MIRACLE
SOFTWARE SYSTEMS

Integrating Gemini APIs with Python and Building a Chatbot App with Streamlit

Open Lab | Digital Summit 2025

Goal

In this OpenLab, you will learn to create a simple, user-friendly chat bot using **Streamlit** and powered by **Google Cloud's Gemini API**. Here's what you'll be working on:

- **Streamlit** is an easy-to-use platform that helps you build interactive applications quickly. With it, you'll create a chatbot where users can ask questions and get accurate answers.
- The chatbot will use **Gemini's AI**, a powerful language model by Google, to understand the user's questions and provide intelligent, helpful responses.

By the end of the OpenLab, you'll have built a chatbot that anyone can use to ask questions and get smart answers, all with a simple interface!

Pre-Requisites

The following installations are required to complete this lab and run successfully,

- Google Account
- Python Installation
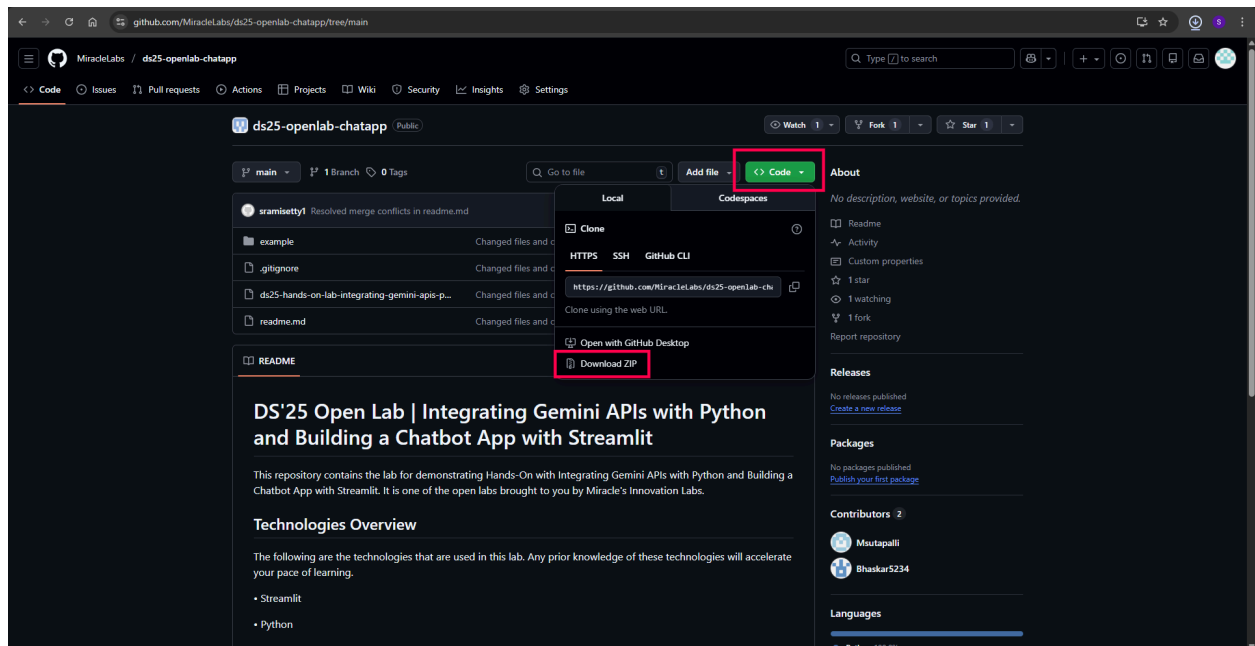- Any Text Editor(VS Code/Pycharm/Notepad++)

Technology Involved
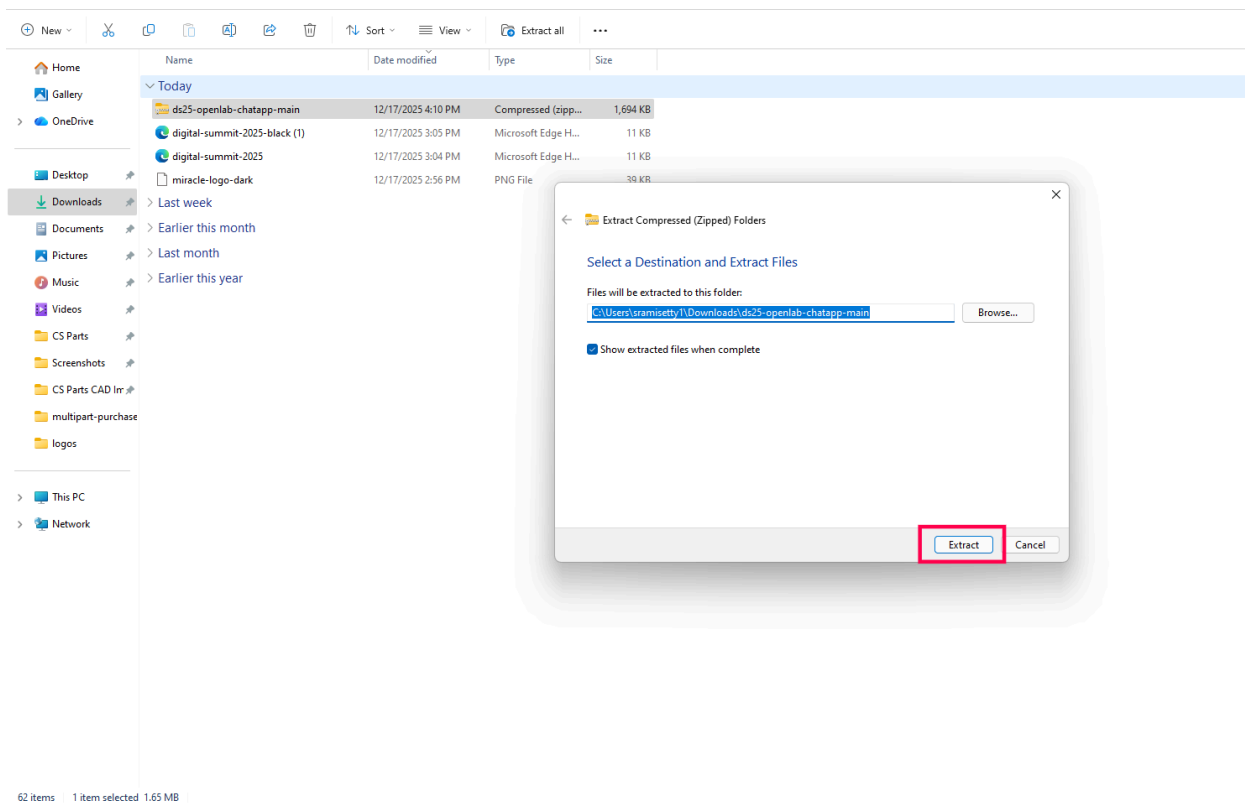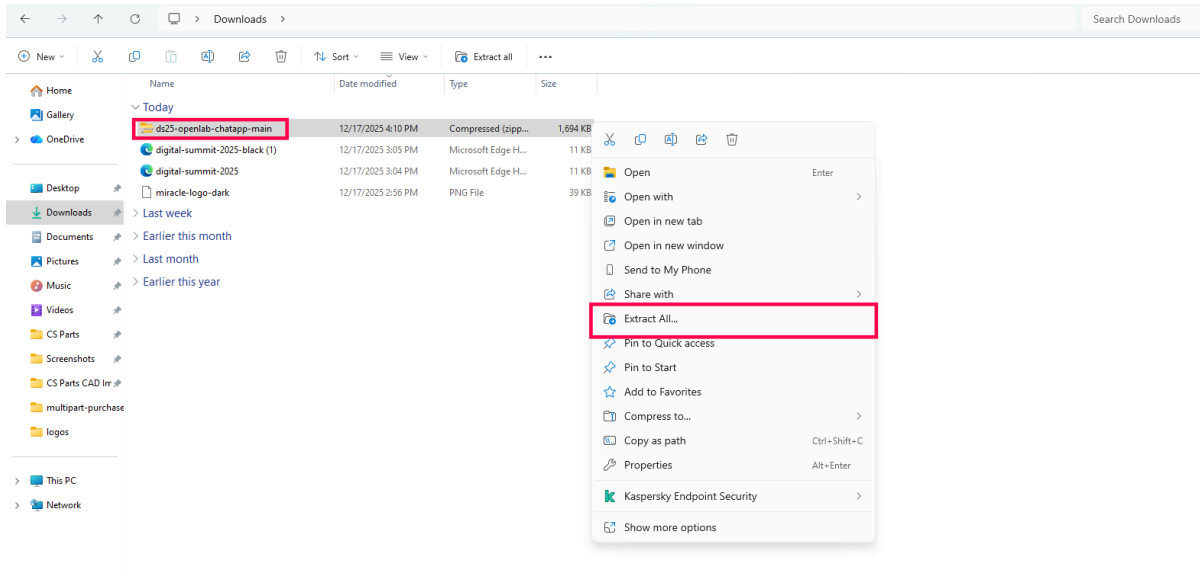
- Python
- Streamlit (HTML + CSS)

