



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++)

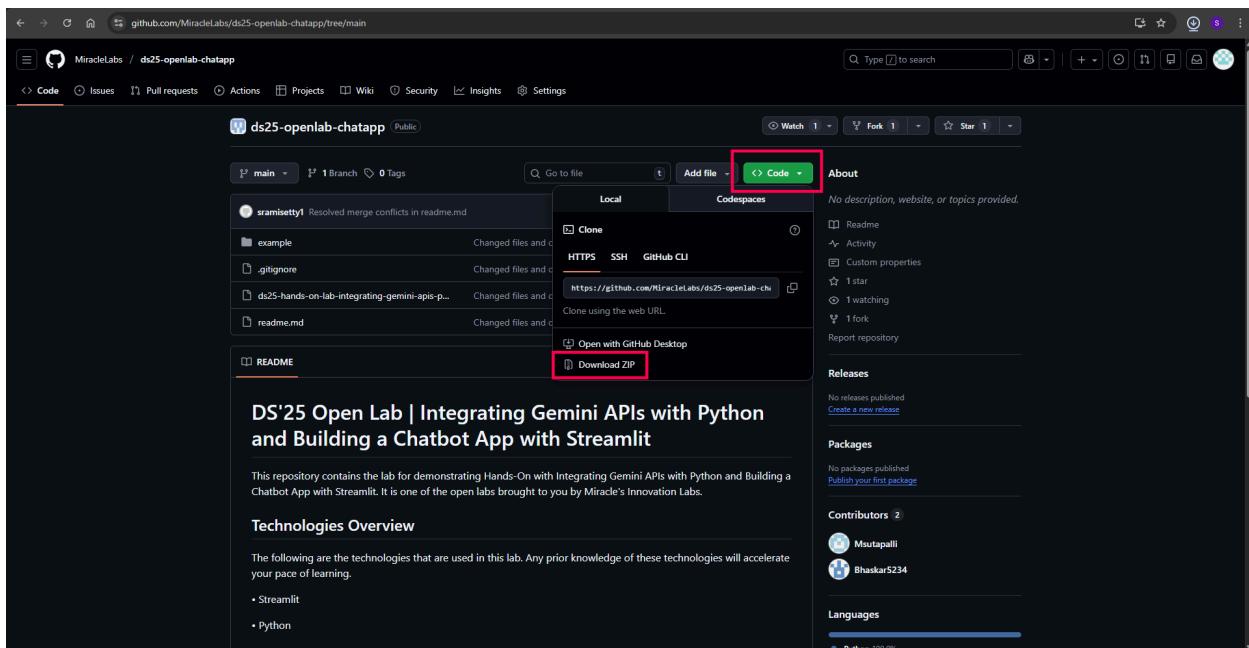
Technology Involved

- Python
- Streamlit (HTML + CSS)

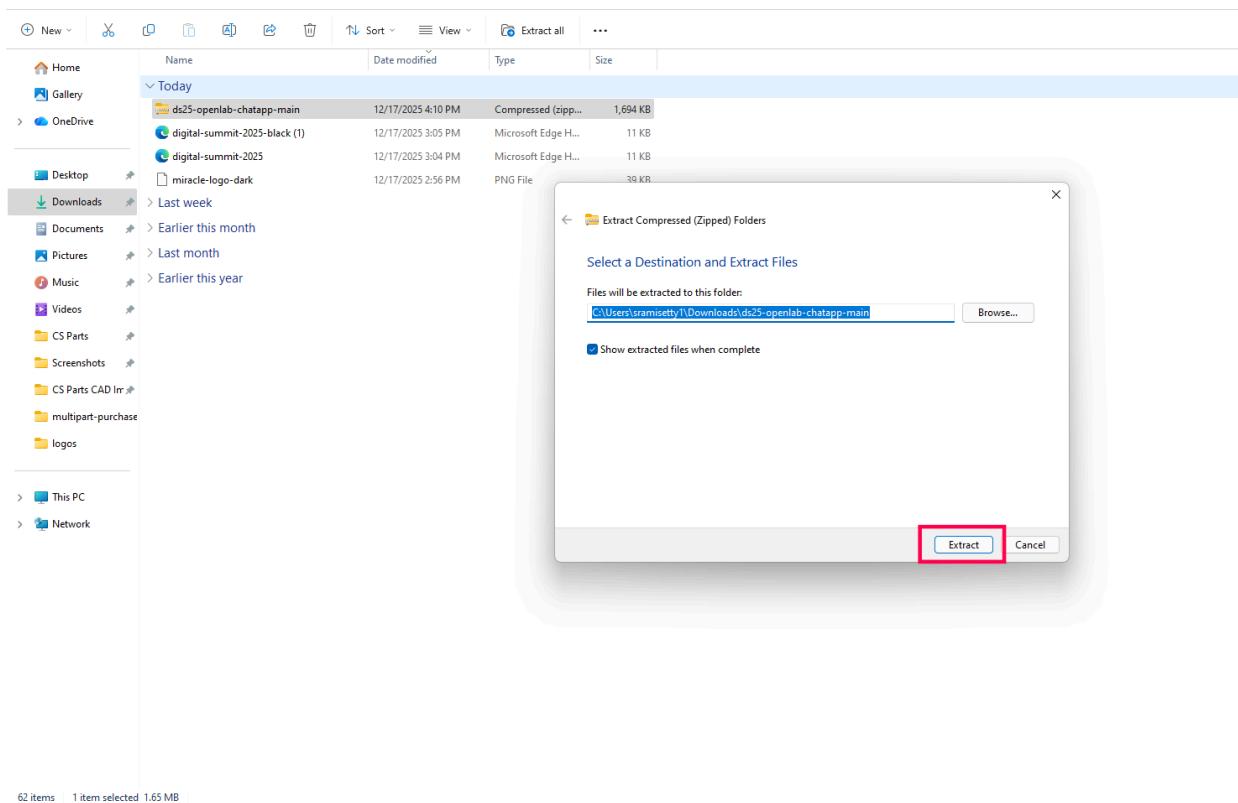
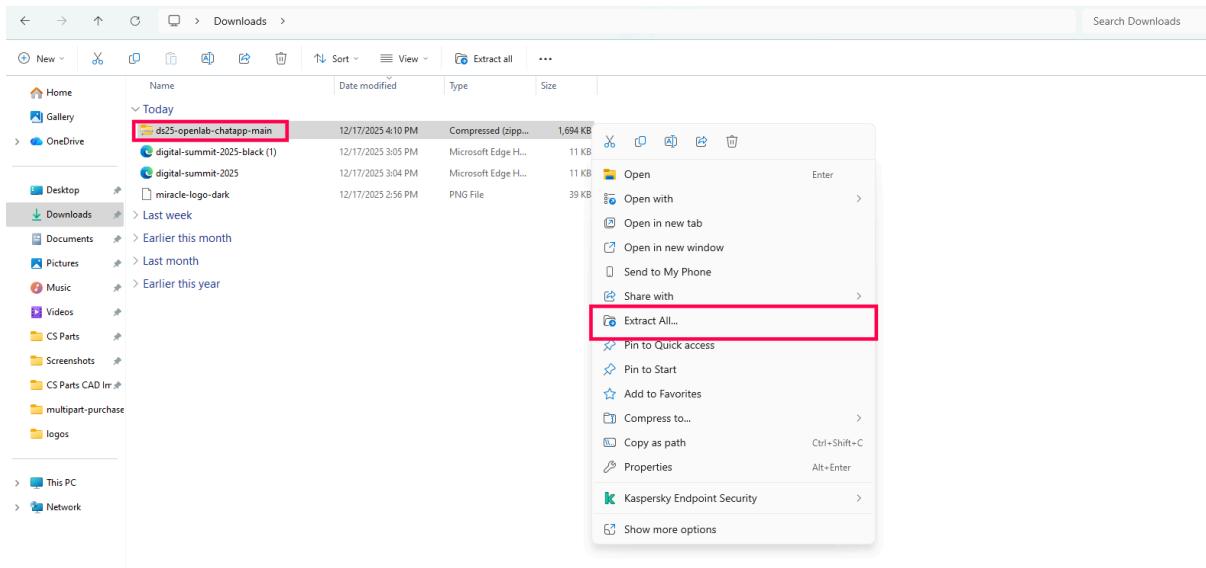
Step 1 | Download Code Repository

