Health AI: Intelligent Healthcare Assistant

Team Member: L Srinithi

Team Member: B Ishwarya

Team Member: K Muthulakshimi

Team Member: G Birundha

1. Introduction

Project Title: Health AI: Intelligent Healthcare Assistant

2. Project Overview

Purpose: The purpose of Health AI is to revolutionize healthcare delivery by providing an intelligent, patient-centered digital assistant. By leveraging artificial intelligence, natural language processing, and predictive analytics, the assistant helps patients manage their health, access medical information, schedule appointments, and receive personalized wellness guidance. For doctors and hospitals, it serves as a clinical decision-support partner, offering insights from medical records, highlighting anomalies, and streamlining workflows.

Ultimately, Health AI bridges the gap between patients, caregivers, and healthcare providers—enabling more accessible, efficient, and proactive healthcare.

Features:

- Conversational Interface: Patients can ask health-related questions, receive explanations, and get reminders in plain language.

- Medical Summarization: Converts lengthy patient histories, lab reports, and prescriptions into concise, actionable summaries.

- Symptom Checker & Risk Assessment: Uses AI to analyze patient inputs and suggest possible conditions or next steps.

- Appointment & Medication Management: Schedules appointments, sends reminders, and tracks medication adherence.

- Health Forecasting: Forecasts patient health trends such as blood pressure, sugar levels, or recovery timelines using historical and real-time data.

- Anomaly Detection in Vitals: Detects unusual readings in vitals or lab reports and alerts patients/doctors.

- Multimodal Input Support: Accepts text, PDFs, lab reports, and wearable device data for analysis.

- User-Friendly Dashboard (Streamlit/Gradio UI): Provides an intuitive dashboard for both patients and healthcare professionals.

3. Architecture

Frontend (Streamlit/Gradio): Provides an interactive web/mobile UI with dashboards, patient chat, symptom checker, appointment manager, and health trackers.

Backend (FastAPI): Handles API endpoints for health queries, medical record analysis, appointment booking, and anomaly detection.

LLM Integration (Watsonx / GPT models): Powers medical summarization, patient interaction, and explanation of complex medical terms.

Vector Search (Pinecone / FAISS): Enables semantic search over medical literature, patient records, and guidelines.

ML Modules (Forecasting & Anomaly Detection): Uses time-series analysis and ML models to forecast patient trends and detect anomalies in vitals.

4. Setup Instructions

Prerequisites:

- Python 3.9+

- pip and virtual environment tools

- API keys (Watsonx, Pinecone/FAISS, healthcare APIs)

- Access to medical datasets

Installation Process:

- Clone repository

- Install dependencies (requirements.txt)

- Configure .env with credentials

- Run FastAPI backend server

- Launch Streamlit frontend

- Upload medical data or interact with assistant

5. Folder Structure

app/ – FastAPI backend logic

app/api/ – API routes: chat, records, appointments, anomaly detection

ui/ – Streamlit/Gradio frontend components

health_dashboard.py – Main entry script for launching UI

llm_module.py – LLM communication for chat & summaries

record_embedder.py – Embeds patient records for semantic search

forecasting.py – Predicts patient health metrics

anomaly_checker.py – Flags abnormal health readings

report_generator.py – Creates AI-generated health summaries/reports

6. Running the Application

1. Start FastAPI backend.

2. Launch Streamlit dashboard.

3. Navigate via sidebar (chat, symptom checker, records, forecasts).

4. Upload medical data or query assistant.

5. View health reports, alerts, and predictions in real-time.

7. API Documentation

POST /chat/ask – Patient query → AI-generated health response

POST /upload-records – Upload and embed patient/medical documents

GET /search-records – Search within patient history or medical guidelines

GET /get-health-tips – Provides personalized wellness guidance

POST /book-appointment – Books or updates appointments

POST /submit-feedback – Collects patient feedback

8. Authentication

- Token-based authentication (JWT/API keys)

- OAuth2 integration with hospital systems

- Role-based access (doctor, patient, admin)

- Planned: patient history tracking and secure EHR integration

9. User Interface

- Sidebar navigation (Chat, Records, Appointments, Forecasts)

- KPI cards for vitals (BP, sugar, heart rate, etc.)

- Real-time alerts for anomalies

- Medical record upload & summarization

- Report download (PDF/CSV)

- Accessible, patient-friendly design

10. Testing

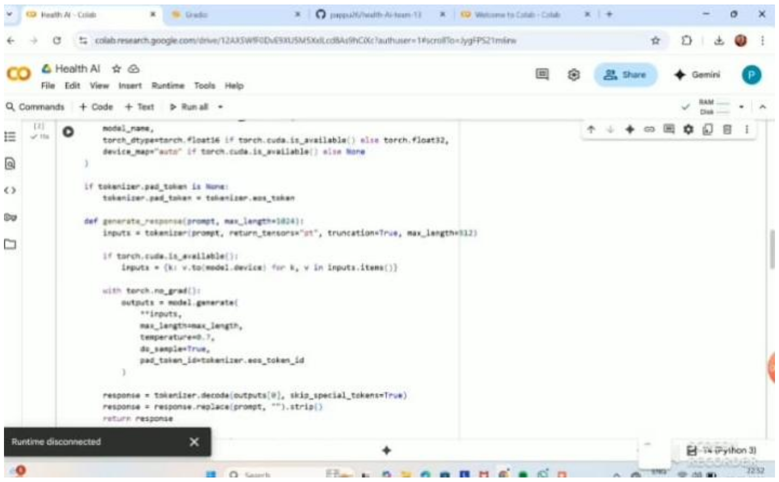
- Unit Testing: AI prompt accuracy, medical data parsing

