

# PROJECT DOCUMENTATION

## 1. Introduction

**Project title:** Health AI : Intelligent Healthcare Assistant

**Team Leader** : Akshaya S

**Team Members:** Vishnupriya R

**Team Members:** Yogalalshmi A

**Team Members:** Bhuvneshwari S

**Team Members:** Gobika S

## 2. Project overview

### Purpose

- The Health AI Assistant is designed to revolutionize the healthcare ecosystem by combining the power of AI, medical knowledge, and patient-centered design. It supports patients, doctors, and researchers by enabling faster diagnosis, personalized health guidance, and streamlined access to medical data.
- By integrating IBM Watsonx AI models, real-time health monitoring, and medical records analysis, the assistant helps in:
- Improving early disease detection.
- Recommending preventive measures.
- Assisting doctors with medical insights and treatment suggestions.
- Enhancing patient engagement with accessible, AI-driven health guidance.

## Key Features

- **Conversational Health Chat:** Natural language interaction for patients to ask medical queries.
- **Symptom Checker:** AI-powered differential diagnosis support.
- **Health Report Summarization:** Converts lengthy medical reports into patient-friendly summaries.
- **Medication & Treatment Guidance:** Provides safe medication reminders and adherence tips.
- **Wellness Recommendations:** Lifestyle, diet, and exercise suggestions tailored to user data.
- **Anomaly Detection in Vitals:** Alerts for unusual health readings from wearables.
- **Doctor's Dashboard:** AI-assisted insights for doctors to optimize patient care.
- **Multimodal Data Support:** Accepts lab reports, prescriptions, images (X-ray, MRI), and CSV health data.

## 3. Architecture

### - Frontend (Streamlit/Gradio):

Intuitive dashboard with patient chat, report uploads, symptom checker, and doctor's analytics view.

### - Backend (FastAPI):

Powers health record processing, AI queries, anomaly detection, and medication reminders.

- **LLM Integration (IBM Watsonx Granite):**

Provides medical language understanding, report summarization, and chatbot responses.

- **Vector Search (Pinecone):**

Enables semantic search over medical research papers and patient history.

- **ML Modules:**

- **Forecasting:** Predicts patient recovery trends or hospital resource usage.

- **Anomaly Detection:** Detects abnormalities in vitals like heart rate, glucose levels, or blood pressure.

## 4. Setup Instructions

Prerequisites

Python 3.9+

IBM Watsonx API key

Pinecone API key

Virtual environment tools

Internet access

## Installation

1. Clone the repository.
2. Install dependencies via requirements.txt.
3. Configure .env with credentials.
4. Start backend with FastAPI.
5. Launch frontend with Streamlit.
6. Upload reports or start health chat.

## 5. Folder structure

app/ – FastAPI backend logic   app/api/ – Routes for chat, reports, anomaly detection   ui/ – Streamlit/Gradio frontend components   health\_dashboard.py – Main entry dashboard   watson\_llm.py – Handles AI interactions with IBM Watsonx   report\_summarizer.py – Converts medical reports into summaries   symptom\_checker.py – AI-powered health query engine   vitals\_analyzer.py – Detects anomalies in patient vitals   forecasting\_module.py – Predicts recovery & health trends

## 6. Running the Application

- Start FastAPI server for backend APIs.
- Run Streamlit dashboard for patient/doctor interface.
- **Navigate via sidebar:** Chat, Symptom Checker, Reports, Doctor's Insights.
- Upload medical documents or wearable data to get real-time outputs.

## 7. API Documentation

- POST /chat/ask – Ask medical/health queries.
- POST /upload-report – Upload lab/medical reports for AI summary.
- GET /health-tips – Personalized wellness advice.
- POST /symptom-checker – Get differential diagnosis suggestions.
- POST /vitals-check – Analyze patient vitals for anomalies.

## 8. Authentication

- Token-based authentication (JWT).
- **Role-based access:** Patient / Doctor / Researcher.

- Secure storage of sensitive health records (HIPAA-compliant setup recommended).

## 9. User interface

1. Sidebar navigation for Patients and Doctors.
2. **Patients:** Symptom chat, health tips, medication reminders.
3. **Doctors:** Patient analytics dashboard, AI-assisted insights.
4. Real-time vitals graphing and PDF download for summaries.

## 10. Testing

1. **Unit Testing:** For symptom checker and report summarizer.
2. **API Testing:** Swagger UI and Postman.
3. **Manual Testing:** Uploading diverse report formats.
4. **Edge Cases:** Handling incomplete symptoms, invalid reports.

