LAB

Use IBM Granite models for code generation and programming tasks

IBM **SkillsBuild**

Contents

Introduction

In this lab, you'll use an IBM Granite model to generate Python code for a given scenario. You'll apply your knowledge of prompting techniques to define and execute prompts to generate code using IBM Granite.

Software requirements

To complete this lab, you'll need access to a Replicate account, which allows you to use AI models to perform tasks. You'll also need an API token from your Replicate account. An API token is like a digital key that lets the lab securely connect to your Replicate account. This token will be securely added to your Google Colab environment so the lab can run correctly.

While you don't need to know Python to follow along, familiarity with it might help you better understand the code created during the lab.

Objective

After completing this lab, you should be able to:

- Use IBM Granite models for code generation and programming tasks

Lab steps

This lab requires you to complete the following steps:

- Step 1: Create a GitHub account
- Step 2: Create a Replicate account
- Step 3: Sign up for Google Colab
- Step 4: Load the Jupyter notebook and initialize the model
- Step 5: Generate code using the IBM Granite model

Estimated duration to complete

30 minutes

Scenario

Background information

Reader's Verse is a local bookstore planning to build an online presence by creating a website that allows readers to search for and view its catalog and check availability before visiting the physical store. You are the web designer assigned to this project, and Reader's Verse is your client.

Challenge

The client expects you to complete the project quickly and requests a working prototype. You are already handling multiple projects and have limited time to work on this request. You have a basic understanding of programing and coding but lack the depth of expertise needed to create a website from a programming perspective.

Solution

You consider using IBM Granite models, which are designed for code generation, code explanation, and code editing tasks. These models are extensively trained with code written in 116 programming languages and are used for various code generation and related activities.

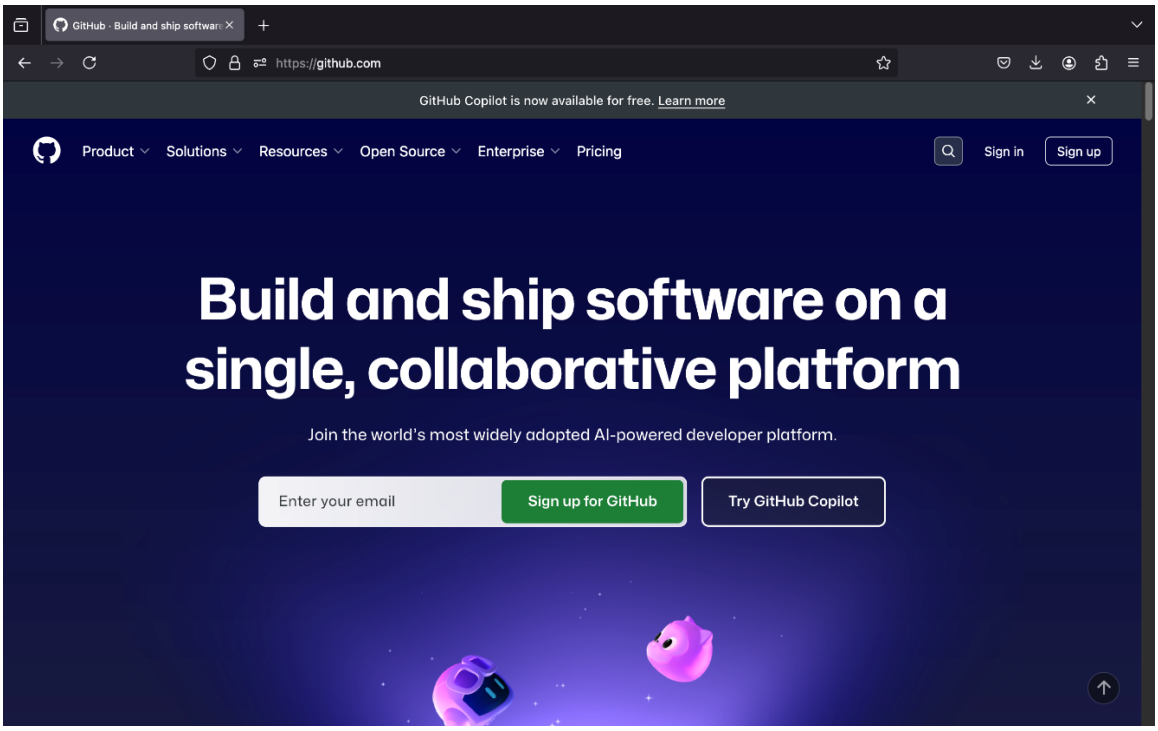
Step 1: Create a GitHub account

Overview

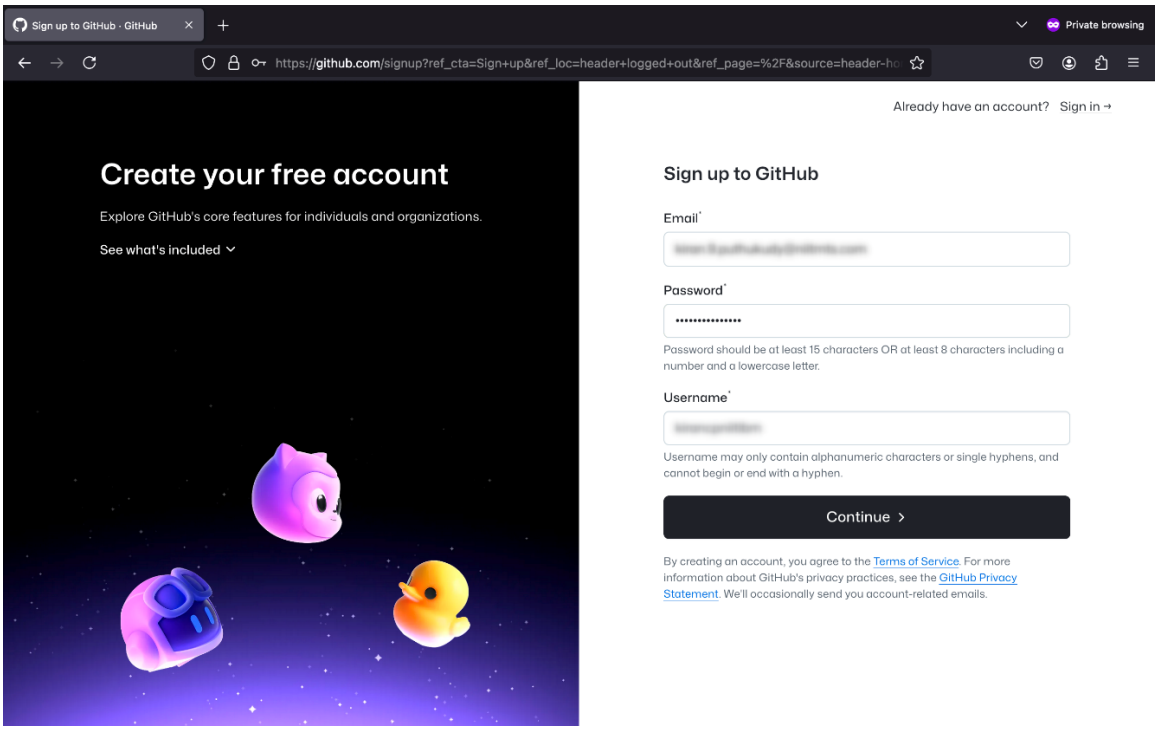
In this step, you'll set up a GitHub account to register for a Replicate account. GitHub is a platform that helps developers store, manage, and share code, while also supporting collaboration through tools like version control, bug tracking, and task management. This setup ensures you have access to the Replicate cloud environment needed to complete the lab efficiently.

Instructions

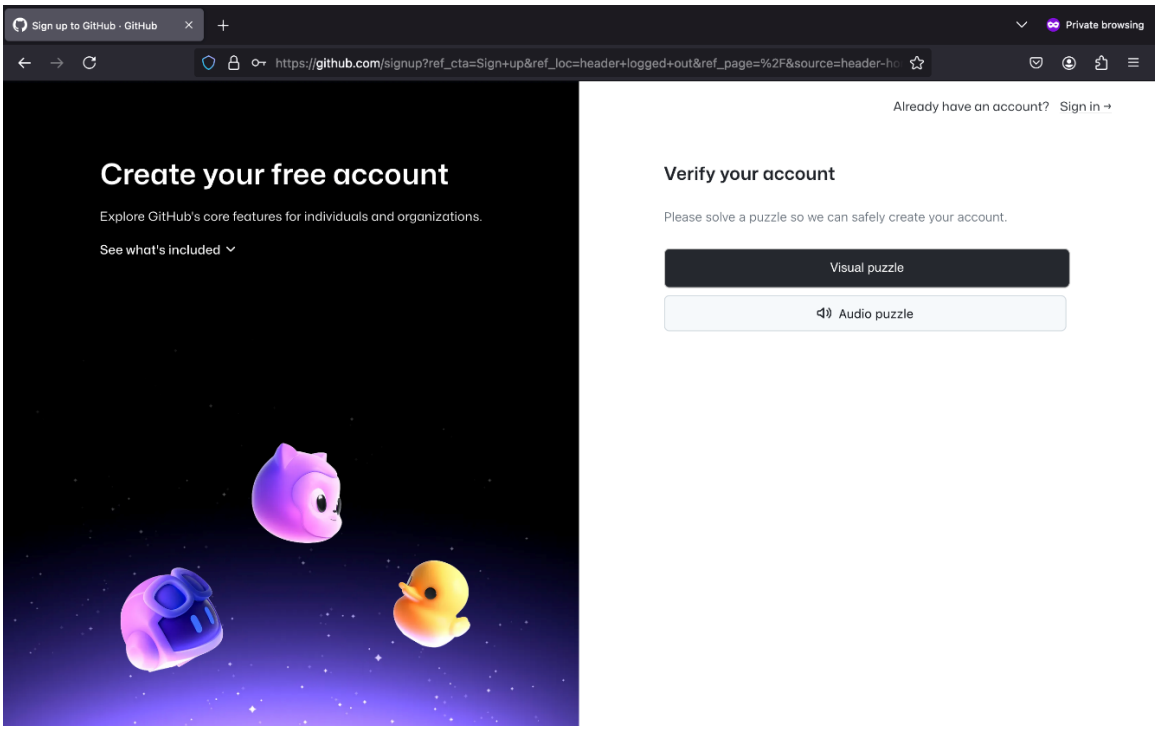
1. Go to the GitHub website to create a GitHub account and select **Sign up**.



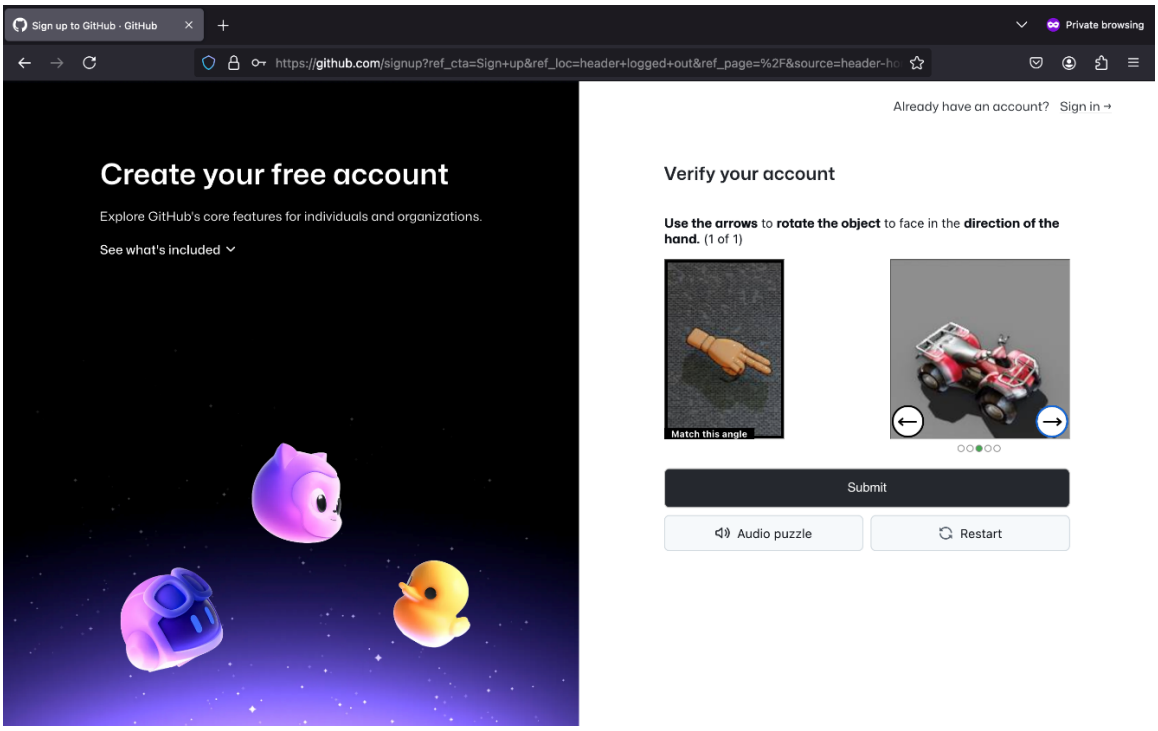
2. Type your details in the **Email**, **Password**, and **Username** fields. Then, select **Continue**.