Get Started

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



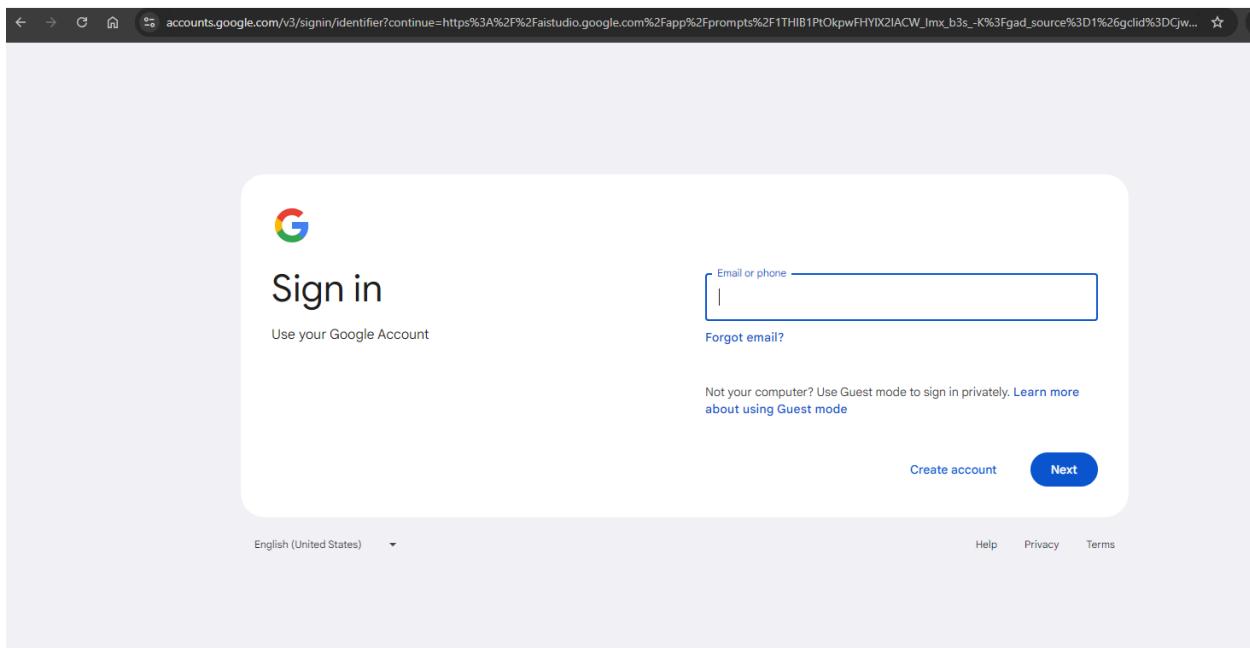
- After downloading, unzip it



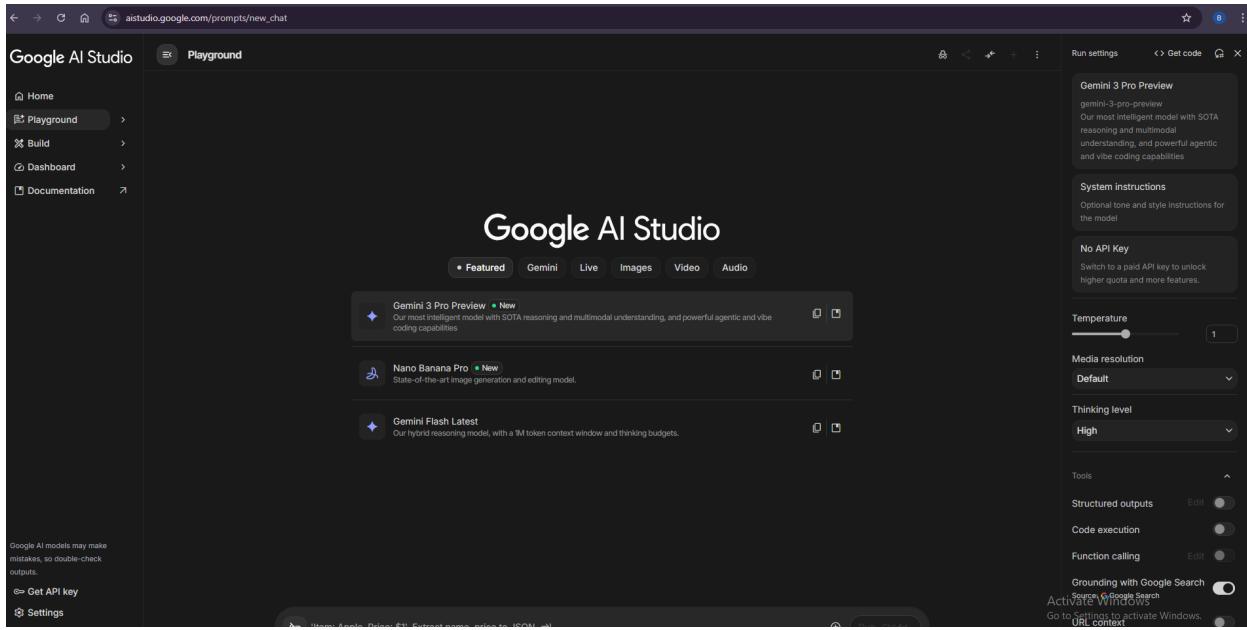
62 items 1 item selected 1.65 MB

Step 2 | Access Google AI Studio

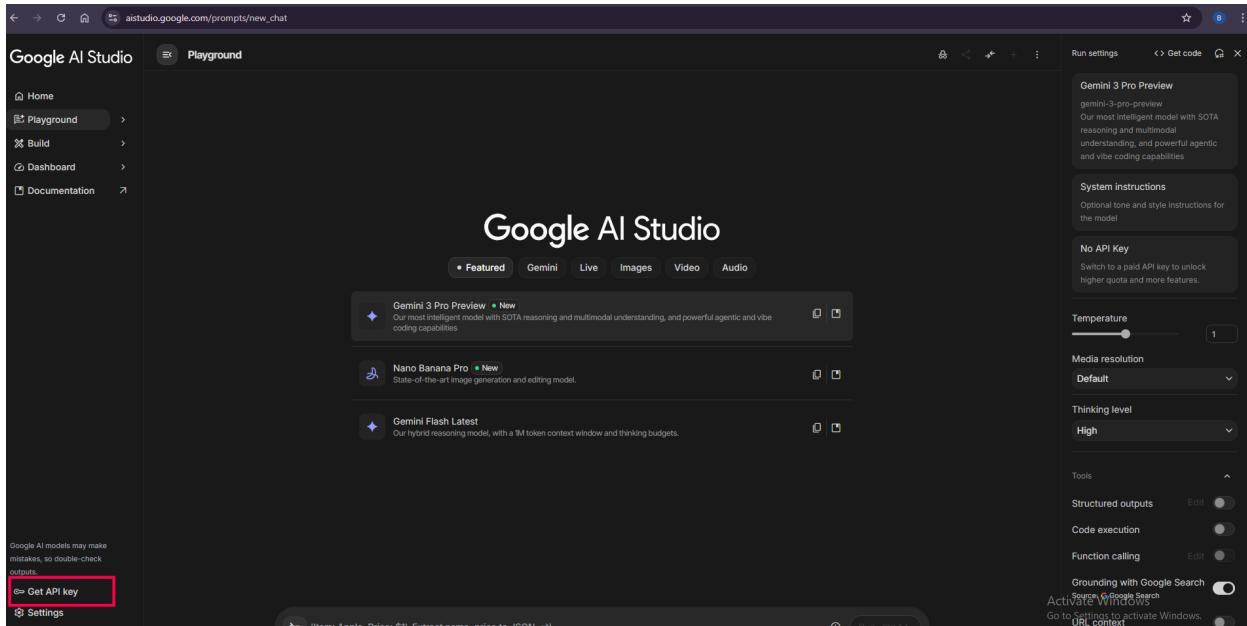
- Click on [Google AI Studio](#)



- 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.



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

The screenshot shows the 'API Keys' section of the Google AI Studio interface. On the left, there's a sidebar with links like 'Dashboard', 'API keys' (which is selected and highlighted in blue), 'Projects', 'Usage and Billing', 'Logs and Datasets', and 'Changelog'. The main area has a dark header with 'API Keys' and three tabs: 'Group by' (selected), 'API key' (highlighted in blue), and 'Project'. Below the header is a table with columns: 'Key', 'Project', 'Created on', and 'Quota tier'. One row is visible: '...0e6E Streamlit App' under 'Gemini Services', 'Dec 16, 2025', and 'Set up billing Free tier'. At the bottom