Step 1 | Download Code Repository
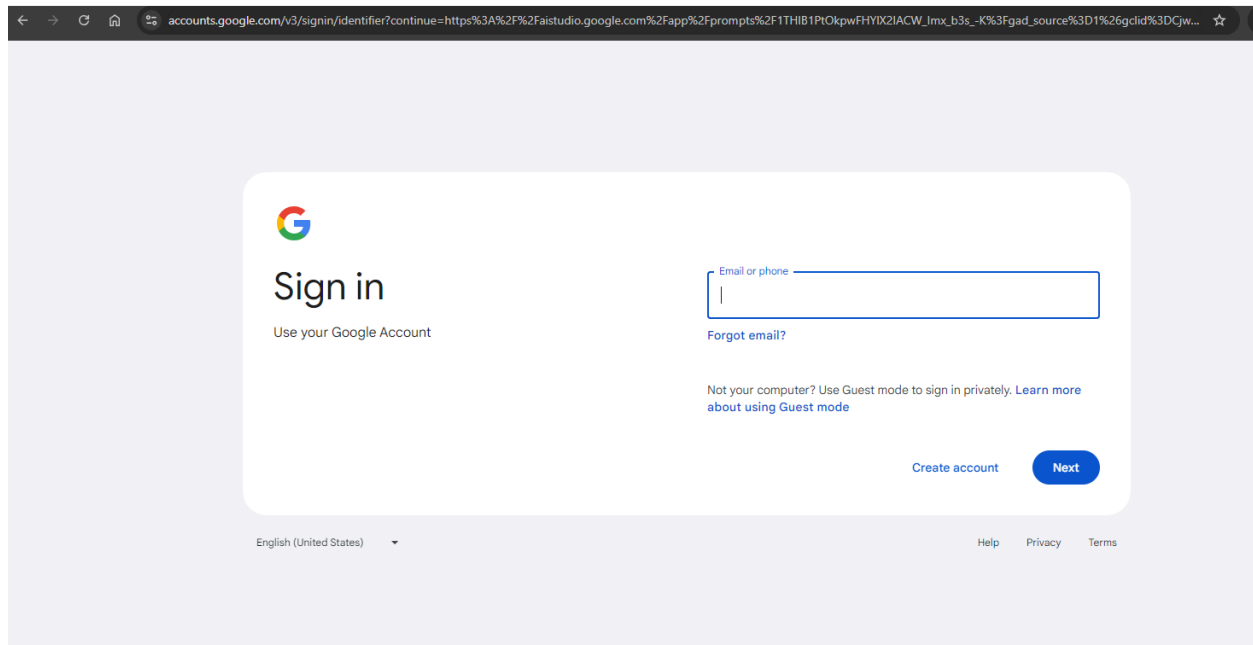## Get Started

- Download the code from the following GitHub Repo link
  https://github.com/MiracleLabs/ds25-openlab-chatapp

- After downloading, unzip it
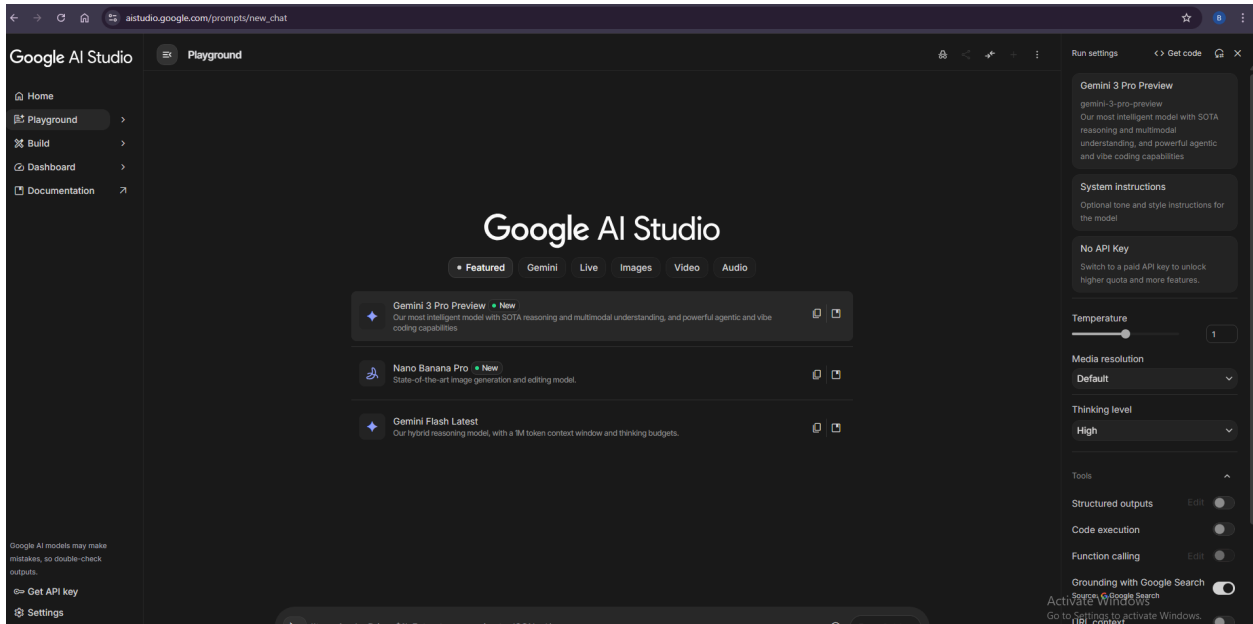
www.miraclesoft.com