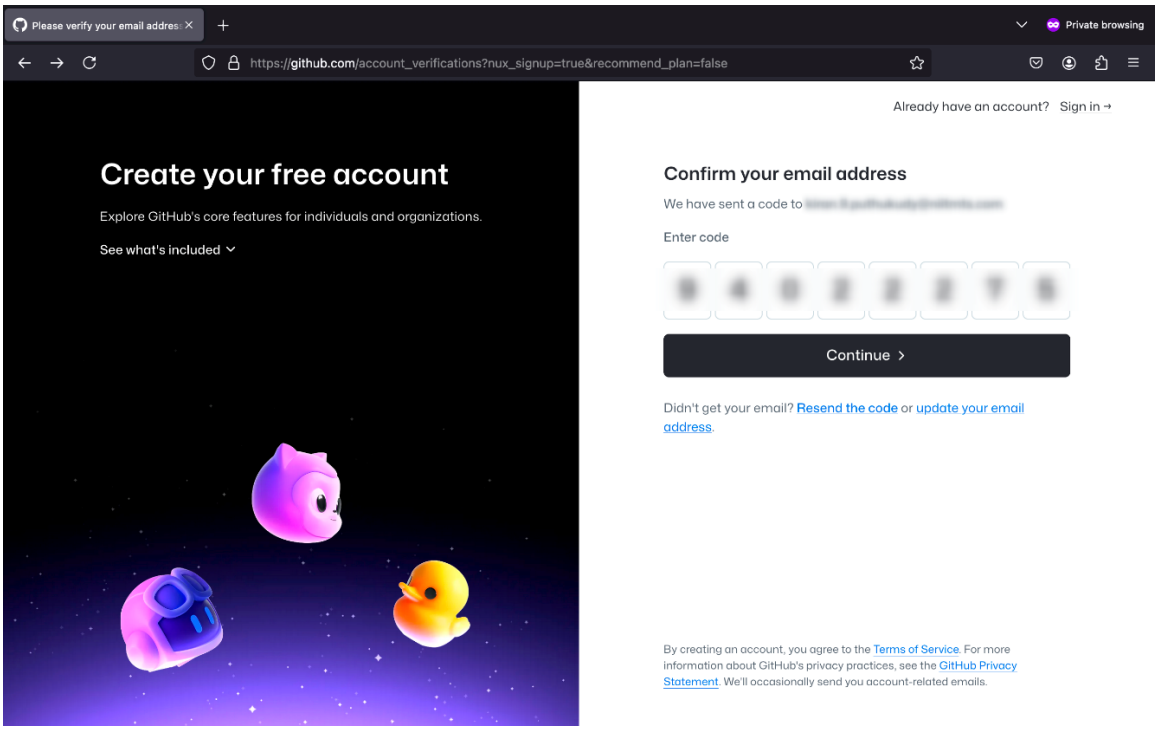
3. Select the **Visual puzzle** option to verify your account.



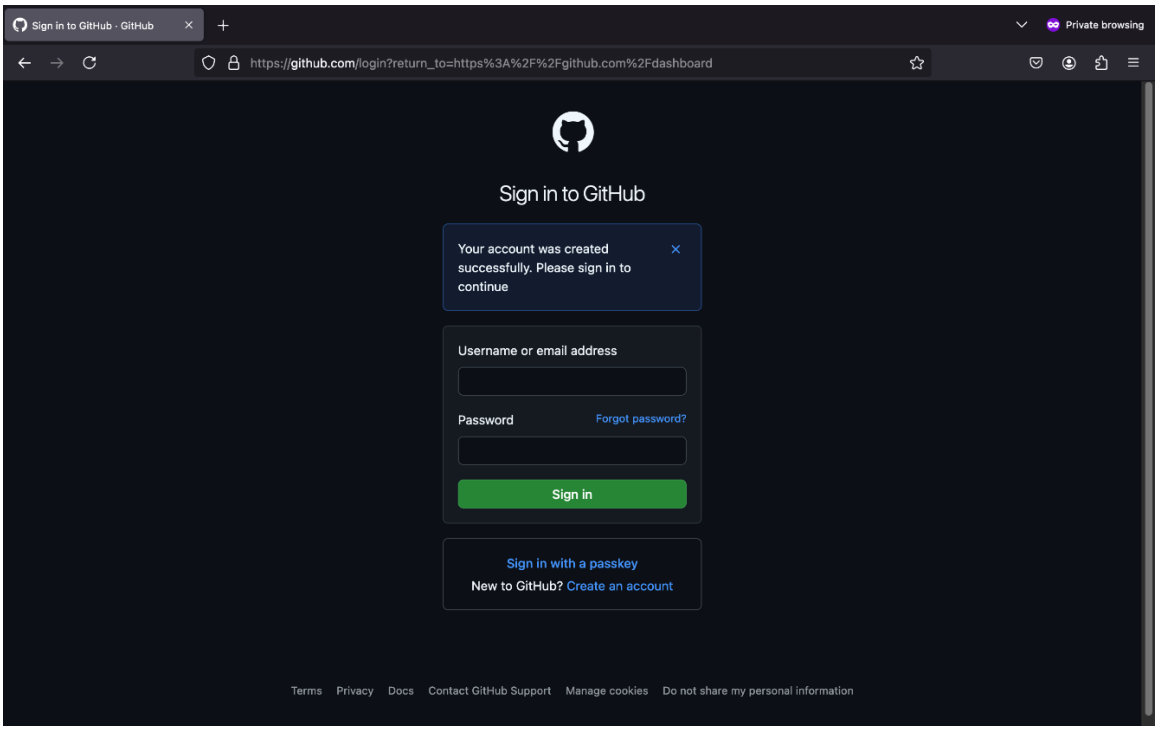
4. Solve the puzzle to verify your account and select **Submit**.



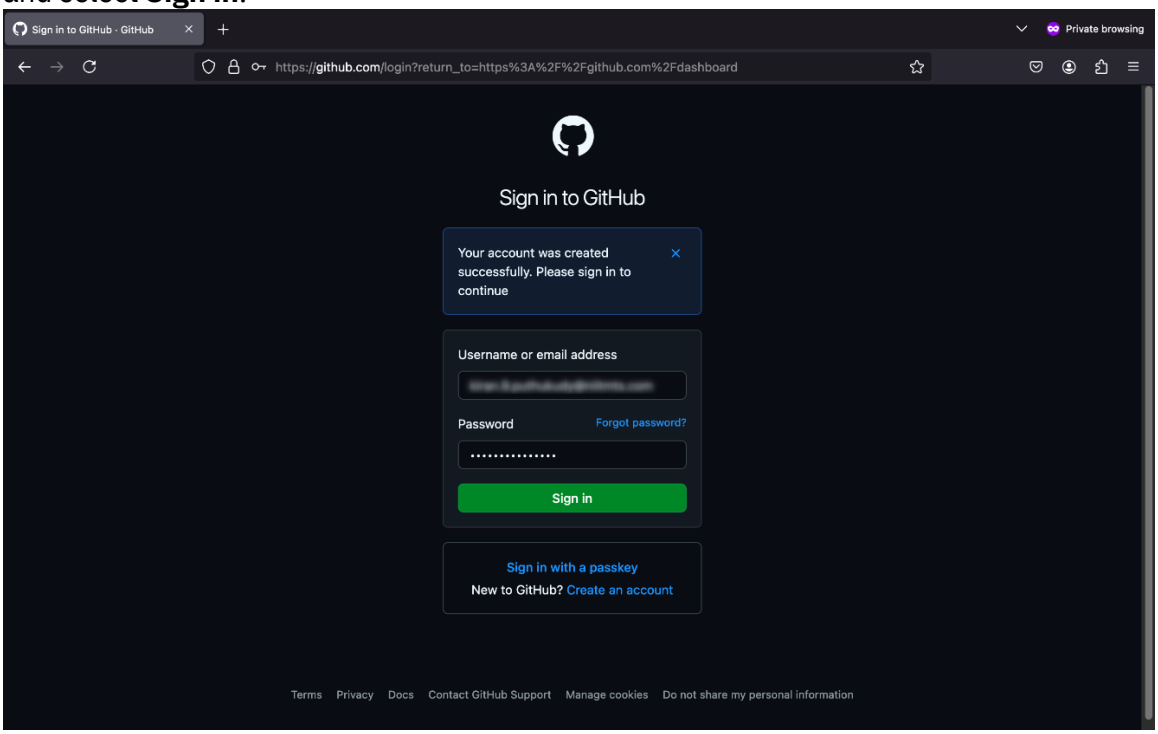
5. Type the confirmation code in the **Enter code** field to confirm your email ID and select **Continue**.



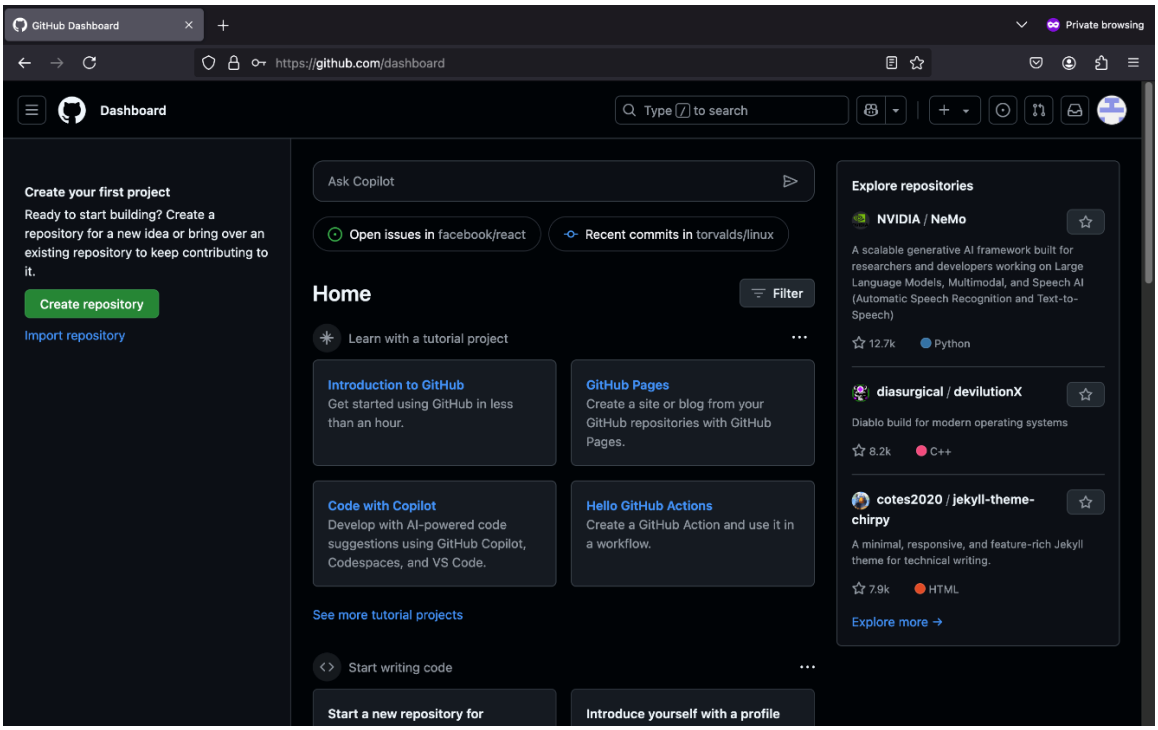
6. You'll see a confirmation message on successful completion of your GitHub account.