of the table area, there's a message: 'Can't find your API keys here? This list only shows API keys for projects imported into Google AI Studio. Import other projects to manage their associated API Keys. You can also create a new API key above. [Learn more](#)'. A pink rectangular box highlights the 'Create API key' button at the top right of the page.

- Select the project

This screenshot shows the same 'API Keys' page as the previous one, but with a modal window open over it. The modal is titled 'Create a new key' and contains fields for 'Name your key' (with 'Streamlit Chat App' typed in) and 'Choose an imported project'. A dropdown menu labeled 'Select a Cloud Project' shows two options: 'Import project' (which is highlighted with a pink box) and '+ Create project'. Other options like 'Gemini Services' and 'Activate Windows' are visible at the bottom of the modal. The background page remains largely the same, showing the table of existing API keys.

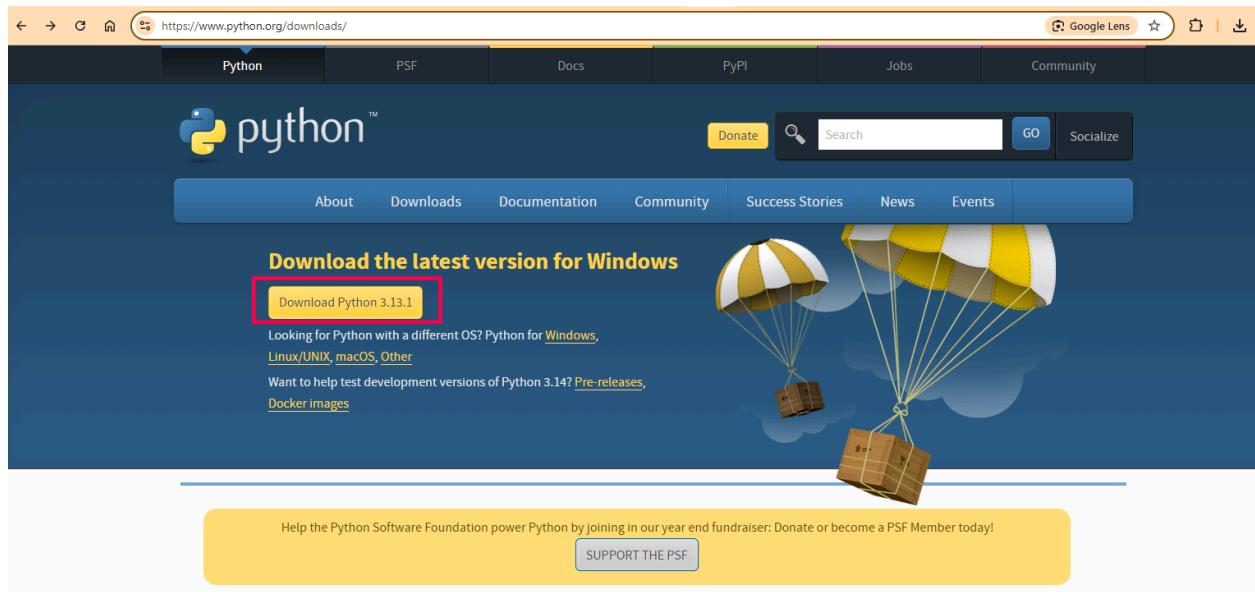
- Now, copy the **API key** and use it in the code.

