

GitHub Copilot

Improvise Python App using GitHub Copilot



Version	Revision Date	Description	Author(s)	Reviewed by	Approved by
V1.0	02-05-2024	Initial Version	jaya.narayana	r.a.perumalsamy / a.g.swaminathan	sreenivasa.cpcl.rao

Table of Contents

Use Case Details: 4

Learning Objectives: 4

Know about Virtual Environment – GitHub Codespaces: 4

Pre-requisites: 4

Activity Overview: (Detailed Instructions are given in the below section)..... 4

Instructions: 5

Clean up activity:..... 17

Learning Outcome:..... 18

Troubleshooting Steps: 18

Use Case Details:

Meet Mr. Mark, who has recently joined the ABC project as a fresher. He wishes to increase his productivity and speed up coding operations with Python.

Mr. John, Mr. Mark's manager, supplied the Python code source and asked for modifications to create an interactive HTML (Hyper Text Markup Language) form and an API endpoint. This will allow Mark to gain considerable experience constructing a Python web app that serves an HTTP API while generating pseudo-random tokens for identifying reasons.

Mark had a close timeline to finish his task and started exploring options to create code using templates or getting code generate assistances. During his study, he discovered GitHub Copilot AI (Artificial Intelligence) tool as a code assistant tool. Also, he discovered that GitHub Copilot is an AI pair programmer that generates suggestions based on context and code patterns. Let us help Mark to complete his task with GitHub Copilot.

Learning Objectives:

1. Experiences VS Code (Web) in GitHub Codespaces as a development environment.
2. Develop interactive HTML form and Application Programming Interface (API) endpoint.
3. Use GitHub Copilot as an assistant to create code, add comments, generate testcase, design web page and explain the code.
4. Build a Python App which contains code generated by GitHub Copilot.

Know about Virtual Environment – GitHub Codespaces:

Codespaces are cloud-based development environments provided by GitHub. They allow developers to code, build, test, and debug their projects entirely in the cloud, removing the need for local development environments. With Codespaces, developers can access their projects from any device with an internet connection, collaborate with teammates in real-time, and seamlessly switch between different development environments.

GitHub Codespaces provide pre-configured development environments with all the necessary tools and dependencies already installed, enabling developers to get started quickly without the hassle of setting up their development environment manually. This is particularly useful for onboarding new team members, working on multiple projects, or accessing development environments from different devices.

Additionally, Codespaces integrate tightly with GitHub repositories, making it easy to spin up a development environment directly from a repository. This tight integration also ensures that changes made in a Codespace are automatically synced back to the GitHub repository, enabling seamless collaboration and version control.

Overall, GitHub Codespaces streamline the development workflow by providing developers with a flexible, scalable, and collaborative development environment in the cloud.

Pre-requisites:

1. Make sure that you have **high and stable internet bandwidth connectivity** to work on remote environments.
2. Recommended to use Google Chrome browser for seamless connectivity.
3. Signup in [GitHub.com \(https://github.com/\)](https://github.com/) using **Accenture mailid** only and share username to get access of GitHub Copilot Subscription. ([License Requestion Link](#) or <https://atcittrainingtracker.accenture.com/GitHubCopilotRequisitionTool/home>), in case if you haven't done.
4. GitHub Copilot subscription invite is accepted and joined in this organization **GitHubCopilotTDLc**.
5. Sign-in to GitHub using this [link \(https://github.com/login\)](https://github.com/login) using **Accenture mailid**. If you have already signed in using your personal email address or any other email address sign out first and then sign in using email address where GitHub Copilot license is activated.

Activity Overview: (Detailed Instructions are given in the below section)

The API already has a single endpoint to generate a token. Let's update the API by adding a new endpoint that accepts text and returns a list of tokens.

1. Add Pydantic model to be used in a new route that will accept JSON.

GitHub Copilot

- Access main.py file and add a comment so that GitHub Copilot can generate a Pydantic model for you.
2. Explore the generate () function in the Python script and use generate () function for establishing a new endpoint.
 - Provide a comment to GitHub Copilot to create a FastAPI endpoint.
 - Endpoint should accept a POST request with a JSON body.
 - JSON Body should contain a single field called "text."
 - It should return a checksum of the text.
3. Add a Welcome Note to the user with customized message with participant(your) name.
4. Create documentation for the new API endpoint for better readability.
 - Explain the code.
 - Add Comments.
5. Create test cases for the generate () route using the FastAPI test client using GitHub Copilot.
6. Fix the warning messages using GitHub Copilot features.
7. Create a Readme.md file to highlight about how to execute the application using uvicorn webserver.

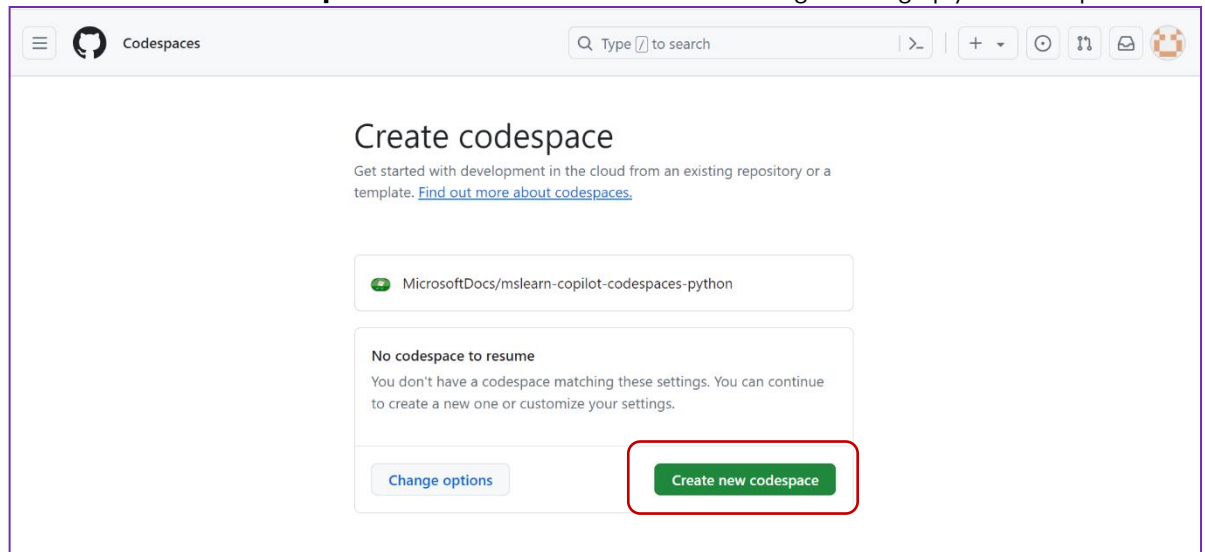
Instructions:

1

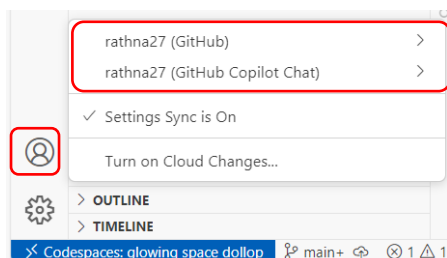
Open the Codespace with the preconfigured environment in your browser with this [link](https://codespaces.new/MicrosoftDocs/mslearn-copilot-codespaces-python), or copy and paste this link in the browser address bar **<https://codespaces.new/MicrosoftDocs/mslearn-copilot-codespaces-python>**

Note: This codespace environment should be used only for training purpose. It is not for production usage. Refrain uploading / adding Accenture / Client specific data in this codespace