- API Testing: Via Swagger/Postman

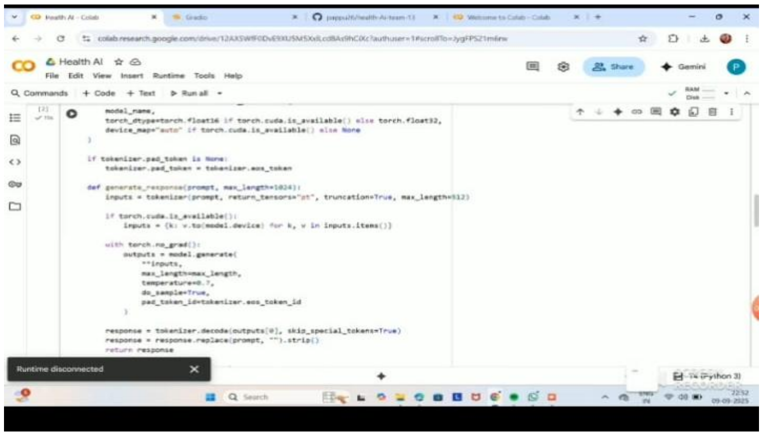
- Manual Testing: For file uploads, chat, and anomaly detection

- Edge Case Handling: Misreported symptoms, missing data, large medical records

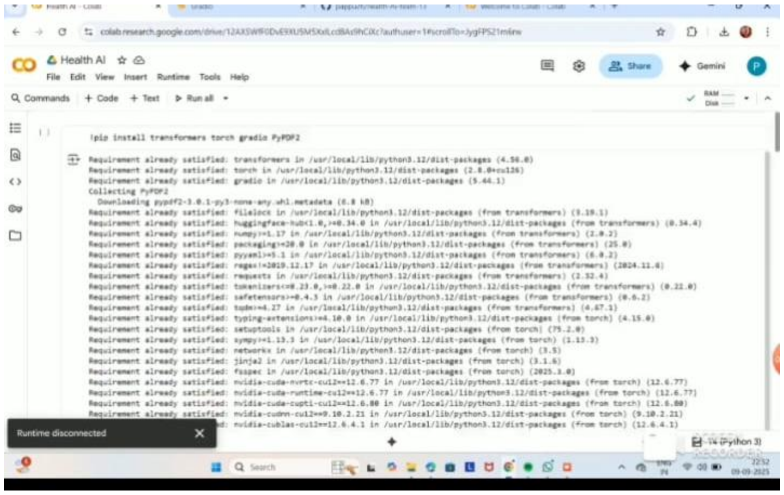
11. Screenshots



```
        model_name,
        torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
        device_map="auto" if torch.cuda.is_available() else None
    )

    if tokenizer.pad_token is None:
        tokenizer.pad_token = tokenizer.eos_token

    def generate_response(prompt, max_length=1024):
        inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

        if torch.cuda.is_available():
            inputs = {k: v.to(model.device) for k, v in inputs.items()}

        with torch.no_grad():
            outputs = model.generate(
                **inputs,
                max_length=max_length,
                temperature=0.7,
                do_sample=True,
                pad_token_id=tokenizer.eos_token_id
            )

        response = tokenizer.decode(outputs[0], skip_special_tokens=True)
        response = response.replace(prompt, "").strip()
        return response
```



```
        model_name,
        torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
        device_map="auto" if torch.cuda.is_available() else None
    )

    if tokenizer.pad_token is None:
        tokenizer.pad_token = tokenizer.eos_token

    def generate_response(prompt, max_length=1024):
        inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

        if torch.cuda.is_available():
            inputs = {k: v.to(model.device) for k, v in inputs.items()}

        with torch.no_grad():
            outputs = model.generate(
                **inputs,
                max_length=max_length,
                temperature=0.7,
                do_sample=True,
                pad_token_id=tokenizer.eos_token_id
            )

        response = tokenizer.decode(outputs[0], skip_special_tokens=True)
        response = response.replace(prompt, "").strip()
        return response
```



```
!pip install transformers torch gradio PyPDF2
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.56.0)
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.8.0+cu126)
Requirement already satisfied: gradio in /usr/local/lib/python3.12/dist-packages (5.44.1)
Collecting PyPDF2
  Downloading pypdf2-3.0.1-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (3.19.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.34.4)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.22.0)
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.6.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (from torch) (4.15.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
                            d: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch) (12.6.4.1)
```

12. Known Issues

- Limited accuracy without integration with real EHR systems

- Requires strong data privacy and HIPAA compliance

- Symptom checker not a substitute for medical diagnosis

13. Future Enhancements

- Integration with wearable IoT health devices

- Multilingual medical assistant support

- Voice-based conversational assistant

- Full EHR/EMR system integration

- Advanced predictive models for chronic disease management