**Generative AI with IBM cloud**

**Project Documentation format**

**1. Introduction**

**• Project Title:** HealthAI: Intelligent Healthcare Assistant Using IBM Granite

**• Team Members:**

* Tumma Sri Naga Venkata Yasaswi (Team Leader)
* Manikanta Koduri
* Maila Rajkumar
* Mummidi Lakshmi Narayana

**2. Project Overview**

**• Purpose:**   
 HealthAI aims to unify symptom tracking, AI-powered health insights, home remedies, and reminders into a single, secure application that promotes proactive and personalized self-care.

**• Features:**

* Email, Google, and Facebook user registration + confirmation
* Symptom input (text/image) and top-3 condition suggestions
* Wellness tips and home remedy recommendations
* Medication and routine reminder system

**3. Architecture**

* **Frontend:**  
   Built with **Gradio** for rapid prototyping of chat and form-based interfaces, integrated with **IBM Watson Assistant** for conversational interactions
* **Backend:**  
   Implemented in **Python** (Flask/FastAPI) to handle API logic, session management, data processing, and ML model inference
* **Database:**
* **SQLite** (development) or **MySQL/PostgreSQL** (production) via SQLAlchemy
* Stores users, symptom logs, reminders, and session history

**4. Setup Instructions**

* **Prerequisites:**
* Python 3.9+
* Optional: SQLite or MySQL/PostgreSQL
* IBM Cloud credentials for Watson Assistant & Watsonx.ai
* **Installation:**

1. Clone the repository: git clone <repo-url>
2. Navigate into project folder
3. Create virtual environment: python -m venv venv && source venv/bin/activate
4. Install dependencies: pip install -r requirements.txt
5. Create .env file with required keys (IBM\_API\_KEY, DB\_URL, etc.)

**5. Folder Structure**

* **Client** (Gradio UI)
* app.py: main UI entrypoint
* components/: reusable UI elements (forms, chat windows)
* **Server** (Python backend)
* main.py or app.py: FastAPI server
* models/: health AI / ML models
* services/: IBM Watson integration
* db/: database models & migration scripts
* tests/: unit and integration tests

**6. Running the Application**

**Frontend:** python app.py

**Backend:** uvicorn main:app --reload

**7. API Documentation**

|  |  |  |  |
| --- | --- | --- | --- |
| **Endpoint** | **Method** | **Request Parameters** | **Response** |
| /register | POST | email, password, confirm | Success/status message |
| /confirm-email | GET | token | Account activation confirmation |
| /login | POST | email, password | Auth token + profile |
| /symptom | POST | text, optional image | Top-3 conditions + guidance |
| /wellness-tips | POST | age, gender, lifestyle | Personalized wellness suggestions |
| /reminders | POST/GET/DELETE | User reminder data | List or confirmation of reminders |

