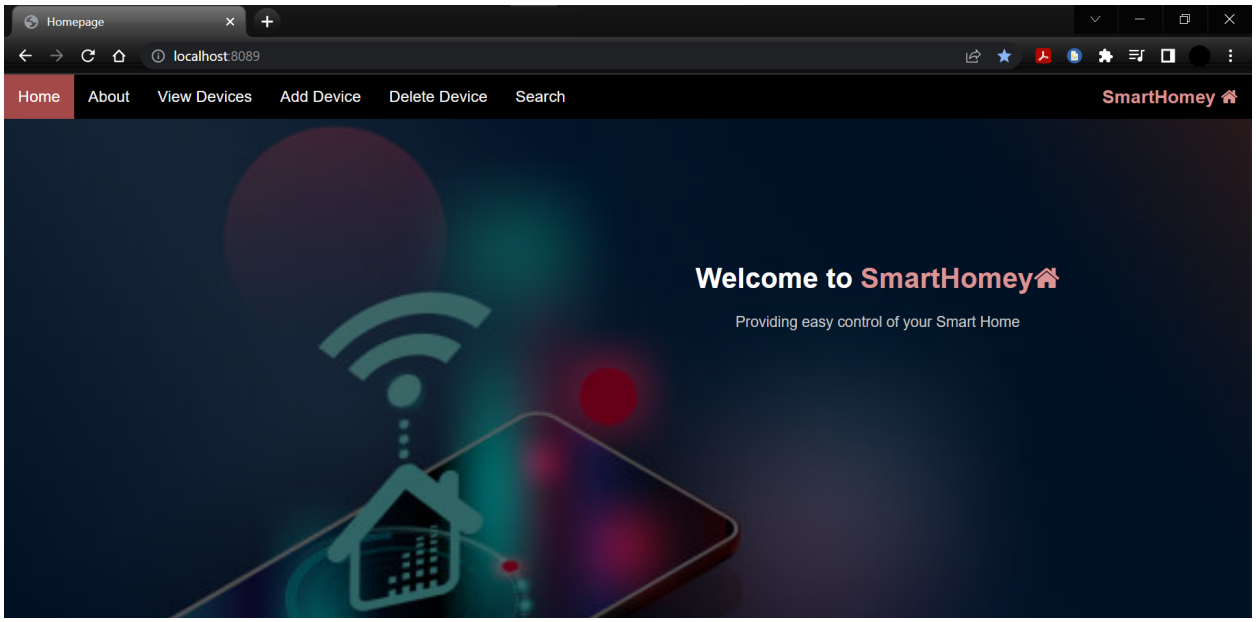
CM2040 Databases, Networks and the Web
Mid-Term Coursework

Highlight achievements

Home Page (index.html)

Homepage displays the name and branding for the homepage.



Part of index.html

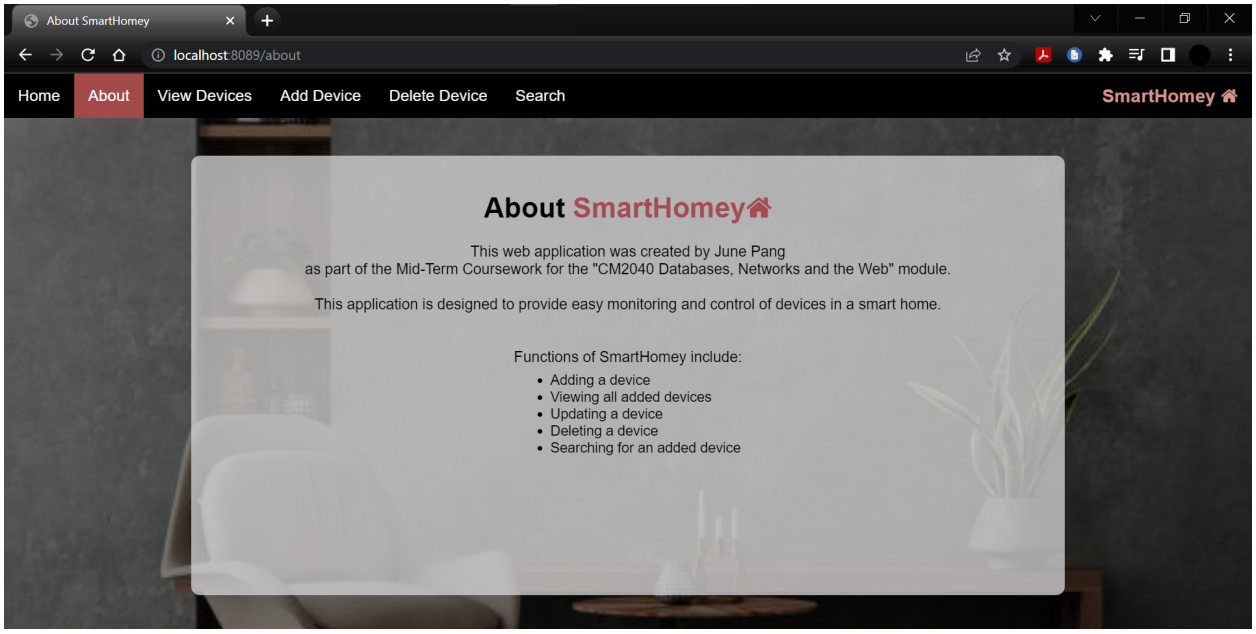
```
<!-- Main content of the page -->
<div class="split right" style="padding-top: 10%; color: white;">
  <h1>Welcome to <b style="color: #db9494">SmartHomey<i class="fa fa-home"></i></b></h1>
  <p style="color: rgb(197, 197, 197);">Providing easy control of your Smart Home</p>
</div>
```

The index.html page is rendered to display the homepage.

```
// Display the homepage (index.html)
app.get("/", function (req, res) {
  res.render("index.html")
});
```

About Page

The about page provides brief information about the web application.



Part of **about.html** displaying the main content of the about page.

```
<div class="about-container" >
 <div class="content">
   <h1> About <span style="color:#a44b54">SmartHomey<i class="fa fa-home"></i></span></h1>
   <p>This web application was created by June Pang <br>
     as part of the Mid-Term Coursework for the "CM2040 Databases, Networks and the Web" module.<br>
     <br>This application is designed to provide easy monitoring and control of devices in a smart home.

     <br>
     Functions of SmartHomey include:
     <ul>
       <li><a href="/addDevice">Adding a device</a></li>
       <li><a href="/viewDevices">Viewing all added devices</a></li>
       <li><a href="/viewDevices">Updating a device</a></li>
       <li><a href="/deleteD">Deleting a device</a></li>
       <li><a href="/search">Searching for an added device</a></li>
     </ul>
   </p>
 </div>
```

The about.html page is rendered to display the about page.

```
// Display the about page (about.html)
app.get("/about", function (req, res) {
  res.render("about.html")
});
```

Add Device Page (addDevice.html)

The add device form consists of 6 fields. The form can only be submitted after all required fields have been filled up.

```
// Display the Add device page (addDevice.html)
app.get("/addDevice", function (req, res) {
  res.render("addDevice.html");
});
```