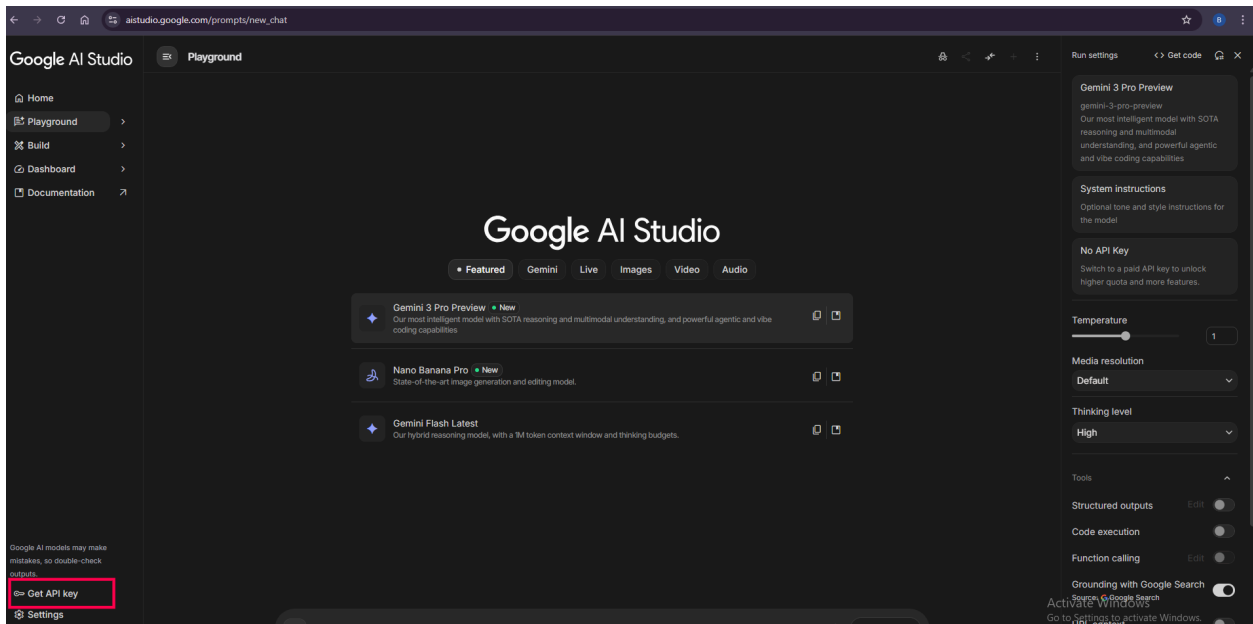www.miraclesoft.com

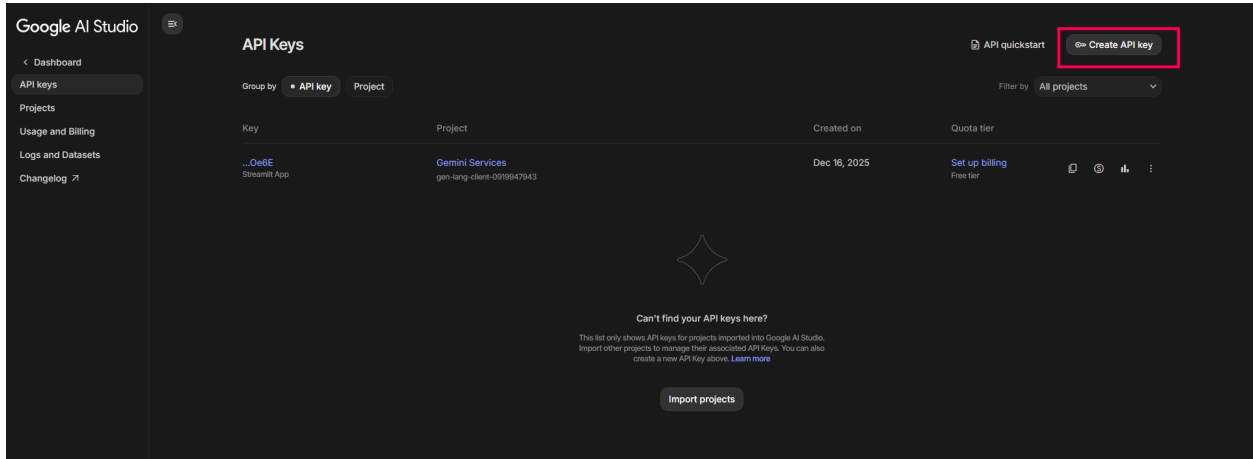## Step 2 | Access Google AI Studio

- Click on [Google AI Studio](#)

