PROJECT REPORT ON HEALTH AI:INTELLIGENT HEALTH CRAE ASSISTANT

HuggingFaceH4/zephyr-7b-beta

TEAM MEMBERS :

S.NO	NAME	REG.NO
01	G.Rajasekar	23053161802111082
02	J.Seethar	23053161802111090
03	G.selvaraj	23053161802111091
04	S.selvavinayagam	23053161802111092

ANNAI VAILANKANNI ARTS AND SCIENCE COLLEGE

OWNED AND MANAGED BY"DIOCESE OF TANJORE SOCIETY"(REGD.NO.S.8 OF 1958)

(AFFILIATED TO BHARATHIDASAN UNIVERSITY,TIRUCHIRAPPALLI-24)

(AFFILIATED TO BHARATHIDASAN UNIVERSITY,TIRUCHIRAPPALLI-24)

BISHOP SUNDARAM CAMPUS,PUDUKOTTAI ROAD,THANJAVUR–613007.

Index

1.Introduction

Health AI: Intelligent Healthcare Assistant is a Generative AI-powered solution designed to enhance healthcare support by providing intelligent, accessible, and personalized medical assistance. The project integrates **IBM's AI ecosystem** with advanced natural language models from **Hugging Face**, specifically leveraging the **HuggingFaceH4/zephyr-7b-beta** model.

The system is built to simulate human-like interactions, allowing users to ask health-related questions, receive AI-driven responses, and access reliable information in an efficient manner. By combining the scalability of IBM tools with the language capabilities of Zephyr-7B, Health AI acts as a bridge between patients and healthcare providers, improving communication and accessibility in the medical domain.

This project highlights how **Generative AI** can be responsibly applied in healthcare to:

1. Provide **instant medical guidance** and general health information.
2. Improve **patient engagement** through conversational support.
3. Reduce **workload for healthcare professionals** by addressing routine queries
4. Support **data-driven decision-making** while maintaining ethical AI standards.

Through its web-based platform (Project Link), Health AI demonstrates the potential of **AI-assisted healthcare**, paving the way for smarter, more inclusive, and technology-driven medical services.

2.Project Overview

Purpose:

The **Health AI: Intelligent Healthcare Assistant** aims to provide a user-friendly, AI-driven tool for addressing health-related queries, disease prediction, and treatment planning. Powered by the **HuggingFaceH4/zephyr-7b-beta** model and **IBM Granite**, the assistant utilizes generative AI to engage with users in natural language. It empowers both individuals and healthcare providers by offering personalized health advice, predicting potential diseases based on symptoms, and offering treatment recommendations. The tool bridges technology and healthcare, contributing to better-informed decisions and improving the overall patient experience.

Features:

- **Conversational Health Assistant**:
    - **Key Point**: Natural language interaction
    - **Functionality**: Users can chat with the assistant to get personalized health advice and answers to various medical questions in simple language.
- **Disease Prediction**:
    - **Key Point**: Predictive diagnostics
    - **Functionality**: Uses symptoms entered by users to predict potential diseases, helping early detection and informed decision-making.
- **Treatment Plans**:
    - **Key Point**: Personalized recommendations
    - **Functionality**: Based on disease predictions, the assistant suggests potential treatment plans and lifestyle changes to improve health.
- **Health Chat Model**:
    - **Key Point**: AI-powered conversation
    - **Functionality**: The model can generate realistic, contextually appropriate responses to medical questions, ensuring reliable advice based on the latest medical data.
- **Generative AI with IBM**:
    - **Key Point**: Advanced AI integration

    - **Functionality**: Combines the strengths of **HuggingFaceH4/zephyr-7b-beta** and **IBM Granite 3.3B** to provide high-quality, accurate health information and predictions.

- **Interactive Interface**:
    - **Key Point**: User-friendly design
    - **Functionality**: The app is deployed on **Hugging Face Spaces** and features an intuitive, interactive interface powered by **Streamlit** or **Gradio**, making it easy for users to interact with the assistant.
- **Anomaly Detection**:
    - **Key Point**: Early health indicators
    - **Functionality**: The model can detect unusual patterns in user inputs and suggest early health interventions or further consultation with healthcare providers.

