, which are mature, well-supported technologies. The team possesses the necessary skills.
* **Operational Feasibility:** The system is designed to be self-contained. Once deployed, the Admin Dashboard allows non-technical staff to maintain the knowledge base.
* **Economic Feasibility:** Using open-source tools (SQLite, Rasa, Python) ensures zero licensing costs, making the project highly cost-effective.

## 3.2 Functional Requirements

* **FR-01:** System must allow users to register and login.
* **FR-02:** System must encrypt user passwords.
* **FR-03:** Chatbot must identify intents: ask\_symptom, ask\_first\_aid, ask\_wellness\_tip, ask\_prevention.
* **FR-04:** Chatbot must support English and Hindi based on user profile.
* **FR-05:** Admin must be able to Add/Delete health tips via the dashboard.
* **FR-06:** Users must be able to rate bot responses (Thumbs Up/Down).

## 3.3 Non-Functional Requirements

* **Performance:** Chatbot response time should be under 2 seconds.
* **Scalability:** The database should handle thousands of health tips.
* **Security:** All API endpoints must be protected via JWT (JSON Web Tokens).
* **Reliability:** The bot must provide a default fallback message if it does not understand a query.

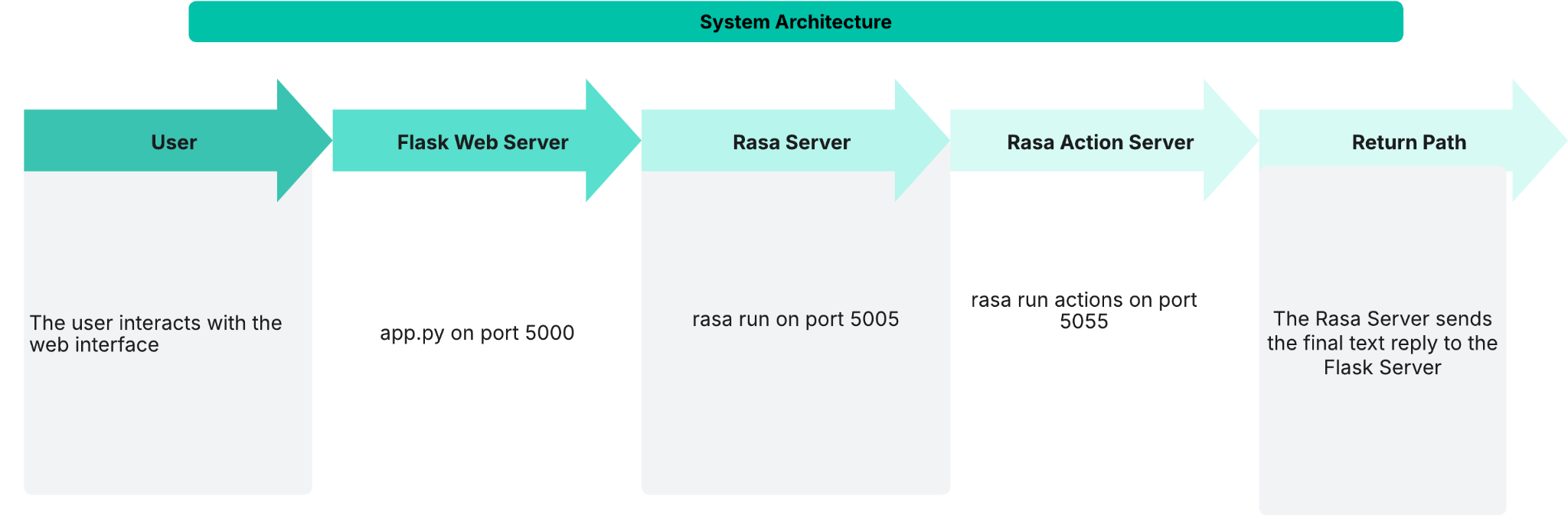
## 3.4 Hardware & Software Requirements

* **OS:** Windows 10/11, Linux, or MacOS.
* **Language:** Python 3.10.
* **Frameworks:** Flask 2.x, Rasa 3.x.
* **Database:** SQLite 3.
* **Browser:** Chrome, Firefox, or Edge.
* **RAM:** Minimum 8GB (required for training Rasa models).

# 4. System Design

## 4.1 System Architecture

The system follows a **Microservices-inspired Architecture**. While currently hosted locally, the components are decoupled to allow independent scaling.



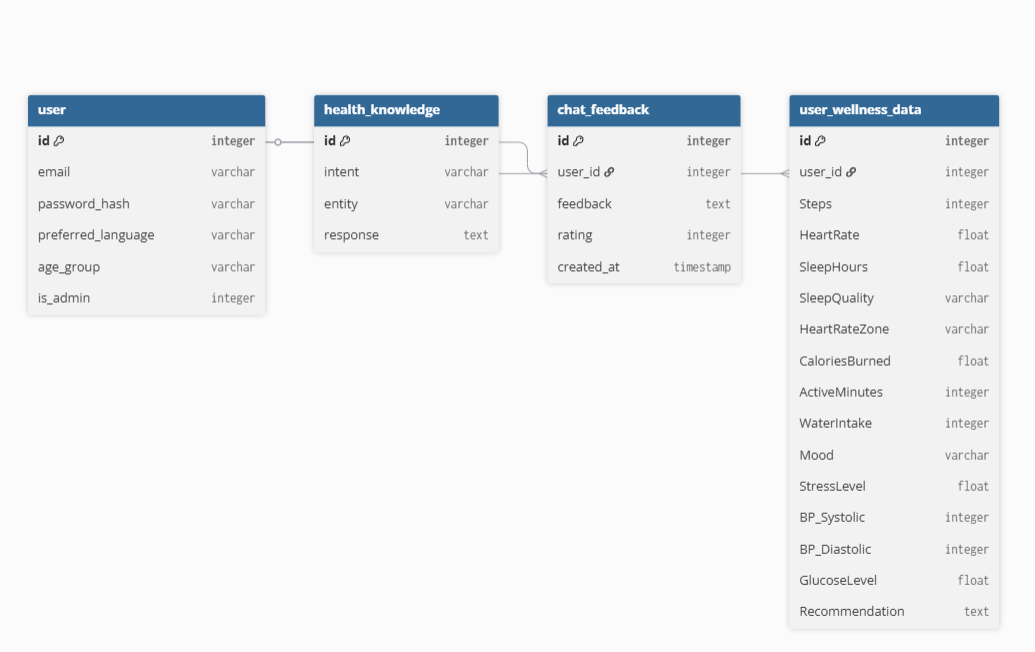
#### Data & Process Flow:

1. **User (Browser):** The user interacts with the web interface (e.g., login.html, chat.html).
   * On **Login**, JavaScript sends a POST request to the Flask API (/login).
   * On **Chat**, JavaScript sends a POST request with the message and JWT to the Flask API (/chat).
2. **Flask Web Server (app.py on port 5000):**
   * **Auth:** Validates the user against the User table in project.db. It sets a secure JWT cookie on login.
   * **Security:** The /chat endpoint is protected by @jwt\_required(), verifying the user's cookie.
   * **Gateway:** It passes the user's message and language preference to the Rasa Server.
3. **Rasa Server (rasa run on port 5005):**
   * **NLU:** Receives the message (in English or Hindi) and parses it using the multilingual NLU model (trained on data/nlu.yml and data/hi/nlu.yml).
   * **Core:** Identifies the intent (e.g., ask\_symptom) and entities (e.g., condition: "cold").
   * **Policy:** The rules.yml or policy model decides the next step. If it's a database query, it calls action\_query\_knowledge\_base.
4. **Rasa Action Server (rasa run actions on port 5055):**
   * **Action:** The action\_query\_knowledge\_base function in actions.py is triggered.
   * **DB Query:** The action server connects to project.db, queries the health\_knowledge table using the intent, entity, and user language.
   * **Response:** The action server sends the formatted text response (e.g., the Hindi symptom description) back to the Rasa Server.
5. **Return Path:** The Rasa Server sends the final text reply to the Flask Server, which in turn sends it as a JSON response to the user's browser, where JavaScript displays it.

.

## 4.2 Database Design (Schema)

The database is normalized to 3NF to reduce redundancy.



**Table: User**

* id (Integer, Primary Key): Unique identifier.
* email (String, Unique): User's login email.
* password\_hash (String): Encrypted password string.
* preferred\_language (String): 'en' or 'hi'.
* is\_admin (Boolean): Flag for administrative access.

**Table: HealthKnowledge**

* id (Integer, Primary Key).
* intent (String): e.g., 'ask\_symptom'.
* entity (String): e.g., 'flu'.
* response\_en (Text): English answer.
* response\_hi (Text): Hindi answer.

**Table: ChatFeedback**

* id (Integer, Primary Key).
* user\_message (Text).
* bot\_response (Text).
* rating (String): 'good' or 'bad'.
* comment (Text): User's optional feedback.

## 4.3 User Interface Design

* **Theme:** Professional "Indigo/Violet" SaaS aesthetic.
* **Layout:** Responsive Sidebar layout for the Admin Dashboard; Two-column layout for the Chat Interface.
* **UX Features:**
  + Hover effects on buttons.
  + Dynamic feedback buttons (only appear on bot messages).