- After logging into your Google account, you'll be redirected to the AI Studio dashboard, where you need to accept the Terms of Service and click **"Continue"** to proceed.
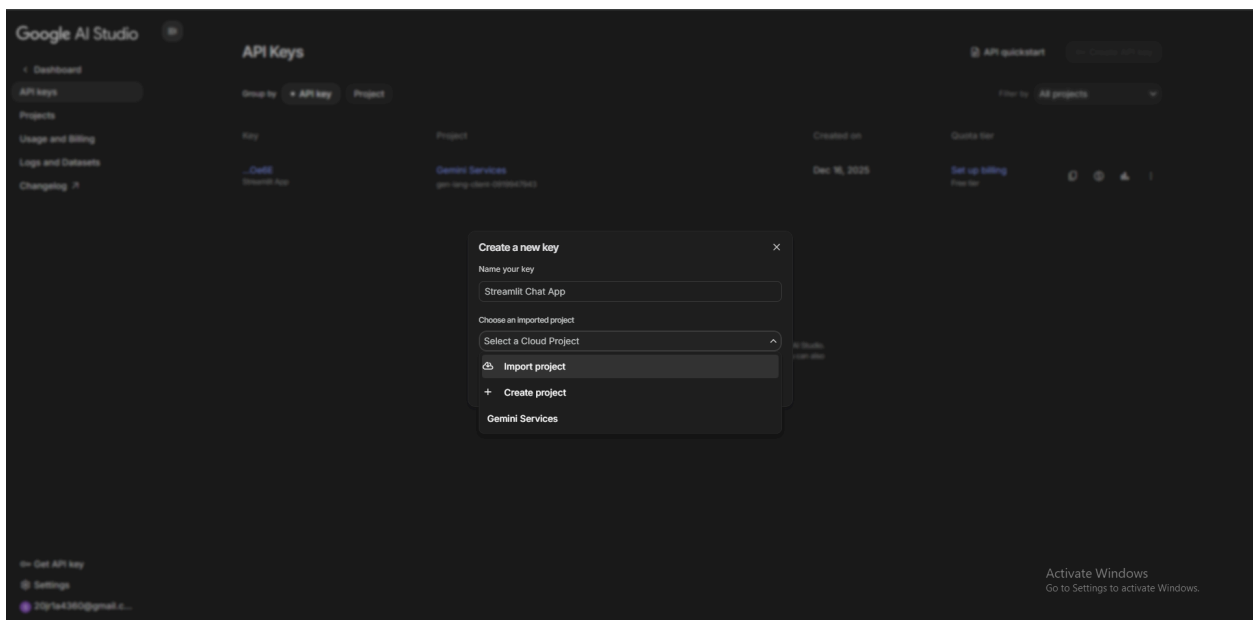
- Click on '**Get API Key**' in the top left corner to proceed to the screen shown below.
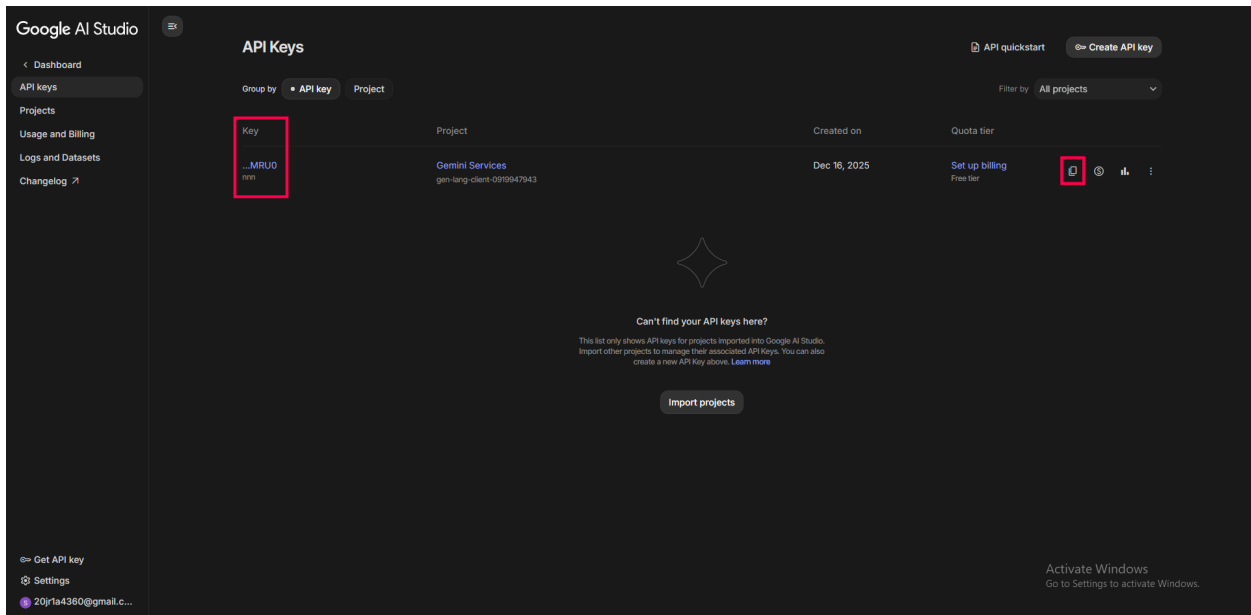
www.miraclesoft.com

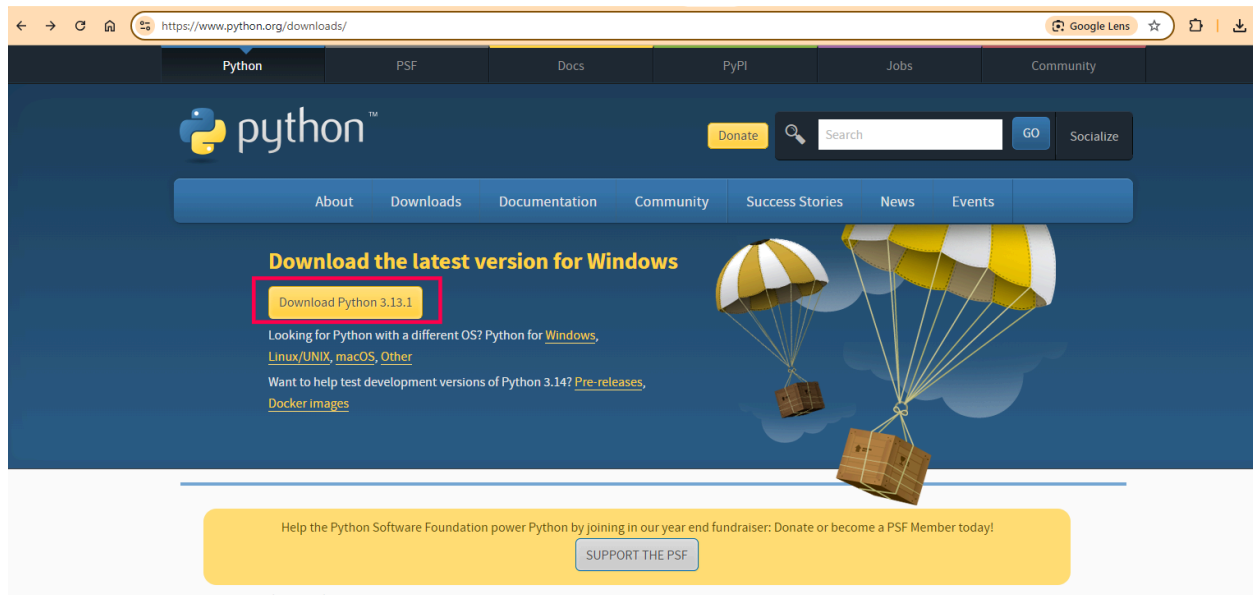- Click on '**Create API Key**' as shown below.

