Open-Source Report

Proof of knowing your stuff in CSE312

[Flask -TCP Connection]

General Information & Licensing

Code Repository	https://github.com/pallets/flask
License Type	BSD 3-Clause License
License Description	This license is similar to the BSD 2-Clause License which contains two clauses. The first clause is that redistribution of the source code must contain a specific copyright notice, a list of conditions and a specific disclaimer. The second clause requires that redistribution that is in binary form must include the same three items mentioned above in the documentation. There is a third clause that comes in with the BSD 3-Clause License which the framework Flask is under. This clause is that neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. The permissions that are allowed with Flask under this license are that it is approved for commercial and/or private use, and is able to be modified and/or distributed. There are limitations with liability and warranty. Essentially, the requirements of this license when copying, modifying or distributing any code is that the full text of the license must be included, along with the original copyright notice (which is actually included in the license). The license must include the year and owner in the form "Copyright (c) $YEAR $OWNER, All rights reserved".
License Restrictions	The BSD 3-Clause License has very minimal restrictions, which is what makes it easier to use and understand since it can be fairly used commercially and privately. A restriction that is does include concerns using the names of contributors to the repository for endorsement when advertising without permission. This is the third clause that was mentioned above, where people using code under the BSD 3-Clause license are unable to include the name of the copyright holder and/or the contributors to endorse any work that was produced from using the open source code under this license. For example, a project created from using Flask cannot be endorsed by the contributors of Flask without special permission.

Magic

Flask is a framework that is widely used in the development of web applications. Flask is implemented using Werkzeug (a comprehensive WSGI library) and Jinja2 (a templating engine). It is simple to use when compared to other frameworks such as Django, and it provides programmers with back-end components such as tools, libraries and functionalities that are able to efficiently host web servers responding to HTTP requests.

We require this technology in our project because TCP uses a three way handshake to establish a reliable connection. As soon as this connection is created, we can exchange messages between the client and the server to implement our Battleship clone. An advantage of TCP is that it uses extra overhead to ensure that the data being exchanges is accurate, therefore, the players of our game can trust that the coordinates that they are choosing are correctly handled on the server side while maintaining order and that their login details are exchanged without a great risk of being lost.

1. Our program starts in app.py where the following line below calls the run() function:
   if __name__=="__main__":
   app.run(debug=True,host='0.0.0.0')
   As mentioned in the documentation, the run() function will run the application on a local development server.
   This function appears in app.py on line 1067:
https://github.com/pallets/flask/blob/066a35dd322f689ec07d7c0e82b19eacadac3c6b/src/flask/app.py#L1067

   However, to begin, we assign the variable app to a flask object implements a WSGI application and acts as the central object, which is accomplished by the line:
   app = Flask(__name__).
   In the provided repository for Flask, this can be found in the app.py file on line 92:
https://github.com/pallets/flask/blob/d0bf462866289ad8bfe29b6e4e1e0f531003ab34/src/flask/app.py#LL92C1-L92C1

2. When we call the run() function on the flask object, a parameter that is included is options. This parameter, as mentioned in the comments are the "the options to be forwarded to the underlying Werkzeug server". Because of Werkzeug, we are able to import a function run_simple: from werkzeug.serving import run_simple. This function is available in Werkzeug library's serving.py on line 945:
https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L907

   This function calls another function which is make_server() which can be found in the same serving.py on line 891:
https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#LL1037C16-L1037C16

   This function (as described in the comments) will create an appropriate WSGI server instance.

3. The make_server() function returns a an instance of BaseWSGIServer(HTTPServer) which can also be found in the same serving.py file mentioned above on line 689:
https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L651

   This is used by the make_server() function to create an instance of a server, as mentioned in the comments of the documentation. To create this instance of a

BaseWSGIServer, we have to pass it an argument that is (class) HTTPServer which can be found in python's server.py on line 129:
https://github.com/python/cpython/blob/5ea052bb0c8fa76867751046c89f69db5661ed4f/Lib/http/server.py#L129

4. We are now at the class HTTPServer(socketserver.TCPServer), where there is a TCP connection made in a similar manner to what has been implemented in our homework.

   On line 139, the line socketserver.TCPServer.server_bind(self) accomplishes exactly this, using the socketserver library. The host and the port are created in lines 137: host, port = self.server_address[:2].

   For the sake of comparison, the line in the homework that achieves the same result by using the socketserver library is:
   socketserver.ThreadingTCPServer((host, port), MyTCPHandler)