Deployment:

The **Health AI** assistant is deployed on **Hugging Face Spaces**, allowing users to interact seamlessly through a web interface. It combines the power of **HuggingFaceH4/zephyr-7b-beta** for language generation and **IBM Granite** for deeper medical insights, making it a reliable tool for anyone seeking health information, from general queries to symptom-based predictions.

3. Architecture

The **Health AI: Intelligent Healthcare Assistant** is built using a modular and scalable architecture that integrates frontend interactivity, backend services, large language models (LLMs), and machine learning modules to deliver a seamless and intelligent healthcare experience.

🖥️ Frontend – Streamlit

- **Technology**: Streamlit, streamlit-option-menu
- **Description**:
  Provides an interactive and intuitive web UI with multiple pages:
  - Health chat interface
  - Disease prediction
  - File upload and report viewer
  - Feedback form and analytics
- **Features**:
  - Sidebar-based navigation
  - Modular page components for easy maintenance and scalability

---

⚙️ Backend – FastAPI

- **Technology**: FastAPI, Uvicorn, Pydantic
- **Description**:
  Acts as the core backend service that powers:
  - API endpoints for health chat, tip generation, feedback, and document handling
  - Fast, asynchronous request handling
  - Auto-generated Swagger API docs for testing and integration

---

🧠 LLM Integration – IBM Watsonx Granite

- **Model Used**: IBM Granite (and optionally HuggingFaceH4/zephyr-7b-beta)

- **Purpose**:
    - Natural language understanding and response generation
    - Summarization of medical documents or user inputs
    - Personalized treatment suggestions and health tips

🔍 Vector Search – Pinecone

- **Technology**: Sentence Transformers, Pinecone
- **Functionality**:
    - Embeds user-uploaded documents (e.g., PDFs)
    - Stores embeddings in Pinecone for fast semantic search
    - Allows users to ask questions and retrieve relevant sections from documents using natural language queries

---

📈 ML Modules – Forecasting & Anomaly Detection

- **Technology**: Scikit-learn, Pandas, Matplotlib
- **Modules**:
    - **Forecasting**

4. Setup Instructions

This section guides you through setting up and running the **Health AI** project on your local machine.

✅ Prerequisites

Before you begin, ensure you have the following installed:

- **Python 3.9 or later**
- **pip** and virtual environment tools (venv or virtualenv)
- **API Keys** for:
    - **IBM Watsonx**
    - **Pinecone**
- **Internet access** to use cloud-based services (LLMs, APIs, etc.)

🛠️ Installation Process

1. **Clone the Repository**
2. git clone https://github.com/your-username/health-ai.git
3. cd health-ai
4. **Create and Activate a Virtual Environment**
5. python -m venv venv
6. source venv/bin/activate # On Windows: venv\Scripts\activate
7. **Install Dependencies**
8. pip install -r requirements.txt
9. **Configure Environment Variables**
    - Create a .env file in the root directory:
    - IBM_API_KEY=your_ibm_api_key
    - PINECONE_API_KEY=your_pinecone_api_key
    - PINECONE_ENV=your_pinecone_environment
    - (Add any other necessary credentials or config values.)
10. **Start the Backend (FastAPI)**

11. uvicorn backend.main:app --reload
12. **Launch the Frontend (Streamlit)**
    In a new terminal window:
13. streamlit run app.py
14. **Upload Data and Use the App**
    - Navigate through the sidebar to access features like:
        - Health Chat
        - Disease Prediction
        - File Upload (PDF/CSV)
        - Report Summarization
        - Feedback & Analytics