The screenshot shows the 'API Keys' page in Google AI Studio. On the left, there's a sidebar with links like 'Dashboard', 'API keys' (which is selected and highlighted in blue), 'Projects', 'Usage and Billing', 'Logs and Datasets', and 'Changelog'. The main area has a dark background with white text. It displays a table with one row of data. The columns are 'Key' (containing '...MRUO'), 'Project' (containing 'Gemini Services'), 'Created on' (containing 'Dec 16, 2025'), and 'Quota tier' (containing 'Set up billing Free tier'). There are also 'Edit' and 'Delete' buttons for each row. A red box highlights the 'Key' column header. Below the table, there's a section titled 'Can't find your API keys here?' with a link to 'Import projects'. At the bottom, there are links for 'Get API key', 'Settings', and an email address '20jrla4360@gmail.c...', along with an 'Activate Windows' button.

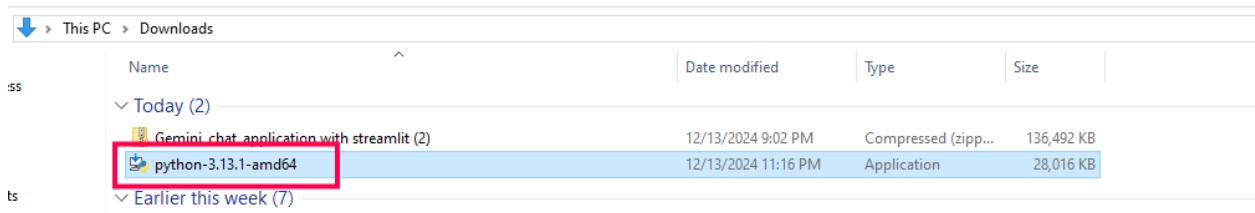
Key	Project	Created on	Quota tier
...MRUO nnn	Gemini Services gen-lang-client-099947943	Dec 16, 2025	Set up billing Free tier

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

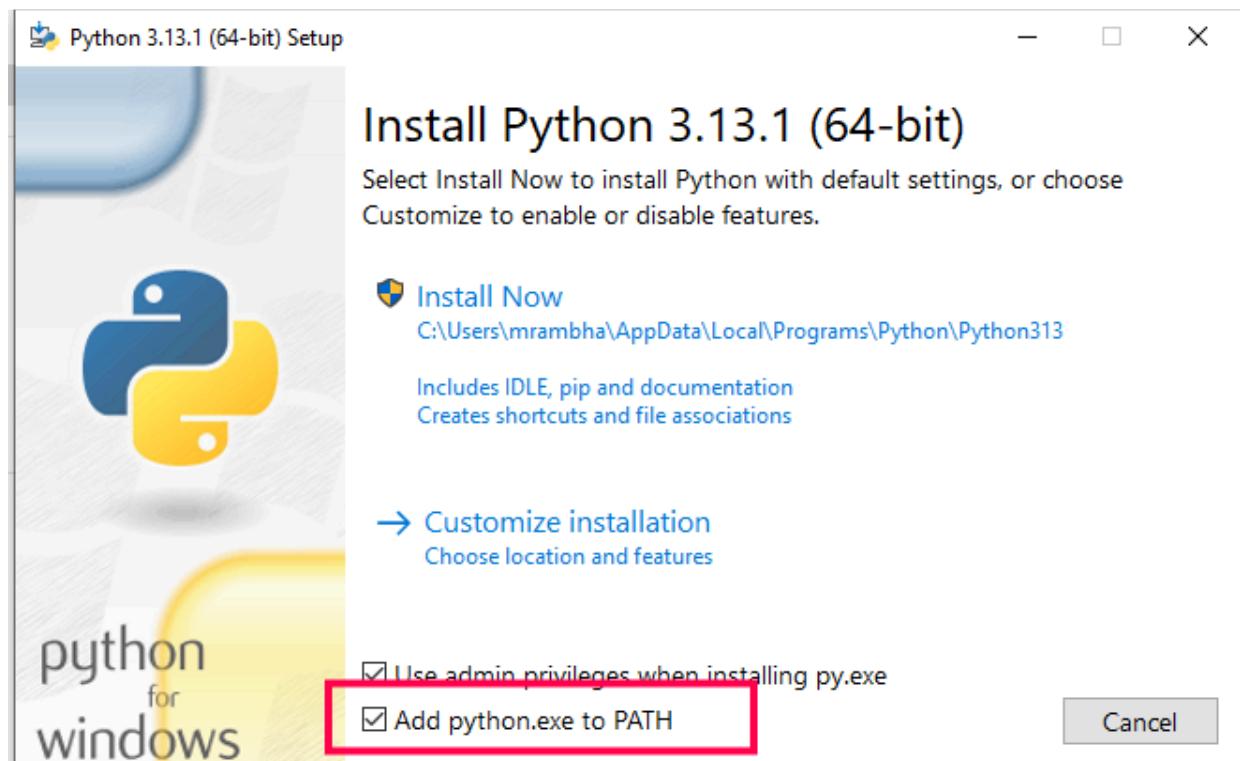
- You must have the Python interpreter installed on your local machine. Click the link below to install Python (version).
- <https://www.python.org/downloads/>



