Building a Gemini AI-based Streamlit Chat Application

Before you begin you will need the following:

To get started, you'll need to have a standard Google account. If you haven't set one up yet, please visit Google's account creation page to register for a new account. This will be essential for accessing your project's full range of features and services. Make sure you have created an account at https://accounts.google.com/SignUp.

Creating a Google Developer Account (Play Console Account):

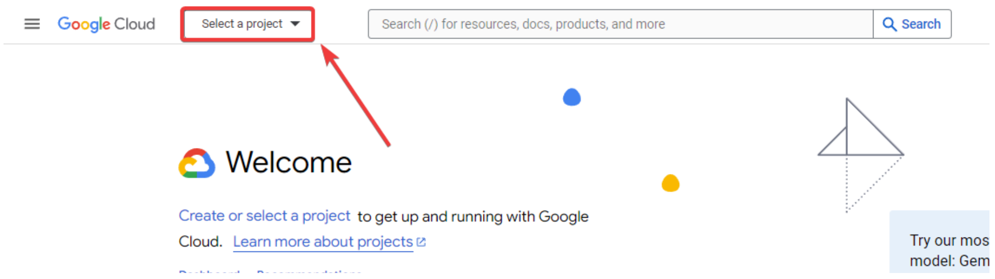
Please note that while creating a Google Developer account (Play Console account) typically requires a $25 registration fee, you can directly create a project in the Google Cloud platform console and obtain an API key from the Google AI Studio for our project without this fee.

Creating a Project in the Google Cloud Platform Console:

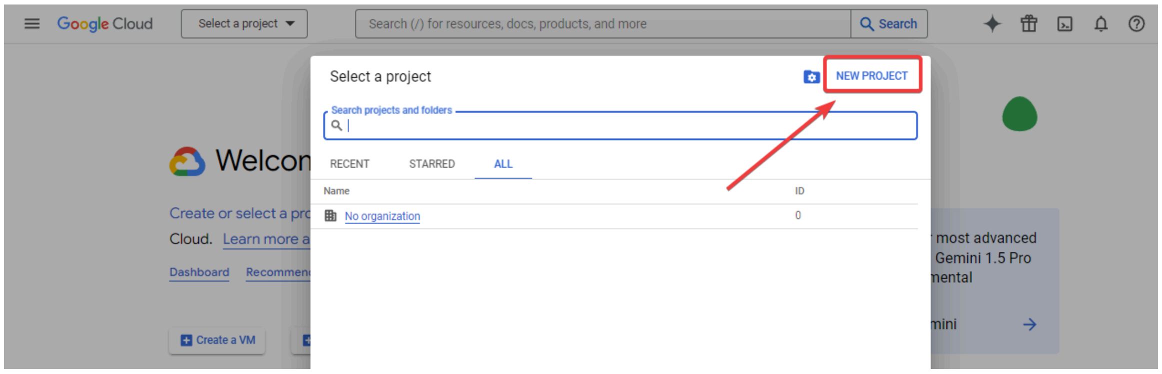
Creating a New Project in the Google Cloud Console:

1. Go to the Google Cloud Console platform, https://console.cloud.google.com/

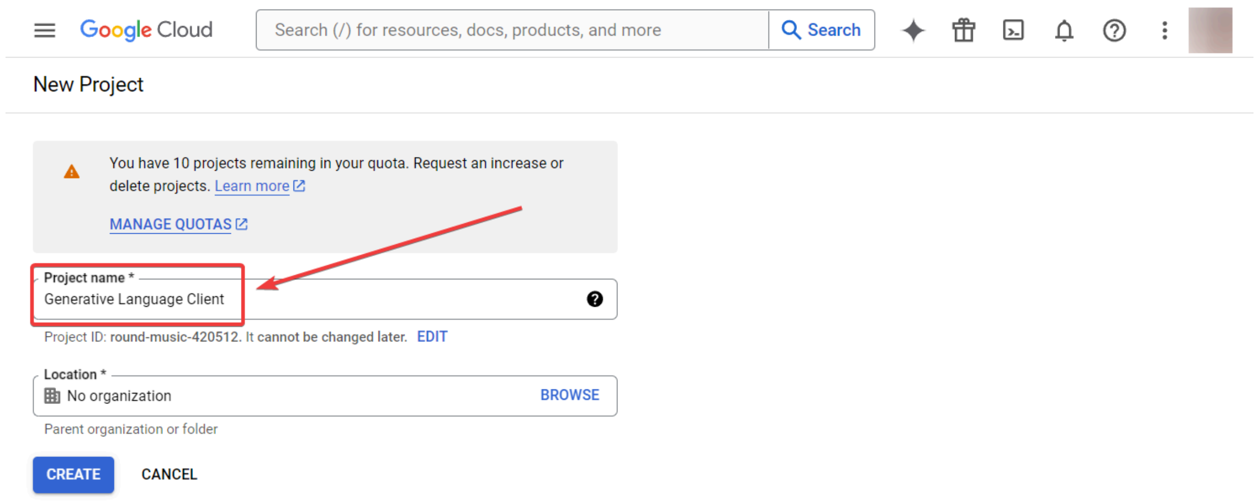
2. In the **Google Cloud Platform interface**, click on the 'Select a Project' button located at the top-left corner, adjacent to the Google Cloud icon.


Google Cloud
Select a project
Search (/) for resources, docs, products, and more
Search
Welcome
Create or select a project to get up and running with Google Cloud. Learn more about projects
Try our mos
model: Gem

3. In the **Select a Project** Dialog Box, Click on the **New Project** Button:



4. In the **New Project** creation page, Enter the Project Name that you want (*For Example Generative Language Client*) and Click on **Create**



New Project

You have 10 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

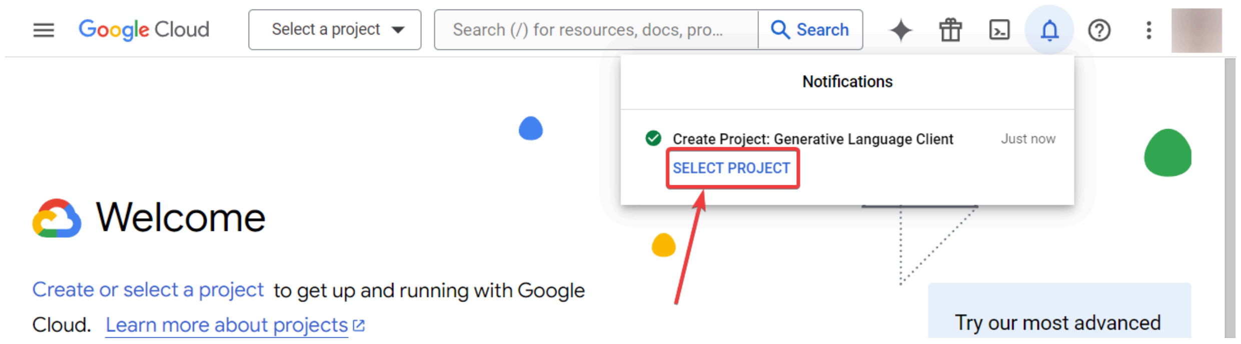
Project name *
Generative Language Client

Project ID: round-music-420512. It cannot be changed later. EDIT

Location *
No organization BROWSE

Parent organization or folder

5. You will be redirected to the welcome page of the console, where on the notifications area, you see the notification of our project created, Click **SELECT PROJECT.**



