Documentation Report

Link to application environment:

https://hub.labs.coursera.org:443/connect/sharedaccyygxl?forceRefresh=false&path=%2F%3Ffolder%3D%2Fhome%2Fcoder%2Fproject

The file package for the mid-term assignment is titled “midterm_v3”, containing the various files to run the web application.

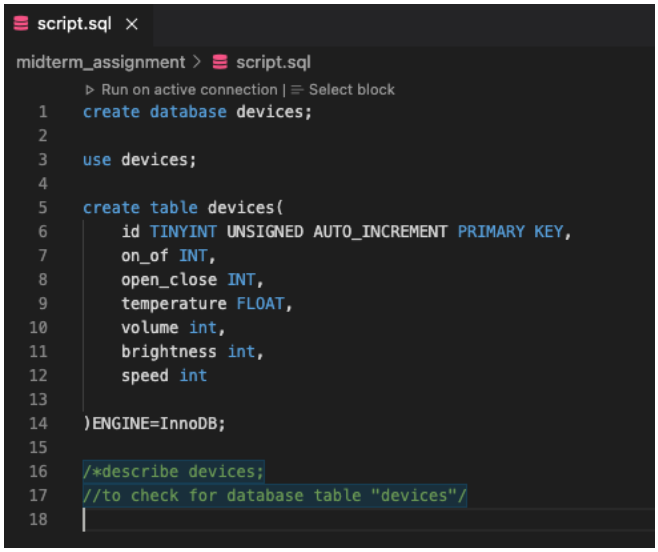
As for the SQL database, I’ve also added a file titled “script.sql”, to show the scripts that are used to create the SQL database and create the tables for the users’ data input for their devices for the web application.

Introduction to web application:

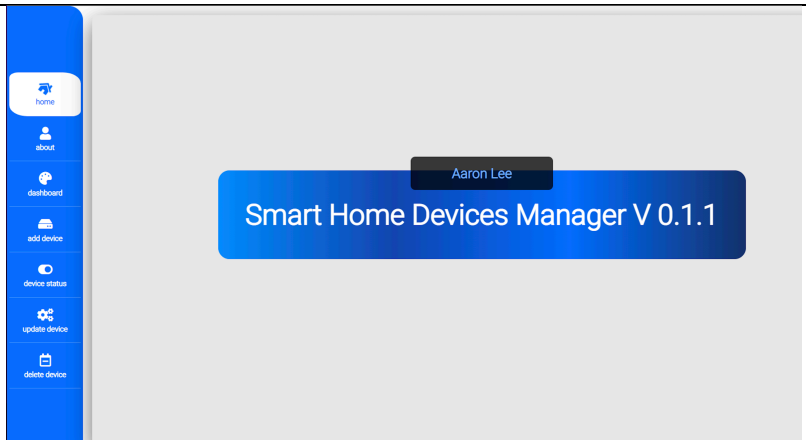
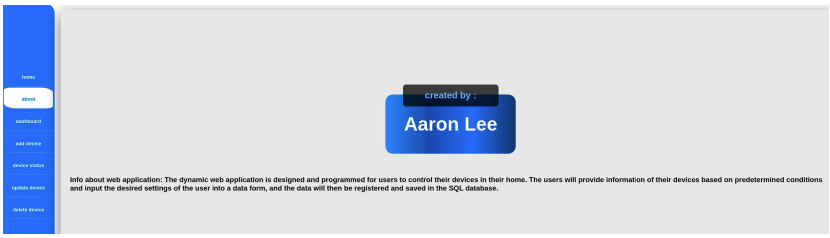
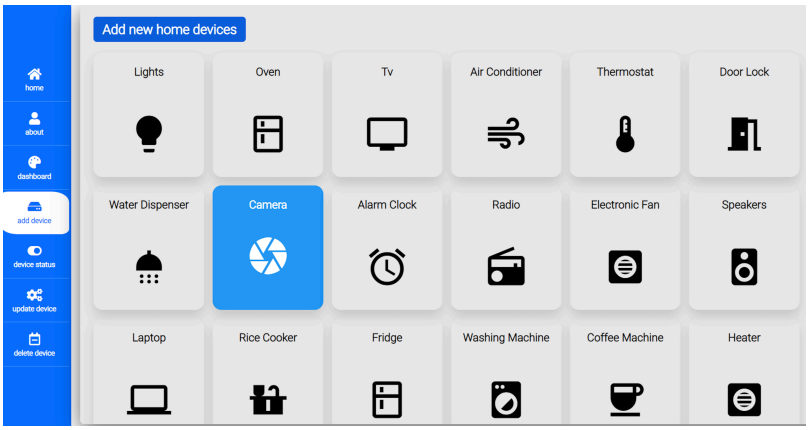
The dynamic web application is designed and programmed for users to control their devices in their home. The users will provide information of their devices based on predetermined conditions and input the desired settings of the user into a data form, and the data will then be registered and saved in the SQL database. For instance, the user can input the temperature settings of the air-conditioner in their room and switch it on using the web application. I used “EJS” for the coding of my web pages and used “CSS” to design the layouts of the grid-lists of the devices. Javascript is used to interact with the user’s inputs and event-programming, which requires on-click actions by the user. With regards to my assignment files, “main.js” primarily affects the web pages, and “app.js” interacts with the user input form, which requires the user to input details of the type of device to add to the database, or simply just delete them.

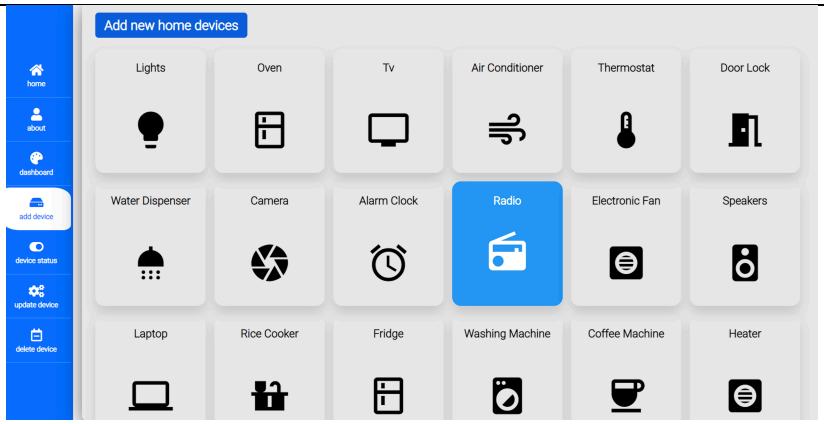
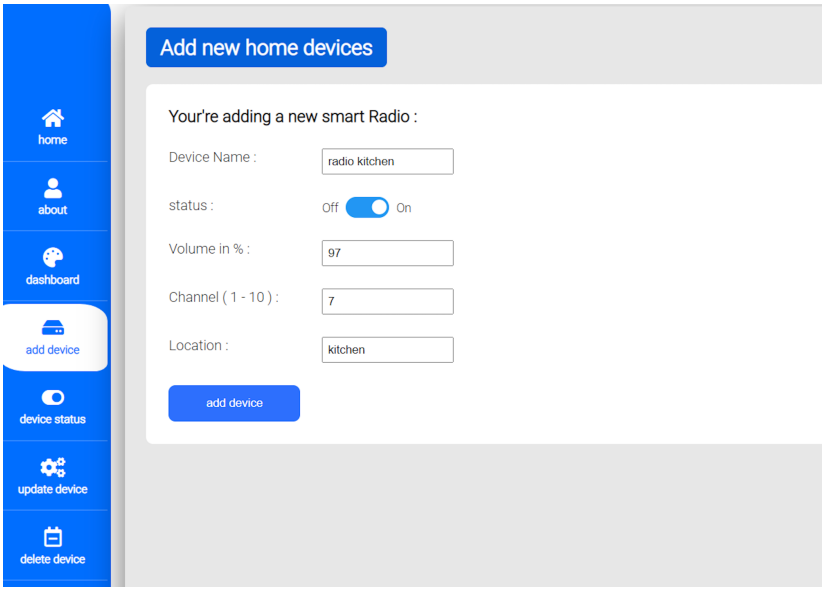
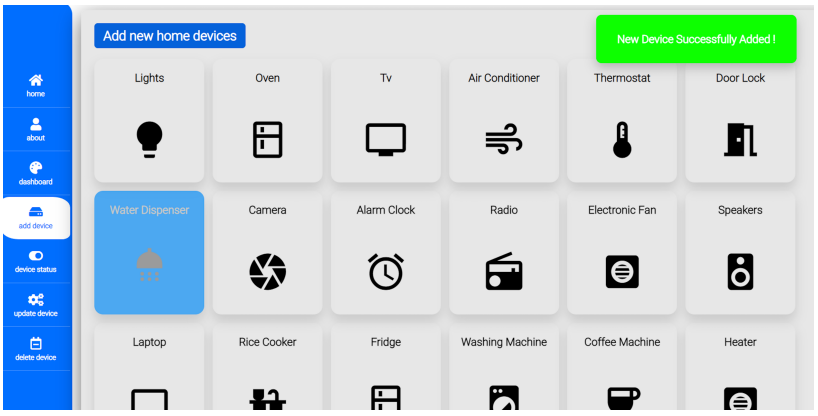
Database in SQL:

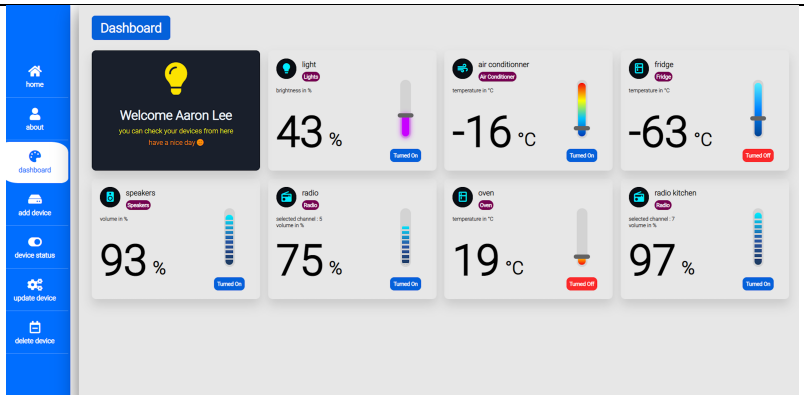
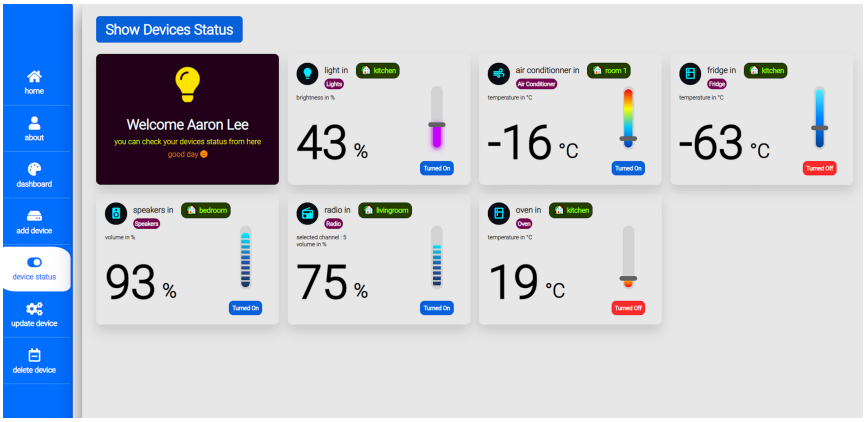
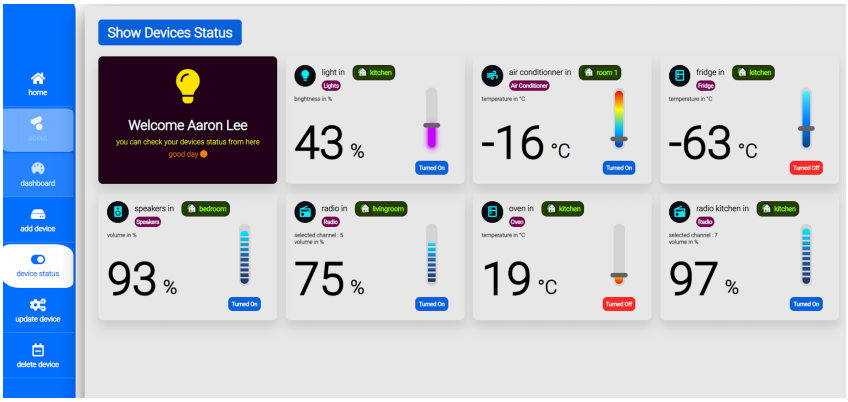
To facilitate the creation of a database for my web application, I’ve used SQL with the following scripts to create my database for query in the web application, based on the users’ input for the selected devices. (found in script.sql)

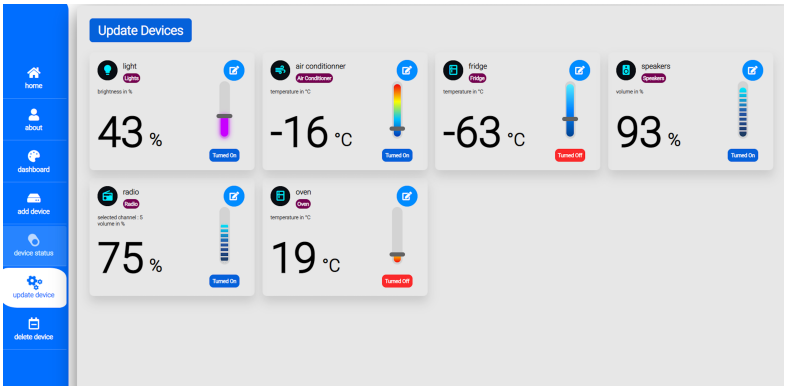
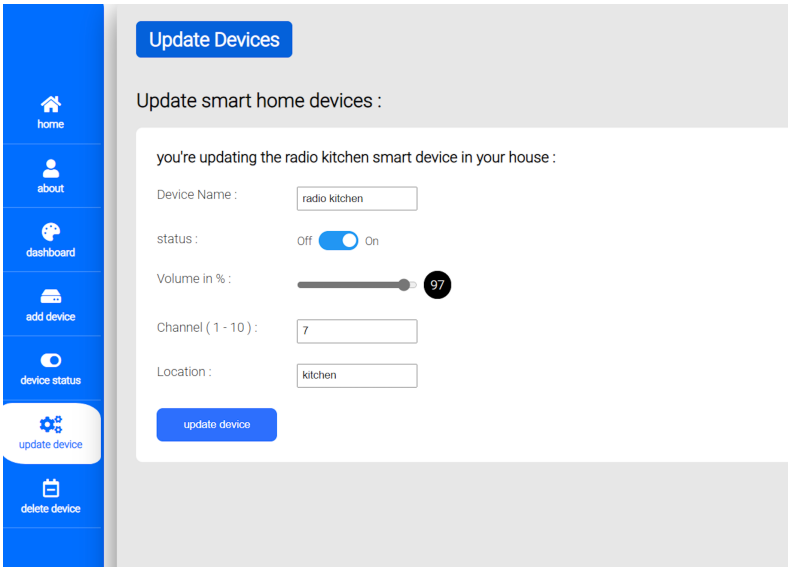
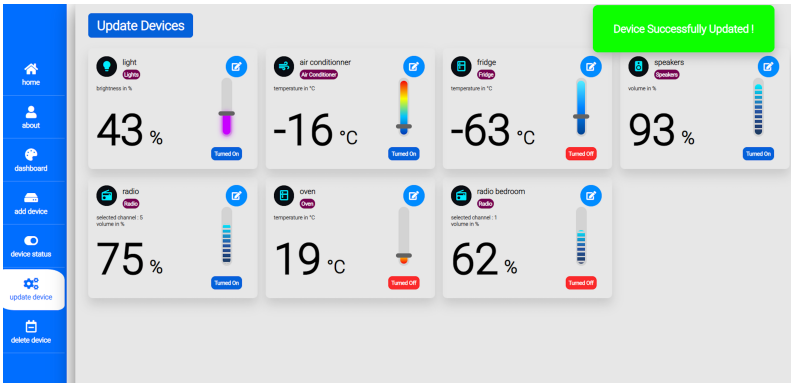


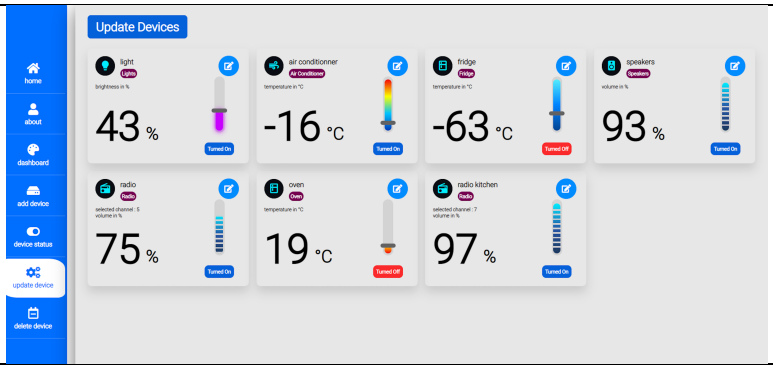
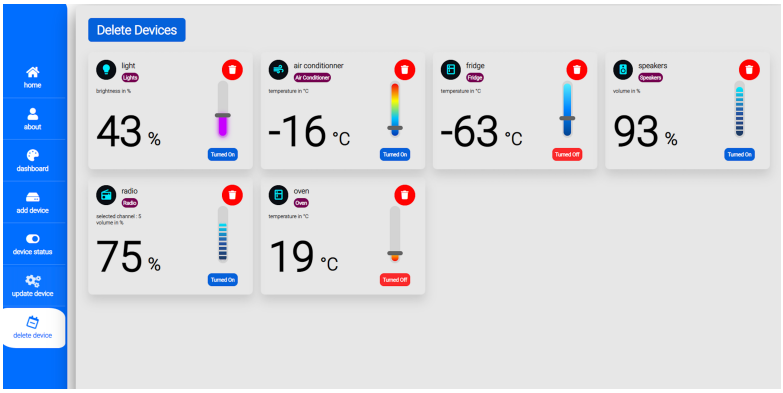
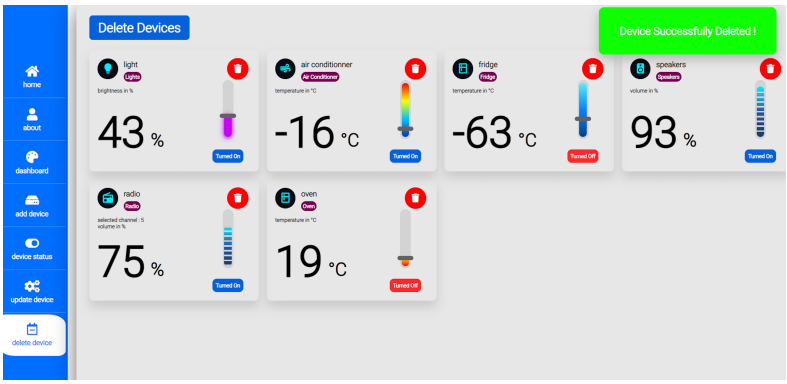
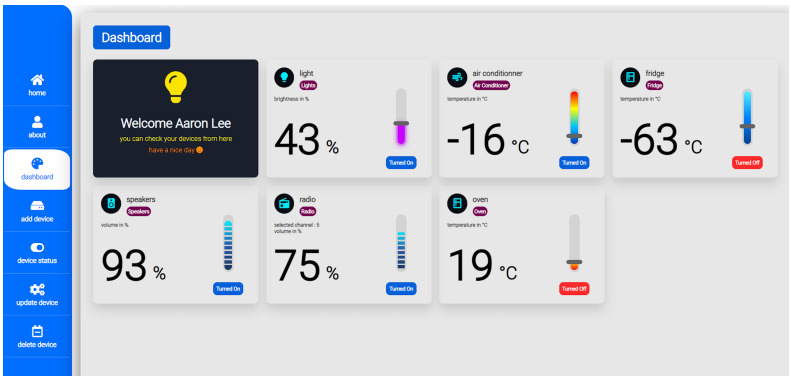
The table below shows the web application pages and its functions:

Coursework Requirements	Functions	Web App Display
R1: Home Page	R1A: Displays the name of the web application R1B: Displays a navigation bar that links to the other pages	
R2: About Page	R2A: Displays information about what the web application does with a navigation bar	
R3: Add device page	R3A: Displays a form for users to input new device, with requirements, based on types of devices saved in database (20 of them) R3B: Collects the data inputted by user in the form to save it in database	

	R3C: Auto-fill invalid input fields for certain devices R3D: Validate the data inputted in form, if not prompt the user to enter details of device again	  

		
R4: Show device status page	R4A: Display list of added devices and allow user to choose device from list R4B: Display data related to chosen device in database to include name of device, and adjusted settings. Also displays message if device not found R4C: Hide input fields not related to device for settings R4D: Display graphical display of device settings like a dashboard	 

R5: Update device status page	R5A: Display list of added devices R5B: Display data relevant to selected device, and allow users to update device settings R5C: Hide input fields not applicable to selected device R5D: Allow user to update device settings graphical display of device settings on a dashboard	  

		
R6: Delete device page	R6A: Display list of added devices and allow user to delete selected device R6B: Seek user's confirmation before deleting device from database	 
Dashboard	Showing active device(s) status	

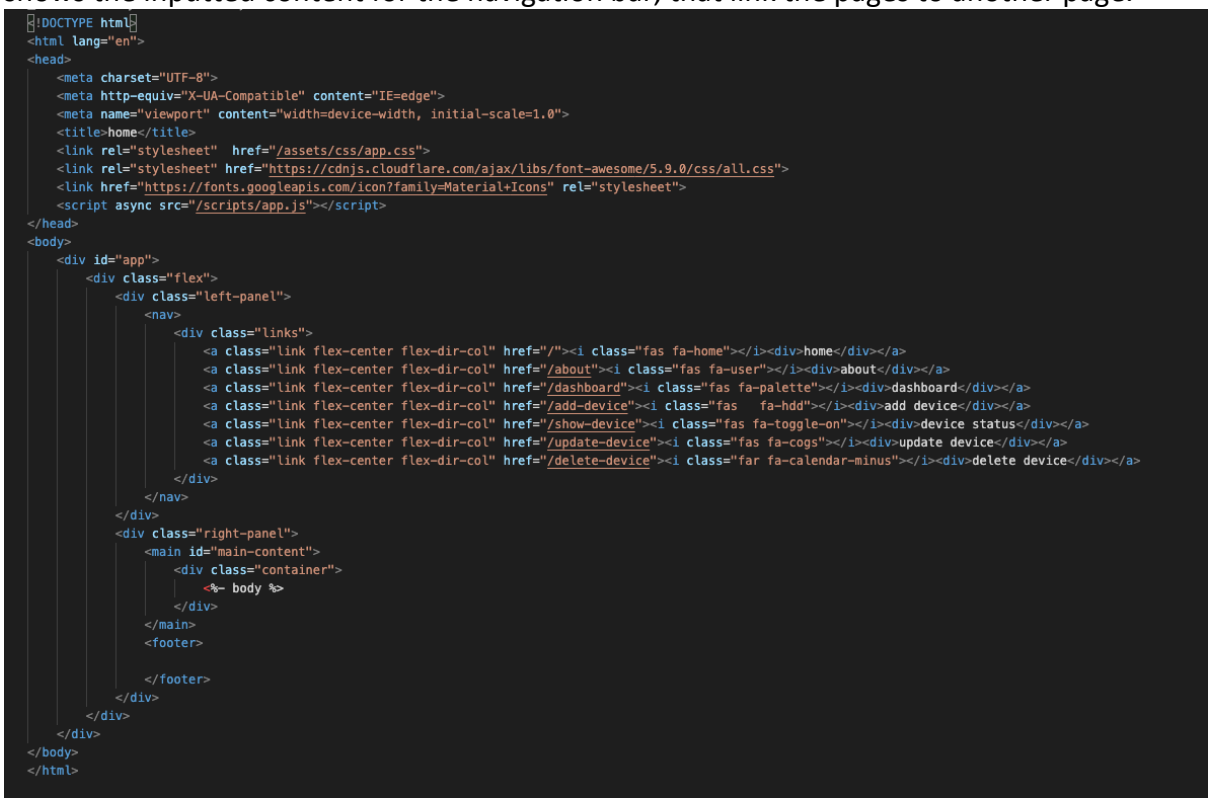
Code Extracts

R1A: Displays the name of the web application and navigation bar:

The main page of the MySmartHome web application, stating a welcome message for the user.
This code extract will be found in my index.ejs file, under the “view” folder, as the “view” folder contains my ejs files for the webpage contents and layouts.

```
<div class="content">
    <div class="flex-center flex-dir-col" style="height: 100%;">
        <h1 class="app-title">Smart Home Devices Manager V 0.1.1</h1>
        <h2 class="maker">Aaron Lee</h2>
    </div>
</div>
```

And in my “view” folder, there is a “layout” folder, that contains the file “main.ejs”, that shows the inputted content for the navigation bar, that link the pages to another page.



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>home</title>
    <link rel="stylesheet"  href="/assets/css/app.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.9.0/css/all.css">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <script async src="/scripts/app.js"></script>
</head>
<body>
    <div id="app">
        <div class="flex">
            <div class="left-panel">
                <nav>
                    <div class="links">
                        <a class="link flex-center flex-dir-col" href="/"><i class="fas fa-home"></i><div>home</div></a>
                        <a class="link flex-center flex-dir-col" href="/about"><i class="fas fa-user"></i><div>about</div></a>
                        <a class="link flex-center flex-dir-col" href="/dashboard"><i class="fas fa-palette"></i><div>dashboard</div></a>
                        <a class="link flex-center flex-dir-col" href="/add-device"><i class="fas   fa-hdd"></i><div>add device</div></a>
                        <a class="link flex-center flex-dir-col" href="/show-device"><i class="fas fa-toggle-on"></i><div>device status</div></a>
                        <a class="link flex-center flex-dir-col" href="/update-device"><i class="fas fa-cogs"></i><div>update device</div></a>
                        <a class="link flex-center flex-dir-col" href="/delete-device"><i class="far fa-calendar-minus"></i><div>delete device</div></a>
                    </div>
                </nav>
            </div>
            <div class="right-panel">
                <main id="main-content">
                    <div class="container">
                        <%- body %>
                    </div>
                </main>
                <footer>

                </footer>
            </div>
        </div>
    </div>
</body>
</html>
```

The CSS designs are contained in the folder “public” > “assets” > “css”, it contains the css files that designs the layout of the webpages. The 3 css files specifically contains the code for the design of the web application, dashboard, and the query form for the user to input the device details.

Code extract from app.css:

```css
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@200&family=Roboto:wght@300&display=swap');
@import url('dashboard.css');
@import url('form.css');

*{
    padding: 0;
    margin: 0;
    box-sizing: border-box;
}

body,html{
    /* background-color: rgb(14, 19, 24); */
    font-family: 'roboto',sans-serif;
    overflow:hidden;
}

#app{
    height:100vh;
    /* background-image: url("/assets/images/2.png"); */
    background-repeat: no-repeat;
    background-size: cover;
}

.container{
    margin: auto 0;
    padding: 1rem;
    height: 100%;
}

.flex{
    display:flex;
}

.flex-center
{
    display: flex;
    align-items: center;
    justify-content: center;
}

.flex-dir-col
{
    flex-direction: column;
}

.left-panel,.right-panel{

    height:100vh;
}

.left-panel{
    overflow: hidden;
    border-top-right-radius: 25px;
    background-color:#006eff;
    padding: 0.25rem;
    width:146px;
}

.right-panel{
    background-color:#ffffff;
    width:100%;
}
```

Code extract from dashboard.css:

This extract shows a feature of the web application, which enables the user to slide the switch on the webpage for changing device settings.

```css
/* The switch - the box around the slider */
.switch {
  position: relative;
  display: inline-block;
  width: 50px;
  height: 24px;
}

/* Hide default HTML checkbox */
.switch input {
  opacity: 0;
  width: 0;
  height: 0;
}

/* The slider */
.slider {
  position: absolute;
  cursor: pointer;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  background-color: #ccc;
  -webkit-transition: .4s;
  transition: .4s;
}

.slider:before {
  position: absolute;
  content: "";
  height: 17px;
  width: 17px;
  left: 4px;
  bottom: 4px;
  background-color: white;
  -webkit-transition: .4s;
  transition: .4s;
}

input:checked + .slider {
  background-color: #2196F3;
}

input:focus + .slider {
  box-shadow: 0 0 1px #2196F3;
}

input:checked + .slider:before {
  -webkit-transform: translateX(26px);
  -ms-transform: translateX(26px);
  transform: translateX(26px);
}

/* Rounded sliders */
.slider.round {
  border-radius: 34px;
}

.slider.round:before {
  border-radius: 50%;
}
```

Code extract from form.css:

I would like to bring attention to notification design for notifying users the successful/ unsuccessful inputs, that works with javascript.

