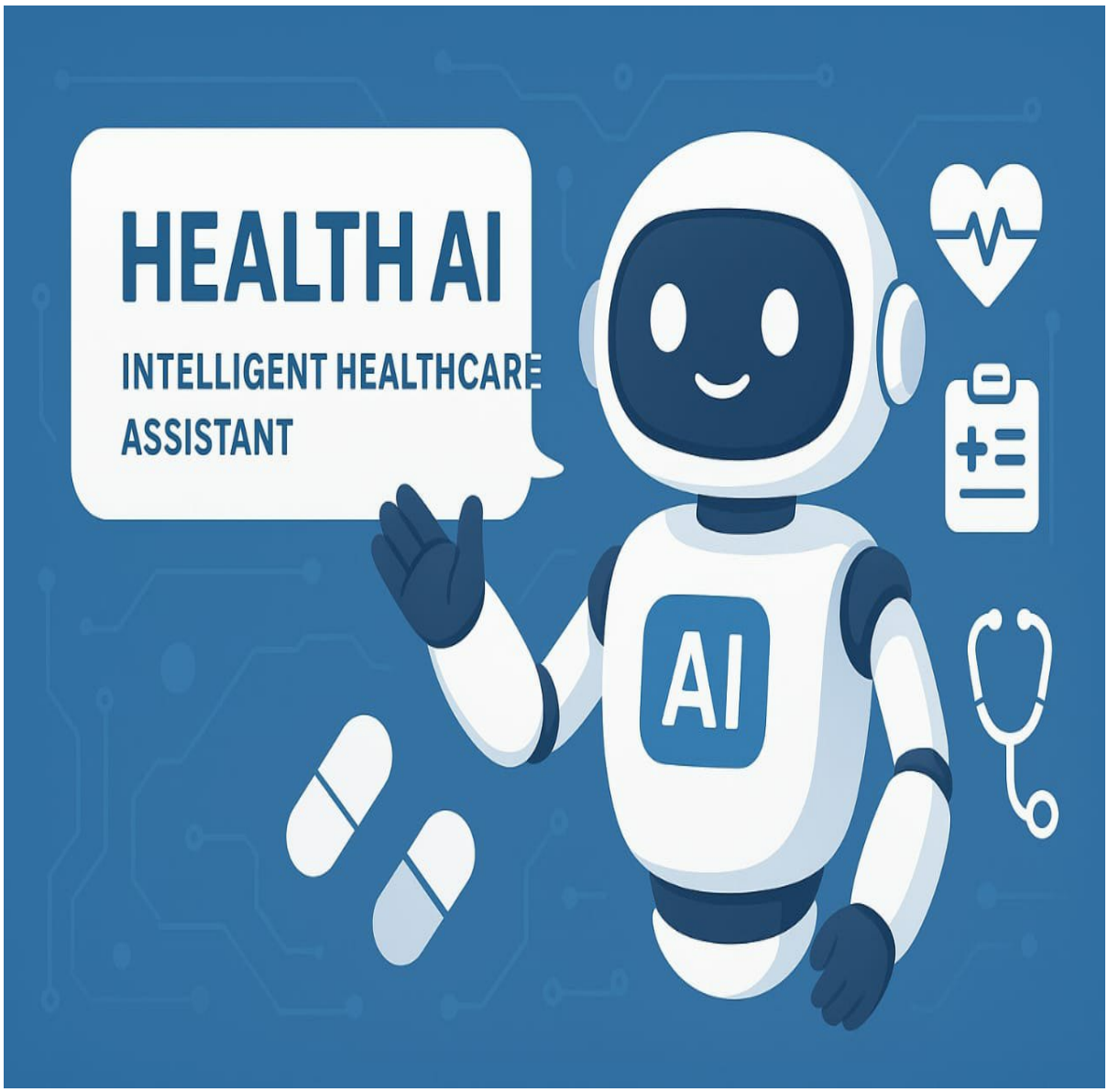
Health AI – Intelligent Healthcare Assistant



1. Introduction

Health AI is a generative AI-based intelligent healthcare assistant built using the IBM Granite model from Hugging Face. The system is designed to provide smart, easy-to-understand medical assistance through features such as patient chat, disease prediction, and treatment planning. The project leverages Google Colab for deployment, ensuring accessibility, security, and GPU-based efficiency. It provides an interactive interface through Gradio, enabling patients and users to get quick guidance for healthcare-related queries.

2. Objectives

To develop a healthcare assistant that provides real-time, AI-driven support.

To integrate IBM Granite models for natural language understanding and response generation.

To create a simple, interactive user interface using Gradio.

To deploy the application on Google Colab for ease of access.

To enable project sharing and collaboration using GitHub.

3. Pre-requisites

Gradio Framework Knowledge (for UI development)

IBM Granite Models on Hugging Face (LLM backend)

Python Programming Proficiency

Version Control with Git

Google Colab with T4 GPU

4. Project Workflow

Activity 1: Exploring Naan Mudhalvan Smart Interz Portal

Access the portal, enroll in the project, and view guided resources.

