Health AI – Intelligent Web Application for Preliminary Medical Guidance

Project Documentation

- **Team member:** Archana. P
- **Team member:** Gayathri. S
- **Team member:** Preethigha. S
- **Team member:** Sandhiya. E
- **Team member:** Sobhana. C

Abstract

Health AI is a browser-based application that delivers informational health guidance by analyzing

user-entered symptoms and personal details. The system combines a Python backend, an interactive

Gradio interface, and the IBM Granite 3.2 2B Instruct large language model to suggest potential medical

conditions and outline basic treatment plans. The application is designed strictly for educational

purposes and includes strong disclaimers to remind users that it is not a substitute for professional

medical care. This report documents the project's objectives, background research, system design,

implementation, testing, and ethical considerations.

Introduction

Healthcare systems everywhere face growing pressure as populations rise and patients increasingly

seek instant answers to health questions. While search engines provide information, they often return unfiltered results that can be confusing or misleading. Recent advances in natural-language processing

(NLP) allow conversational AI to provide more contextual responses, but responsible use is critical in a

medical setting.

Health AI explores this space by offering a first-level guidance tool. It lets users describe symptoms or

provide known conditions and basic demographic details. The system then generates a plain-language

explanation of possible conditions and general home-care measures. Every output contains a visible

disclaimer emphasizing that only qualified healthcare professionals can diagnose or treat disease.

Background and Literature Review

AI in Healthcare: Artificial intelligence has been applied to diagnostics, imaging, triage chatbots, and

electronic medical record analysis. Examples include IBM Watson's oncology recommendations and

triage bots such as Ada Health. However, most commercial systems rely on curated medical databases

and structured decision trees.

Large Language Models (LLMs): The arrival of transformer-based LLMs (e.g., GPT, LLaMA, Granite)

allows free-form dialogue and reasoning over natural language input. They can synthesize knowledge

from diverse training data and produce fluent text, but they also hallucinate—producing confident yet

inaccurate statements. This limitation requires strict safeguards when applying LLMs to health contexts.

Motivation for Health AI: Many people want quick preliminary advice before visiting a doctor. Health AI demonstrates how an LLM can be combined with clear disclaimers and a minimalistic interface to

deliver such advice responsibly, without claiming diagnostic authority.

Objectives

• Collect user symptoms and personal details through a friendly web interface.

• Generate potential medical conditions and general treatment suggestions.

• Educate users about the importance of professional diagnosis.

• Showcase the integration of Python, Gradio, and a modern LLM in a real-world scenario.

System Requirements

Hardware:

• Minimum: dual-core CPU, 4 GB RAM

• Recommended: quad-core CPU, 8 GB RAM or higher

• Internet connection for model download and hosting

Software:

• Operating System: Windows / macOS / Linux

• Python 3.10 or later

• Libraries: transformers, torch, gradio

Technology Stack

Programming Language – Python 3.10+: Handles backend logic, prompt engineering, and AI model

inference.

Frontend – Gradio Framework: Generates a responsive browser interface with two main tabs: Disease

Prediction and Treatment Plan. Provides text boxes, dropdown menus, and buttons. Automatically

launches a local web server and can create a temporary public URL.

Backend – Python Application Layer: Functions collect and sanitize user inputs, build structured

prompts, and call the language model. Uses PyTorch to leverage GPU acceleration if available. Cleans

model output before sending it back to the interface.

AI Model – IBM Granite 3.2 2B Instruct: A transformer-based LLM accessed through Hugging Face.

Generates natural-language predictions of conditions and generalized treatment ideas.

Deployment: Runs locally with python health_ai.py and can generate a shareable public link.

Architecture

The architecture follows a three-layer design:

1. User Interface Layer – Browser front end built entirely with Gradio.

2. Application Logic Layer – Python functions for prompt creation, device selection, and output

formatting.

3. AI Processing Layer – IBM Granite LLM performing natural-language inference.

Data flow: User Input → Prompt Construction → Model Inference → Output Cleaning → Browser

Display.

Workflow

1. Input Stage: The user opens the web app, chooses a tab, and provides symptoms or condition

details.

2. Prompt Generation: Backend functions create a carefully worded prompt that always embeds a

disclaimer.

3. Model Execution: The prompt is tokenized and sent to the Granite model. GPU is used when

available.

