# SmartSDLC – AI-Enhanced Software Development Lifecycle

Project Documentation

## 1. Introduction

- Project Title: SmartSDLC – AI-Enhanced Software Development Lifecycle
- Team Leader: Kavimalar. E
- Team Members
    1. Lavanya. M
    2. Arthi. R
    3. Oviyaa. M

## 2. Project Overview

• **Purpose:**

SmartSDLC leverages IBM Granite models (via Hugging Face) to accelerate software development. It allows users to upload PDFs, generate clear requirements, convert prompts into code, create tests, fix bugs, write documentation, and interact with an AI helper. The project is deployed on Google Colab using Granite for easy setup and reliable performance.

• **Features:**

- PDF upload and automatic requirement extraction
- Prompt-to-code generation
- Automated testing and bug fixing
- Documentation generation
- AI-powered chat assistant for development support
- Google Colab deployment with GPU acceleration
- Integration with GitHub for version control

## 3. Architecture

Frontend: Gradio interface for interactive use.

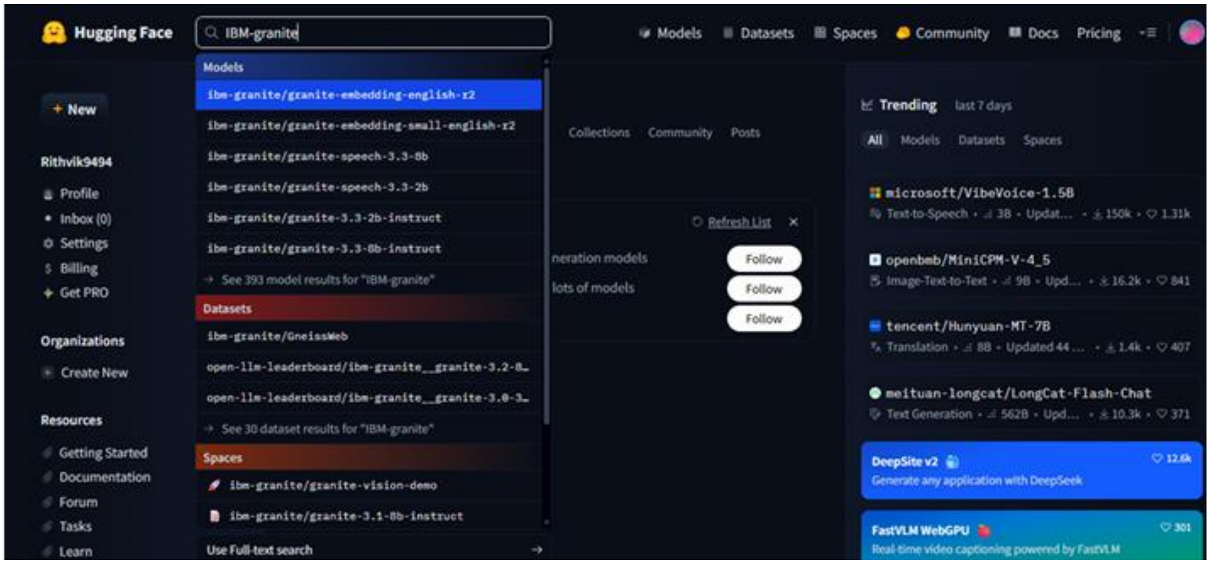
Backend: Python with Hugging Face IBM Granite models for AI-powered generation and analysis.

Deployment: Google Colab (T4 GPU support).

Version Control: GitHub repository for code storage and collaboration.

## 4. Setup Instructions

Upload code to GitHub repository search for “IBM-Granite models”



IBM-granite

Models

ibm-granite/granite-embedding-english-r2
ibm-granite/granite-embedding-small-english-r2
ibm-granite/granite-speech-3.3-8b
ibm-granite/granite-speech-3.3-2b
ibm-granite/granite-3.3-2b-instruct
ibm-granite/granite-3.3-8b-instruct
→ See 393 model results for "IBM-granite"

Datasets

ibm-granite/GneissWeb
open-llm-leaderboard/ibm-granite__granite-3.2-8…
open-llm-leaderboard/ibm-granite__granite-3.0-3…
→ See 30 dataset results for "IBM-granite"