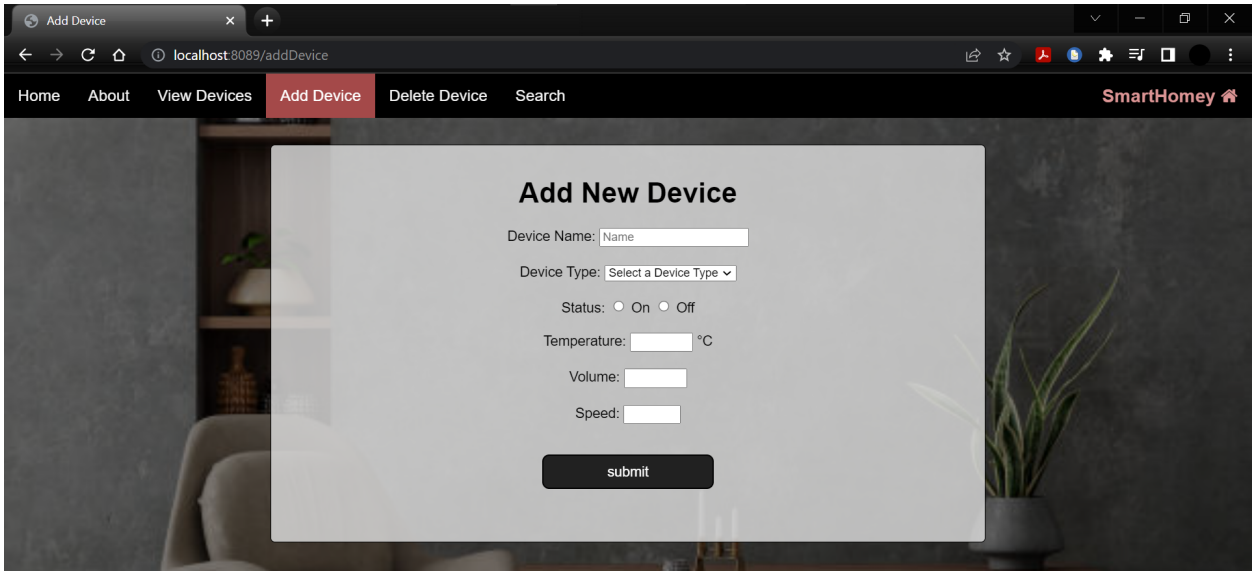
A POST action is used to pass the form input to the server.

```
<!-- ask the user to enter device name, device type, status, temperature, volume, speed -->
<form method="POST" action="/addDevice">

  <!-- Name of Device (maximum 50 characters, special characters not allowed) -->
  <label> Device Name: </label>
  <input type="text" id="first" name="name" placeholder="Name" maxlength="50" required
  pattern="[a-zA-Z0-9 ]{0,50}" oninvalid="this.setCustomValidity('No special characters allowed')" oninpu

  <!-- 6 different pre-defined device types displayed using a dropdown list -->
  <label> Device Type: </label>
```

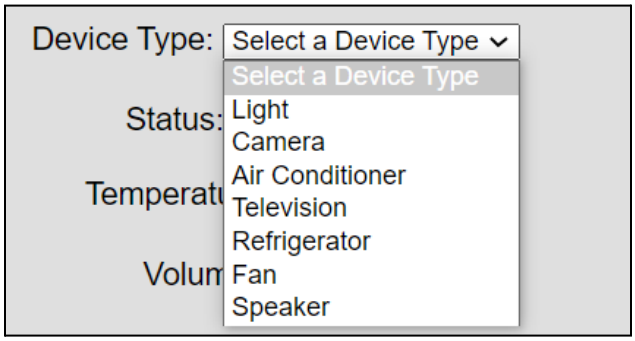
Values with no input passed in from the form are set to null before they are inserted into the database. The form will be re-displayed if the sql query fails to execute.

```
// Insert values passed in from addDevice.html form into the database
app.post("/addDevice", function (req, res) {
  // sql query to insert a new record into the database, based on the values taken from the form in addDevice.html
  let sqlquery = "INSERT INTO devices (device_name, device_type, status, temperature, volume, speed) VALUES (?,?,?,?,?,?)";

  // set value of temperature, volume or speed to null before querying the db
  // if the field is empty (means it is not required for that device type)
  if (req.body.temperature == '') req.body.temperature = null;
  if (req.body.volume == '') req.body.volume = null;
  if (req.body.speed == '') req.body.speed = null;

  // Form inputs taken from addDevice.html (to insert into the database)
  let newrecord = [req.body.name, req.body.devices, req.body.status, req.body.temperature, req.body.volume, req.body.speed];

  // execute sql query
  db.query(sqlquery, newrecord, (err, result) => {
    if (err) { // if the sql query is executed unsuccessfully, redirect back to the "addDevice.html" page
```

(Add device page)

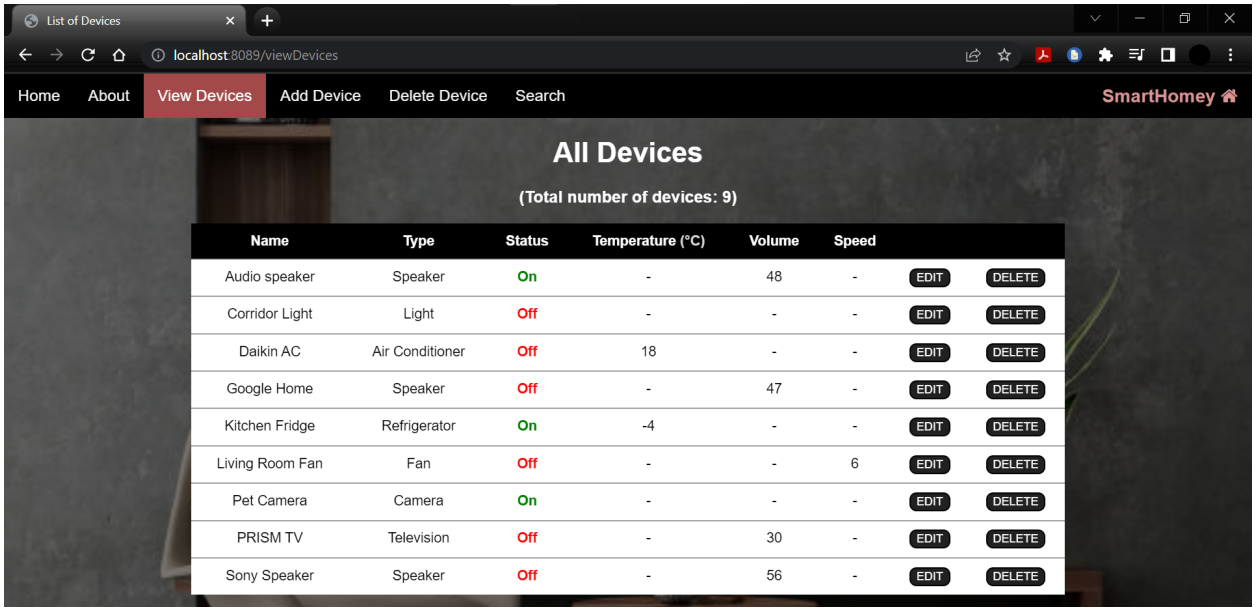


Device type field has a drop-down list with 7 options



All Devices Page (viewDevices.html)
Displays information of all devices added, and a count of the number of devices in the table. Clicking on each **device name** in the “All Devices” page will direct users to the “device status” page.