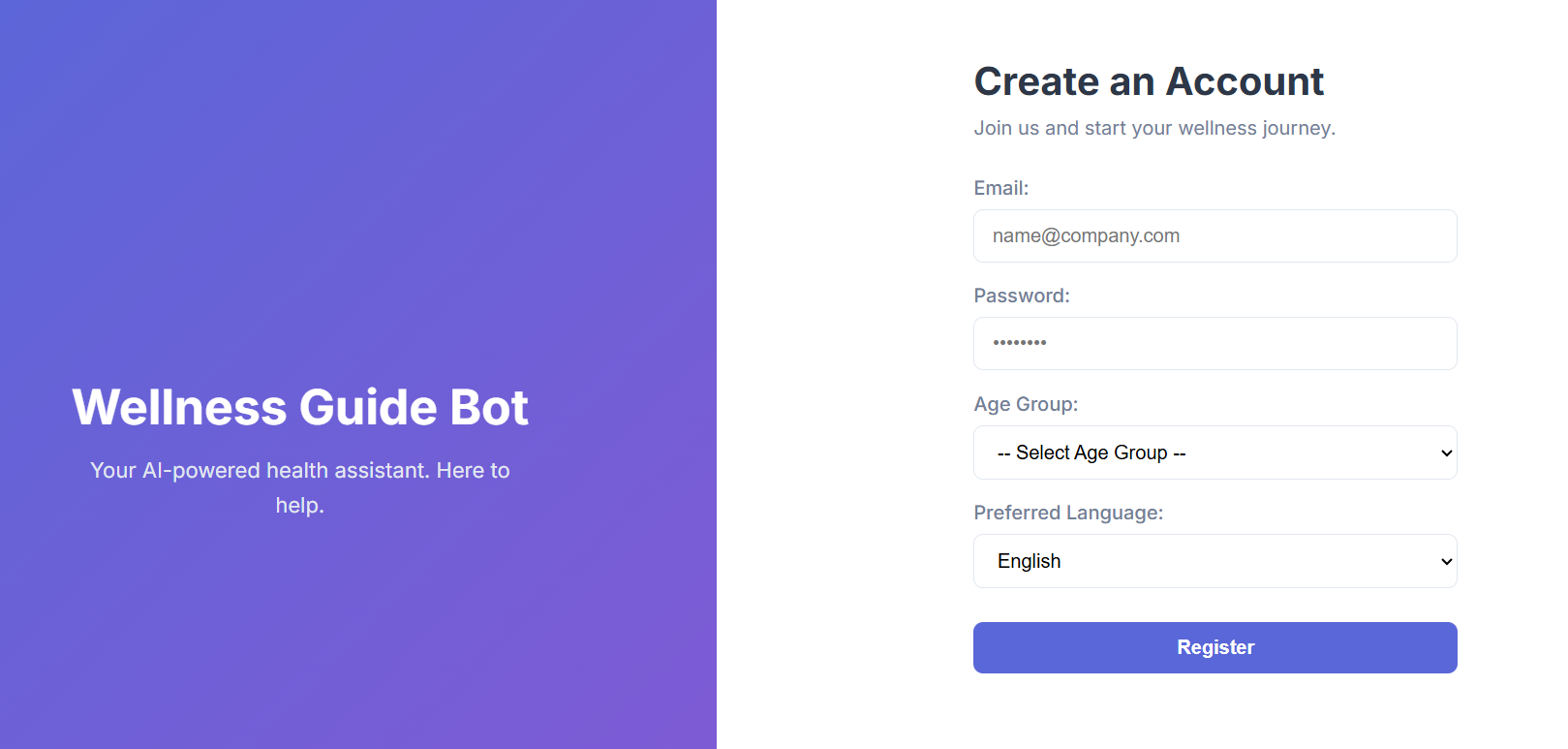
# 5. Implementation Details

### 5.1. Milestone 1: Foundation - User Authentication & Profile

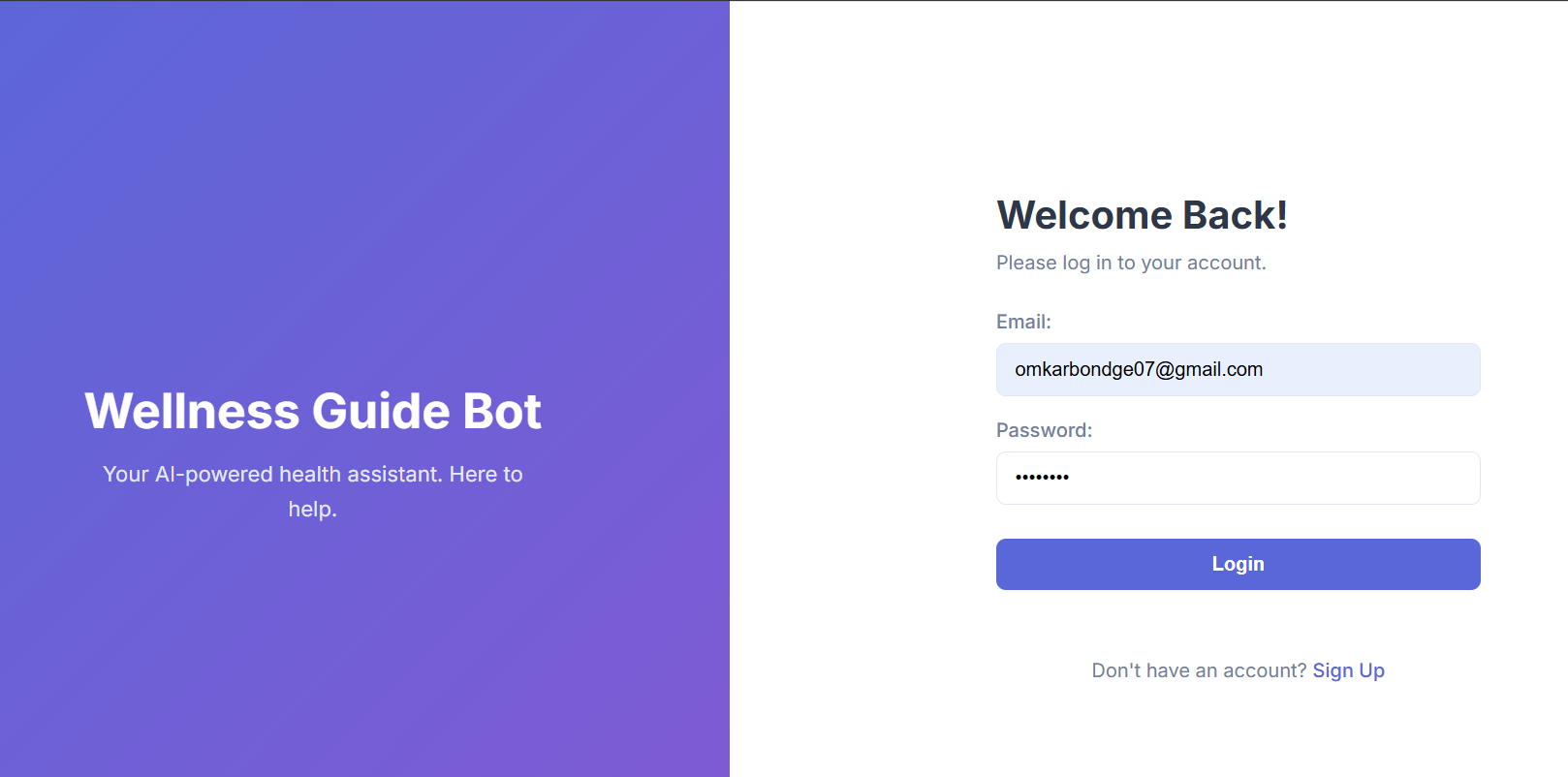
**Objective:** To build a secure and robust backend for managing user accounts and profiles.

#### Key Features Implemented:

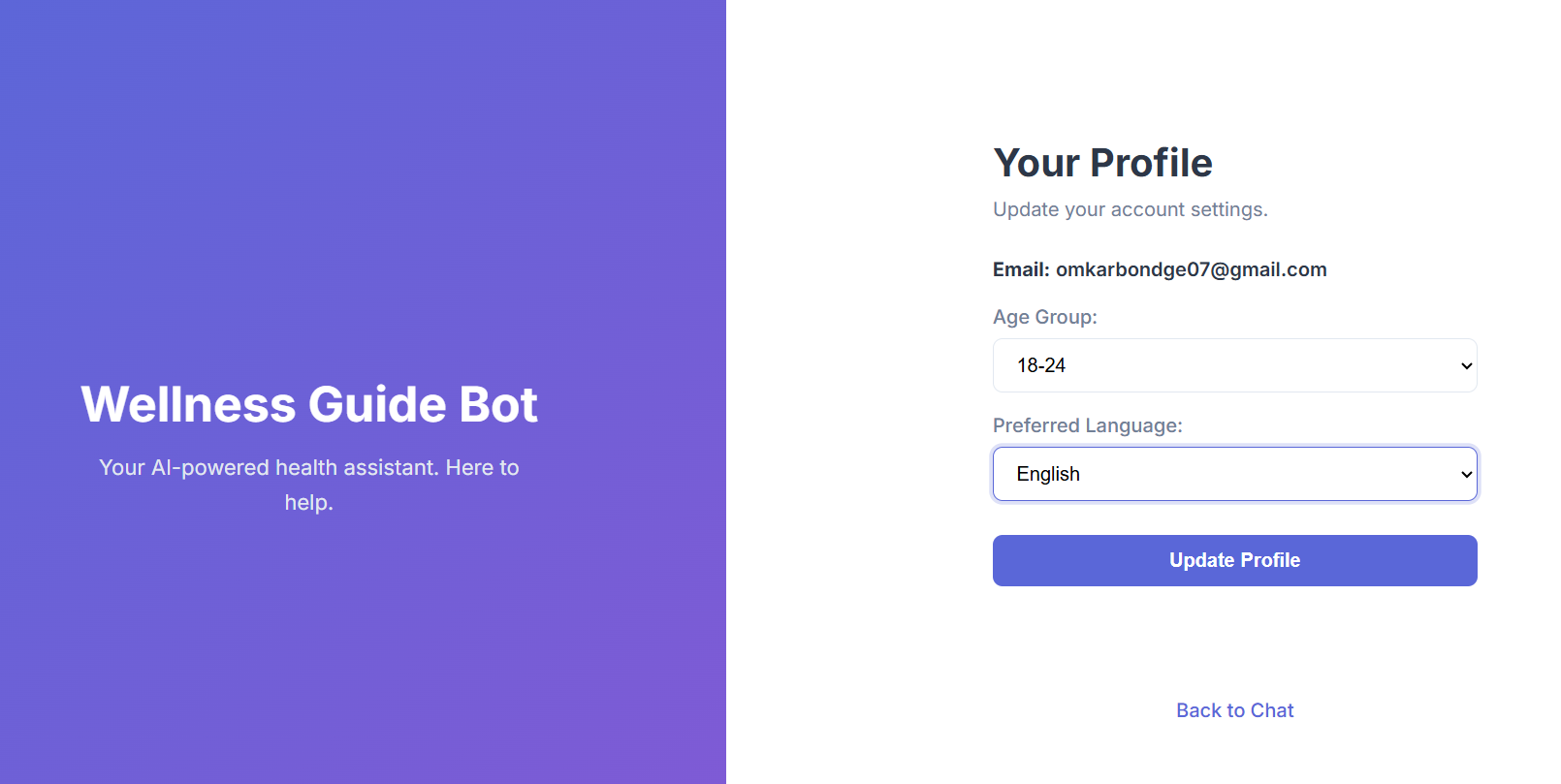
* **User Registration (/register):** Securely creates a new user in the user table of the project.db database. Passwords are salted and hashed using werkzeug.security. This endpoint now captures:
  + Email
  + Password
  + Age Group
  + Preferred Language (e.g., en or hi)



**User Login (/login):** Validates user credentials. On success, it generates a JSON Web Token (JWT) and sets it as a secure, HTTP-only cookie in the user's browser, creating a persistent login session.