```css
.result_val
{
    background-color: black;
    width: 35px;
    height: 35px;
    margin-left: 8px;
    border-radius: 50%;
    color: white;
}

.btn
{
    outline: none;
    color: white;
    background-color: rgb(45, 111, 253);
    border-radius: 8px;
    padding: 0.8rem 1.2rem !important;
    cursor: pointer;
    border: none;
}

.notification
{
    z-index: 999;
    background-color: rgb(9, 255, 0);
    color: white;
    font-weight: bold;
    font-size: 20px;
    box-shadow:0 8px 20px #0000003b;
    padding:2rem 2.4rem;
    border-radius:10px;
    position:absolute;
    right:0;
    transform: translateY(-180px);
    opacity: 0;
    top:0;
    margin:25px 85px;
    transition:all 0.8s cubic-bezier(0.075, 0.82, 0.165, 1);
}

.fade-in
{
    transform: translateY(0px) !important;
    opacity: 1 !important;
}
```

R2A: Displays information about what the web application does with a navigation bar:

The code extract below is from “views”> “about.ejs”, which shows the content for the “About” page.

```
<div class="content">
    <div class="flex-center flex-dir-col" style="height: 100%;">
        <h1 class="app-title">Aaron Lee</h1>
        <h2 class="maker">created by : </h2>
        <h3 class="description"> Info about web application: The dynamic web application is designed and programmed for users to
            control their devices in their home. The users will provide information of their devices based on predetermined
            conditions and input the desired settings of the user into a data form, and the data will then be registered and
            saved in the SQL database. </h3>
    </div>
</div>

```

The navigation bar code extract is demonstrated above, which stays the same throughout the webpages.

R3A: Displays a form for users to input new device, with requirements, based on types of devices saved in database (20 of them):

The code extract for the “Add Device” page is shown below, that can be found in “views”>”add-device.ejs”

```
<div class="content">
    <h2 class="top-title">Add new home devices </h2>
    <br>
    <% if(message !== "") { %>
        <div class="notification hide"><%= message %></div>
    <% } %>
    <div class="devices grid-select">
        <div class="device-card" style="border-radius:1rem;">
            <div class="top flex-center">
                <div class="device-name" style="font-size: 20px;">
                    Lights
                </div>
            </div>
            <div class="bottom flex-center">

                <div class="flex-center">
                    <li class="material-icons" style="font-size: 80px;position: relative;top: 15px;">lightbulb</li>
                </div>
            </div>
        </div>
        <div class="device-card" style="border-radius:1rem;">
            <div class="top flex-center">
                <div class="device-name" style="font-size: 20px;">
                    Oven
                </div>
            </div>
            <div class="bottom flex-center">
                <div class="flex-center">
                    <li class="material-icons" style="font-size: 80px;position: relative;top: 15px;">kitchen</li>
                </div>
            </div>
        </div>
        <div class="device-card" style="border-radius:1rem;">
            <div class="top flex-center">
                <div class="device-name" style="font-size: 20px;">
                    Tv
                </div>
            </div>
            <div class="bottom flex-center">

                <div class="flex-center">
                    <li class="material-icons" style="font-size: 80px;position: relative;top: 15px;">tv</li>
                </div>
            </div>
        </div>
        <div class="device-card" style="border-radius:1rem;">
            <div class="top flex-center">
                <div class="device-name" style="font-size: 20px;">
                    Air Conditioner
                </div>
            </div>
            <div class="bottom flex-center">

                <div class="flex-center">
                    <li class="material-icons" style="font-size: 80px;position: relative;top: 15px;">air</li>
                </div>
            </div>
        </div>
        <div class="device-card" style="border-radius:1rem;">
            <div class="top flex-center">
                <div class="device-name" style="font-size: 20px;">
                    Thermostat
                </div>
            </div>
            <div class="bottom flex-center">
```

The code extract shows the layout and design of the “add device” page for users. When the user decides to add a new device, on-click, it will bring about a form to query the user’s

device’s details, after the user selects the specific type of device to add to the database, such as an air-conditioner.

The code extract below is from “update-device-form.ejs”, which contains the contents of the form for user input.

```
<div class="content">
    <h2 class="top-title">Update Devices </h2>
    <br><br>
    <h2>Update smart home devices : </h2>
    <br>
    <div class="devices-form">
        <form action="/update-device" method="post">
            <div class="form-input name flex">
                <h3 class="form-title">you're updating the <%= row.name %> smart device in your house : </h3>
                <input id="device_type" type="hidden" name="type" value="<%= row.type%>">
                <input id="device_id" type="hidden" name="device_id" value="<%= row.id %>">
            </div>
            <div class="form-input name flex">
                <label class="label" for="">Device Name : </label>
                <div class="flex">
                    <input type="text" name="dev_name" value="<%= row.name %>" required>
                </div>
            </div>
            <div class="form-input flex">
                <label class="label" for="onof">status : </label>
                <div class="flex-center">
                    <span style="margin-right: 8px;font-size:14px;">Off </span>
                    <label class="switch" >
                        <% if(row.on_of) { %>
                        <input type="checkbox" name="onof" id="onof" value="1" checked>
                        <% }else { %>
                        <input type="checkbox" name="onof" id="onof" value="0">
                        <% } %>
                        <span class="slider round"></span>
                    </label>
                    <span  style="margin-left: 8px;font-size:14px;"> On</span>
                </div>
            </div>
            <% if(row.type.includes("Door Lock")) { %>
            <div class="form-input open flex">
                <label class="label" for="onof">open/close</label>
                <div class="flex-center">
                    <span style="margin-right: 8px;font-size:14px;">Close </span>
                    <label class="switch">
                        <% if(row.open_close) { %>
                        <input type="checkbox" name="openclose" id="openclose" value="1" checked>
                        <% }else { %>
                        <input type="checkbox" name="openclose" id="openclose" value="0">
                        <% } %>
                        <span class="slider round"></span>
                    </label>
                    <span  style="margin-left: 8px;font-size:14px;"> Open</span>
                </div>
            </div>
```

The code extract below shows the css layout for the user device form.

```css
form
{
    background-color: rgb(255, 255, 255);
    border-radius: 0.5rem;
    padding: 0.8rem;
}

.form-input
{
    width:100%;
    padding:0.8rem;

}

.form-input input
{
    outline:none;
    width:150px;
    padding:0.3rem;
}

.form-input .label
{
    margin-right:25px;
    width:150px;
}

.delete-btn,.update-btn
{
    pointer-events: all;
    position: absolute;
    right: 0;
    margin-right: 1.2rem;
}

.delete-btn a
{
    cursor: pointer;
    width: 40px;
    height: 40px;
    display: flex;
    align-items: center;
    background-color: red;
    justify-content: center;
    border-radius: 50%;
}

.update-btn a
{
    cursor: pointer;
    width: 40px;
    height: 40px;
    display: flex;
    align-items: center;
    background-color: rgb(0, 140, 255);
    justify-content: center;
    border-radius: 50%;
}
```

R3B: Collects the data inputted by user in the form to save it in database:

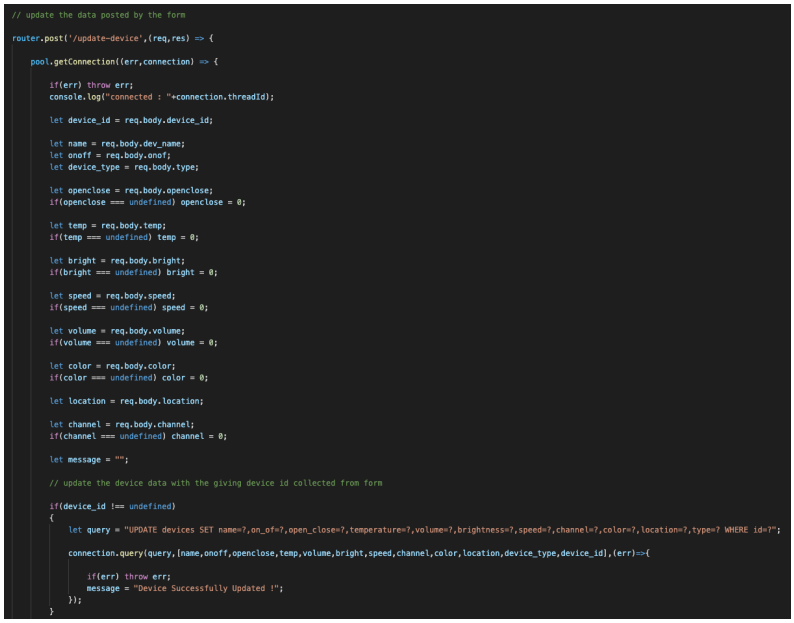
routes> “main.js” will configure the database connection, from the user’s input to the SQL database.

```
const { query } = require('express');
const express = require('express');
const { append, render } = require('express/lib/response');
const mysql = require('mysql');
const path = require('path');
const router = express.Router();

// config the database connection

const pool = mysql.createPool({
    connectionLimit : 100,
    host : "localhost",
    user : "root",
    password : "",
    database : "devices"
});

```

It will then bring the user to the specific page to add a specific device.

```
// redirect to the add-device page for chosing the a specific device to add

router.get('/add-device',(req,res) => {

    let message = "";

    res.render('add-device',{message});
});
```

This will allow the user to choose specifications for the selected type of device to add to the “SmartHome” devices’ database.



