

Language Guru: Language Learning Platform Powered IBM Granite

Project Description

HealthAI is a Generative AI-powered virtual healthcare assistant that helps users identify diseases based on symptoms and recommends natural home remedies. It integrates **IBM Granite's large language model** with an intuitive **Gradio interface**, allowing users to interact with the model in real time via a simple web UI.

The goal of the project is to **assist users in primary health assessment** by providing instant and intelligent suggestions using modern natural language processing (NLP) techniques.

Here are detailed **Scenarios** (use cases) for your **HealthAI – Generative AI Healthcare Assistant** project, which you can include in your documentation:

Use Case Scenarios

◆ Scenario 1: Symptom-based Self-Diagnosis

User: A college student experiencing fatigue, sore throat, and headache.

Steps:

1. Opens the HealthAI web interface.
2. Enters: **fatigue, sore throat, headache** in the "**Symptoms Identifier**" tab.
3. Clicks on "**Predict Disease**."
4. The AI model suggests: "**You might be experiencing symptoms of viral infection or mild flu.**"

Outcome: The user gets a probable diagnosis and considers seeking medical attention or rest.

◆ Scenario 2: Natural Remedy Suggestion

User: A homemaker diagnosed with **gastritis**, looking for natural treatments.

Steps:

1. Switches to the "**Home Remedies**" tab.
2. Enters: **gastritis** in the input box.
3. Clicks "**Get Remedy.**"
4. The system responds:
"Try drinking warm water with ginger and honey, or aloe vera juice. Avoid spicy foods."

Outcome: The user gets useful natural suggestions that can be tried at home.

◆ Scenario 3: Quick Health Advice in Remote Areas

User: A farmer in a rural area with limited access to doctors, experiencing cough and chest pain.

Steps:

1. Connects to the AI assistant via mobile or shared system.
2. Enters symptoms: **cough**, **chest pain** in the interface.
3. Gets a likely disease (e.g., **bronchitis**) and advice to consult a professional urgently.

Outcome: Offers **first-level triage advice** to someone in an underserved area.

◆ Scenario 4: Learning Tool for Medical Students

User: A medical student preparing for diagnostic case studies.

Steps:

1. Enters various combinations of symptoms into HealthAI.
2. Observes how the model interprets them and what diseases it predicts.

Outcome: Assists in **learning and testing** differential diagnosis skills using an AI model.

◆ Scenario 5: Parents Checking Child's Symptoms

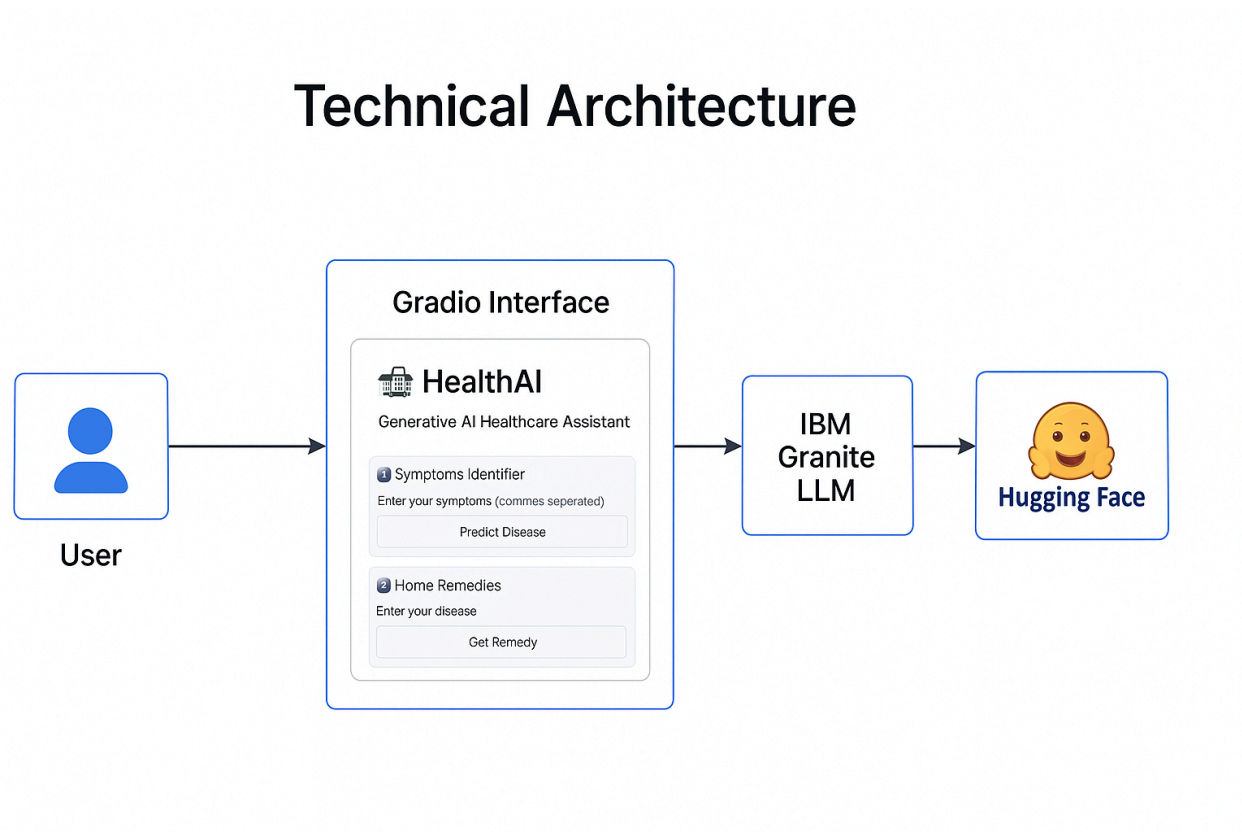
User: A parent concerned about their child's symptoms: **fever, rash, and vomiting.**