* **Profile Management (/profile):** A protected API endpoint that allows a logged-in user to GET their current profile information or PUT updates (e.g., changing their preferred language or age group).



* **Page Security:** The main chat page (/chat\_page) and profile page (/profile\_page) are protected, automatically redirecting any unauthenticated users to the login page.

### 5.2. Milestone 2: Core - Conversational AI & Web Interface

**Objective:** To build the core chatbot, connect it to a knowledge base, and create a functional web interface for users to interact with.

#### Key Features Implemented:

* **Rasa NLU Model:** Trained an initial NLU model to understand basic intents (greet, goodbye, ask\_symptom, ask\_first\_aid).
* **SQLite Knowledge Base:** Created a health\_knowledge table in the project.db database to store all health information, including columns for intent, entity, response\_en, and response\_hi.
* **Data Loading:** Wrote a load\_knowledge.py script using Pandas to read a health\_knowledge.csv file and populate the SQLite database.
* **Custom Action Server:** Developed an actions.py file containing ActionQueryKnowledgeBase. This action connects to the SQLite database, queries for the correct information based on the user's intent and detected entity, and formulates a response.
* **Integrated** Web **UI:** Replaced the initial test UI with a full-fledged Flask-based front-end. This involved:
  + Creating login.html, register.html, chat.html, and profile.html templates.
  + Developing style.css for a professional, multi-column layout.
  + Writing auth.js to handle login/registration API calls and chat.js to handle sending chat messages to the /chat API endpoint.

