HEALTH AI: INTELLIGENT HEALTHCARE ASSISTANT DOCUMENATATION

1.Introduction

- **PROJECT TITLE:** Health Ai: Intelligent Healthcare Assistant
- **TEAM LEADER:** BARADHAN.G
- **TEAM MEMBER:** AJITH KUMAR.S
- **TEAM MEMBER:** ARCHUNAN.P
- **TEAM MEMBER:** AKASH.V

2.PROJECT OVEREVIEW

This project focuses on developing an Artificial Intelligence (AI) system designed to improve healthcare services by enabling faster diagnosis, personalized treatment, efficient patient management, and data-driven decision-making.

Objectives:

- Enhance diagnostic accuracy using AI and machine learning.
- Provide personalized treatment recommendations.
- Support remote monitoring and telehealth.
- Automate clinical tasks and documentation.
- Analyze large-scale healthcare data for predictive insights.

3. Healthcare AI Architecture

1. **Data Sources:**
   EHRs, medical images, wearables, lab reports.
2. **Data Ingestion:**
   APIs and secure pipelines (HL7/FHIR).
3. **Storage:**
   Cloud or on-premise, HIPAA/GDPR compliant.
4. **Processing:**
   Data cleaning, NLP for text, image preprocessing.
5. **AI/ML Layer:**
   Models for prediction, diagnosis, image analysis, NLP.
6. **Application Layer:**
   Clinical decision support, alerts, dashboards.
7. **User Interface:**
   Web/mobile apps for doctors and patients.
8. **Security:**
   Encryption, access control, compliance.

4. 📁 Folder Structure: Health AI Project

```
health-ai-project/
├── data/
│   ├── raw/
│   └── processed/
│
├── models/
│   └── checkpoints/
│
├── notebooks/
│
├── src/
│   ├── data_preprocessing/
│   ├── training/
│   ├── inference/
│   └── utils/
│
├── app/
│   ├── api/
│   └── ui/
│
├── config/
│
```

├── **tests/**

│

├── **reports/**

│

├── **docs/**

│

├── **requirements.txt**

└── **README.md**

5.Setup Instructions

1.Clone the repo

```
git clone <repo_url>&& cd health-ai-project
```

2. **Create & activate virtual environment**

```
python –m venv venv && source venv/bin/activate
```

3. **Install dependencies**

```
pip install -r requirements.txt
```

4. **Set environment variables**
   Create a `.env` file with keys like `API_KEY`, `DB_URI`, `MODEL_PATH`.
5.
6. **Prepare data & run app**

```
python src/data_preprocessing/preprocess.py
uvicorn app.api.main:app --reload  # or streamlit run app/ui/app.py
```

6.Running the Application

1. **Activate your virtual environment**

```
source venv/bin/activate
```

2. **Start the backend (API)**

```
uvicorn app.api.main:app --reload
```

3. **(Optional) Start the frontend/UI**

```
streamlit run app/ui/app.py
```

4. **Access the app:**

- API: `http://localhost:8000`
- UI: `http://localhost:8501`

7. API Documentation

1. GET /

Description: Health check endpoint
Response:

```
{ "status": "ok" }
```

---

2. POST /predict

Description: Get disease prediction or diagnosis from input data
Request Body (JSON):

```
{
  "age": 45,
  "gender": "male",
  "symptoms": ["chest pain", "shortness of breath"],
  "vitals": {
    "heart_rate": 110,
    "bp": "140/90"
  }
}
```

Response:

```
{
  "prediction": "High risk of heart disease",
  "confidence": 0.92
}
```

---

3. POST /upload-image

Description: Upload medical image (e.g., X-ray) for AI analysis
Request: `multipart/form-data`
Response:

```
{
  "diagnosis": "Possible pneumonia",
  "confidence": 0.87
}
```

---

4. GET /docs

Description: Interactive API docs (Swagger UI)

8.Authentication in Healthcare AI

- **Purpose:** Secure user access to sensitive patient data and AI services.
- **Methods:**
    - **JWT (JSON Web Tokens):** For stateless, scalable authentication.
    - **OAuth2:** Allows secure third-party access and single sign-on (SSO).
    - **Multi-Factor Authentication (MFA):** Adds extra security layer (e.g., SMS or authenticator apps).
- **User Roles:** Role-based access control (RBAC) to restrict permissions (e.g., doctor, nurse, admin).
- **Encryption:** Passwords stored hashed (e.g., bcrypt), secure token storage.
- **Compliance:** Adhere to HIPAA/GDPR for protecting health data privacy.