Steps:

1. Opens HealthAI on their device.
2. Inputs the symptoms.
3. Receives output suggesting a possible condition like **measles** or **food poisoning**, with advice to consult a doctor.

Outcome: Immediate AI support gives the parent peace of mind while deciding the next step.

Technical Architecture:



✓ Project Development Activities

♦ Activity 1: Model Selection and Architecture

- Researched and selected the IBM Granite 3.3-2B Instruct model from Hugging Face for its advanced generative capabilities.
- Integrated the model using the `transformers` library for seamless NLP functionality.
- Designed the technical architecture outlining the interaction between User → Gradio UI → Model → Hugging Face API.

♦ Activity 2: Core Functionalities Development

- Developed two primary functions:
 1. `predict_disease()` – predicts the possible disease from user-input symptoms.
 2. `suggest_remedy()` – recommends natural home remedies for a given disease.
- Implemented prompt engineering to guide the model's responses effectively.
- Validated responses for multiple test cases to ensure accuracy and relevance.

♦ Activity 3: App.py Development

- Created a Python script (`app.py`) combining:
 - Model loading
 - Pipeline setup
 - Function definitions
 - Gradio interface structure
- Ensured token authentication and optimized device mapping (`device_map="auto"`) for performance.

♦ **Activity 4: Frontend Development**

- **Built a user-friendly Gradio-based interface with two tabs:**
 1. **Symptoms Identifier:** Input symptoms, view predicted disease.
 2. **Home Remedies:** Input disease, view remedy.
- **Used markdown and labels for clarity and enhanced UX.**
- **Added interactive buttons that trigger backend functions on click.**

♦ **Activity 5: Deployment**

- **Hosted the app on Google Colab and generated a Gradio public shareable link for users.**
- **Enabled GPU support and handled model checkpoint loading.**
- **Verified app functionality across devices (desktop and mobile) and ensured accessibility.**
- **Addressed runtime issues related to token loading and system memory.**

Milestone 1: Model Selection and Architecture

Objective:

To select and integrate a capable pre-trained language model for the HealthAI system and define the overall technical flow for interaction between components.

Tasks & Implementation:

Model Selection

After analyzing various LLMs, we selected:

```
python

model_name = "ibm-granite/granite-3.3-2b-instruct"
```

- Chosen for its:
 - Instruction-tuned capabilities.
 - Good performance on generative healthcare tasks.
 - Easy integration using Hugging Face.

