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/TCP connection]

General Information & Licensing

Code Repository	https://github.com/Ai-Jesse/Blue-Jesse
License Type	MIT
License Description	• Any user can obtain this software free of charge • Any user is allowed to copy, modify, and distribute • Users are allowed to sell the software
License Restrictions	• The software is provided "as is" without warranty • The authors or copyright holders can't be liable for any claim, damages, or other liability

Magic

- How does this technology do what it does? Please explain this in detail, starting from after the TCP socket is created
    - For the TCP connection by Flask, the program executes the method app.**run**()[1] which is a built-in function of class Flask. It will take the host and port we input as variables and create a server with the variables. But if we didn't input the host and port, this method will set the host as 127.0.0.1 and port as 5000.
    - This method will first check the method run by the command or not, so it will not start another server. And then check some necessary files(dotenv) are load.
    - Then it will check every variables that we input or not, and set them the specific value. Then display some information in terminal. After all of these, the method will import another method **run_simple**[2] from werkzeug.
    - The method run_simple will receive the variables pass by the method run. Then it will check the validate of the port, exist of static files,usage of debugger and the server run from reloader or not.
    - If the server does not run from the reloader, it will prepare a socket for use by the server and reloader. The socket is marked inheritable so that it can be kept across reloads instead of breaking connections. Catch errors during bind and show simpler error messages. For "address already in use", show instructions for resolving the issue, with special instructions for macOS. https://github.com/pallets/werkzeug/blob/3ead75c971bbe8681ec9fe644d23cae600896589/src/werkzeug/serving.py#L880-L891(It's the link of this comment)
    - The method **prepare_socket**[3] will call the method **select_address_family**[4] to choose a address family for the socket. **select_address_family** will takes the host and port as variables, and select a address family from "AF_INET4", "AF_INET6" and "AF_UNIX" depend on the host and port.
    - Then **prepare_socket** select the server address by calling the method **get_sockaddr**[5] which will return a fully qualified socket address. After that, the socket will be create by calling socket.socket
    - Next, **prepare_socket** will check the socket file is not exist, otherwise, it will be remove. Then the socket will bind to the server address and listen the incoming connections. Finally the socket will be return to the method **run_simple.**
    - The method **run_simple** then create a WSGI server by the method **make_server**. Depending on the threaded and processes, it will creadte a ThreadedWSGIServer, ForkingWSGIServer, or BaseWSGIServer.
    - After all, **run_simple** will check the Flask application using reloader or not. If yes, then it will import the ._reloader and use the method

run_with_reloader[7] to host the server forever until it shut down. If no, it will simply run the method **serve_forever** [8] to host the server until it shut down.

- Where is the specific code that does what you use the tech for? You ***must*** provide a link to the specific file in the repository for your tech with a line number or number range.
    - [1]run: https://github.com/pallets/flask/blob/main/src/flask/app.py#L1067
    - [2]run_simple: https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L907
    - [3]prepare_socket: https://github.com/pallets/werkzeug/blob/3ead75c971bbe8681ec9fe644d23cae600896589/src/werkzeug/serving.py#L879
    - [4]select_address_family: https://github.com/pallets/werkzeug/blob/3ead75c971bbe8681ec9fe644d23cae600896589/src/werkzeug/serving.py#L607
    - [5]get_sockaddr: https://github.com/pallets/werkzeug/blob/3ead75c971bbe8681ec9fe644d23cae600896589/src/werkzeug/serving.py#L617
    - [6] make_server: https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L853
    - [7] run_with_reloader https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/_reloader.py#L417