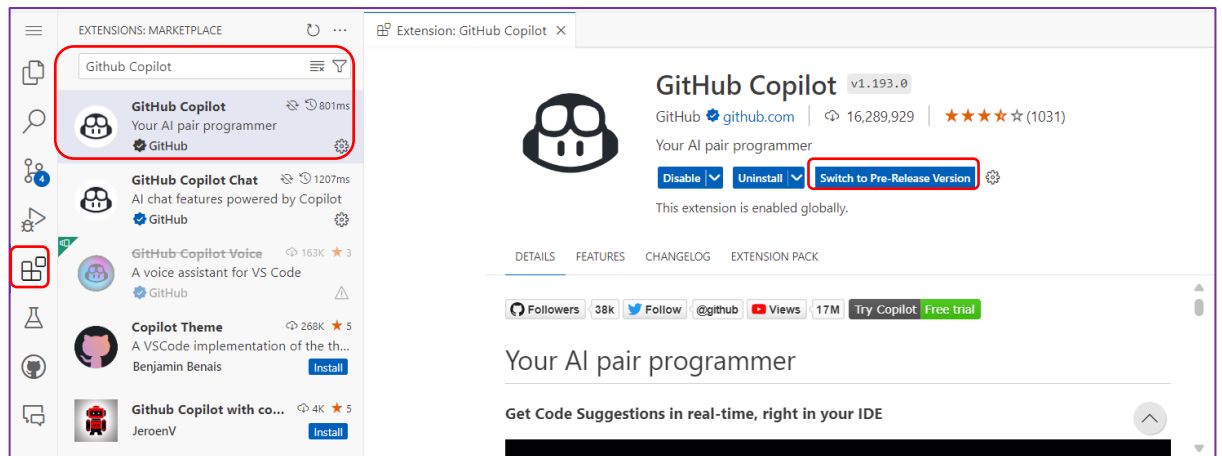
Click on **"Create new Codespace"**. It will take 2 to 5 minutes for creating & setting up your Codespace.



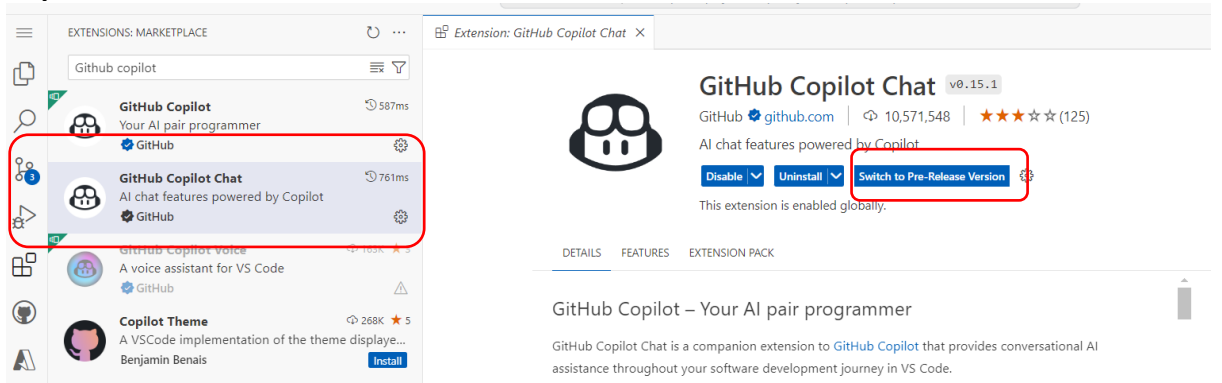
Ensure your Codespace environment is logged in with your GitHub Co-pilot license enabled username by clicking on profile icon at the bottom left in the GitHub Codespace environment.



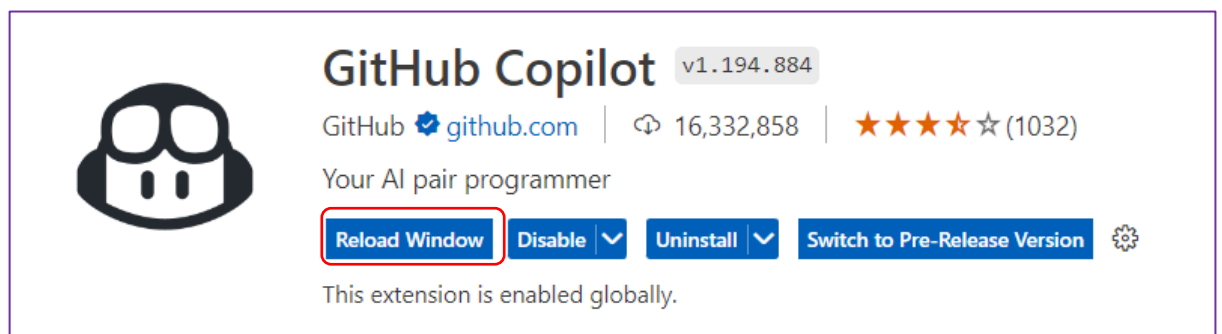
Switch to Pre-release version of GitHub Copilot extension by clicking on **Extensions -> GitHub Copilot -> select "Switch to Pre-Release Version."**



Switch to Pre-release version of GitHub Copilot Chat extension by clicking on **Extensions -> GitHub Copilot Chat -> select "Switch to Pre-Release Version."**



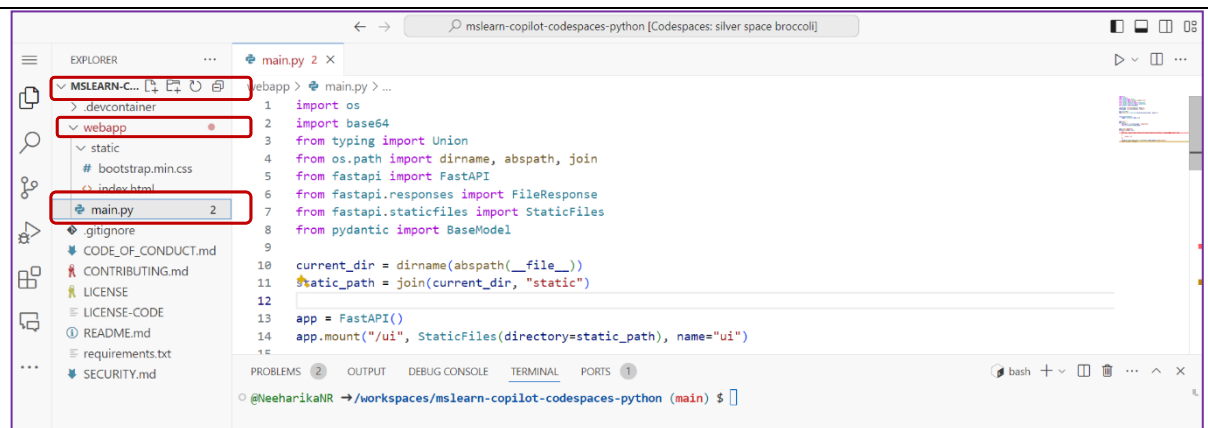
Click on Reload Window and the Codespaces will be reloaded.



Case1 Add Pydantic model to be used in a new route that will accept JSON.

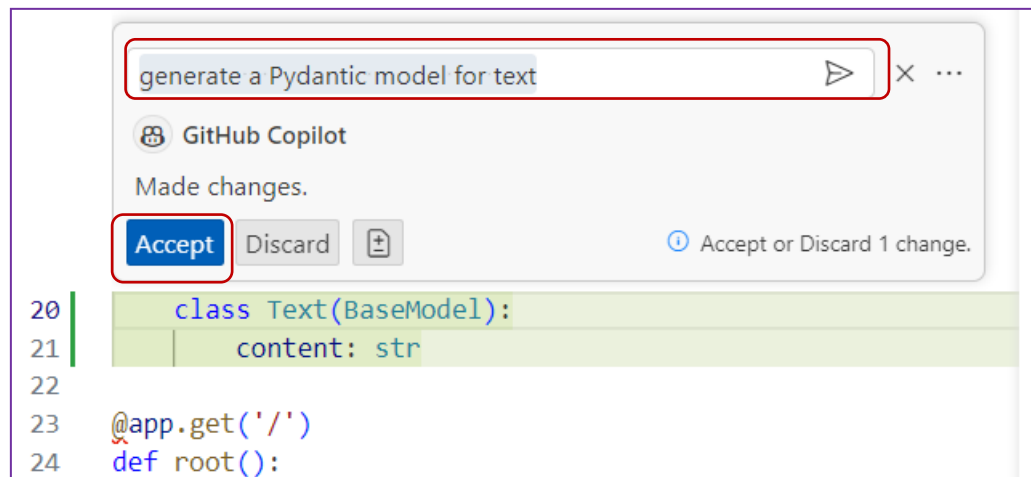
For implementing this case, navigate to main.py file.