Spaces

ibm-granite/granite-vision-demo
ibm-granite/granite-3.1-8b-instruct

Trending last 7 days

microsoft/VibeVoice-1.5B
Text-to-Speech · 3B · Updat… · 150k · 1.31k

openbmb/MiniCPM-V-4_5
Image-Text-to-Text · 9B · Upd… · 16.2k · 841

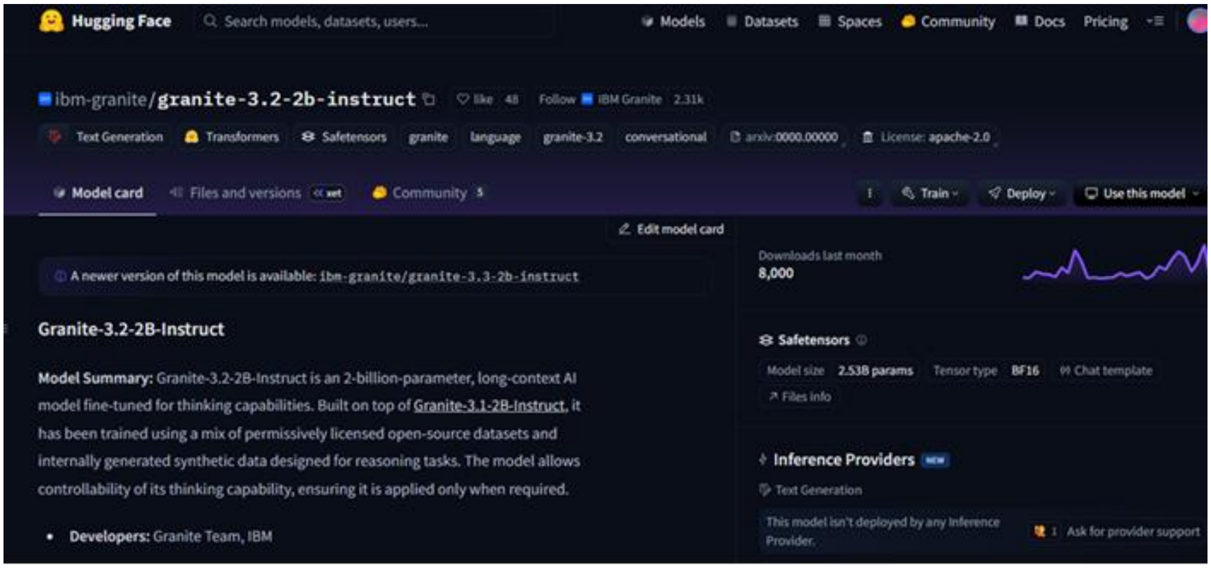
tencent/Hunyuan-MT-7B
Translation · 8B · Updated 44 … · 1.4k · 407

meituan-longcat/LongCat-Flash-Chat
Text Generation · 562B · Upd… · 10.3k · 371

DeepSite v2 12.6k
Generate any application with DeepSeek

FastVLM WebGPU 301
Real-time video captioning powered by FastVLM

● Here for this project we are using “granite-3.2-2b-instruct” which is compatible fast and light weight.



ibm-granite/granite-3.2-2b-instruct

A newer version of this model is available: ibm-granite/granite-3.3-2b-instruct

Granite-3.2-2B-Instruct

**Model Summary:** Granite-3.2-2B-Instruct is an 2-billion-parameter, long-context AI model fine-tuned for thinking capabilities. Built on top of Granite-3.1-2B-Instruct, it has been trained using a mix of permissively licensed open-source datasets and internally generated synthetic data designed for reasoning tasks. The model allows controllability of its thinking capability, ensuring it is applied only when required.

- **Developers:** Granite Team, IBM

Model size 2.53B params Tensor type BF16

This model isn't deployed by any Inference Provider.