(viewDevices.html)



Javascript function to count the number of rows in the table. (code in *public\script\script.js*)

```
/* Counts the total number of rows in a table (not including the table heading)
   | Used in viewDevices.html */
function count() {
  var rowCount = document.getElementById('table1').rows.length;
  document.getElementById("value").innerHTML = rowCount - 1;
}
```

Function used in viewDevices.html

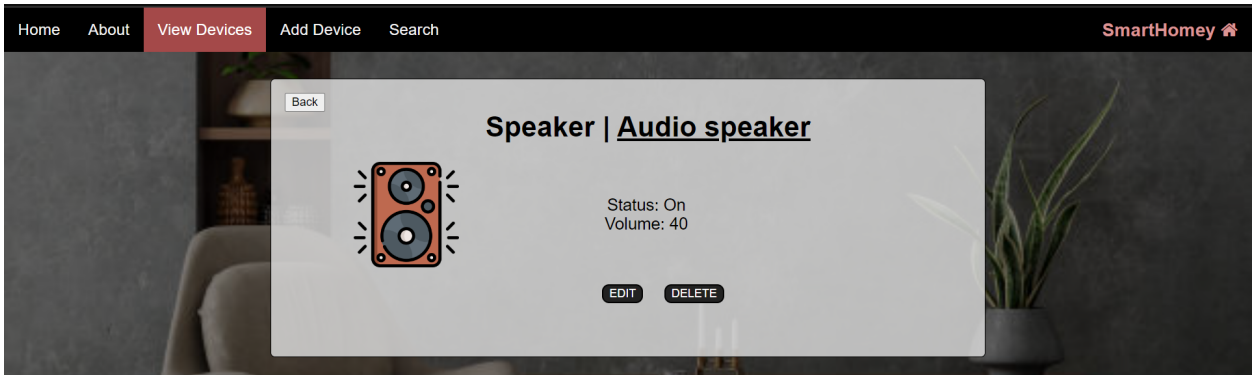
```
<!-- Execute the count() function after the page loads -->
<body class="bg2" onload="count()">
```

Device name is linked to the Device status page for that specific device.

```
<!-- Display device name, and link to the view.html page displaying the device settings of that device -->
<td>
  <a href="/view?name=<%= device.device_name %>">
    <%= device.device_name %>
  </a>
</td>
```

Device Status Page (view.html)
Displays information for that specific device.

(view.html page for device ‘Audio speaker’)




```
/** Get device values for a specific device from the database to display in the device status page (view.html) */
app.get("/view", function (req, res) {
  // sql statement to get records where the device name matches req.query.name
  let sqlquery = "SELECT * FROM devices WHERE device_name = ?";
  // execute sql query
  db.query(sqlquery, req.query.name, (err, result) => {
    if (err) {
      res.redirect("/viewDevices");
    }
    // If successful, pass the result as a parameter to view.html page
    res.render("view.html", { availableDevices: result });
  });
```

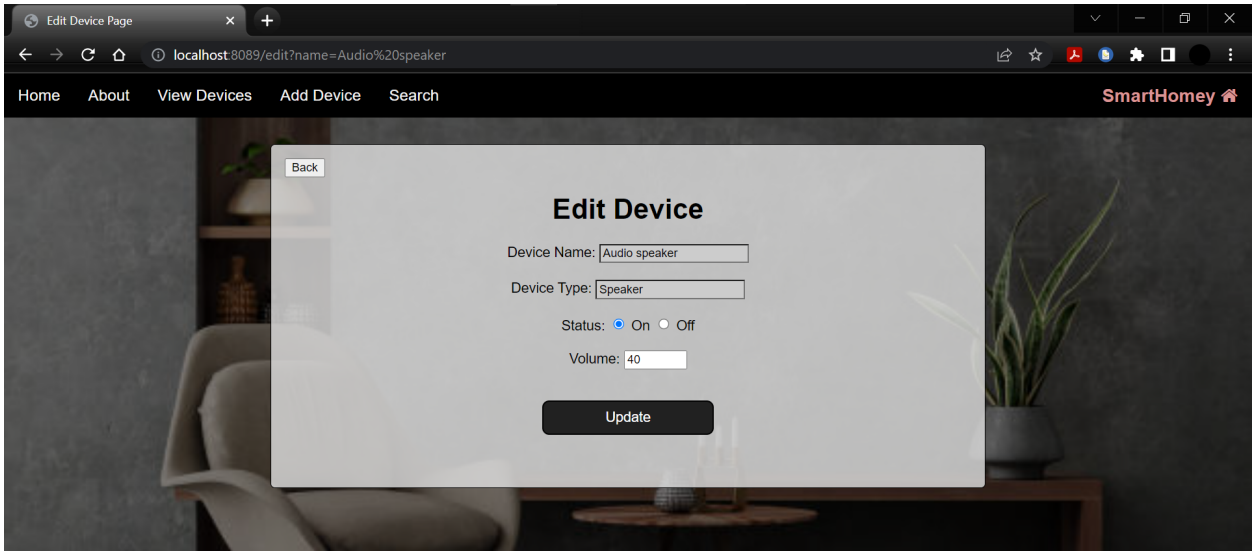
Different image and fields displayed according to the type of device

```
<!-- Display different images depending on the device type. All image
<% if (device.device_type == 'Light') {%>
  <img src="https://cdn-icons.flaticon.com/png/512/2988/premium/29880
  alt="image of a lightbulb" class="image2">
<% } %>

<% if (device.device_type == 'Camera') {%>
  <img src="https://cdn-icons-png.flaticon.com/512/394/394620.png"
  alt="image of a camera" class="image2">
<% } %>
```

```
<!-- Only display the temperature volume and/or speed if the value is not null -->
<% if (device.temperature != null ) {%>
Temperature: <%= device.temperature %> °C
<br>
<% } %>

<% if (device.volume != null ) {%>
Volume: <%= device.volume %>
<br>
<% } %>

<% if (device.speed != null ) {%>
Speed: <%= device.speed %>
<% } %>
```

Control Device (editDevice.html)

Clicking on the “EDIT” button in either the All Devices page will direct to the edit device page, with the form pre-populated with values from the database. The device name and type cannot be modified in the edit form.



A POST action is used to pass data to the server.