In Explorer (Ctrl+Shift+E) left-side pane, Click on **MSLEARN-COPILOT-CODESPACES-PYTHON -> webapp -> main.py**

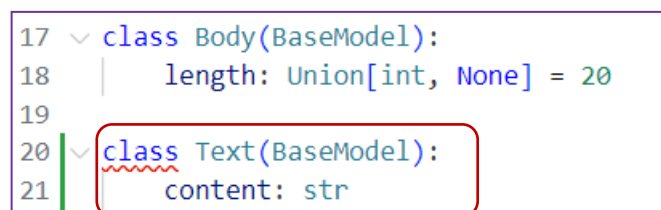


In line 20, Press Ctrl + I and instruct GitHub Copilot by typing **generate a Pydantic model for text**.

Accept the changes and if there is any issue faced in this step, you can follow the troubleshooting instructions given in page number **18**.



Align the newly added line, like Line number 17 and 18 (Make it to be left-aligned).



Press **Ctrl+S** to save the changes.

Case
2

Explore the generate () function in the Python script and use generate () function for establishing a new endpoint.

- Provide a comment to GitHub Copilot to create a FastAPI endpoint.
- Endpoint should accept a POST request with a JSON body.
- JSON Body should contain a single field called "text." It should return a checksum of the text.

We need to implement this case in **main.py**.

Add a new line after line number 39, Press Ctrl + I and instruct GitHub Copilot by typing **Create a FastAPI endpoint that accepts a POST request with a JSON body containing a single field called "text" and returns a checksum of the text.**

```

37     """
38     string = base64.b64encode(os.urandom(64))[:body.length]
39
40     Create a FastAPI endpoint that accepts a POST
41     request with a JSON body containing a single field
42     called "text" and returns a checksum of the text
43

```

Review the code changes suggested by GitHub Copilot and accept the changes.

Change 1:

```

from pydantic import BaseModel

```

Create a FastAPI endpoint that accepts a POST request with a JSON body containing a single field called "text" and returns a checksum of the text.

GitHub Copilot

Made changes.

Accept Discard

Accept or Discard 2 changes.

```

import hashlib

```

Change 2:

Create a FastAPI endpoint that accepts a POST request with a JSON body containing a single field called "text" and returns a checksum of the text

GitHub Copilot

Made changes.

Accept Discard

Accept or Discard 1 change.

```

@app.post('/checksum')
def calculate_checksum(text: Text):
    """
    Calculate the checksum of the given text.
    Example POST request body:

```

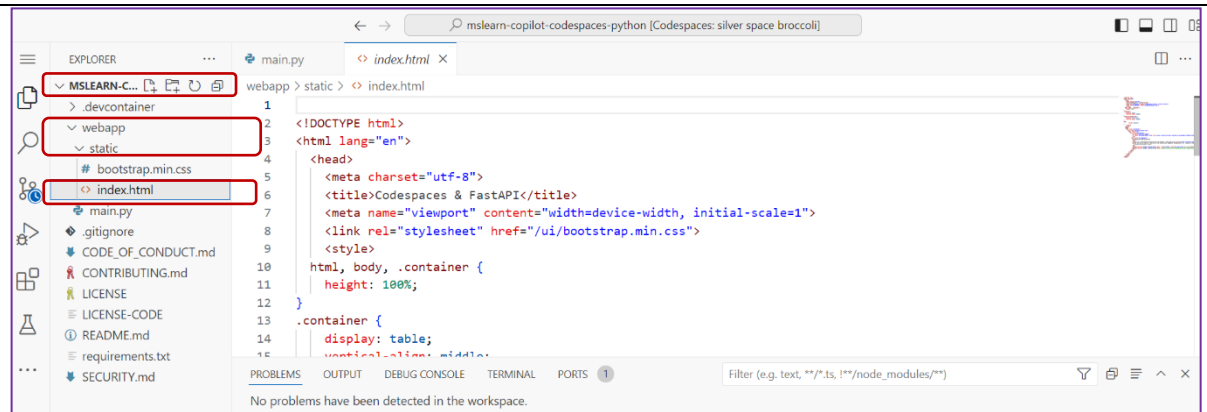
Press **Ctrl+S** to save the changes.

Case 3

Add a Welcome Note to the user with customized message and participant(your) name.

For completing this case, we need to navigate to **"index.html"**.

MSLEARN-COPILOT-CODESPACES-PYTHON -> webapp -> static -> index.html



Instruct GitHub Copilot to add welcome message with your name.

After line 28, insert a new line and type this //Add this message – Welcome to the webpage, **YourName**.

Press **enter** to view the suggestion and Press **Tab** to confirm the suggestion.

Remove the line //Add this message – Welcome to the webpage, **YourName** once the suggestions are confirmed after pressing tab.

Replace your first name instead of **YourName** in above message.

Ctrl+S to save the file.

```

27 | <div class="container">
28 | <div class="vertical-center-row">
29 | //Add this message - Welcome to the webpage, Jaya Narayana!
30 | <h1>Welcome to the webpage, Jaya Narayana!</h1>
31 | <div class="row">
32 | <div class="col-lg-12">

```

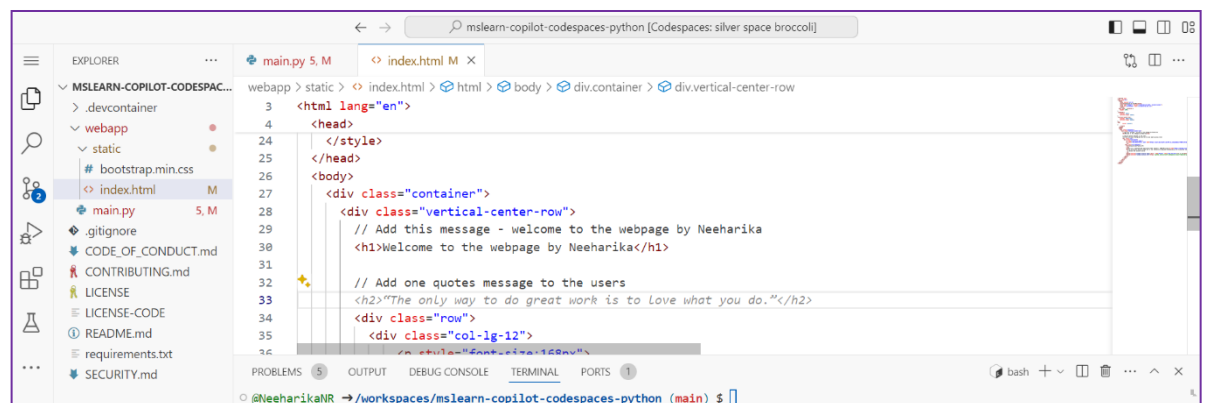
Now insert new line after the last update and type // Add one quotes message to the users and press enter.

You will see the suggestion. Press **Tab** to confirm the suggestion.

Remove the line // Add one quotes message to the users once the suggestion is confirmed after pressing tab.

Ctrl+S to save the file.

Note: Every time you will be able to see different quotes.