```
// update the data posted by the form

router.post('/update-device',(req,res) => {

    pool.getConnection((err,connection) => {

        if(err) throw err;
        console.log("connected : "+connection.threadId);

        let device_id = req.body.device_id;

        let name = req.body.dev_name;
        let onoff = req.body.onof;
        let device_type = req.body.type;

        let openclose = req.body.openclose;
        if(openclose === undefined) openclose = 0;

        let temp = req.body.temp;
        if(temp === undefined) temp = 0;

        let bright = req.body.bright;
        if(bright === undefined) bright = 0;

        let speed = req.body.speed;
        if(speed === undefined) speed = 0;

        let volume = req.body.volume;
        if(volume === undefined) volume = 0;

        let color = req.body.color;
        if(color === undefined) color = 0;

        let location = req.body.location;

        let channel = req.body.channel;
        if(channel === undefined) channel = 0;

        let message = "";

        // update the device data with the giving device id collected from form

        if(device_id !== undefined)
        {
            let query = "UPDATE devices SET name=?,on_of=?,open_close=?,temperature=?,volume=?,brightness=?,speed=?,channel=?,color=?,location=?,type=? WHERE id=?";

            connection.query(query,[name,onoff,openclose,temp,volume,bright,speed,channel,color,location,device_type,device_id],(err)=>{

                if(err) throw err;
                message = "Device Successfully Updated !";
            });
        }
```

R3C: Auto-fill invalid input fields for certain devices:

scripts>”app.js” will check will the specific type of device the user wants to add and then hide/show specific fields for the user to input. So if the user selects a specific device to add, the input fields will vary according to the specifications of the device to allow for user input. For instance, the settings for an air-conditioner will be different from that of a television(TV).

```
if(selected_devs !== null)
selected_devs.forEach(device => {

    device.addEventListener('click',(e)=>{

        let _this = e.target;
        let name = _this.querySelector('.device-name').innerHTML;

        document.querySelector('.devices').classList.add('hide');
        document.querySelector('.devices-form').classList.remove('hide');
        document.querySelector('.devices-form').querySelector('.form-title').innerHTML = "Your're adding a new smart "+name+" : ";
        document.getElementById('device_type').value = name;

        if(name.includes("Lights"))
        {
            document.querySelector('.bright').classList.remove('hide');
            document.querySelector('.color').classList.remove('hide');
        }

        if(name.includes("Tv"))
        {
            document.querySelector('.chan').classList.remove('hide');
            document.querySelector('.vol').classList.remove('hide');
        }

        if(name.includes("Thermostat"))
        {
            document.querySelector('.temp').classList.remove('hide');
            document.querySelector('.temp #temper').setAttribute('max',100);
            document.querySelector('.temp #temper').setAttribute('min',0);
        }

        if(name.includes("Air Conditioner"))
        {
            document.querySelector('.temp').classList.remove('hide');
            document.querySelector('.temp #temper').setAttribute('max',50);
            document.querySelector('.temp #temper').setAttribute('min',-25);
        }

        if(name.includes("Air Purifier"))
        {
            document.querySelector('.speed').classList.remove('hide');
        }

        if(name.includes("Heater"))
        {
            document.querySelector('.temp').classList.remove('hide');
            document.querySelector('.temp #temper').setAttribute('max',100);
            document.querySelector('.temp #temper').setAttribute('min',0);
        }

        if(name.includes("Coffee Machine"))
        {
            document.querySelector('.temp').classList.remove('hide');
            document.querySelector('.temp #temper').setAttribute('max',100);
            document.querySelector('.temp #temper').setAttribute('min',0);
        }
```

R3D: Validate the data inputted in form, if not prompt the user to enter details of device again:

The code extract below shows that if the user did not manage to input valid details in the form to query the settings/ details of the specific device to add, an error message will be shown.

```
// redirect to the update-device page for chosing the a specific device to update

router.get('/update-device',(req,res) => {

    pool.getConnection((err,connection) => {

        let query = "SELECT * FROM devices";
        let message = "";

        connection.query(query,(err,rows)=>{

            connection.release();

            if(!err)
            {
                res.render('update-device',{rows,message});
            }
            else
            {
                console.log("error");
            }
        });
    });
});
```

The code extract below shows the specific input fields for certain devices, depending on the user's type of device selection. Each device has a specific form for the user to fill up to input the desired settings.

```
// add the device by posting data with the specific form

router.post('/add-device',(req,res)=>{

    pool.getConnection((err,connection) => {
        if(err) throw err;
        console.log("connected : "+connection.threadId);

        let name = req.body.dev_name;
        let onoff = req.body.onof;
        let device_type = req.body.type;

        let openclose = req.body.openclose;
        if(openclose === undefined) openclose = 0;

        let temp = req.body.temp;
        if(temp === undefined) temp = -1;

        let bright = req.body.bright;
        if(bright === undefined) bright = -1;

        let speed = req.body.speed;
        if(speed === undefined) speed = -1;

        let volume = req.body.volume;
        if(volume === undefined) volume = -1;

        let color = req.body.color;
        if(color === undefined) color = -1;

        let location = req.body.location;

        let channel = req.body.channel;
        if(channel === undefined) channel = -1;

        let query = "INSERT INTO devices(name,on_of,open_close,temperature,volume,brightness,speed,channel,color,location,type) VALUES(?,?,?,?,?,?,?,?,?,?,?)"
```

After querying the user's input into the database for the specific device, a notification will be shown on the top right that the device has been successfully added.

```
        // add a new device by inserting data in the devices table

        connection.query(query,[name,onoff,openclose,temp,volume,bright,speed,channel,color,location,device_type],(err,result)=>{

            connection.release();

            if(err) throw err;
        });
    });

    let message = "New Device Successfully Added !";

    res.render("add-device",{message});
});

module.exports = router;
```

The code extract below shows the css layout for the notification bar. The notification will auto fade after a while.

```
.notification
{
    z-index: 999;
    background-color: rgb(9, 255, 0);
    color: white;
    font-weight: bold;
    font-size: 20px;
    box-shadow:0 8px 20px #0000003b;
    padding:2rem 2.4rem;
    border-radius:10px;
    position:absolute;
    right:0;
    transform: translateY(-180px);
    opacity: 0;
    top:0;
    margin:25px 85px;
    transition:all 0.8s cubic-bezier(0.075, 0.82, 0.165, 1);
}

.fade-in
{
    transform: translateY(0px) !important;
    opacity: 1 !important;
}
```

R4A: Display list of added devices and allow user to choose device from list:

The code extract below shows the devices' list with their specific settings, which can be found in the "views" folder > "show-device.ejs" file.

```
<div class="bottom flex">
        <div class="left">
            <div class="device-value">
                <% if(device.type.includes('Door Lock')){ %>
                    <% if(device.open_close === 1) { %>
                        <span class="openclosespan opened">Opened</span>
                    <% }else{ %>
                        <span class="openclosespan closed">Closed</span>
                    <% } %>
                <% } %>
                <% if(device.type.includes('Lights')){ %>
                    <%= device.brightness %><span>%</span>
                <% } %>
                <% if(device.type.includes('Tv') || device.type.includes('Speakers')|| device.type.includes('Radio')){ %>
                    <%= device.volume %><span>%</span>
                <% } %>
                <% if(device.type.includes('Heater') || device.type.includes('Oven') || device.type.includes('Water Dispenser') || device.type.includes('Rice Cooker') || d
                    <%= device.temperature %><span>°C</span>
                <% } %>
                <% if( device.type.includes('Electronic Fan') || device.type.includes('Air Purifier')){ %>
                    <%= device.speed %><span style="font-size: 18px;">rad/s</span>
                <% } %>
                <% if( device.type.includes("Laptop") || device.type.includes("computer") || device.type.includes("Camera")){ %>
                    <% if(device.on_of) { %>
                        <span style="font-size: 18px;background: rgb(34, 59, 24);border-radius:10px;padding:0.5rem 0.8rem;color: rgb(157, 255, 0);">Device is ON</span>
                    <% }else { %>
                        <span style="font-size: 18px;background: rgb(59, 24, 24);border-radius:10px;padding:0.5rem 0.8rem;color: rgb(255, 17, 0);">Device is OFF</span>
                    <% } %>
                <% } %>
            </div>
```

Depending on the type of device, the display will show the specific preset conditions of the device in the grid list. For instance, an air-conditioner would be able to show it's temperature, device on/off status and location indication.

R4B: Display data related to chosen device in database to include name of device, and adjusted settings. Also displays message if device not found:

The grid-list displays the list of added device to the database prior, and since there is a dashboard available for viewing for added devices, a display message indicating that device is not found is not needed.

R4C: Hide input fields not related to device for settings:

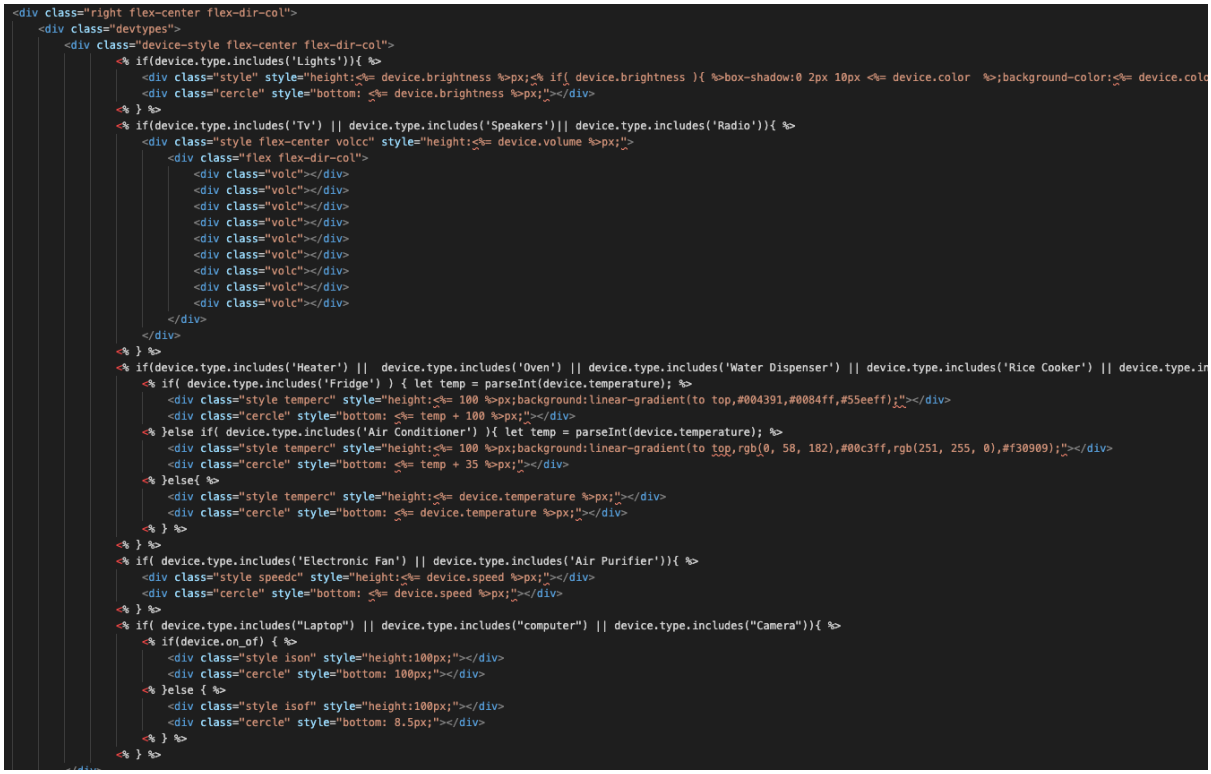
The code extract shown in R4A also shows a bunch of “if” conditions for specific device types and thus, the fields for certain settings like “temperature” and “speed” are only displayed on applicable devices.

The code extract below further shows the input of different kinds of integer values for temperature in celcius and speed in percentage.