5. Folder Structure

The project is organized to separate frontend, backend, and AI/ML modules for better modularity, scalability, and maintainability.

```
health-ai/
│
├── app/                        # Backend logic (FastAPI)
│   ├── api/                    # Modular API routes
│   │   ├── chat.py             # Chat endpoint using IBM Granite
│   │   ├── feedback.py         # Feedback form handler
│   │   ├── report.py           # Report generation endpoint
│   │   └── vectorization.py    # Document embedding and search
│   ├── models/                 # Pydantic models and schemas
│   └── utils/                  # Helper functions for processing
│
├── ui/                         # Frontend components (Streamlit)
│   ├── home_page.py            # Landing page
│   ├── chat_interface.py       # Health chat interface
│   ├── prediction_page.py      # Disease prediction UI
│   ├── file_upload.py          # Document uploader
│   └── feedback_form.py        # User feedback form
│
├── smart_dashboard.py          # Streamlit app entry point
├── granite_llm.py              # Communicates with IBM Watsonx Granite LLM
├── document_embedder.py        # Embeds documents and stores vectors in Pinecone
├── kpi_file_forecaster.py      # Forecasts KPI trends (e.g., water, energy)
├── anomaly_file_checker.py     # Detects anomalies in uploaded KPI data
├── report_generator.py         # Generates AI-driven reports based on inputs
├── requirements.txt            # Python dependencies
└── .env                        # Environment variables (API keys, config)
```

📁 Highlights of Key Files & Folders

- **app/** – Houses all FastAPI backend logic, organized into API routes and utility modules.
- **ui/** – Contains individual Streamlit pages for modular and clean UI design.
- **smart_dashboard.py** – Main entry point to run the Streamlit interface.
- **granite_llm.py** – Manages interactions with IBM Watsonx Granite for summarization and natural language responses.
- **document_embedder.py** – Handles document-to-vector conversion and indexing in Pinecone.
- **kpi_file_forecaster.py** – Implements forecasting models for predicting resource usage trends.
- **anomaly_file_checker.py** – Identifies unusual or critical patterns in user-uploaded KPI data.
- **report_generator.py** – Uses LLMs to create structured sustainability or health reports.

6. Running the Application

This section explains how to launch and use the **Health AI** application, including its frontend and backend components.

✅ Step-by-Step Instructions

1. **Start the Backend (FastAPI)**
   Launch the FastAPI server to expose backend endpoints:
2. uvicorn app.main:app --reload
    - Access the auto-generated **API docs** at:
      http://localhost:8000/docs
3. **Start the Frontend (Streamlit)**
   In a **new terminal window**, launch the user interface:
4. streamlit run smart_dashboard.py
    - The dashboard will open in your browser (e.g., http://localhost:8501)
5. **Explore the App via Sidebar Navigation**
   Use the **Streamlit sidebar** (powered by streamlit-option-menu) to navigate through:
    - Health Chat
    - Disease Prediction
    - File Upload (PDF/CSV)
    - Report Viewer
    - Feedback Form
    - Health Analytics
6. **Interact with Modules**
    - **Upload Documents**: PDFs or CSVs for analysis, forecasting, and summarization
    - **Chat with AI**: Ask medical questions, get treatment suggestions, and receive health tips
    - **View Outputs**: Receive reports, predictions, summaries, or anomaly alerts in real time

---

🖥️ Frontend – Streamlit

- Built using **Streamlit** with modular pages for scalability.
- Includes dashboards, health chat, document upload, and feedback features.
- Easy navigation with streamlit-option-menu.
- Fully interactive and user-friendly UI.

⚙️ Backend – FastAPI

- Built with **FastAPI** for high-performance asynchronous API services.
- Handles:
    - Health chat via LLMs (IBM Watsonx / Zephyr 7B)
    - File processing & embedding
    - Forecasting and anomaly detection
    - Report and tip generation
- Integrated Swagger UI for easy testing of API endpoints.

🔄 Real-time Interaction

All user interactions—from file uploads to AI chat—are processed through backend APIs and reflected immediately on the frontend, enabling a seamless and dynamic healthcare experience.

7.API Documentation

1. POST /chat/ask

- **Description**: Accepts a user query and responds with an AI-generated message.
- **Request**:
  {"query": "What are the symptoms of diabetes?"}
- **Response**:
  {"response": "Common symptoms include increased thirst, frequent urination, and unexplained weight loss."}

2. POST /upload-doc

- **Description**: Uploads and embeds documents into Pinecone for semantic search.
- **Request**:
  {"document": "<binary data or raw text>"}
- **Response**:

  {"status": "success",

  "document_id": "12345"}

3. GET /search-docs

- **Description**: Returns semantically similar documents to the input query.
- **Request**: /search-docs?query=Diabetes management guidelines
- **Response**:
- { "results": [{
  "title": "Diabetes Management Protocol",
  "url": "https://example.com/diabetes-protocol" }]}