Notifications

Create Project: Generative Language Client Just now

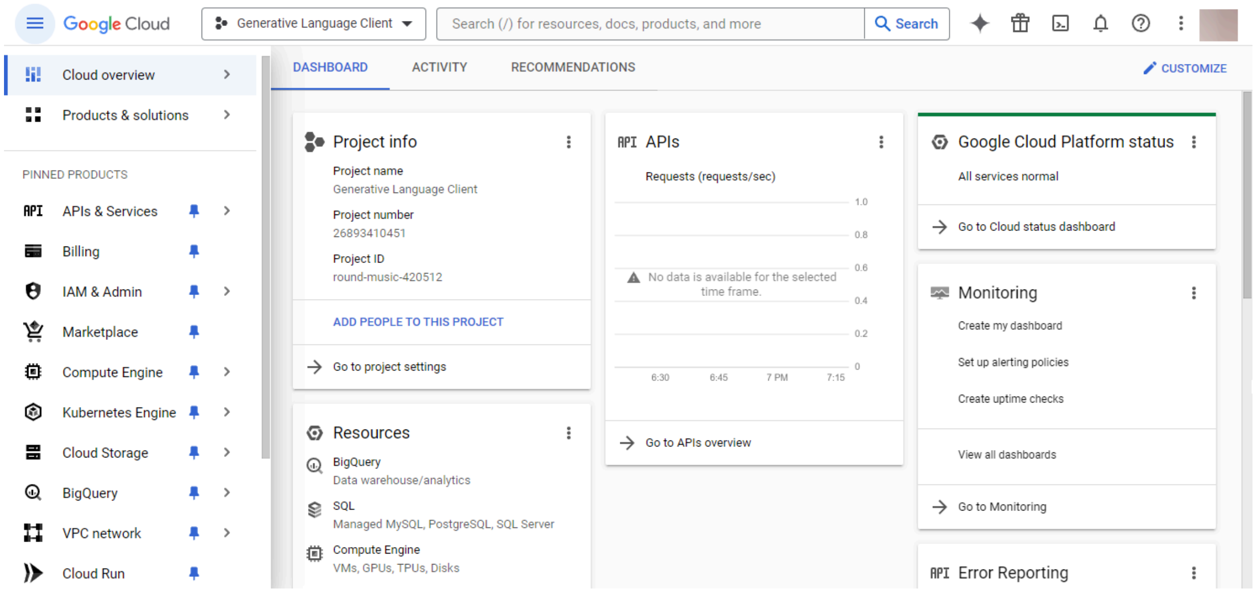
SELECT PROJECT

Welcome

Create or select a project to get up and running with Google Cloud. Learn more about projects

Try our most advanced

6. After Clicking the **SELECT PROJECT** you will be redirected to the dashboard of your project in the console



Project name
Generative Language Client
Project number
26893410451
Project ID
round-music-420512

Requests (requests/sec)
No data is available for the selected time frame.

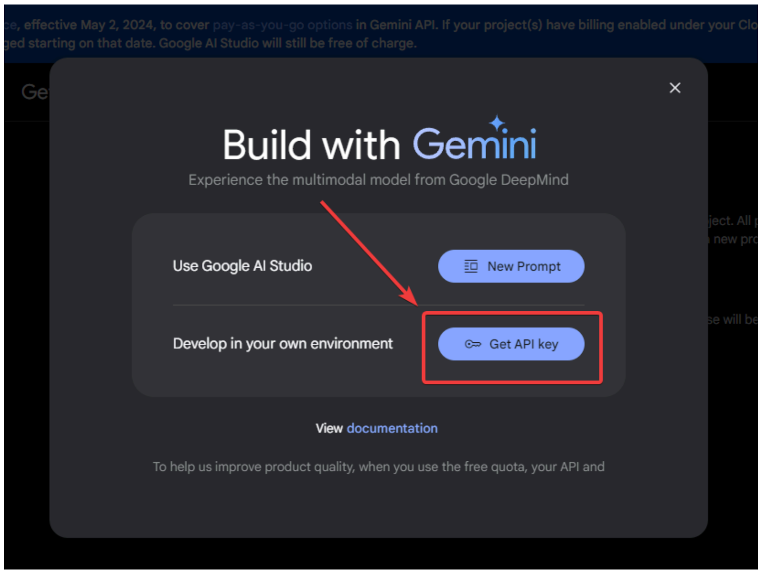
BigQuery
Data warehouse/analytics
SQL
Managed MySQL, PostgreSQL, SQL Server
Compute Engine
VMs, GPUs, TPUs, Disks

Creating an API Key from Google AI Studio:

1. Go to **Google AI Studio**'s API key page (sign in with your Google account that you use to create the project in the GCP console):