- Select the project
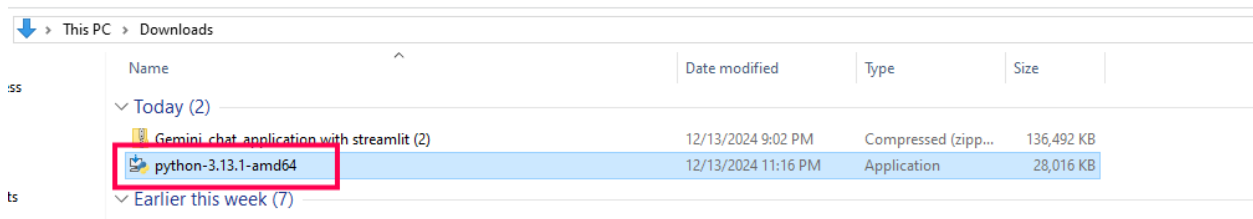
- Now, copy the **API key** and use it in the code.

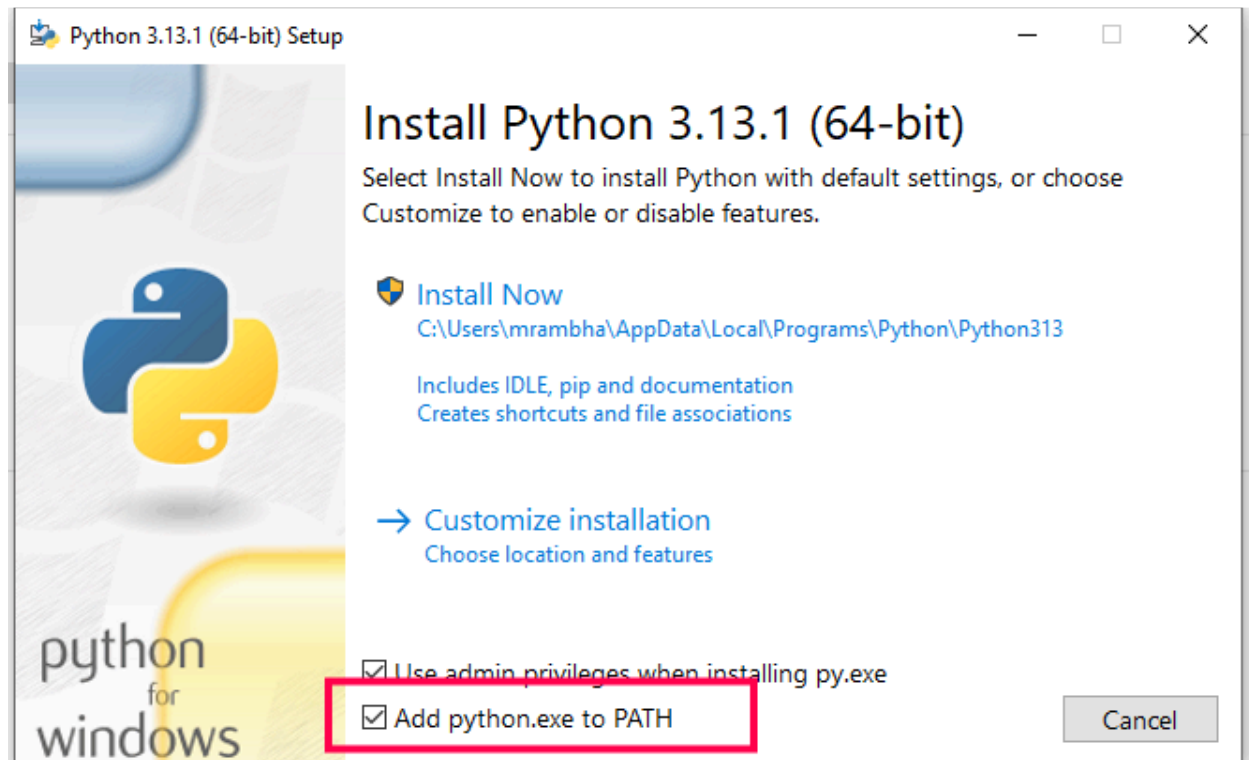## Step 3 |Steps to Install and Set Up Python on Your Local

- You must have the Python interpreter installed on your local machine. Click the link below to install Python (version).
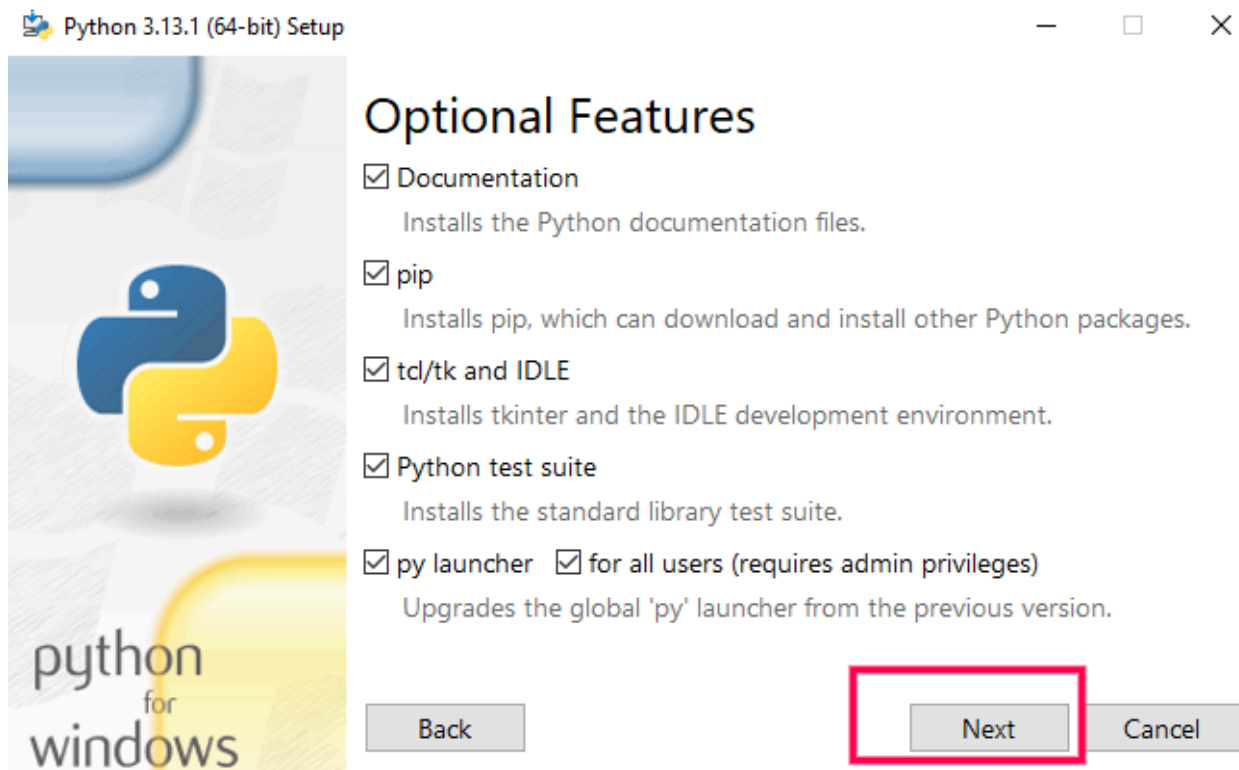- https://www.python.org/downloads/

