health ai with ibm

project documentation

1.INTRODUCTION

* PROJECT TITLE: HealthAI :Intelligent Healthcare Assistant
* TEAM LEADER: Muhsina Nazeeha Banu.M.Z
* TEAM MEMBER: Lavanya.S
* TEAM MEMBER: Jenniyammal.J
* TEAM MEMBER: Ragavi.K
* TEAM MEMBER: Vaishali.S

2.PROJECT OVERVIEW

Purpose:

The purpose of the Health AI using IBM project is to leverage artificial intelligence and data analytics powered by IBM technologies to enhance healthcare delivery, improve patient outcomes, and optimize healthcare operations. By utilizing AI-driven insights and predictive analytics, the project aims to assist healthcare providers in making informed decisions, personalizing patient care, and streamlining clinical workflows.

Features:

* Conversational Interface:

- Key Point: Natural language interaction for patients and healthcare providers.

- Functionality: Allows users to ask questions, get updates on health conditions, and receive guidance in plain language.

* Medical Data Analysis:

- Key Point: Advanced data analytics for healthcare insights.

- Functionality: Analyzes patient data, medical histories, and treatment outcomes to provide actionable insights.

* Personalized Treatment Plans:

- Key Point: Tailored healthcare recommendations.

- Functionality: Generates personalized treatment suggestions based on patient data, medical history, and evidence-based guidelines.

* Predictive Analytics:

- Key Point: Forecasting patient outcomes and risks.

- Functionality: Estimates potential health risks and outcomes using historical data and real-time patient information.

* Anomaly Detection:

- Key Point: Early warning system for healthcare anomalies.

- Functionality: Identifies unusual patterns in patient data or clinical workflows to flag potential issues.

* Multimodal Input Support:

- Key Point: Flexible data handling for healthcare analytics.

- Functionality: Accepts text, PDFs, and CSVs for document analysis and predictive modeling.

* Streamlit or Gradio UI:

- Key Point: User-friendly interface for healthcare providers and patients.

- Functionality: Provides an intuitive dashboard for interacting with the Health AI system.

3.ARCHITECTURE

Frontend (Stream lit):

The frontend is built with Stream lit, offering an interactive web Ul with multiple pages including patient dashboards, data uploads, chat interface for patient queries, feedback forms, and report viewers. Navigation is handled through a sidebar using the stream lit-option-menu library. Each page is modularized for scalability.

Backend (Fast API):

Fast API serves as the backend REST framework that powers API endpoints for data processing, patient interactions, health tip generation, report creation, and vector embedding. It is optimized for asynchronous performance and easy Swagger integration.

LLM Integration (IBM Watsonx Granite):

Granite LLMs from IBM Watsonx are used for natural language understanding and generation. Prompts are carefully designed to generate personalized health tips, summaries, and reports based on patient data.

Vector Search (Pinecone):

Uploaded health documents are embedded using Sentence Transformers and stored in Pinecone. Semantic search is implemented using cosine similarity to allow users to search documents using natural language queries.

ML Modules (Predictive Analytics and Anomaly Detection):

Lightweight ML models are used for predictive analytics and anomaly detection in patient data using Scikit-learn. Time-series health data is parsed, modeled, and visualized using pandas and matplotlib.

4. Setup Instructions

Prerequisites:

* Prerequisites:

* Python programming knowledge.
* Gradio framework
* IBM granite model accesses(via hugging face)
* Google colab with T4 GPU
* Github account steps:

1. Access the Naan mudhalvan Smart Internz Portal.

2. Choose an IBM Granite model from Hugging face.

3. Run the application in Google colab with required libraries.

4. Upload final project files to Github

5. Folder Structure

* app/ Contains all Health Al backend logic including routers, models, and integration modules.
* app/api/Subdirectory for modular API routes like patient data, predictions, and document analysis.
* ui/Contains frontend components for Health Al pages, dashboards, and form interactions.
* smart\_dashboard.py
* Entry script launching the main Health Al Streamlit app.
* genai\_llm.py - Handles communication with IBM Watsonx Granite model for health predictions and summarization.
* document\_embedder.py
* Converts health documents to embeddings and stores in Pinecone.
* kpi\_forecaster.py Forecasts future health trends using regression.
* anomaly\_flagger.py Flags
* unusual values in uploaded health data.
* report\_generator.py Constructs Al-generated health reports.