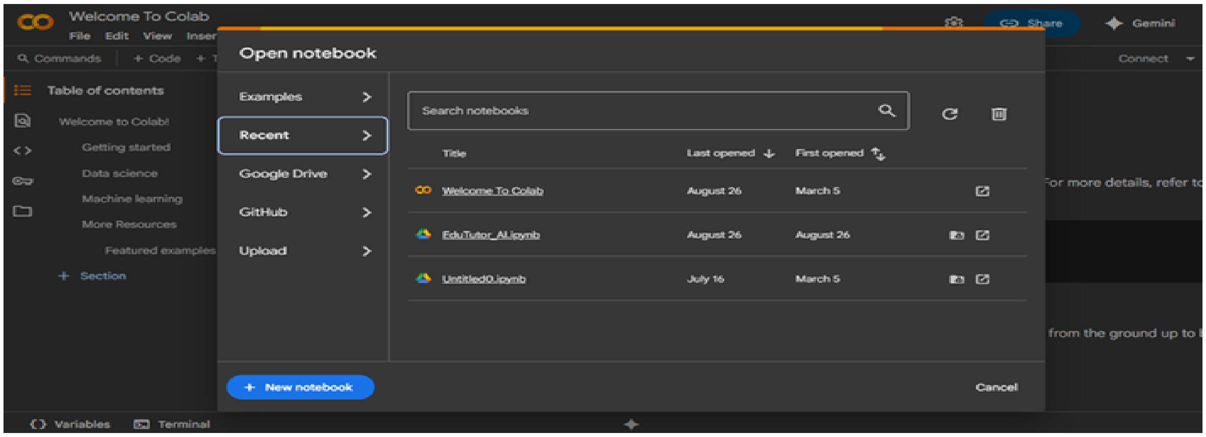
**Google colab(with T4 GPU):**

Search for “Google collab” in any browser.

● Click on the first link

https://colab.research.google.com/

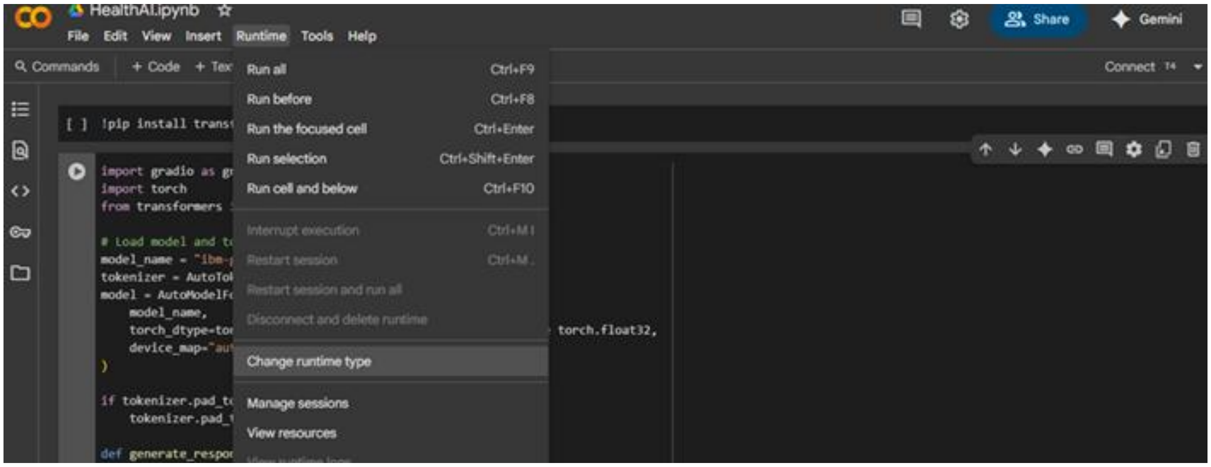
then click on “Files” and then “Open Notebook”.



| Title | Last opened | First opened |
|---|---|---|
| Welcome To Colab | August 26 | March 5 |
| EduTutor_AI.ipynb | August 26 | August 26 |
| Untitled0.ipynb | July 16 | March 5 |

