LAB

Evaluate the performance of an AI agent using MLFlow

IBM **SkillsBuild**

Contents

Introduction 2
Software requirements 2
Objective 2
Lab steps 2
Estimated duration to complete 2
Scenario 2
Background information 2
Step 1: Create a GitHub account 3
Overview 3
Instructions 3
Step 2: Create a Replicate account 9
Overview 9
Instructions 9
Step 3: Sign up for Google Colab 15
Overview 15
Instructions 15
Step 4: Initialize the AI agent and tools 21
Overview 21
Instructions 21
Step 5: Trajectory analysis and final response evaluation 32
Overview 32
Instructions 32
Conclusion 35

Introduction

In this lab, you'll assume the role of an AI ~~Specialist~~ specialist to evaluate a pilot version of an AI agent using MLFlow.

Software requirements

To complete this lab, you'll need access to a Replicate account, which allows you to use artificial intelligence (AI) models to perform tasks. You'll also need a Replicate ~~Application~~ application ~~Programming~~ programming ~~Interface~~ interface (API) token, which acts as a secure key to connect your Colab environment so that the lab can run smoothly.

Basic familiarity with Python will enable you to better follow how the agent works and its tool usage.

Objective

After completing this lab, you should be able to:

- Evaluate the performance of an AI agent~~.~~

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:** Initialize the AI agent and tools~~.~~
- **Step 5:** Evaluate the trajectory and final response of ~~an~~ the AI agent

Estimated duration to complete

30 minutes

Scenario

Background information

TechMart, a fast-growing e-commerce platform, is transforming its shopping experience with an AI agent built on IBM Granite. The agent replaces static catalogs, offering real-time recommendations, answering questions, and personalizing the customer's journey.

You are part of TechMart's newly formed AI ~~Evaluation~~ evaluation team, serving as the AI ~~Specialist~~ specialist for this project. Your role is to assess the agent's performance by evaluating product recommendations, response accuracy, and the overall user experience.

Challenge

TechMart users often leave the site when the AI agent fails to provide relevant product recommendations or assist effectively with catalog navigation. Additionally, missing or inconsistent details about products, pricing, or stock levels can erode the ~~user's~~ user's trust.

Solutions

To support this, your team uses MLFlow to track, evaluate, and visualize the agent's performance. In this lab, ~~you'll~~ you'll determine whether the AI agent is ready for full-scale deployment.

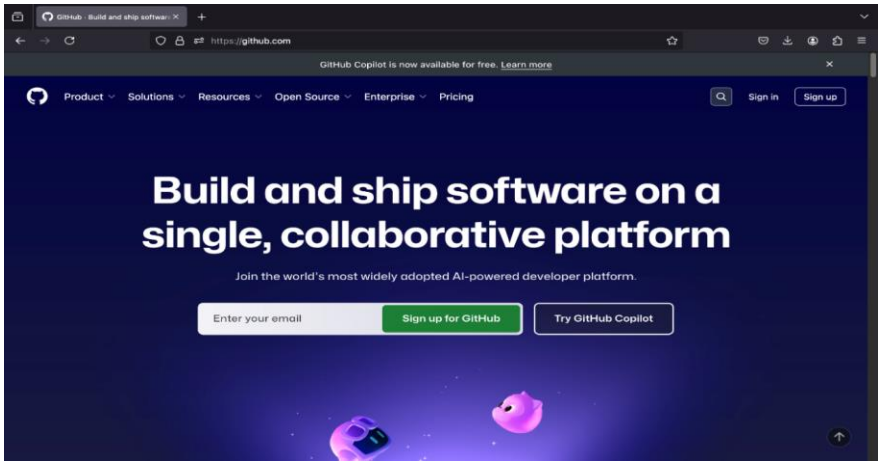
Step 1: Create a GitHub account

Overview

In this step, you'll set up a GitHub account. GitHub is a platform that helps developers store, manage, and share code, while also supporting collaboration through tools such as version control, bug tracking, and task management. Setting up a GitHub account ensures access to the Replicate platform, which is ~~needed~~ required to complete the lab efficiently.

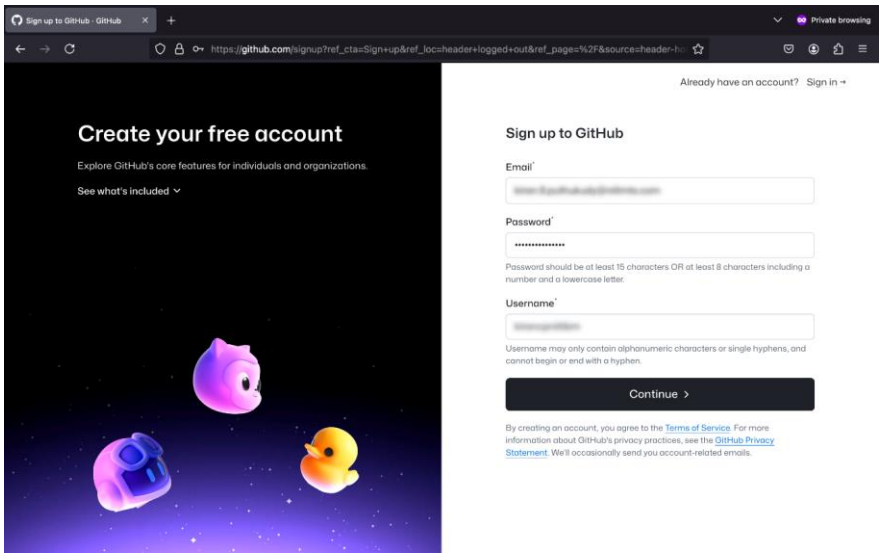
Instructions

1. To create a GitHub account, go to the GitHub website and select the **Sign up for GitHub** button.



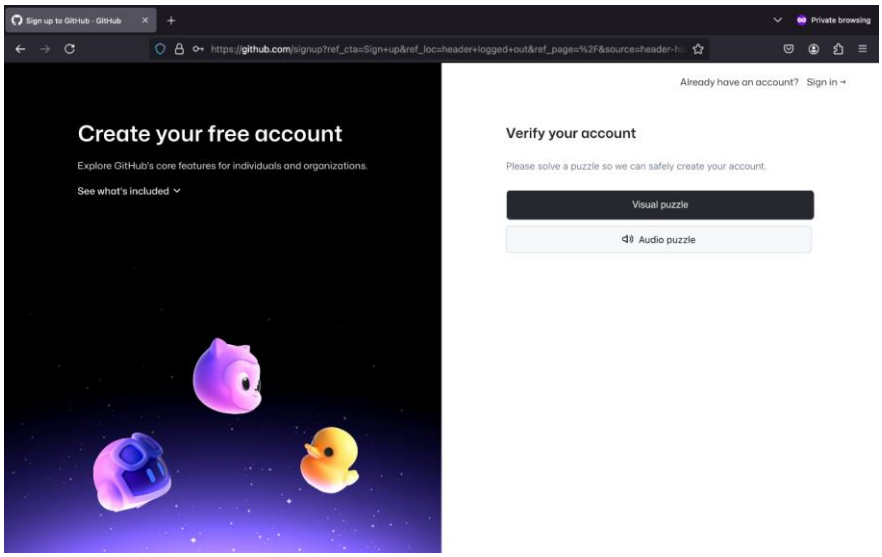
Alt text: An image showing the ~~homepage~~ home page of the GitHub website with the **Sign up for GitHub** button

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



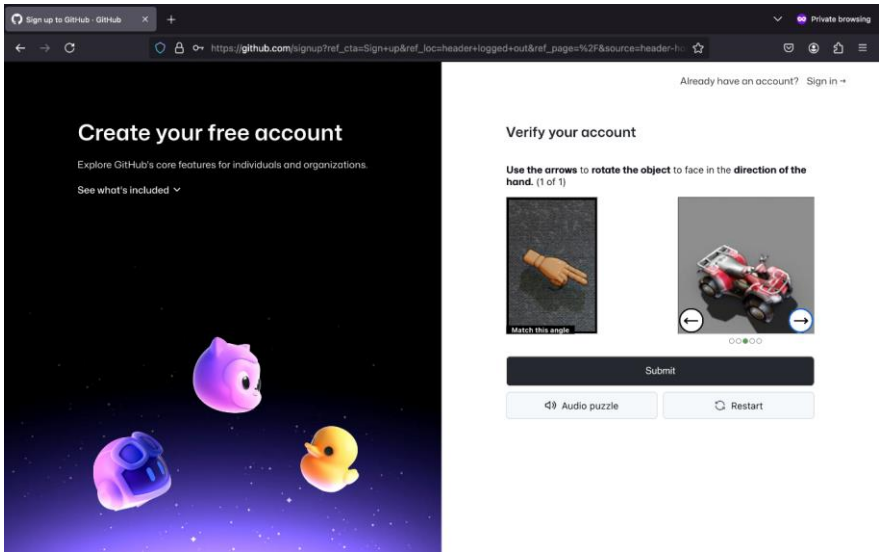
Alt text: An image showing the "GitHub sign-up" page with the **Email**, **Password**, and **Username** fields

3. To verify your account, select the **Visual puzzle** option.



Alt text: An image showing the "GitHub sign-up" page and the visual and audio puzzle verification step

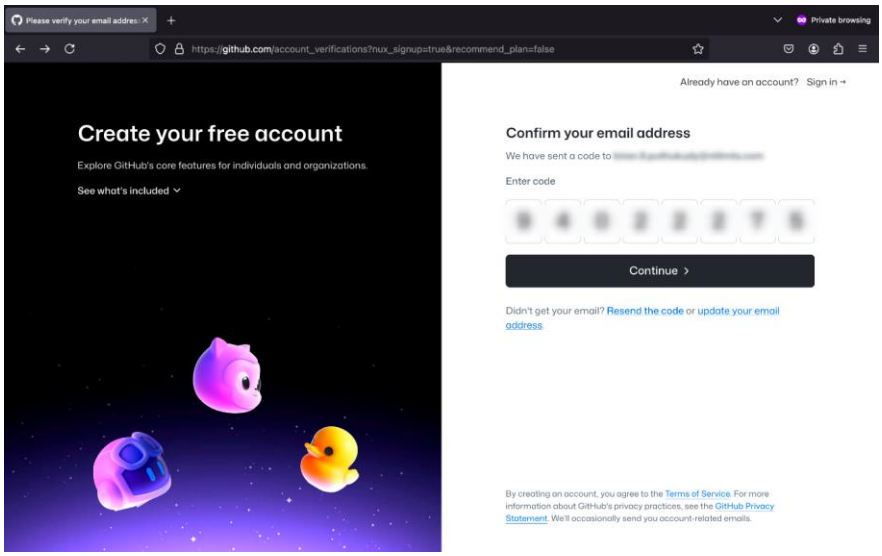
4. Next, solve the visual puzzle and select **Submit**.