```
    </div>
    <div class="device-desc">
        <% if(device.type.includes('Lights')){ %>
            brightness in %
        <% } %>
        <% if(device.type.includes('Tv') || device.type.includes('Radio')){ %>
            selected channel : <%= device.channel %><br>
        <% } %>
        <% if(device.type.includes('Tv') || device.type.includes('Speakers')|| device.type.includes('Radio')){ %>
            volume in %
        <% } %>
        <% if(device.type.includes('Heater') ||  device.type.includes('Oven') || device.type.includes('Water Dispenser')
            temperature in °C
        <% } %>
        <% if( device.type.includes('Electronic Fan') || device.type.includes('Air Purifier')){ %>
            speed in %
        <% } %>
    </div>
```

R4D: Display graphical display of device settings like a dashboard:

The code extract is found further down in “views” > “show-device.ejs”. It templates the devices added to the list as a graphical display like a dashboard. Also note that certain fields are only applicable to certain devices with the specific attributes (temperature, speed etc.).



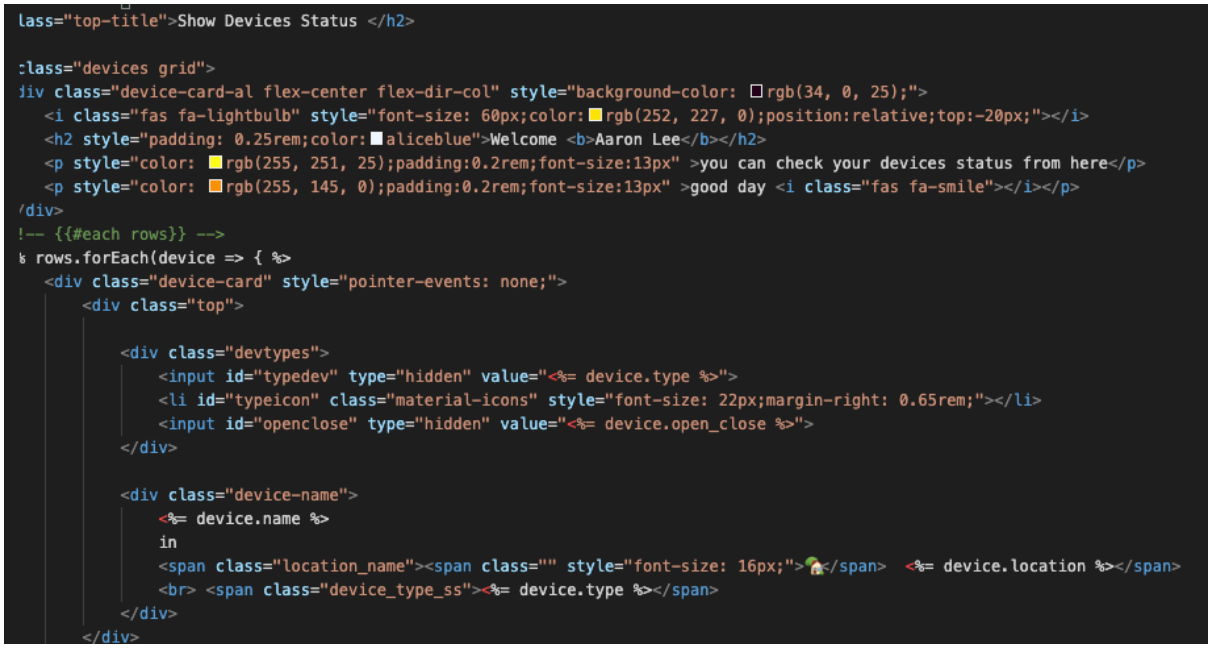
```
<div class="right flex-center flex-dir-col">
    <div class="devtypes">
        <div class="device-style flex-center flex-dir-col">
                <% if(device.type.includes('Lights')){ %>
                    <div class="style" style="height:<%= device.brightness %>px;<% if( device.brightness ){ %>box-shadow:0 2px 10px <%= device.color  %>;background-color:<%= device.colo
                    <div class="cercle" style="bottom: <%= device.brightness %>px;"></div>
                <% } %>
                <% if(device.type.includes('Tv') || device.type.includes('Speakers')|| device.type.includes('Radio')){ %>
                    <div class="style flex-center volcc" style="height:<%= device.volume %>px;">
                        <div class="flex flex-dir-col">
                            <div class="volc"></div>
                            <div class="volc"></div>
                            <div class="volc"></div>
                            <div class="volc"></div>
                            <div class="volc"></div>
                            <div class="volc"></div>
                            <div class="volc"></div>
                            <div class="volc"></div>
                            <div class="volc"></div>
                        </div>
                    </div>
                <% } %>
                <% if(device.type.includes('Heater') ||  device.type.includes('Oven') || device.type.includes('Water Dispenser') || device.type.includes('Rice Cooker') || device.type.in
                    <% if( device.type.includes('Fridge') ) { let temp = parseInt(device.temperature); %>
                        <div class="style temperc" style="height:<%= 100 %>px;background:linear-gradient(to top,#004391,#0084ff,#55eeff);"></div>
                        <div class="cercle" style="bottom: <%= temp + 100 %>px;"></div>
                    <% }else if( device.type.includes('Air Conditioner') ){ let temp = parseInt(device.temperature); %>
                        <div class="style temperc" style="height:<%= 100 %>px;background:linear-gradient(to top,rgb(0, 58, 182),#00c3ff,rgb(251, 255, 0),#f30909);"></div>
                        <div class="cercle" style="bottom: <%= temp + 35 %>px;"></div>
                    <% }else{ %>
                        <div class="style temperc" style="height:<%= device.temperature %>px;"></div>
                        <div class="cercle" style="bottom: <%= device.temperature %>px;"></div>
                    <% } %>
                <% } %>
                <% if( device.type.includes('Electronic Fan') || device.type.includes('Air Purifier')){ %>
                    <div class="style speedc" style="height:<%= device.speed %>px;"></div>
                    <div class="cercle" style="bottom: <%= device.speed %>px;"></div>
                <% } %>
                <% if( device.type.includes("Laptop") || device.type.includes("computer") || device.type.includes("Camera")){ %>
                    <% if(device.on_of) { %>
                        <div class="style ison" style="height:100px;"></div>
                        <div class="cercle" style="bottom: 100px;"></div>
                    <% }else { %>
                        <div class="style isof" style="height:100px;"></div>
                        <div class="cercle" style="bottom: 8.5px;"></div>
                    <% } %>
                <% } %>
```

The design of the layout is done in the “dashboard.css” file, where each device is shown as a card with their specific settings shown.

```css
.grid-select
{
    display: grid;
    grid-template-columns: repeat(auto-fit,minmax(200px,200px));
    gap: 15px;
}

.devices
{

}

.device-card,.device-card-al
{
    position: relative;
    height: 225px;
    padding: 15px;
    border-radius: 0.5rem;
    cursor: pointer;
    overflow: hidden;
    box-shadow: 0 12px 20px rgba(0, 0, 0, 0.13);
    transition: 0.6s cubic-bezier(0.075, 0.82, 0.165, 1);
}

.grid-select .device-card
{
    pointer-events:visible;
}

.grid-select .device-card:hover{
    background-color: #2196F3;
    color: rgb(255, 255, 255);
}

.material-icons
{
    transition: 0.6s cubic-bezier(0.165, 0.84, 0.44, 1);
}
```

R5A: Display list of added devices:

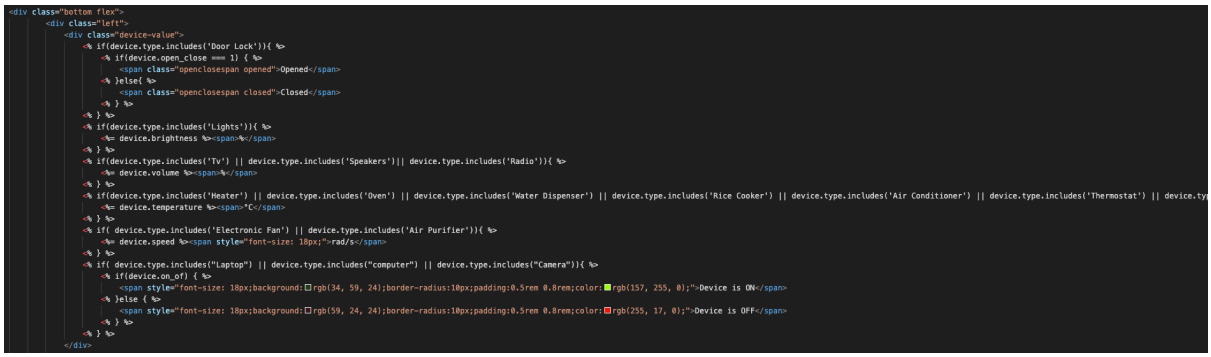
The code extract below is found in “show-device.ejs”, that formats the list of added devices in a grid graphic format.