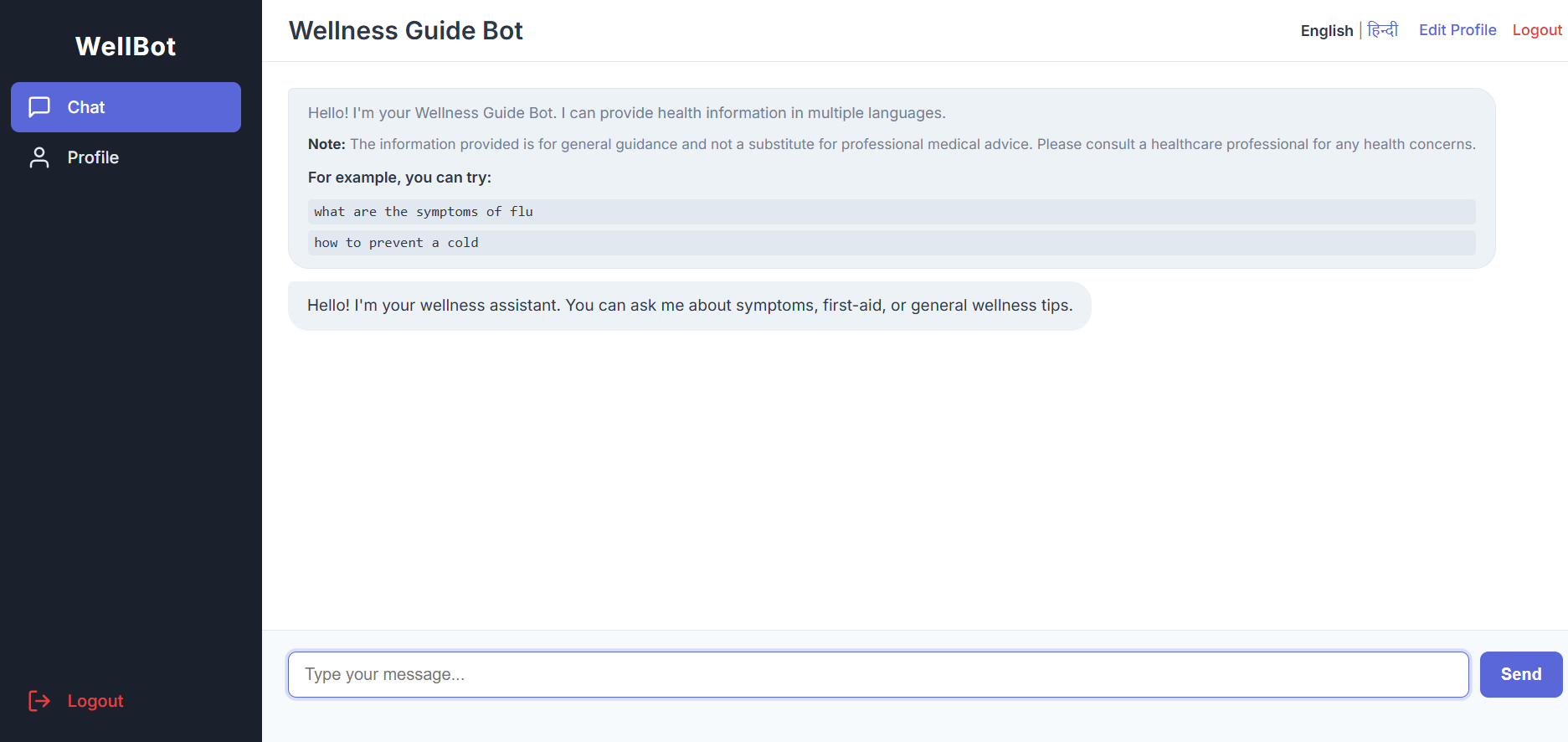


Fig : ChatBot Interface

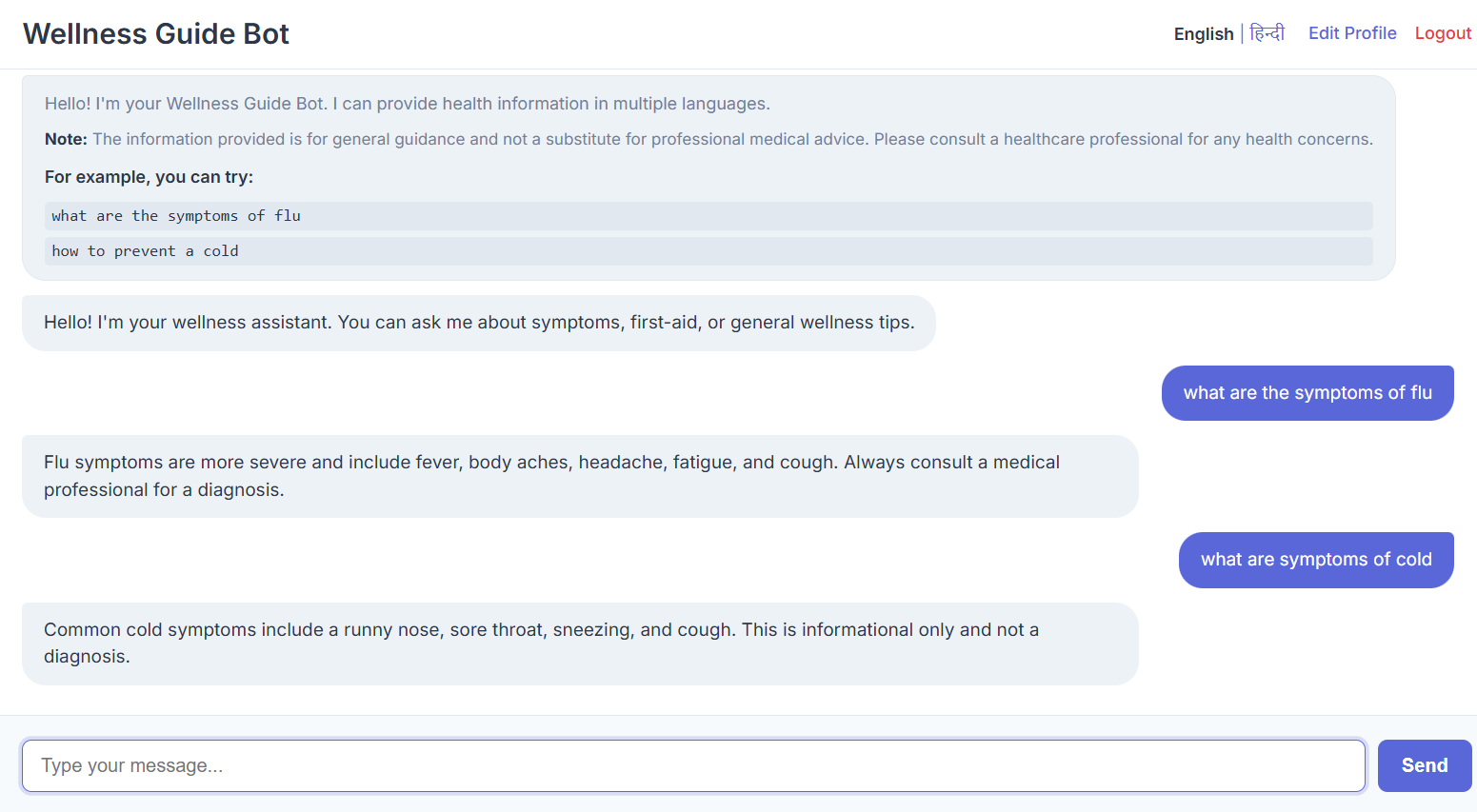


Fig : ChatBot Interactions

### 5.3. Milestone 3: Enhancement - Advanced NLP & Expansion

**Objective:** To significantly enhance the chatbot's intelligence, safety, and scope, fulfilling the project's advanced requirements.

#### Key Features Implemented:

* **Expanded Knowledge Base:** The health\_knowledge.csv file was expanded to over **100 entries**, including a much wider range of illnesses, first-aid procedures, wellness tips, and prevention advice, all styled after public health sources like the WHO and CDC.
* **Ethical Guardrails:**
  + An utter\_disclaimer was added to the domain.yml.
  + A new rule was added to rules.yml to make the bot automatically send this disclaimer at the start of every new conversation, ensuring users are aware it is not a medical professional.
* **Multilingual NLU (Hindi):**
  + The config.yml was updated to support multilingual processing.
  + A new training file, data/hi/nlu.yml, was created with a comprehensive set of Hindi examples, enabling the bot to *understand* intents and entities in Hindi.
* **Multi-language Responses:**
  + The app.py Flask server was updated to pass the user's preferred\_language (from their profile) to the Rasa Server as metadata.
  + The actions.py file was updated to read this metadata, and dynamically select either the response\_en or response\_hi column from the database, making the bot truly bilingual.
* **Advanced Entity Extraction:**
  + The domain.yml was updated to include new entities: duration, severity, and location.
  + Both data/nlu.yml (English) and data/hi/nlu.yml (Hindi) were updated with new examples to train the NLU to extract this more specific information (e.g., "I have a [severe](https://www.google.com/search?q=severity) [cold](https://www.google.com/search?q=condition) [for 3 days](https://www.google.com/search?q=duration)").