4. Response Handling: The generated text is decoded, special tokens removed, and the final response

returned to the interface.

5. Display: The user sees possible conditions or a suggested treatment plan with the disclaimer

prominently displayed.

Detailed Design

Model Initialization: Code loads tokenizer and model from Hugging Face and configures GPU use.

Core Functions: generate_response() manages tokenization and inference. disease_prediction() builds

a medical prompt for symptom analysis. treatment_plan() creates a personalized plan.

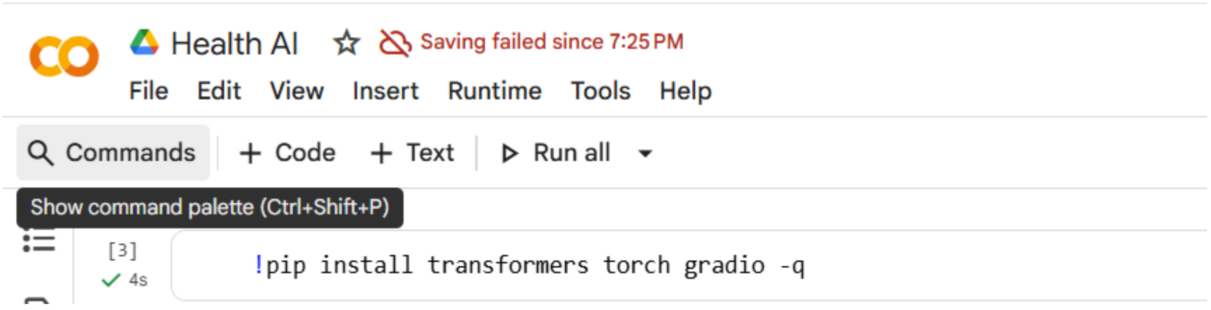
Interface Construction: Implemented with gr.Blocks(), using gr.TabItem, gr.Textbox, gr.Dropdown, and

gr.Button for a clean layout.

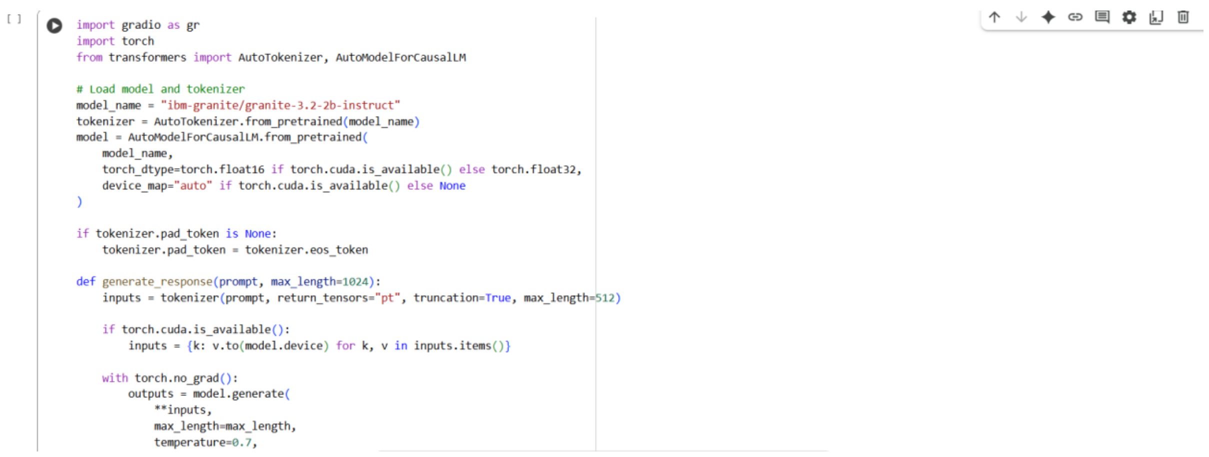
Prompt Engineering: Each prompt stresses that the output is for information only, with max_length and

temperature settings for concise, varied responses.