6.Running the Application

1.Open Google colab and load the project notebook.

2.Install dependencies and configure runtime with GPU.

3.Run the notebook cells to start the Gradio app.

4.Access the provided link to interact with HEALTH AI.

7.API Documentation

Health Al available endpoints include:

* POST /patient-data Accepts patient query and responds with Al-generated health insights.
* POST /upload-docs Uploads and embeds health documents in Pinecone.
* GET /health-tips Returns health tips for selected topics like diet, exercise, or mental health.
* POST/submit-feedback
  + Stores user feedback for later review or analytics.

8.Authentication

Each endpoint is tested and documented in Swagger Ul for quick inspection and trial during development.

* Token-based authentication (JWT or API keys).
* OAuth2 with IBM Cloud credentials.
* Role-based access (admin, doctor, patient).
* Planned enhancements include user sessions and history tracking for Health Al.

9. Interface

Minimalist and functional Ul focusingon accessibility for non-technical users.It includes

* Sidebar with navigation.
* KPI visualizations with summary cards.
* Tabbed layouts for chat,health tips, and forecasting.
* Real-time form handling.

Health Al Project Documentation

10.Testing

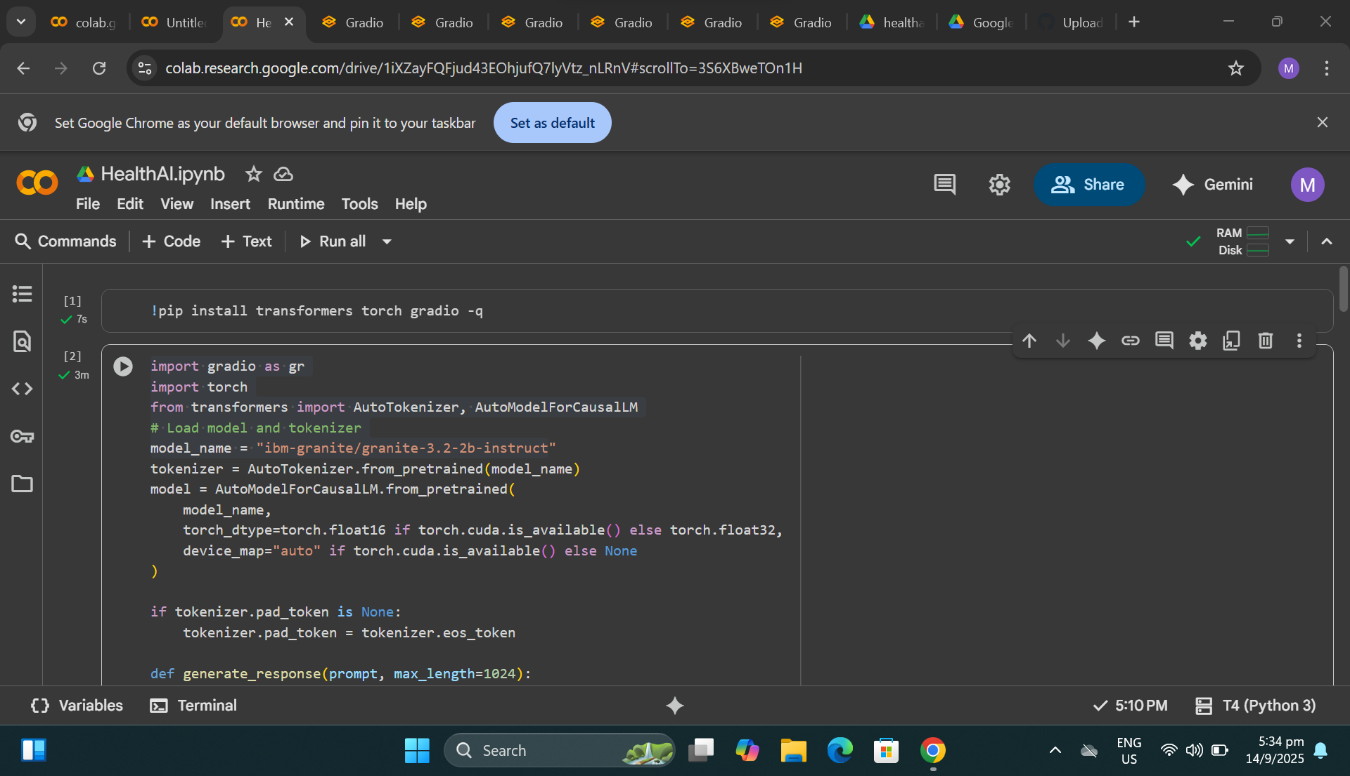
The Health Al project includes comprehensive testing to ensure the reliability and accuracy of its features. Testing includes:

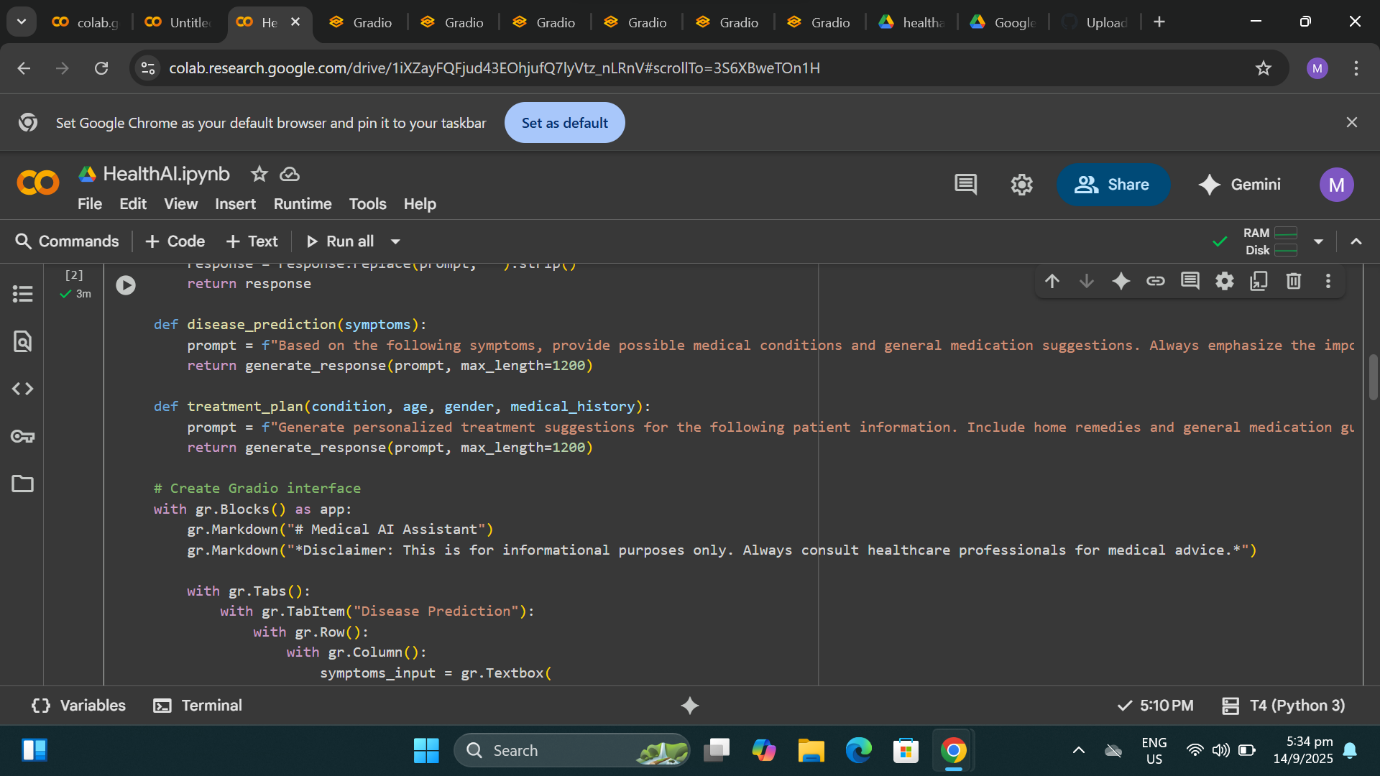
Unit Testing: Verifies the functionality of individual components, such as the disease prediction and treatment plan generation modules.