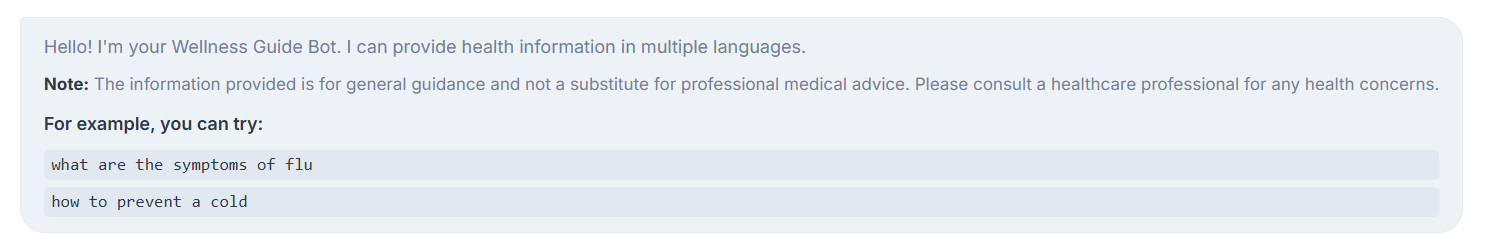


Fig:utter\_disclaimer

### 

Fig : Advanced Entity Extraction

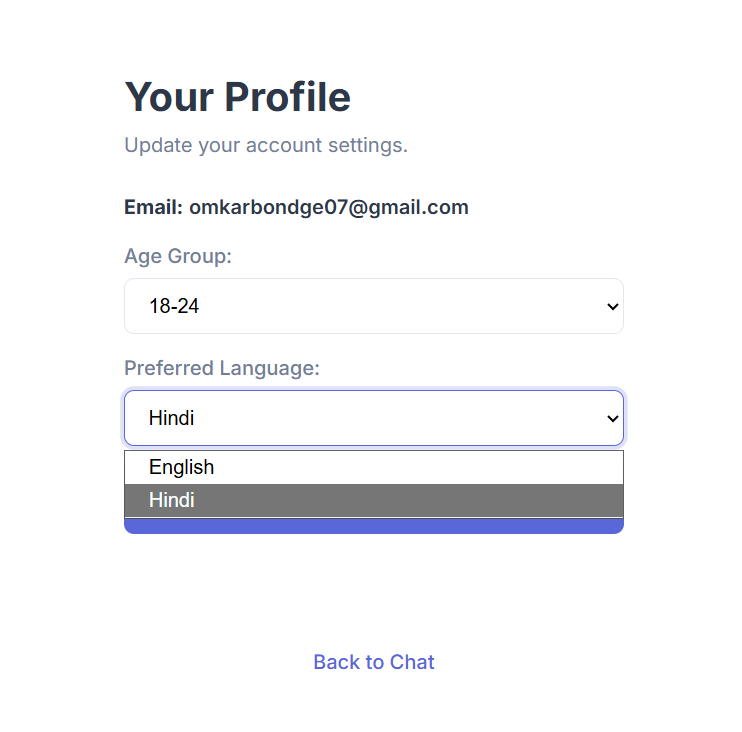


Fig: Multilingual Support

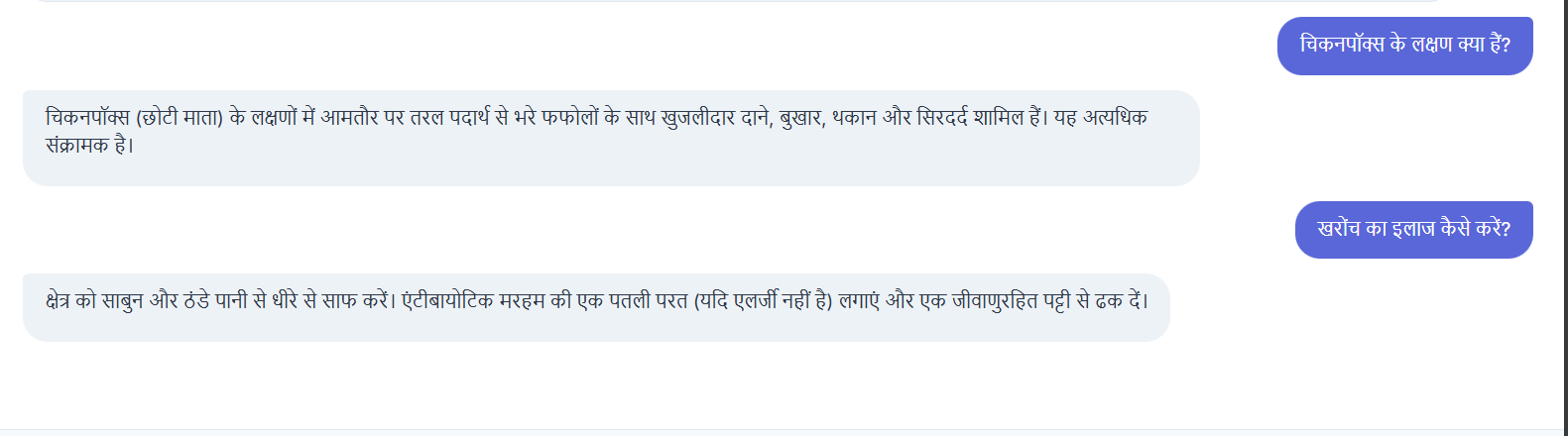


Fig : Hindi Responses

### Milestone 4: Admin Dashboard & System Refinement

#### Key Features Implemented:

**Admin Dashboard :**

* Developed a secure, password-protected web interface for administrators.
* **Knowledge Base Management:** Allows admins to view all current health tips stored in the SQLite database. Includes a form to add new intent-entity-response mappings directly from the UI, and the ability to delete outdated or incorrect entries.
* **User Analytics:** Displays a view of user interaction logs (from the user\_wellness\_data table), enabling admins to monitor common queries and usage patterns.
* **Feedback Review:** A dedicated table to review user feedback (ratings and comments), helping admins identify areas where the chatbot needs improvement.



Fig : Knowledge Base Management



Fig : User Analytics

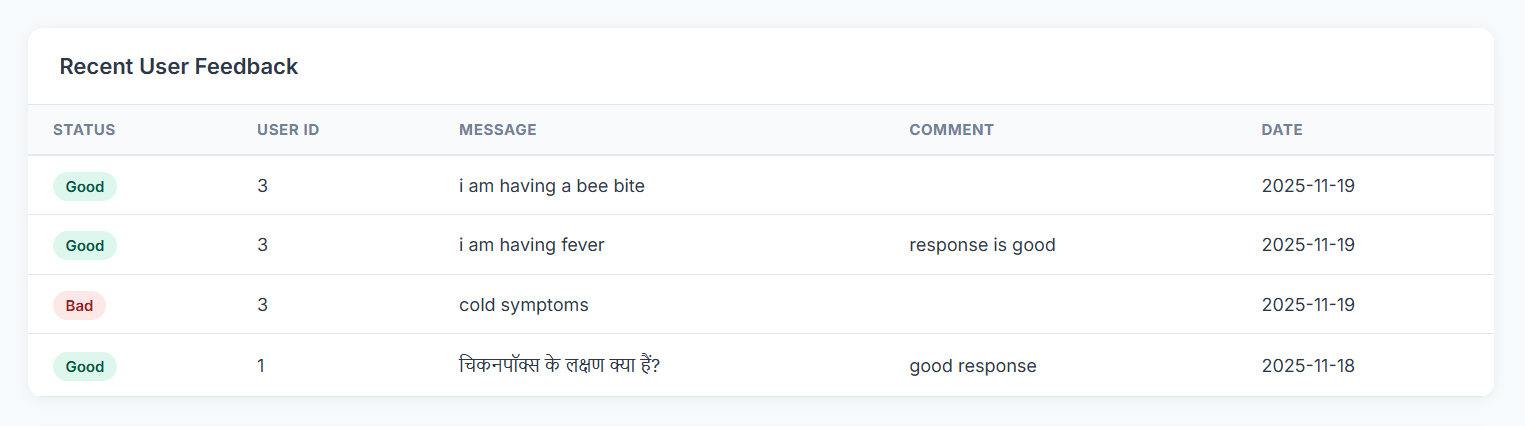


Fig : User Feedback Table

**User Feedback Mechanism:**

* Implemented a "Thumbs Up / Thumbs Down" rating system within the chat interface.
* Added a dynamic comment box that appears upon rating, allowing users to provide specific qualitative feedback (e.g., "Helpful advice" or "Incorrect information").
* Created a new database model ChatFeedback and API endpoint /feedback to securely store this data with timestamps and user IDs.