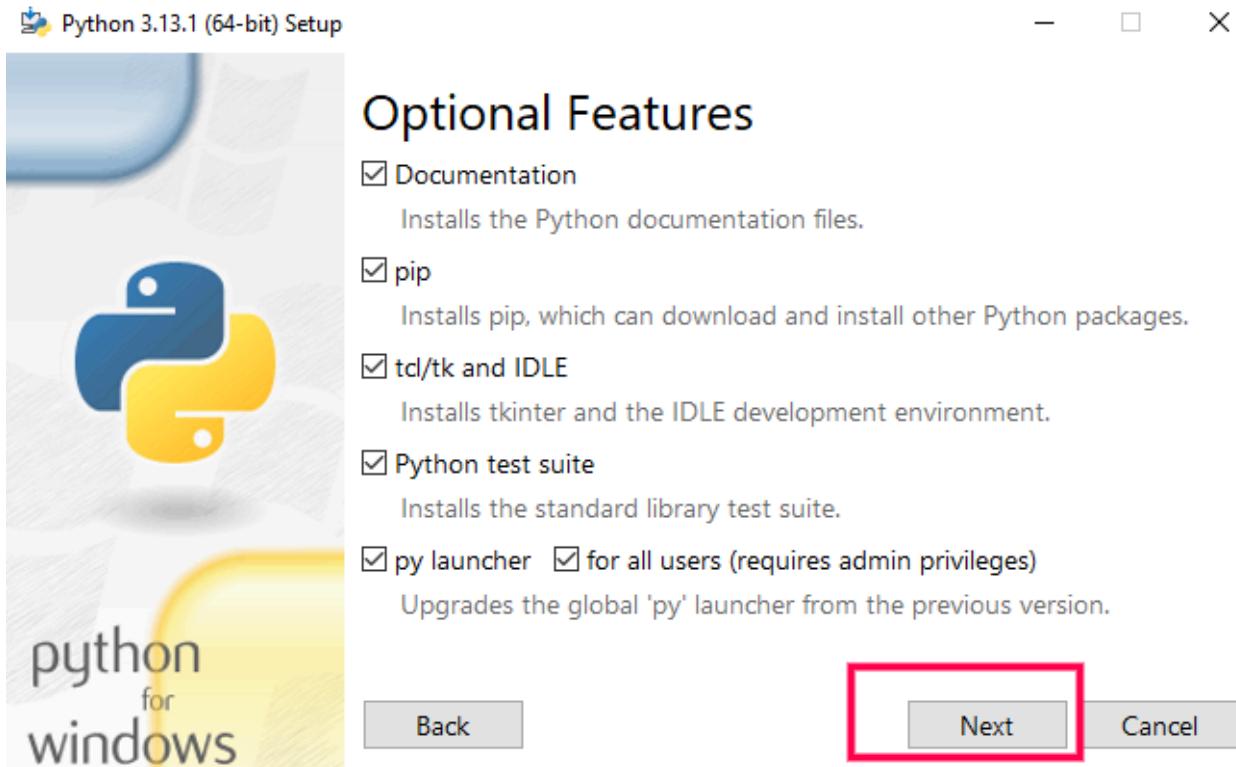
- Open the downloaded installer file, as shown in the image below.



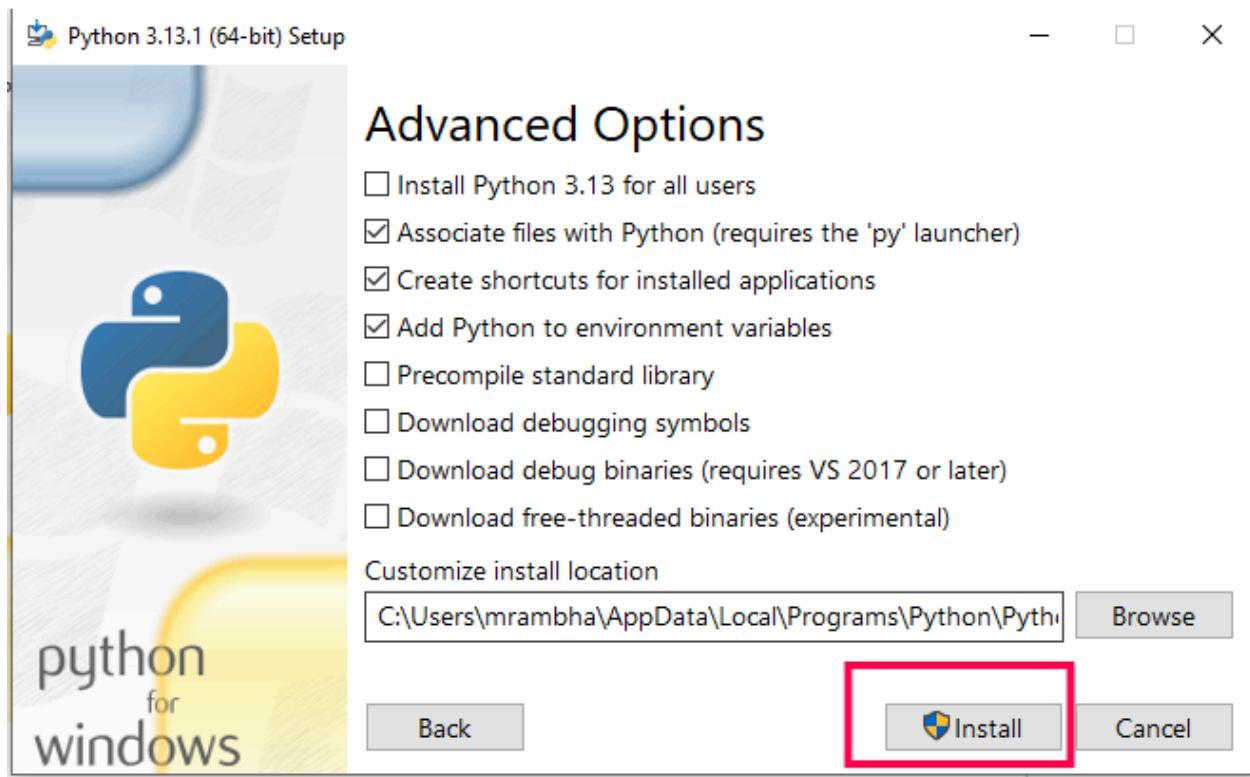
- Click on "Add Python to PATH" checkbox