Navigate to the project workspace and track progress.

Activity 2: Choosing IBM Granite Model from Hugging Face

Create an account on Hugging Face.

Search and select an IBM Granite model (e.g., granite-3.2-2b-instruct).

Use the model for healthcare AI integration.

Activity 3: Running Application in Google Colab

Open Google Colab and create a new notebook.

Set runtime to T4 GPU.

Install dependencies:

!pip install transformers torch gradio -q

Run the provided Health AI code to launch the Gradio application.

Access the app through the generated public link.

Activity 4: Uploading Project to GitHub

Create a GitHub account and repository.

Download the .py file from Colab.

Upload the file to GitHub and commit changes.

Maintain project version control and collaboration.

5. System Architecture

1. Frontend: Gradio interface for patient interaction.
2. Backend: IBM Granite LLM for processing and generating medical insights.
3. Deployment: Google Colab with GPU acceleration.
4. Version Control: GitHub repository for project files.

6. Features

Patient Chat for natural interaction.

Disease Prediction and preliminary diagnosis assistance.

Treatment Plan suggestions (educational, not medical advice).

Cloud-based, secure, and scalable deployment.

7. Testing

Validate model responses using sample medical queries.

Ensure Gradio UI loads correctly in the browser.

Test disease prediction accuracy with dummy datasets.

Cross-check model's suggestions against standard medical references.

8. Known Issues

Model may generate generic or non-specialized advice.

Dependency on internet connection and Colab GPU availability.

Limited to educational purposes (not a substitute for real medical consultation).

9. Future Enhancements

Integration with real-time patient health records (EHR).

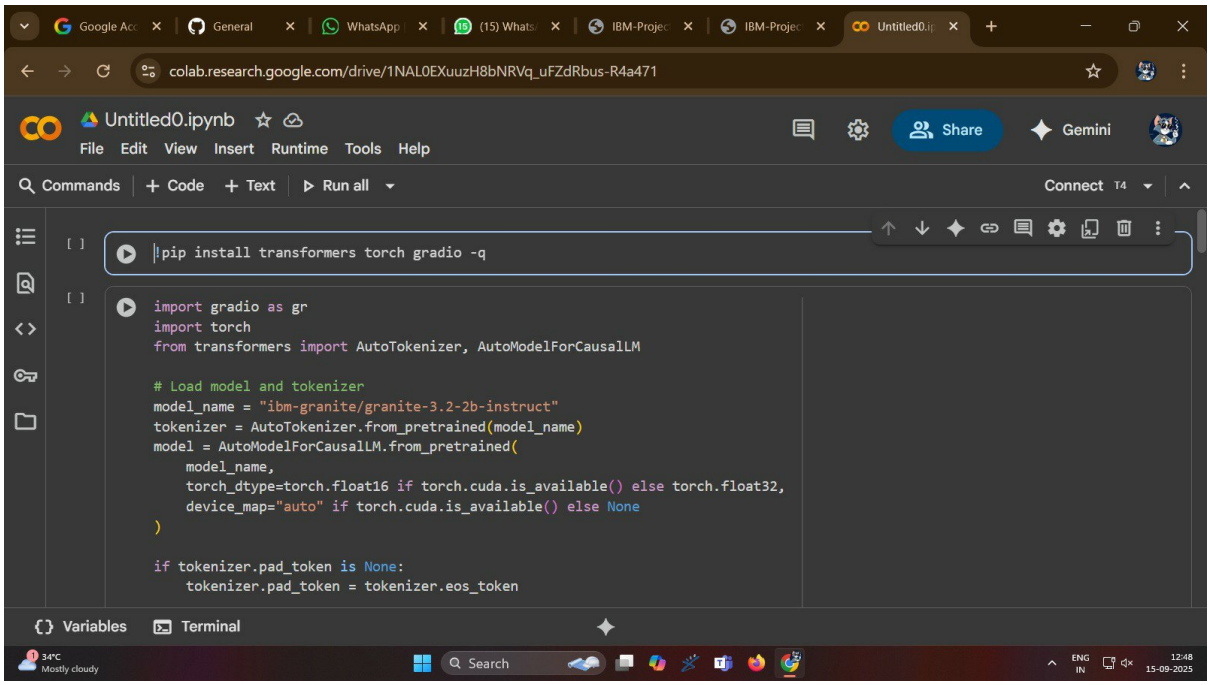
Adding multilingual support for broader accessibility.

Enhancing disease prediction accuracy with custom datasets.

Deploying on cloud platforms (AWS, IBM Cloud, Azure) for production use.

Expanding functionality to include appointment scheduling, reminders, and personalized recommendations.