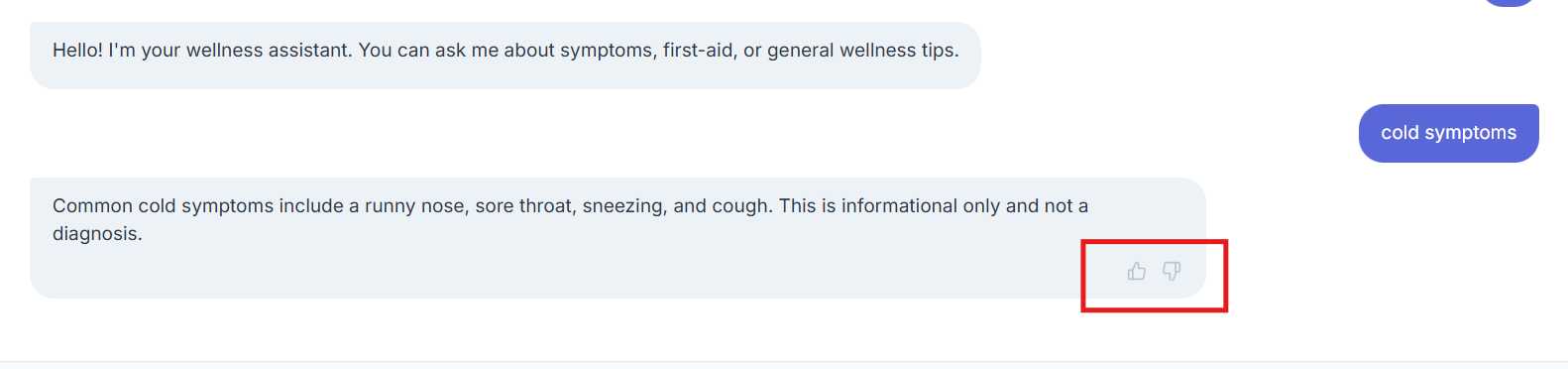


Fig : Thumbs Up / Thumbs Down Response

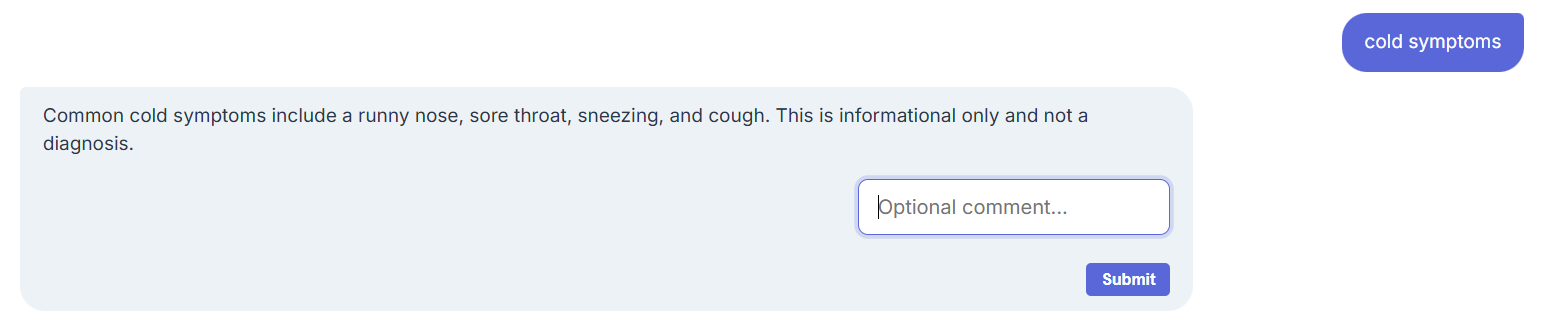
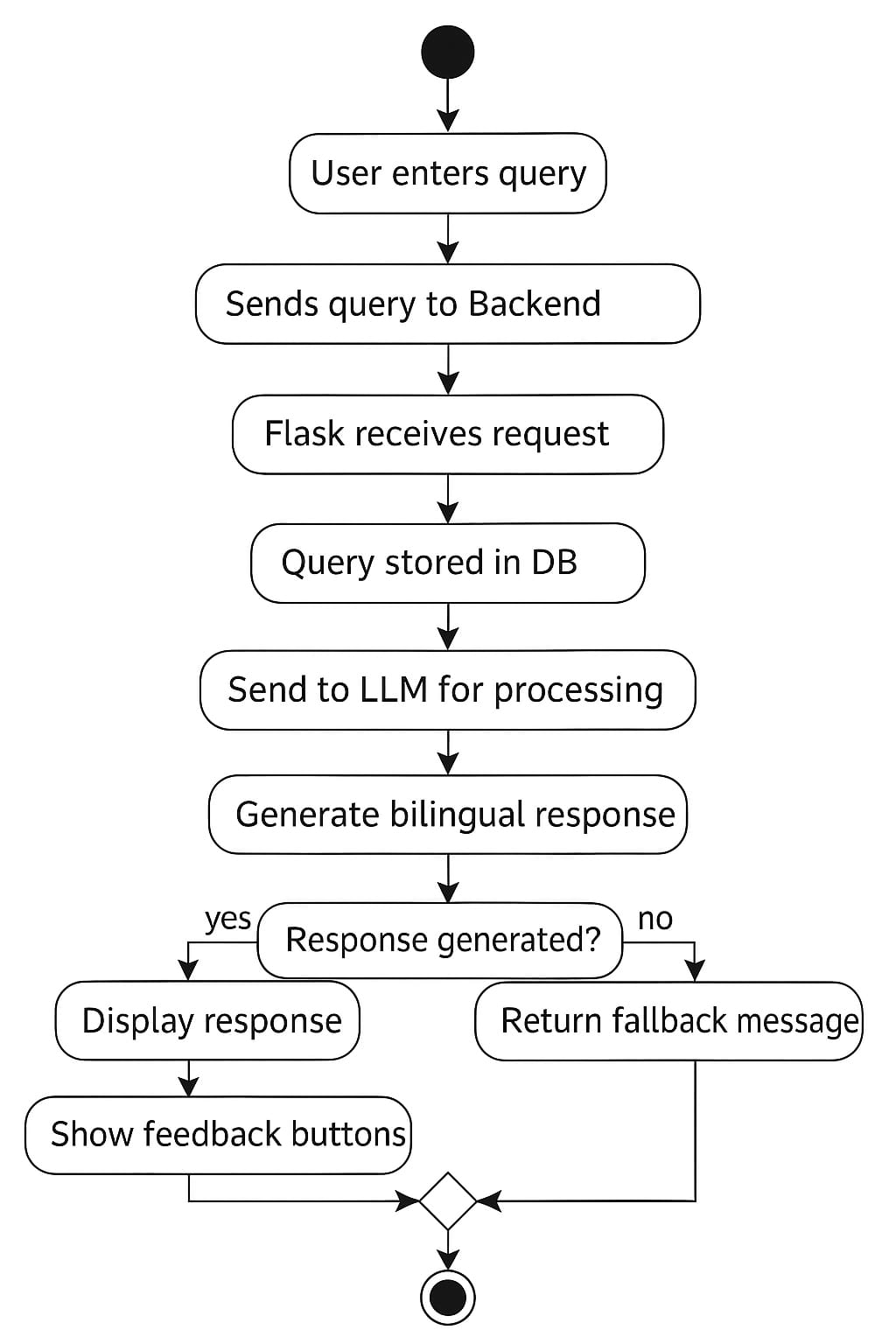
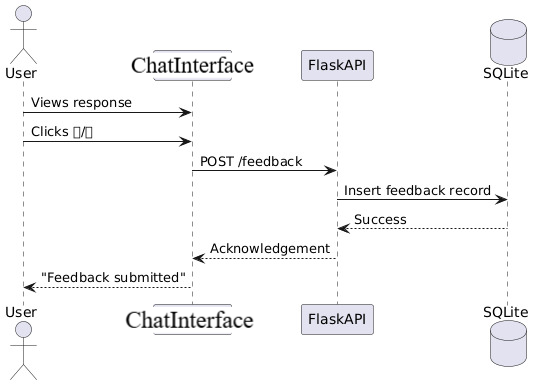