Alt text: An image showing the “GitHub sign-up” page with visual and audio puzzles and the **Submit** button

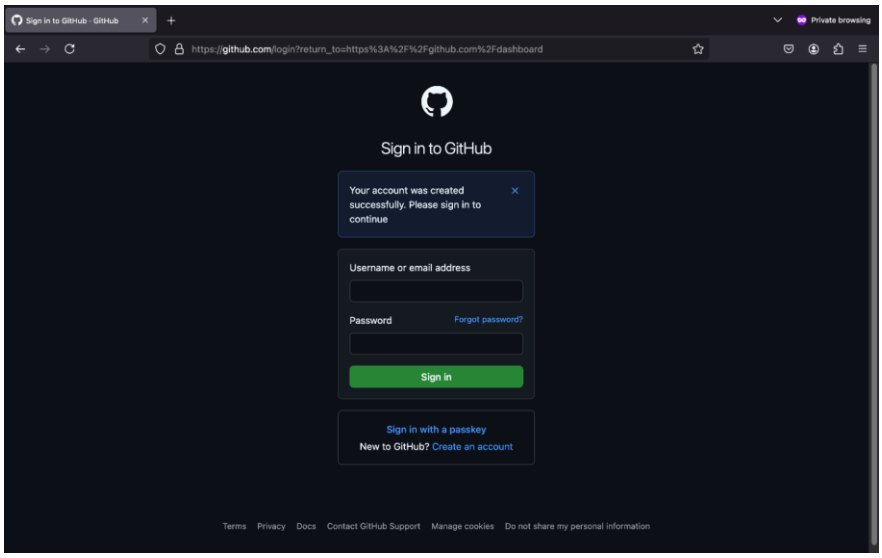
Commented [DA1]: Shouldn’t it be “visual and audio puzzles”

5. To confirm your email, enter the confirmation code sent to your registered email in the **Enter code** field and select **Continue**.



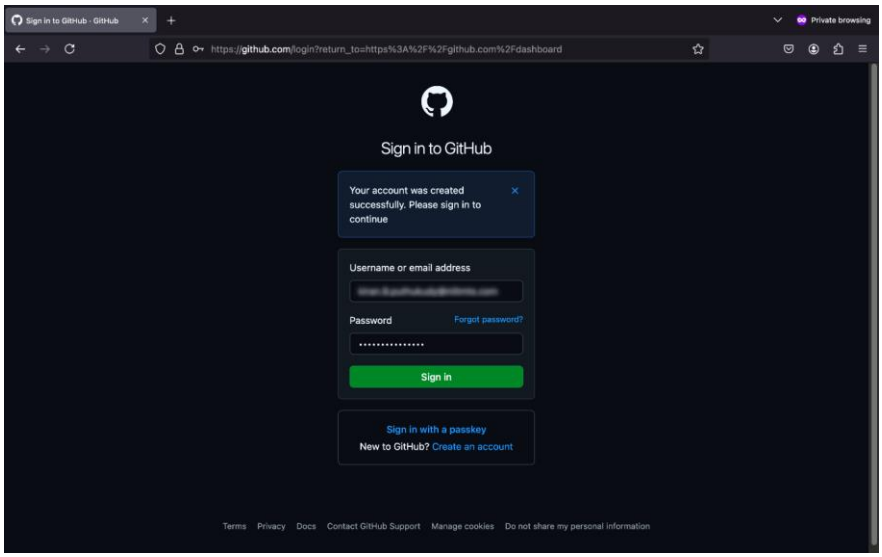
Alt text: An image showing the “GitHub sign-up" page with the email confirmation step

6. You'll see a confirmation message after your GitHub account has been successfully created.



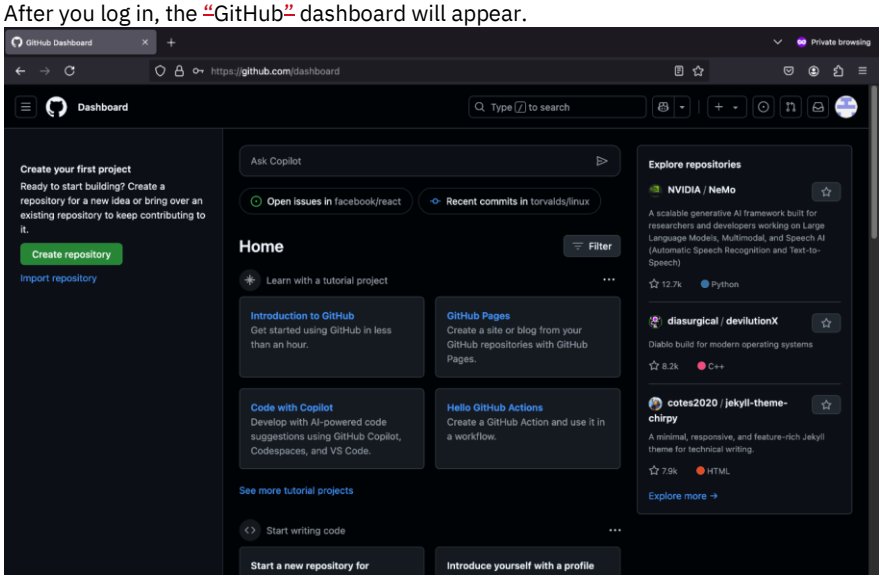
Alt text: An image showing the "GitHub sign-in" page with a message indicating that the account has been successfully created

7. To sign in to your account, enter your credentials in the **Username or email address** and **Password** fields, and then select **Sign in**.



Alt text: An image showing the "GitHub sign-in" page with the **Username or email address** and **Password** fields, and ~~a~~ the **Sign in** button

8. After you log in, the "GitHub" dashboard will appear.



Formatted: Font: Bold

Formatted: Font: Bold

Alt text: An image showing the ~~"~~GitHub~~"~~ dashboard with options to create a project, explore repositories, access learning resources and interact with various ~~interactive~~ elements

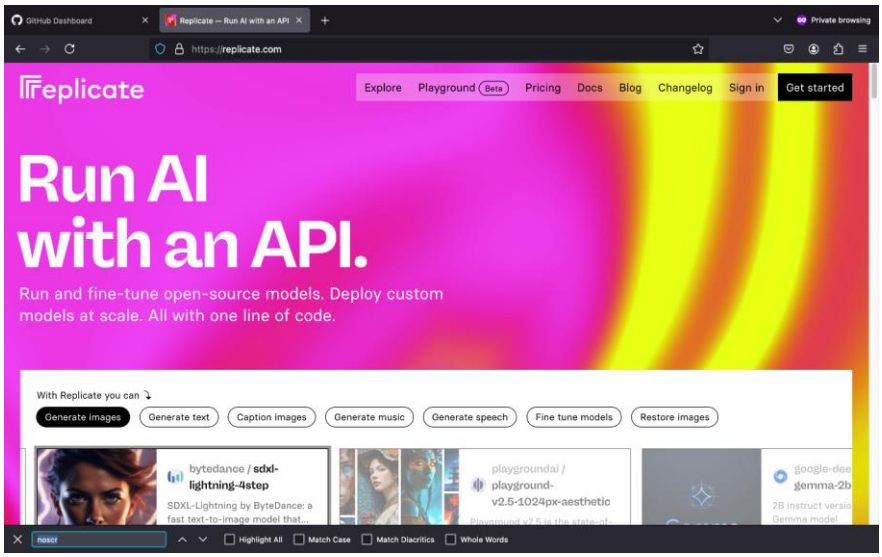
Step 2: Create a Replicate account

Overview

In this step, you'll use your GitHub account to register for a Replicate account. Replicate is a cloud-based platform that lets you use AI models such as IBM Granite without ~~needing~~ requiring advanced hardware. As part of this step, you'll create a Replicate token. A token is a secure access key that allows the lab environment to authenticate with Replicate and run models from your account in Google Colab.