- Open the downloaded installer file, as shown in the image below.
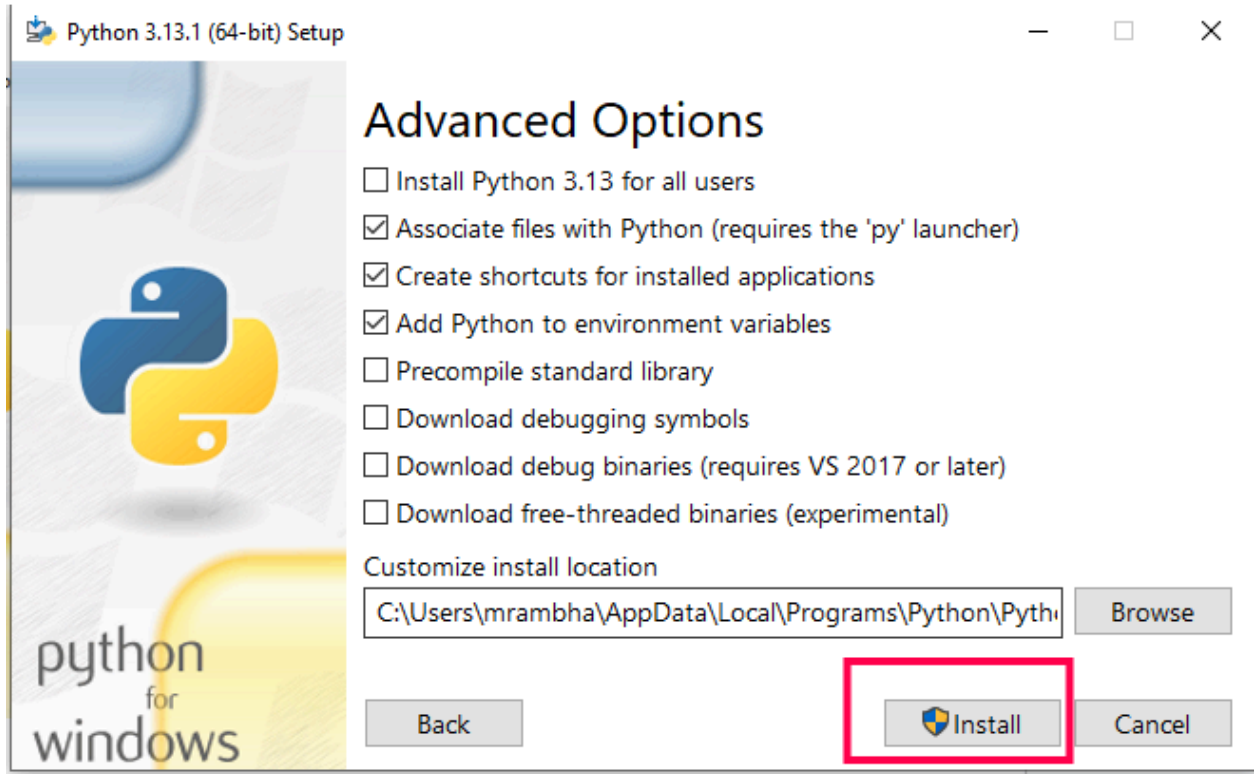

- Click on "**Add Python to PATH**" checkbox

www.miraclesoft.com

- Click on "**Next**" as shown in the image below.

- Click on "**Install Now**" as shown in the image below.

www.miraclesoft.com

Step 4 |Steps to Set Up The Python Project

1. Project with Virtual Environment:

- Open a terminal or command prompt in the directory where you want to create the virtual environment. Then, use the following command to install the required libraries: **python -m venv venv**

www.miraclesoft.com

Once the environment is created, run the command to activate it:

- For Windows: `venv\Scripts\activate`
- For Linux/Mac: `source venv/bin/activate`

After activation, your terminal prompt will look like the image below.

Open the terminal and navigate to the example folder. Use the below command

- For Windows: cd example

2. **Install Required Packages:**
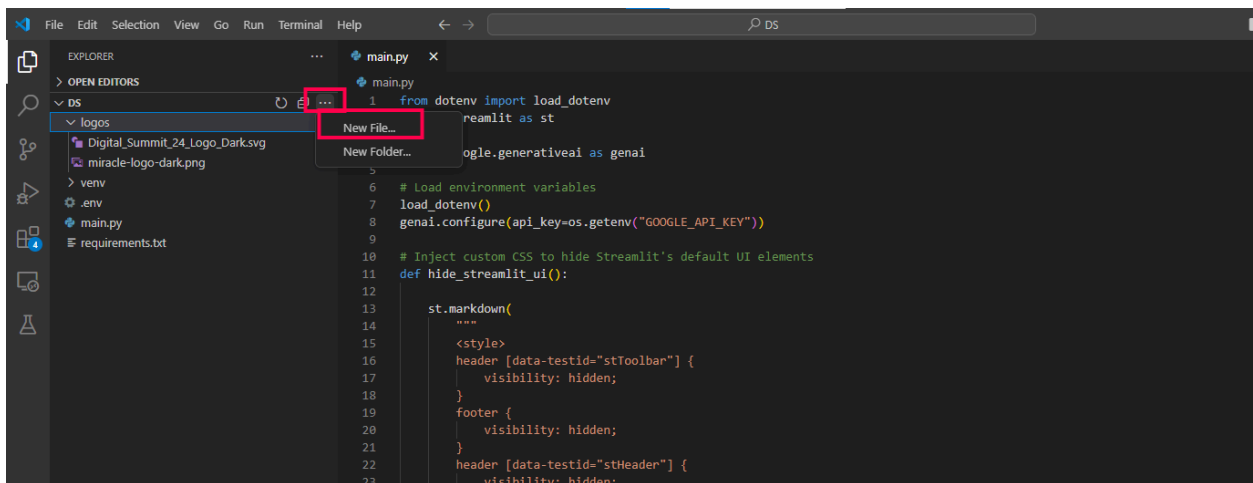   - Install all the dependencies required for this usecase by using the below command:
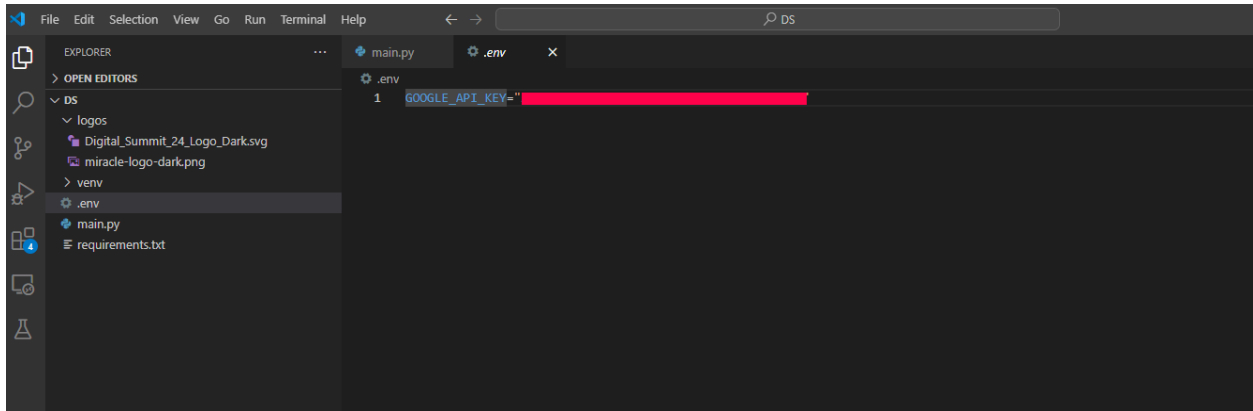
www.miraclesoft.com

