GitHub Copilot, Flask, OpenAPI and Swagger Editor

Prerequisites

* Swagger Editor: <https://editor.swagger.io>
  + Just make sure you can access this tool, this is a workflow step in the process that converts APIFlask generated JSON to YAML. Yaml is consumed by the OpenAPI VS Code extensions
* Python/Pip
  + $ pip install apiflask
* API Flask
  + Website: <https://apiflask.com>
  + Documentation: <https://apiflask.com/docs>
  + PyPI Releases: <https://pypi.python.org/pypi/APIFlask>

API Flask is a lightweight Python web API framework based on [Flask](https://github.com/pallets/flask) and [marshmallow-code](https://github.com/marshmallow-code) projects. It's easy to use, highly customizable, ORM/ODM-agnostic, and 100% compatible with the Flask ecosystem.

* + With APIFlask, you will have:
    - More sugars for view function (@app.input(), @app.output(), @app.get(), @app.post() and more)
    - Automatic request validation and deserialization
    - Automatic response formatting and serialization
    - Automatic [OpenAPI Specification](https://github.com/OAI/OpenAPI-Specification) (OAS, formerly Swagger Specification) document generation
    - Automatic interactive API documentation
    - API authentication support (with [Flask-HTTPAuth](https://github.com/miguelgrinberg/flask-httpauth))
    - Automatic JSON response for HTTP errors
  + h
* VSCode: <https://code.visualstudio.com/download>
* Install these VS Code extensions as they will be used in this lab:  
  + Python VS Code ExtensionA screenshot of a computer

    Description automatically generated
  + OpenAPI VS Code Extension

A screenshot of a computer

Description automatically generated

## Requirements

* Python 3.8+
* Flask 2.0+

## Prompting – Getting Started

1. Create a new file called **app.py** in VS Code
2. In your **app.py** file, add a prompt string to generate an API Flask app and create a “GET” method. There is an example provided below, but feel free to experiment with your own

|  |
| --- |
| # Generate APIFlask app  # Add a GET method called “SayHello” that accepts input as a query string using a string as a parameter for the name to say a greeting |

1. Press **CTRL – [Return].** This should open a Copilot editor to the right that provides Github Copilot suggestions for app.py.  
     
   A screenshot of a computer screen

   Description automatically generated

1. Select the code solution of your choice from the list of suggestions by clicking “**Accept Solution**”. This should add the suggested code to your app.py file.
2. After you have used Copilot to generate the Imports and application Object, prompt it for Schema Objects. Type **“# Schema for Input**” followed by “**Return**” and Copilot should suggest a class for you. Hit “**Tab**” to accept the suggestion and autocomplete the schema.

|  |
| --- |
| # Schema for input  # Schema for output |

1. From here it will generate the Input and Output Schema objects and look something like this:

A computer screen with text

Description automatically generated

1. Add another prompt for an entry point (below) and repeat the process of clicking “**Return**” > “**Tab**” to autocomplete

|  |
| --- |
| # GET method – Use Query string parameters |

1. You should have something that resembles:  
     
   A screen shot of a computer program

   Description automatically generated
2. This should match your virtual environment. Save your progress: **[CTRL] + s or [CMD] + s**

## Testing – Getting Started

1. Make sure your vnev is set up, I use conda:  
     
   conda info –env

conda activate snow-3.8

1. Make sure you get the right Python Interpreter in your VS Code session.
2. Click VS-Code menu
3. View | Command Palette, then enter Python: Select Interpreter  
     
   A screenshot of a computer program

   Description automatically generated
4. I open a terminal and run the code, you can optionally create a Launch Profile for Flask in VS Code
5. Click the Debug Icon  
      
    
6. Choose **“Add Configuration”**

A screenshot of a computer program

Description automatically generated

1. Add this configuration to your **.vscode/launch.json**

|  |
| --- |
| {  "name": "Python: Flask",  "type": "python",  "request": "launch",  "module": "flask",  "env": {  "FLASK\_APP": "app.py",  "FLASK\_DEBUG": "1"  },  "args": [  "run",  "--no-debugger",  "--no-reload"  ],  "jinja": true,  "justMyCode": true  } |

1. Click on the file: **app.py**
2. To enter a breakpoint, click in the margin, left of the line numbers. It will look like this:  
     
   A screen shot of a computer code

   Description automatically generated
3. From the VS Code Menu: **Run | | Start Debugging**
4. Bring up a browser: <http://127.0.0.1:5000/SayHello?name=John>
5. You will see an error handler response. Why?  
     
   A screenshot of a computer

   Description automatically generated
6. Lets take advantage of OpenAPI and use the executable Documentation
7. [APIFlask 0.1.0 - Swagger UI](http://127.0.0.1:5000/docs)
8. The problem is evident, the data is in the request body (which you cant do with a GET), not being passed as query parameters  
     
     
   A screenshot of a computer

   Description automatically generated

## Debugging – Fixing Generated Code

1. We will need to modify the generated code -- <https://apiflask.com/request/>  
     
   A screenshot of a computer program

   Description automatically generated
2. Change from the body to query string  
     
   A screen shot of a computer code

   Description automatically generated
3. This works  
     
   A screenshot of a computer

   Description automatically generated
4. Add a prompt to add one more method to add 2 numbers and return the sum

|  |
| --- |
| # GET method – Add two query string parameters and return the sum |

1. This is great, CoPilot learned from the last method and from appearances did this method properly  
     
     
   A screenshot of a computer code

   Description automatically generated

1. Let’s go back to our OpenAPI documentation to test this: <http://127.0.0.1:5000/docs>  
     
   A white and blue line

   Description automatically generated
2. We see this works  
     
   A screenshot of a computer

   Description automatically generated
3. So far, this has been a Code First API approach, now we will pull our design documentation into VS Code
4. Click the Blue “/openapi.json” link

A close up of a logo

Description automatically generated

1. Copy the JSON into the Clipboard
2. No add the Swagger URL to your browser: <https://editor.swagger.io>
3. From the menu: **File | clear editor**
4. Next paste the JSON into the editor, then you will see:  
     
   A screenshot of a computer

   Description automatically generated
5. Select OK. You will see:  
     
   A screenshot of a computer

   Description automatically generated
6. The VS Code OpenAPI extension works from the command palette, and all commands are prefixed with: OpenAPI  
     
   A screenshot of a computer

   Description automatically generated
7. Paste in our API specification in JSON and save it  
     
   A screenshot of a computer

   Description automatically generated
8. Next run the Swagger Interface:
9. Click: **View | Command Palette**
10. Enter: **OpenAPI**   
      
    A screenshot of a computer program

    Description automatically generated
11. To make the Swagger/OpenAPI editor work, I ran the Flask API in another terminal, set your default directory where your python/flask app is located:

A screenshot of a computer

Description automatically generated