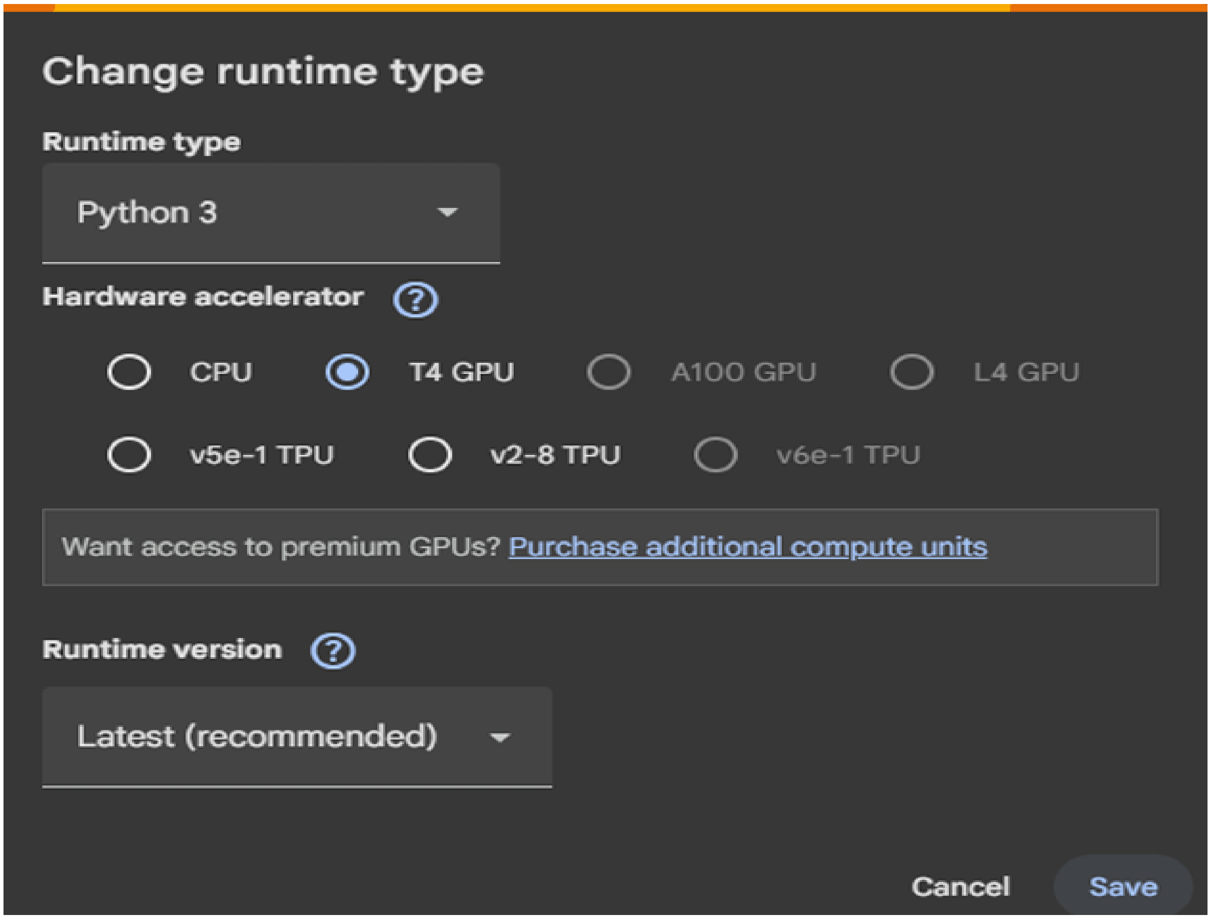
- Click on “New Notebook”

- Change the title of the notebook “Untitled” to “Health AI”. Then click on “Runtime”, then go to “Change Runtime Type”. 8



```
[ ] !pip install trans

import gradio as gr
import torch
from transformers

# Load model and to
model_name = "ibm-
tokenizer = AutoTok
model = AutoModelFo
    model_name,
    torch_dtype=tor                    torch.float32,
    device_map="au
)

if tokenizer.pad_to
    tokenizer.pad_t

def generate_respo
```

- Choose “T4 GPU” and click on “Save”



## Change runtime type

**Runtime type**

Python 3

**Hardware accelerator**

○ CPU ● T4 GPU ○ A100 GPU ○ L4 GPU

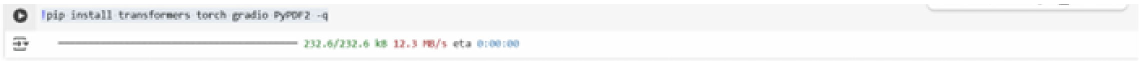
○ v5e-1 TPU ○ v2-8 TPU ○ v6e-1 TPU

Want access to premium GPUs? Purchase additional compute units

**Runtime version**

Latest (recommended)

● Then run this command in the first cell “!pip install transformers torch gradio PyPDF2 -q”. To install the required libraries to run our application.



```
!pip install transformers torch gradio PyPDF2 -q
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 12.3 MB/s eta 0:00:00
```

● Then run the rest of the code in the next cell. 9 10

## 5. Folder Structure

- notebooks/ – Google Colab notebooks for running the project
- models/ – IBM Granite model integrations
- app.py – Main application script

## 6. Running the Application

1. Open the project notebook in Google Colab.

2. Enable T4 GPU runtime.

3. Install dependencies using '!pip install transformers torch gradio PyPDF2 -q'.

4. Run all cells to launch the Gradio interface.

## 7. API Documentation

The system primarily interacts through a Gradio UI rather than REST APIs. Future enhancements may include API endpoints for requirement extraction, code generation, and testing.

## 8. Authentication

Currently runs in an open Colab environment. Secure deployments may include Hugging Face API key authentication and GitHub integration tokens.