9. Interface in Healthcare AI

- **Purpose:** Provide intuitive access for doctors, patients, and staff to AI-powered tools and insights.
- **Key Features:**
    - **Dashboard:** Real-time patient data, alerts, and AI predictions.
    - **Data Input Forms:** For symptoms, vitals, and patient history.
    - **Medical Image Viewer:** To display AI-analyzed scans (X-rays, MRIs).
    - **Reports & Visualizations:** Easy-to-understand charts and summaries.
    - **Multi-Platform:** Web and mobile apps for accessibility anywhere.
- **Technologies:** React, Angular, Flutter, or Streamlit for rapid UI development.
- **Focus:** Usability, accessibility, data privacy, and seamless integration with backend AI services

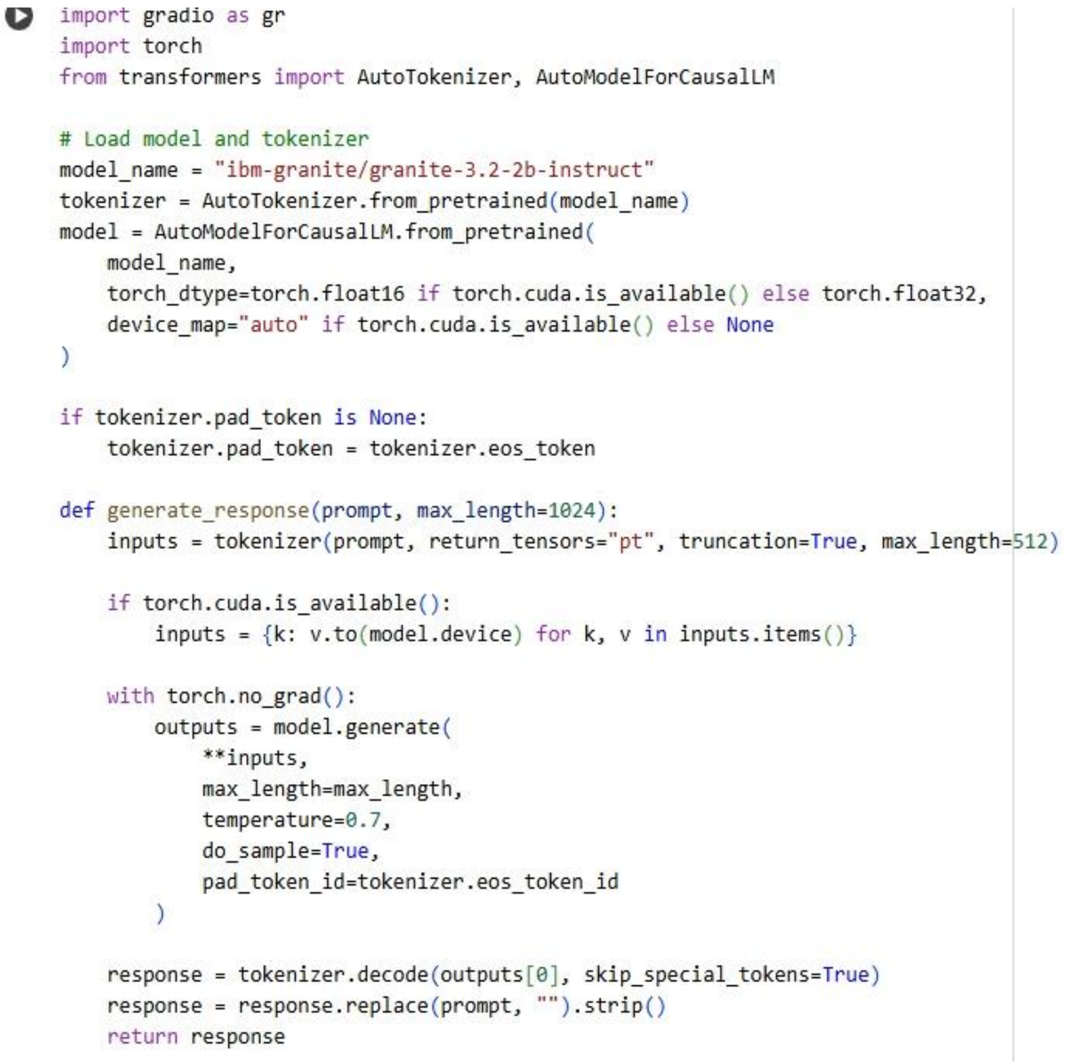
10. Testing

- **Types of Testing:**
    - **Unit Testing:** Verify individual functions/modules (e.g., data preprocessing, model functions).
    -
    - **Integration Testing:** Check interactions between components like data pipeline and model inference.
    -
    - **Model Validation:** Evaluate AI model accuracy, precision, recall on test datasets.
    - **Performance Testing:** Ensure responsiveness and scalability under load.
    - **Security Testing:** Test data privacy, authentication, and compliance with regulations.