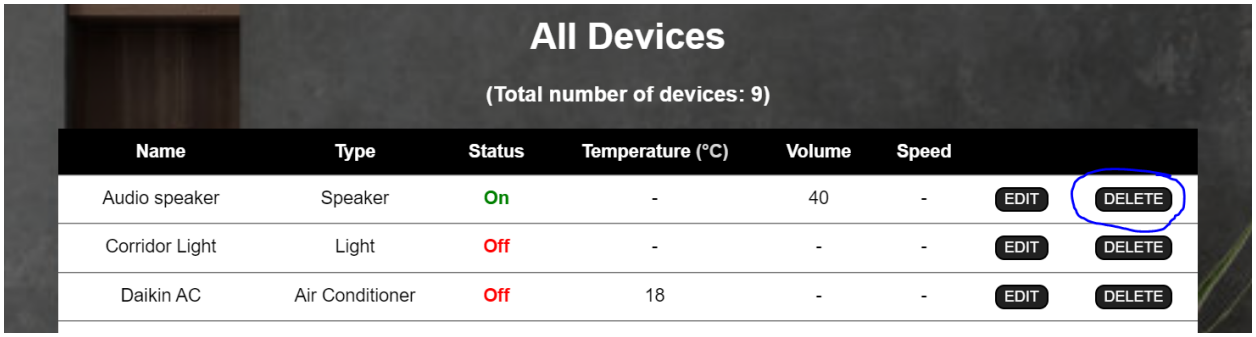
```
// Update the database with values passed in from the editDevice.html form
app.post("/edit", function (req, res) {
  // set value of temperature, volume or speed to null if the field is empty (means it is not required for that device type)
  if (req.body.temperature == '') req.body.temperature = null;
  if (req.body.volume == '') req.body.volume = null;
  if (req.body.speed == '') req.body.speed = null;

  let sqlquery = "UPDATE devices SET device_type = ?, status = ?, temperature = ?, volume = ?, speed = ? WHERE device_name = ?";

  var updated = [
    req.body.devices, req.body.status,
    req.body.temperature, req.body.volume,
    req.body.speed, req.body.name]

  db.query(sqlquery, updated, function(err, result) {
    if (err) {
```

POST endpoint is used to update the database using the device name.

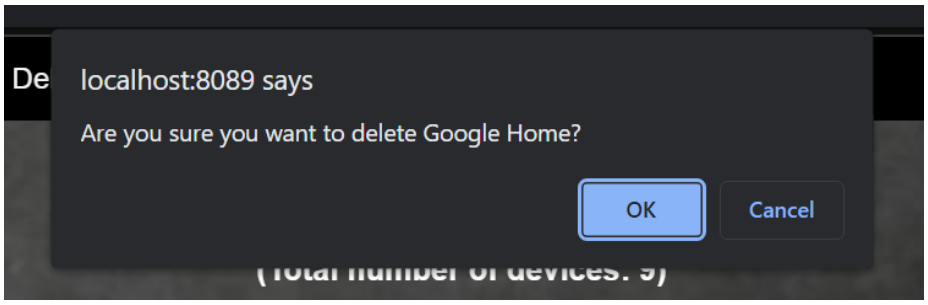
```
<form method="POST" action="/edit">
  <% availableDevices.forEach(function(device){ %>

  <!-- Name of Device (maximum 50 characters, special characters not allowed) -->
  <label> Device Name: </label>
  <input type="text" name="name" value="<%= device.device_name %>" readonly /><br /><br />
```

Delete Device (2 ways)

1. Clicking on the “DELETE” button in the “All Device” page, will show a confirmation message.



(Extension) Confirmation message is displayed after clicking on “DELETE” button. Clicking “OK” will delete the device from the database.

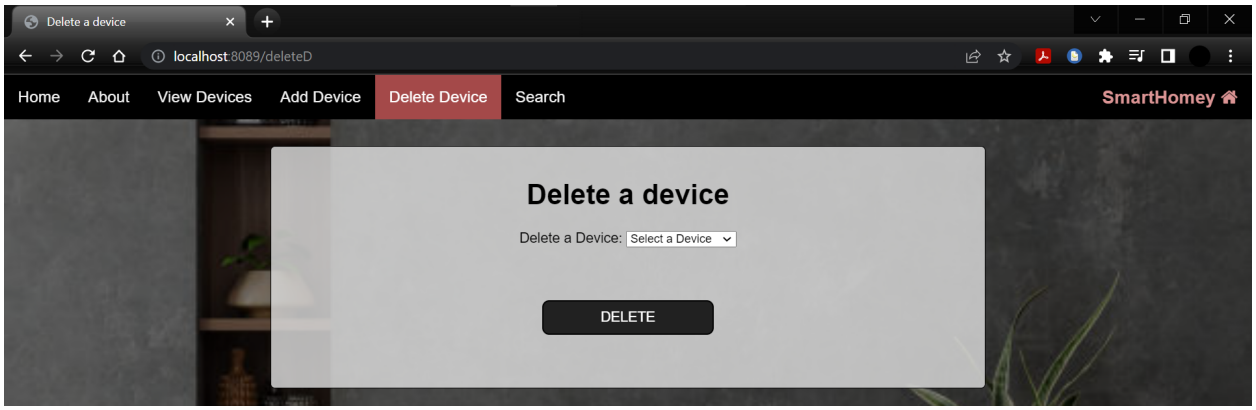



```
<!-- DELETE button -->
<td><a href="/delete?name=<%= device.device_name %>"
  onclick="return confirm('Are you sure you want to delete <%= device.device_name %>?')">
  <button class="button">DELETE</button>
</a></td>
```

```
/** Deleting the device that matches the specific device name
* (after clicking on the delete button in viewDevices.html) */
  app.get("/delete", function (req, res) {
  // sql statement to delete the device with the name of req.query.name
  let sqlquery = "DELETE FROM devices WHERE device_name = ?";

  // query the database using the sql statement above
  db.query(sqlquery, req.query.name, (err, result) => {
    if (err) {
      // redirect back to viewDevices page if the query fails
      res.redirect("/viewDevices");
    }
    // display the deleteSuccess.html page after deleting the device from the database.
    res.render("deleteSuccess.html")
  });
});
```

2. In the “Delete Device” tab on the navigation bar, the delete page has a dropdown list with all added devices. A device can be deleted by selecting that device from the list and clicking delete.