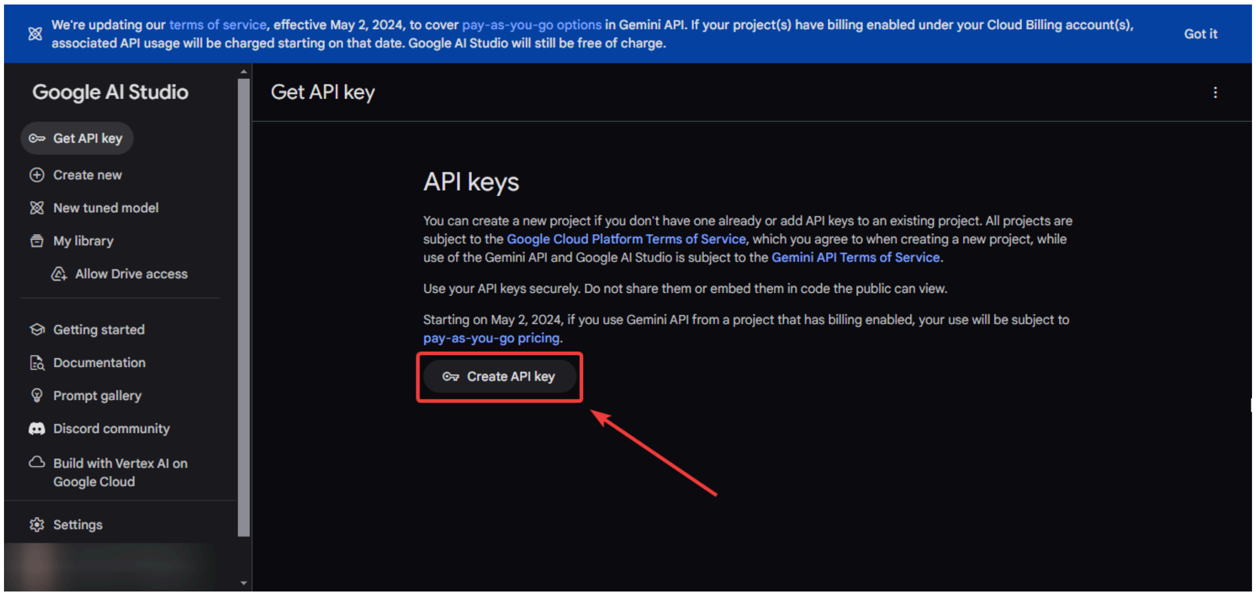
   https://aistudio.google.com/app/apikey

2. Where on the welcome dialogue box, click on the **Get API key** button


Build with Gemini
Experience the multimodal model from Google DeepMind
Use Google AI Studio
New Prompt
Develop in your own environment
Get API key
View documentation

To help us improve product quality, when you use the free quota, your API and

3. On the API Key page click on the **Create API Key** button:



We're updating our terms of service, effective May 2, 2024, to cover pay-as-you-go options in Gemini API. If your project(s) have billing enabled under your Cloud Billing account(s), associated API usage will be charged starting on that date. Google AI Studio will still be free of charge.