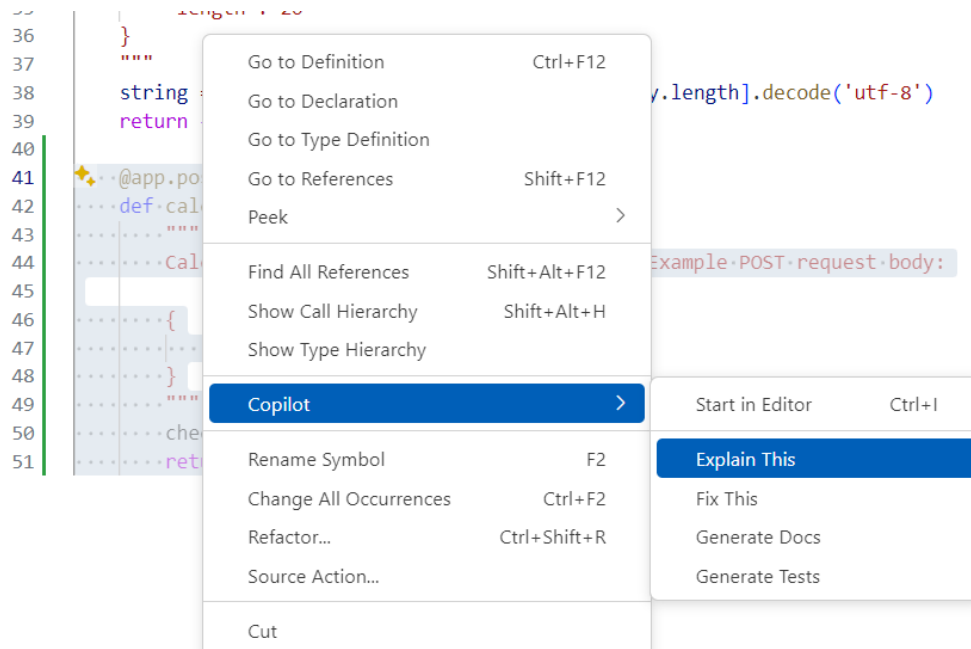
Case
4

Create documentation for the new API endpoint for better readability.
➤ Explain the code.
➤ Add Comments

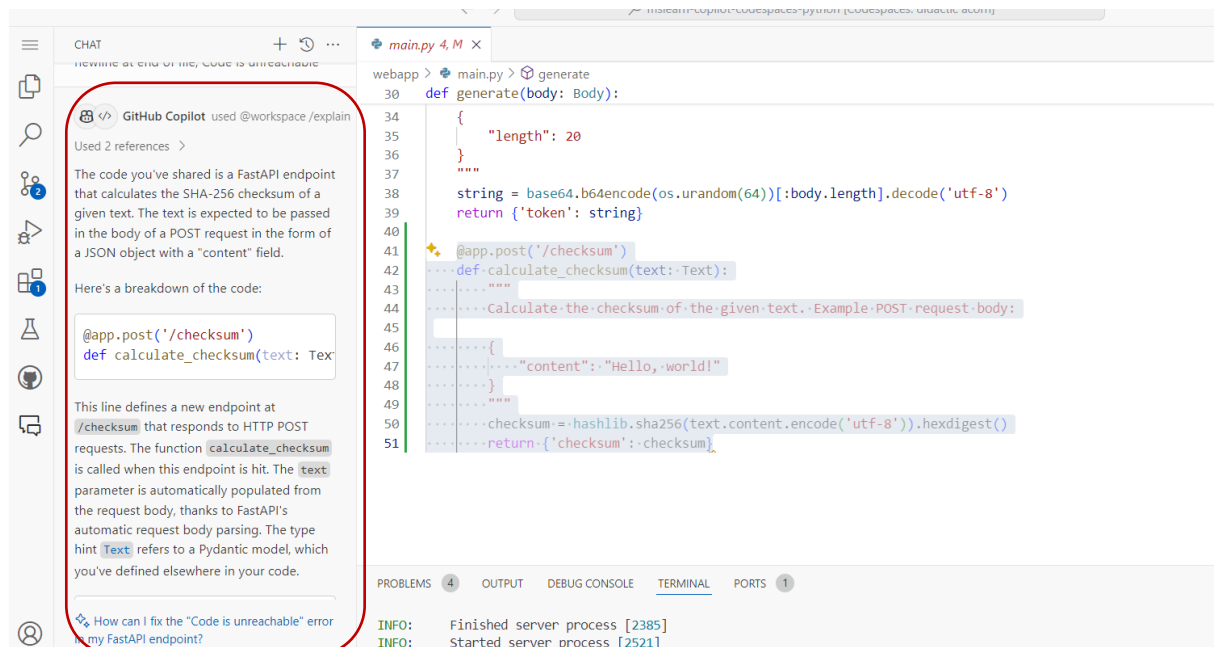
For implementing this case, navigate to **main.py** file.

MSLEARN-COPILOT-CODESPACES-PYTHON -> webapp -> main.py

Select the newly generated code (created during case 2 which is added at the end of the file) and right-click to select **Copilot -> Explain This** and instruct GitHub Copilot to explain the code.

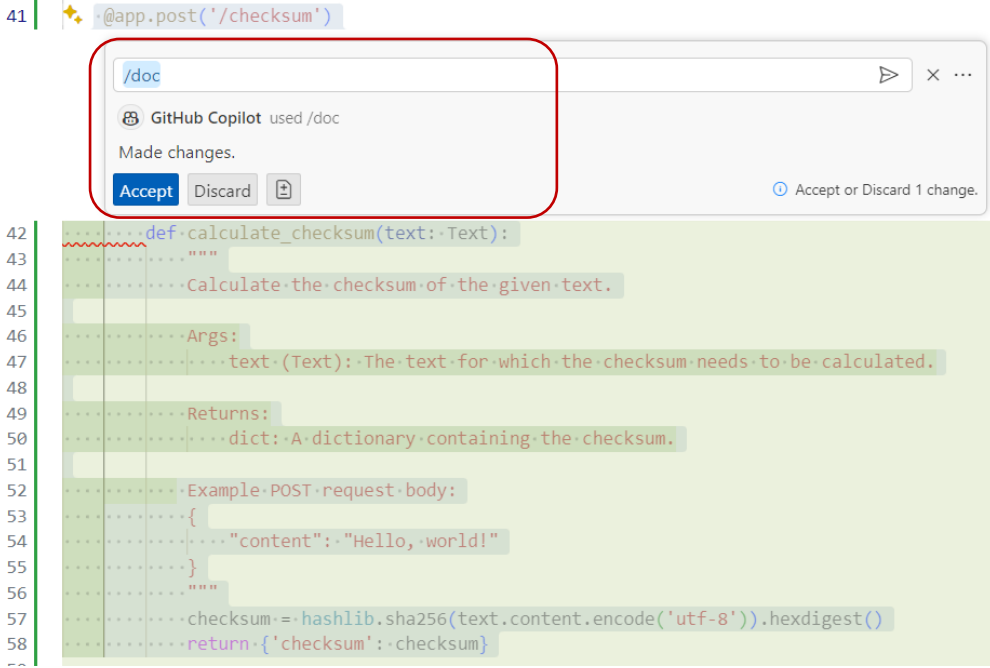
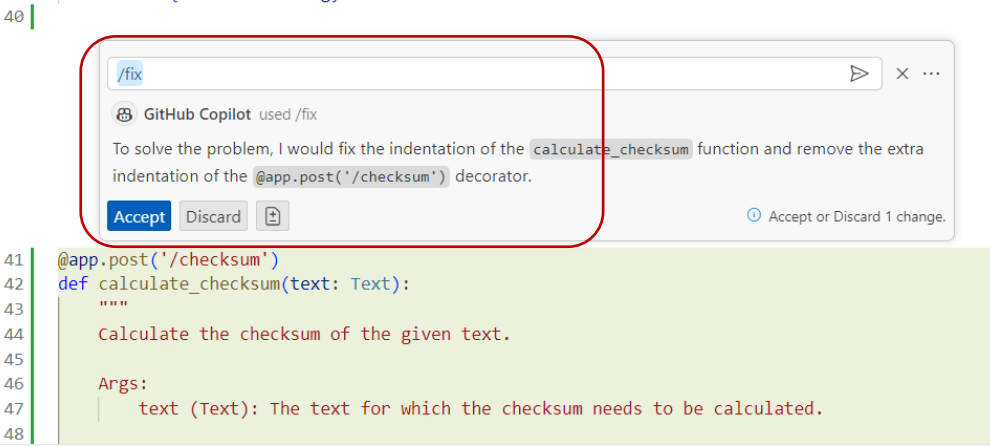