4. GET /get-eco-tips

- **Description**: Provides sustainability tips for topics like energy, water, or waste.
- **Request**: /get-eco-tips?topic=energy
- **Response**:

  { "tips": [
  "Switch off equipment when not in use.",
  "Install energy-efficient lighting."]}

5. POST /submit-feedback

- **Description**: Stores citizen feedback for review or analytics.
- **Request**:

  { "user_id": "67890",

  "feedback": "The AI assistant is helpful, but I would like more info on mental health topics."}

Response:

{ "status": "feedback received"}

8.Authentication

This version of the project runs in an **open environment** for demonstration purposes. However, secure deployments can integrate various authentication mechanisms to ensure safety and role-based access:

- **Token-based Authentication**:
  - Use **JWT (JSON Web Tokens)** or **API keys** for secure access to the APIs.
- **OAuth2 Integration**:
  - Leverage **IBM Cloud credentials** for secure and seamless authentication across IBM services.
- **Role-based Access Control (RBAC)**:
  - **Admin**: Full access to all endpoints and administrative features.
  - **Citizen**: Limited access to healthcare-related queries and tips.
  - **Researcher**: Access to policy documents, feedback data, and research-related endpoints.
- **Planned Enhancements**:
  - **User Sessions**: Implement session management to retain user states.
  - **History Tracking**: Log and track user interactions to personalize experiences.

9.User Interface & Testing

The interface is **minimalist** and **functional**, designed to prioritize **accessibility** for non-technical users. Key features include:

- **Sidebar with Navigation**: Easy access to all sections and features.
- **KPI Visualizations**: Summary cards to highlight key performance indicators.
- **Tabbed Layouts**: Quick access to chat, eco tips, and forecasting sections.
- **Real-time Form Handling**: Dynamic forms that update instantly based on user input.
- **PDF Report Download**: Allows users to download reports in PDF format for offline access.

The design focuses on **clarity, speed**, and **user guidance**, with help texts and intuitive flows to ensure a seamless user experience.

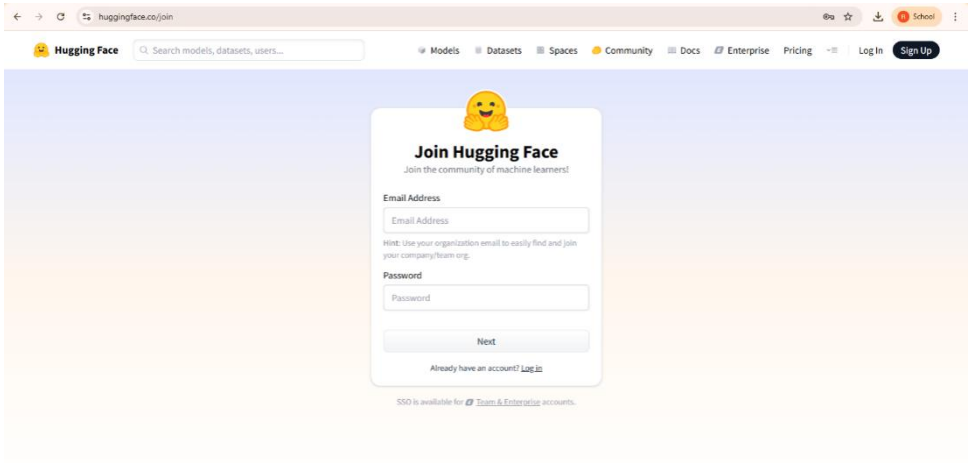
Testing:

Testing was performed in multiple phases to ensure reliability and performance:

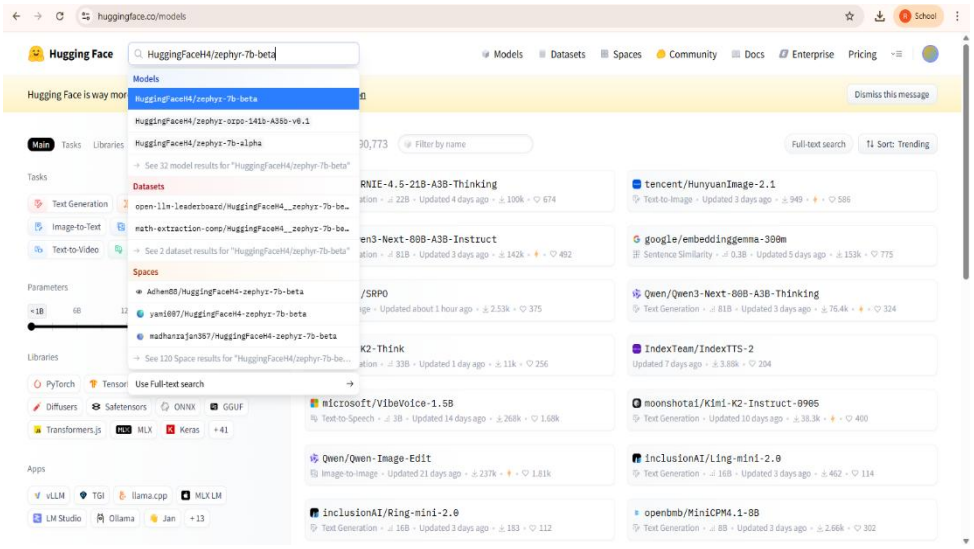
- **Unit Testing**: Validated prompt engineering functions and utility scripts for accuracy.
- **API Testing**: Conducted using **Swagger UI**, **Postman**, and custom test scripts for endpoint verification.
- **Manual Testing**: Tested file uploads, chat responses, and output consistency to ensure the system works as expected.
- **Edge Case Handling**: Ensured proper handling of malformed inputs, large files, and invalid API keys.

Each function was validated to ensure **reliability** in both **offline** and **API-connected** modes.

10.Screen shots

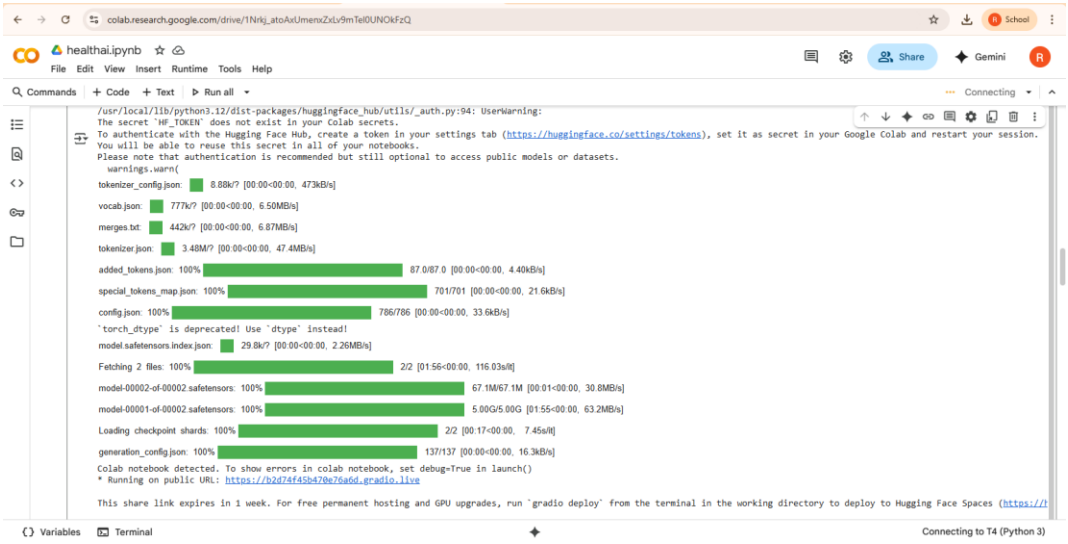


Creating account on hugging face:

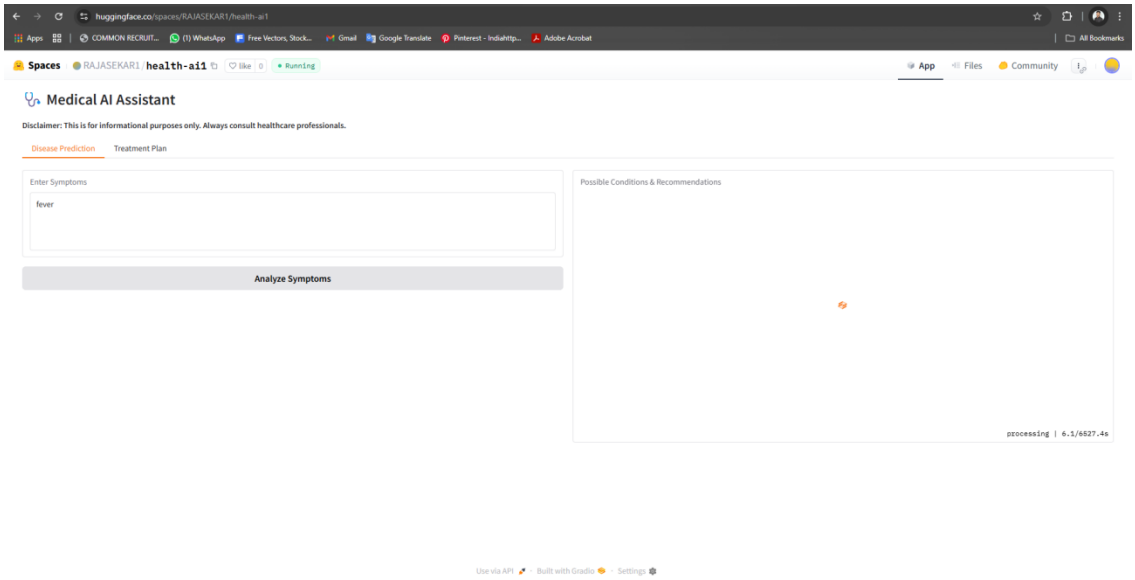


Selecting model:

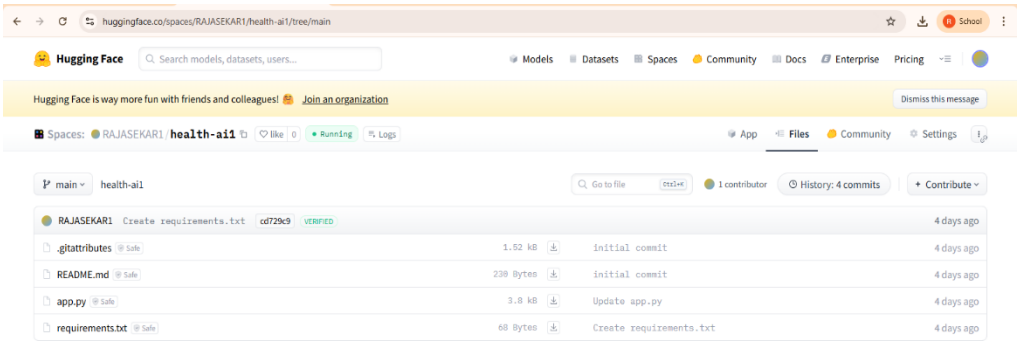
Uploading code for app.py ,recquirement.txt file to new space:



runapp.pycode in google collab and downloading healthai.ipnyb file:



Testing code in hugging face and the result will be displayed:



11.Future Enhancements

The project has several planned improvements to enhance functionality and user experience:

- **User Sessions & History Tracking**:
  Implementing user session management and tracking interaction history to personalize the experience and retain state across sessions.
- **Advanced Authentication Features**:
  Further integration of **OAuth2** and **role-based access control** to support more granular user permissions and secure access.
- **Real-time Analytics & Forecasting**:
  Expanding the system's ability to provide **real-time analytics**, trends, and predictions, especially related to healthcare and sustainability.
- **Multi-language Support**:
  Adding support for multiple languages to make the assistant accessible to a wider, global audience.
- **Integration with Third-party Systems**:
  Future integration with **Electronic Health Records (EHR)**, **patient management systems**, and other healthcare platforms for seamless data exchange.
- **AI Model Enhancements**:
  Fine-tuning the AI model for even more accurate and contextually relevant responses, especially in specialized medical domains.