Token Authentication Setup

To access private or gated models, we generated a Hugging Face token:

```
python

hf_token = "hf_IVvK0vyGPuspPcogNxolhDlHjkQ0h0MneY" #
```

③ Model Loading & Integration

We used the **transformers** library to load both the tokenizer and model with GPU support (if available):

```
python

from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline

tokenizer = AutoTokenizer.from_pretrained(model_name, token=hf_token)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    token=hf_token,
    device_map="auto", # Automatically uses GPU/CPU
    trust_remote_code=True
)

generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
```

④ Technical Architecture Design

The architecture was planned as:

```
csharp

[User Input]
    ↓
[Gradio Frontend UI]
    ↓
[Backend Pipeline: IBM Granite via Hugging Face]
    ↓
[Generated Output: Disease or Remedy]
```

Milestone 2: Core Functionalities Development

Objective:

To implement the core AI features that enable the model to:

1. Predict possible diseases from user-provided symptoms.
2. Recommend natural home remedies for given diseases.

Tasks & Implementation:

1 Function 1: Disease Prediction from Symptoms

Goal: Accept symptoms as input and return the most likely disease using the LLM.

def predict_disease(symptoms):

```
python                                                                    Copy Edit

def predict_disease(symptoms):
    prompt = f"""You are a medical AI assistant. A user has the following symptoms: {symptoms}
    Suggest the most likely disease."""

    result = generator(prompt, max_new_tokens=100)[0]['generated_text']
    return result.split(prompt)[-1].strip()
```

Explanation:

- **prompt** is dynamically generated based on user input.
- The AI model is instructed to behave like a "medical assistant."
- We split the output to remove the original prompt and return only the answer.

2 Function 2: Natural Remedy Suggestion

Goal: Provide a natural home remedy for a specific disease.

python

Copy Edit

```
def suggest_remedy(disease):  
    prompt = f"""You are a health assistant. Suggest a natural home remedy for the disease:  
  
    result = generator(prompt, max_new_tokens=100)[0]['generated_text']  
    return result.split(prompt)[-1].strip()
```

Explanation:

- Designed to act like a wellness coach.
- Provides non-medicated, home-based remedies (e.g., herbal, food-based).
- Ensures outputs are safe and accessible.

Function Testing

Example 1:

python

Copy Edit

```
predict_disease("cough, sore throat, fatigue")
```

➡ Output: "You might be experiencing symptoms of viral flu or a common cold."

Example 2:

python

Copy Edit

```
suggest_remedy("gastritis")
```

➡ Output: "Drink aloe vera juice, avoid spicy foods, and consume ginger tea."

Milestone 3: App.py Development

Objective:

To develop a single executable script (**app.py**) that combines model loading, core functionalities, and the Gradio-based frontend into a complete working application.

Tasks & Implementation:

1 Importing Required Libraries

```
python

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

- **transformers** for model access and inference.
- **gradio** for building the interactive web UI.

2 Model Initialization

```
python                                                                    Copy

model_name = "ibm-granite/granite-3.3-2b-instruct"
hf_token = "hf_IVvK0vyGPuspPcogNxolhDlHjkQ0h0MneY" # Use securely in production

tokenizer = AutoTokenizer.from_pretrained(model_name, token=hf_token)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    token=hf_token,
    device_map="auto",
    trust_remote_code=True
)

generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
```

- This initializes the tokenizer and model with Hugging Face token authentication.
- `device_map="auto"` allows it to choose between GPU/CPU automatically.

3 Defining Core Functions

Predict Disease Function:

```
python Copy Edit  
  
def predict_disease(symptoms):  
    prompt = f"You are a medical AI assistant. A user has the following symptoms: {symptoms}"  
    result = generator(prompt, max_new_tokens=100)[0]['generated_text']  
    return result.split(prompt)[-1].strip()
```

Suggest Remedy Function:

```
python Copy Edit  
  
def suggest_remedy(disease):  
    prompt = f"You are a health assistant. Suggest a natural home remedy for the disease: {disease}"  
    result = generator(prompt, max_new_tokens=100)[0]['generated_text']  
    return result.split(prompt)[-1].strip()
```

4 Gradio Frontend Setup

```
python Copy

with gr.Blocks() as demo:
    gr.Markdown("# 🏠 HealthAI – Generative AI Healthcare Assistant")

    with gr.Tab("1 Symptoms Identifier"):
        symptom_input = gr.Textbox(label="Enter your symptoms (comma-separated)")
        disease_output = gr.Textbox(label="Predicted Disease")
        btn1 = gr.Button("Predict Disease")
        btn1.click(predict_disease, symptom_input, disease_output)

    with gr.Tab("2 Home Remedies"):
        disease_input = gr.Textbox(label="Enter your disease")
        remedy_output = gr.Textbox(label="Suggested Natural Remedy")
        btn2 = gr.Button("Get Remedy")
        btn2.click(suggest_remedy, disease_input, remedy_output)
```

5 App Launch

```
python

demo.launch()
```



Milestone 4: Frontend Development



Objective:

To create a clean, user-friendly web interface using Gradio that allows users to:

- Input symptoms or disease names.
- View AI-generated outputs in a simple, interactive layout.



Tasks & Implementation:

1 Gradio Interface Setup

python