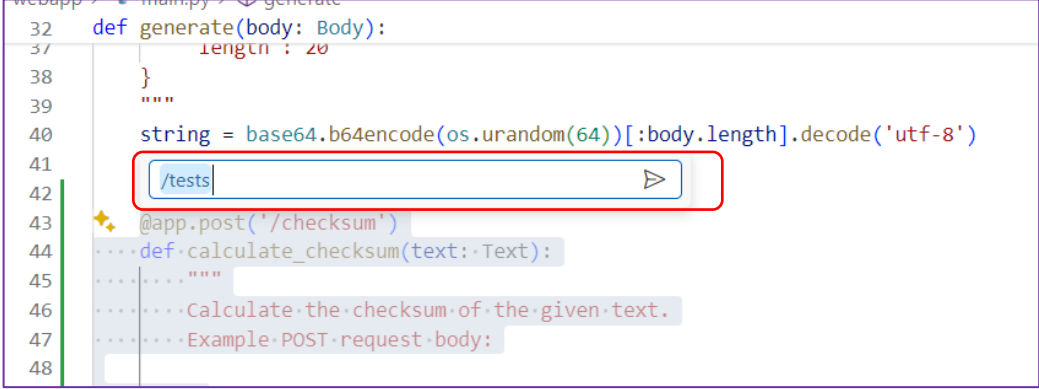
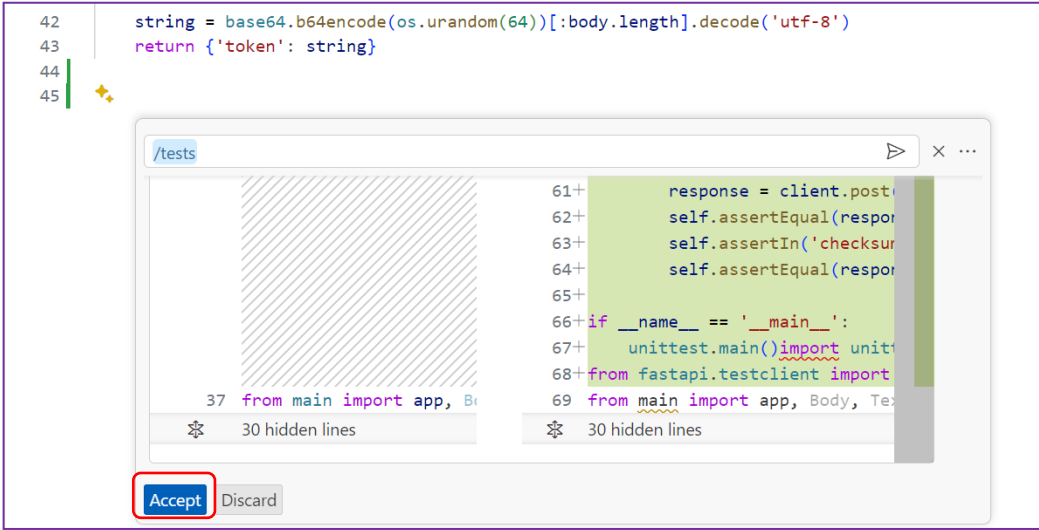
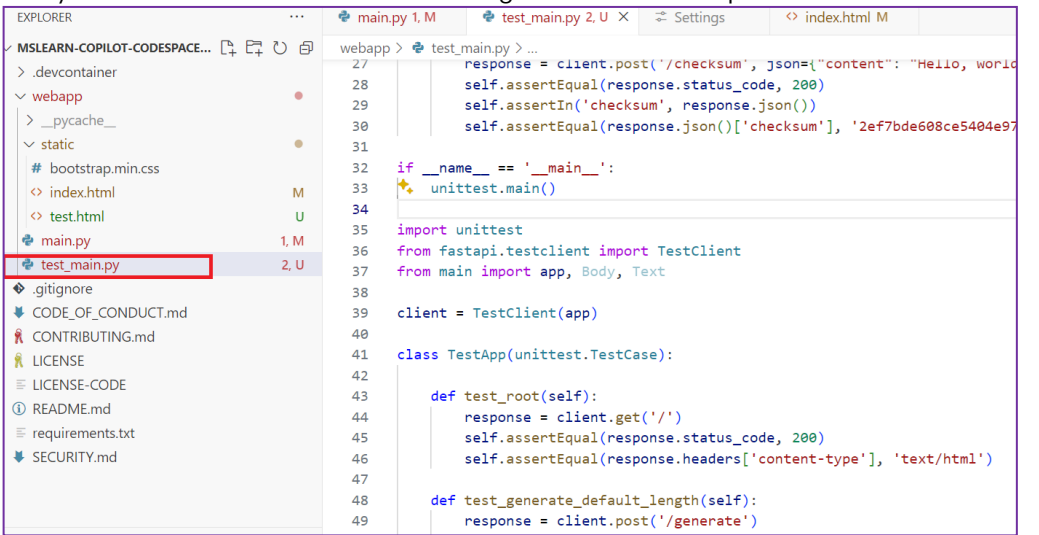
You can observe, GitHub Copilot chat window will be launch in the left panel and you can see the detail description about the code.

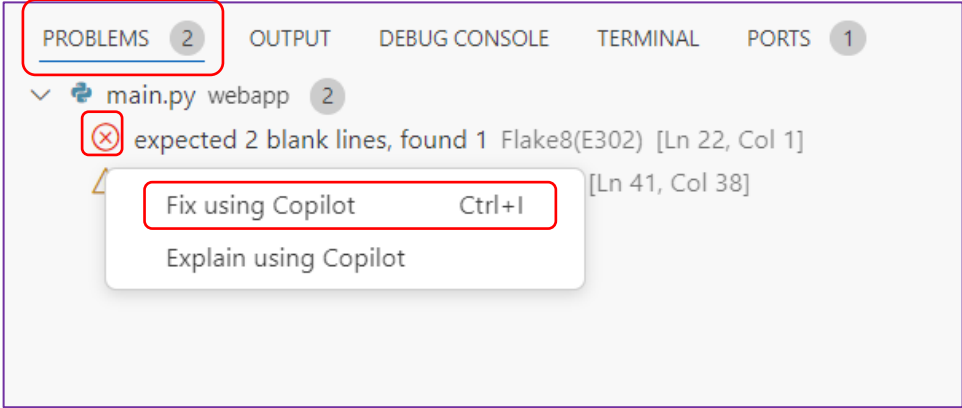
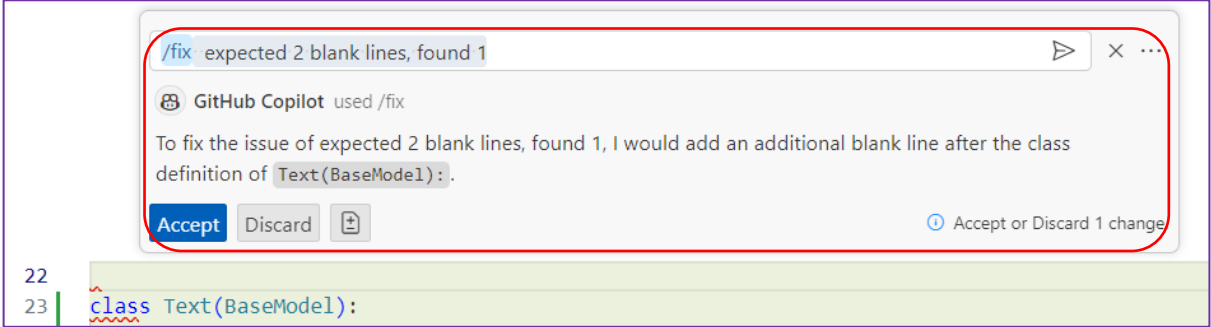