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



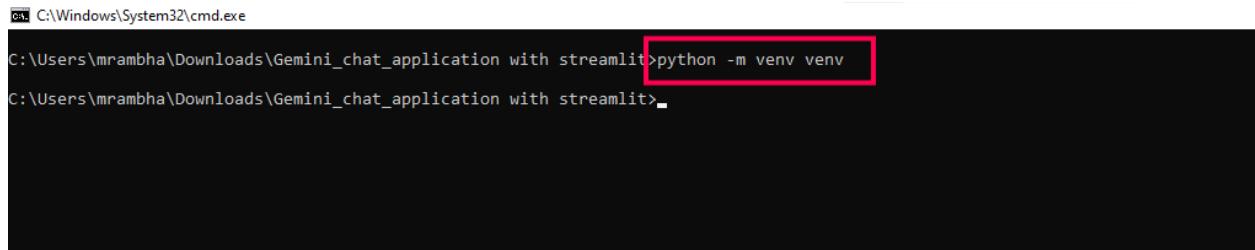
- Click on "**Install Now**" as shown in the image below.



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**



```
C:\Windows\System32\cmd.exe
C:\Users\mrambah\Downloads\Gemini_chat_application with streamlit>python -m venv venv
C:\Users\mrambah\Downloads\Gemini_chat_application with streamlit>
```

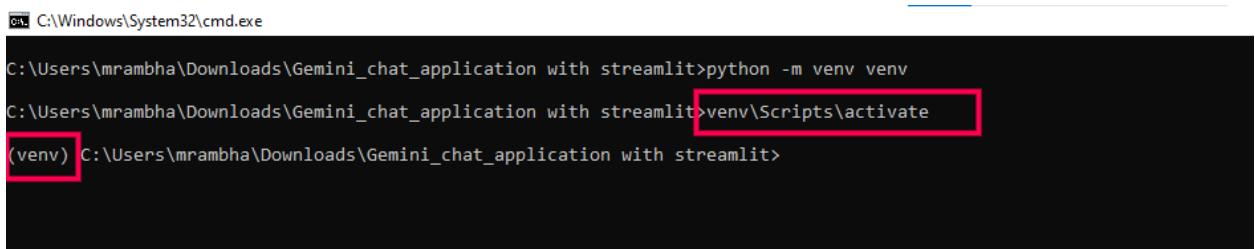
Open the terminal and navigate to the example folder. Use the below command

- For Windows: `cd example`

Once the environment is created, you need 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.

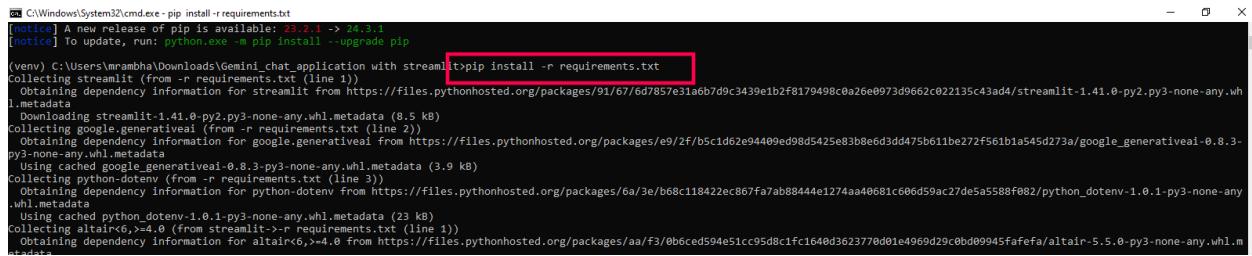


```
C:\Windows\System32\cmd.exe
C:\Users\mrambah\Downloads\Gemini_chat_application with streamlit>python -m venv venv
C:\Users\mrambah\Downloads\Gemini_chat_application with streamlit>venv\Scripts\activate
(venv) C:\Users\mrambah\Downloads\Gemini_chat_application with streamlit>
```

2. Install Required Packages:

- Install all the dependencies required for this usecase by using the below command:

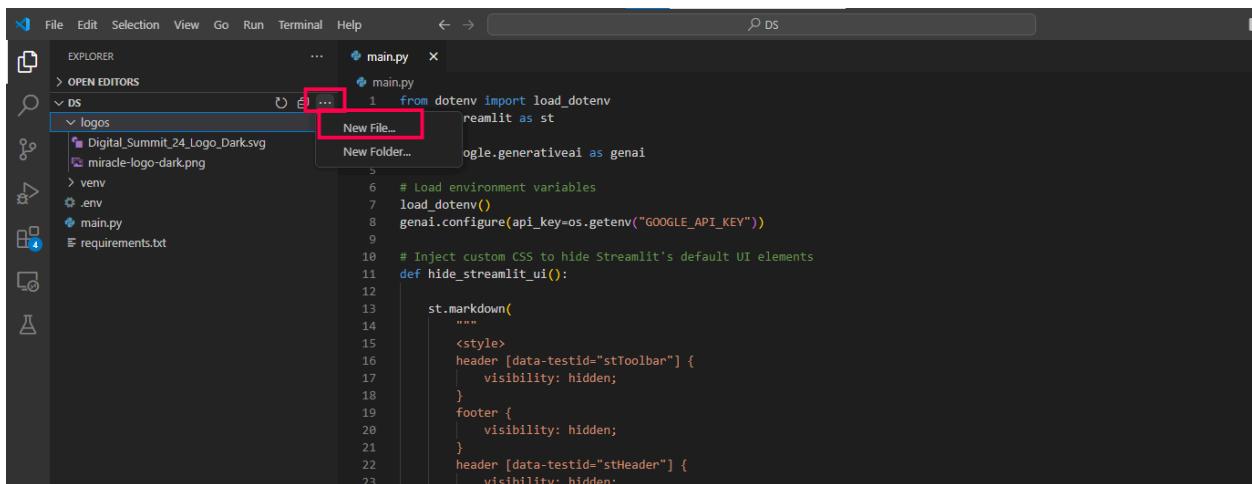
pip install -r requirements.txt



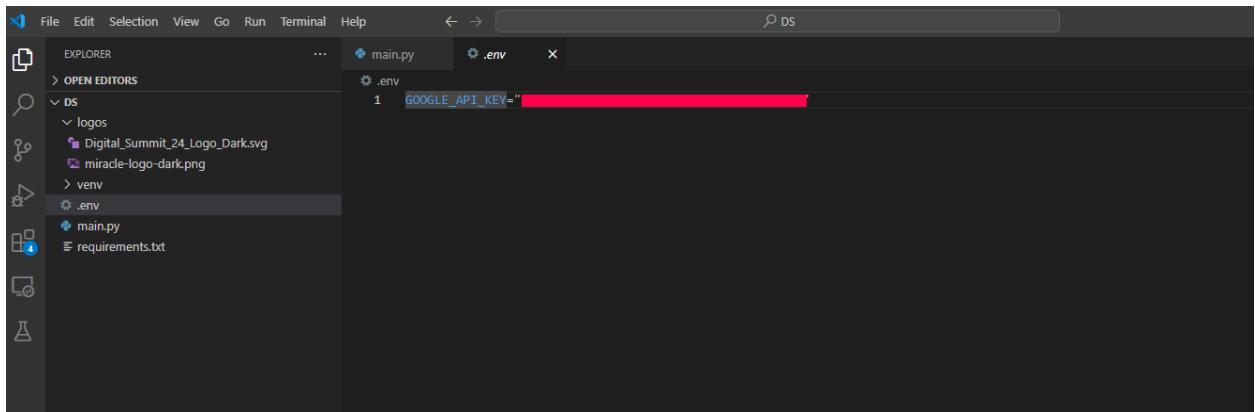
```
C:\Windows\System32\cmd.exe - pip install -r requirements.txt
[notice] A new release of pip is available: 23.2.1 => 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) C:\Users\vrmbha\Downloads\Gemini_chat_application with streamlit>pip install -r requirements.txt
Collecting streamlit (from -r requirements.txt (line 1))
  Obtaining dependency information for streamlit from https://files.pythonhosted.org/packages/9f/67/bd7857e31a6b7d9c3430e1b2f8179498c0a26e0973d9662c022135c43ad4/streamlit-1.41.0-py2.py3-none-any.whl.metadata
    Downloading streamlit-1.41.0-py2.py3-none-any.whl.metadata (9.5 kB)
Collecting google_generativeai (from -r requirements.txt (line 2))
  Obtaining dependency information for google.generativai from https://files.pythonhosted.org/packages/e9/2f/b5c1d62e94409ed98d5425e83b8e6d3dd475b61be272f561b1a545d273a/google_generativeai-0.8.3-py3-none-any.whl.metadata
    Using cached google_generativeai-0.8.3-py3-none-any.whl.metadata (3.9 kB)
Collecting python_dotenv (from -r requirements.txt (line 3))
  Obtaining dependency information for python-dotenv from https://files.pythonhosted.org/packages/6a/3e/b68c118422ec867fa7ab88444e1274aa40681c606d59ac27de5a5588f082/python_dotenv-1.0.1-py3-none-any.whl.metadata
    Using cached python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Collecting altair<6,>=4.0 (from streamlit->r requirements.txt (line 4))
  Obtaining dependency information for altair<6,>=4.0 from https://files.pythonhosted.org/packages/aa/f3/0b6ced594e51cc95d8c1fc1640d3623770d01e4969d29c0bd09945fafefa/altair-5.5.0-py3-none-any.whl.metadata
```

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

```
C:\Windows\System32\cmd.exe - streamlit run main.py
(venv) C:\Users\mrambha\Downloads\Gemini_chat_application with streamlit>streamlit run main.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://172.17.10.63:8501
```

A screenshot of a terminal window showing the command 'streamlit run main.py' being executed. The output indicates that the Streamlit app is running and provides local and network URLs for viewing it.

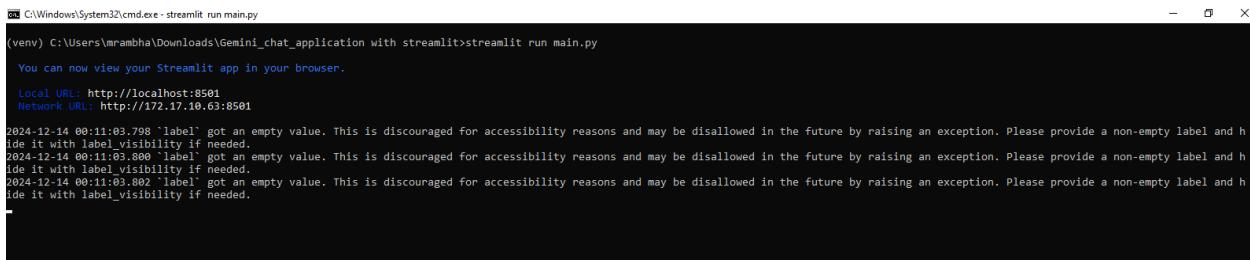
- After running the command, your terminal will look like below.

```
C:\Windows\System32\cmd.exe - streamlit run main.py
(venv) C:\Users\mrambha\Downloads\Gemini_chat_application with streamlit>streamlit run main.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://172.17.10.63:8501
```

A second screenshot of a terminal window showing the same command and output as the previous one, confirming the Streamlit app is running.