## 9. User Interface

The project uses Gradio as the interface, allowing users to:

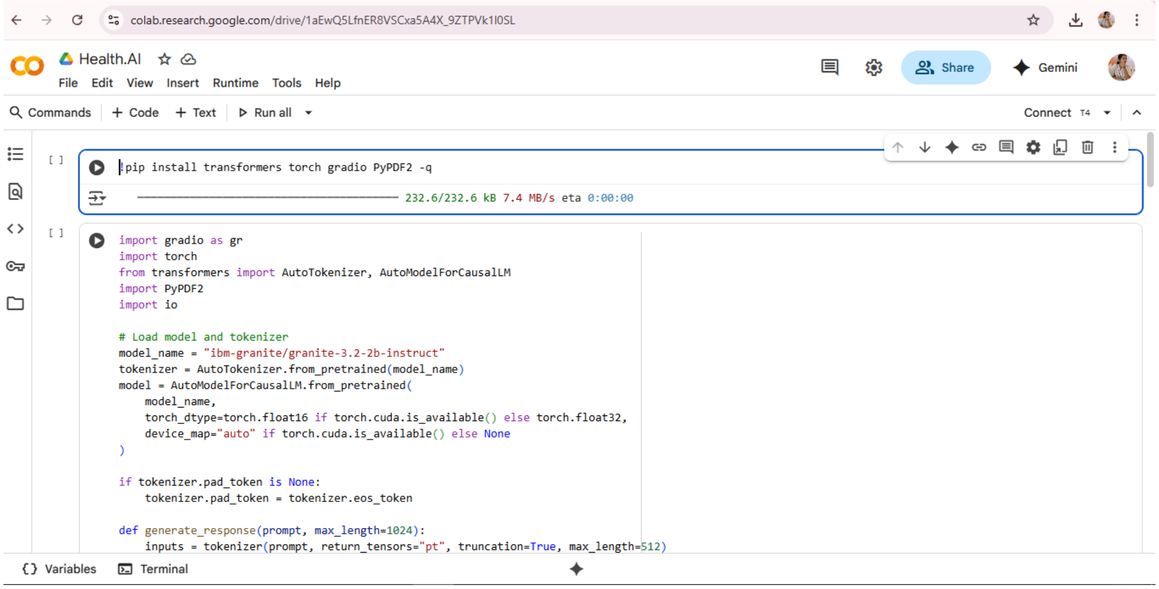
- Upload project PDFs
- Interact with AI assistant through chat
- View generated requirements, code, and test cases
- Access bug fixes and documentation outputs

## 10. Testing

Testing was performed manually by running workflows in Colab and validating output correctness. Future work may include automated unit tests and integration tests.

## 11. Screenshots

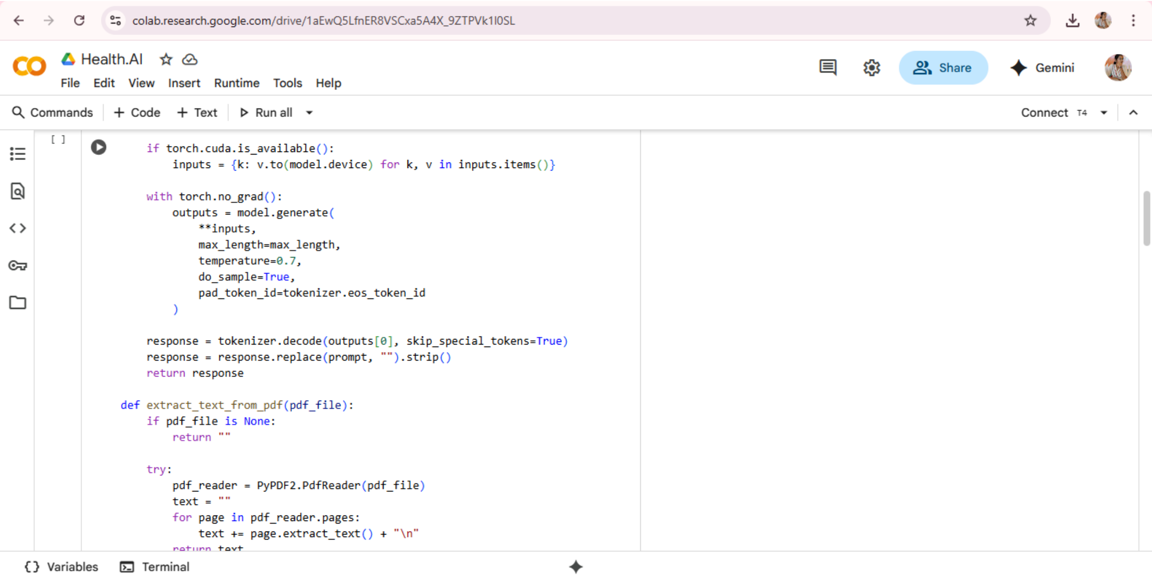
### Coding


colab.research.google.com/drive/1aEwQ5LfnER8VSCxa5A4X_9ZTPVk1I0SL
Health.AI
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands + Code + Text Run all
Connect T4
Variables
Terminal

