

HEALTH AI INTELLIGENT HEALTH CARE ASSISTANT

PROJECT DOCUMENTATION

1. Introduction

Project Title: Health AI Intelligent Health Care Assistant

Team member: RAJKESH K

Team member : RODRIGUE T

Team member : MOHAMED SADHU H

Team member : DEEPIKA R

2. Project Overview

Purpose: The Medical AI Assistant is designed to provide informational support on possible medical conditions and personalized treatment suggestions. It serves as a tool for general health information and is not a substitute for professional medical advice.

Features:

Disease Prediction: Users can enter symptoms and receive suggestions for possible medical conditions and general medication advice.

Treatment Plan Generation: The assistant can generate personalized treatment plans based on a user's condition, age, gender, and medical history.

User Interface: The application uses a Gradio interface with separate tabs for each function.

LLM Integration: It uses the ibm-granite/granite-3.2-2b-instruct model for generating responses.

3. Solution Components

AI Model: The project uses ibm-granite/granite-3.2-2b-instruct for natural language understanding and generation.

Frontend: The frontend is a web interface built with the Gradio library. It includes textboxes for user input and output, a number field for age, a dropdown for gender, and buttons to trigger the AI analysis.

Backend: The logic is handled by a Python script that uses the Hugging Face Transformers library to interact with the LLM.

4. Technology Stack

Frameworks: Gradio.

Libraries: torch and transformers are used for loading and running the AI model.

API Endpoints: The code itself does not define explicit API endpoints, but the Gradio interface handles the communication between the UI and the backend functions (disease_prediction and treatment_plan).

5. Prompt Engineering

Purpose: Prompts are carefully crafted to guide the AI model to provide medical information. The `disease_prediction` and `treatment_plan` functions use specific prompts that include the user's input and a clear disclaimer about consulting a healthcare professional.

6. Deployment

The application can be launched and shared via a public URL using the `app.launch(share=True)` command. The provided document also shows that the deployment was in a Colab notebook.

7. API Endpoints

The application's functionality is handled by two main Python functions: `disease_prediction` and `treatment_plan`. These functions take user inputs and return AI-generated responses.

8. Authentication

The provided code does not include authentication. A warning is noted that the Hugging Face Hub token does not exist, but authentication is optional to access public models.

9. User Interface

The interface is created using `gr.Blocks()`. It features a main title and a disclaimer using `gr.Markdown` and is organized into two tabs with `gr.Tabs` and `gr.TabItem` for different functionalities. The layout uses `gr.Row` and `gr.Column` to arrange input and output components.

10. Testing

- `UnitTesting`– Prompt responses and ML models.
- `APITesting`– Swagger & Postman.
- `ManualTesting`– File uploads, summarization, anomaly detection.
- `EdgeCases`– Invalid inputs, empty PDFs, missing API keys.

11. Screenshot

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

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages  
Requirement already satisfied: torch in /usr/local/lib/python3.12/dist-packages (2.8.0)  
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (filelock)  
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (huggingface-hub)  
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (numpy)  
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (packaging)  
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (pyyaml)  
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (regex)  
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (requests)  
Requirement already satisfied: tokenizers<0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (tokenizers)  
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (safetensors)  
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (tqdm)  
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.12/dist-packages (typing-extensions)  
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (setuptools)  
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (sympy)  
Requirement already satisfied: networkx in /usr/local/lib/python3.12/dist-packages (networkx)  
Requirement already satisfied: Jinja2 in /usr/local/lib/python3.12/dist-packages (Jinja2)  
Requirement already satisfied: fsspec in /usr/local/lib/python3.12/dist-packages (fsspec)  
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (nvidia-cuda-nvrtc-cu12)  
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (nvidia-cuda-runtime-cu12)  
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (nvidia-cuda-cupti-cu12)  
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (nvidia-cudnn-cu12)  
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (nvidia-cublas-cu12)  
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (nvidia-cufft-cu12)  
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (nvidia-curand-cu12)  
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (nvidia-cusolver-cu12)  
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (nvidia-cusparse-cu12)
```



```
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 gene
    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 informa
    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  
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")  
  
        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)
```

➡ 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>)

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.

```
warnings.warn(
```

vocab.json: 777k/? [00:00<00:00, 6.02MB/s]

```
merges.txt: 442k/? [00:00<00:00, 8.93MB/s]
```

```
└─ added_tokens.json: 100% ██████████ 87.0/87.0 [00:00<00:00, 3.19kB/s]
```

```
config.json: 100% ██████████ 786/786 [00:00<00:00, 22.5kB/s]
```

```
model.safetensors.index.json: 29.8k/? [00:00<00:00, 3.11MB/s]
```

```
└─ model-00001-of- [REDACTED] 5.00G/5.00G [01:11<00:00, 76.2MB/s]
```

```
model-00002-of- 67.1M/67.1M [00:01<00:00, 53.3MB/s]
```

```

Loading checkpoint shards: 100%  2/2 [00:19<00:00, 8.11s/it]

```

Colab notebook detected. To show errors in colab notebook, set debug=True in launch(
* Running on public URL: <https://095cdbd76ff5073b14.gradio.live>

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run

Colab notebook detected. To show errors in colab notebook, click the icon above.
* Running on public URL: <https://44e8be4f8de382703a.gcpcolab.live>



This share link expires in 1 week. For free permanent hosting and GPU upgrades, run

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

- Occasional long response time for large PDFs.
- Forecasting limited to structured CSV data.
- Requires stable internet for IBM API access.

13. Future Enhancements

- Add voice-based interaction.
- Expand forecasting to include traffic & pollution data.
- Develop a mobile app version.
- Integrate with IoT smart sensors.
- Support multi-language outputs for local communities.