```
lass="top-title">Show Devices Status </h2>

:lass="devices grid">
iiv class="device-card-al flex-center flex-dir-col" style="background-color: rgb(34, 0, 25);">
   <i class="fas fa-lightbulb" style="font-size: 60px;color: rgb(252, 227, 0);position:relative;top:-20px;"></i>
   <h2 style="padding: 0.25rem;color: aliceblue">Welcome <b>Aaron Lee</b></h2>
   <p style="color: rgb(255, 251, 25);padding:0.2rem;font-size:13px" >you can check your devices status from here</p>
   <p style="color: rgb(255, 145, 0);padding:0.2rem;font-size:13px" >good day <i class="fas fa-smile"></i></p>
/div>
!-- {{#each rows}} -->
% rows.forEach(device => { %>
   <div class="device-card" style="pointer-events: none;">
      <div class="top">

         <div class="devtypes">
            <input id="typedev" type="hidden" value="<%= device.type %>">
            <li id="typeicon" class="material-icons" style="font-size: 22px;margin-right: 0.65rem;"></li>
            <input id="openclose" type="hidden" value="<%= device.open_close %>">
         </div>

         <div class="device-name">
            <%= device.name %>
            in
            <span class="location_name"><span class="" style="font-size: 16px;">🏠</span>  <%= device.location %></span>
            <br> <span class="device_type_ss"><%= device.type %></span>
         </div>
      </div>
```

R5B: Display data relevant to selected device, and allow users to update device settings:

The code extract below displays the data(settings) to specific added devices.



```
<div class="bottom flex">
      <div class="left">
         <div class="device-value">
            <% if(device.type.includes('Door Lock')){ %>
               <% if(device.open_close === 1) { %>
                  <span class="openclosespan opened">Opened</span>
               <% }else{ %>
                  <span class="openclosespan closed">Closed</span>
               <% } %>
            <% } %>
            <% if(device.type.includes('Lights')){ %>
               <%= device.brightness %><span>%</span>
            <% } %>
            <% if(device.type.includes('Tv') || device.type.includes('Speakers')|| device.type.includes('Radio')){ %>
               <%= device.volume %><span>%</span>
            <% } %>
            <% if(device.type.includes('Heater') || device.type.includes('Oven') || device.type.includes('Water Dispenser') || device.type.includes('Rice Cooker') || device.type.includes('Air Conditioner') || device.type.includes('Thermostat') || device.ty
               <%= device.temperature %><span>°C</span>
            <% } %>
            <% if( device.type.includes('Electronic Fan') || device.type.includes('Air Purifier')){ %>
               <%= device.speed %><span style="font-size: 18px;">rad/s</span>
            <% } %>
            <% if( device.type.includes("Laptop") || device.type.includes("computer") || device.type.includes("Camera")){ %>
               <% if(device.on_of) { %>
                  <span style="font-size: 18px;background: rgb(34, 59, 24);border-radius:10px;padding:0.5rem 0.8rem;color: rgb(157, 255, 0);">Device is ON</span>
               <% }else { %>
                  <span style="font-size: 18px;background: rgb(59, 24, 24);border-radius:10px;padding:0.5rem 0.8rem;color: rgb(255, 17, 0);">Device is OFF</span>
               <% } %>
            <% } %>
         </div>
```

R5C: Hide input fields not applicable to selected device:

The code extract below shows conditional “if”s being stated for different types of devices, as different category of devices have unique settings, like “temperature”.

```
<div class="devtypes">
    <div class="device-style flex-center flex-dir-col">
        <% if(device.type.includes('Lights')){ %>
            <div class="style" style="height:<%= device.brightness %>px;<% if( device.brightness ){ %>box-shadow:0 2px 10px <%= device.color  %>;background-color:<%= device.color  %><% } %>"></div>
            <div class="cercle" style="bottom: <%= device.brightness %>px;"></div>
        <% } %>
        <% if(device.type.includes('Tv') || device.type.includes('Speakers')|| device.type.includes('Radio')){ %>
            <div class="style flex-center volcc" style="height:<%= device.volume %>px;">
                <div class="flex flex-dir-col">
                    <div class="volc"></div>
                    <div class="volc"></div>
                    <div class="volc"></div>
                    <div class="volc"></div>
                    <div class="volc"></div>
                    <div class="volc"></div>
                    <div class="volc"></div>
                    <div class="volc"></div>
                    <div class="volc"></div>
                </div>
            </div>
        <% } %>
        <% if(device.type.includes('Heater') || device.type.includes('Oven') || device.type.includes('Water Dispenser') || device.type.includes('Rice Cooker') || device.type.includes('Air Conditioner') || device.type.includes('Thermost
            <% if( device.type.includes('Fridge') ) { let temp = parseInt(device.temperature); %>
                <div class="style temperc" style="height:<%= 100 %>px;background:linear-gradient(to top,#004391,#0084ff,#55eeff);"></div>
                <div class="cercle" style="bottom: <%= temp + 100 %>px;"></div>
            <% }else if( device.type.includes('Air Conditioner') ){ let temp = parseInt(device.temperature); %>
                <div class="style temperc" style="height:<%= 100 %>px;background:linear-gradient(to top,rgb(0, 58, 182),#00c3ff,rgb(251, 255, 0),#f30909);"></div>
                <div class="cercle" style="bottom: <%= temp + 35 %>px;"></div>
            <% }else{ %>
                <div class="style temperc" style="height:<%= device.temperature %>px;"></div>
                <div class="cercle" style="bottom: <%= device.temperature %>px;"></div>
            <% } %>
        <% } %>
        <% if( device.type.includes('Electronic Fan') || device.type.includes('Air Purifier')){ %>
            <div class="style speedc" style="height:<%= device.speed %>px;"></div>
            <div class="cercle" style="bottom: <%= device.speed %>px;"></div>
        <% } %>
        <% if( device.type.includes("Laptop") || device.type.includes("computer") || device.type.includes("Camera")){ %>
            <% if(device.on_of) { %>
                <div class="style ison" style="height:100px;"></div>
                <div class="cercle" style="bottom: 100px;"></div>
            <% }else { %>
                <div class="style isof" style="height:100px;"></div>
                <div class="cercle" style="bottom: 8.5px;"></div>
            <% } %>
        <% } %>
    </div>
</div>
```

R5D: Allow user to update device settings graphical display of device settings on a dashboard:

By clicking on the device in the graphical grid list, it will being the user to an “update device” form, which allows the user to adjust the settings of the device. The “update-device.ejs” file, which is the “Update Device” page on my web application, will link the user to another page, and the contents can be found in “update-device-form.ejs”.

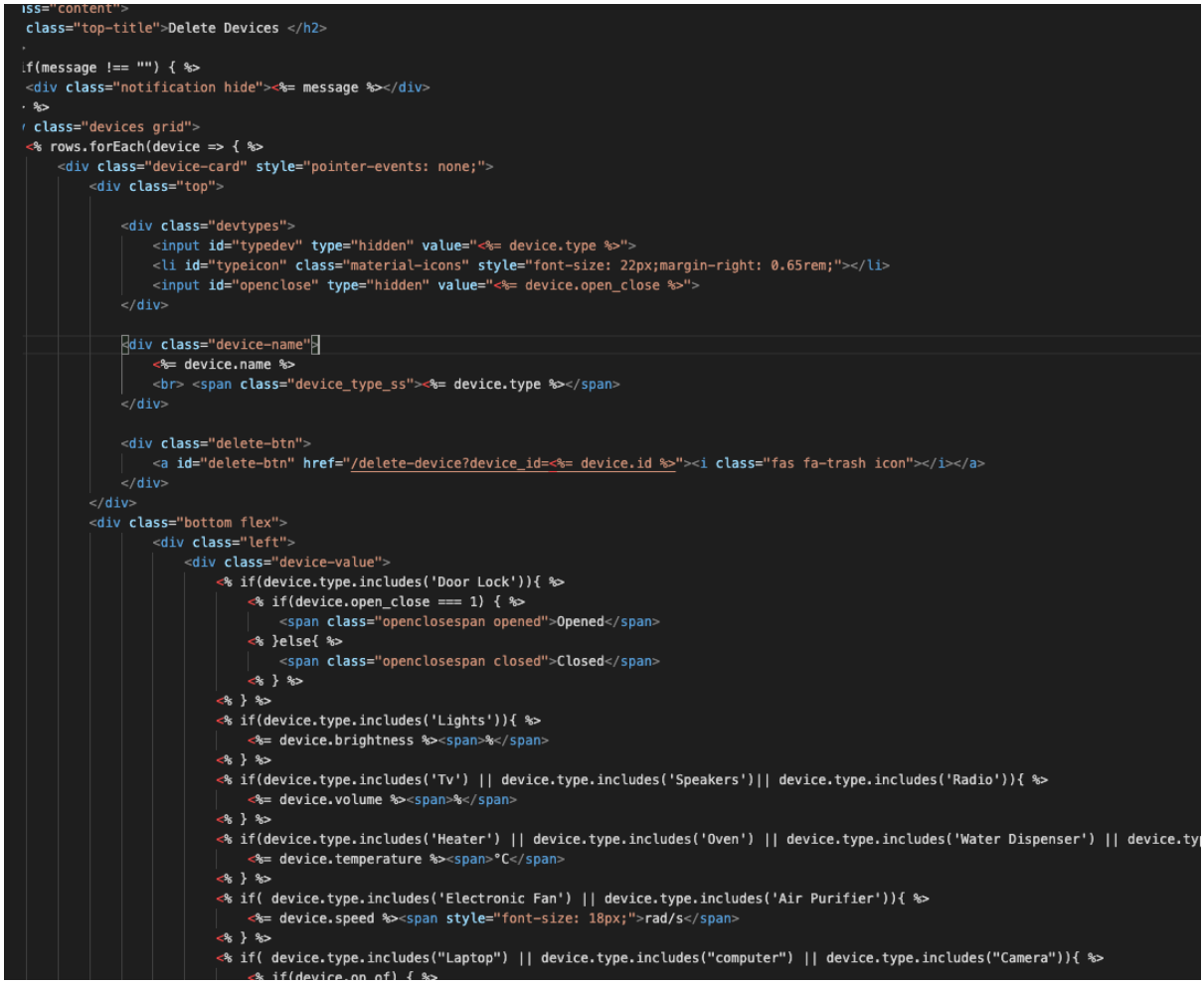
```
<div class="devices grid">
    <% rows.forEach(device => { %>
        <div class="device-card" style="pointer-events: none;">
            <div class="top">