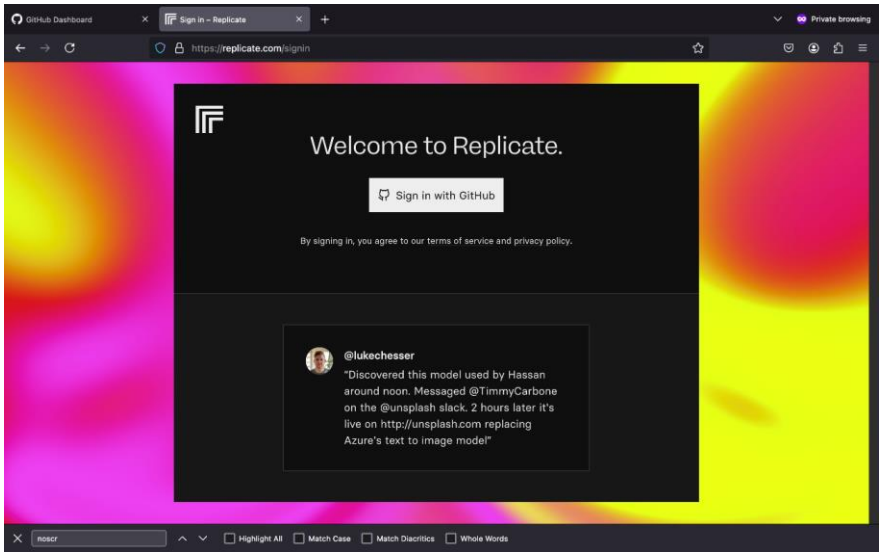
Instructions

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



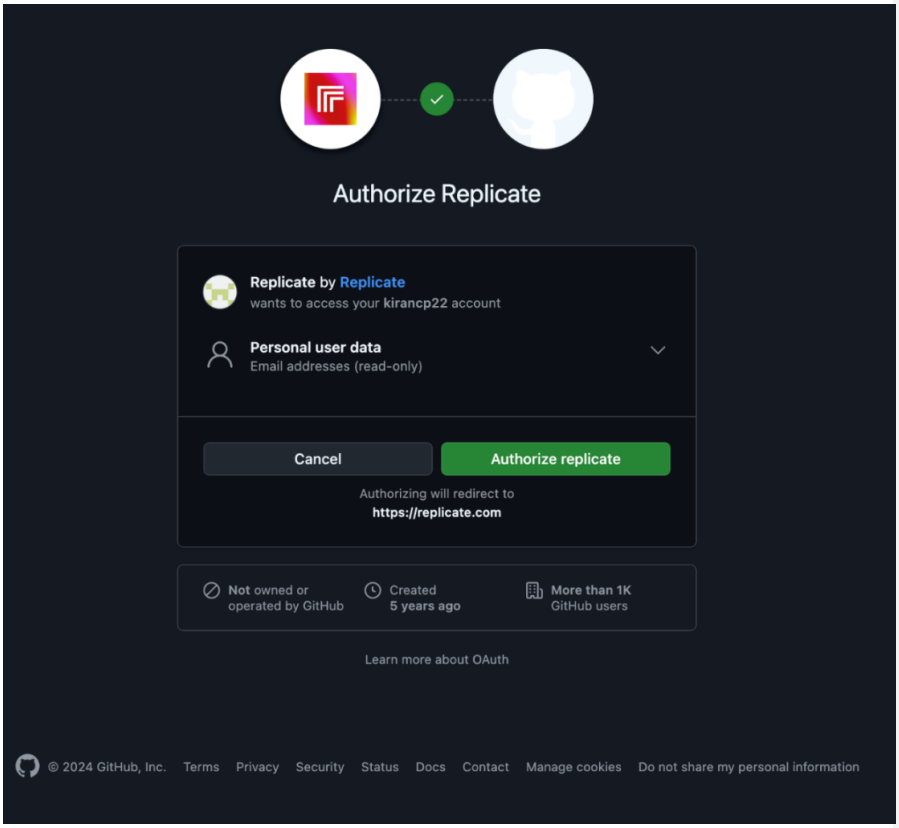
Alt text: An image showing the Replicate website with ~~a~~ the **Get started** button

10. Select the **Sign in with GitHub** option on the "Welcome to Replicate" page.



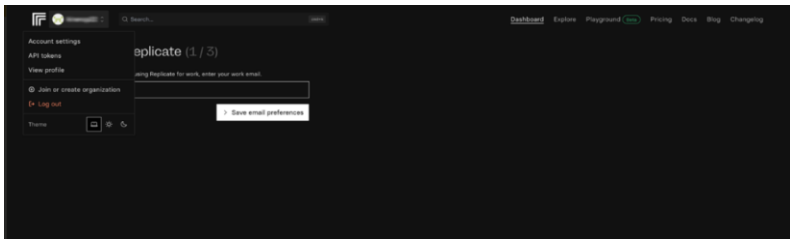
Alt text: An image showing the "Welcome to Replicate" page with the **Sign in with GitHub** option

11. Select **Authorize replicate** to continue.



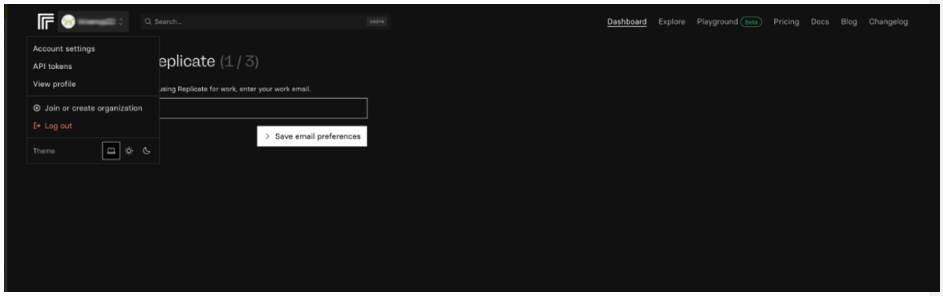
Alt text: An image showing the authorization screen for Replicate, requesting access to the user's account

12. After your Replicate account is created, you'll be taken to the ~~"~~Replicate~~"~~ dashboard. To create a Replicate token, select the **Account settings** option ~~in~~ from the navigation bar.



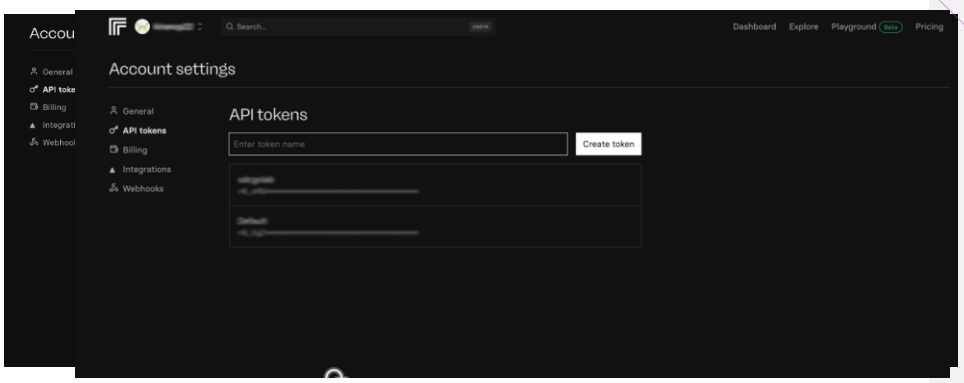
Alt text: An image showing the ~~"~~Replicate~~"~~ dashboard with a drop-down list containing options such as **API tokens**, **View profile,** and **Log out**, and an email prompt with ~~a~~ the **Save email preferences** button

13. Select **API tokens** from the **Account settings** menu.



Alt text: An image showing the ~~"~~Replicate~~"~~ dashboard with a drop-down list containing options such as~~,~~ **API tokens**, **View profile**, and **Log out**, and an email prompt with ~~a~~ the **Save email preferences** button

15. Enter a After your API token is created, it should appear on the “Account settings” page.

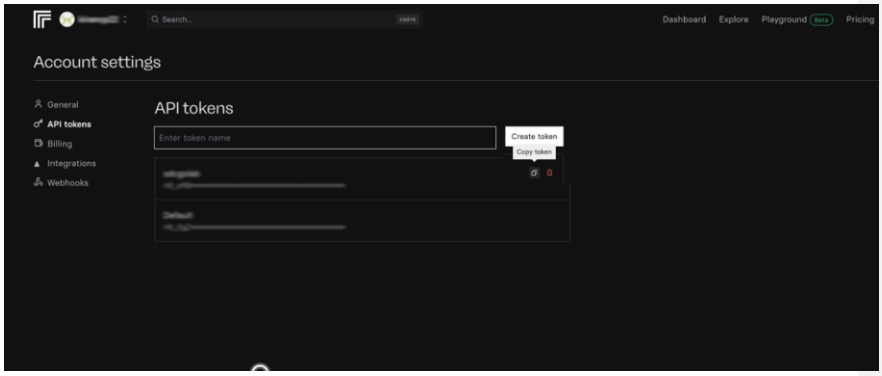


Alt text: An image showing the “Account settings” page displaying the **API tokens** field and a **Create token** button

~~**Alt text:** An image showing the “Account settings” page with the created API tokens~~

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

Note: Save the Replicate token because you’ll need it to authenticate with the Replicate API when running code in the Google Colab environment later in this lab.