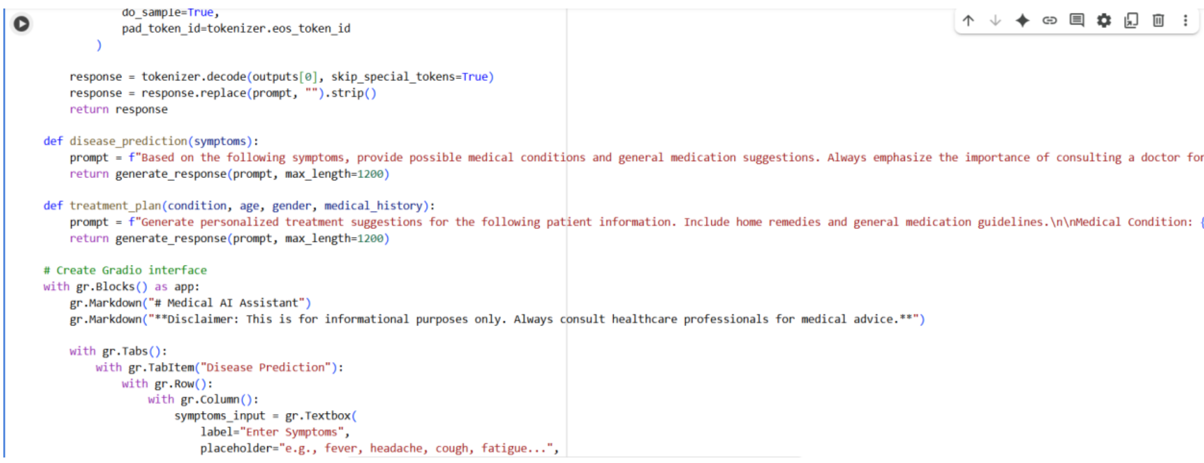
Implementation and Deployment


Health AI
Saving failed since 7:25 PM
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Show command palette (Ctrl+Shift+P)
[3]
4s

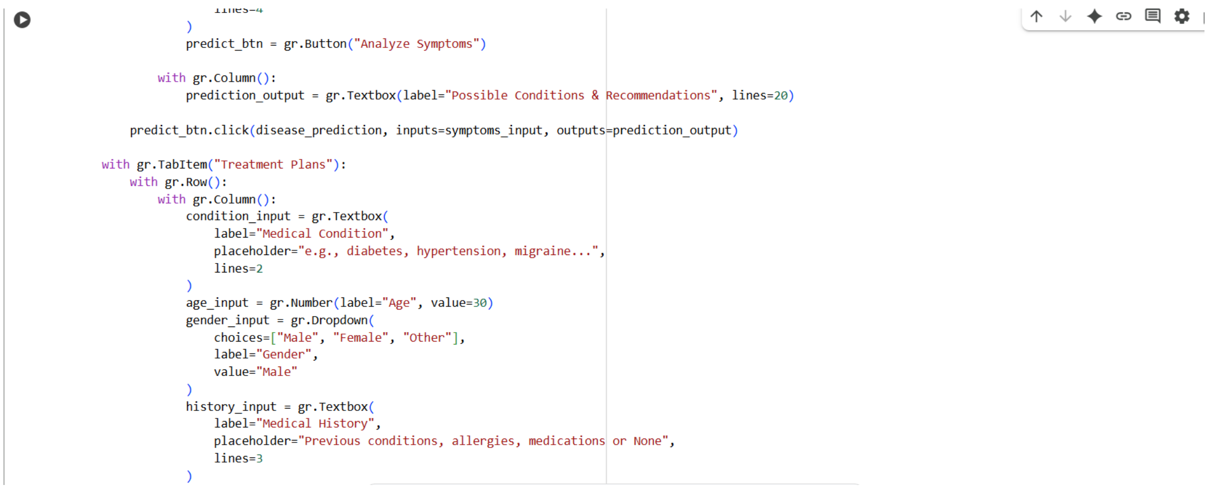
```
!pip install transformers torch gradio -q
```



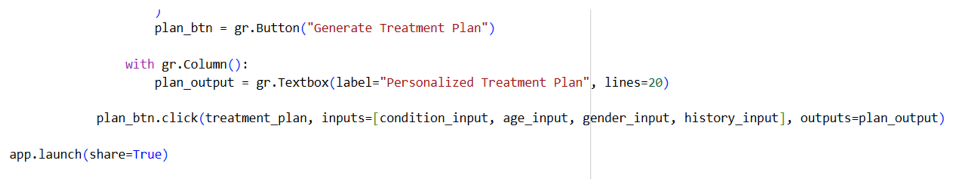
```
[ ]
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# 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 the importance of consulting a doctor for
    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 medication guidelines.\n\nMedical Condition: {
    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...",
```