                <div class="devtypes">
                    <input id="typedev" type="hidden" value="<%= device.type %>">
                    <li id="typeicon" class="material-icons" style="font-size: 22px;margin-right: 0.65rem;"></li>
                    <input id="openclose" type="hidden" value="<%= device.open_close %>">
                </div>

                <div class="device-name">
                    <%= device.name %>
                    <br> <span class="device_type_ss"><%= device.type %></span>
                </div>

                <div class="update-btn">
                    <a id="update-btn" href="/update-device-form?device_id=<%= device.id %>"><i class="fas fa-edit icon"></i></a>
                </div>
            </div>
```

So similar to when the user first adds the device to the database via a form, this form is the form used to input the details of the selected device's settings. (code extract from "update-device-form.ejs")

```
<div class="content">
    <h2 class="top-title">Update Devices </h2>
    <br><br>
    <h2>Update smart home devices : </h2>
    <br>
    <div class="devices-form">
        <form action="/update-device" method="post">
            <div class="form-input name flex">
                <h3 class="form-title">you're updating the <%= row.name %> smart device in your house : </h3>
                <input id="device_type" type="hidden" name="type" value="<%= row.type%>">
                <input id="device_id" type="hidden" name="device_id" value="<%= row.id %>">
            </div>
            <div class="form-input name flex">
                <label class="label" for="">Device Name : </label>
                <div class="flex">
                    <input type="text" name="dev_name" value="<%= row.name %>" required>
                </div>
            </div>
            <div class="form-input flex">
                <label class="label" for="onof">status : </label>
                <div class="flex-center">
                    <span style="margin-right: 8px;font-size:14px;">Off </span>
                    <label class="switch" >
                        <% if(row.on_of) { %>
                        <input type="checkbox" name="onof" id="onof" value="1" checked>
                        <% }else { %>
                        <input type="checkbox" name="onof" id="onof" value="0">
                        <% } %>
                        <span class="slider round"></span>
                    </label>
                    <span  style="margin-left: 8px;font-size:14px;"> On</span>
                </div>
            </div>
```

R6A: Display list of added devices and allow user to delete selected device:

Code extract below is from “delete-device.ejs”, which displays the content of the devices active on the “SmartHome” web application, allowing the user to select them to delete them from the list of devices.



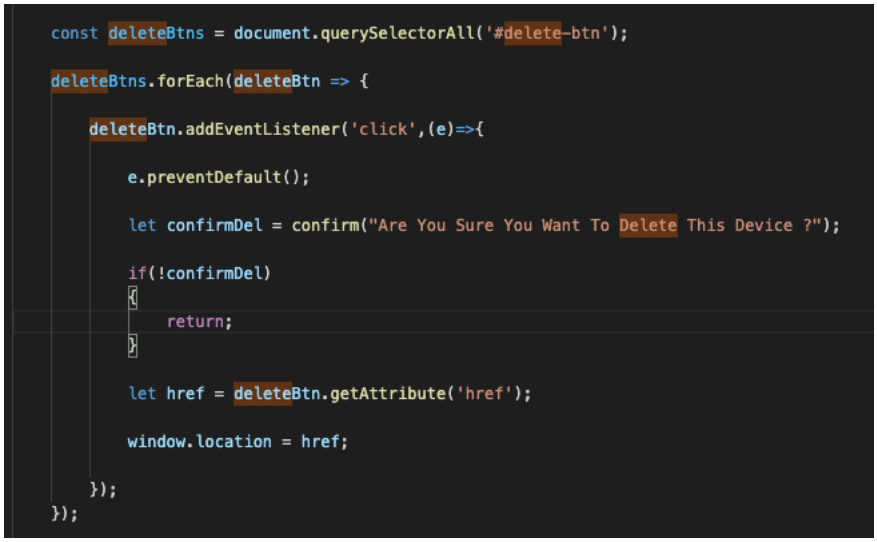
```
iss="content">
 class="top-title">Delete Devices </h2>

if(message !== "") { %>
 <div class="notification hide"><%= message %></div>
 %>
 class="devices grid">
 <% rows.forEach(device => { %>
    <div class="device-card" style="pointer-events: none;">
        <div class="top">

            <div class="devtypes">
                <input id="typedev" type="hidden" value="<%= device.type %>">
                <li id="typeicon" class="material-icons" style="font-size: 22px;margin-right: 0.65rem;"></li>
                <input id="openclose" type="hidden" value="<%= device.open_close %>">
            </div>

            <div class="device-name">
                <%= device.name %>
                <br> <span class="device_type_ss"><%= device.type %></span>
            </div>

            <div class="delete-btn">
                <a id="delete-btn" href="/delete-device?device_id=<%= device.id %>"><i class="fas fa-trash icon"></i></a>
            </div>
        </div>
        <div class="bottom flex">
                <div class="left">
                    <div class="device-value">
                        <% if(device.type.includes('Door Lock')){ %>
                            <% if(device.open_close === 1) { %>
                                <span class="openclosespan opened">Opened</span>
                            <% }else{ %>
                                <span class="openclosespan closed">Closed</span>
                            <% } %>
                        <% } %>
                        <% if(device.type.includes('Lights')){ %>
                            <%= device.brightness %><span>%</span>
                        <% } %>
                        <% if(device.type.includes('Tv') || device.type.includes('Speakers')|| device.type.includes('Radio')){ %>
                            <%= device.volume %><span>%</span>
                        <% } %>
                        <% if(device.type.includes('Heater') || device.type.includes('Oven') || device.type.includes('Water Dispenser') || device.ty
                            <%= device.temperature %><span>°C</span>
                        <% } %>
                        <% if( device.type.includes('Electronic Fan') || device.type.includes('Air Purifier')){ %>
                            <%= device.speed %><span style="font-size: 18px;">rad/s</span>
                        <% } %>
                        <% if( device.type.includes("Laptop") || device.type.includes("computer") || device.type.includes("Camera")){ %>
                            <% if(device.on_of) { %>
```

The code extract below is from “main.js”, where the user is redirected to the delete device page and deletes the device from the database, that was added by the user before the web application, by using the ID given to the device as an item reference.

```
// redirect to the delete-device page

router.get('/delete-device',(req,res) => {

    pool.getConnection((err,connection) => {

        if(err) throw err;

        let device_id = req.query.device_id;
        let message = "";

        // delete device by the giving device id collected

        if(device_id !== undefined)
        {

            let query = "DELETE FROM devices WHERE id = ?";

            connection.query(query,[device_id],(err,result)=>{

                if(err) throw err;
                message = "Device Successfully Deleted !";
            });
        }

        let query = "SELECT * FROM devices";

        connection.query(query,(err,rows)=>{

            connection.release();

            if(!err)
            {
                res.render('delete-device',{rows,message});
            }
            else
            {
                console.log("error");
            }
        });
    });
});
```

R6B: Seek user's confirmation before deleting device from database:

The code extract below is from "app.js", that shows a confirmation message, prompting the user to confirm that he wants to delete the device from the active list of saved devices in the "SmartHome" web application.



```
const deleteBtns = document.querySelectorAll('#delete-btn');

deleteBtns.forEach(deleteBtn => {

    deleteBtn.addEventListener('click',(e)=>{

        e.preventDefault();

        let confirmDel = confirm("Are You Sure You Want To Delete This Device ?");

        if(!confirmDel)
        {
            return;
        }

        let href = deleteBtn.getAttribute('href');

        window.location = href;

    });
});
```

Disclaimer/ Issues faced:

The lab environment and server has some issues with SQL and browser previews, hence the coding was done locally, and on another IDE environment, then subsequently imported to the lab environment for sharing. All errors are resolved in my local environment and I've managed to get the program to work, however, the same success is unable to be translated to the Cousera lab environment, due to some lags, bugs and errors with the environment.

The browser previews do not show the images/ icons that I've included for the interpretations of various devices and their statuses and adjustments to be made. There are also some difficulties experienced when launching the SQL in the terminal occasionally. Also, there are some issues I faced after filling up the form to input details, after selecting the type of device to be added to the web application, as I am unable to load the next page after submitting the form and the notification message does not appear.