Alt text: An image showing the “Account settings” page with the **Create token** field and the **Copy token** option

Formatted: Highlight

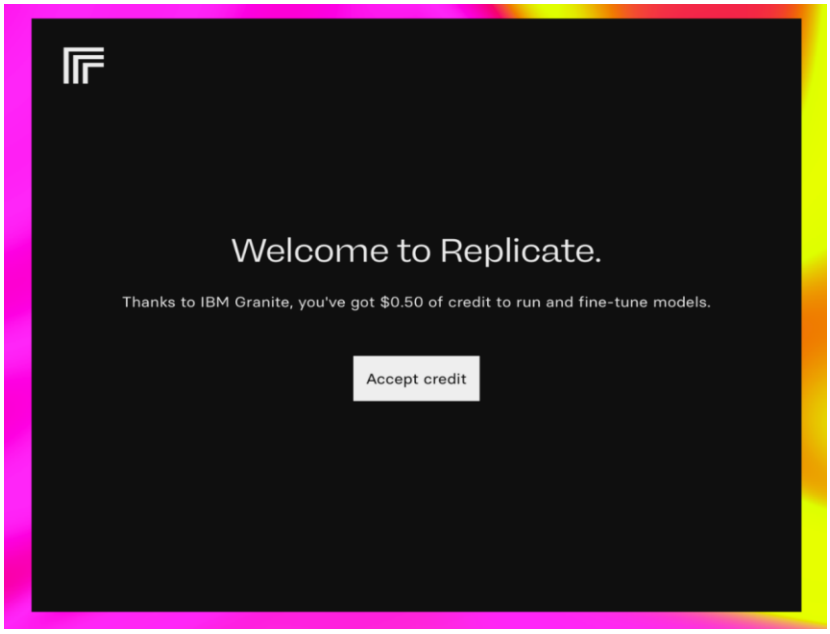
Formatted: Highlight

Formatted: Highlight

Commented [DA3]: Also, it should be “the” Create token button

Commented [DA4]: Delete the background text and correct the alignment.

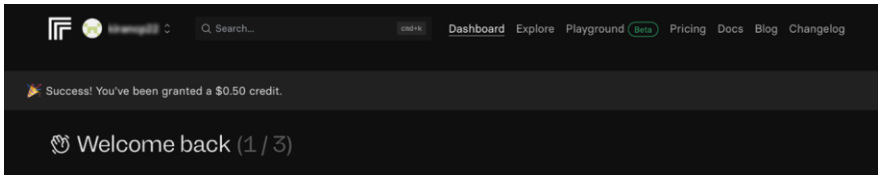
17. To run the lab without interruptions, you'll require some Replicate credits. Go to the Replicate invite link to claim your free $0.50 credit. Then, select **Accept credit** to claim the amount.



Alt text: An image showing the **Welcome to Replicate** screen, ~~and~~ with an option to accept a $0.50 credit

Commented [DA5]: Check if it should be in bold. Otherwise, unbold and don't highlight, as it is already in title casing.

18. A confirmation message will appear ~~confirming~~ indicating that a $0.50 credit has been added to your Replicate account.



Alt text: An image showing the "Welcome back" message and ~~a~~ the confirmation message about the $0.50 credit amount

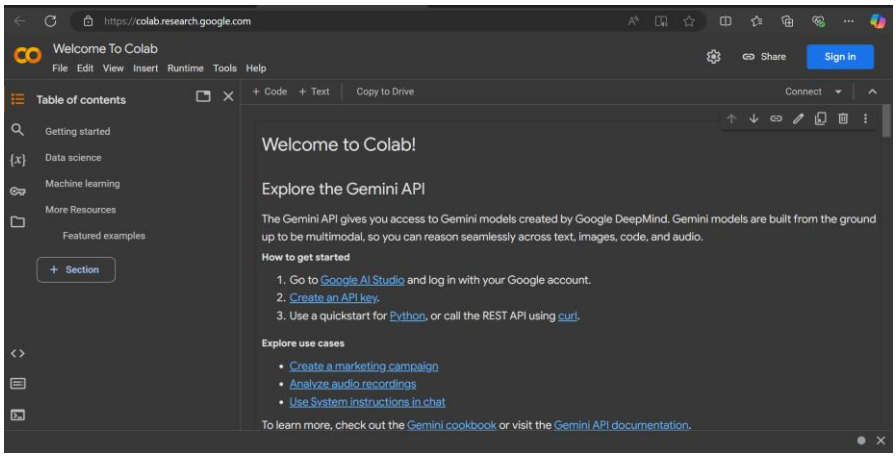
Step 3: Sign up for Google Colab

Overview

In this step, you'll 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. A Google Colab account will allow you to install and use the tools needed to complete this lab.

Instructions

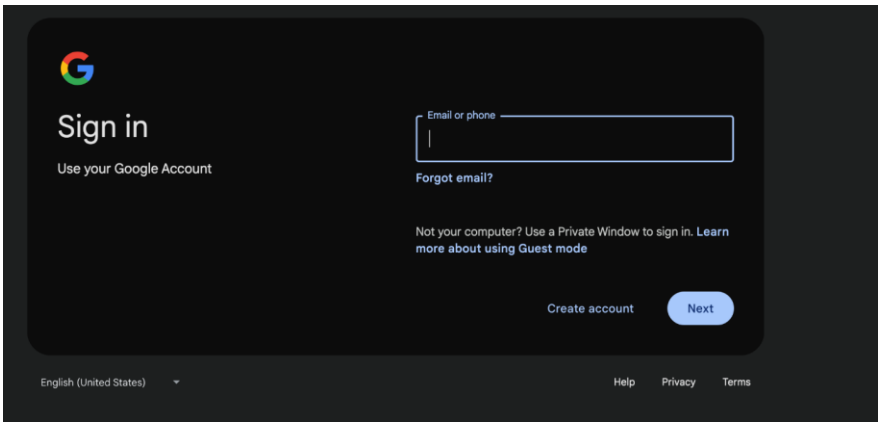
19. To sign up, go to the Google Colab website and select **Sign in**.



Alt text: An image showing the **Welcome to Colab** screen, the **Sign in** button, and ~~the~~ instructions on how to get started

Commented [DA6]: Check highlighting. Otherwise, unbold.

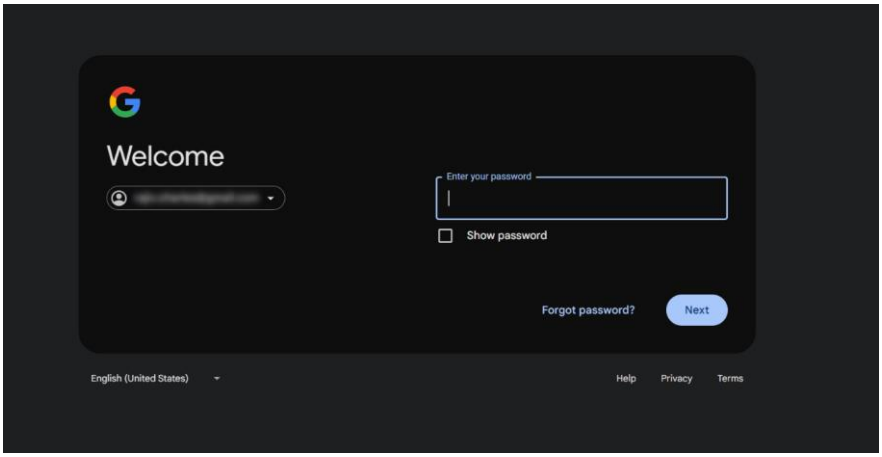
20. Enter your email or phone number in the **Email or phone** field, and then select **Next**.



Alt text: An image showing ~~a~~ the Google Sign-~~-~~in page with the **Email or phone** ~~filed~~ field, ~~and with~~ the **Create account** option, and the **Next** button

Formatted: Font: Not Bold

21. Enter your password in the **Enter your password** field, and then select **Next**.

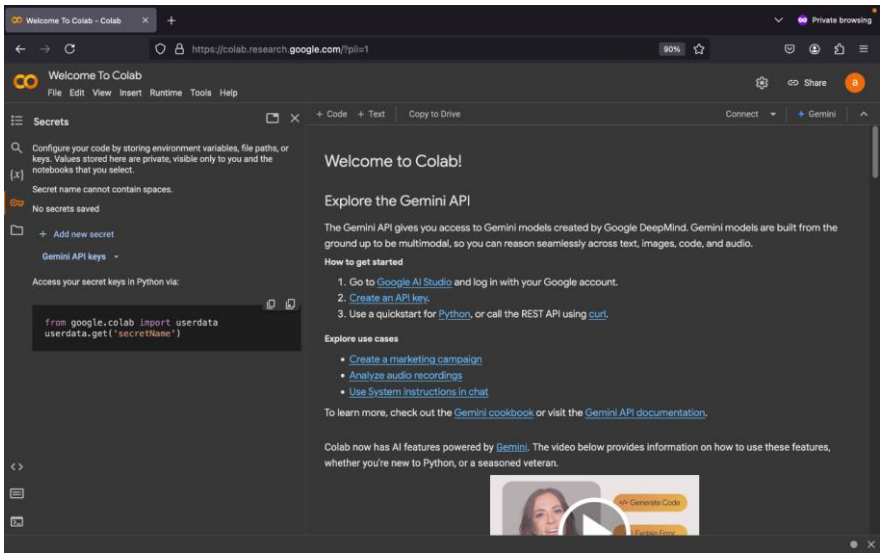


Alt text: An image showing the **Welcome** screen with the **Enter your password** field, ~~and~~ the **Forgot password** option, and the **Next** button

Commented [DA7]: Should screen names be in bold? Please check. It is already in title casing, so no need to use quotes either