10.Project Screenshots



colab.research.google.com/drive/1NAL0EXuuzH8bNRVq_uFZdRbus-R4a471

Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Connect T4

```python
)

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
        )
```

Variables Terminal

colab.research.google.com/drive/1NAL0EXuuzH8bNRVq_uFZdRbus-R4a471

Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Connect T4

```python
            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
    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 gen
    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 advi
```

Variables Terminal

colab.research.google.com/drive/1NAL0EXuuzH8bNRVq_uFZdRbus-R4a471

Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Connect T4

```
[ ]
        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(
```

Variables Terminal

colab.research.google.com/drive/1NAL0EXuuzH8bNRVq_uFZdRbus-R4a471

Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Connect T4

```
[ ]
                            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_outpu
```

Variables Terminal

colab.research.google.com/drive/1NAL0EXuuzH8bNRVq_uFZdRbus-R4a471

Untitled0.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share Gemini Connect T4

```
                    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_outpu

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
```

Variables Terminal

colab.research.google.com/drive/1NAL0EXuuzH8bNRVq_uFZdRbus-R4a471

Untitled0.ipynb

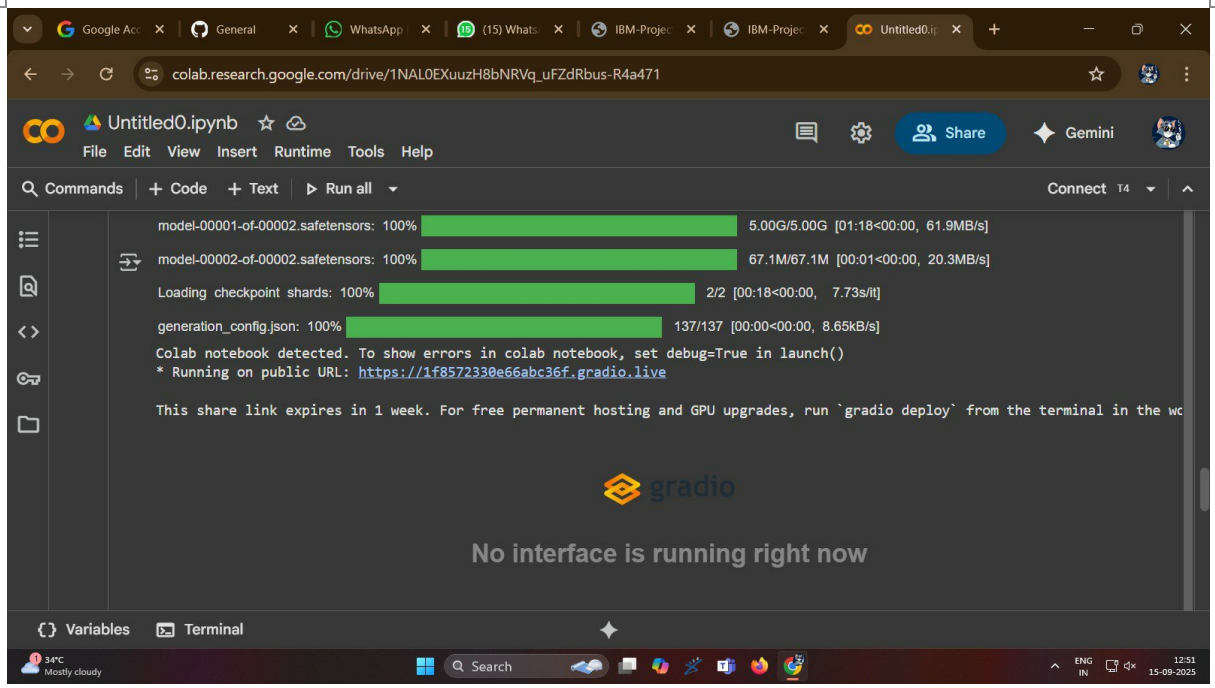
File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share Gemini Connect T4

```
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, 398kB/s]
vocab.json: 777k/? [00:00<00:00, 27.8MB/s]
merges.txt: 442k/? [00:00<00:00, 20.3MB/s]
tokenizer.json: 3.48M/? [00:00<00:00, 99.3MB/s]
added_tokens.json: 100% 87.0/87.0 [00:00<00:00, 9.29kB/s]
special_tokens_map.json: 100% 701/701 [00:00<00:00, 83.0kB/s]
config.json: 100% 786/786 [00:00<00:00, 69.0kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json: 29.8k/? [00:00<00:00, 2.47MB/s]
Fetching 2 files: 100% 2/2 [01:18<00:00, 78.80s/it]
model-00001-of-00002.safetensors: 100% 5.00G/5.00G [01:18<00:00, 61.9MB/s]
```

Variables Terminal



Conclusion:

Health AI – Intelligent Healthcare Assistant project was successfully done by Kalpana.K , Kiruthika.K , Kiruthika Devi.R.

Demo video link

https://1drv.ms/v/c/e502c8164e3c18dc/EUo13hyZ0ulBrFbBYp8PjWsBxiuDLtd9BcPkUBMj46vBhQ