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.

Express.js - Parsing HTTP headers

General Information & Licensing

Code Repository	https://github.com/expressjs/express
License Type	MIT License
License Description	● Private use ● Commercial use ● Allows for modification with sublicense ● Free distribution
License Restrictions	● No Warranty ● Liability

Purpose

Express.js handles our http requests and sends our responses. It is a layer on top of Node.Js that is efficient in managing servers and routes. The express application is exported from the express module and handles top-level functions.

In our use, it can determine what HTTP requests were made and retrieve any information from it that we request.

This functionality was used in server.js where:

- Express was imported [Line 1]
    - Syntax → ***const express = require('express')***
- app "element" was created using express() [Line 12]
    - This is a JavaScript Function designed to be passed to Node's HTTP Servers as a callback to handle request
    - App was used on Lines 39-43, 46, 48, and 53 for routing functionality and ensuring the connection was good between server and user
- Used express.static() to route user to homepage [Line 46]

Magic

Within the express repo, its call functions would not hyperlink to their origin or show their references which made locating them to show how data was parsed not possible at the moment.

- Using app.use() we were able to put specified middleware** functions at a specific path, organizing and simplifying our routings. (It differs from app.get() by being able to handle more than just GET requests of a specific path/route)
    - **Definition of middleware → functions that run between a client request and response from server, have access to request object (req), response object (res), and next function.
    - General format of app.use() → ***app.use([path], callback, [callback])***
    - Usage on Lines 45-49 of source file for backend (server.js)
        - Handled routes of products, users, auction, purchase, selling
        - app.use() lets us create route specific middleware, meaning that for a specific HTTP request, we are attaching a stack of operations uniquely for a request, such as a path to user accounts, bidding, etc.
    - (need actual technical on how it does this 0_0)

- App.get() is a method of the Express app, which is used to handle HTTP GET requests that the server receives - server uses this function to determine what to do when a get request at a given route is called.
    - General format of app.get() → ***app.get([path], callback)***
    - Usage of this method can be found on Line 52-55, which specifies the root URL of the server.
    - (go into technical)

- App.listen() is a method used to bind and listen to connections on a specified port and host. In our implementation the port is 8000 → local host
    - General format of app.listen() → ***app.listen([port[, host[, backlog]]][, callback])***
    - Usage can be found on line 57-62, which must be at the end of our backend source file
    - Returns a node http.Server

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

Sources

Express Method	https://expressjs.com/en/5x/api.html