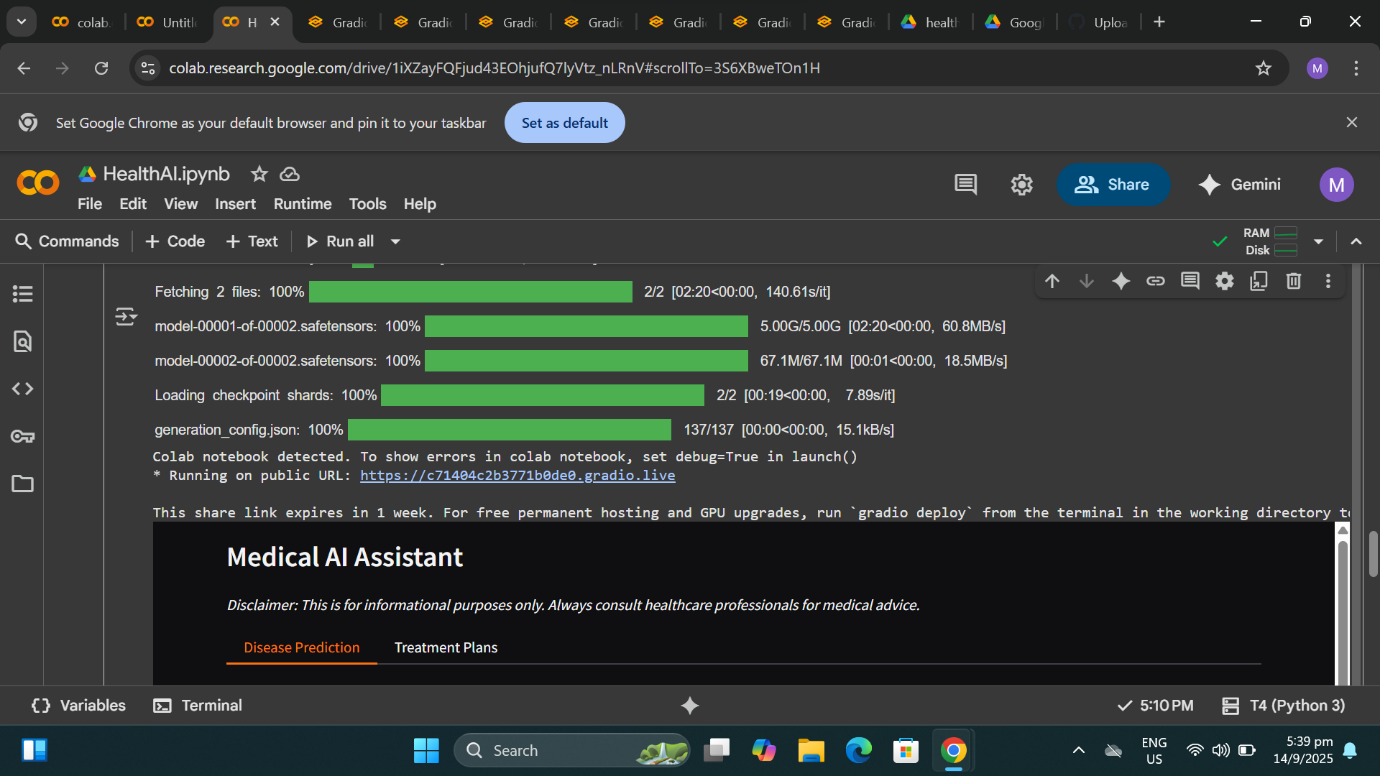
Integration Testing: Ensures that different components work together seamlessly, including the interaction between the frontend and backend.

