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	https://github.com/miguelgrinberg/Flask-SocketIO
License Type	MIT license
License Description	It is an open-source software license used by the Flask-SocketIO project. The license allows for free use, modification, and distribution of the software, with limited conditions.
License Restrictions	1. Redistributions of the software, in source code or binary form, must retain the copyright notice, the permission notice, and the disclaimer. 2. The name of the copyright holder and the names of its contributors cannot be used to endorse or promote products derived from this software without specific prior written permission.

Magic

Brief description:

socketio.run() is a method in the Flask-SocketIO extension that starts the WebSocket server. It handles incoming WebSocket connection requests and configures server behavior.
Once a WebSocket connection is established, Flask-SocketIO, along with the gevent-websocket library, handles WebSocket frames carrying data between the client and the server. The **geventwebsocket.websocket.WebSocket.receive()** method, invoked upon receipt of a WebSocket frame, calls **geventwebsocket.websocket.WebSocket.read_frame()** to read the frame data. This method further calls **geventwebsocket.Header.decode_header()** to decode the frame header, helping interpret the frame type and facilitate real-time communication.

The whole chain from framework code to HW code:

1. We call **socketio.run(app, debug=True, host='0.0.0.0', port=8000):** https://github.com/fhuang566/CSE-312/blob/f7ee72cefecfc44074431ae06fc4a5146662efd3/code/app.py - L302
    - Flask-SocketIO's run method is responsible for starting the WebSocket server.

2. Flask-SocketIO's **socketio.run()** method calls SocketIO's run method: https://github.com/miguelgrinberg/Flask-SocketIO/blob/288119a11664d887c47522509d010f502ff742e8/src/flask_socketio/__init__.py - L553
    - The **run()** method in SocketIO wraps Flask's **app.run()** method and starts a server instance based on the given configuration.

3. SocketIO's run method calls Python's **gevent** library (depending on the async_mode configuration): https://github.com/miguelgrinberg/Flask-SocketIO/blob/288119a11664d887c47522509d010f502ff742e8/src/flask_socketio/__init__.py - L689
    - The gevent or eventlet library is responsible for managing the WebSocket connections in an asynchronous manner.

4. If gevent's WebSocket is used, the **WSGIServer** is created with **WebSocketHandler** as **handler_class**: https://github.com/gevent/gevent/blob/1e412d35526183b26c1abf2eb658cbef661f5f70/src/gevent/pywsgi.py - L1399
    - **WebSocketHandler** handles the WebSocket handshake and upgrades the HTTP connection to a WebSocket connection.

5. WebSocketHandler's **upgrade_connection** method performs the WebSocket handshake:https://github.com/jgelens/gevent-websocket/blob/35cba7aa107f183acfc954b713718b3630509e8a/geventwebsocket/handler.py - LL224C16-L224C16
    - This method sends the '**Upgrade**' headers and the **Sec-WebSocket-Accept** key in the response to the client's handshake request.

6. Once the WebSocket connection has been successfully established, the **geventwebsocket.handler.WebSocketHandler.run_application()** method is invoked: https://github.com/jgelens/gevent-websocket/blob/35cba7aa107f183acfc954b713718b3630509e8a/geventwebsocket/handler.py - L36
    - This method transforms the WSGI application into a WebSocket-based application.
7. The **geventwebsocket.websocket.WebSocket.receive()** method is then called to listen for incoming messages: https://github.com/jgelens/gevent-websocket/blob/35cba7aa107f183acfc954b713718b3630509e8a/geventwebsocket/websocket.py - L309
    - **receive()** waits for data to be sent from the client side, and once data is received, it is decoded.

8. Inside the **receive()** method, **geventwebsocket.websocket.WebSocket.read_frame()** is invoked to read WebSocket frames: https://github.com/jgelens/gevent-websocket/blob/35cba7aa107f183acfc954b713718b3630509e8a/geventwebsocket/websocket.py - L193
    - **read_frame()** reads one frame at a time, if a frame is available.

9. **geventwebsocket.websocket.WebSocket.read_frame()** then calls **geventwebsocket.Header.decode_header()** to parse the WebSocket frame header:https://github.com/jgelens/gevent-websocket/blob/35cba7aa107f183acfc954b713718b3630509e8a/geventwebsocket/websocket.py - L487
    - **decode_header()** decodes the header of the WebSocket frame, which includes information such as the final bit, opcode, mask, length, and other flags.

Stack trace:

- socketio.run(app, debug=True, host='0.0.0.0', port=8000)
- flask_socketio.SocketIO.run()
- gevent.pywsgi.WSGIServer.__init__(handler_class=WebSocketHandler)
- geventwebsocket.handler.WebSocketHandler.upgrade_connection()
- geventwebsocket.handler.WebSocketHandler.run_websocket()
- geventwebsocket.websocket.WebSocket.receive()
- geventwebsocket.websocket.WebSocket.read_frame()
- geventwebsocket.websocket.Header.decode_header()