**Health AI: Intelligent Healthcare Assistant Using IBM Granite**

**1. Introduction**

**• Project title:** health AI: Intelligent Health Care Assistant using IBM Granite

• **Team member**: Kavitha P

**• Team member:** Kavibharathi P

**• Team member:** Poojitha V

**• Team member**: Parimala S

**2. Project Overview**

• **Purpose:**

The purpose of health AI is to provide an AI-powered healthcare assistant that empowers individuals

and medical professionals with intelligent, accessible, and real-time medical insights. By leveraging

IBM Granite large language models, the assistant can analyze symptoms, suggest possible

Conditions, generate treatment recommendations, and provide home remedies. It serves as a

supportive tool for patients while reminding them to always consult certified doctors for final

diagnosis and treatment.

• **Features:**

• **Conversational Interface**: Natural language interaction with patients for symptom checking

and advice.

• **Disease Prediction**: Analyzes input symptoms and provides possible medical conditions with

cautionary disclaimers.

• **Treatment Plan Generator**: Generates personalized treatment suggestions considering age,

gender, and medical history.

• **Medication & Home Remedies**: Recommends general medications and simple remedies for

common conditions.

• **Health Education**: Provides preventive care tips and awareness about lifestyle-related health

issues.

• **Patient Feedback Loop**: Allows users to share feedback to improve the system over time.

• **Multimodal Input Support**: Accepts text-based symptom descriptions and structured health

data.

• **Gradio UI**: User-friendly interface for both patients and healthcare professionals.

**3. Architecture**

Frontend (Gradio): The frontend is built with Gradio, providing an intuitive and accessible interface

where users can enter symptoms, request treatment plans, and view generated recommendations.

The UI is designed for simplicity, accessibility, and reliability in medical contexts. Backend (Python

& Transformers): The backend integrates Hugging Face’s transformers library with IBM Granite

LLM models to generate context-aware medical responses. Disease prediction and treatment

planning modules are built as custom functions for response generation. LLM Integration (IBM

Granite): IBM Granite models are responsible for natural language understanding and generation.

They provide accurate, contextually relevant suggestions and medical insights, while being

designed with prompt engineering to ensure disclaimers and safe recommendations.

**4. Setup Instructions**

**Prerequisites:**

• Python 3.9 or later

• pip and virtual environment tools

• Hugging Face transformers library

• IBM Granite model access

• Internet connection for model downloads

**Installation Process:**

• Clone the repository

• Install dependencies from requirements.txt

• Run the Gradio interface script

• Access the application in your browser

• Enter symptoms or conditions to interact with the assistant

**5. Folder Structure**

• app/ – Contains backend logic including disease prediction and treatment modules.

• ui/ – Gradio interface for healthcare assistant.

• granite\_model.py – Handles communication with IBM Granite model.

• symptom\_checker.py – Implements disease prediction pipeline.

• treatment\_planner.py – Generates personalized treatment plans.

• report\_generator.py – Creates summaries for patient reports.

**6. Running the Application**

To start the project: â Run the Gradio application to launch the user interface. â Enter symptoms

to receive possible conditions and recommendations. â Provide details like age, gender, and

medical history for treatment planning. â Review the personalized treatment suggestions and

home remedies. â All interactions are real-time, leveraging IBM Granite to generate safe,context-aware outputs.The application can be run both in a **local development environment** and in a **production deployment environment**.

**Local Setup (Development Mode)**

1. **Start the Backend**
   * Activate your Python virtual environment:
   * source venv/bin/activate # Linux/Mac
   * venv\Scripts\activate # Windows
   * Run the Gradio application:
   * python app/main.py
2. **Access the Application**
   * Open your browser and navigate to:
   * http://localhost:7860
   * You will see the **healthAI interface** with tabs for:
     + Symptom Prediction
     + Treatment Planning
     + Feedback Submission
3. **Interacting with the Assistant**
   * Enter **symptoms** (e.g., "fever, sore throat") in the input box.
   * Provide **additional info** (age, gender, medical history).
   * Review generated **possible conditions** and **treatment recommendations**.
   * Provide **feedback** for system improvement.