7. Type your credentials in the **Username or email address** and **Password** fields and select **Sign in**.



8. The GitHub dashboard displays to indicate that you have successfully logged in to your GitHub account.



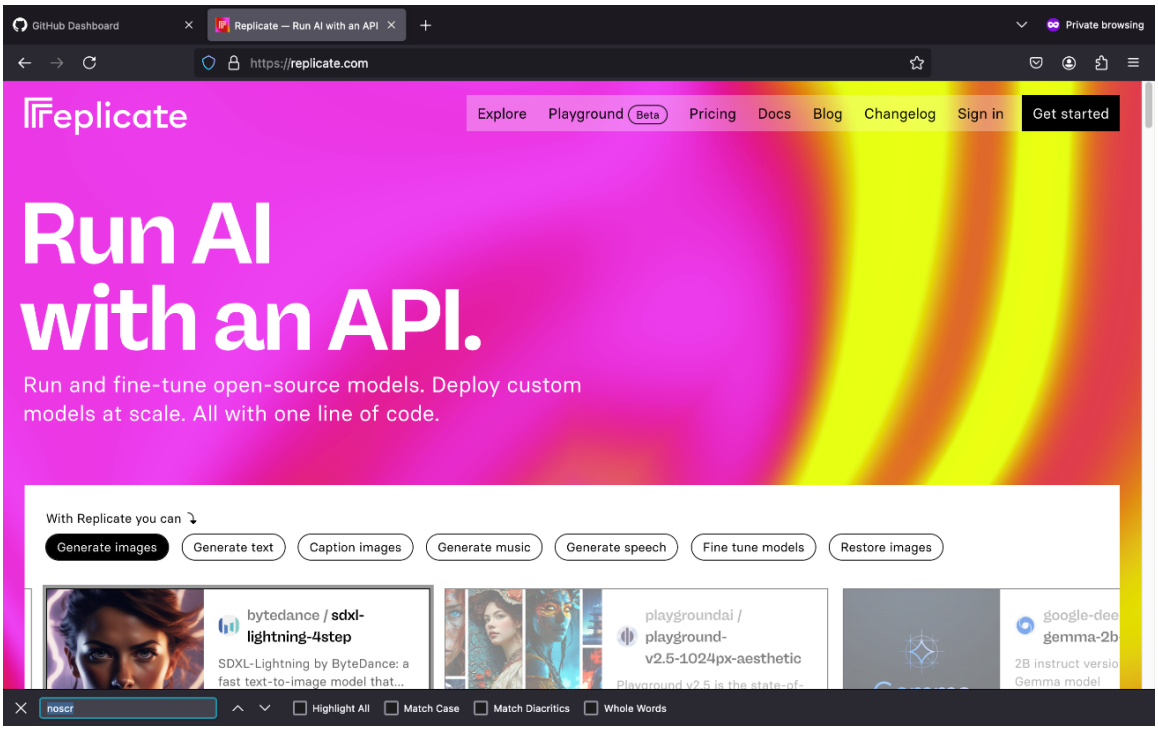
Step 2: Create a Replicate account

Overview

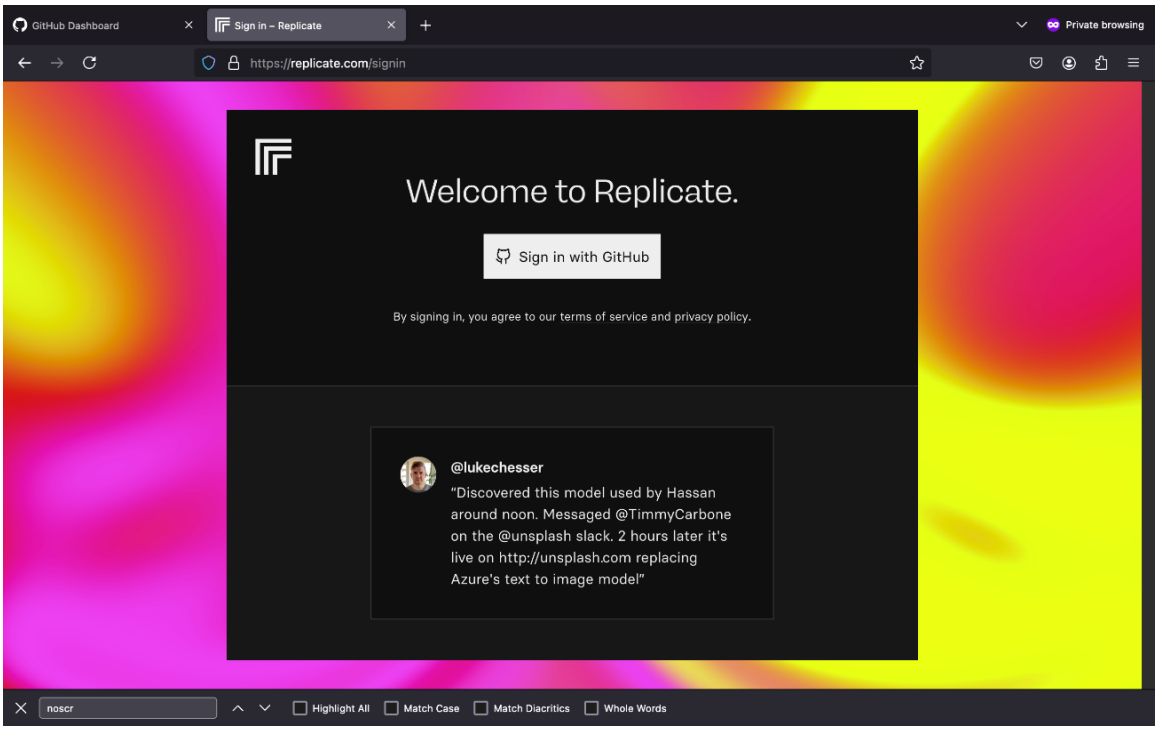
In this step, you will use your GitHub account to register for a Replicate account. Replicate is a cloud-based platform that lets you use AI models without needing complex hardware. As part of this step, you'll create a Replicate token. A token is like a secure digital key that allows the lab to connect to your Replicate account and access the tools needed to run the lab in Google Colaba.

Instructions

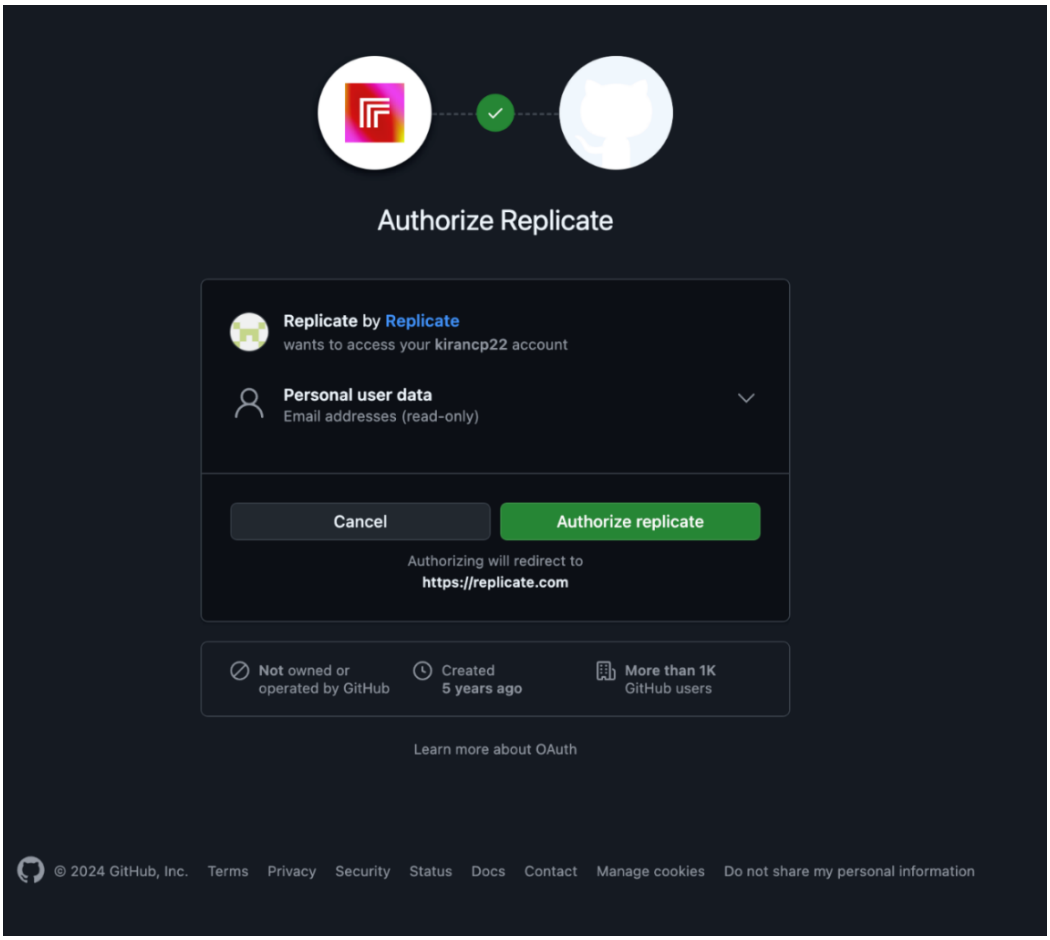
1. Go to the Replicate website and select **Get started**.



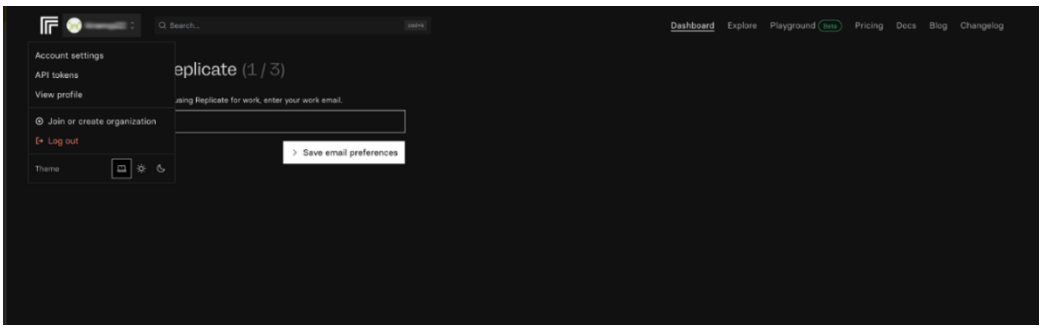
2. Select **Sign in with GitHub** on the Welcome to Replicate page.



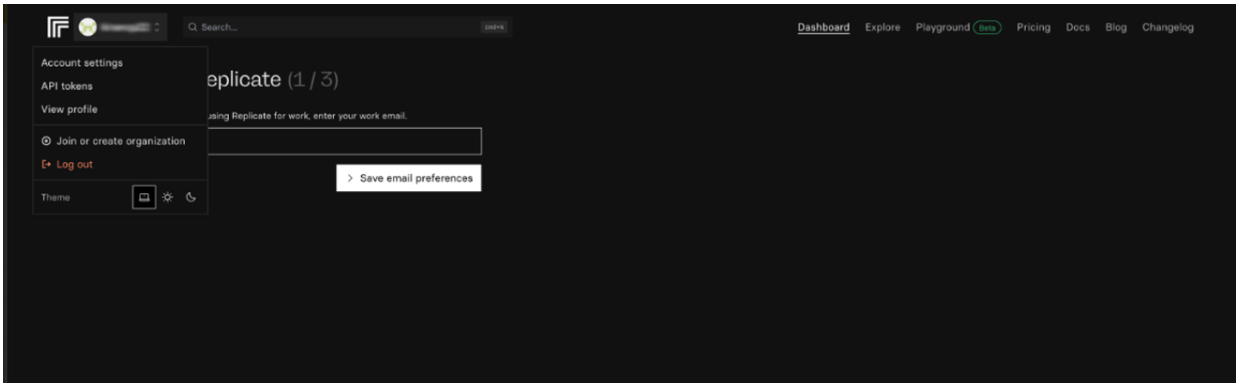
3. Select **Authorize replicate** to continue.



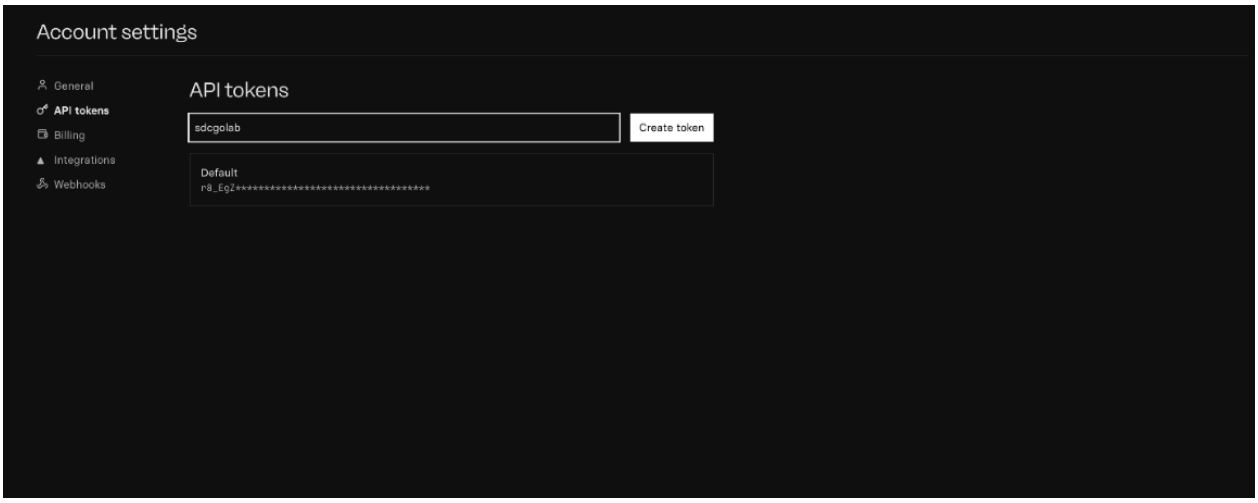
4. To create a Replicate token, select the **account settings** icon on the navigation bar.



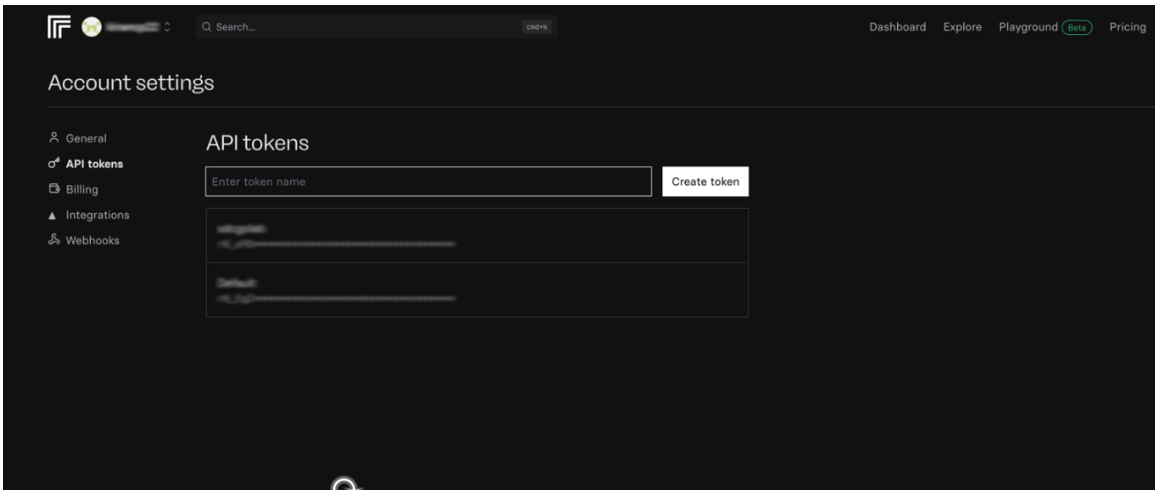
5. Select **API tokens** on the “Account settings” menu.