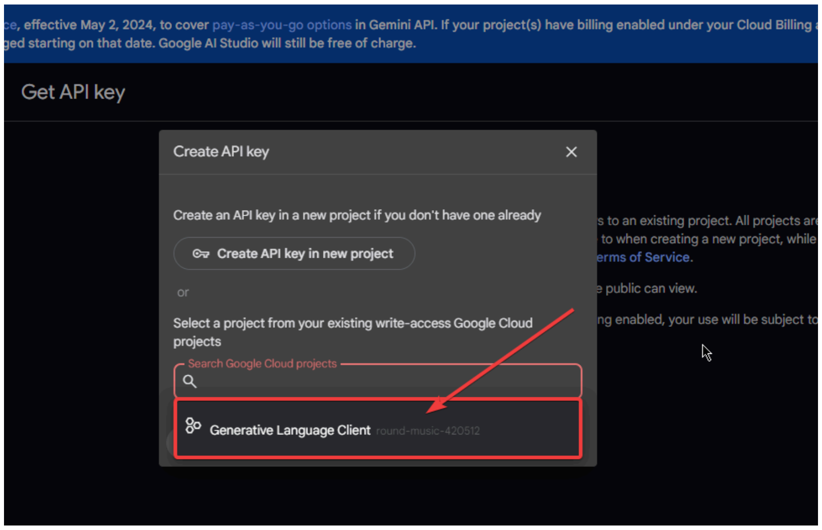
API keys

You can create a new project if you don't have one already or add API keys to an existing project. All projects are subject to the Google Cloud Platform Terms of Service, which you agree to when creating a new project, while use of the Gemini API and Google AI Studio is subject to the Gemini API Terms of Service.

Use your API keys securely. Do not share them or embed them in code the public can view.

Starting on May 2, 2024, if you use Gemini API from a project that has billing enabled, your use will be subject to pay-as-you-go pricing.

4. In the Create API Key dialogue box, select the project that we created on the GCP:



Get API key
Create API key

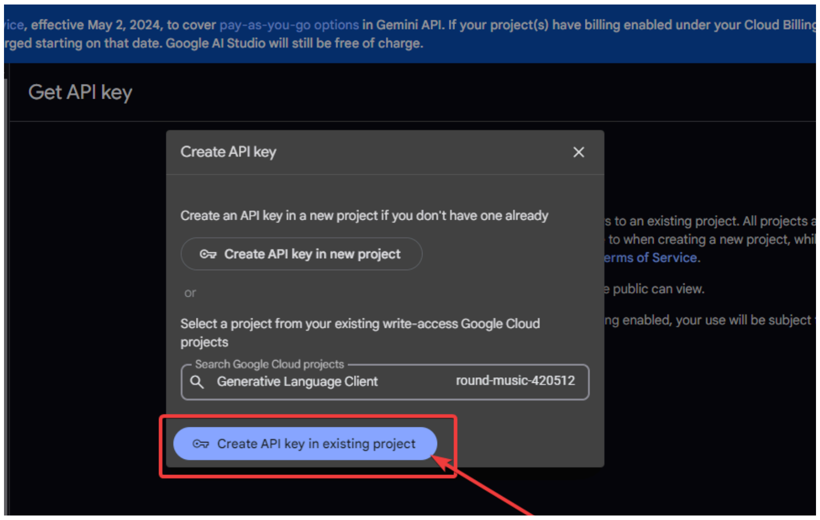
Create an API key in a new project if you don't have one already

Create API key in new project
or

Select a project from your existing write-access Google Cloud projects

Search Google Cloud projects
Generative Language Client round-music-420512

5. Then Click on **Create API Key in the existing project** button:


ice, effective May 2, 2024, to cover pay-as-you-go options in Gemini API. If your project(s) have billing enabled under your Cloud Billing
rged starting on that date. Google AI Studio will still be free of charge.

Get API key
Create API key
s to an existing project. All projects a
to when creating a new project, whi
erms of Service.
e public can view.
ng enabled, your use will be subject

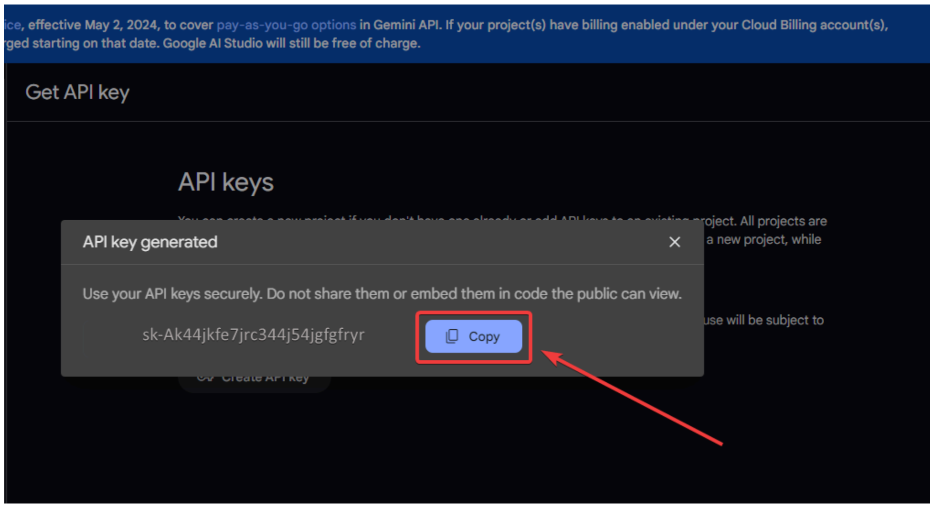
Create an API key in a new project if you don't have one already