**Production Deployment**

For real-world usage, you can deploy the application to a server or cloud platform:

* **Docker Deployment**
  + Build the Docker image:
  + docker build -t healthai-app .
  + Run the container:
  + docker run -p 8080:7860 healthai-app
  + Access via: http://<server-ip>:8080
* **Cloud Hosting Options**
  + **IBM Cloud / AWS / GCP** for scalable deployments.
  + Integrate **authentication** (JWT / OAuth2) before making it public.
  + Enable **logging & monitoring** for healthcare compliance.

**Running with API Mode**

The backend also exposes APIs that can be consumed by third-party apps:

Example call with curl:

curl -X POST http://localhost:8000/predict-symptoms \

-H "Content-Type: application/json" \

-d '{"symptoms": "headache, nausea"}'

Example response:

{

"conditions": ["Migraine", "Dehydration"],

"confidence\_scores": [0.82, 0.61]

}

**7. API Documentation**

 **Base URL**: http://localhost:8000/api (configurable for deployment)

1. **POST /predict-symptoms**

**Description**: Accepts patient symptoms and predicts possible conditions.

**Request Body**:

{

"symptoms": "headache, nausea, dizziness",

"age": 30,

"gender": "male"

}

**Response Example**:

{

"conditions": ["Migraine", "Vertigo"],

"confidence\_scores": [0.78, 0.65]

}

1. **POST /treatment-plan**

**Description**: Generates a personalized treatment plan.

**Request Body**:

{

"condition": "Migraine",

"medical\_history": ["hypertension"]

}

**Response Example**:

{

"medications": ["Paracetamol (500mg)", "Hydration"],

"home\_remedies": ["Rest in a quiet room", "Avoid bright lights"]

}

1. **POST /feedback**

**Description**: Collects patient feedback for improving recommendations.

**Request Body**:

{

"user\_id": "12345",

"feedback": "The suggestions were helpful" }

**8. Authentication**

Currently, the application is for demonstration purposes and runs in an open environment. For

secure deployments, authentication methods such as JWT tokens, API keys, or OAuth2 with IBM

Cloud credentials can be integrated. Role-based access (doctor, patient, admin) can be added in

future versions.

* Current state: Open environment (no authentication).
* Recommended strategies for secure deployment:

**JWT Tokens**: For stateless authentication across API requests.

**API Keys**: Assigned per user/organization for controlled access.

**OAuth2 Integration**: Seamless authentication using IBM Cloud or healthcare provider systems.

**Role-based Access Control (RBAC)**: Define roles (Patient, Doctor, Admin) with varying permissions.

**Encryption**: Secure patient data using TLS for transport and AES for storage.

**9. User Interface**

The interface is designed with simplicity and accessibility in mind. Features include:

• Tabs for Disease Prediction and Treatment Planning

• Text input boxes for symptom and condition entry

• Real-time output display for recommendations

• Clear disclaimers about medical advice limitations

• Easy navigation and patient-friendly design

**10. Testing**

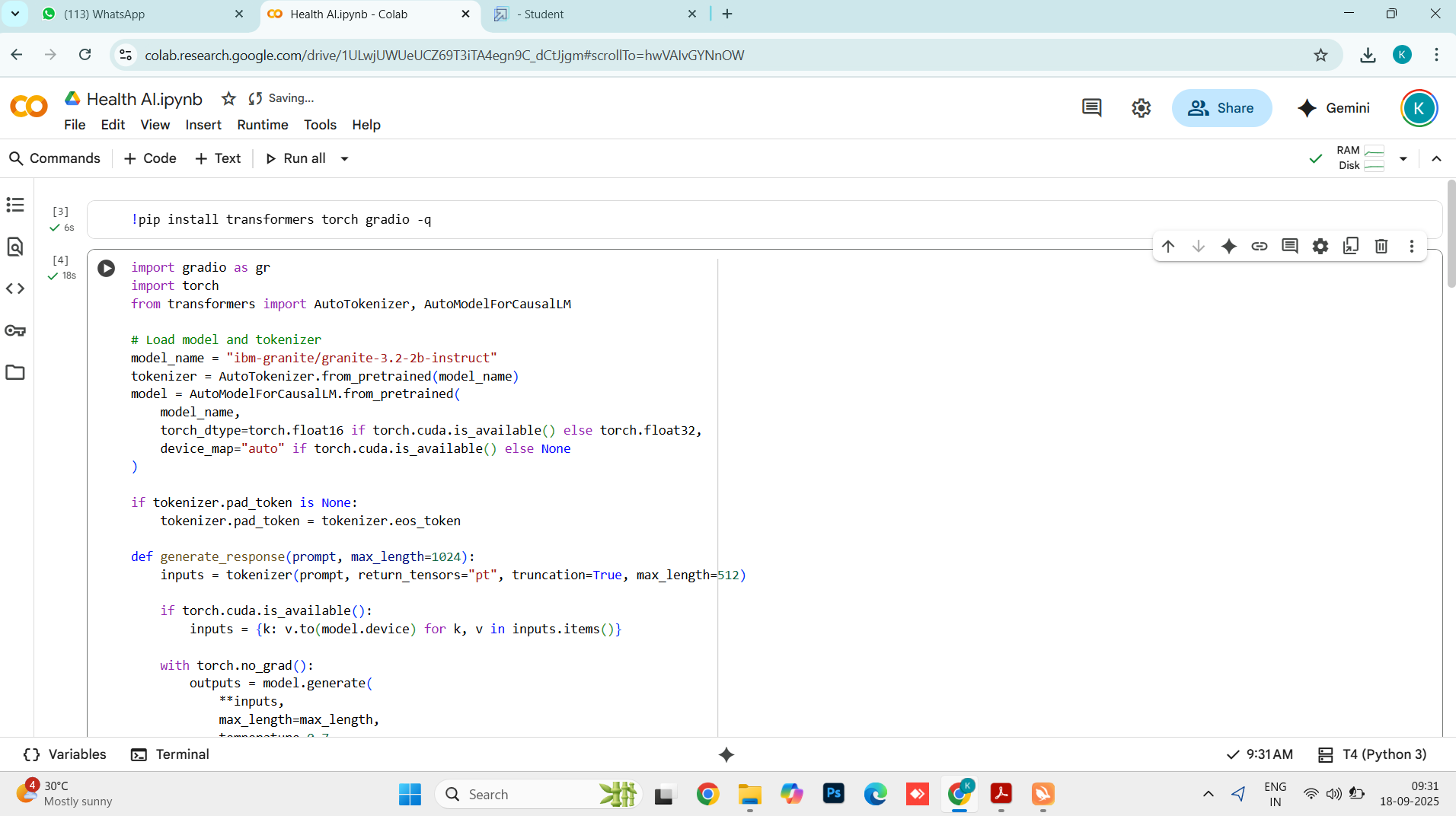
Testing was performed in multiple phases:

• Unit Testing: For prediction and treatment functions

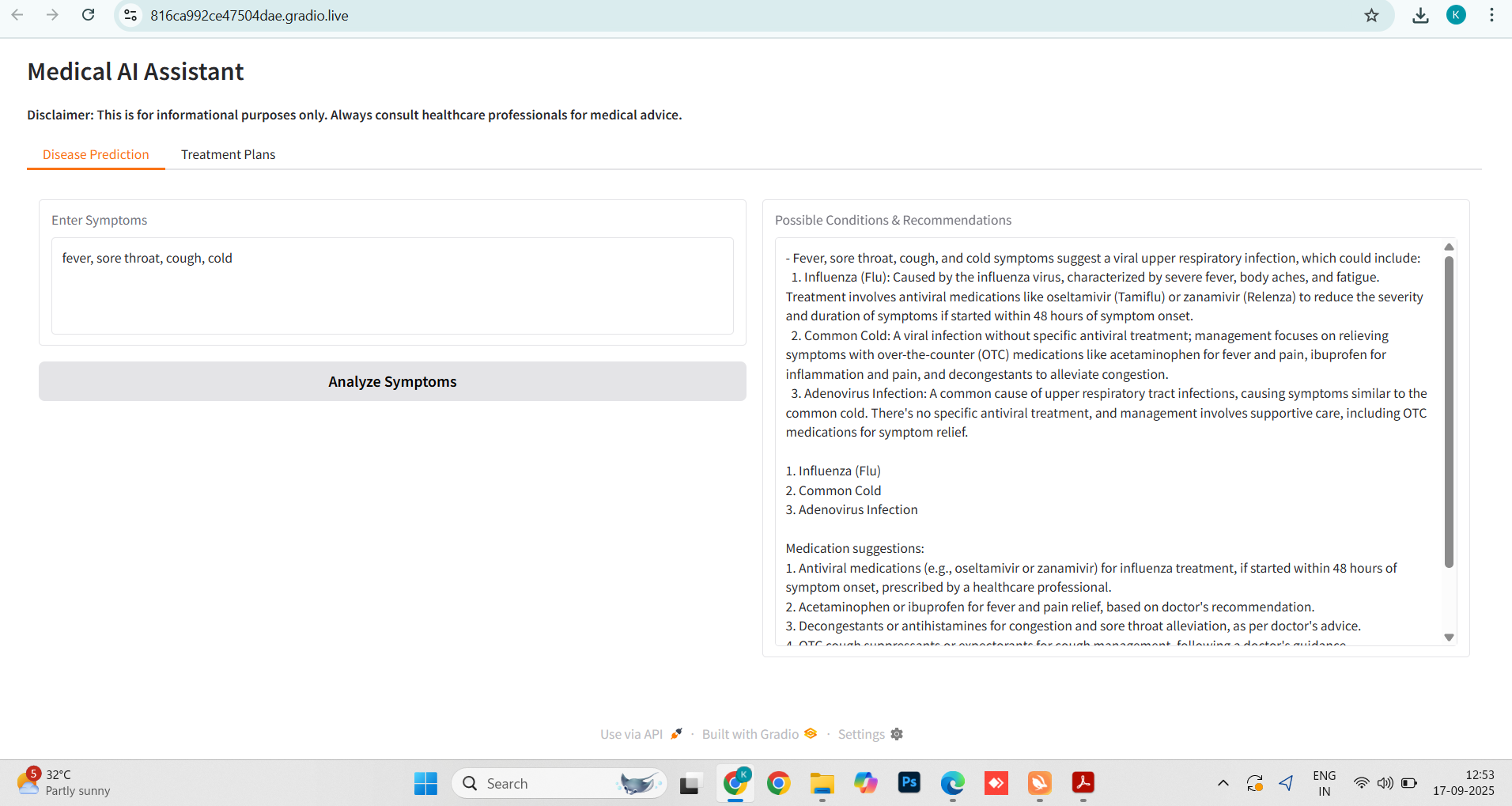
•API Testing: Using Postman and test scripts

• Manual Testing: For UI interactions and response Accuracy

• Edge Case Handling: Tested for empty inputs, invalid data, and large text entries each function was validated for reliability and safe response generation.

**11. Screenshots**



****

**12. Known Issues**

* Repetition in AI-generated responses.
* Requires stable internet for IBM Granite inference.
* Limited language support (currently only English).
* No offline mode available.
* Limited dataset for rare diseases → may reduce accuracy.

**13. Future Enhancements**

* **Wearable Device Integration**: Real-time vitals from smartwatches, fitness bands.
* **Multilingual Support**: Local language inputs/outputs for global accessibility.
* **Doctor Dashboard**: Secure portal for reviewing patient interactions and suggesting treatments.
* **EHR Integration**: Sync with hospital/clinic Electronic Health Records.
* **Enhanced Security**: HIPAA/GDPR compliance, encrypted storage for sensitive data.
* **Mobile App Version**: Lightweight Android/iOS app with offline caching.
* **Voice Assistant Integration**: Enable symptom input through voice commands.