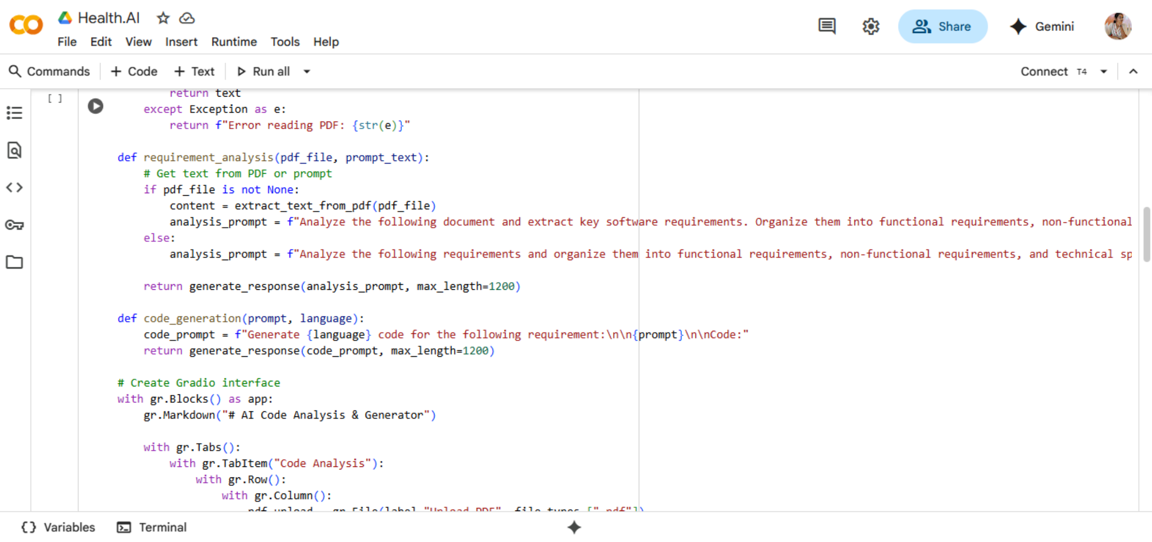
```
!pip install transformers torch gradio PyPDF2 -q
```

```
    ---------------------------------------- 232.6/232.6 kB 7.4 MB/s eta 0:00:00
```

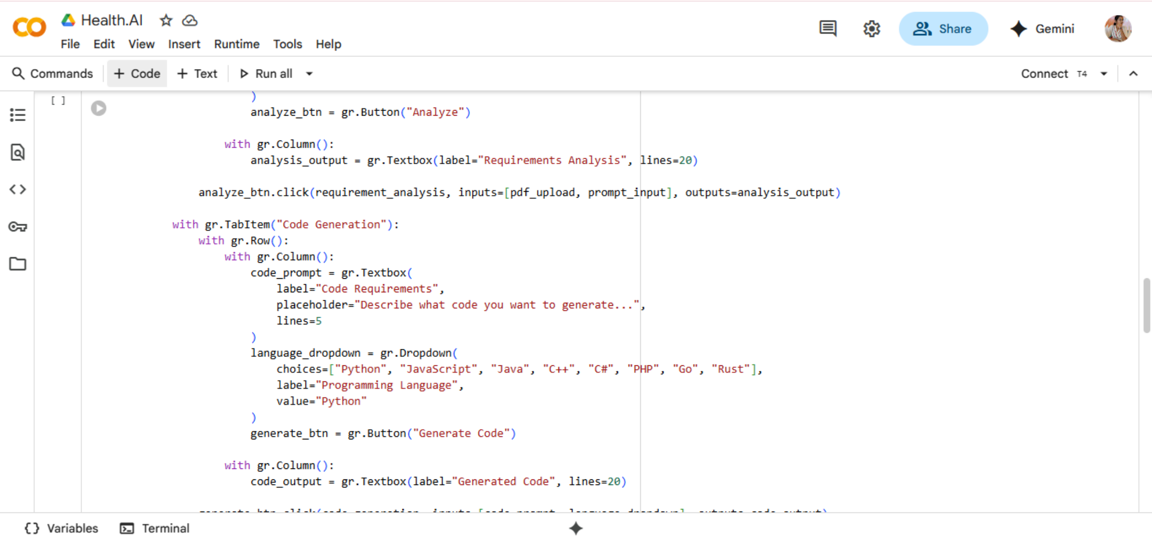
```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
import PyPDF2
import io

# 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)
```