6. Type a name for the token in the **API tokens** field and select **Create token**.

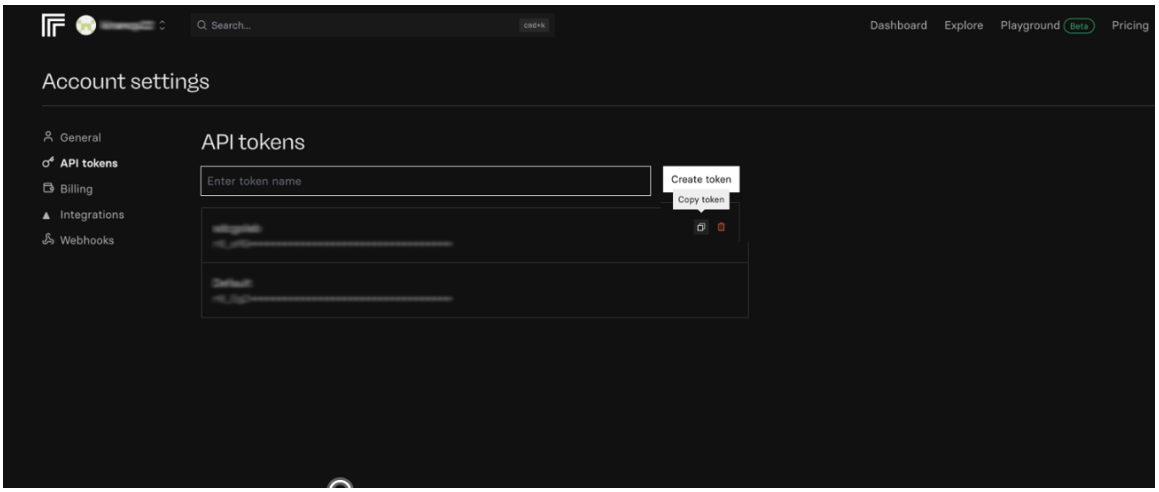


7. Your API token displays on the screen following the **API tokens** field.

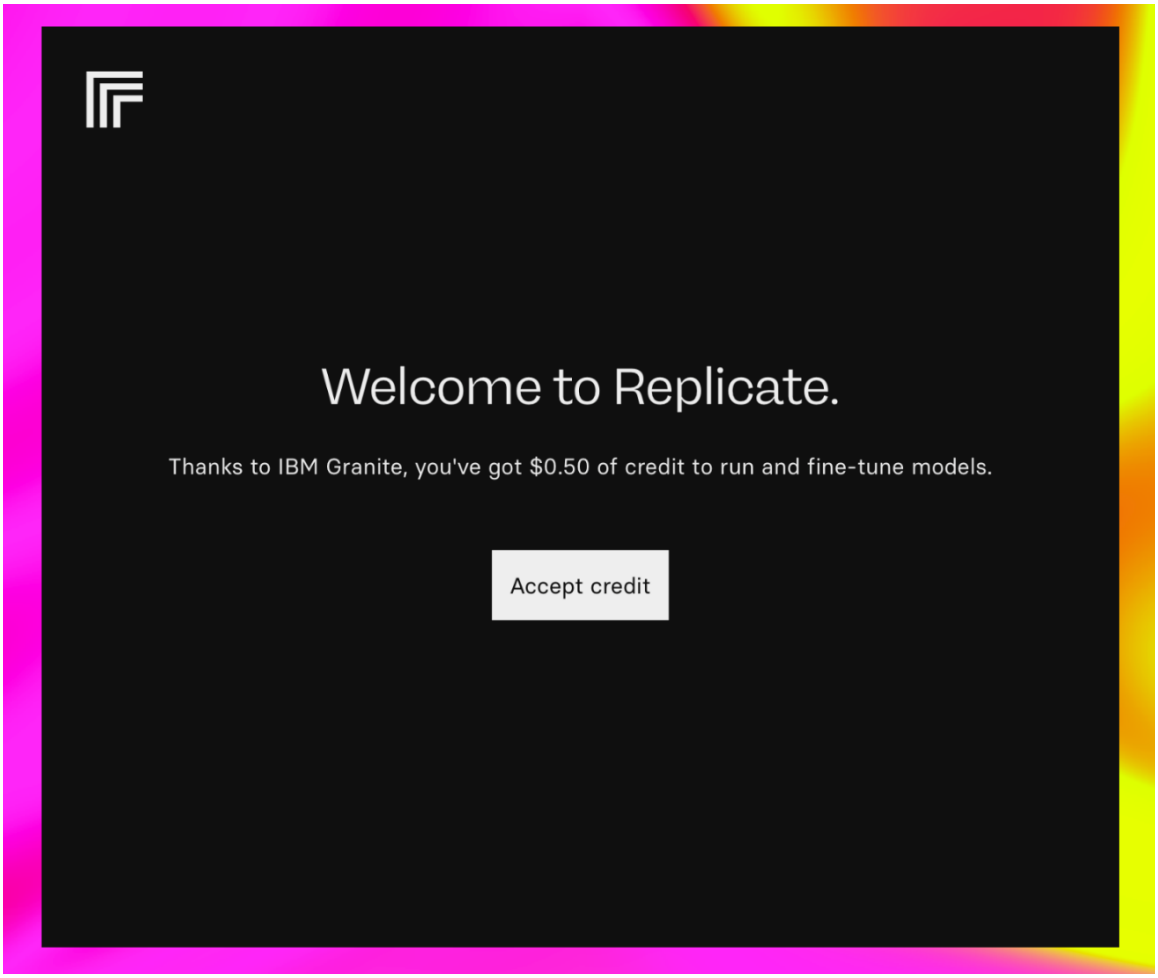


8. Select the **Copy token** icon to copy the Replicate API token.

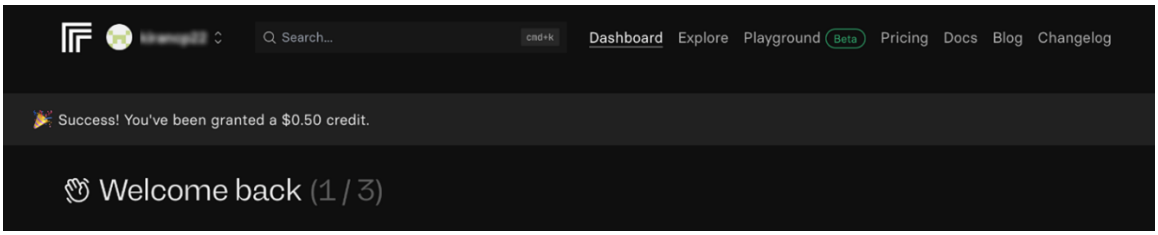
   Note: Save the Replicate token because you'll need the Replicate API token to run the Google Colab instance later in this lab.



9. Go to the Replicate invite link to claim 50 cents in Replicate credits, which you'll use to run the lab. Select **Accept credit** to claim the credits.



10. You will see a confirmation message indicating that 50 cents credit has been added to your Replicate account as follows.



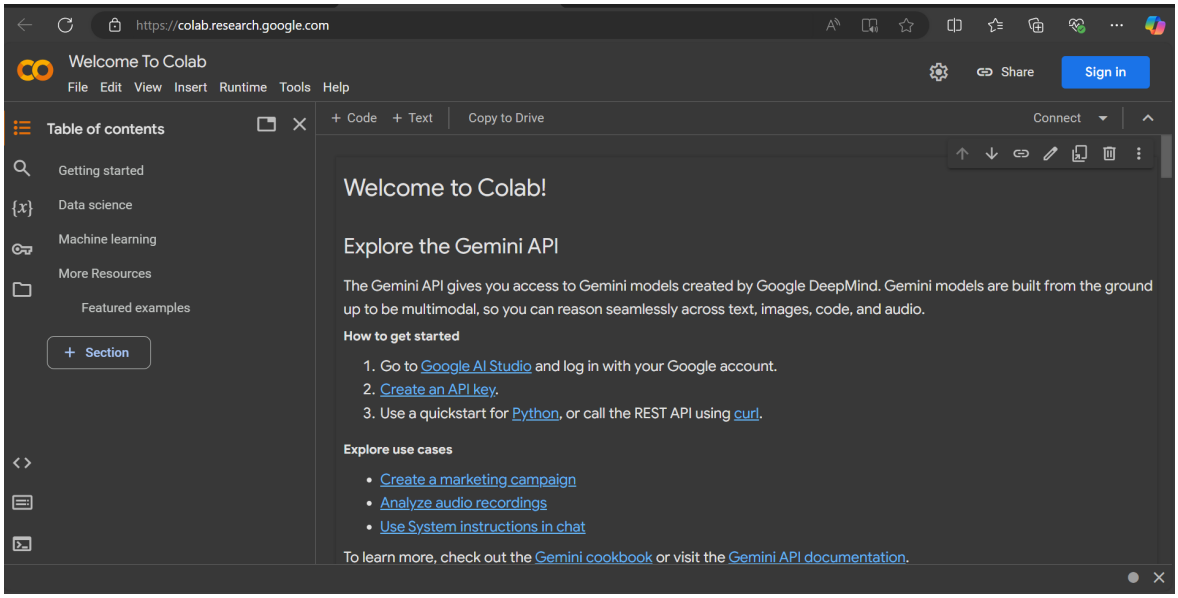
Step 3: Sign up for Google Colab

Overview

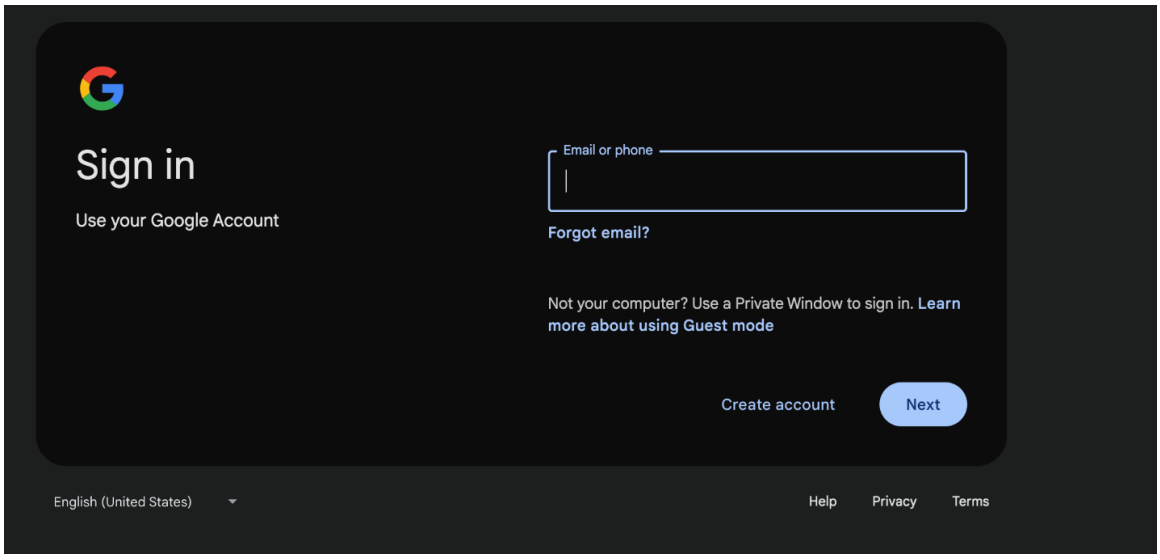
In this step, you will set up a Google Colab account. Google Colab is a free cloud platform that lets you run code in notebooks, which are commonly used for machine learning, data science, and AI tasks. This account will allow you to install and use the tools needed to complete the lab.

Instructions

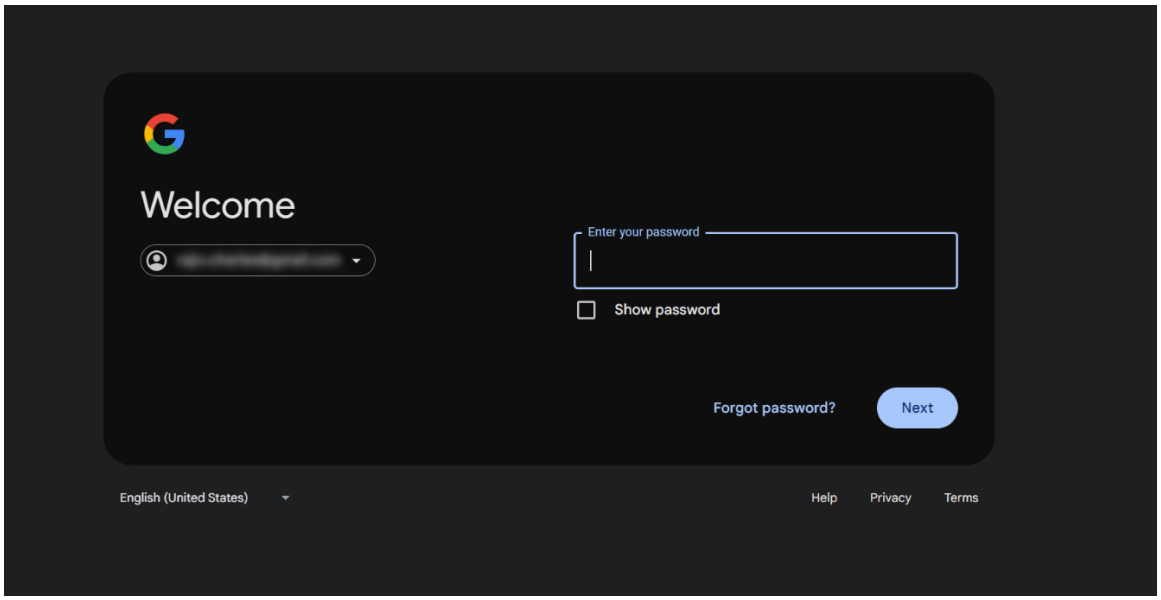
1. Go to the Google Colab website to sign up for a Google Colab account and select **Sign in**.