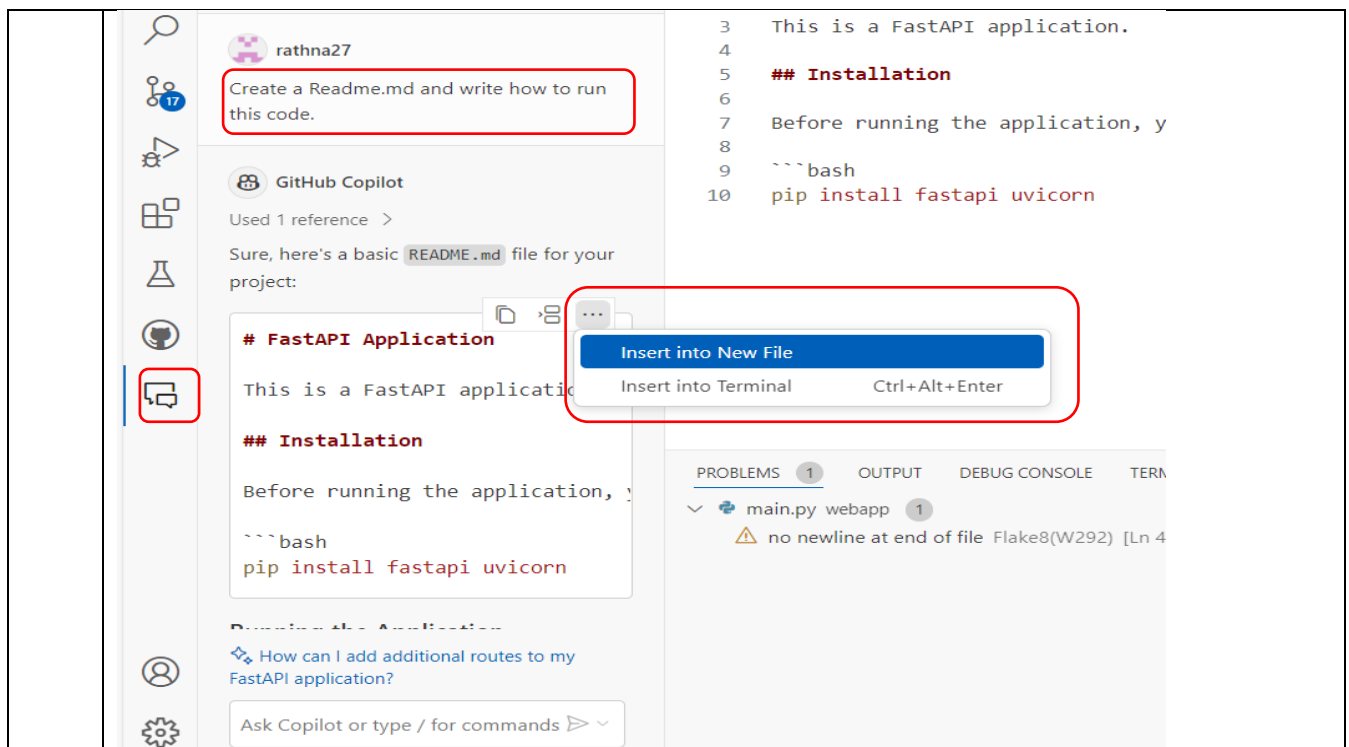
Again, select the newly generated code (created during case 2) and press **Ctrl + I** and instruct GitHub Copilot to add comment to the code using `/doc`.

Press **Ctrl+I** and type `/doc` and then press enter. You can Accept the changes using accept button.

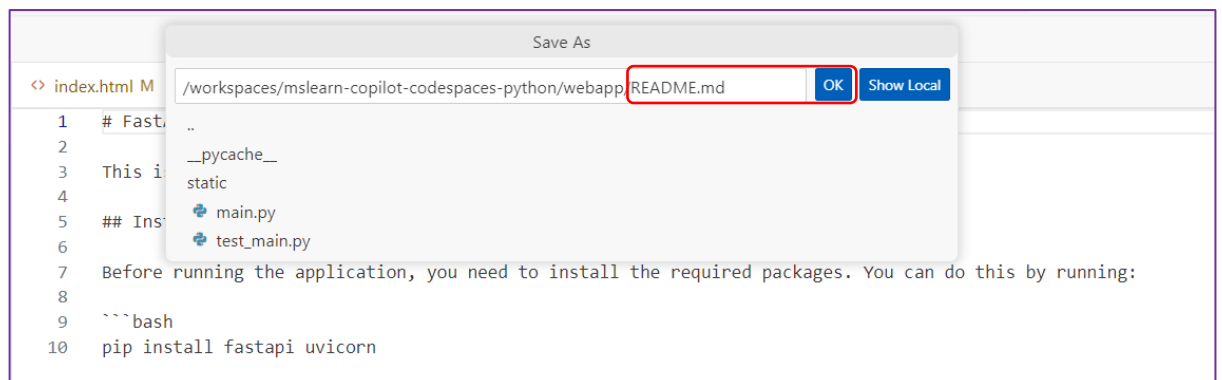
	<div data-bbox="311 197 1305 862">  <pre> 41 @app.post('/checksum') 42 def calculate_checksum(text: Text): 43 """ 44 Calculate the checksum of the given text. 45 46 Args: 47 text (Text): The text for which the checksum needs to be calculated. 48 49 Returns: 50 dict: A dictionary containing the checksum. 51 52 Example POST request body: 53 { 54 "content": "Hello, world!" 55 } 56 """ 57 checksum = hashlib.sha256(text.content.encode('utf-8')).hexdigest() 58 return {'checksum': checksum} </pre> </div> <p>Now you can observe the indentation error in the def Calculate_checksum line. To fix this click anywhere on the line and press Ctrl+I and then type /fix and press enter.</p> <p>Accept the fix using Accept button and press Ctrl+S to save the file.</p> <div data-bbox="311 1146 1305 1590">  <pre> 40 41 @app.post('/checksum') 42 def calculate_checksum(text: Text): 43 """ 44 Calculate the checksum of the given text. 45 46 Args: 47 text (Text): The text for which the checksum needs to be calculated. 48 </pre> </div>
<p>Case 5</p>	<p>Create test cases for the generate () route using the FastAPI test client using GitHub Copilot.</p> <p>For implementing this case, navigate to main.py file.</p> <p>MSLEARN-COPILOT-CODESPACES-PYTHON -> webapp -> main.py</p> <p>Select the newly generated code (created during case 2), press Ctrl + I and instruct GitHub Copilot to generate unit testcases for the code by typing /tests.</p>