(delete.html)




```
<!-- dropdown list containing all added devices for the user to select a device to delete -->
<form method="POST" action="/deleteD">

  Delete a Device:
  <select name="name">
    <option value="" disabled selected>Select a Device</option>
  <% availableDevices.forEach(function(device){ %>
    <option name="<%= device.device_name %>" value="<%= device.device_name %>"><%= device.device_name %></option>
  <% }) %>
    </select><br>
```

```
/** Delete the device using the name (primary key) of the device */
app.post("/deleteD", function (req, res) {
  // sql statement to delete device information relating to device_name
  let sqlquery = "DELETE FROM devices WHERE device_name = ?";

  // execute sql query
  db.query(sqlquery, req.body.name, function(err, result) {
    if (err) {
      console.log(err);
      res.redirect("/viewDevices"); // redirect to view devices page if unsuccessful
    }
    else { // If successful, render the deleteSuccess.html page
```

Navigation

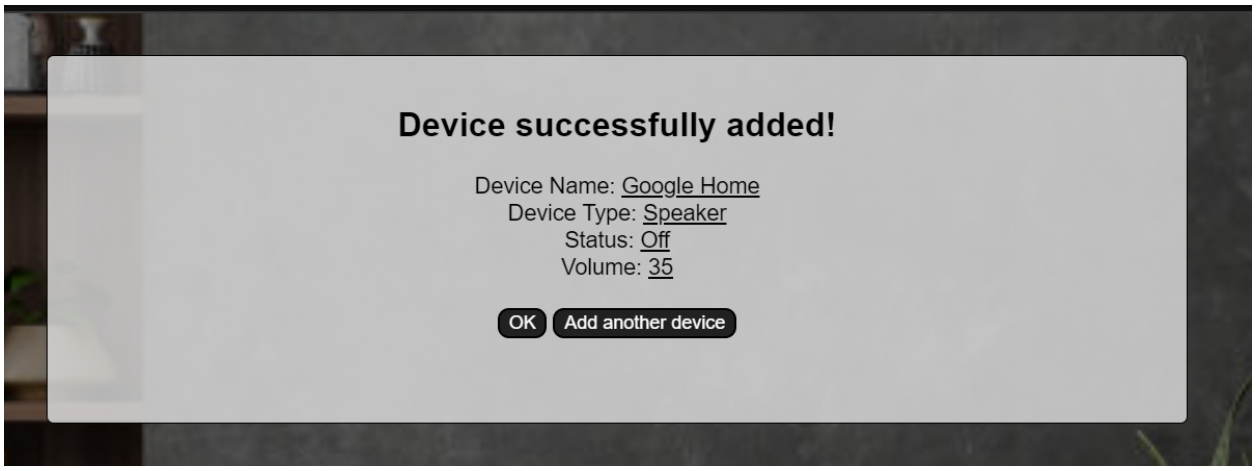
Navigation bar linking pages of the main functions is displayed at the top of every page, with the active tab coloured red.

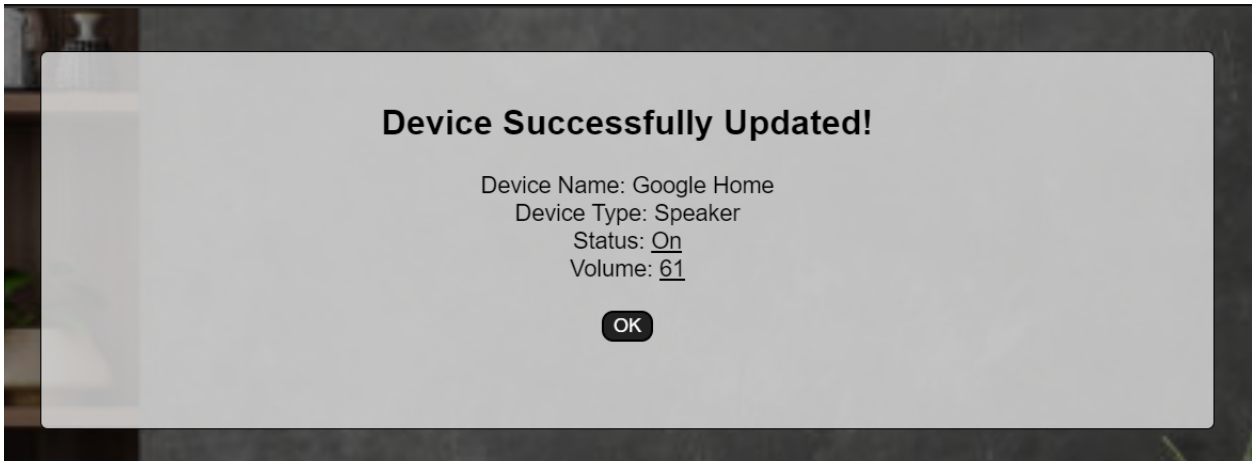

```
<!-- Navigation Bar -->
<div class="navbar" >
  <a href="/" class="active">Home</a>
  <a href="/about">About</a>
  <a href="/viewDevices">View Devices</a>
  <a href="/addDevice">Add Device</a>
  <a href="/search">Search</a>
  <span class = "logo"> SmartHomey <i class="fa fa-home"></i></span>
</div>
```