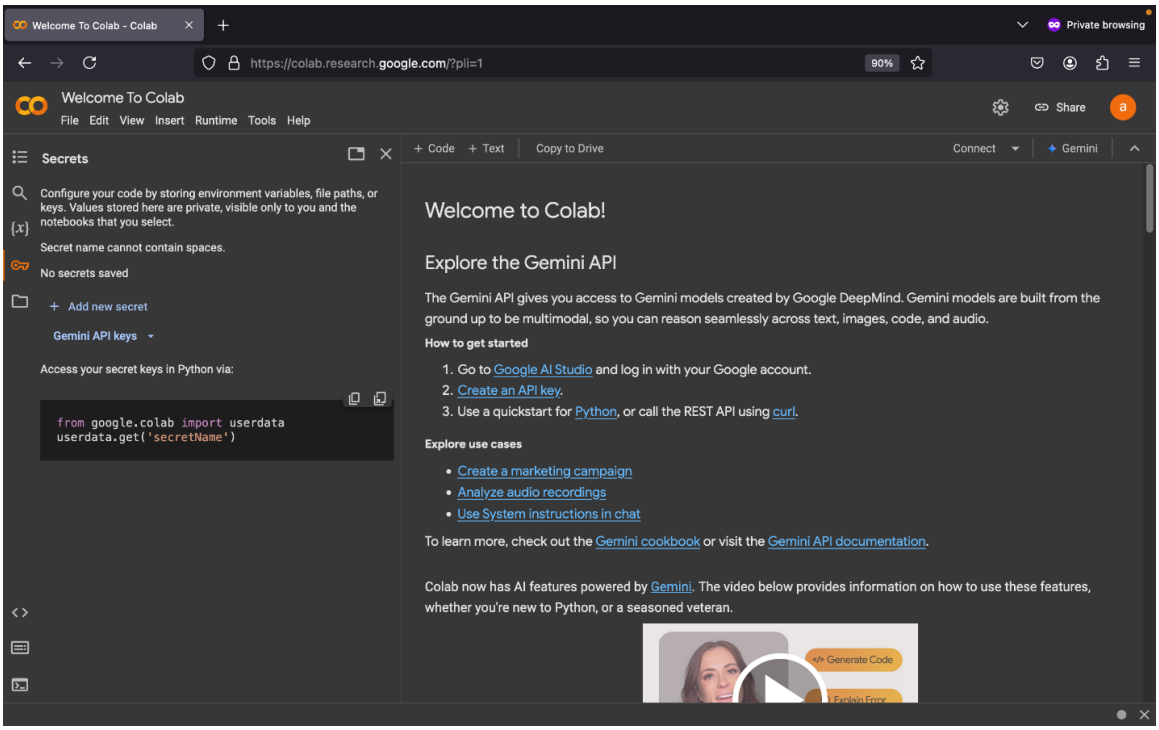
2. Type your Google email or phone number in the **Email or phone** field and select **Next**.



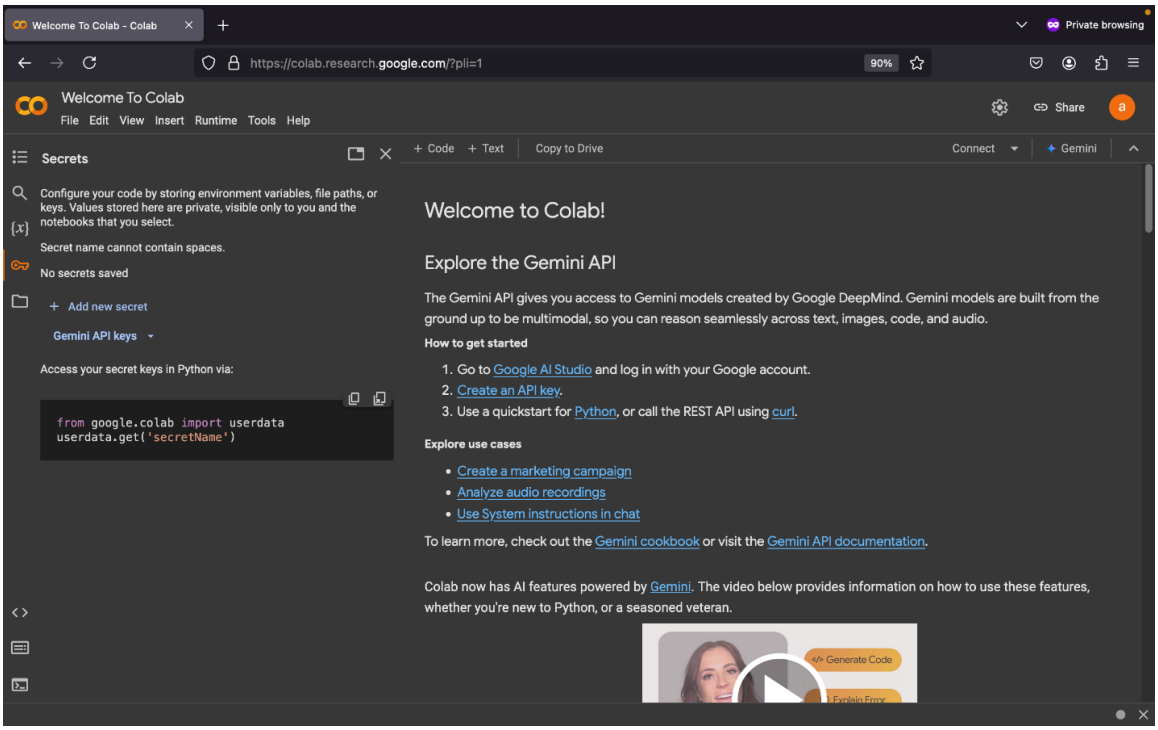
3. Type your password in the **Enter your password** field and select **Next**.



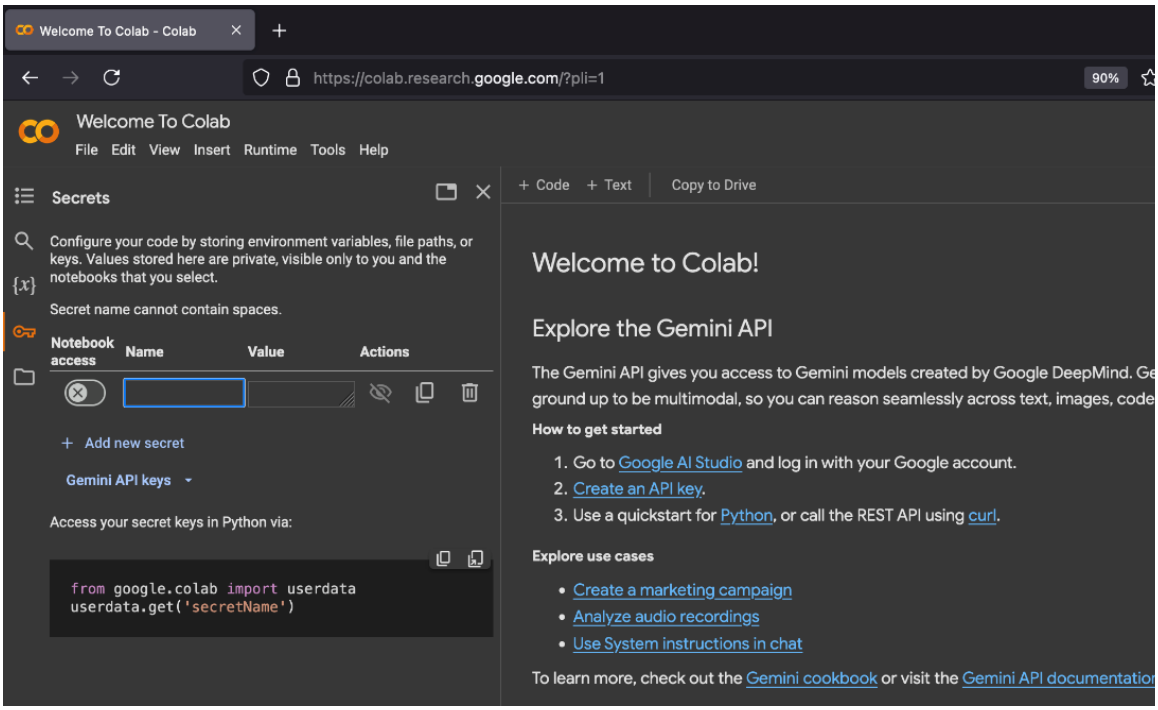
4. Next, select the **Key** icon on the **sidebar** menu of the Welcome to Colab homepage to store your Replicate API token in Google Colab secret.



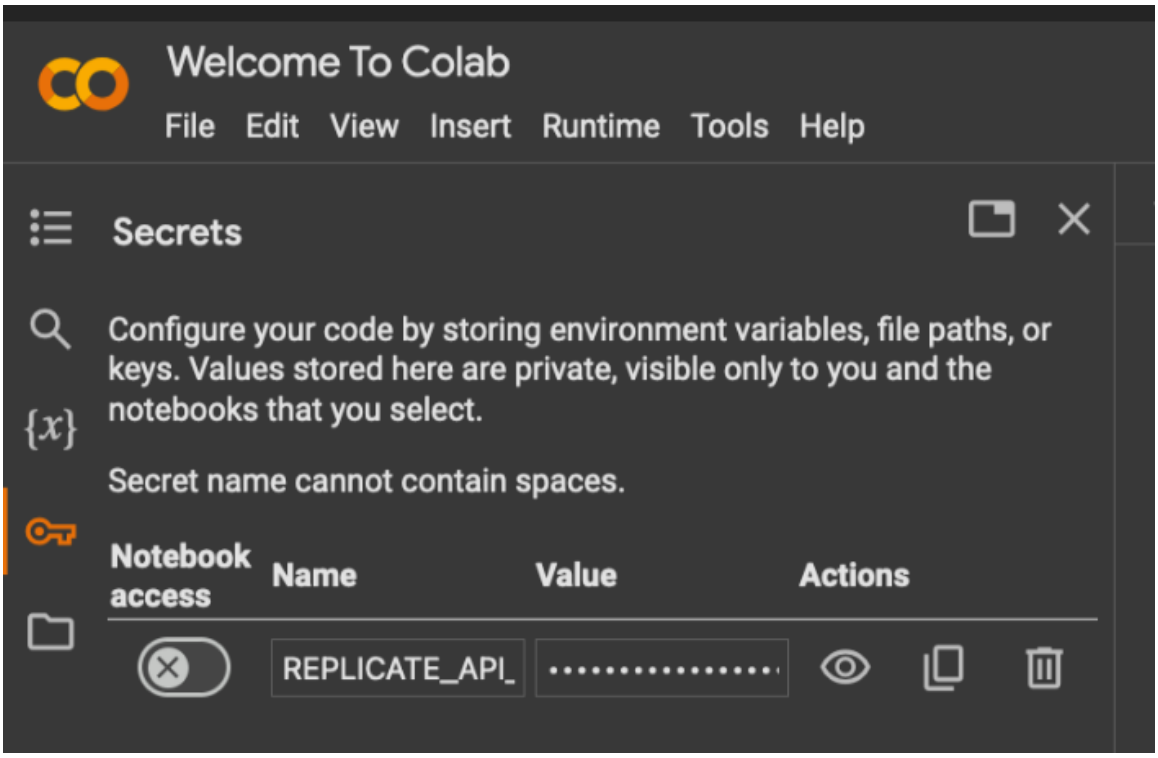
5. Select **Add new secret**.



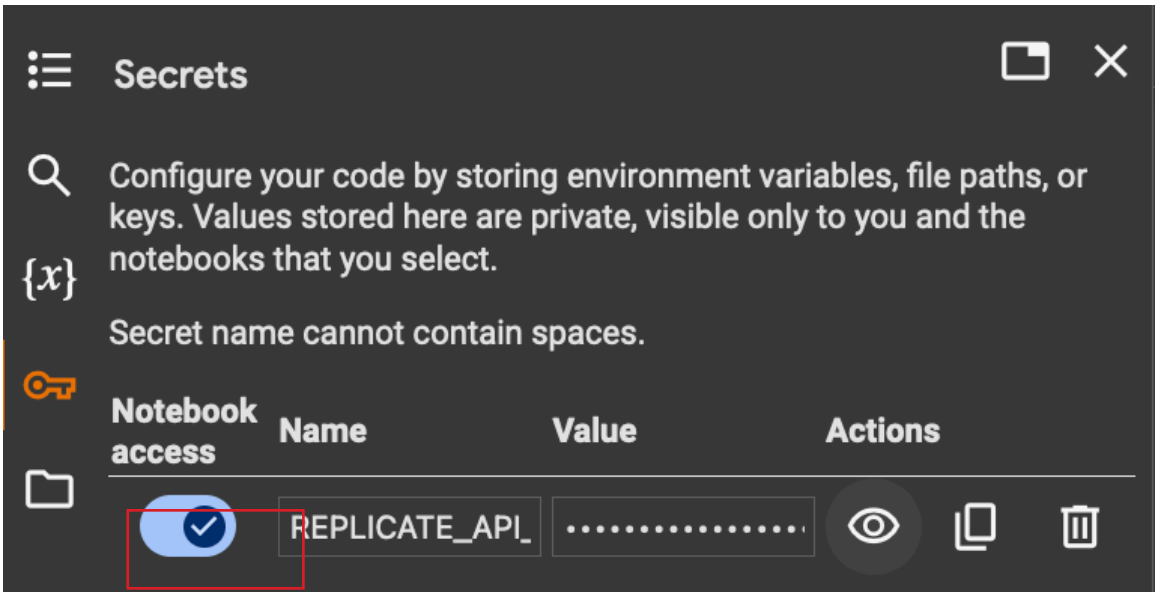
6. Type **REPLICATE_API_TOKEN** in the **Name** field.



