Date: 21 June 2024 Day: Friday

Overview:

Day 13 of the internship focused on advancing skills in handling POST requests in Node.js applications and exploring middleware concepts. This included implementing logic to parse request body data for POST operations, understanding middleware functions for request processing, and strategies for sharing data across middleware and the request lifecycle.

Handling POST Requests:

- Implementing POST Request Handling:
    - Explored the process of handling POST requests in Node.js to facilitate data submission and creation operations.
    - Implemented server-side logic using frameworks like Express.js to capture and process incoming POST data from clients.
- Parsing Request Body Data:
    - Covered techniques for parsing request body data from POST requests:
    - Utilized middleware such as body-parser or built-in functionality to extract form data, JSON payloads, or multipart data from HTTP POST requests.
    - Implemented validation and sanitization routines to ensure data integrity and security before processing.

Middleware Concepts:

- Exploring Middleware Usage:
    - Investigated the role of middleware in Node.js frameworks for intercepting and modifying incoming requests and outgoing responses.
    - Explored built-in and custom middleware functions for tasks such as logging, authentication, error handling, and data validation.
- Implementing Custom Middleware:
    - Developed custom middleware functions in Node.js to extend application functionality and modularize request processing logic.
    - Integrated middleware chains to execute sequential tasks during the request lifecycle, enhancing code reusability and maintainability.

Data Sharing Strategies:

- Discussed strategies for sharing data across middleware functions and throughout the request-response cycle:
- Leveraged req (request) and res (response) objects to store and retrieve data across middleware layers.
- Implemented middleware patterns like closure-based middleware for encapsulating shared data and context-specific operations.

```
const http = require('http')

const server = http.createServer()
const products = [{name:"apple"},{name:"melon"}];

// post
function parse(req) {
    return new Promise ((resolve,reject)=>{
        let body= "";

        req.on('data',(chunk)=>{
            body+= chunk.toString();
        });
        req.on("end",()=>{
            if(body.includes("productsName=")){
                resolve({name: body.replace("productsName=","")});
                // console.log(body)
            }
            else{
                reject("invalid data");
            }
        })
    })
}
// middleware
server.prependListener('request',(req,res)=>{
    console.log(`Incoming ${req.method} for ${req.url}`)
    req.message = "message for middleware";
    req.error = "error coming from middleware"
})

server.on("request",(req,res)=>{
    console.log(req.message,"\n" ,req.error)
    if(req.url==="/"){
        res.setHeader("Content-Type","text/html");
        res.end(`
            <form action="/products" method="POST">
            <input type="text" name="productsName000" />
            <button type="submit">Post</button>
          </form>
        `);
    }else if (req.url==="/products"){
        if (req.method === "POST"){
            parse(req).then((product)=>{
                products.push(product);

                res.end(`product created \n
                    ${JSON.stringify(products)}`)
            }).catch((err)=>{
                res.statusCode =400;
                res.end(err)
            })
        }else if (req.method === "GET"){
            res.setHeader("Content-Type","application/json");
           res.statusCode =200;

            res.end(JSON.stringify(products));
           }else{
            res.setHeader("Content-Type","text/plain");
            res.statusCode =405;
```

```
            res.statusCode =200;

            res.end(JSON.stringify(products));
           }else{
            res.setHeader("Content-Type","text/plain");
            res.statusCode =405;
            res.end("method not allowed ");

        }
    } else{
        res.setHeader("Content-Type","text/plain");
        res.statusCode =404;
        res.end("not found ");
    }

  });


server.listen(3000,()=>{
    console.log('server is up and listening on 3000')
});


("GET POST PUT DELETE");
```

Activities and Insights:

- **Practical Implementation of POST Requests:**
    - Implemented Express.js routes and middleware to handle incoming POST requests for creating, updating, or processing data.
    - Validated and sanitized request body data to prevent security vulnerabilities and ensure compliance with application requirements.
- **Middleware Integration and Customization:**
    - Integrated third-party middleware libraries (e.g., body-parser, cors) and developed custom middleware functions tailored to specific application needs.
    - Explored middleware patterns for cross-cutting concerns such as error handling and authentication, optimizing application security and performance.
- **Documentation and Collaboration:**
    - Documented middleware configurations, request handling processes, and data sharing strategies to facilitate team collaboration and knowledge sharing.
    - Collaborated with team members to review middleware implementations, solicit feedback, and optimize middleware chains for improved code efficiency.