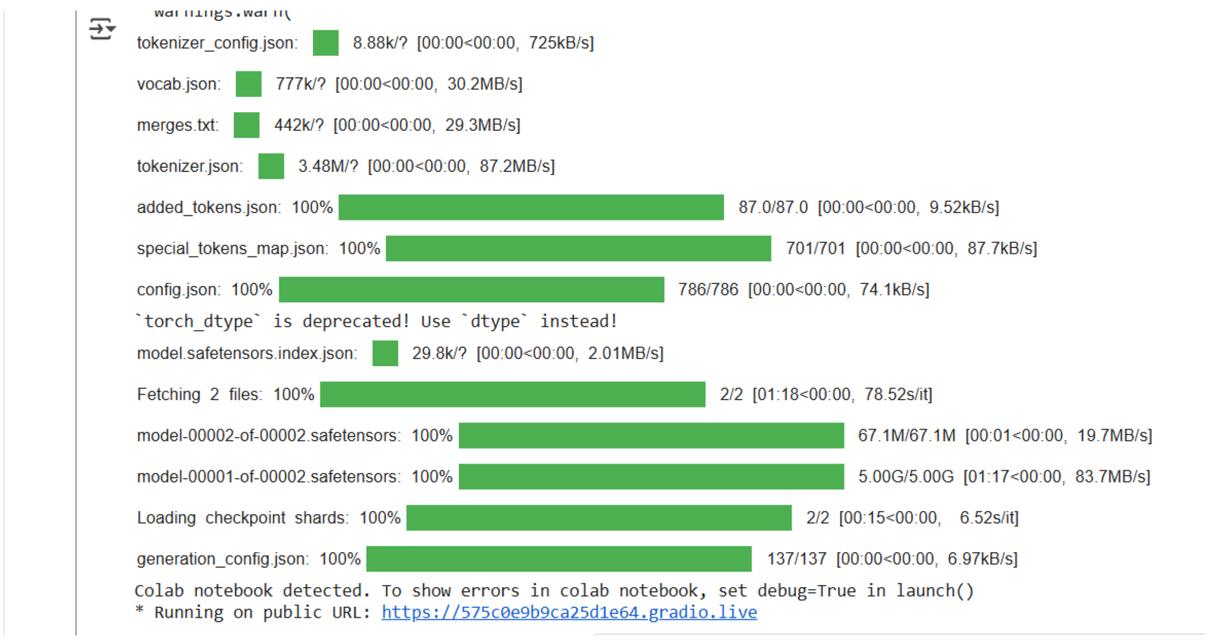


```
                        lines=4
                    )
                    predict_btn = gr.Button("Analyze Symptoms")

                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
                    )
```

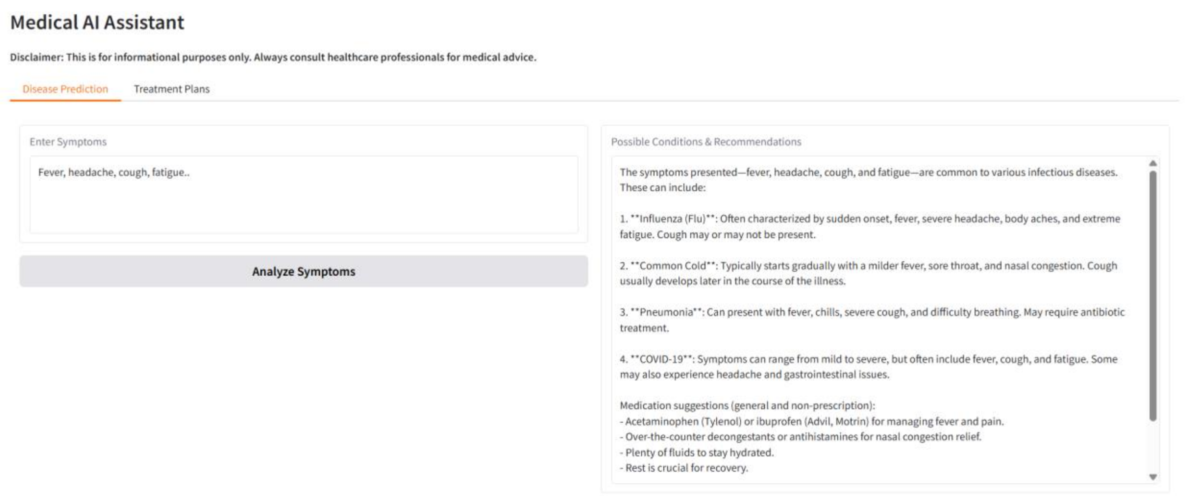


```
                    )
                    plan_btn = gr.Button("Generate Treatment Plan")

                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)
```