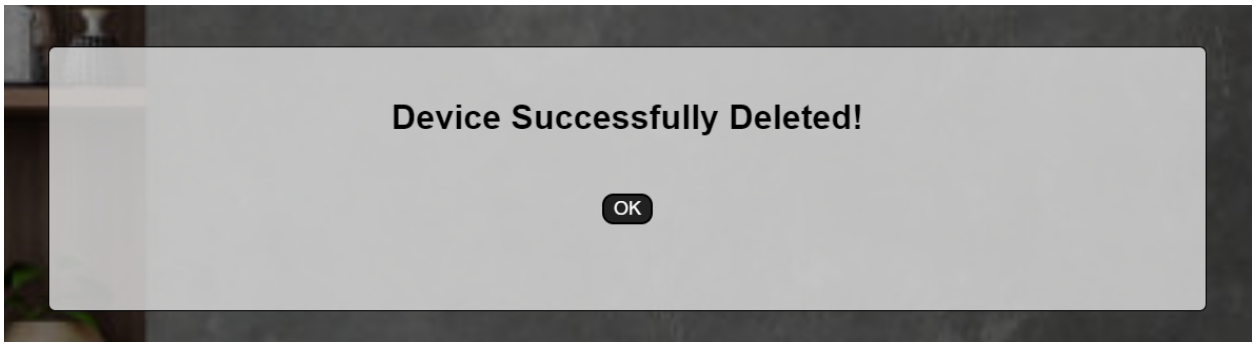
Extensions implemented

- **Success Feedback**

1. Success page displayed after successfully adding, updating, and deleting a device.



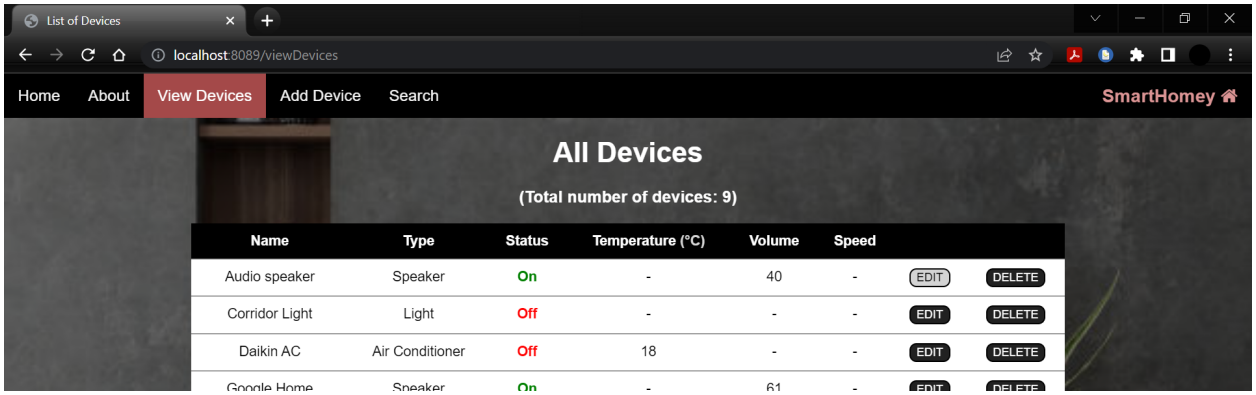




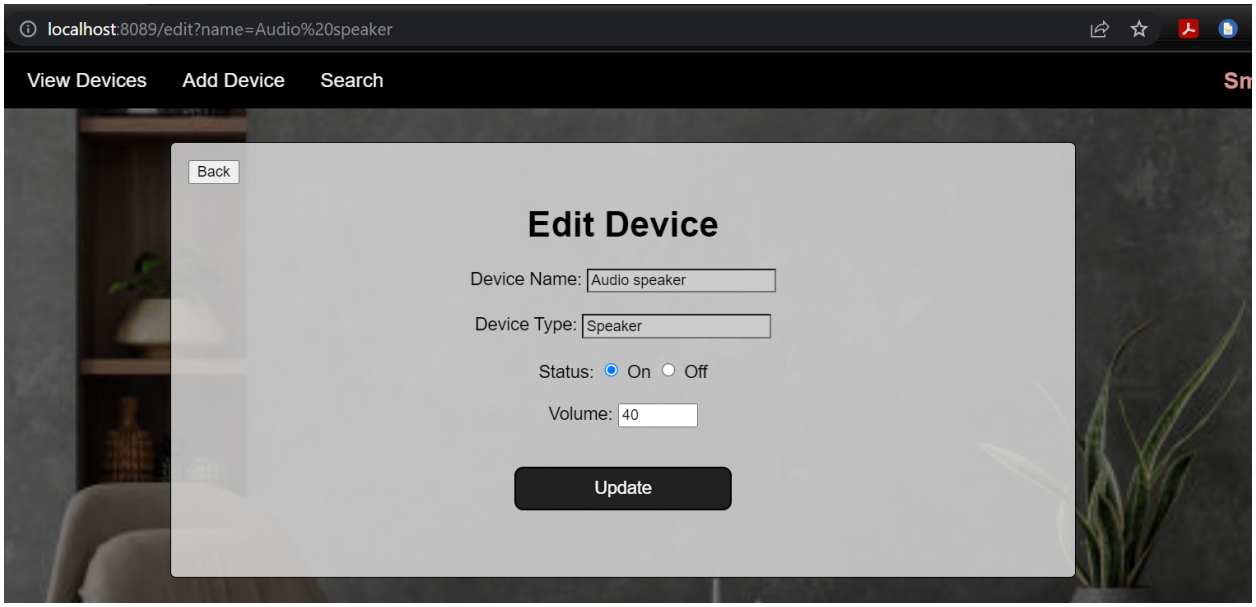
- **Hot Reloading**

1. Update page pre-populated with values from the database.

   (Clicking EDIT for ‘Audio Speaker’)



   Edit Device Page pre-populated with values of ‘Audio Speaker’




```
<label> Device Name: </label>
<input type="text" name="name" value="<%= device.device_name %>" readonly /><br /><br />

<!-- Type of Device (Cannot be edited) -->
<label> Device Type: </label>
<input type="text" name="devices" value="<%= device.device_type %>" readonly /><br /><br />
```

2. After clicking on “OK” after deleting a device in the All Devices page, the device is immediately removed from the table (demonstrated in video).

- **Non-applicable fields disabled**

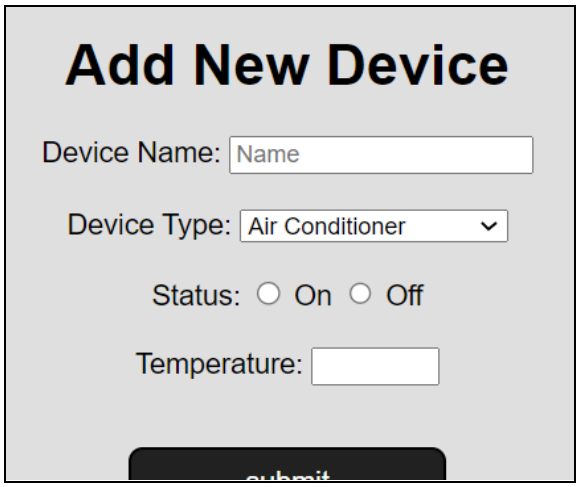
Javascript function used to hide non-applicable fields (filtered based on chosen device type).. This is done by comparing the value passed in, to the 7 different device types available.

```
function showHideInfo(value) {
  var notemp = document.getElementById('temp').style.display = 'none'; // don't display temperature
  var novol = document.getElementById('vol').style.display = 'none'; // don't display volume
  var nospeed = document.getElementById('spd').style.display = 'none'; // don't display speed

  // don't display temperature, volume and speed if device type is light or camera
  if (value == "Light" || value == "Camera") {
    notemp; novol; nospeed;
  }

  // don't display volume and speed if device type is Air Conditioner or Refrigerator
  else if (value == "Air Conditioner" || value == "Refrigerator") {
    novol; nospeed;
    document.getElementById('temp').style.display = 'block'; // display temperature
  }

  // don't display temperature and speed if device type is television or speaker
  else if (value == "Television" || value == "Speaker") {
    notemp; nospeed;
    document.getElementById('vol').style.display = 'block'; // display volume
  }

  // don't display temperature and volume if device type is fan
  else if (value == "Fan") {
    notemp; novol;
    document.getElementById('spd').style.display = 'block'; // display speed
  }

  else { // Otherwise, display everything
    document.getElementById('temp').style.display = 'block'; // temp
    document.getElementById('vol').style.display = 'block'; // vol
    document.getElementById('spd').style.display = 'block'; //speed
  }
```

(Found in *topic7\public\script\script.js*)

This function is used in “Device Type” field in addDevice.html through an onchange attribute. The value of the selected option is passed into the function.

```
<label> Device Type: </label>
<select name="devices" onchange="showHideInfo(this.value)" required>
  <option value="" disabled selected>Select a Device Type</option>
  <option value="Light">Light</option>
```



Non-applicable fields set to null before inserted into database.

