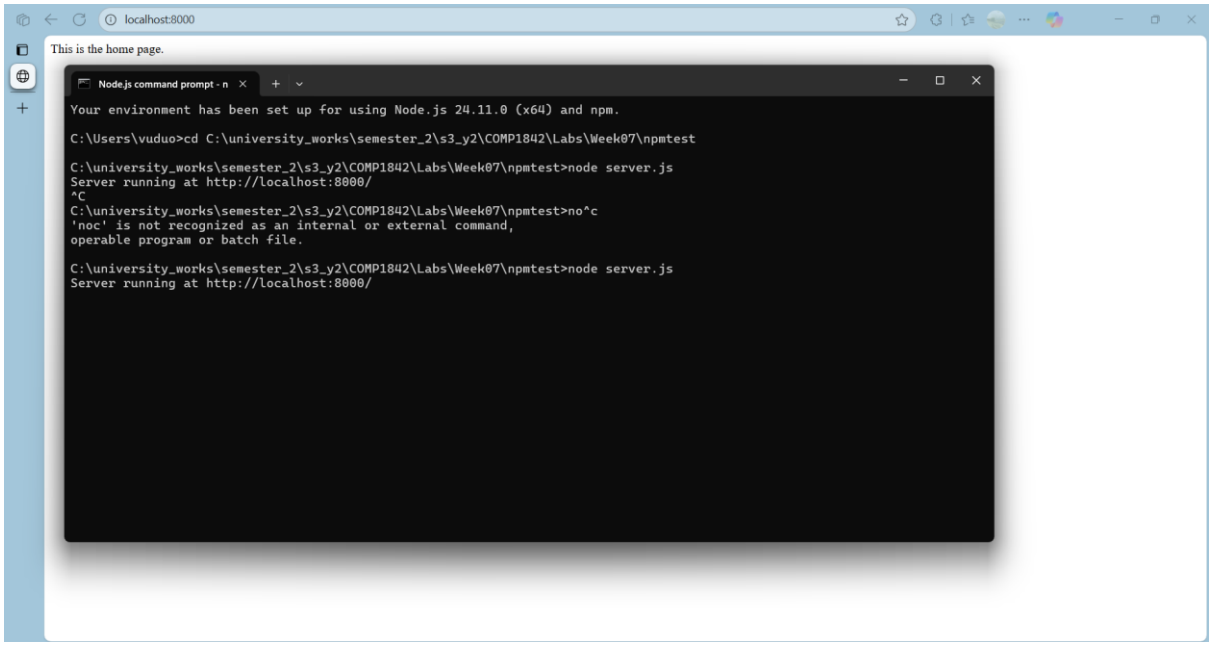
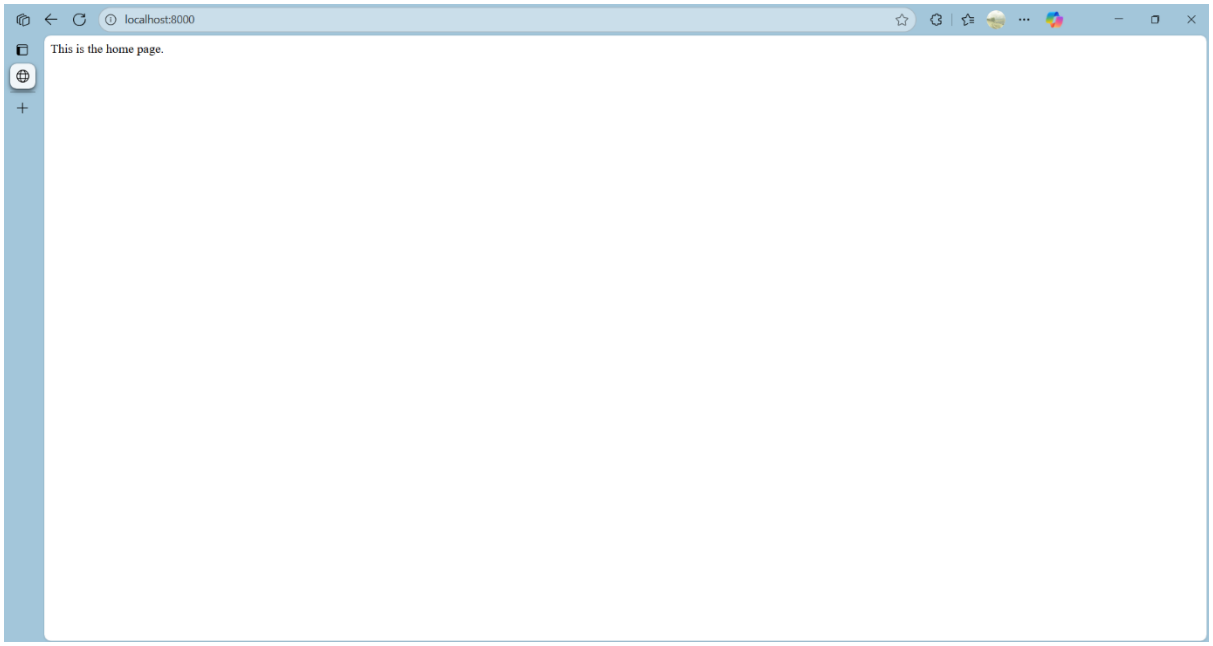
Part I – Evidence