Output:



```
    warnings.warn(
tokenizer_config.json:      8.88k/? [00:00<00:00, 725kB/s]
vocab.json:      777k/? [00:00<00:00, 30.2MB/s]
merges.txt:      442k/? [00:00<00:00, 29.3MB/s]
tokenizer.json:      3.48M/? [00:00<00:00, 87.2MB/s]
added_tokens.json: 100%      87.0/87.0 [00:00<00:00, 9.52kB/s]
special_tokens_map.json: 100%      701/701 [00:00<00:00, 87.7kB/s]
config.json: 100%      786/786 [00:00<00:00, 74.1kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:      29.8k/? [00:00<00:00, 2.01MB/s]
Fetching 2 files: 100%      2/2 [01:18<00:00, 78.52s/it]
model-00002-of-00002.safetensors: 100%      67.1M/67.1M [00:01<00:00, 19.7MB/s]
model-00001-of-00002.safetensors: 100%      5.00G/5.00G [01:17<00:00, 83.7MB/s]
Loading checkpoint shards: 100%      2/2 [00:15<00:00, 6.52s/it]
generation_config.json: 100%      137/137 [00:00<00:00, 6.97kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://575c0e9b9ca25d1e64.gradio.live
```



Medical AI Assistant

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

Enter Symptoms

Fever, headache, cough, fatigue..

Possible Conditions & Recommendations

The symptoms presented—fever, headache, cough, and fatigue—are common to various infectious diseases. These can include:

1. **Influenza (Flu)**: Often characterized by sudden onset, fever, severe headache, body aches, and extreme fatigue. Cough may or may not be present.

2. **Common Cold**: Typically starts gradually with a milder fever, sore throat, and nasal congestion. Cough usually develops later in the course of the illness.

3. **Pneumonia**: Can present with fever, chills, severe cough, and difficulty breathing. May require antibiotic treatment.

4. **COVID-19**: Symptoms can range from mild to severe, but often include fever, cough, and fatigue. Some may also experience headache and gastrointestinal issues.

Medication suggestions (general and non-prescription):
- Acetaminophen (Tylenol) or ibuprofen (Advil, Motrin) for managing fever and pain.
- Over-the-counter decongestants or antihistamines for nasal congestion relief.
- Plenty of fluids to stay hydrated.
- Rest is crucial for recovery.

Testing and Evaluation

Functional Testing: Entered diverse symptom sets to confirm that responses remain coherent.

Performance Testing: Measured average response times on CPU vs GPU. Verified memory usage remains stable for extended sessions.

User Testing: Non-technical volunteers navigated the interface without guidance, confirming usability.

Safety Verification: Ensured the disclaimer appears in all outputs, even with unexpected input.

Ethical and Privacy Considerations

Because the system touches on personal health information, several safeguards were considered:

• No Data Storage: Inputs are processed in memory only and not logged.

• Transparency: Repeated disclaimers prevent users from mistaking the app for a medical professional.

• Model Limitations: Users are informed that the AI may produce inaccuracies or omissions.

These principles align with responsible-AI guidelines such as explainability, transparency, and human

oversight.

Limitations

• Accuracy depends entirely on the LLM's training data.

• No real-time integration with verified clinical databases.

• Not suitable for emergency or critical medical situations.

Future Enhancements

• Integration with vetted medical knowledge bases for improved reliability.

• Multilingual interface and voice input/output for accessibility.

• Optional secure login to store personal history for repeat users.

• Enhanced analytics to measure common symptom trends (with privacy safeguards).

Conclusion

Health AI demonstrates the practical fusion of Python, Gradio, and a transformer-based LLM to create a responsive health-information platform. It illustrates how conversational AI can provide first-level

guidance while underscoring the need for professional medical consultation. This project can serve as a

foundation for future research into responsible AI in healthcare, balancing technological capability with

ethical safeguards.