Fig : Comment box for Feedback

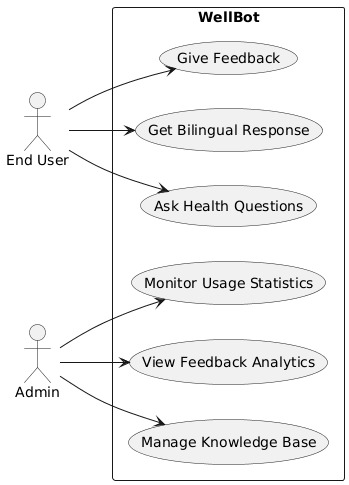
**chatbot activity diagram :-**



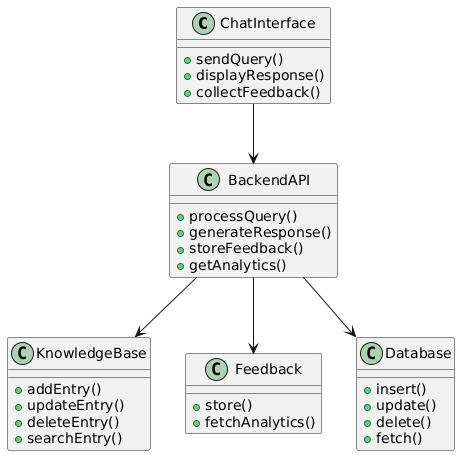
**Sequence class Diagram:-**



**Userflow Diagram : -**



**Class Diagram :-**



# 6. Testing & Validation

## 6.1 Unit Testing

We tested individual API endpoints using Invoke-WebRequest (PowerShell) to ensure they return correct HTTP status codes.

* **Login Success:** Returns 200 and a Cookie.
* **Login Fail:** Returns 401 and JSON error.
* **Chat without Token:** Returns 401/Redirects to Login.

## 6.2 Integration Testing

We verified the data flow across the entire stack. **Test Case 1: English Query**

* **Input:** "What are symptoms of flu?"
* **Process:** Flask -> Rasa -> Action Server (Lang=EN) -> DB.
* **Output:** "Flu symptoms include fever..." (Correct).

**Test Case 2: Hindi Query**

* **Input:** "फ्लू के लक्षण"
* **Process:** Flask -> Rasa (Hindi NLU) -> Action Server (Lang=HI) -> DB.
* **Output:** "फ्लू के लक्षण बुखार..." (Correct).

## 6.3 User Acceptance Testing (UAT)

We simulated a user session where the user registers, sets their language to Hindi, asks a question, rates the response as "Bad", and leaves a comment. We then logged in as Admin and verified that the negative rating and comment appeared in the Feedback Logs table.

# 7. Conclusion & Future Scope

## 7.1 Conclusion

The development and deployment of the Wellness Guide Bot represents a significant milestone in making reliable health information accessible and inclusive. This project successfully navigated the complexities of modern software engineering, integrating diverse technologies—from a robust Flask backend and secure SQLite database to the sophisticated natural language processing capabilities of the Rasa framework.

### Achievement of Core Objectives

The primary goal of creating a non-diagnostic, informational health assistant has been fully realized. By strictly adhering to ethical guardrails, the bot successfully navigates the fine line between helpful guidance and medical advice. The implementation of a mandatory disclaimer at the start of every session ensures that users are constantly aware of the bot's limitations, mitigating the risk of misinformation. This ethical framework is crucial in the health domain, where user safety is paramount.

### Impact of Multilingual Support

Perhaps the most impactful achievement is the successful implementation of bilingual support. By training the NLU model on parallel datasets for English and Hindi, the bot breaks down significant language barriers. It does not merely translate text; it understands intent in both languages. This "native understanding" capability allows Hindi-speaking users to interact naturally with the system, receiving responses that are culturally and contextually appropriate. This feature directly addresses the project's goal of inclusivity, ensuring that vital wellness information is not restricted to English speakers alone.

In summary, the Wellness Guide Bot stands as a comprehensive, secure, and scalable solution. It effectively bridges the gap between complex medical databases and the everyday user, providing a friendly, intelligent interface for health education.

## 7.2 Future Scope

While the current iteration of the Wellness Guide Bot is fully functional and deployed, the landscape of AI and healthcare is rapidly evolving. Several avenues for future development could significantly enhance the platform's capabilities, reach, and utility.

### 1. Voice-Enabled Interaction (Speech-to-Text & Text-to-Speech)

Currently, the bot relies entirely on text-based interaction. However, for many users—particularly the elderly, those with visual impairments, or those with lower literacy levels—typing can be a barrier. Integrating voice capabilities would be a transformative upgrade.

### 2. Personalized Health Insights & Predictive Analytics

The system currently collects user wellness data (steps, sleep, calories) but stores it passively. The next logical step is to actively utilize this data to provide personalized recommendations.

### 3. Integration with External Health APIs & Telemedicine

To move beyond static information, the bot could integrate with live external services to facilitate real-world healthcare actions.

### 4. Enhanced NLU with Large Language Models (LLMs)

While the current BERT-based model is effective for classified intents, integrating a Generative AI model (like GPT-4 or Llama 3) via a RAG (Retrieval-Augmented Generation) architecture could vastly improve the fluidity of conversation.

### 5. Expansion to Mobile Application

Currently, the Wellness Guide Bot is a responsive web application. Developing a dedicated mobile app (using React Native or Flutter) would provide a more persistent presence on a user's device.

# 8. References

1. Rasa Technologies Inc. "Rasa Open Source Documentation." <https://rasa.com/docs/rasa/>
2. Pallets Projects. "Flask Documentation (v2.3.x)." <https://flask.palletsprojects.com/>
3. Chart.js Contributors. "Chart.js: Simple yet flexible JavaScript charting for designers & developers." <https://www.chartjs.org/docs/latest/>
4. World Health Organization (WHO). "Health Topics." <https://www.who.int/health-topics/>
5. Flask-JWT-Extended. "Documentation for Flask-JWT-Extended." <https://flask-jwt-extended.readthedocs.io/en/stable/>
6. SQLAlchemy. "SQLAlchemy 1.4 Documentation." <https://docs.sqlalchemy.org/en/14/>
7. Hugging Face. "Transformers Documentation." <https://huggingface.co/docs/transformers/index>
8. Font Awesome. "Font Awesome Icon Gallery." <https://fontawesome.com/icons>
9. MDN Web Docs. "Web technology for developers (HTML, CSS, JavaScript)." <https://developer.mozilla.org/en-US/>