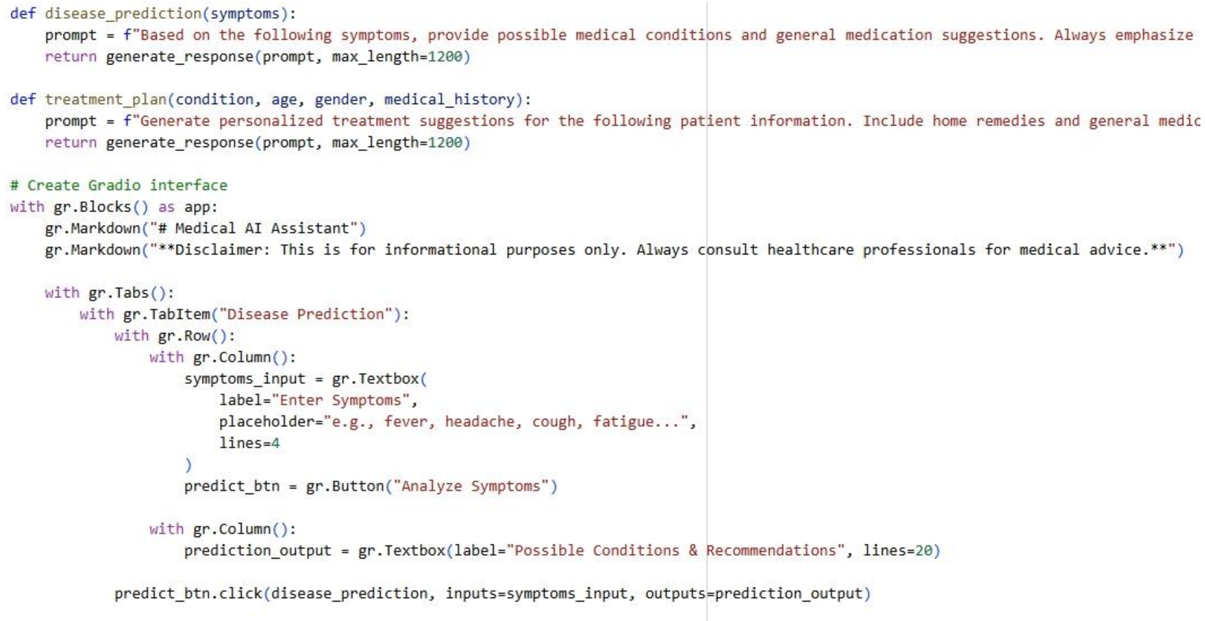
    - **User Acceptance Testing (UAT):** Validate UI/UX with end-users (clinicians, patients).
- **Tools:**
    - pytest, unittest (Python)
    - Postman (API testing)
    - TensorBoard, MLflow (model monitoring)
- **Importance:** Ensures reliability, safety, and compliance of AI in critical healthcare environments.