colab.research.google.com/drive/1aEwQ5LfnER8VSCxa5A4X_9ZTPVk1I0SL
Health.AI
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
Connect T4
[ ]
Variables
Terminal

```
        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 extract_text_from_pdf(pdf_file):
        if pdf_file is None:
            return ""

        try:
            pdf_reader = PyPDF2.PdfReader(pdf_file)
            text = ""
            for page in pdf_reader.pages:
                text += page.extract_text() + "\n"
            return text
```

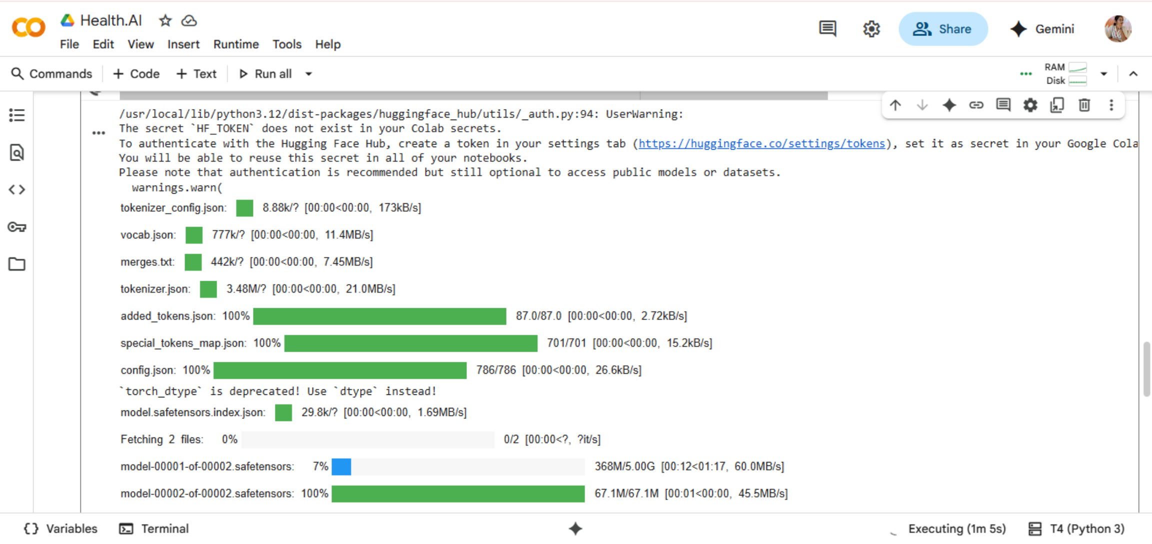


```
            return text
        except Exception as e:
            return f"Error reading PDF: {str(e)}"

    def requirement_analysis(pdf_file, prompt_text):
        # Get text from PDF or prompt
        if pdf_file is not None:
            content = extract_text_from_pdf(pdf_file)
            analysis_prompt = f"Analyze the following document and extract key software requirements. Organize them into functional requirements, non-functional
        else:
            analysis_prompt = f"Analyze the following requirements and organize them into functional requirements, non-functional requirements, and technical sp

        return generate_response(analysis_prompt, max_length=1200)

    def code_generation(prompt, language):
        code_prompt = f"Generate {language} code for the following requirement:\n\n{prompt}\n\nCode:"
        return generate_response(code_prompt, max_length=1200)

    # Create Gradio interface
    with gr.Blocks() as app:
        gr.Markdown("# AI Code Analysis & Generator")

        with gr.Tabs():
            with gr.TabItem("Code Analysis"):
                with gr.Row():
                    with gr.Column():
                        pdf_upload = gr.File(label="Upload PDF", file_types=[".pdf"])
```