7. Paste your Replicate API token into the **Value** field.



8. Select the **toggle** button to enable **Notebook access**. Next, select **Close** to exit the configuration.



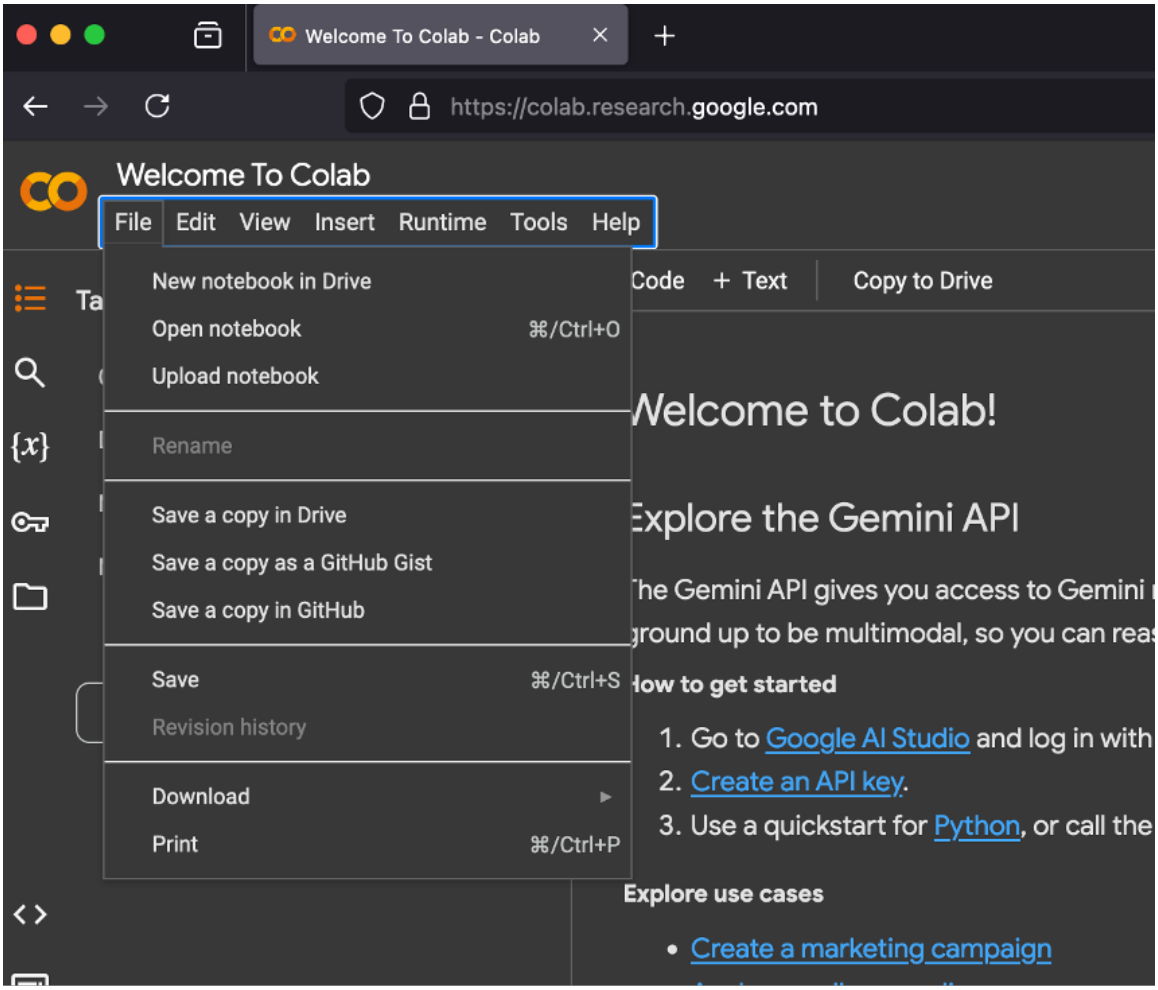
Step 4: Load the Jupyter notebook and initialize the model

Overview

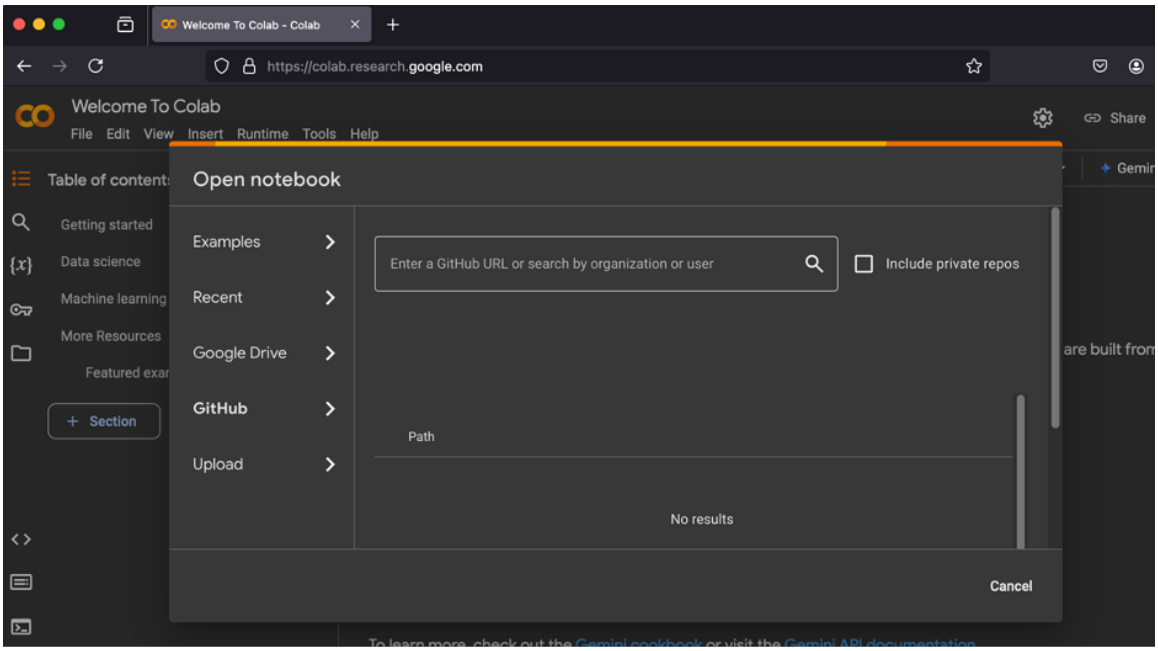
In this step, you will load a Jupyter notebook containing the code required to perform this lab. A Jupyter notebook is a shareable document that combines computer code, plain language descriptions, data, and visualizations. You'll load a Jupyter notebook file from GitHub into Google Colab and initialize the IBM Granite model to generate code for the given scenario.

Instructions

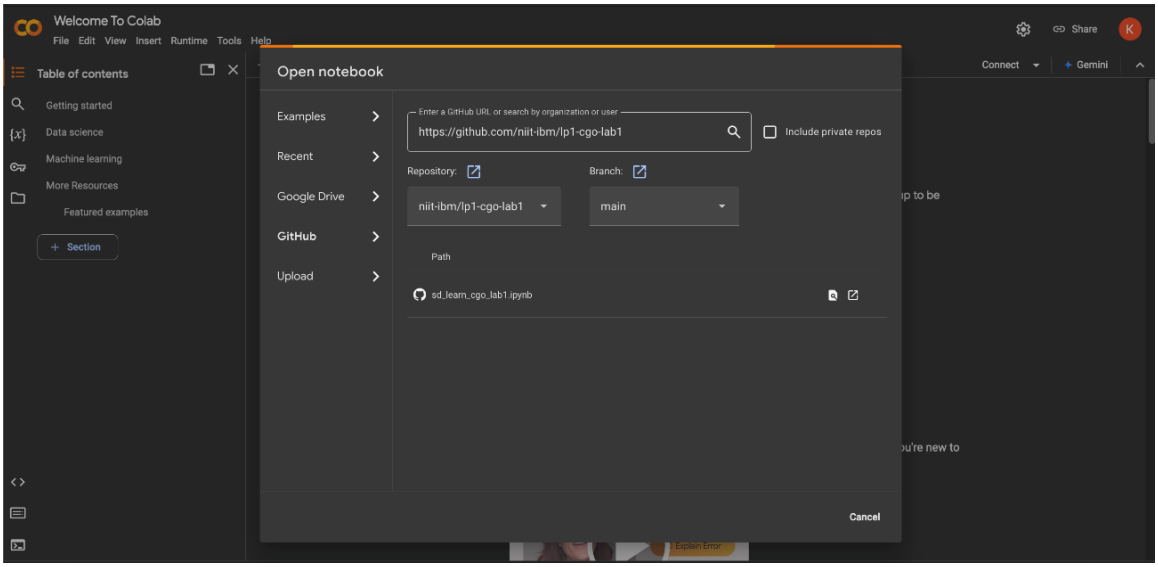
1. In your Google Colab workspace, on the **File** menu, select **Open notebook**.



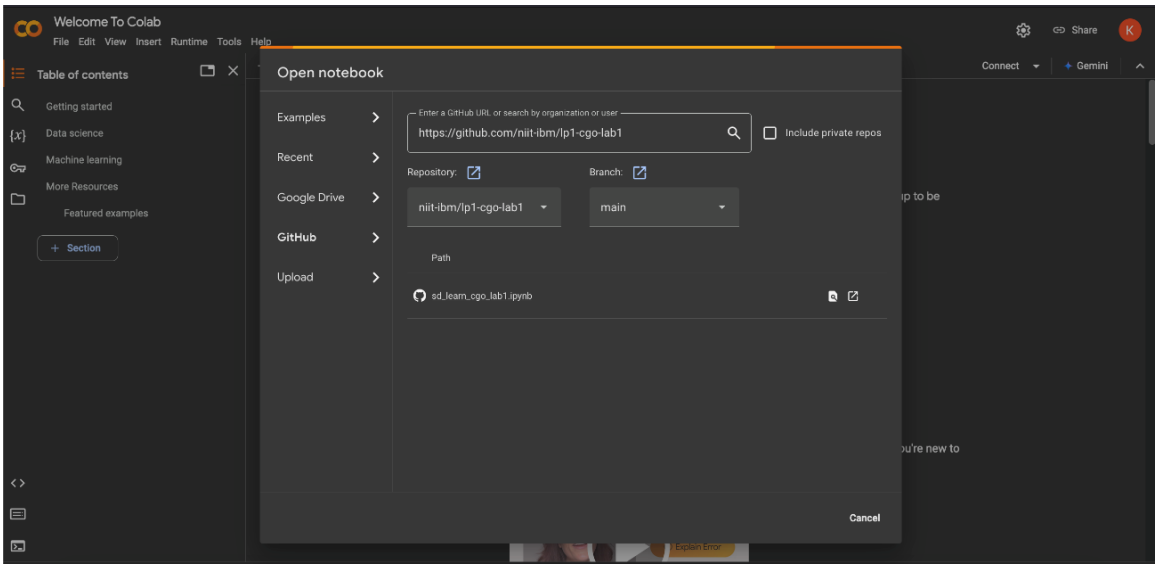
2. Select the **GitHub** tab.



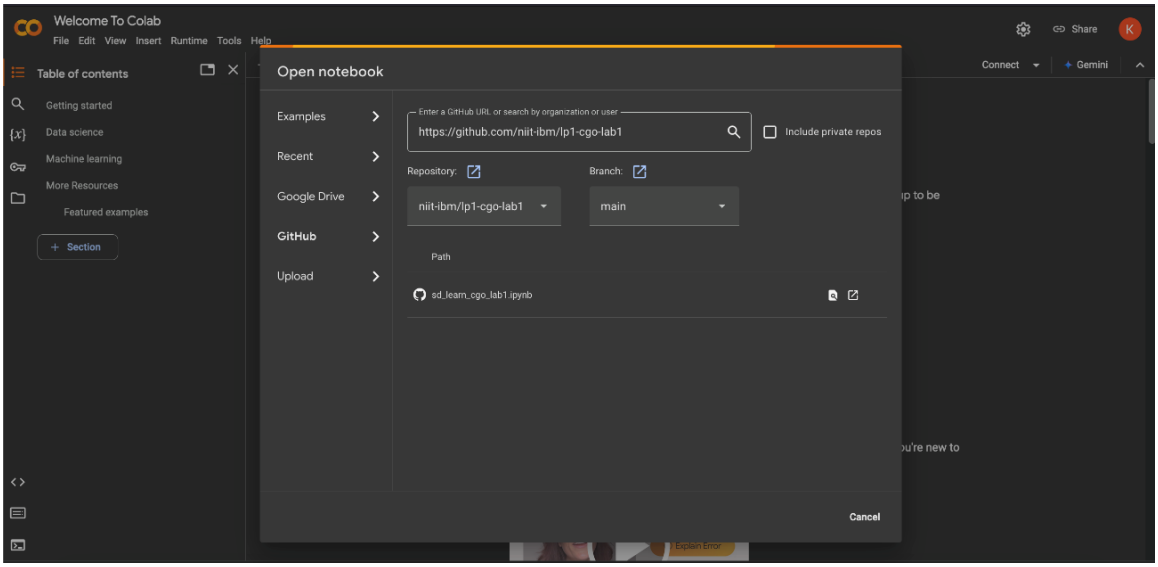
3. In the **Enter a GitHub URL or search by organization or user** field, copy the following URL: https://github.com/niit-ibm/lp1-cgo-lab1 and select the **magnifying glass** icon.



