# SIT314 – Week 6 Technical Task Report

## Task 6a: Develop a Simple Web Server with Express.js

In Task 6a, I developed a simple web server using Node.js and the Express framework. The server performs basic back-end routing and serves both plain text and HTML content.

Key actions:

* • Initialized a Node.js project (npm init)
* • Installed the Express library (npm install express)
* • Created a folder structure with a /web folder to hold static content
* • Developed an HTML file welcome.html and served it using a /welcome route
* • Implemented additional endpoints: / (hello world), /secret (warning), \* (404 page)

To enhance usability, I added a navigation menu inside welcome.html to allow switching between pages easily. To simplify development, I used nodemon to auto-restart the server when server.js changes.

## Task 6b: Serving IoT Sensor Data

In Task 6b, I extended the web server to simulate a real-world IoT API by adding a new endpoint: /sensorData.  
This endpoint creates a mock JSON object representing sensor data (e.g., temperature), generates a random temperature reading on each request, and returns the data with the appropriate application/json header.

To test this, I created a simple client (client.js) using the request module. The client makes a GET request to /sensorData and parses the JSON response, printing the sensor ID, name, address, timestamp, and temperature.

I later replaced request with the modern axios module to address vulnerability warnings.

## Task 6c: Controlling IoT Devices via Web Calls

In Task 6c, I simulated a basic IoT node with on/off control:  
  
• Created iot\_node.js, which listens on port 3001 and provides:  
 - /lightOn – Logs “Switching Light On”  
 - /lightOff – Logs “Switching Light Off”  
• Created iot\_client.js to send HTTP GET requests to both endpoints.  
• When run, it triggers the light on/off logic on the server.

Both programs were executed in separate terminals:  
1. Run the IoT node: node iot\_node.js  
2. Run the client: node iot\_client.js  
Console logs confirmed successful interaction between the client and the IoT node.

## Reflection

This task demonstrated how powerful yet simple Express.js can be for creating lightweight web services in IoT. I learned how to serve static files, create dynamic API endpoints, simulate real-time sensor readings, and control remote devices through REST-style interfaces. These skills will be directly useful in future IoT microservice designs and cloud-based interactions.