Create API key in new project
or

Select a project from your existing write-access Google Cloud projects

Search Google Cloud projects: Generative Language Client round-music-420512

6. Then the API key for your project gets created, where click the **Copy** button:


ice, effective May 2, 2024, to cover pay-as-you-go options in Gemini API. If your project(s) have billing enabled under your Cloud Billing account(s),
rged starting on that date. Google AI Studio will still be free of charge.

Use your API keys securely. Do not share them or embed them in code the public can view.

sk-Ak44jkfe7jrc344j54jgfgfryr

After Creating the API Key, secure it as recommended. Remember to use your API keys securely and follow best practices to protect them.

Building a Streamlit Application with Gemini-Pro 1.0 Model:

Installation Guide for Gemini API Python SDK:

To begin using the Gemini API's Python SDK, you'll need to install the `google-generativeai` package. This package contains all the necessary tools to integrate Gemini API into your Python applications. Follow these steps to install the package:

1. **Open Your Command Line Interface**: Access your terminal on macOS/Linux or Command Prompt/PowerShell on Windows.

2. **Install the Package**: Execute the following command:

```
pip install -q -U google-generativeai
```

This command will download and install the google-generativeai package from the Python Package Index (PyPI). Once the installation is complete, you can import the package into your Python scripts and start building the Gemini AI based Streamlit Chat Application 😊.

Let's Start Building the Streamlit Chat Application with Gemini:

Importing the necessary libraries:

```
import streamlit as st
import google.generativeai as genai
```

This step involves adding the required Python libraries to your script. It's the foundation of your project, where you include packages that provide functionalities essential for your application, such as Streamlit for web app development and the Gemini API for AI-powered features.

Configuring the Gemini API:

```
# Configure the Gemini API
GOOGLE_API_KEY = 'ENTER_YOUR_API_KEY_HERE'
genai.configure(api_key=GOOGLE_API_KEY)
model = genai.GenerativeModel('gemini-pro')
```

Here, you'll set up the Gemini API by initializing it with your unique API key. This configuration is crucial as it authenticates your application's requests to the Gemini services, allowing you to access its AI capabilities.

Streamlit page configuration and title setup:

```
# Set Streamlit page configuration
st.set_page_config(
    page_title="Gemini Clone",
    page_icon="✨",
    layout="wide"
)
```

In this phase, you'll tailor the Streamlit page settings to suit your application's needs. You'll also define the title of your web page, which is displayed as the header in the browser tab and the main title on your app's interface.

Session and messages store:

```
# Check for messages in session and create a title if not exists
if "messages" not in st.session_state.keys():
    st.session_state.messages = [
        {"role": "assistant", "content": "Hello, this is Gemini and how I
can help you today?"}
    ]
    st.title(":rainbow[Howdy, How may I help you today?]")

# Display all messages
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.write(message["content"])
```

This part of the code is responsible for managing the user's session and storing the conversation messages. It ensures that the user's interaction with the AI is persistent across the session, providing a seamless chat experience.

User input and storing in session:

```
# Receive user input
user_input = st.chat_input()

# Store user input in session
if user_input is not None:
    st.session_state.messages.append({"role": "user", "content":
user_input})
    with st.chat_message("user"):
        st.write(user_input)
```

This section deals with capturing the user's input through the web interface. Once the user submits their message, the application will store it in the session, maintaining the flow of the conversation.