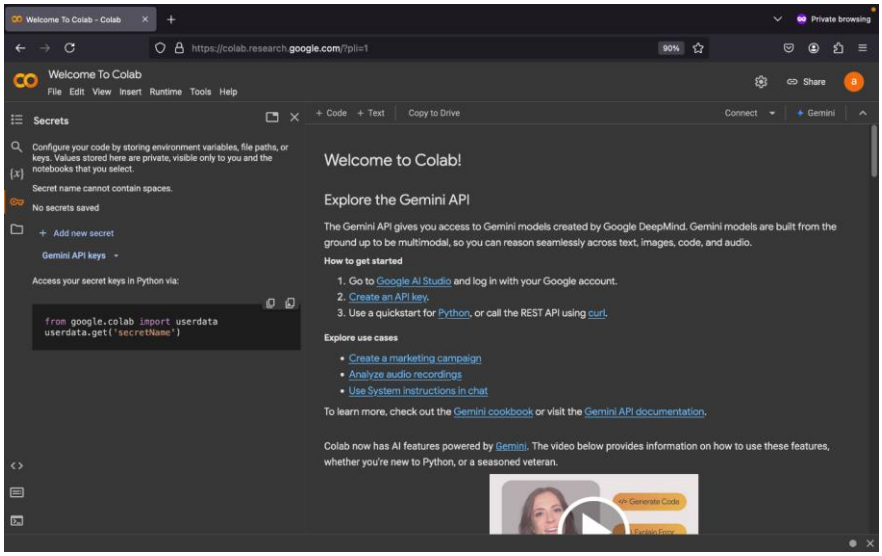
22. Next, to store your Replicate API token in the Google Colab **Secrets** tab, select the key ~~**Key**~~ icon ~~on~~ from the sidebar menu ~~of~~ on the ~~"~~Welcome to Colab~~"~~ page.

Formatted: Font: Not Bold



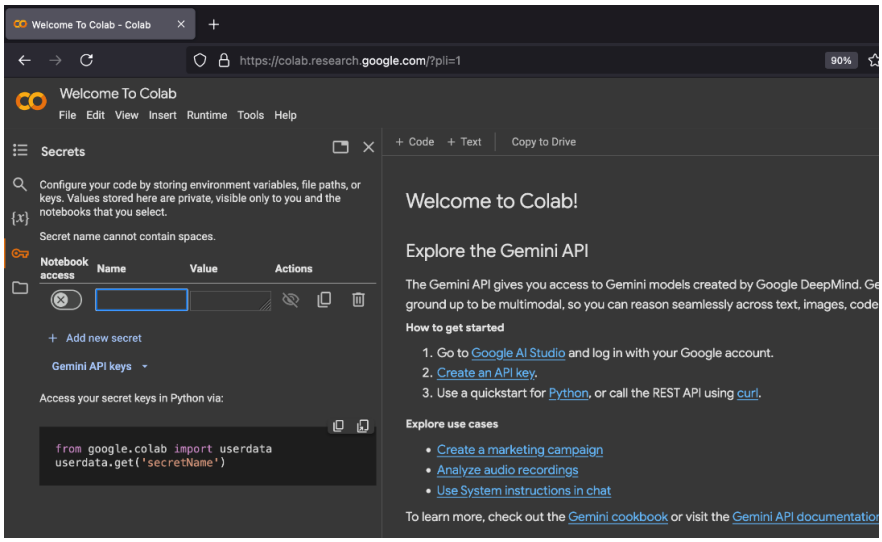
Alt text: An image showing the "Welcome to ~~colab~~Colab" page with the **Secrets** tab

23. Select **Add new secret**.



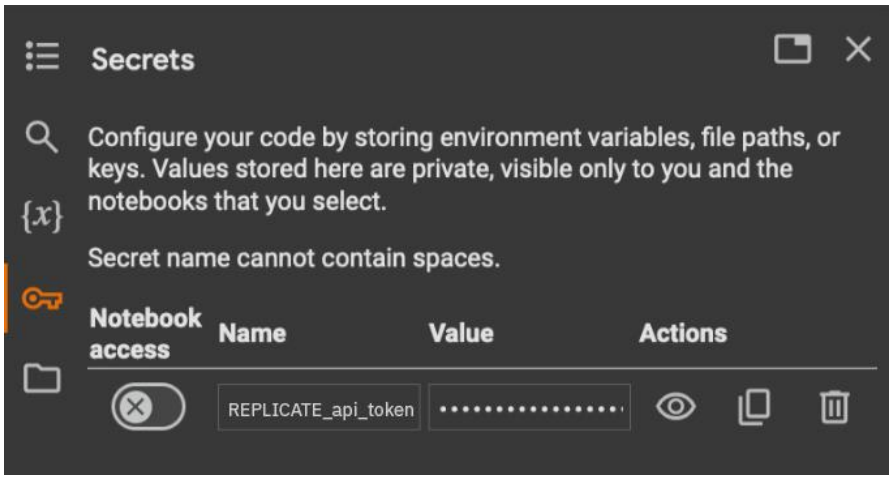
Alt text: An image showing the "Welcome to ~~colab~~Colab" page with the **Secrets** tab and the **Add new secret** option

24. Type **REPLICATE_api_token** in the **Name** field.



Alt text: An image showing the **Secrets** tab of Google Colab with the **Name** and **Value** fields

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

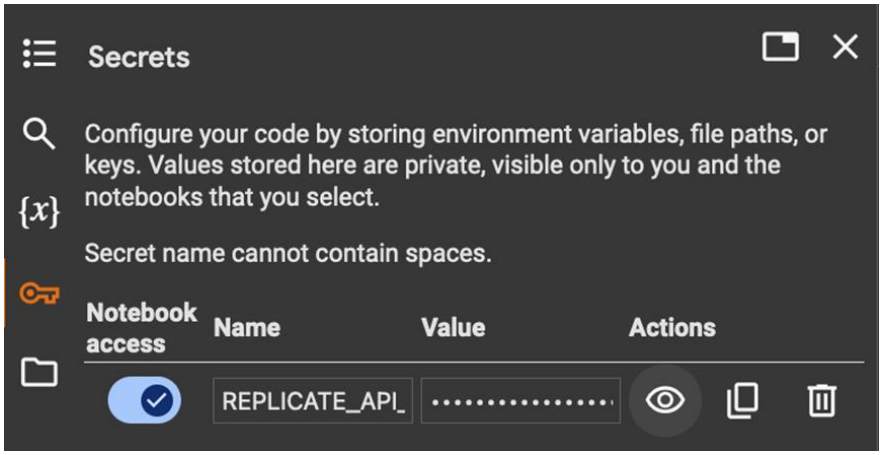


Alt text: An image showing the **Secrets** tab displaying ~~labelled~~ the **Name** and **Value** fields

26. Select the toggle~~toggle~~ button to enable **Notebook access**. Next, select the close ~~close~~ icon to exit the configuration.

Commented [DA8]: Please check the highlihgting

Formatted: Font: Not Bold



Alt text: An image showing the **Secrets** section displaying ~~labelled~~ the **Name** and **Value** fields and ~~an~~the enabled toggle button

Step 4: Initialize the AI agent and tools

Overview

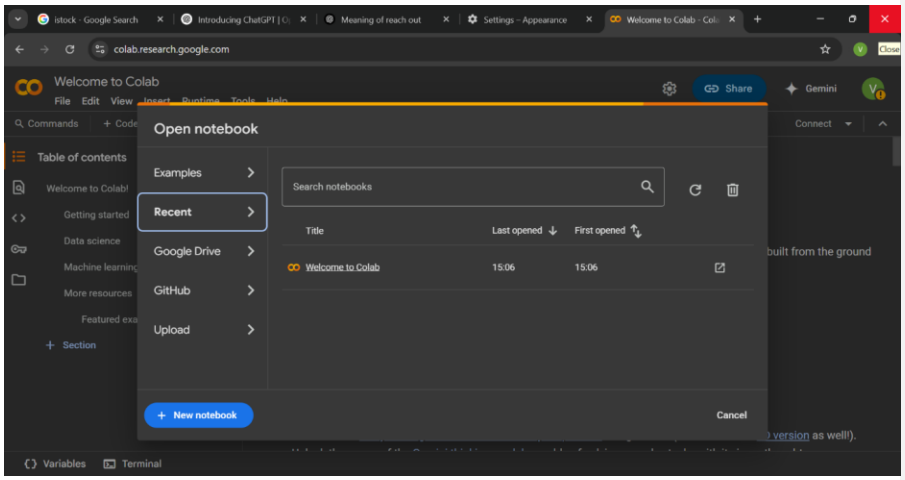
In this step, you'll initialize ~~the~~ TechMart's AI agent and the tools it uses by loading a Jupyter notebook.

In the context of an AI agent, **tools** are functions that help an AI ~~agents~~ agent perform specific tasks such as searching for information, checking product stock, or looking up data. The user provides prompts that define which tools the agent can access by including them in the agent's setup. When responding, the agent sends a query to the selected tool, receives the result, and uses that information to generate its reply.

A **Jupyter notebook** is a shareable document that combines computer code, plain language descriptions, data, and visualizations.