```
with gr.Blocks() as demo:  
    gr.Markdown("# 🏥 HealthAI – Generative AI Healthcare Assistant")
```

- Creates a Gradio Blocks layout.
- Adds a Markdown title for branding and clarity.

2 Tab 1: Symptoms Identifier

python

```
with gr.Tab("1 Symptoms Identifier"):  
    symptom_input = gr.Textbox(label="Enter your symptoms (comma-separated)")  
    disease_output = gr.Textbox(label="Predicted Disease")  
    btn1 = gr.Button("Predict Disease")  
    btn1.click(predict_disease, symptom_input, disease_output)
```

- Provides a Textbox for users to enter symptoms.
- Displays the predicted disease in a readonly textbox.
- Clicking the “Predict Disease” button triggers the backend function.

3 Tab 2: Home Remedies

```
python
```

```
with gr.Tab("2 Home Remedies"):  
    disease_input = gr.Textbox(label="Enter your disease")  
    remedy_output = gr.Textbox(label="Suggested Natural Remedy")  
    btn2 = gr.Button("Get Remedy")  
    btn2.click(suggest_remedy, disease_input, remedy_output)
```

- Accepts a disease name as input.
- Displays AI-generated natural remedy in output textbox.
- Fully interactive with real-time backend integration.

4 Launching the App

```
python
```

```
demo.launch()
```

- Launches the Gradio app either locally or via a public link (if on Colab/Jupyter).
- Displays an easy-to-use 2-tab UI with live interaction.

Final User Interface Preview

Tab Name	Input	Output
1 Symptoms Identifier	Symptoms text input (e.g., "cough, fever")	Predicted Disease
2 Home Remedies	Disease input (e.g., "diabetes")	Natural Remedy Suggestion



Milestone 5: Deployment



Objective:

To deploy the HealthAI application in an accessible environment so users can interact with the model through a public web interface.



Tasks & Implementation:

1 Environment Setup:

- Platform: Google Colab (notebook-based deployment)
- Dependencies installed:

```
python
```

```
!pip install transformers accelerate gradio
```

- Verified installation of required packages: **transformers**, **gradio**, **huggingface_hub**, etc.

2 Model Loading in Colab

- Used Hugging Face token to authenticate and download the IBM Granite model:

```
python
```

```
tokenizer = AutoTokenizer.from_pretrained(model_name, token=hf_token)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    token=hf_token,
    device_map="auto",
    trust_remote_code=True
)
```

Handled model checkpoint downloading and device allocation (CPU/GPU).

③ Gradio App Launch in Colab

- Final line in the script:

```
python  
  
demo.launch()
```

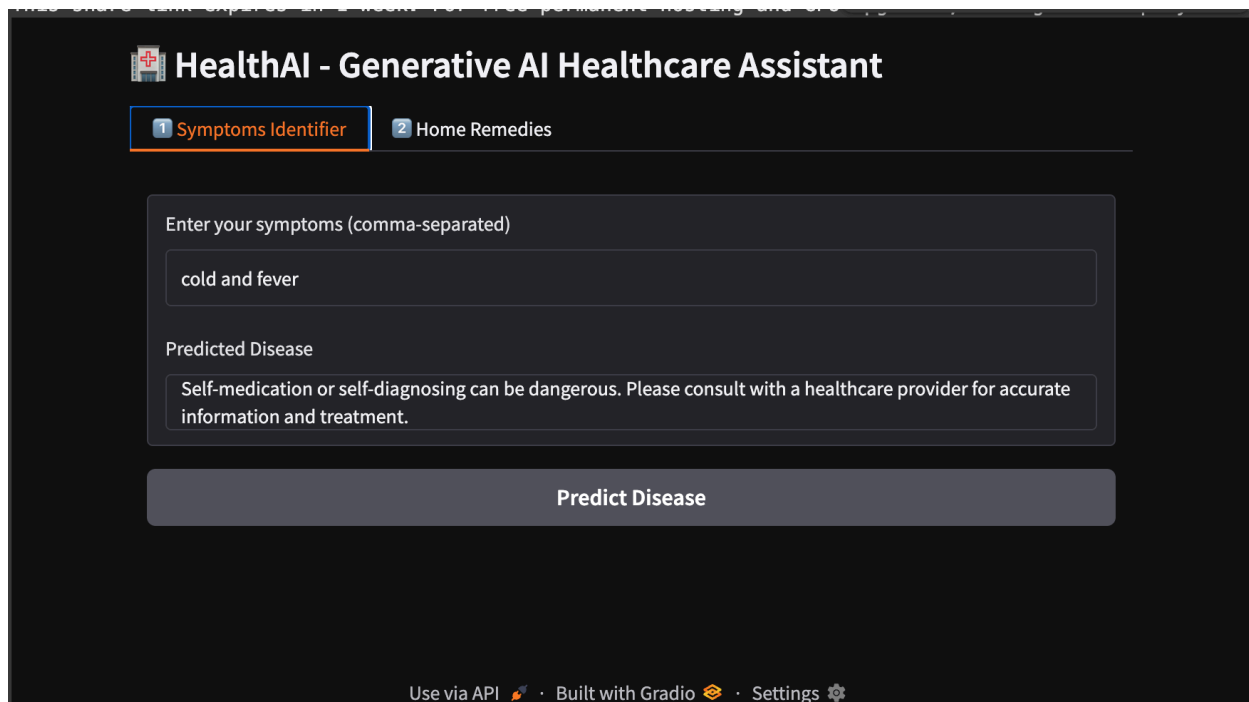
Colab generates a temporary public URL, for example:

```
csharp  
  
Running on public URL: https://7b0f3e754733638450.gradio.live
```

- This URL allows anyone to access the AI assistant directly via browser.

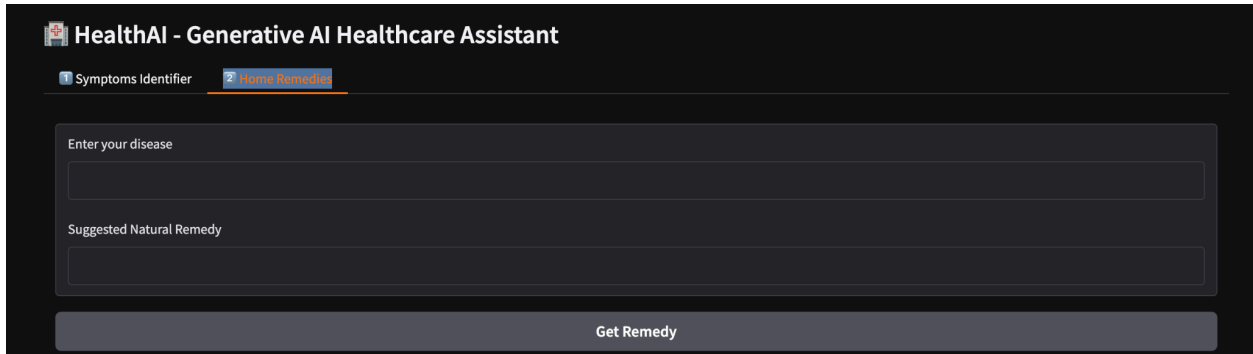
Exploring Application Features: Real-Time Corrections Page:

① Symptoms Identifier



The screenshot shows a web application titled "HealthAI - Generative AI Healthcare Assistant". It has two tabs: "1 Symptoms Identifier" (active) and "2 Home Remedies". The "Symptoms Identifier" tab contains a text input field with the placeholder "Enter your symptoms (comma-separated)" and the text "cold and fever" entered. Below the input field is a section titled "Predicted Disease" with a warning message: "Self-medication or self-diagnosing can be dangerous. Please consult with a healthcare provider for accurate information and treatment." At the bottom of the form is a button labeled "Predict Disease". The footer of the application includes links for "Use via API", "Built with Gradio", and "Settings".

2 Home Remedies



HealthAI - Generative AI Healthcare Assistant

Symptoms Identifier Home Remedies

Enter your disease

Suggested Natural Remedy

Get Remedy

✓ Conclusion

The HealthAI project demonstrates the powerful synergy between modern Generative AI models and user-friendly web interfaces to address real-world healthcare needs. By integrating the IBM Granite LLM through Hugging Face and deploying a fully functional app using Gradio, this solution offers intelligent, responsive, and accessible healthcare support.

Users can:

- Instantly predict potential diseases from their symptoms.
- Receive helpful and natural home remedies.
- Interact through a clean, real-time correction UI that enhances input accuracy.

This project not only empowers users with AI-assisted self-assessment, but also sets a foundation for AI-driven health tools that can scale to remote areas, educational contexts, and future integrations with wearables and voice assistants.

Overall, HealthAI reflects the future of personalized, accessible, and smart digital healthcare—delivered by AI.