4. Select **main** in the **Branch** section.

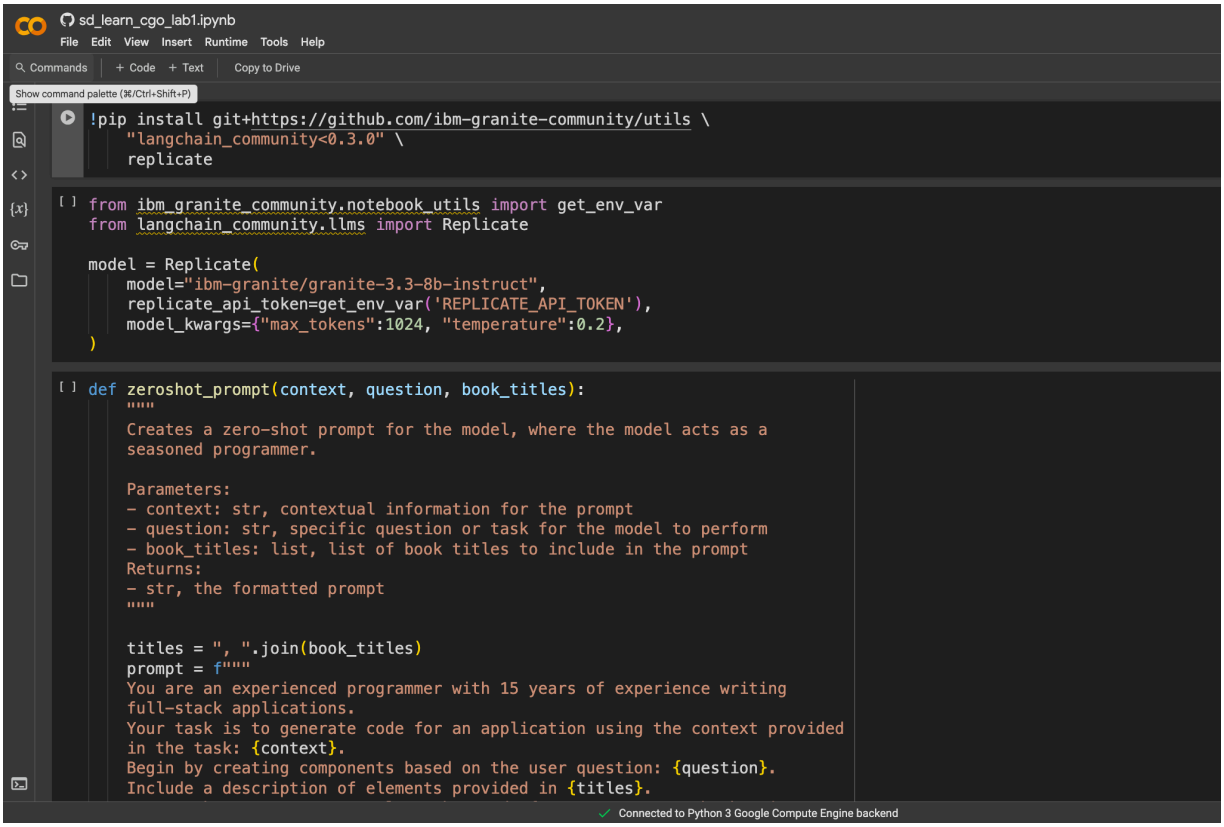


5. Select the **sd_learn_cgo_lab1.ipynb** notebook in the **Path** section to open the notebook in Colab.

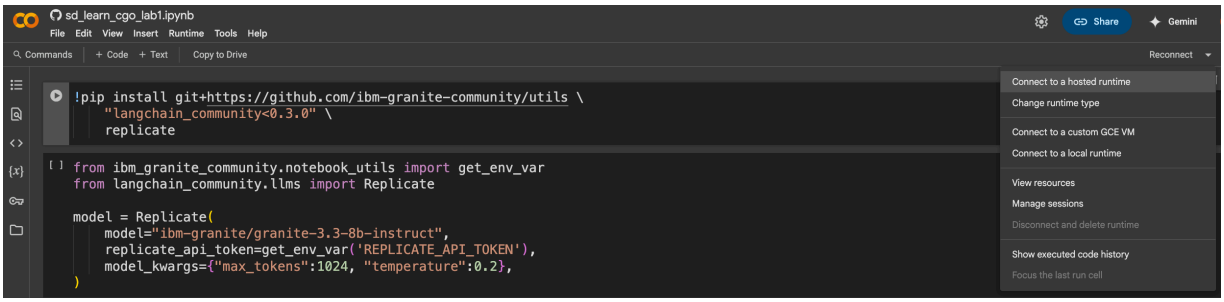


6. You'll now have the sd_learn_cgo_lab1 notebook opened in the Colab workspace.

   Note that each row in the notebook is referred to as a **cell**. The cells in the notebook are not numbered. The notebook is organized in such a way that you'll need to run the code in each cell sequentially, beginning from the first cell, to complete this lab.



7. You'll need to start a new instance in the Google Colab runtime to run the code in the Jupyter notebook. Select **Connect** on the Google Colab navigation bar.



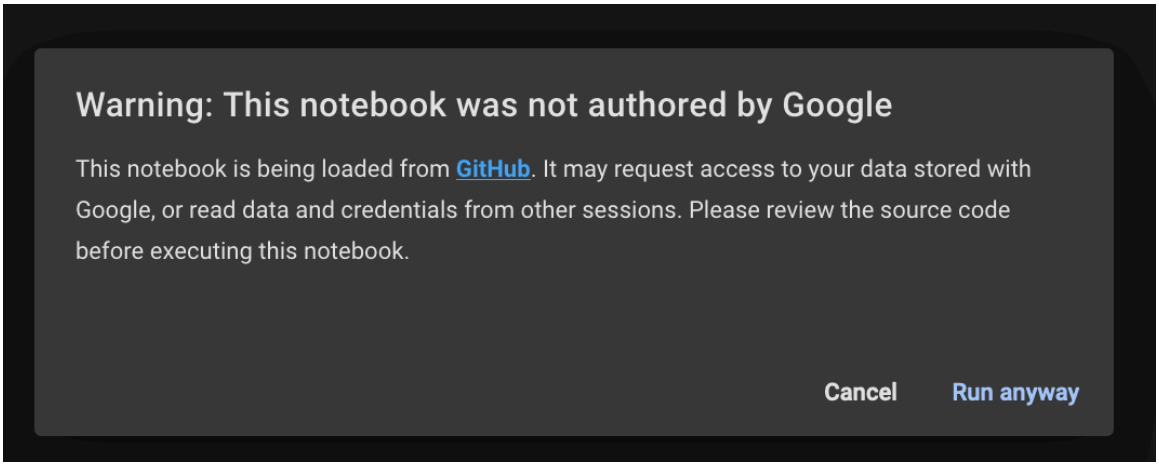
8. Select **Connect to a hosted runtime** on the **Connect** menu.

9. A green checkmark indicates that you have successfully connected to the hosted runtime.

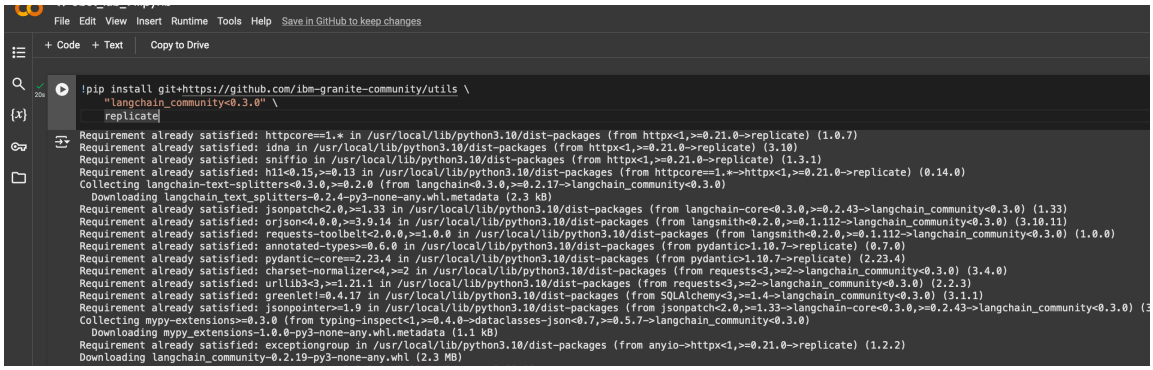
10. In the first cell of the notebook, select the **Play** button to install the required libraries from the Granite community.

Note: You'll need to run the cells sequentially beginning from the first cell in the notebook.

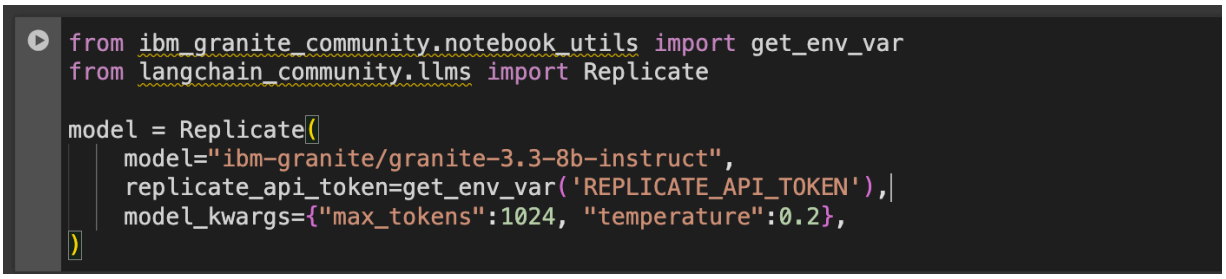
11. Select **Run anyway** to proceed loading the required libraries.


Warning: This notebook was not authored by Google
This notebook is being loaded from GitHub. It may request access to your data stored with Google, or read data and credentials from other sessions. Please review the source code before executing this notebook.
Cancel
Run anyway

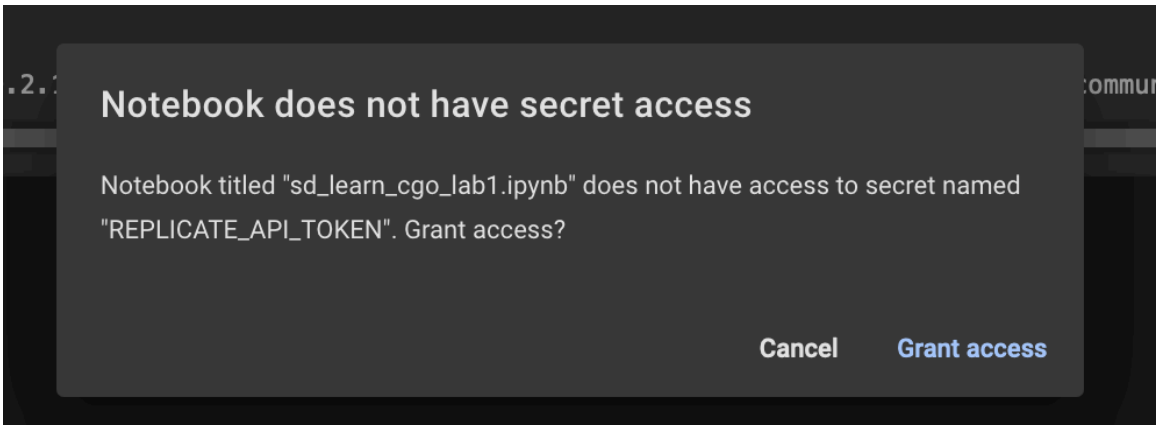
12. A green checkmark appears beside the **Play** button to indicate that the required libraries are successfully installed.



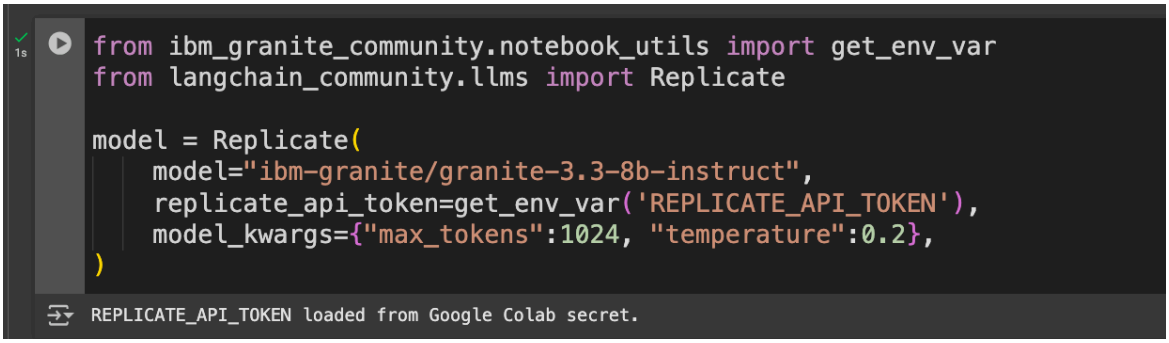
13. In the second cell of the notebook, select the **Play** button to initialize the IBM Granite model using Replicate for code generation.



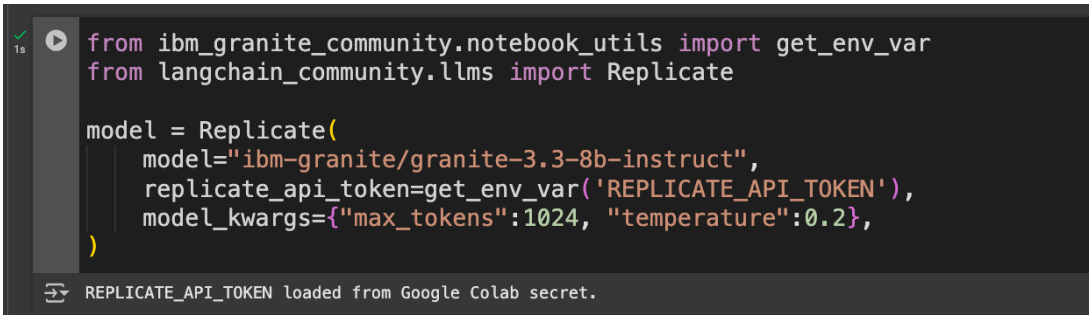
14. Select **Grant access** to proceed loading the model from Replicate.