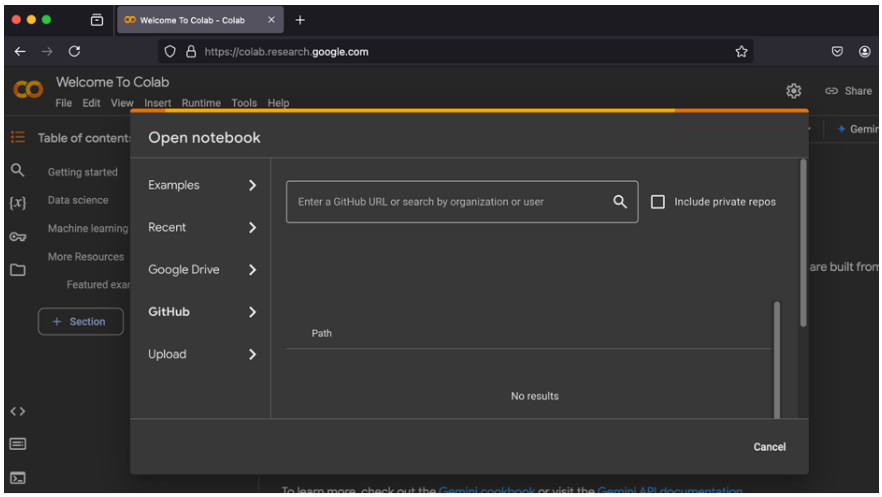
Instructions

27. Select Google Colab to open the Google Colab welcome screen with the "Open notebook" dialog box where you can open your Jupyter notebook.



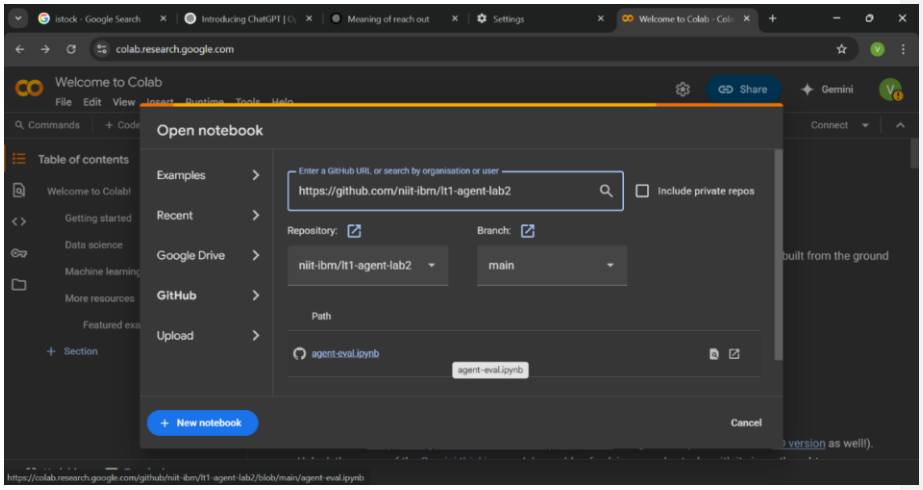
Alt text: An image showing the Google Colab welcome screen with the "Open notebook" dialog box

28. In the “Open notebook” dialog box, select the **GitHub** option. To find and load the Jupyter notebook directly in Google Colab, type the uniform resource locator (URL) https://github.com/niit-ibm/lt1-agent-lab2 in the search~~search~~ field. Then, press **Enter**.



Alt text: An image showing the “Open notebook” dialog box with GitHub as the selected option

29. After you enter the link in the search~~search~~ field, the **Repository** and **Branch** fields will be automatically populated. Now, to open the notebook, select **agent-eval.ipynb**.



Commented [DA9]: Can we write this as “tab”?

Commented [DA10]: Since “search” is not the name of the field, no need to make it bold.

Formatted: Font: Not Bold

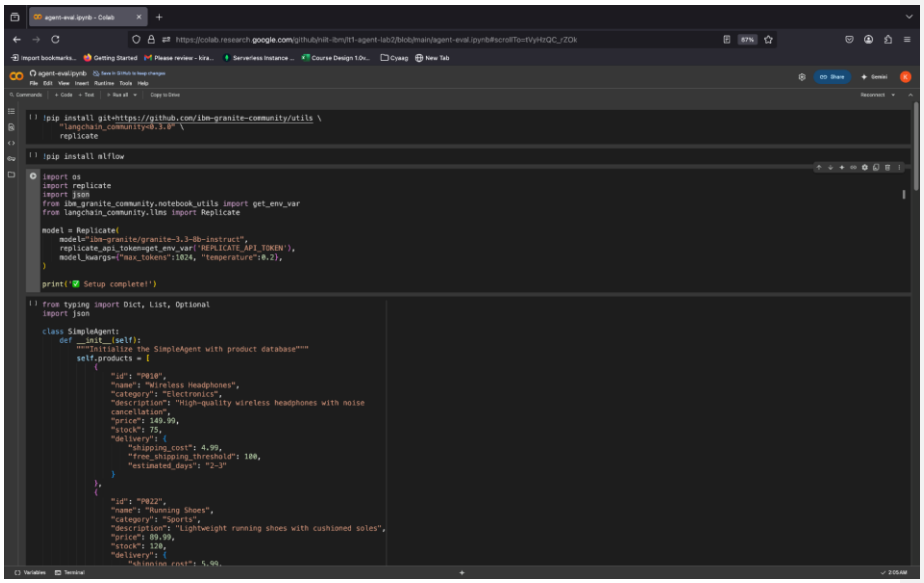
Commented [DA11]: option or tab?

Formatted: Font: Not Bold

Alt text: An image showing the “Open notebook” dialog box with the GitHub URL in the search~~search~~ field

Formatted: Font: Not Bold

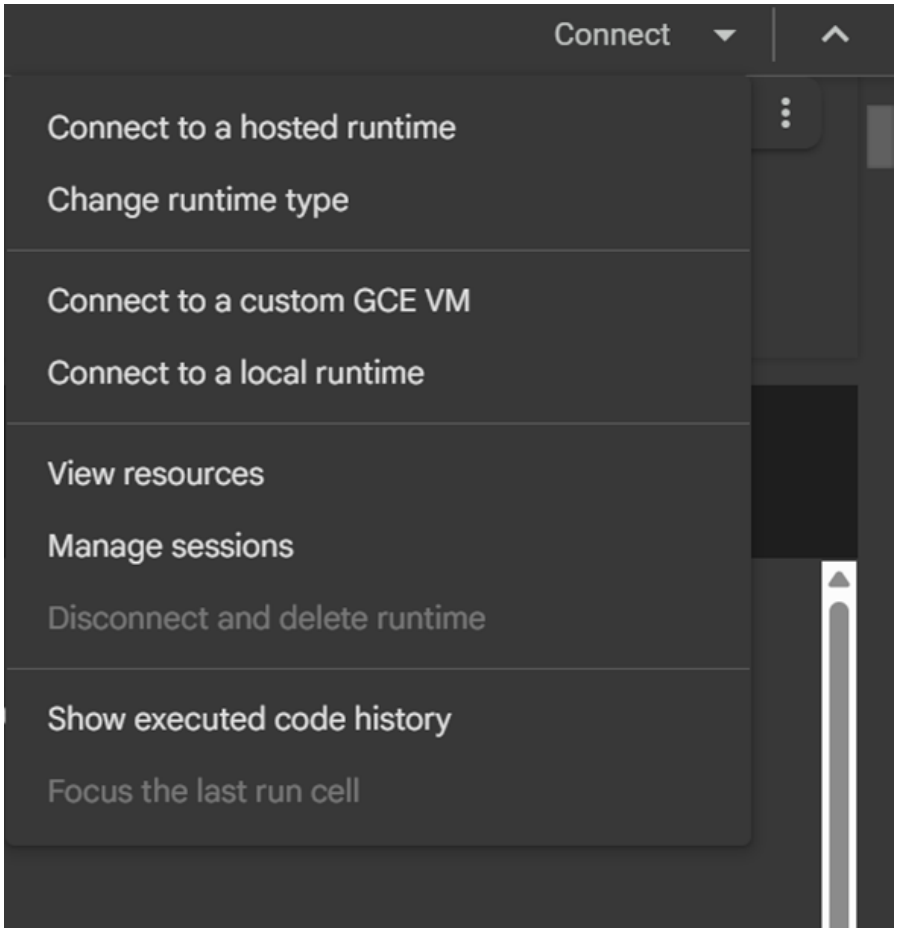
30. The “agent-eval.ipynb” notebook opens in the Colab workspace.



Alt text: An image showing the Google Colab workspace displaying the cells in the “agent-eval.ipynb” ~~selected~~ Jupyter notebook~~, agent-eval.ipynb~~

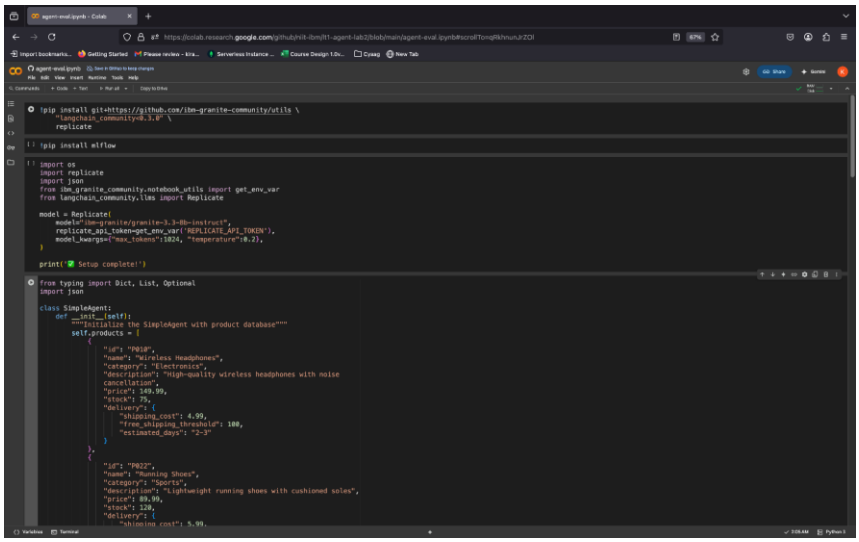
31. To run the code, select the **Connect to a hosted runtime** option from the **Connect** drop-down list of the notebook.