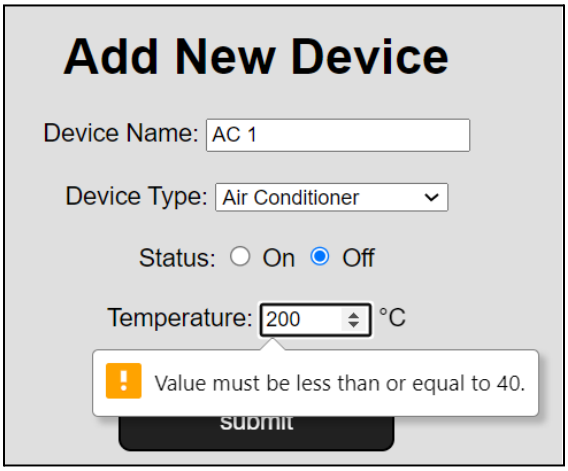
```
// adding a new device into the databases
app.post("/addDevice", function (req, res) {
  // insert a new record into the database, based on the values taken from the
  let sqlquery = "INSERT INTO devices (device_name, device_type, status, temper

  // set value of temperature, volume or speed to null before querying the db
  // if the field is empty (means it is not required for that device type)
  if (req.body.temperature == '') req.body.temperature = null;
  if (req.body.volume == '') req.body.volume = null;
  if (req.body.speed == '') req.body.speed = null;

  // The values to insert into the database
  let newrecord = [req.body.name, req.body.devices, req.body.status, req.body.t

  // execute sql query
  db.query(sqlquery, newrecord, (err, result) => {
```

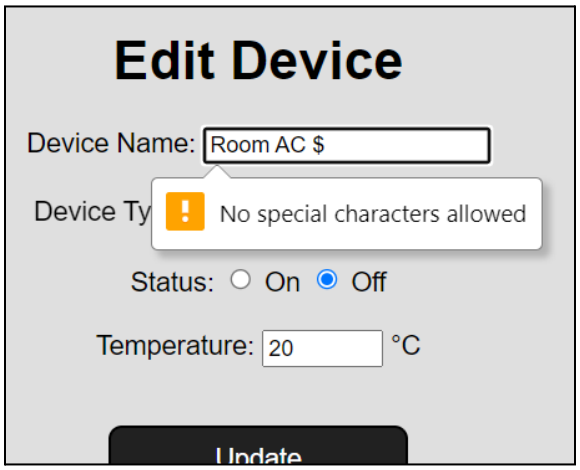
- **Data Sanitisation**

html attributes “maxlength”, “min”, and “max” and “pattern” used. No special characters allowed for text inputs to prevent sql injection.

```
<!-- Name of Device (maximum 50 characters, special characters not allowed) -->
<label> Device Name: </label>
<input type="text" id="first" name="name" value="<%= device.device_name %>" maxlength="50" required
pattern="[a-zA-Z0-9 ]{0,50}" oninvalid="this.setCustomValidity('No special characters allowed')" onir
```

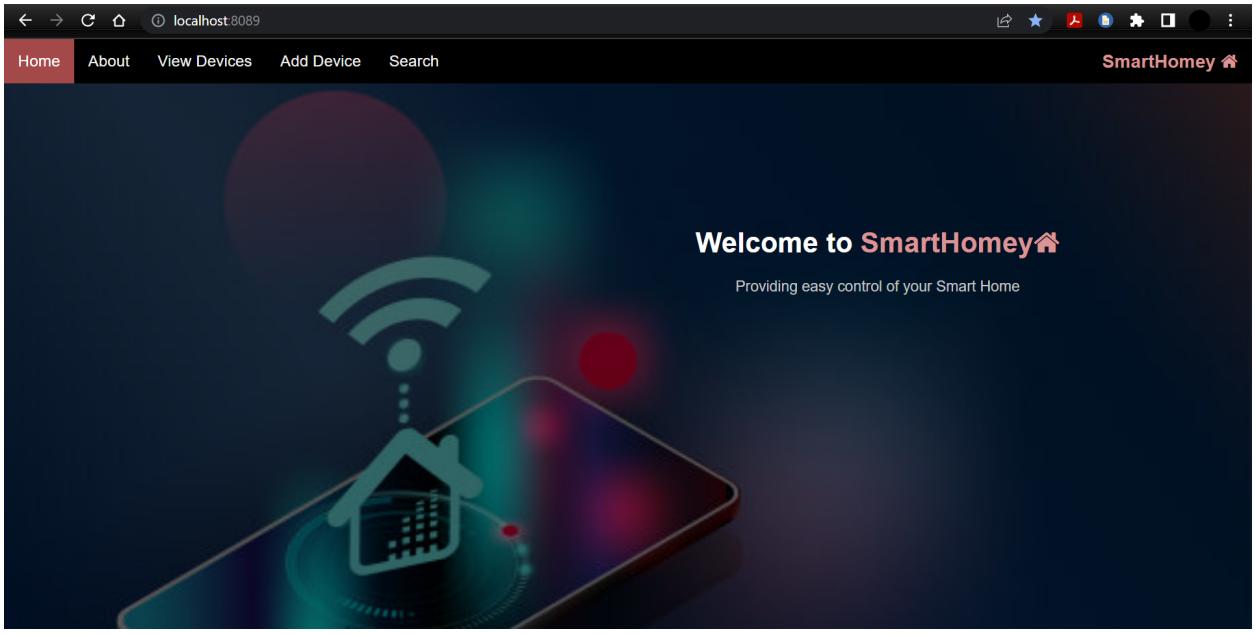
```
<!-- Temperature (maximum of 40 and minimum of -25 °C) -->
<div id="temp">
  <label> Temperature: </label>
  <input type="number" min="-40" max="40" name="temperature" step="1" /> °(
</div>

<!-- Volume (maximum of 100 and minimum of 0 dB)-->
<div id="vol">
  <label> Volume: </label>
  <input type="number" min="0" max="100" name="volume" step="1" /> dB<br />
</div>

<!-- Speed (maximum of 100 and minimum of 0 RPM)-->
<div id="spd">
  <label> Speed: </label>
  <input type="number" min="0" max="100" name="speed" step="1" /> RPM<br />
</div>
```

