Open-Source Report

Proof of knowing your stuff in CSE312

Guidelines

Provided below is a template you must use to write your reports for your project.

Here are some things to note when working on your report, specifically about the **General Information & Licensing** section for each technology.

- **Code Repository**: Please link the code and not the documentation. If you'd like to refer to the documentation in the **Magic** section, you're more than welcome to, but we need to see the code you're referring to as well.
- **License Type**: Three letter acronym is fine.
- **License Description**: No need for the entire license here, just what separates it from the rest.
- **License Restrictions**: What can you *not* do as a result of using this technology in your project? Some licenses prevent you from using the project for commercial use, for example.

Also, feel free to extend the cell of any section if you feel you need more room.

If there's anything we can clarify, please don't hesitate to reach out! You can reach us using the methods outlined on the course website or see us during our office hours.

[Flask SocketIO]

General Information & Licensing

Code Repository	Flask SocketIO: https://github.com/miguelgrinberg/Flask-SocketIO/blob/main/src/flask_socketio/__init__.py Python Websocket: https://github.com/python/cpython/blob/3.11/Lib/multiprocessing/managers.py
License Type	MIT
License Description	• This license permits the use of the software and related documentation files to anyone that chooses to utilize it • Outlines this software is free of charge to everyone • The only part related to this is the copyright notice that outlines that the author is not responsible for any claims made against the code

License Restrictions	• There are no restrictions to this license

Magic

Chain of calls:

- The chain of calls starts off from the function call of socketio.run. This function (and the following references in this bullet point) definition can be found starting on line 553 (run() function)of the __init__.py file of the “Flask SocketIO” github link above. Within this function, if code is functioning properly, it will hit the elif case on line 686 (line 686). This check is the library's way of determining if “gevent” is the type of asynchronous model that is being used. If this is the case, then the code creates an instance of a WSGIServer with the handler_class being WebSocketHandler. This is ensuring that the server created is handling requests as websockets. Once the code reaches past this point, it then calls the serve_forever() function on line 715 (line 715). The call to serve_forever() then leads us to the second bullet point.
- In order to find the definition of the previous server_forever() call, we must look in a different library. This library is the python “mulitprocessing” library. The corresponding code can be found at the “Python Websocket” link provided above. The exact function definitions referenced from this point on are found in the managers.py file of the library. The serve_forever() function can be found on line 164 (line 164) in the managers.py file. Within this function, it calls a subsequent function named accepter() (line 186). This subsequent call is necessary to start the server up and run it forever. In accepter(), it is a continuous while loop that forever runs the server. Now, back in serve_forever() there is functionality for if there is a keyboard interrupt or or exit, the server will stop running.
- Another key function in the manager.py file is the serve_client() function. This function is defined on line 238 (line 238) . This function has similar functionality to our HW3. Within this function, there is a constant while() loop that runs as long as stop_event is not set. Within this while loop, there is a recv() call that receives the bytes of the request. Then, once the bytes are received, the function calls getattr() (line 270). The functionality of getattr() allows the program to extract the value of a specific attribute of a class. Both the class and attribute are defined in the parameter. Then, if no exceptions are raised, the program then packs the response in the msg variable. If no further exceptions are raised, the program will send msg back to the open connection that initialized the request(line 308).