Generating AI response and displaying:

```
# Generate AI response and display
if st.session_state.messages[-1]["role"] != "assistant":
    with st.chat_message("assistant"):
        with st.spinner("Loading..."):
            ai_response = model.generate_content(user_input)
            st.write(ai_response.text)
    new_ai_message = {"role": "assistant", "content": ai_response.text}
    st.session_state.messages.append(new_ai_message)
```

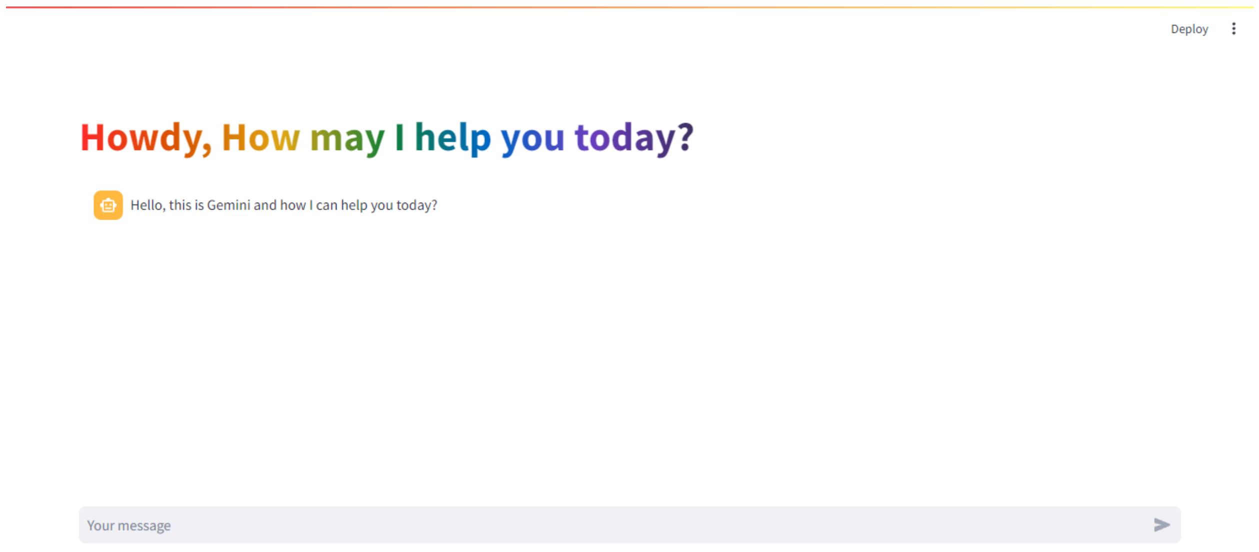
Finally, the application will generate a response using the Gemini API based on the user's input. The AI's reply is then displayed in the chat interface, allowing the user to receive feedback or answers to their queries.

Our Final Code would be like:

```python
import streamlit as st
import google.generativeai as genai

# Configure the Gemini API
GOOGLE_API_KEY = 'ENTER_YOUR_API_KEY_HERE'
genai.configure(api_key=GOOGLE_API_KEY)
model = genai.GenerativeModel('gemini-pro')

# Set Streamlit page configuration
st.set_page_config(
    page_title="Gemini Clone",
    page_icon="✨",
    layout="wide"
)

# Check for messages in session and create a title if not exists
if "messages" not in st.session_state.keys():
    st.session_state.messages = [
        {"role": "assistant", "content": "Hello, this is Gemini and how I
can help you today?"}
    ]
    st.title(":rainbow[Howdy, How may I help you today?]")

# Display all messages
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.write(message["content"])
# Receive user input
user_input = st.chat_input()

# Store user input in session
if user_input is not None:
    st.session_state.messages.append({"role": "user", "content":
user_input})
    with st.chat_message("user"):
        st.write(user_input)
# Generate AI response and display
```

```
if st.session_state.messages[-1]["role"] != "assistant":
    with st.chat_message("assistant"):
        with st.spinner("Loading..."):
            ai_response = model.generate_content(user_input)
            st.write(ai_response.text)
    new_ai_message = {"role": "assistant", "content": ai_response.text}
    st.session_state.messages.append(new_ai_message)
```

Our Application would be like this:



Howdy, How may I help you today?

Hello, this is Gemini and how I can help you today?