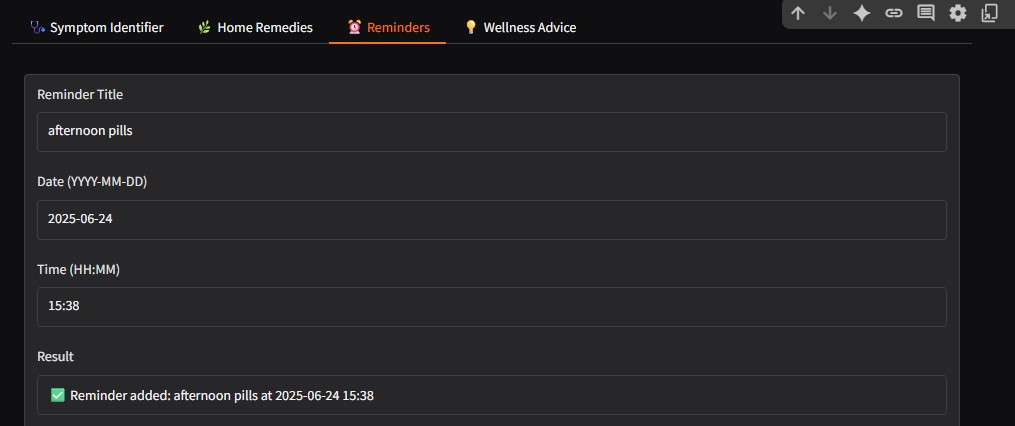
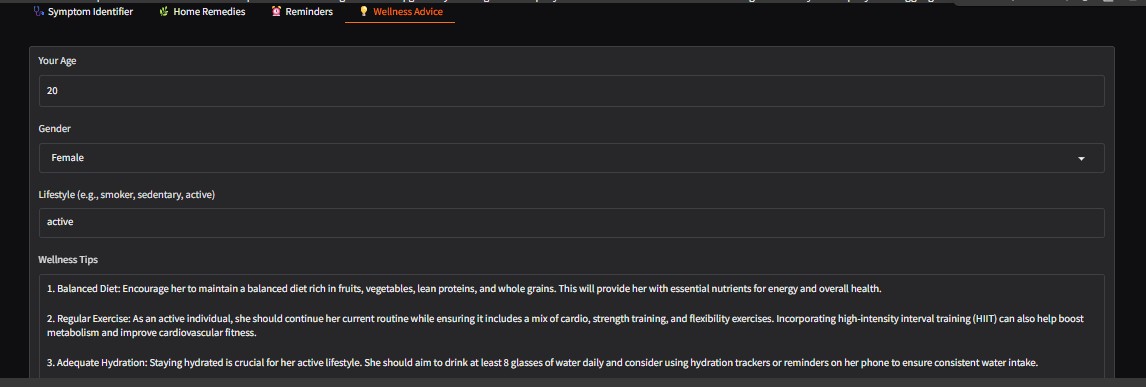
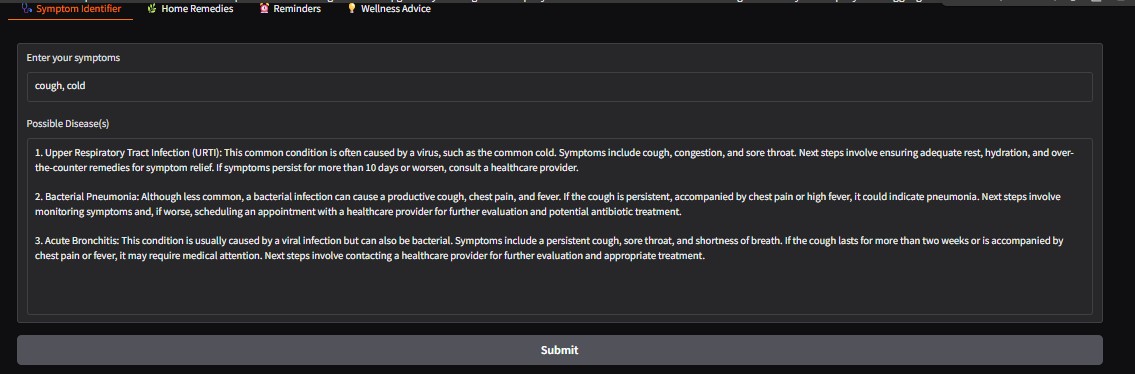
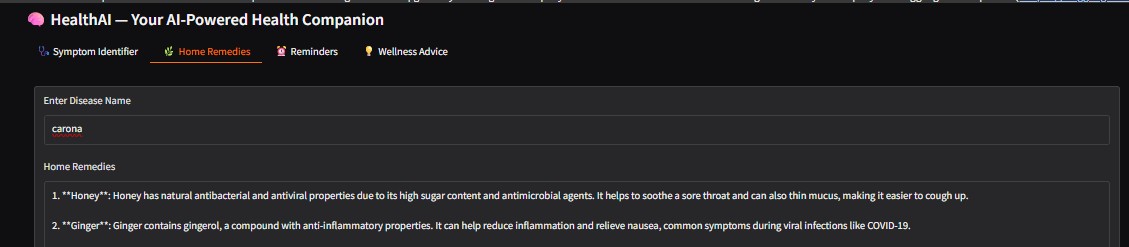
**8. Authentication**

* Uses **JWT tokens** issued after login
* Tokens validated via middleware on protected routes
* Expiry set to 1 hour; refresh enabled via refresh tokens for session persistence

**9. User Interface**

**Screenshots/Demos:**

* Chat-style symptom input and advice



**10. Testing**

**Strategy & Tools:**

* Unit Testing: pytest for logic in models and services
* Integration Testing: FastAPI’s TestClient for endpoints
* UI Testing: Manual tests via Gradio interface

**11. Screenshots or Demo**

Demo Video Link: https://drive.google.com/file/d/1k6RupLtBK1MQOdGoAQxhloFtpHUtD4-L/view?usp=drivesdk

**12. Known Issues**

* Intermittent Watson Assistant latency (1–2 sec delay)
* Image symptom uploads occasionally fail on large files (>5 MB)
* Reminder notifications may misfire if backend is down

**13. Future Enhancements**

* Multi-language support (e.g., Hindi, Spanish)
* Wearables integration for vitals (heart rate, temperature)
* Admin dashboard for customer care and analytics
* AI explanations improvements and model fine-tuning