	<div data-bbox="304 197 1347 584">  <pre> 32 def generate(body: Body): 33 length: 20 34 } 35 """ 36 string = base64.b64encode(os.urandom(64))[:body.length].decode('utf-8') 37 38 @app.post('/checksum') 39 def calculate_checksum(text: Text): 40 """ 41 Calculate the checksum of the given text. 42 Example POST request body: 43 """ </pre> </div> <p>Review and accept the changes.</p> <div data-bbox="304 640 1347 1167">  <pre> 42 string = base64.b64encode(os.urandom(64))[:body.length].decode('utf-8') 43 return {'token': string} 44 45 </pre> <p>Accept Discard</p> </div> <p>Now you can observe new testcase has been generated in the left panel.</p> <div data-bbox="304 1223 1347 1756">  <pre> 27 response = client.post('/checksum', json={"content": "Hello, world!"}) 28 self.assertEqual(response.status_code, 200) 29 self.assertIn('checksum', response.json()) 30 self.assertEqual(response.json()['checksum'], '2ef7bde608ce5404e97') 31 32 if __name__ == '__main__': 33 unittest.main() 34 35 import unittest 36 from fastapi.testclient import TestClient 37 from main import app, Body, Text 38 39 client = TestClient(app) 40 41 class TestApp(unittest.TestCase): 42 43 def test_root(self): 44 response = client.get('/') 45 self.assertEqual(response.status_code, 200) 46 self.assertEqual(response.headers['content-type'], 'text/html') 47 48 def test_generate_default_length(self): 49 response = client.post('/generate') </pre> </div> <p>Now access the new testcase file and review the generated test cases.</p>
Case 6	<p>Fix the warning messages using GitHub Copilot features.</p> <p>To implement this case, navigate to Problem tab which appears in the bottom of the IDE.</p> <p>View the warning messages, Move the cursor in the warning icon and click which pops up options. Select Fix using Copilot and get the warnings fixed.</p>

	 <p>Review the suggestions and accept the changes to fix the warnings.</p> <p>NOTE: GitHub Copilot while providing fixes for issues, generates multiple suggestions. Review the generated suggestions and accept the appropriate one.</p> 
<p>Case 7</p>	<p>Create a Readme.md file to highlight about how to execute the application using uvicorn webserver.</p> <p>For implementing this case, navigate to “main.py” and access GitHub Copilot Chat by clicking on Chat icon exists in left side panel.</p> <p>Type Create a Readme.md and write how to run this code in Copilot Chat. Review the suggestion and insert the generated content into new file using the option as highlighted below.</p> <p>Note: If “Insert into New File” option is not visible, then you can create a new README.md file manually and copy paste the content generated in the GitHub Copilot chat window.</p>



Once the new file is generated with the instructions, save file by pressing CTRL+S and type README.md as filename. (Highlighted below)



Now open the terminal in codespace

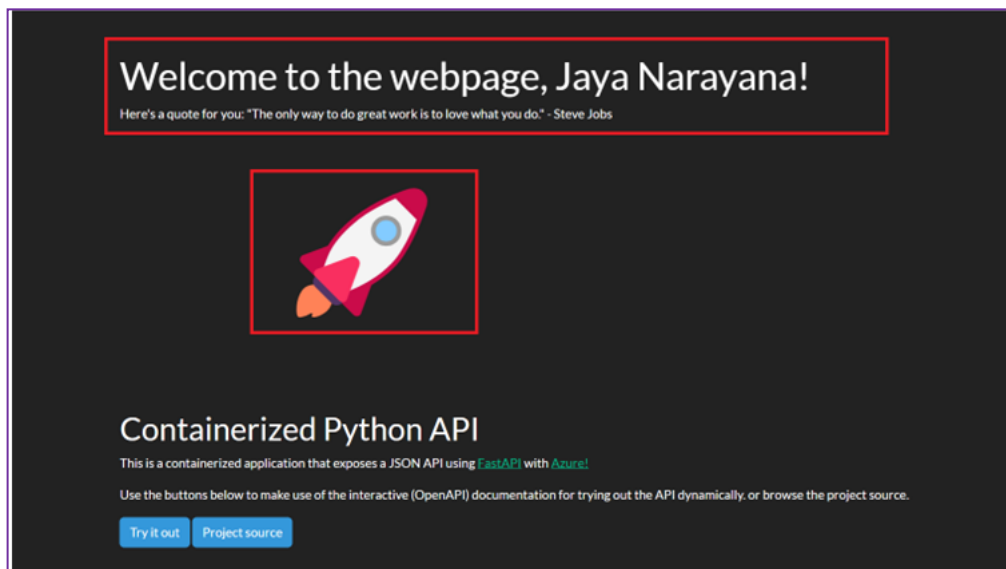
Press Ctrl + C and execute the application using below command generated by GitHub Copilot in the chat window.

`uvicorn --host localhost webapp.main:app --reload`



Now you can access application in the Browser by following the link enabled in terminal window **OR** Follow Link enabled in port window under Forwarded Address column.

```
@rathna27 → /workspaces/mslearn-copilot-codespaces-python (main) $ uvicorn --host localhost webapp.main:app --reload
INFO: Will watch for changes in these directories: ['/workspaces/mslearn-copilot-codespaces-python']
INFO: Uvicorn running on http://localhost:8000 (Press CTRL+C to quit)
INFO: Started reloader process [42978] using WatchFiles
INFO: Started server process [42980]
INFO: Waiting for application startup.
INFO: Application startup complete.
```

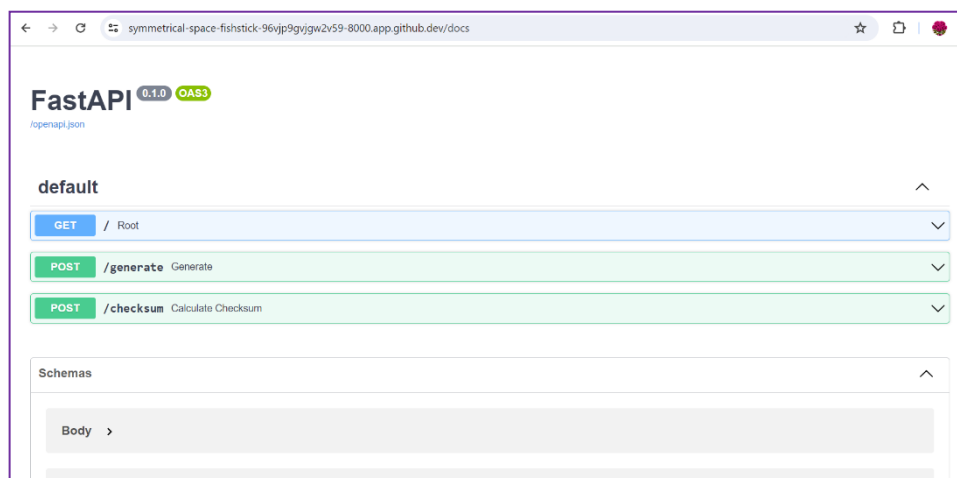


Instruction to submit in TDLC portal.

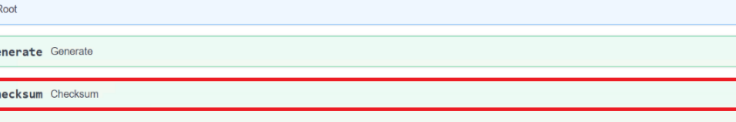
To get credit of completion of this activity, you must take screenshot of above output page from your Codespaces environment (**Your name must be visible**) and submit it in the Activity submission option enabled in TDLC university portal.

Now click on **Try it out** button on the webpage.

You will be redirected to new API endpoint, verify the API with name checksum.



Expand the Checksum endpoint and click on Try it out.



The screenshot shows the Swagger UI for the `/checksum` endpoint. The endpoint is highlighted with a red box. The description states: "Calculate the checksum of the provided text. Example POST request body: { 'content': 'Hello, world!' } :param text: The text for which the checksum needs to be calculated. :type text: Text :return: A dictionary containing the checksum. :rtype: dict". The "Parameters" section is empty. The "Request body" is required and set to `application/json`. An example value is shown as `{ "content": "string" }`. A red box highlights the "Try it out" button.

In the Request Body you can replace and string message to generate a token.