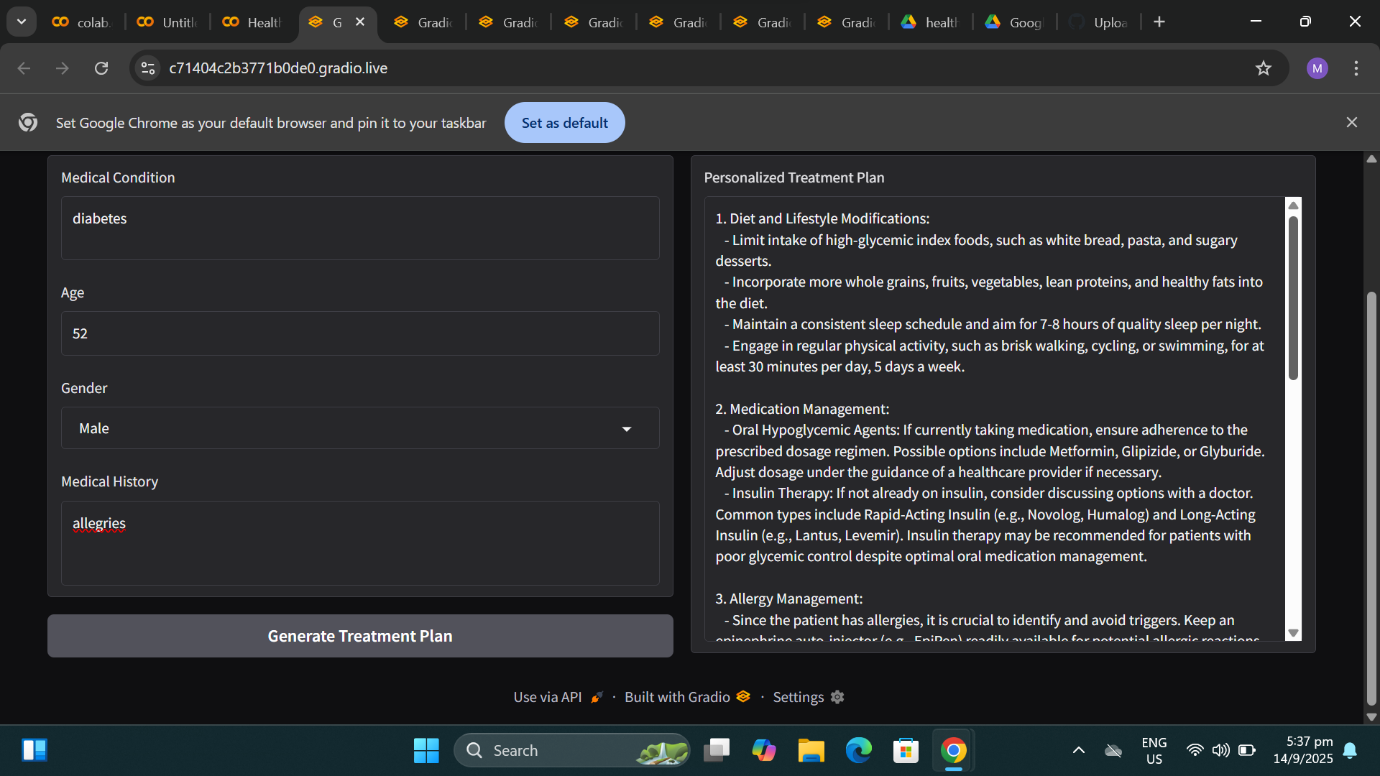
User Testing: Validates the user interface and experience, ensuring that the application is intuitive and easy to use.

11.screen shots









12.Known Issues

The current version of the Health Al project has the following known issues:

* Data Quality: The accuracy of the disease prediction and treatment plan generation modules depends on the quality of the training data. If the data is biased or incomplete, the results may not be accurate.
* Limited Domain Knowledge: The application's domain knowledge is limited to the data it was trained on. It may not perform well on rare or complex cases.
* User Input Validation: The application does not thoroughly validate user input, which could lead to errors or security vulnerabilities.

13.Future Enhancements

Future enhancements to the Health Al project may include:

* Improved Data Quality: Collecting more diverse and comprehensive data to improve the accuracy of the disease prediction and treatment plan generation modules.
* Expanded Domain Knowledge: Integrating additional domain knowledge and expertise to improve the application's performance on rare or complex cases.
* Enhanced User Input Validation: Implementing robust user input validation to prevent errors and security vulnerabilities.
* Integration with Electronic Health Records (EHRs): Integrating the application with EHR systems to provide seamless access to patient data and improve the accuracy of the disease prediction and treatment plan generation modules.
* Personalized Medicine: Developing personalized treatment plans based on individual patient characteristics, such as genetic profiles and medical histories.