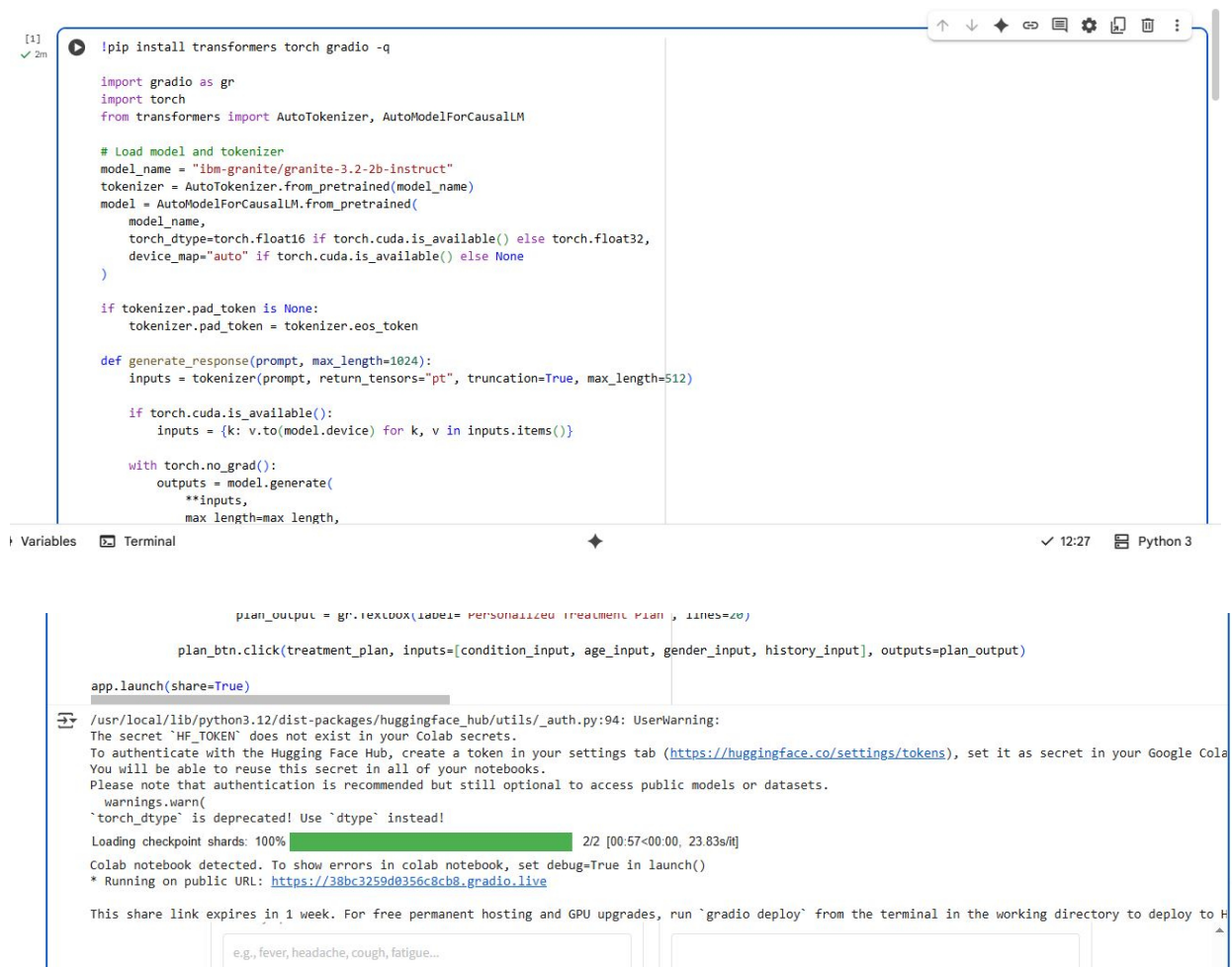
## 11. Future enhancement

- Integration with wearable IoT devices (Fitbit, Apple Watch).
- Multi-language support for global health accessibility.
- AI-powered telemedicine consultation.

- Predictive disease outbreak monitoring.

## 12. Conclusion

Health AI is no longer science fiction—it is reshaping how we think about and access healthcare. With continuous development, it has the potential to make healthcare more affordable, accessible, and accurate, empowering individuals and transforming lives.



```
[1]
✓ 2m
!pip install transformers torch gradio -q

import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    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,
            plan_output = gr.textbox(label="Personalized Treatment Plan", lines=20)

    plan_btn.click(treatment_plan, inputs=[condition_input, age_input, gender_input, history_input], outputs=plan_output)

app.launch(share=True)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab notebook. You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
`torch_dtype` is deprecated! Use `dtype` instead!
Loading checkpoint shards: 100% 2/2 [00:57<00:00, 23.83s/it]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://38bc3259d0356c8cb8.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to H
e.g., fever, headache, cough, fatigue...
```