15. The following message displays at the bottom of the cell: REPLICATE_API_TOKEN loaded from Google Colab secret.



16. A green checkmark besides the **Play** button indicates that the IBM Granite model is initialized using Replicate. You are now ready to prompt the model to generate code for the given scenario.



Step 5: Generate code using the IBM Granite model

Overview

In this step, you'll prompt the IBM Granite model to generate code for the landing page of the Reader's Verse website. You'll define a zero-shot prompt for the model and run the prompt to create the UI components for the landing page of the client's website.

Instructions

1. Define and execute a function that creates a zero-shot prompt for the model. A zero-shot prompt describes a task which a model needs to perform without providing examples to guide the model's output.

   In the third cell of the notebook, select the **Play** button to define the **zeroshot_prompt** function containing a set of instructions that informs the model what output to generate. In this scenario, the instructions inform the model to generate code for the online bookstore based on the given parameters, which are context, question, and book titles.

```
[1] !pip install git+https://github.com/ibm-granite-community/utils \
        "langchain_community<0.3.0" \
        replicate
```

```
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)
```

```
REPLICATE_API_TOKEN loaded from Google Colab secret.
```

```
def zeroshot_prompt(context, question, book_titles):
    """
    Creates a zero-shot prompt for the model, where the model acts as a
    seasoned programmer.

    Parameters:
    - context: str, contextual information for the prompt
    - question: str, specific question or task for the model to perform
    - book_titles: list, list of book titles to include in the prompt
    Returns:
    - str, the formatted prompt
    """

    titles = ", ".join(book_titles)
    prompt = f"""
    You are an experienced programmer with 15 years of experience writing
    full-stack applications.
    Your task is to generate code for an application using the context provided
    in the task: {context}.
    Begin by creating components based on the user question: {question}.
```

Note: You won't see any output when you run this cell in the notebook.

2. Next, create a function to generate a response from the model using the zero-shot prompt. You'll send the instructions defined in the zero-shot prompt to the model and prepare to get a response from the model.

   In the fourth cell of the notebook, select the **Play** button to define the **get_answer_using_zeroshot** function.

```
def get_answer_using_zeroshot(context, question, book_titles):
    """
    Generates the response from the model based on a zero-shot prompt.

    Parameters:
    - context: str, contextual information for the prompt
    - question: str, specific question for the model to answer
    - book_titles: list, list of book titles to include in the prompt

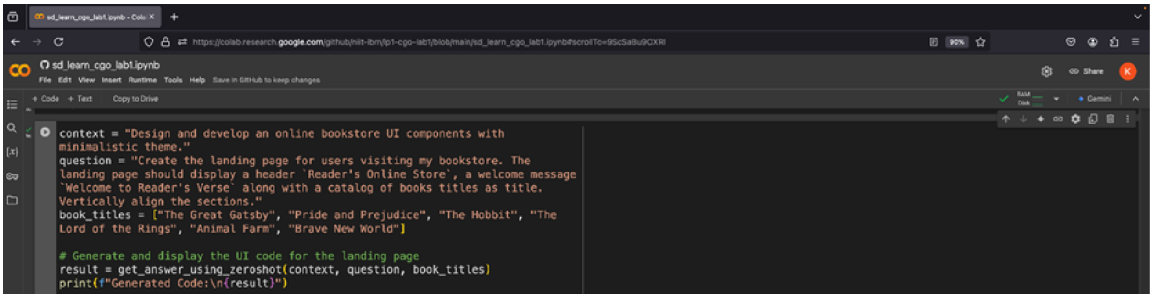
    Returns:
    - str, the generated result from the model
    """
    prompt = zeroshot_prompt(context, question, book_titles)
    result = model.invoke(prompt)

    return result
```

   Note: You won't see any output when you run this cell in the notebook.

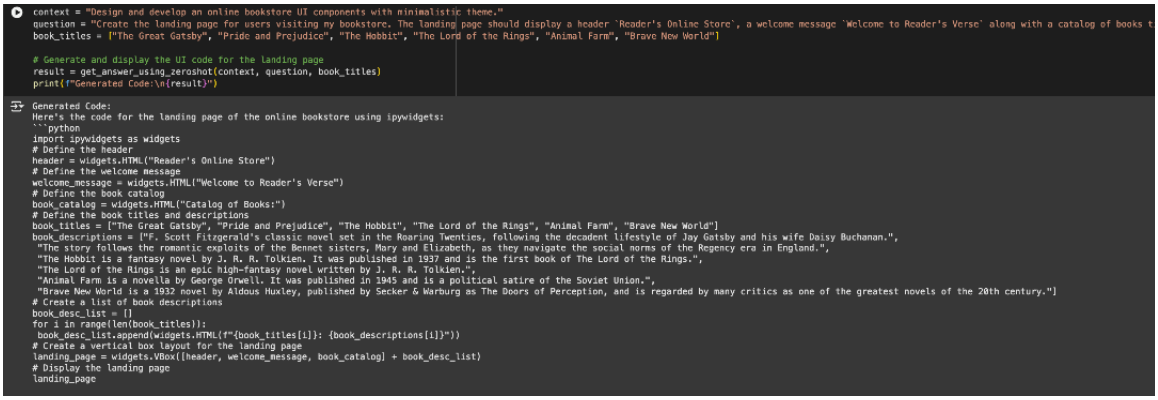
3. Next, you'll provide the required values for parameters defined in the zero-shot prompt, and then call the get_answer_using_zeroshot function to generate the code output. In this scenario, the values provided are as follows:
   - **Context:** Prompts the model to design the UI with a minimalistic theme
   - **Question:** Defines the attributes of the landing page of the website such as header and welcome message
   - **Titles:** Defines the list of book titles to display on the landing page

   In the fifth cell of the notebook, select the **Play** button to run the **get_answer_using_zeroshot** function. This generates and displays the Python code for the landing page UI of the online bookstore following the code cell.



4. The generated output includes the Python code for creating UI elements such as a landing page and a catalog of books, using a native UI framework of Jupyter Notebook. This code will create the bookstore's landing page UI, including a title and a catalog of book titles.

   Note: When you generate the code, it might be slightly different from the following code output. These minor differences are normal and will not affect the overall functionality or outcome of the lab.

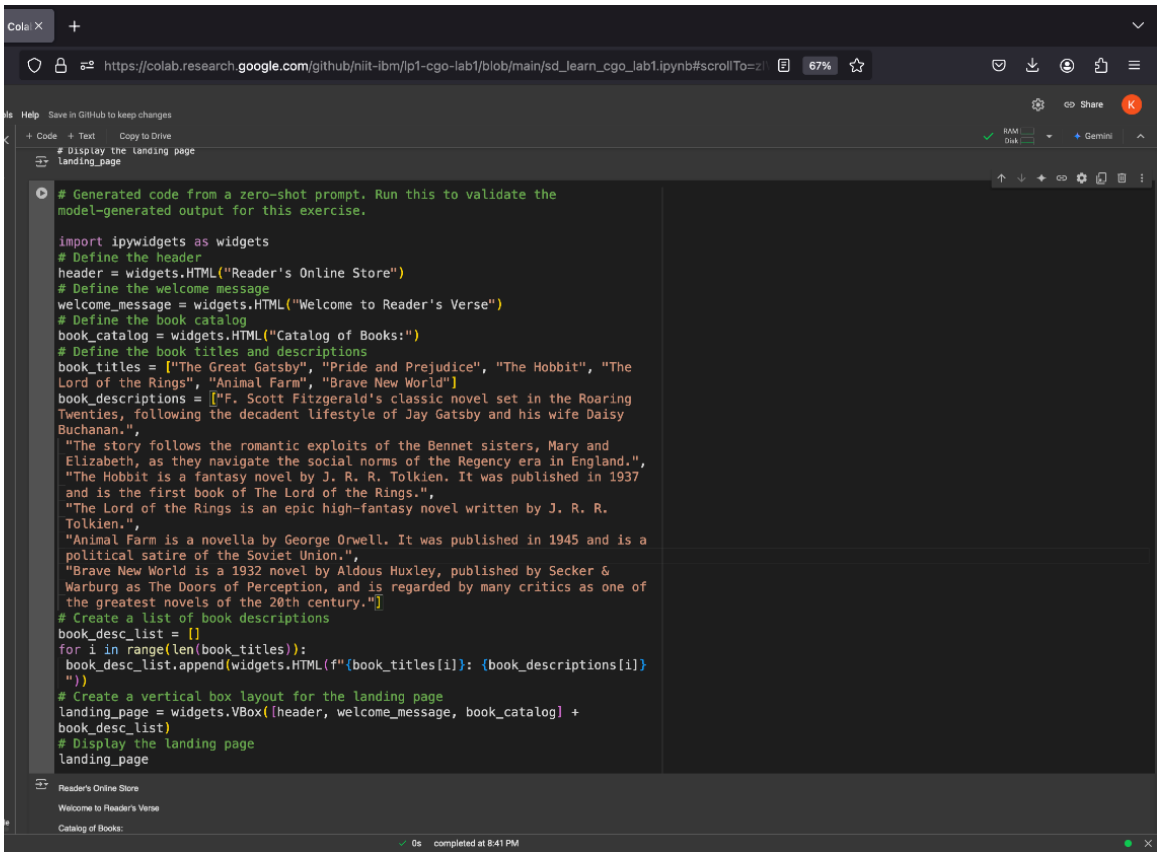


```
context = "Design and develop an online bookstore UI components with minimalistic theme."
question = "Create the landing page for users visiting my bookstore. The landing page should display a header `Reader's Online Store`, a welcome message `Welcome to Reader's Verse` along with a catalog of books t
book_titles = ["The Great Gatsby", "Pride and Prejudice", "The Hobbit", "The Lord of the Rings", "Animal Farm", "Brave New World"]

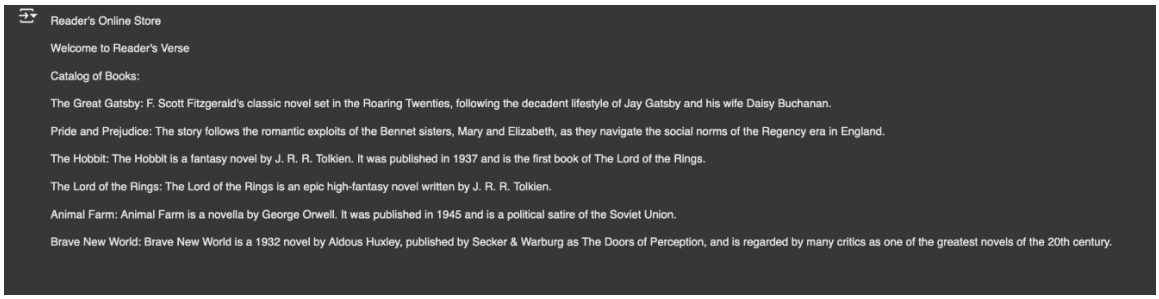
# Generate and display the UI code for the landing page
result = get_answer_using_zeroshot(context, question, book_titles)
print(f"Generated Code:\n{result}")
```

```
Generated Code:
Here's the code for the landing page of the online bookstore using ipywidgets:
```python
import ipywidgets as widgets
# Define the header
header = widgets.HTML("Reader's Online Store")
# Define the welcome message
welcome_message = widgets.HTML("Welcome to Reader's Verse")
# Define the book catalog
book_catalog = widgets.HTML("Catalog of Books:")
# Define the book titles and descriptions
book_titles = ["The Great Gatsby", "Pride and Prejudice", "The Hobbit", "The Lord of the Rings", "Animal Farm", "Brave New World"]
book_descriptions = ["F. Scott Fitzgerald's classic novel set in the Roaring Twenties, following the decadent lifestyle of Jay Gatsby and his wife Daisy Buchanan.",
 "The story follows the romantic exploits of the Bennet sisters, Mary and Elizabeth, as they navigate the social norms of the Regency era in England.",
 "The Hobbit is a fantasy novel by J. R. R. Tolkien. It was published in 1937 and is the first book of The Lord of the Rings.",
 "The Lord of the Rings is an epic high-fantasy novel written by J. R. R. Tolkien.",
 "Animal Farm is a novella by George Orwell. It was published in 1945 and is a political satire of the Soviet Union.",
 "Brave New World is a 1932 novel by Aldous Huxley, published by Secker & Warburg as The Doors of Perception, and is regarded by many critics as one of the greatest novels of the 20th century."]
# Create a list of book descriptions
book_desc_list = []
for i in range(len(book_titles)):
 book_desc_list.append(widgets.HTML(f"{book_titles[i]}: {book_descriptions[i]}"))
# Create a vertical box layout for the landing page
landing_page = widgets.VBox([header, welcome_message, book_catalog] + book_desc_list)
# Display the landing page
landing_page
```
```

5. For ease of execution, the output code is pre-populated in a new cell of the Jupyter notebook. Select the **Play** button in the sixth cell of the notebook to validate the Python code output generated by the IBM Granite model for the given scenario.



6. The output should display the bookstore's landing page UI following the code cell, featuring the following components:
    - A title: "Welcome to Reader's Verse!"
    - A catalog grid displaying book titles, authors, and descriptions



Conclusion

You have used an IBM Granite model to generate code for creating the landing page of a website for Reader's Verse, an online bookstore. The client is delighted with how fast you developed a working prototype of the UI for the website's welcome page. You have also successfully defined a zero-shot prompt to generate Python code output from the IBM Granite model and validated the AI-generated code to ensure that it correctly renders the expected output.