    Note: When you use Colab, your code doesn't run on your local machine. Instead, it runs ~~on~~ in a cloud-based environment provided by Google, known as a **hosted runtime**. This means you don't need to install Python or any libraries on your device; they're already available in the cloud environment.



Alt text: An image showing a Jupyter notebook with the **Connect to a hosted runtime** option in the **Connect** drop-down list

32. Check the **Colab** toolbar for a green checkmark. This indicates that the connection is successful and your code is now ready to run.



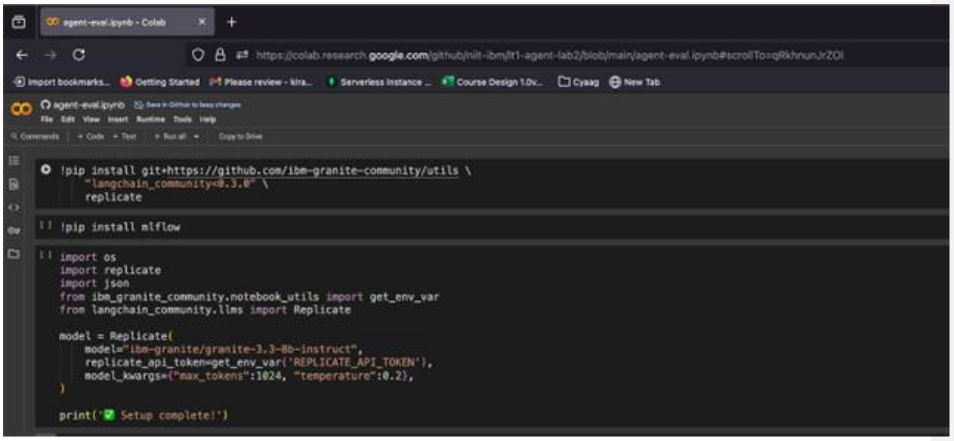
Alt text: An image showing the Google Colab workspace displaying a green checkmark indicating successful connection to the Google Colab ~~Google Colab~~ environment

Formatted: Font: Not Bold

7. To enable your AI agent to work with the IBM Granite model and generate helpful responses, certain libraries must be installed. In the first cell, select the run icon to begin the installation of the libraries.

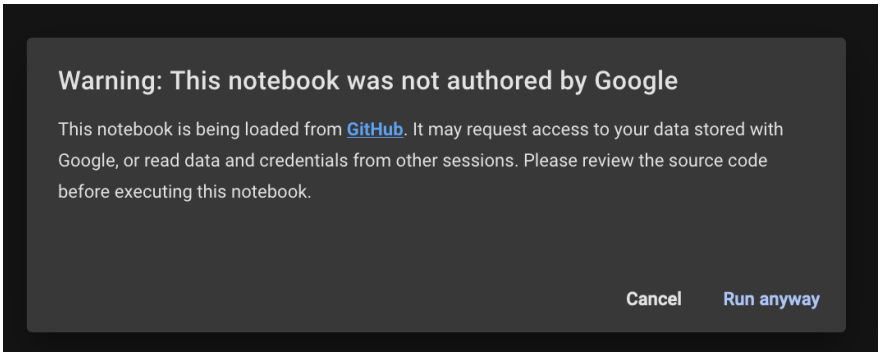
Note: In a Jupyter notebook, each row is called a **cell**. These cells are not numbered~~,~~ but are organized to run sequentially, beginning from the first cell. Each code cell includes a run icon (▶), which you can use to execute the code ~~inside~~ within that cell.

Formatted: Font: Bold



Alt text: An image showing a cell ~~of~~ in the "agent-eval-ipynb" Jupyter notebook, with the run icon and a prompt to install the required libraries

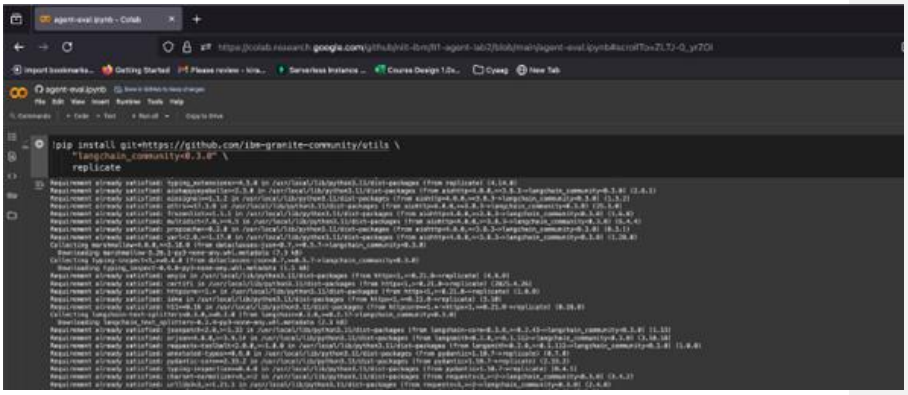
8. Select the **Run anyway** option continue loading the required libraries.



Alt text: An image showing the "Warning: This notebook was not authored by Google" message with the **Run anyway** option

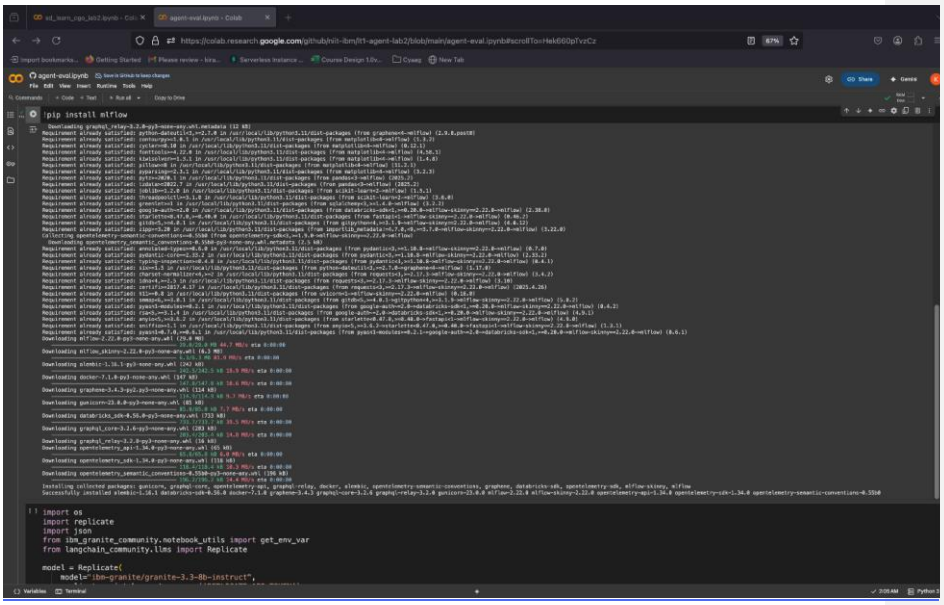
Formatted: Font: Not Bold

9. A green checkmark will appear next to the run icon, indicating that the required libraries have been successfully installed.



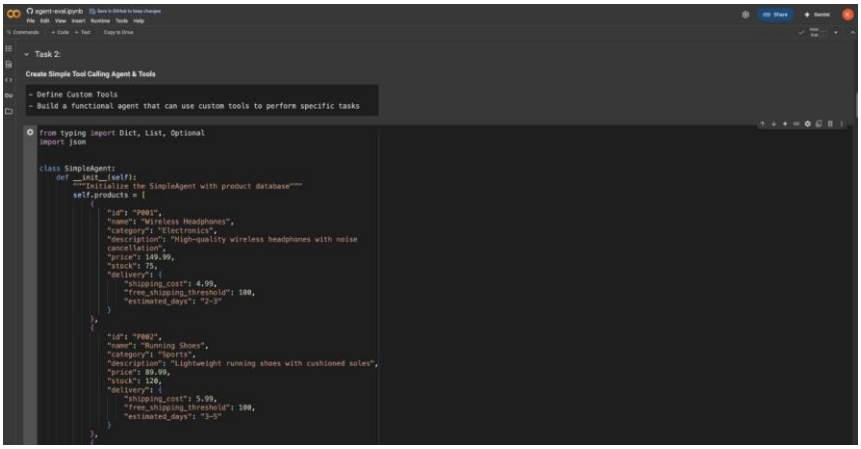
Alt text: An image showing the Google Colab workspace, ~~displaying~~ with a ~~check mark~~checkmark in the first cell of the Jupyter notebook ~~to indicate~~indicating successful installation of the required libraries

33. Next, install MLFlow by selecting the run icon in the second cell.



Alt text: An image showing ~~a~~ Python code with the run icon, used to install MLFlow

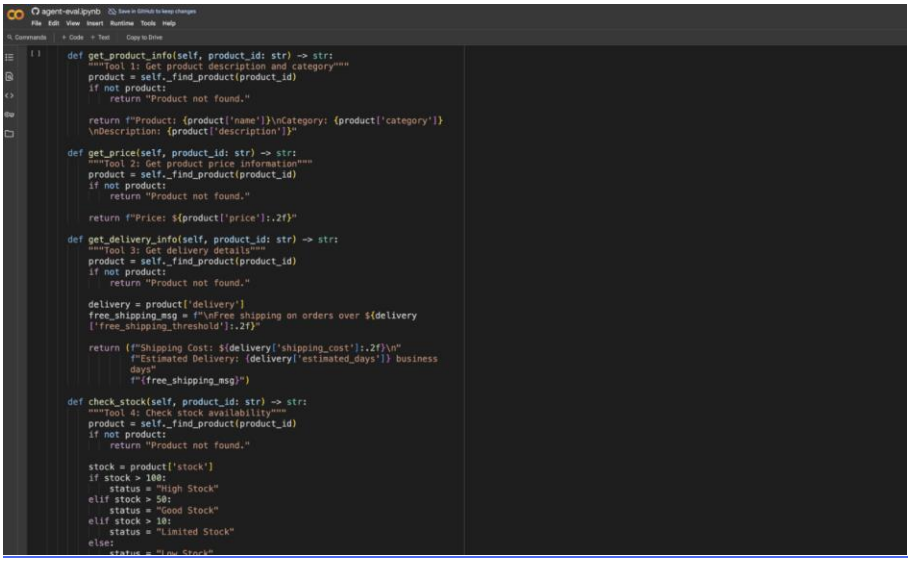
11. Next, to create a database for TechMart's product inventory, select the run icon in the first cell of Task 2.