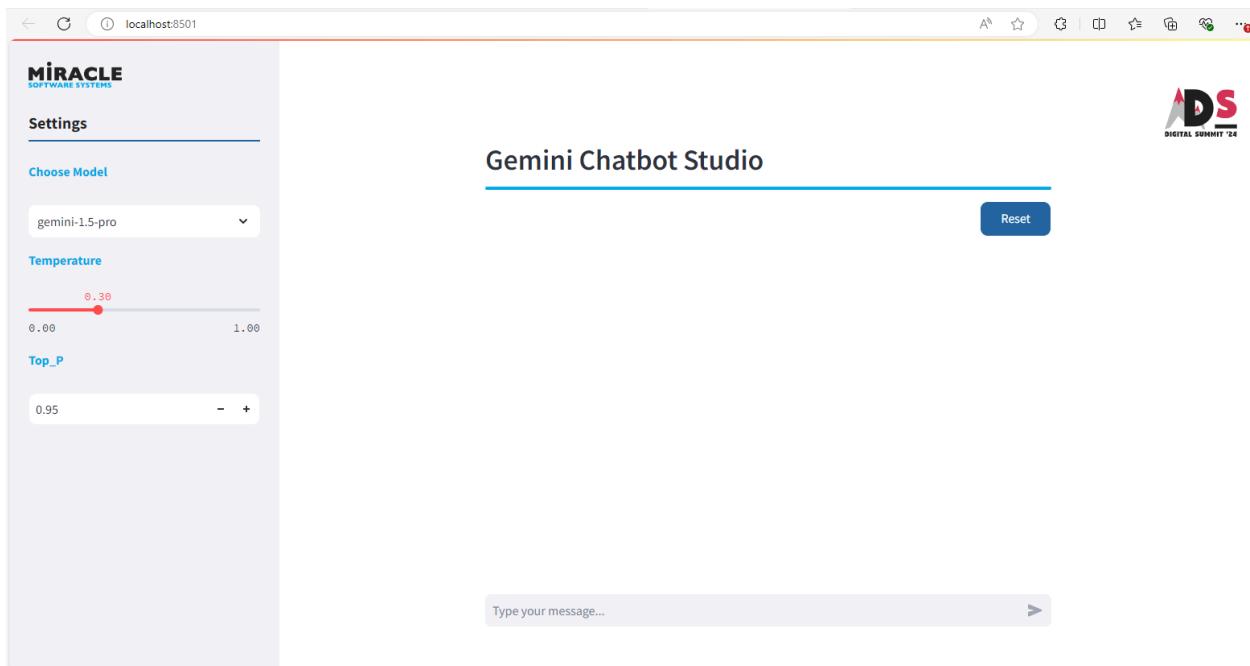
Step 6 | Test the Chatbot Application

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



```
C:\Windows\System32\cmd.exe - streamlit run main.py
(venv) C:\Users\mrambha\Downloads\Gemini_chat_application with streamlit>streamlit run main.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://172.17.10.63:8501
2024-12-14 00:11:03.798 'label' got an empty value. This is discouraged for accessibility reasons and may be disallowed in the future by raising an exception. Please provide a non-empty label and hide it with label_visibility if needed.
2024-12-14 00:11:03.800 'label' got an empty value. This is discouraged for accessibility reasons and may be disallowed in the future by raising an exception. Please provide a non-empty label and hide it with label_visibility if needed.
2024-12-14 00:11:03.802 'label' got an empty value. This is discouraged for accessibility reasons and may be disallowed in the future by raising an exception. Please provide a non-empty label and hide it with label_visibility if needed.
```

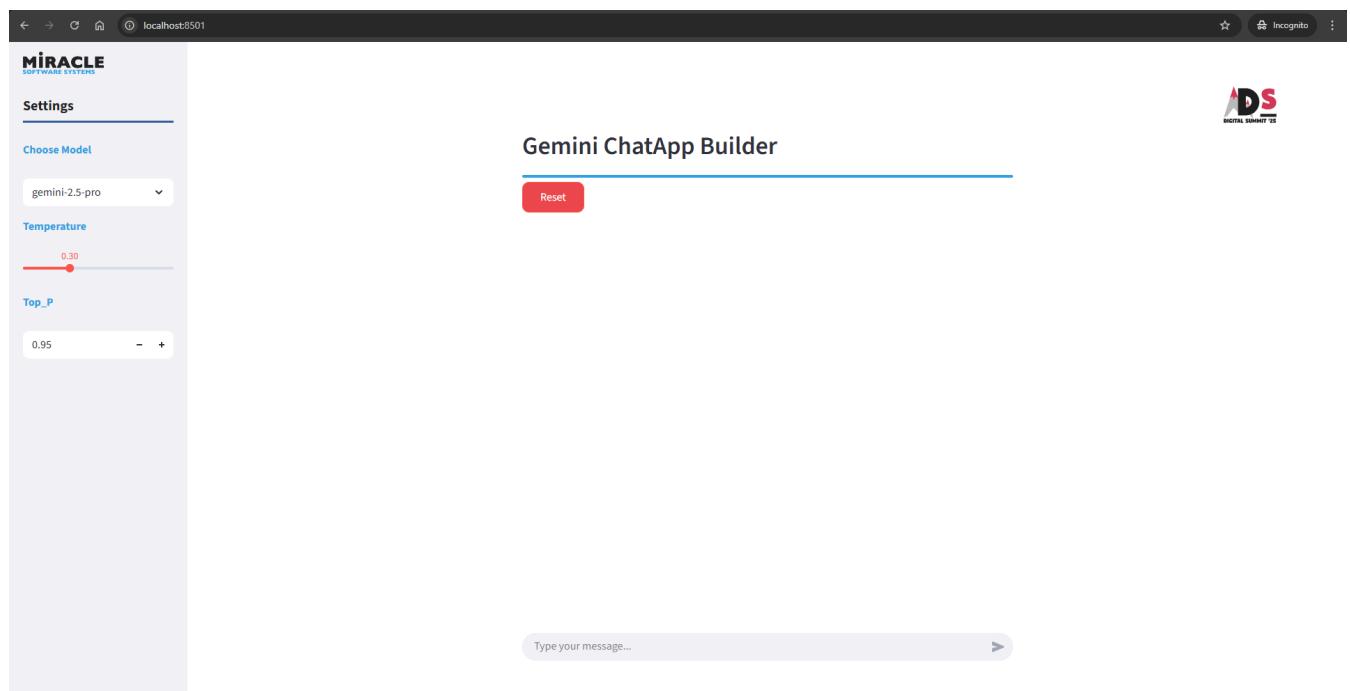
- Open this URL in your browser to interact with the chatbot application, which will redirect you to the UI.



- 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.

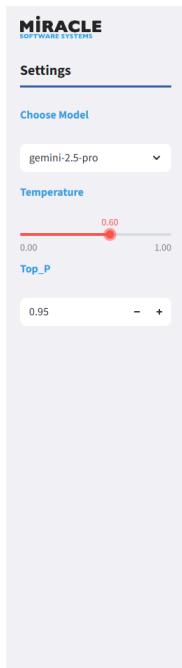
Example Prompts/ Questions:

- 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?



The screenshot shows the Gemini ChatApp Builder interface. On the left, there is a sidebar with the MIRACLE SOFTWARE SYSTEMS logo and a 'Settings' section. It includes a dropdown for 'Choose Model' set to 'gemini-2.5-pro', a 'Temperature' slider set to 0.30, and a 'Top_P' input field set to 0.95. The main area is titled 'Gemini ChatApp Builder' and features a red 'Reset' button. A message from the user says 'You: What is Python's Multi-Processing?' and an AI response says '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.' Below this, a section titled '1. What is Python's Multi-Processing?' provides a detailed explanation. At the bottom, there is a text input field with placeholder 'Type your message...' and a send button.

- **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

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... ➤