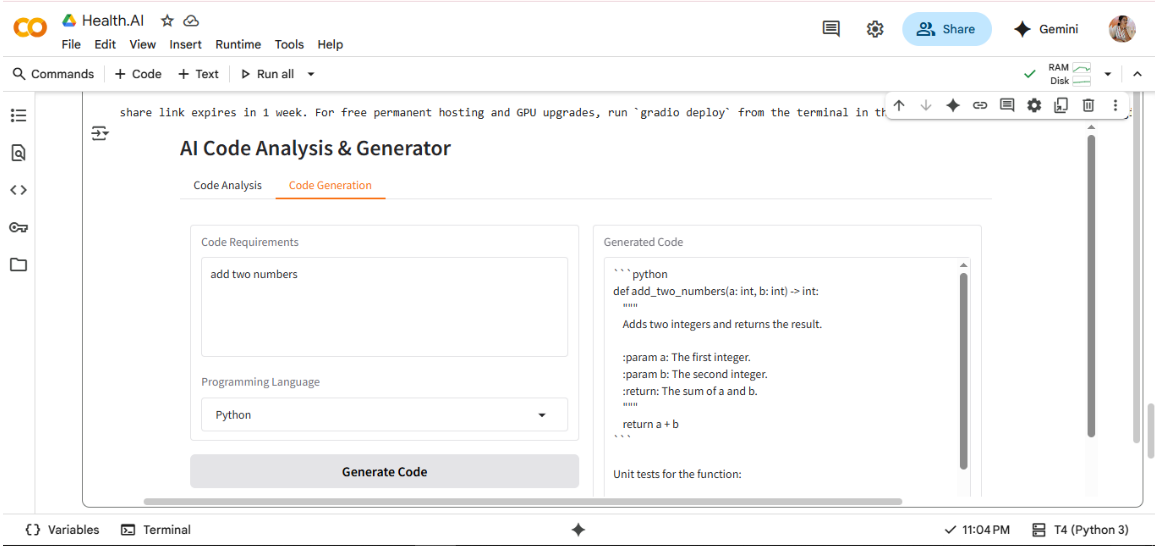


```
            )
            analyze_btn = gr.Button("Analyze")

        with gr.Column():
            analysis_output = gr.Textbox(label="Requirements Analysis", lines=20)

    analyze_btn.click(requirement_analysis, inputs=[pdf_upload, prompt_input], outputs=analysis_output)

with gr.TabItem("Code Generation"):
    with gr.Row():
        with gr.Column():
            code_prompt = gr.Textbox(
                label="Code Requirements",
                placeholder="Describe what code you want to generate...",
                lines=5
            )
            language_dropdown = gr.Dropdown(
                choices=["Python", "JavaScript", "Java", "C++", "C#", "PHP", "Go", "Rust"],
                label="Programming Language",
                value="Python"
            )
            generate_btn = gr.Button("Generate Code")

        with gr.Column():
            code_output = gr.Textbox(label="Generated Code", lines=20)

    generate_btn.click(code_generation, inputs=[code_prompt, language_dropdown], outputs=code_output)
```

## Project outcome:



```
/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 Cola
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, 173kB/s]
vocab.json:  777k/? [00:00<00:00, 11.4MB/s]
merges.txt:  442k/? [00:00<00:00, 7.45MB/s]
tokenizer.json:  3.48M/? [00:00<00:00, 21.0MB/s]
added_tokens.json: 100%  87.0/87.0 [00:00<00:00, 2.72kB/s]
special_tokens_map.json: 100%  701/701 [00:00<00:00, 15.2kB/s]
config.json: 100%  786/786 [00:00<00:00, 26.6kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:  29.8k/? [00:00<00:00, 1.69MB/s]
Fetching 2 files:   0%  0/2 [00:00<?, ?it/s]
model-00001-of-00002.safetensors:   7%  368M/5.00G [00:12<01:17, 60.0MB/s]
model-00002-of-00002.safetensors: 100%  67.1M/67.1M [00:01<00:00, 45.5MB/s]
```



```
share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in th
```

AI Code Analysis & Generator

Code Requirements: add two numbers

Programming Language: Python

Generated Code:

~~~
```python
def add_two_numbers(a: int, b: int) -> int:
  """
  Adds two integers and returns the result.

  :param a: The first integer.
  :param b: The second integer.
  :return: The sum of a and b.
  """
  return a + b
```

Unit tests for the function:
~~~

## 12. Known Issues

- Dependent on stable internet connection (Colab environment)
- Limited compute resources on free Colab tier
- Granite model download time may vary

## 13. Future Enhancements

- Add REST API endpoints for integration
- Improve UI with more customization options
- Integrate automated CI/CD pipeline with GitHub
- Enable persistent storage for user projects