Alt text: An image showing ~~a~~ Python code in ~~a~~ the Google Colab workspace ~~displaying the definition of~~ that defines a ~~dataset~~ data set for product inventory, including product **id**, **name**, **category**, **description**, **price**, **stock**, and **delivery** information

Commented [DA12]: Why are they in bold?

12. To enable the agent to process user queries and generate meaningful responses, define the required tools such as **get_product_info**, **get_price, get_delivery_info**, and **check_stock**.



```
def get_product_info(self, product_id: str) -> str:
    """Tool 1: Get product description and category"""
    product = self._find_product(product_id)
    if not product:
        return "Product not found."

    return f"Product: {product['name']}\nCategory: {product['category']}
    \nDescription: {product['description']}"

def get_price(self, product_id: str) -> str:
    """Tool 2: Get product price information"""
    product = self._find_product(product_id)
    if not product:
        return "Product not found."

    return f"Price: ${product['price']:.2f}"

def get_delivery_info(self, product_id: str) -> str:
    """Tool 3: Get delivery details"""
    product = self._find_product(product_id)
    if not product:
        return "Product not found."

    delivery = product['delivery']
    free_shipping_msg = f"\nFree shipping on orders over ${delivery
    ['free_shipping_threshold']:.2f}"

    return (f"Shipping Cost: ${delivery['shipping_cost']:.2f}\n"
            f"Estimated Delivery: {delivery['estimated_days']} business
            days"
            f"{free_shipping_msg}")

def check_stock(self, product_id: str) -> str:
    """Tool 4: Check stock availability"""
    product = self._find_product(product_id)
    if not product:
        return "Product not found."

    stock = product['stock']
    if stock > 100:
        status = "High Stock"
    elif stock > 50:
        status = "Good Stock"
    elif stock > 10:
        status = "Limited Stock"
    else:
        status = "Low Stock"
```

Alt text: An image showing ~~a~~ Python code displaying the definition of several functions, including **get_product_info**, **get_price**, **get_delivery_info**, and **check_stock**

13. To initialize the agent, select the run icon. The agent analyzes the user's input to identify whether it's asking for a product description or cost. Based on this, it uses the appropriate tools. For other types of input, the agent generates a general response.

```
from langchain.agents import initialize_agent, AgentType, tools

# Create the agent with tools
agent = initialize_agent(
    tools=tools,
    llm=model,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True,
    handle_parsing_errors=True,
    max_iterations=3
)

print("Agent initialized successfully with tool calling capabilities")
```

Alt text: An image showing Python code displaying the initialization of an AI agent using the **langchain.agents** library

Step 5: Trajectory analysis and final response evaluation

Commented [DA13]: Above, Step 5 is written as **Evaluate the trajectory and final response of the AI agent** . Make this consistent at both the places, like all other steps.

Overview

In this step, you'll evaluate the agent's trajectory and assess the accuracy, clarity, and relevance of its final responses using MLFlow.

Instructions

34. To test how the agent responds, select the run icon to create an evaluation set. This set includes different types of user requests, expected factual answers, and operational rules that guide how the agent should behave.

    Note: This evaluation set will be used by MLFlow to test the ~~agent's~~ agent's . Each entry specifies the ~~user's~~ user's request, the expected output, and any specific guidelines the ~~agent's~~ agent's response should follow.

```
[ ] def try_agent(example_name: str, user_input: str):
        with mlflow.start_run(run_name=example_name):
            mlflow.log_param("user_input", user_input)

            # Simulate agent response
            response = try_agent(user_input)

            mlflow.log_param("agent_response", response)
            print(f"\n{example_name}")
            print("Input:", user_input)
            print("Response:", response)

            # Optionally log output to a text file
            with open(f"{example_name.replace(' ', '_')}_output.txt", "w") as f:
                f.write(f"Input: {user_input}\nResponse: {response}")
            mlflow.log_artifact(f"{example_name.replace(' ', '_')}_output.txt")
```

    Alt text: An image showing the code for analyzing the trajectory of the ~~**trajectory of the**~~ "agent-eval.ipynb" AI agent ~~**agent-eval.ipynb AI agent**~~

Formatted: Font: Not Bold

Formatted: Font: Not Bold

35. Analyze the tool-calling results ~~Tool Calling Results~~ to determine whether the agent correctly interprets the ~~user's~~ user's query ~~correctly~~ and ~~calls or~~ uses the appropriate tools to generate a response.

Commented [DA14]: Since it is the generic description, writing it this way.

```
==========================================================
Eval 1: Product Information
==========================================================

Tool Calling Results:

  Tool Call 1:
    Tool Name   : get_product_info
    User Input  : I'm looking for an ergonomic wireless mouse
    Output      : Product P045 is a wireless mouse with ergonomic support and programmable buttons.
==========================================================
```

    Alt text: An image showing the Tool Calling Results of **Eval 1: Product Information**

Commented [DA15]: Please check if we can rephrase it and describe in it a generic way, which will be more clear. Like "An image showing the tool-calling results from the evaluation of product information

36. To evaluate whether the agent is able to handle follow-up questions by using multiple tools while ~~still~~ retaining the user's original query and context, select the run icon. The code snippet ~~here~~ shows the user asking, "Is the product available in stock?"~~, and~~ prompting the agent ~~calls~~

to call multiple backend tools, such as **get_product_info** and **check_inventory**~~.~~, ~~The agent~~and ~~returns~~ return a response based on this query.

```
# Your tests here!
try_agent(
    query="Is the product available in stock?",  # Your question
    tool=agent-tool
)
```

Alt text: An image showing a code snippet in ~~a~~ the Google Colab notebook with a -function call to **try_agent()** and a query by the user

Commented [DA16]: This part is not clear. Please check if we can revise and make it generic for clarity.

```
============================================================
Eval 1: Product Information
============================================================

Tool Calling Results:

  Tool Call 1:
    Tool Name   : get_product_info
    User Input  : I'm looking for an ergonomic wireless mouse
    Output      : Product P045 is a wireless mouse with ergonomic support and programmable buttons.

  Tool Call 2:
    Tool Name   : check_inventory
    User Input  : Is the product available in stock?
    Output      : Product P045 is currently available in your selected region.
============================================================
```

Alt text: An image showing the results of **Eval 1: Product Information**, with two tool calls

Commented [DA17]: Please write it in a generic way. Something like - **An image showing the results from the evaluation of product information based on two tool calls**

37. For the final agent evaluation, select the run icon.

Task 4:

Final Response Evaluation

- Evaluating Final responses from agent against the defined metrics

```
def try_agent(example_name: str, user_input: str):
    with mlflow.start_run(run_name=example_name):
        mlflow.log_param("user_input", user_input)

        # Simulate agent response
        response = try_agent(user_input)

        mlflow.log_param("agent_response", response)
        # Evaluate the response
          evaluation = evaluate_response(query, tool, response)

          # Print results
          print(f"Query: {query}")
          print(f"Tool Used: {tool}")
          print(f"Response: {response}")
          print(f"Evaluation Score: {evaluation['score']}/5")
          print(f"Feedback: {evaluation['explanation']}")
```

Alt text: An image showing Python code that evaluates the final responses from the agent against the defined metrics

Commented [DA18]: Changed to “responses” in alignment with the graphic

5. After the execution is finished, you’ll see ~~a **Final Agent Response Summary**~~the response summary. Review the output and evaluate the results for accuracy and relevance.

Formatted: Font: Not Bold

Formatted: Font: Not Bold

Note: This step shows that even when the agent successfully uses tools to gather information (as in Tool Calls 1 and 2), its final response still requires evaluation and refinement. The ~~Evaluation~~ evaluation ~~Score~~ score and ~~Feedback~~ feedback are important for improving the ~~agent's~~ agent’s performance over time, even when its answers are already relevant and accurate.

Alt text: An image showing the agent's final response output from different tool calls, along with ~~and a **Final Agent Response Summary** is presented~~the response summary

Conclusion

Congratulations! ~~You have~~You've effectively evaluated TechMart's AI agent. Through structured testing and analysis with MLFlow, you assessed the agent's tool-calling efficiency and the quality of its final responses, ensuring accurate, relevant, and clear product information delivery.