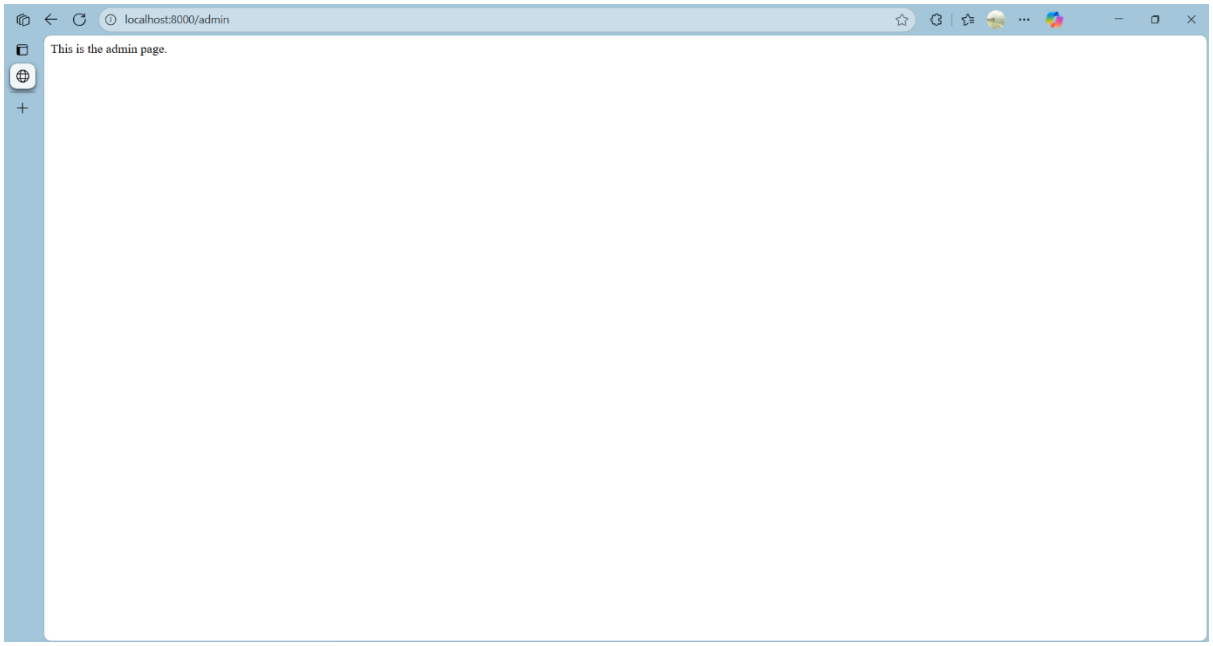
Screenshot 1 – Evidence of server.js is running on port 8000



