Open-Source Technology Use Report

Proof of knowing your stuff in CSE312

Guidelines

Provided below is a template you must use to write your report for each of the technologies you use in 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'd like 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.

TCP connections

General Information & Licensing

Code Repository	https://github.com/ahssanja/CSE_312_Group_Project
License Type	BSD 3-Clause "New" or "Revised"
License Description	● Allows unlimited redistribution for any purpose while copyright notices and disclaimers of warranty are maintained ● It can be used for commercial or private purposes ● It can be modified
License Restrictions	● Prohibits others from using copyright holder's name or contributions to promote derived products without written consent ● The license doesn't provide warranty ● The license includes a limitation of liability

Use as many of the sections below as needed, or create more, to explain every function, method, class, or object type you used from this library/framework.

Purpose

The TCP connections our tic tac toe application uses are managed mainly through the Node.js built-in http module and the 'ws' WebSocket library. These modules provide the tools necessary for creating server-side TCP connections, allowing our app to listen for and respond to client requests.

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 for you in the **Purpose** section of this report? Please explain this in detail, starting from after the TCP socket is created. Remember, to be allowed to use a technology in your project, you must be able to know how it works.
- 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 may grow beyond the page for many features.

This report focuses on the management of TCP connections used in our tic tac toe web app, which utilizes the Express.js framework along with the 'http' and 'ws' (WebSocket) library modules. The TCP connections play an important role in enabling communication between the server and clients in our app which in our case allows the application to handle incoming requests and provide appropriate responses such as loading files, displaying content, and having a proper connection handled by our code's backend.

In the provided Node.js application, TCP connections are established through the following code snippet:

```
const server = http.createServer(app);
server.listen(8080, () => {
console.log('Server started on port 8080');
});
```

The code snippet shows us the initialisation of an HTTP server object using the Express application 'app'. The 'http.createServer()' function is utilized to generate a new instance of 'http.Server', which represents an HTTP server capable of accepting client connections. A

thing to be noted here is that the server does not begin accepting connections until the 'server.listen()' function is invoked.

By calling 'server.listen(8080)', the server is configured to listen for incoming TCP connections on port 8080. Once the server starts accepting connections, a 'listening' event is emitted. In response to this event, a callback function is executed, logging the message "Server started on port 8080" to the console.

In addition to the 'http' module, the application also utilizes the 'ws' module, a third-party library for working with WebSockets. The 'ws' module is employed to establish WebSocket connections, utilizing the 'http' module for initial HTTP connections and subsequently upgrading them to WebSocket connections upon client request. The following code snippet showcases the management of WebSocket connections:

```
const wss = new WebSocket.Server({ server });
```

By creating a new instance of the WebSocket server, represented by the 'wss' object, the 'ws' module establishes a connection to the existing HTTP server. This enables the WebSocket server to handle WebSocket requests. Upon receiving a WebSocket request, the 'ws' module facilitates the upgrade of the HTTP connection to a WebSocket connection.

This code creates a new instance of the WebSocket server, represented by the 'wss' object. The WebSocket server is bound to the existing HTTP server, enabling it to handle WebSocket requests. When a WebSocket request is received, the 'ws' module facilitates the upgrade of the HTTP connection to a WebSocket connection.

In conclusion, TCP connections form the foundation for communication between the server and clients in our game. Through the utilization of the built-in 'http' module and the 'ws' WebSocket library, TCP connections are effectively established and managed. This report provided an overview of the relevant code snippets in the application and further explored the underlying mechanisms involved in establishing and handling TCP connections.