Request body required

application/json

```
{
  "content": "Jaya"
}
```

Click on Execute and check the token for the text you enter.

```
Curl
curl -X 'POST' \
  'http://localhost:8000/checksum' \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
    "content": "Jaya"
  }'
```

Request URL

```
http://localhost:8000/checksum
```

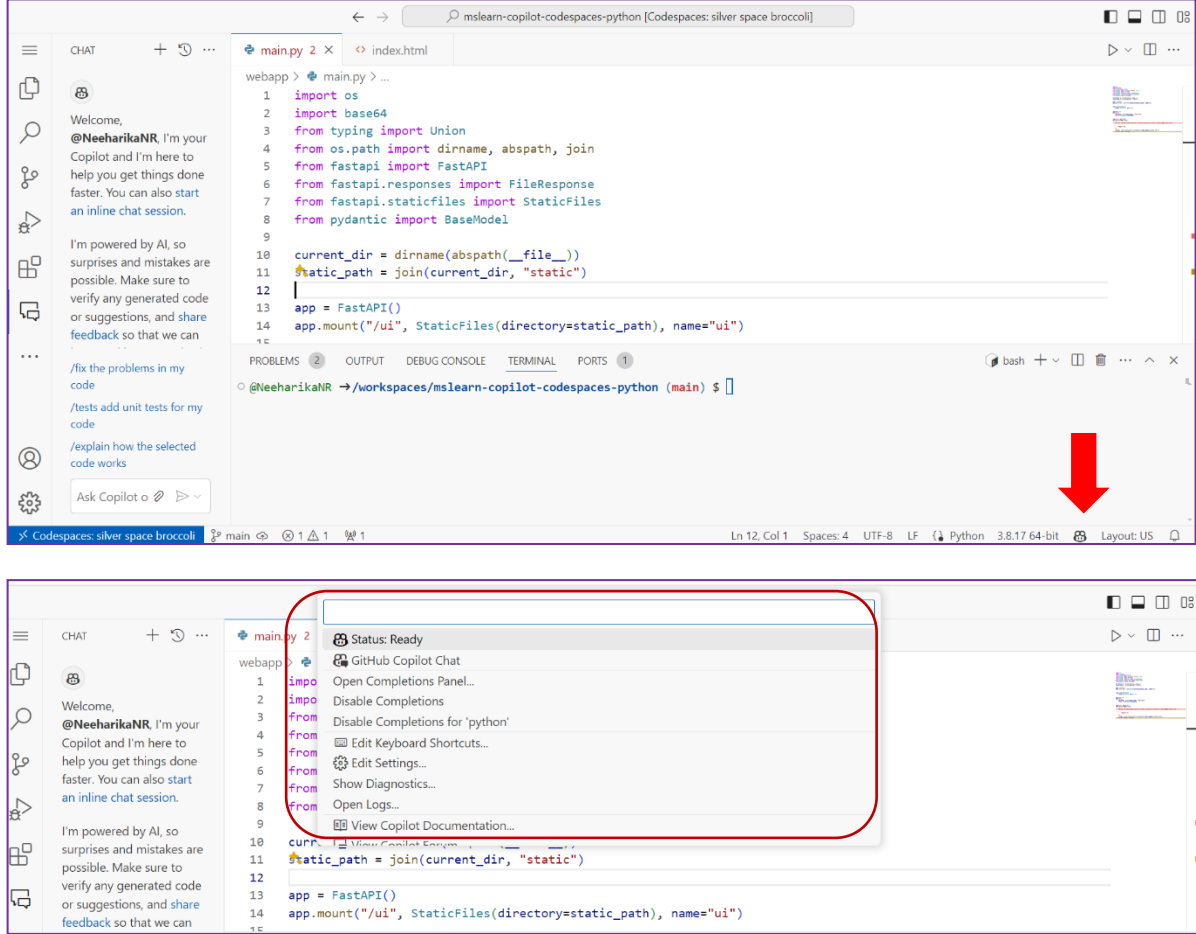
Server response

Code	Details
200	<div><p>Response body</p><pre>{ "checksum": "d3f464b84d4094858ca9e0a77bf7cb8" }</pre><p>Download</p></div> <div><p>Response headers</p><pre>content-length: 47 content-type: application/json date: Tue, 30 Apr 2024 12:18:32 GMT server: uvicorn</pre></div>

Appendix

You can check GitHub Copilot by clicking on copilot symbol on right-bottom side of the screen.

GitHub Copilot



NOTE: If you want to edit Keyboard Shortcuts or any other settings related to GitHub Copilot, you can do it through second option.

Few Shortcuts:

- Ctrl + Enter key: Show combined suggestions.
- Alt +] key: See next suggestion
- Alt + [key: See previous suggestion.
- Ctrl + I key: Suggest Terminal Command
- @workspace: Copilot will use the entire project as context
- @terminal: context about the integrated terminal shell and its contents

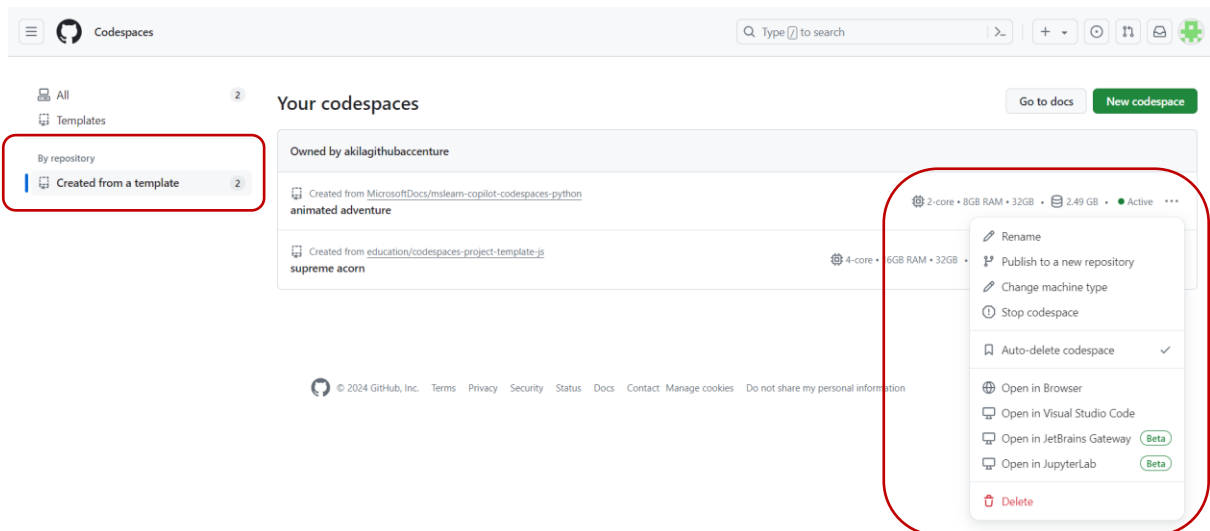
@vscode: about commands and features in the VS Code editor

Clean up activity:

Once the above activity is completed successfully and necessary screenshot taken from the output page, you need to delete the Codespaces environment created in your account. To delete Codespaces

1. visit <https://github.com/codespaces> URL and Click on **Created from a template** option present in the left side pane.
2. Click on **show more actions for codespace** (three dots at the end of codespace environment) and select **Delete** to clean up the environment.

GitHub Copilot



Learning Outcome:

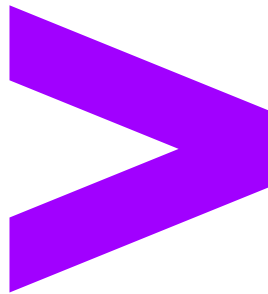
By the end of this activity, Mark can generate prompts and get code assistant for different scenarios from GitHub Copilot. He can use GitHub Copilot to improve his existing project, increase his productivity, and speed up coding activities in Python.

Troubleshooting Steps:

Adding pydantic model to text

Option 1: In line 20, Add a Comment as “# **Generate a Pydantic model for text**” and Press **Enter**. Review the suggestions provided by GitHub Copilot and to accept those suggestions press “**Tab**”.

Option 2: Use GitHub Copilot Chat.



Copyright © 2024 Accenture
All rights reserved.
Accenture and its logo are trademarks of Accenture.