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.

Websockets

General Information & Licensing

Code Repository	https://github.com/ahssanja/CSE_312_Group_Project.git
License Type	MIT License
License Description	● permissive open-source license ● is permissive, it does require users to provide attribution to the original authors of the software ● provides a limited liability protection for users
License Restrictions	● provides a limited liability protection for users, but it is not a guarantee of fitness for any particular purpose or non-infringement ● does not grant users any rights to use the trademarks or logos associated with the software. ● does not include any copyleft provisions, which require users to share their changes or improvements with the community

Magic

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:

- How does this technology do what it does? Please explain this in detail, starting from after the TCP socket is created
- 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.
    - If there is more than one step in the chain of calls *(hint: there will be)*, you must provide links for the entire chain of calls from your code, to the library code that actually accomplishes the task for you.
    - Example: If you use an object of type HttpRequest in your code which contains the headers of the request, you must show exactly how that object parsed the original headers from the TCP socket. This will often involve tracing through multiple libraries and you must show the entire trace through all these libraries with links to all the involved code.

*This section will likely grow beyond the page

This report provides an overview of the WebSocket connection establishment process and frame parsing inexpress and node.js. In our code the use of webSockets provide a full-duplex communication channel over a single TCP connection, allowing real-time, bidirectional data transfer between a client and a server. In our case this bidirectional real-time transferis used for pliers playing tic-tac-Toe in real time and being able to see eachother's moves in real time as well.

WebSocket Connection Establishment

```
const ws = new WebSocket('ws://localhost:8080');
```

In the above code, we make a new WebSocket object and initialize it with the server's URL (ws://localhost:8080). The websocket library handles the underlying connection establishment process.

WebSocket Event Handlers

```
ws.onopen = () => {
ws.send(JSON.stringify({ type: 'create', username: username }));
};

ws.onmessage = (event) => {
// Handle received WebSocket messages
};
```

In the above code, the *onopen* event handler is assigned to handle the WebSocket connection's open event. The code sends a JSON message to the server indicating the type of action ('create') . From here the onmessage event handler is assigned to handle incoming WebSocket messages from the server. It parses the received data, which is expected to be in JSON format, and performs actions based on the message type.

WebSocket Frame Parsing

The WebSocket frames consist of a header and payload where the websocket library handles the parsing of thse WebSocket frames.

How WebSocket Frames Parsing are parsed

```
ws.onmessage = (event) => {
let data = JSON.parse(event.data);
switch (data.type) {
// Handle different message types
}
};
```

In the above code the onmessage event handler receives the WebSocket message as an event object. In this case, the payload is expected to be in JSON format, and JSON.parse is used to parse the payload into a JavaScript object. After parsing the payload, the code performs different actions based on the message type by using a switch statement.

In conclusion we have explored the WebSocket connection establishment process and frame parsing in our code. We have also traced the sequence of calls from our code to the library responsible for establishing WebSocket connections and handling frame parsing which helped design our app.