11. SCREEN SHOTS

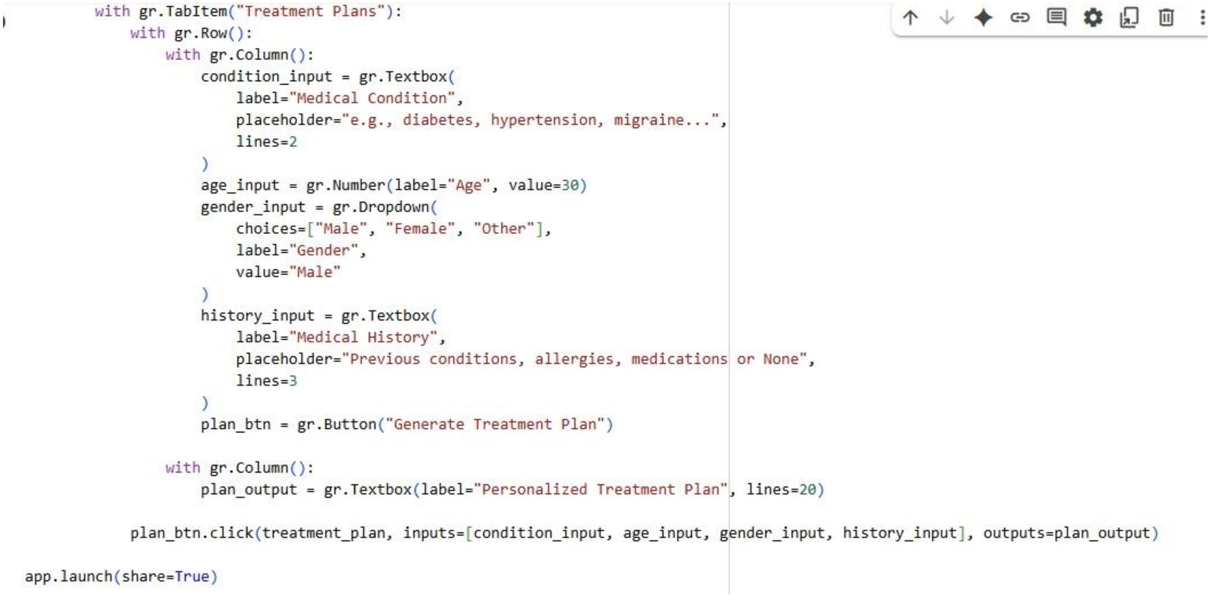
```
!pip install transformers torch gradio -q
```



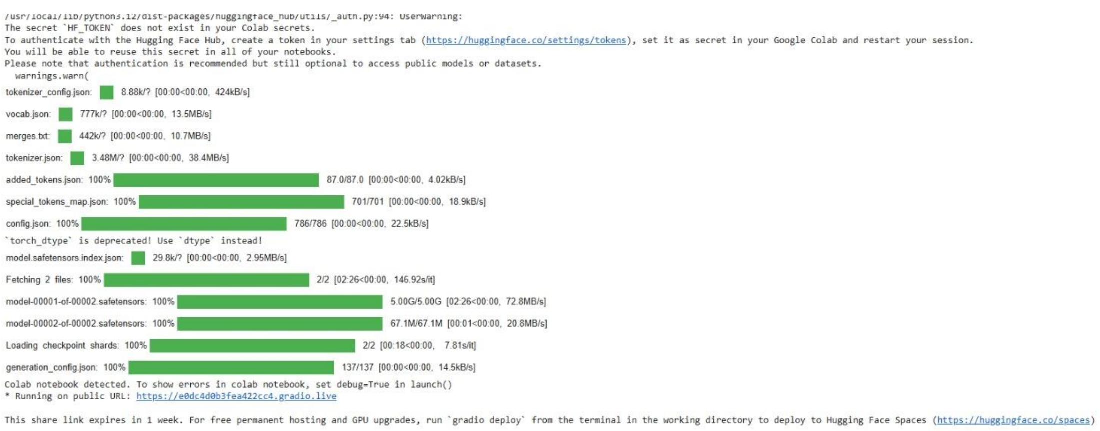
```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# 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,
            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
```