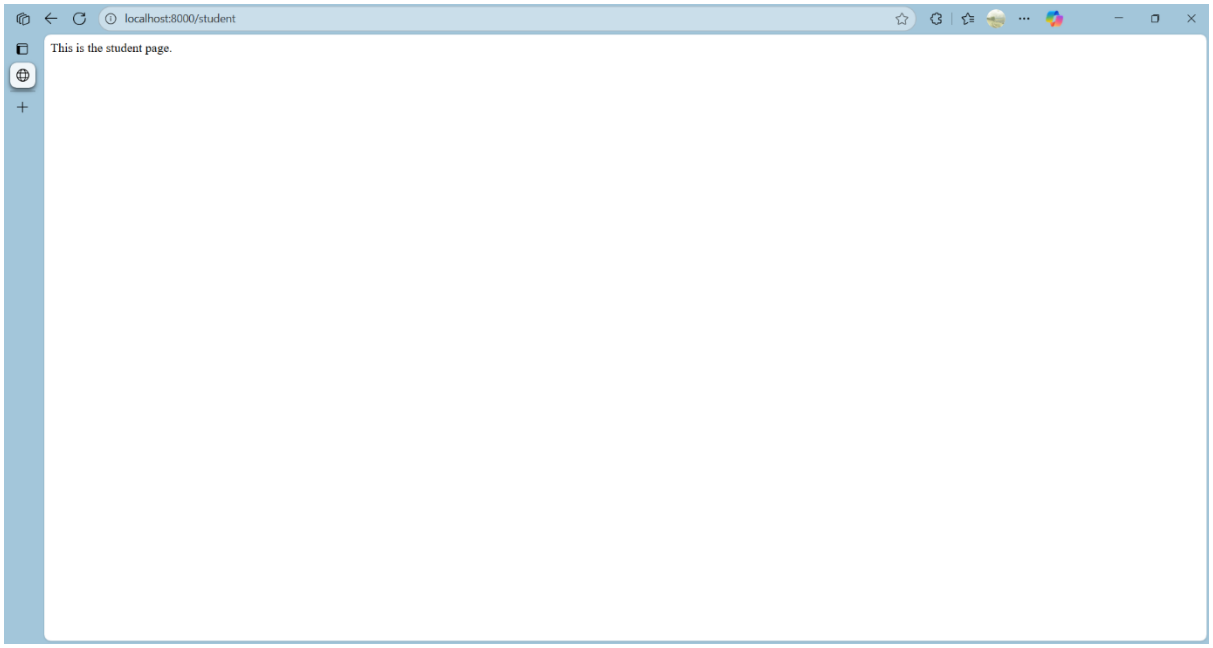
Screenshot 2 – Home Page



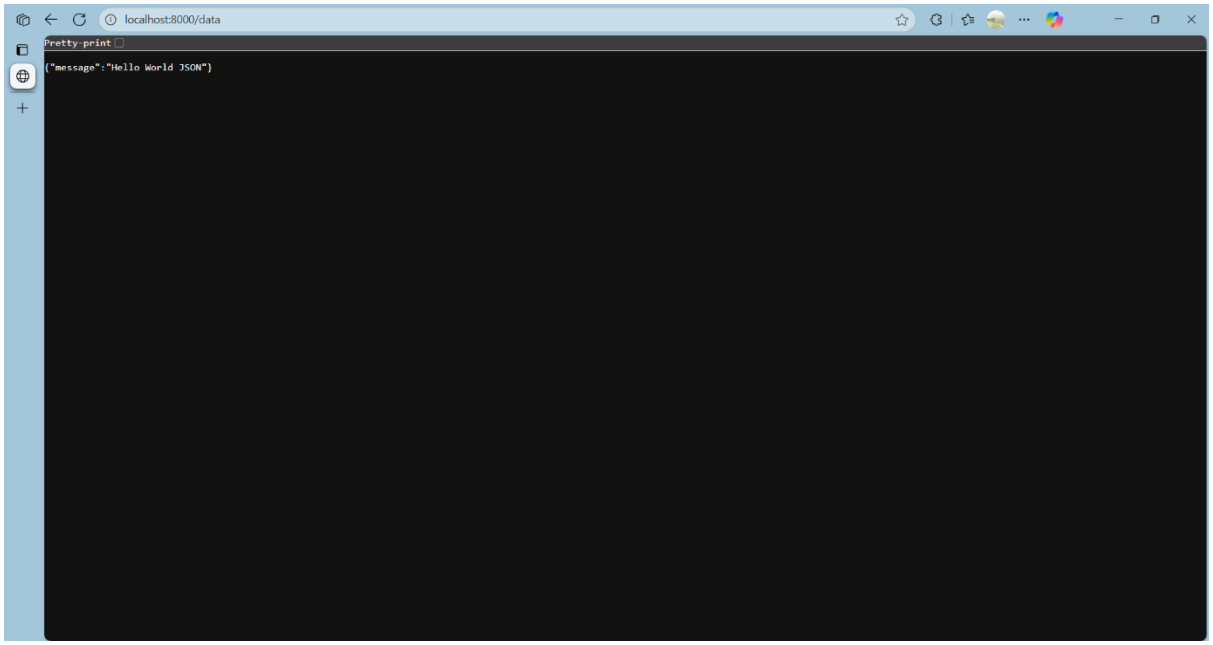
Screenshot 3 – Admin Page



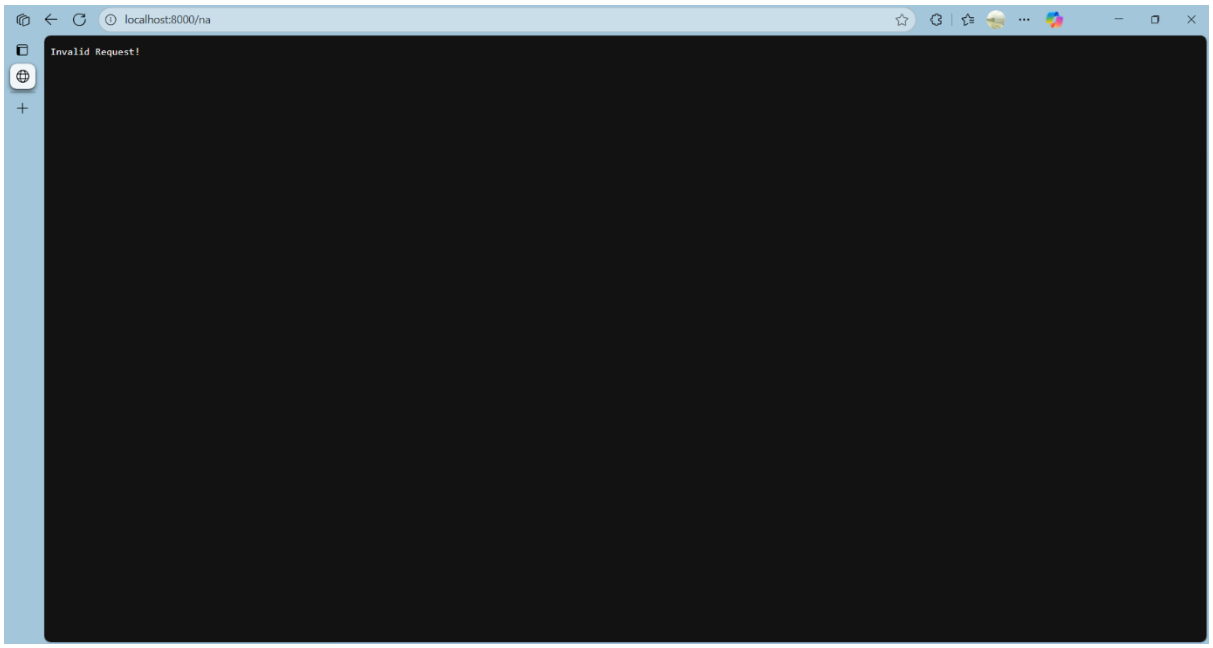
Screenshot 4 – Student Page



Screenshot 5 – Data Page



Screenshot 6 – Invalid Path



Part II – Understanding

From this lesson, I learned how Node.js can be used to create a simple web server using only JavaScript. Node.js enables JavaScript to run outside the browser, handling backend processes such as receiving and responding to HTTP requests. It operates on a single thread and utilises non-blocking asynchronous programming, allowing it to continue handling new requests while others are still being processed. This makes it very fast and efficient.

The lesson showed me how to use the built-in HTTP module to create a server. By using require('http'), I can import the module and utilise HTTP.createServer() I can make a server that listens for requests. The two main objects, request and response, are automatically passed

into the function. The request object provides details about what the client is requesting, such as the URL, while the response object is used to send data back to the browser.

I also learned to set response headers using writeHead() and send output with write() and end(). In the example, different URLs such as "/home", "/student", "/admin", and "/data" return different messages or even a JSON response. Running the command node server.js starts the server on port 8000. Typing localhost:8000 into a browser displays the result.

Following this exercise helped me understand how a web server actually handles requests and how Node.js can generate both HTML and JSON responses directly without relying on software like Apache or PHP.