pip install -r requirements.txt

3. Environment Variables (.env)

- Create a new file, name it as **.env** in the root of your project directory.

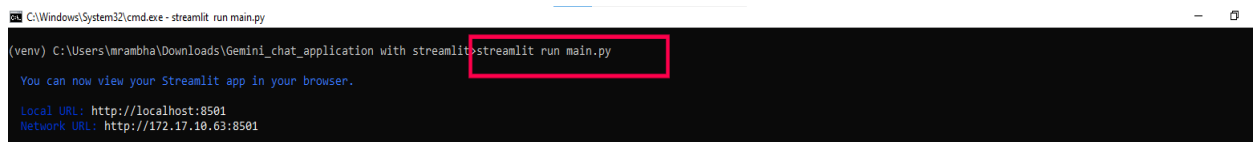- Inside the **.env** file, add your variables in the **KEY=VALUE** format, one per line `GOOGLE_API_KEY=<Your-Google-API-Key>`

- Replace **`<Your-Google-API-Key>`** with your actual API key that we got in the **Step-2**

Step 5 | Run the Chatbot Application

- Use the below command to start the Streamlit app:
  streamlit run main.py

- After running the command, your terminal will look like below.
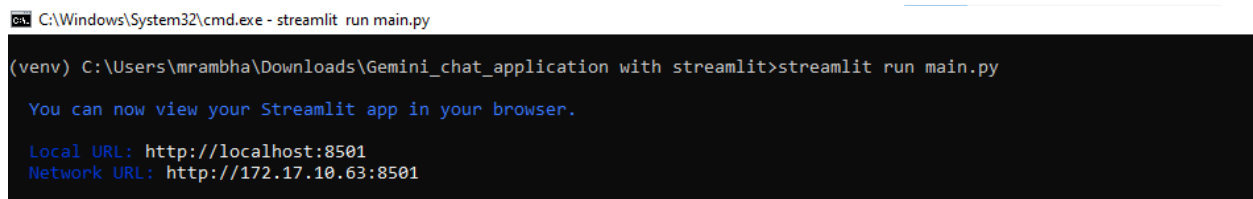
Step 6 |Test the Chatbot Application

- After running the command, Streamlit will provide a URL (**http://localhost:8501**) in the terminal

- Open this URL in your browser to interact with the chatbot application, which will redirect you to the UI.

www.miraclesoft.com
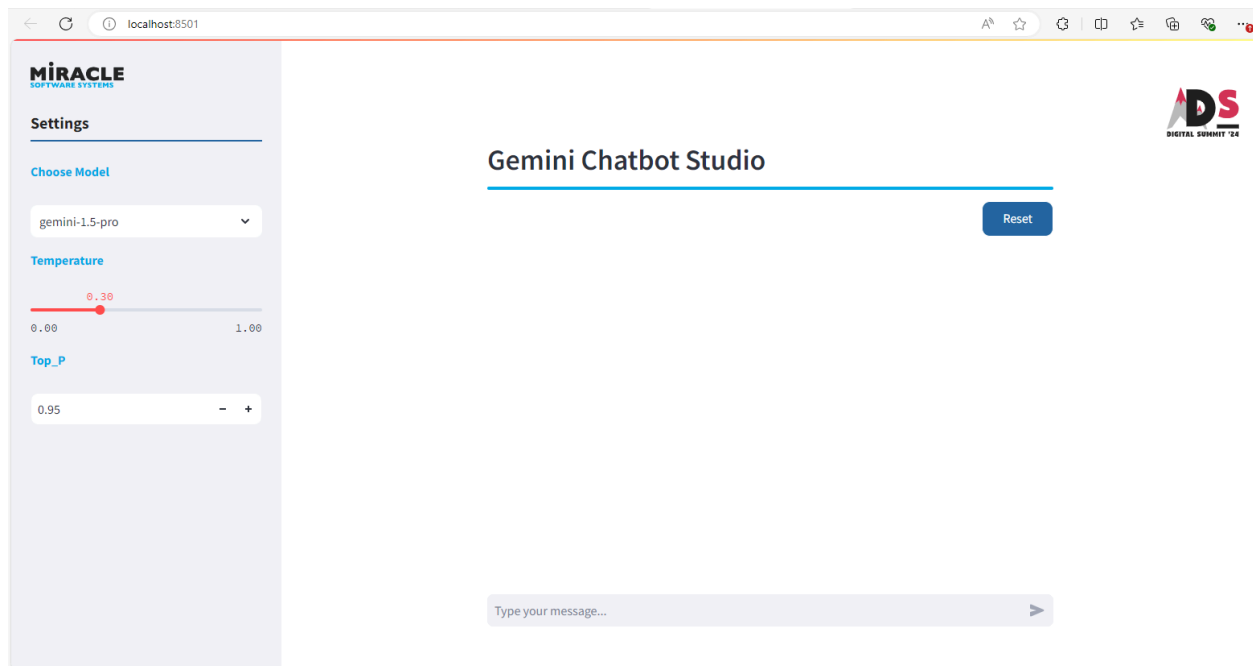
- Enter a question or prompt in the input box and submit it.
- Ask a variety of questions from different domains, powered by Google's Gemini Pro.

