Tutorial Report

Yash Karandikar
ykarandikar@oxy.edu
Occidental College

The idea of the tutorial [1] [2] [3] is to provide technical guidance for development in Flask and to support users who are unfamiliar with Visual Studio Code. [1] The tutorial covers configuration of a project environment through Python's virtual environment tool, generation of a minimally rendered page through Jinja templates and setting up good code practices with requirements.txt. It also walks users through Visual Studio Code's built-in debugger to streamline the process. The final deliverable is a website served on localhost which has an about, contact, and home page. Overall, the tutorial aims to get users situated in a Visual Studio Code development environment so that they can build upon their "Hello World" Flask application.

1 Methods

1.1 Environment

A virtual environment (using Python's venv tool) is created and Flask is obtained using the virtual environment's pip install. Since the tutorial is meant to use Visual Studio Code as the primary code editor, the virtual environment is sourced every time Visual Studio Code is launched.

1.2 Creating a minimal app

A basic Flask file (app.py) is created and filled with a "Hello World" content for the root router. The file is then run on the development server within the shell that is integrated into Visual Studio Code.

After the initial test run, the "Hello World" content is replaced with an endpoint that allows for user name input (e.g., "/hello/<name>" rather than just "/hello/") to provide user interactivity. In order to ensure that malicious content (such as a cross-site scripting attack, or XSS) is not inserted from the endpoint, the endpoint input is checked with regular expressions and cleaned before HTML is rendered.

[1] Visual Studio Code tutorial
[2] Overleaf source
[3] GitHub repository

1.3 Debugging tools

Visual Studio Code's debugger is presented as a way to practice debugging the current Flask app. A launch.json configuration file is created so that Flask can identify which Python file should be used at startup. The "Run and Debug" command, as well as changing variable contents in the "Debug Console," are explained to the user. The tool "Peek Definition" is also introduced to explain how Visual Studio Code can neatly reference the instantiation of a specific variable or function in its original file.

1.4 Templates

Templates are introduced as a streamlined way to structure Flask web pages and protect the Flask app application against XSS attacks. These pre-rendered Jinja templates replace the previous app.py with a much more concise layout. More templates are then generated for various pages (such as contact information, about, and so on) and placed into the root level app.py. Other add-ons, such as CSS and static file serving (like JSON), are also implemented in the sample app.

1.5 Reproducibility

A requirements.txt file is generated using `pip freeze`, which allows others to pip install the exact same versions of the same libraries on their machine.

2 Evaluation Metrics

2.1 Correctness and UX

The intention of the Flask tutorial is to set up a basic web page (using Flask) that routes to templates containing about, contact, and home pages. This goal is ultimately met, since the Jinja templates provided in the tutorial lead to the construction of the web page. The basic home page (with endpoint "/") serves the main page as expected, and alternate endpoints ("/about/" and "/contact/") lead to the appropriate HTML content as well. Even hitting the endpoint "/home/<name>" - where <name> is some user-provided

argument - prints the HTML content as expected. All of these components of the user experience occurs in milliseconds, meeting the time requirement.

2.2 Reliability

The Flask application is able to execute in a variety of ways. While development is ongoing, Visual Studio Code's "Run and Debug" feature robustly runs the application in a way that makes it easy to visualize variable contents. After constructing the Flask application, it can be run from the integrated terminal within Visual Studio Code with `python -m flask run`. It's also possible to run it from a separate terminal such as Git Bash by exporting the Flask app (`export set FLASK_APP=hello_app.webapp`) and then performing the same Python command for running the Flask app.

3 Results and Discussion

Overall, I found this tutorial to be an insightful and helpful look into Flask web development. Though I had heard of many of the individual components (such as Python's venv and `pip freeze` for reproducibility), this tutorial was the first time that I actively used them in my own work. I found the concept of virtual environments in Python to be very clean and useful - particularly when it comes to modularly developing my work to avoid package version conflicts later on - and I will continue to use venv in future Python projects. Similarly, I now know how to use pip to generate requirements.txt to help others reconstruct the project in their own virtual environments. These concepts are big-picture Python development techniques that I can abstract to other programming projects as well.

At the more granular code level, since the tutorial featured Visual Studio Code, I also learned how to use Visual Studio Code to its full potential. I had previously used it in various classes (such as Computer Organization and Algorithms Analysis), but I was only considering its use as a code editor (in the same style as Sublime or Notepad++). As part of its structure, the tutorial demonstrated various Visual Studio Code features, such as the built-in debugger, the debug console, and integrated terminals. In addition to standard print debugging, I now know how to use the built-in debugger to set breakpoints and run the code step by step to see exactly which line the program breaks. I also used the debug console to print the contents of variables, and Visual Studio Code enabled me to change these contents dynamically at runtime for easier debugging. Even though the goal of the tutorial was to learn about Flask, I feel that incorporating these built-in concepts made the process easier, and I will use these in future development for comps and beyond.

In addition to good practice at the granular development level, I also learned about big-picture code architecture concerns like cross-site scripting. While it's important to have code that's optimized and runs in efficient time, it is also a crucial security concern (particularly for web applications which center interactivity and user input!) to ensure that inputs are checked for safety. Cross-site scripting attacks (XSS) are one such input (embedded in the endpoint which the user hits) which can leak data if implementation does not follow good practice. Through this tutorial, I learned to be cognizant of the dangers of potential XSS attacks and actively strive to use templates or other prebuilt tools to avoid giving attackers such an opportunity. Furthermore, I consulted the current literature to see what good practices are to protect applications against XSS attacks at the implementation level. Gupta and Gupta suggest performing static code analysis to observe potential loopholes in the source code. [2] Rodriguez et al. discuss ways to prevent the execution of untrusted data (which could potentially inject JavaScript code in an unauthorized way) through denying unnecessary permissions and checking data values in code. [3] Considering these suggestions as I move from research into active implementation can help to make my application more secure and robust before it is presented to the end user.

References

[1] *Flask Tutorial in Visual Studio Code*. `https://code.visualstudio.com/docs/python/tutorial-flask`. Online; accessed 28 April 2023. 2023.

[2] Shashank Gupta, B.B. Gupta. *Cross-Site Scripting (XSS) attacks and defense mechanisms: classification and state-of-the-art*. `https://link.springer.com/article/10.1007/s13198-015-0376-0`. Online; accessed 30 April 2023. 2015.

[3] German Rodriguez, Jenny Torres, Eduardo Benavides. *Cross-Site Scripting (XSS) Attacks And Mitigation: A Survey*. `https://www.researchgate.net/profile/German-Rodriguez-22/publication/336995980_Cross-Site_Scripting_XSS_Attacks_And_Mitigation_A_Survey/links/5dcc4e31a6fdcc7e137e2eb9/Cross-Site-Scripting-XSS-Attacks-And-Mitigation-A-Survey.pdf`. Online; accessed 30 April 2023. 2019.