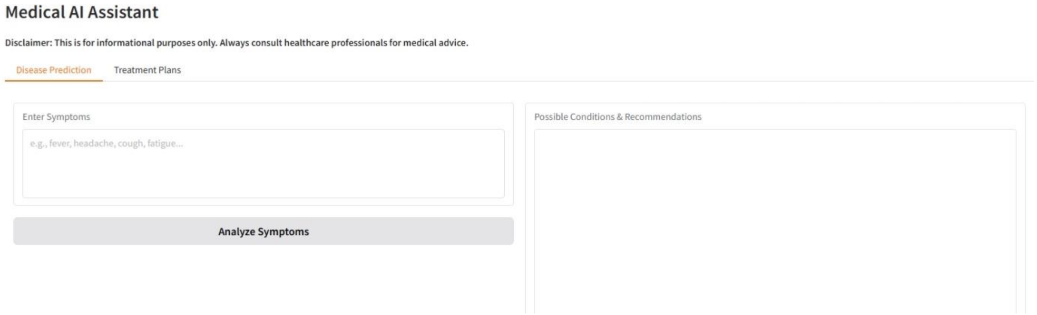
```
def disease_prediction(symptoms):
    prompt = f"Based on the following symptoms, provide possible medical conditions and general medication suggestions. Always emphasize
    return generate_response(prompt, max_length=1200)

def treatment_plan(condition, age, gender, medical_history):
    prompt = f"Generate personalized treatment suggestions for the following patient information. Include home remedies and general medic
    return generate_response(prompt, max_length=1200)

# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# Medical AI Assistant")
    gr.Markdown("**Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.**")

    with gr.Tabs():
        with gr.TabItem("Disease Prediction"):
            with gr.Row():
                with gr.Column():
                    symptoms_input = gr.Textbox(
                        label="Enter Symptoms",
                        placeholder="e.g., fever, headache, cough, fatigue...",
                        lines=4
                    )
                    predict_btn = gr.Button("Analyze Symptoms")

                with gr.Column():
                    prediction_output = gr.Textbox(label="Possible Conditions & Recommendations", lines=20)

            predict_btn.click(disease_prediction, inputs=symptoms_input, outputs=prediction_output)
```



```
        with gr.TabItem("Treatment Plans"):
            with gr.Row():
                with gr.Column():
                    condition_input = gr.Textbox(
                        label="Medical Condition",
                        placeholder="e.g., diabetes, hypertension, migraine...",
                        lines=2
                    )
                    age_input = gr.Number(label="Age", value=30)
                    gender_input = gr.Dropdown(
                        choices=["Male", "Female", "Other"],
                        label="Gender",
                        value="Male"
                    )
                    history_input = gr.Textbox(
                        label="Medical History",
                        placeholder="Previous conditions, allergies, medications or None",
                        lines=3
                    )
                    plan_btn = gr.Button("Generate Treatment Plan")

                with gr.Column():
                    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 and restart your session.
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(
tokenizer_config.json: █ 8.88k/? [00:00<00:00, 424kB/s]
vocab.json: █ 777k/? [00:00<00:00, 13.5MB/s]
merges.txt: █ 442k/? [00:00<00:00, 10.7MB/s]
tokenizer.json: █ 3.48M/? [00:00<00:00, 38.4MB/s]
added_tokens.json: 100% ██████████ 87.0/87.0 [00:00<00:00, 4.02kB/s]
special_tokens_map.json: 100% ██████████ 701/701 [00:00<00:00, 18.9kB/s]
config.json: 100% ██████████ 786/786 [00:00<00:00, 22.5kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: █ 29.8k/? [00:00<00:00, 2.95MB/s]
Fetching 2 files: 100% ██████████ 2/2 [02:26<00:00, 146.92s/it]
model-00001-of-00002.safetensors: 100% ██████████ 5.00G/5.00G [02:26<00:00, 72.8MB/s]
model-00002-of-00002.safetensors: 100% ██████████ 67.1M/67.1M [00:01<00:00, 20.8MB/s]
Loading checkpoint shards: 100% ██████████ 2/2 [00:18<00:00, 7.81s/it]
generation_config.json: 100% ██████████ 137/137 [00:00<00:00, 14.5kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://e0dc4d0b3fea422cc4.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 Hugging Face Spaces (https://huggingface.co/spaces)
```



Medical AI Assistant

Disclaimer: This is for informational purposes only. Always consult healthcare professionals for medical advice.

Disease Prediction
Treatment Plans
Enter Symptoms
e.g., fever, headache, cough, fatigue...
Analyze Symptoms
Possible Conditions & Recommendations

12. Known Issues in Healthcare AI

- **Data Quality & Bias:** Incomplete, inconsistent, or biased medical data can lead to inaccurate predictions.
- **Interpretability:** AI models often act as “black boxes,” making it hard for clinicians to trust decisions.
- **Privacy & Security:** Handling sensitive health data requires strict compliance (HIPAA/GDPR) and robust security measures.
- **Regulatory Challenges:** Navigating healthcare regulations slows deployment and innovation.
- **Integration Complexity:** Difficulties in integrating AI tools with existing hospital systems (EHRs, PACS).
- **Generalization:** Models trained on specific populations may not perform well across diverse groups.
- **Clinical Validation:** Extensive testing and validation are needed before clinical adoption.

13. Future Enhancements in Healthcare AI (Brief)

- **Explainable AI:** Improve transparency to build clinician trust and meet regulatory needs.
- **Personalized Medicine:** Tailor treatments using genomics and real-time patient data.
- **Advanced Multi-modal AI:** Combine imaging, clinical notes, and sensor data for richer insights.
- **Real-time Remote Monitoring:** Enhance telehealth with continuous AI-powered vitals tracking.
- **Integration with Robotics:** Support surgeries and patient care through AI-driven robotics.
- **Federated Learning:** Enable collaborative model training across hospitals without sharing sensitive data.
- **Improved Data Privacy:** Use techniques like differential privacy and homomorphic encryption.