Example Prompts/ Questions:

- What does the temperature setting do in generative AI models?
- Can you help me debug this error: KeyError: 'value'?
- Explain the theory of relativity?
- Can you suggest a workout plan for beginners?
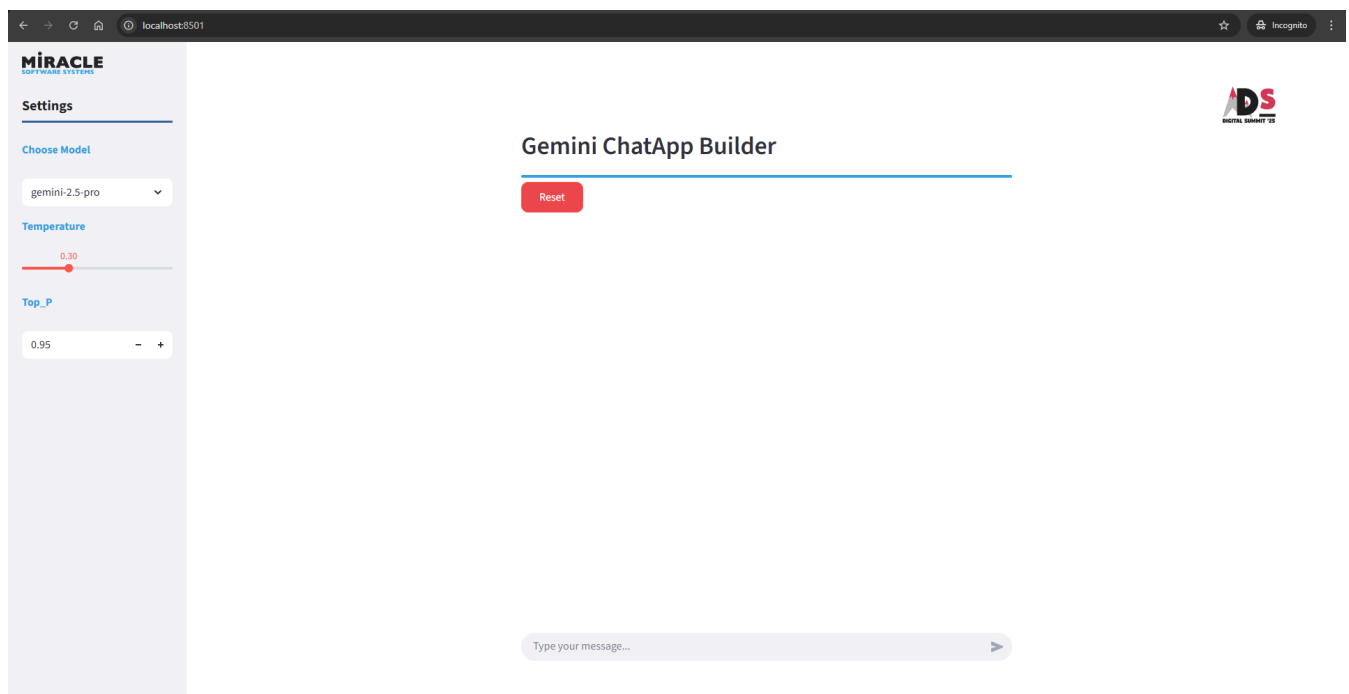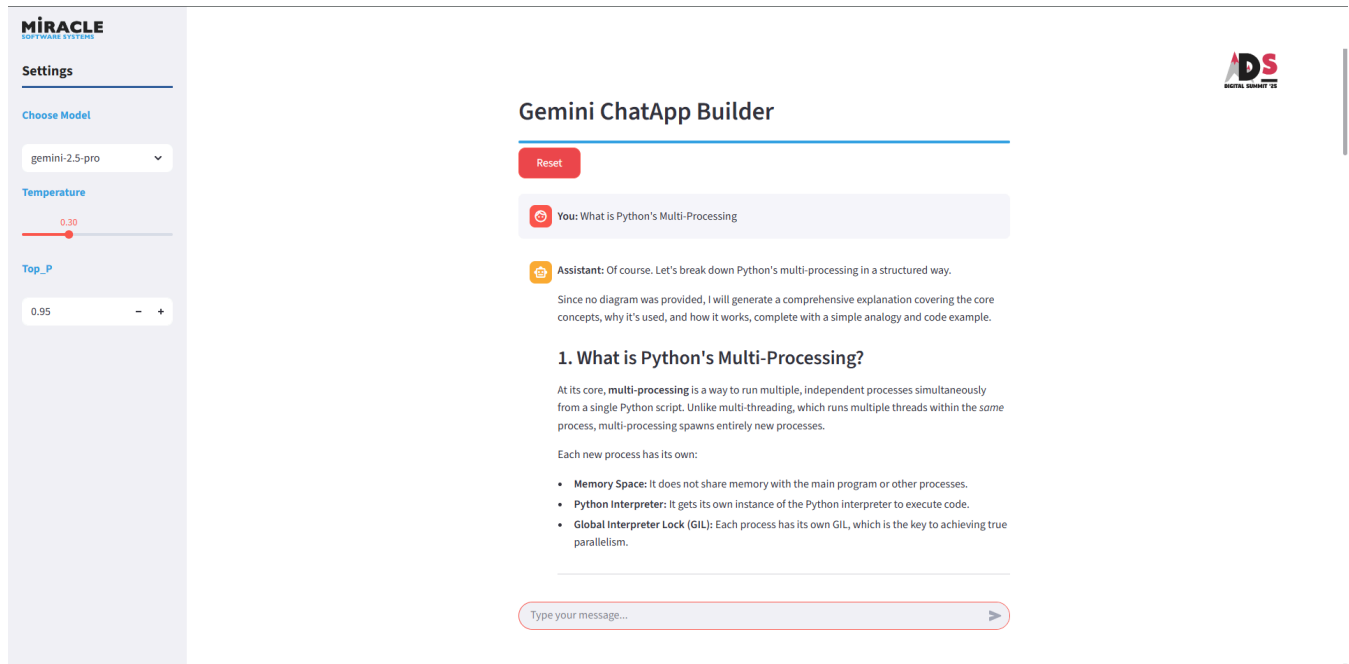
- **Temperature:** Controls randomness in responses. Lower values (e.g., 0.2) lead to more focused answers, while higher values (e.g., 1.0) encourage more creative responses.
- **Top_P:** Limits choices to the top tokens with a cumulative probability above P. Higher values (e.g., 0.95) increase diversity in responses.

Gemini ChatApp Builder

Settings

Choose Model

gemini-2.5-pro

Temperature

0.60

0.00 — 1.00

Top_P

0.95 − +

Reset

You: What is Python's Multi-Processing

Assistant: Of course. Let's break down Python's multi-processing in a structured way.

Since no diagram was provided, I will generate a comprehensive explanation covering the core concepts, why it's used, and how it works, complete with a simple analogy and code example.

1. What is Python's Multi-Processing?

At its core, **multi-processing** is a way to run multiple, independent processes simultaneously from a single Python script. Unlike multi-threading, which runs multiple threads within the *same* process, multi-processing spawns entirely new processes.

Each new process has its own:

- **Memory Space:** It does not share memory with the main program or other processes.
- **Python Interpreter:** It gets its own instance of the Python interpreter to execute code.
- **Global Interpreter Lock (GIL):** Each process has its own GIL, which is the key to achieving true parallelism.

Type your message...