Open-Source Report

Proof of knowing your stuff in CSE312

[Flask - WebSockets]

General Information & Licensing

Code Repository	There are three different types of libraries that we need to reference for our implementation of websockets in this project. 1. The first is Flask-SocketIO, which can be found here: https://github.com/miguelgrinberg/Flask-SocketIO 2. The second is gevent-websocket, which can be found here: https://github.com/jgelens/gevent-websocket 3. The last is socket.io, we can be found here: https://github.com/socketio/socket.io
License Type	1. The license type of Flask-SocketIO is MIT License 2. The license type of gevent-websock is Apache License Version 2.0 3. The license type of socket.io is MIT License
License Description	MIT License: This is known as one of the simplest open source license agreements. The MIT License will allow permission for users to reuse code under this license for the use of any purpose, this is true in some cases where the code involved is included in a portion of proprietary software. The requirement of programmer that may want to use and modify code under this license is that they include the original copy of the MIT license in their description. A benefit of this license is that it can be used by developers who intend to create programs for personal or proprietary use, and that they are able to modify and distribute as they wish. This is assuming that developers include the full copyright and license text. Apache 2.0 License: This license is very similar to the MIT License described above, since it allows developers to modify and distribute code that is under the Apache 2.0 license for commercial and private use. Additionally, it allows for sublicensing, meaning that it is possible to extend the license to the software. It includes the ability to place a warranty on the software that is licensed and it can use patent claims of contributors to the code. To use code under this license, developers must include the copyright statement and license text (which is also true in the case of the MIT License). Additionally, they must also include a notice file with attribution notes included and state any significant changes that were made to the software.

License Restrictions	MIT License: A restriction that this license has is that it cannot hold the owner of the software/license liable for damages, which is a good restriction to have. Generally, it is known that the MIT License is the least restrictive amongst the other licenses. Apache 2.0 License: Similarly to the MIT License, a restriction of this license is liability. Additionally, it is forbidden to use any of the contributors' names, trademarks or logos.

Magic

It is necessary for our project to establish a websocket connection so that multiple players are able to create lobbies for playing a game of battleship and join open lobbies to start playing. We are able to establish a websocket connection and parse frames sent between the client and server by using several libraries that were mentioned above. The advantage that this gives us is the ability to use real-time communication so we are able to determine which player was able to sink the opponent's ship first.

1. We can begin in our project code where we create a Flask-SocketIO server for the purpose of handling websocket connections through this line: socketio=SocketIO(app=app).
   This class is found on line 54 of the Flask_SocketIO repository's __init__.py file:
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L54

   In this SocketIO class, on line 695 the websocket is equal to True: websocket = True
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L695

   On line 700, there is a conditional which must occur since the websocket is now equal to True:
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L700

   Within this conditional, the WSGIServer is set:
   self.wsgi_server = pywsgi.WSGIServer((host, port), app, log=log,**kwargs).
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#LL701C25-L701C25

   There is a function included, named run_server() which runs the server forever using serve_forever() on self.wsgi_server
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L714

2. The gevent-websocket library comes in at this point, where it is used to parse the frame data received from the
   websocket frames. This can be found on line 190 of the gevent-websocket library's websocket.py file:
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L190

In line 209 of the same file, it reads the headers of the payload and gets the payload:
payload = self.raw_read(header.length)
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#LL209C26-L209C26

Once it knows the payload is the correct length after doing a check on line 216, it can begin to unmask it on line 220.
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L216

https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L220

Now that it is unmasked, it can begin to read the message that was sent through the payload on line 233 using the function read_message()
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L233

This function calls the function read_frame() within a while True loop which resembles the way that reading websockets were implemented in the homework.
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L190

If the finbit indicates that it is not the last message, it will keep reading a new message from line 247:
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L247

3. To send a websocket frame, the send_frame() function in the gevent-websocket library's websocket.py file is used:
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L315

The header of the message is created on line 282:
https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#L328

Since the message is given as a parameter to the function, it can be encoded and combined with the header and sent on
line 331:

https://github.com/sinank/gevent-websocket/blob/5020669b0439fd49f054830c51b1aa1602b7d086/geventwebsocket/websocket.py#LL331C1-L332C1

This is very similar to the homework code because it receives payload data in a loop, waits until the finbit indicates that it is the last message that is being sent and encodes a response in bytes to send back to the client.