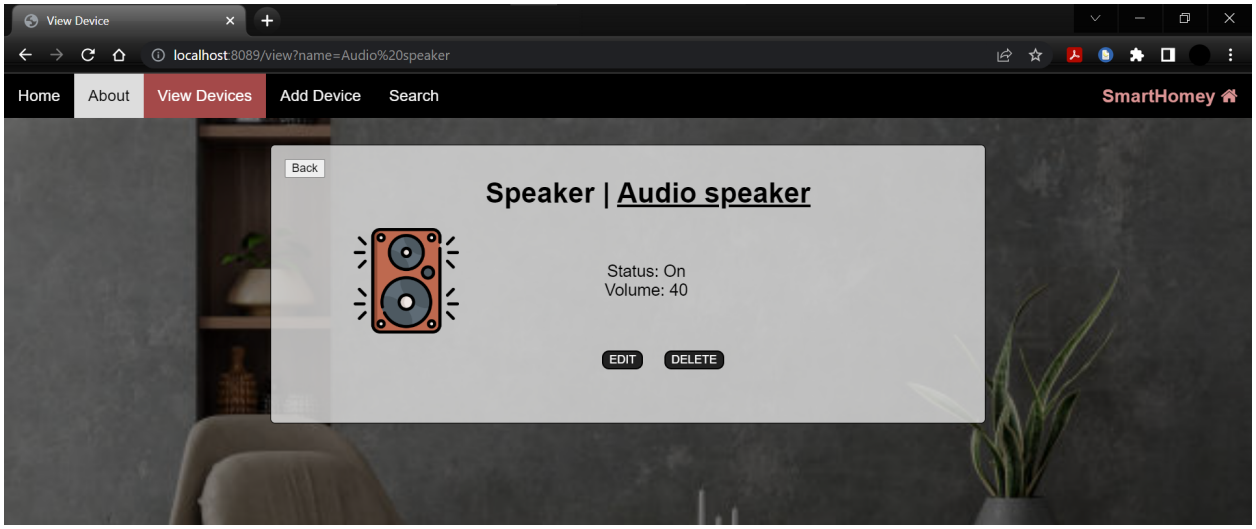




- **Front end styling and GUI**

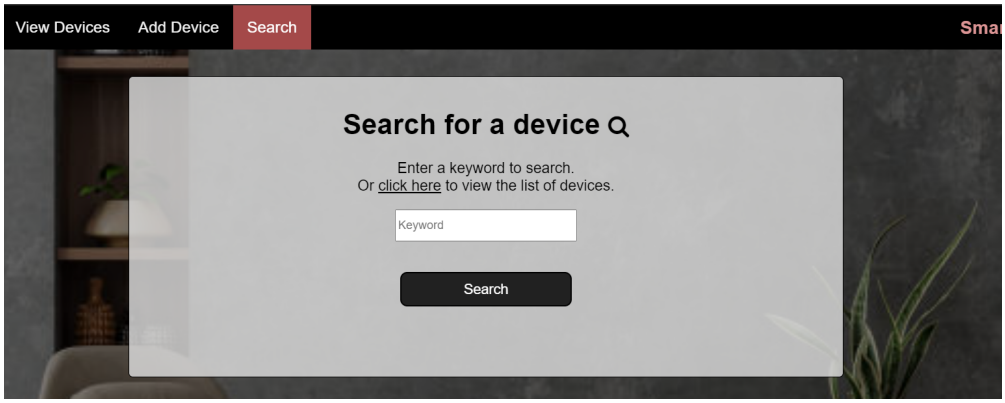
CSS used to style every page. (All css code in *topic7\public\css\style.css)*



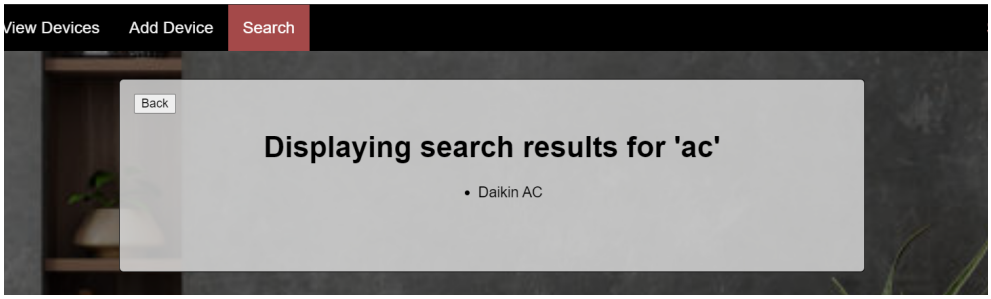


- **Search page**

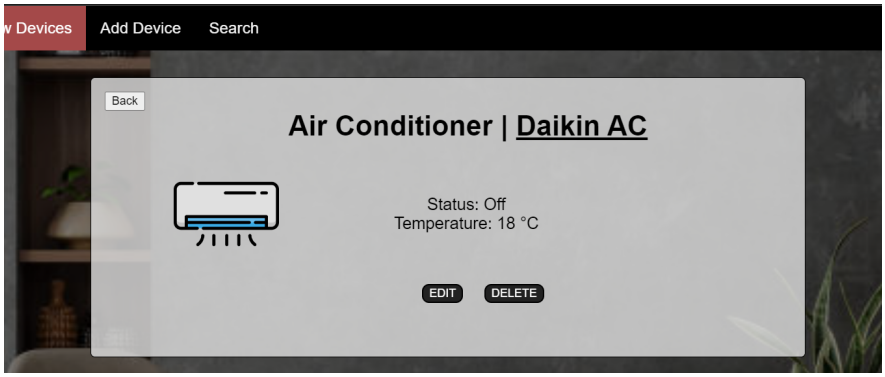
Enter keyword to search.



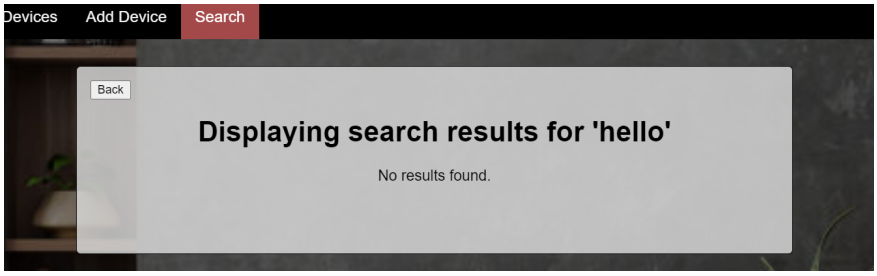
Displays results containing the keyword after pressing ‘search’.



Clicking on any of the search results will direct to the status page of that specific device.



When no results are found.

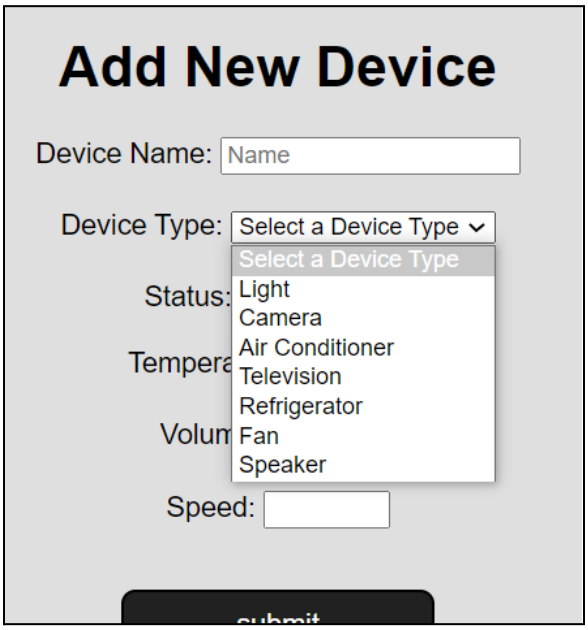


Struggles

1. The “update device” function. Whenever I update a device, a new record would be added instead of the existing record being updated. After reviewing coursera lab exercises, I realised that I rendered the ‘addDevice’ view instead of the ‘editDevice’ view.

2. Hiding fields non-applicable fields (E.g. temperature field for speaker). I overcame this by searching on stackoverflow.com until I came across a post on javascript onchange event which I was able to apply in my addDevice.html page to hide irrelevant fields depending on the “device type” dropdown field selection.

   (All fields visible if device type is not chosen)



   (Only volume field displayed if speaker is chosen)

Improvements I would make to the app

- Ability to sort added devices by category of device type, or status (on/off).
- Displaying success messages as a pop-up.
- More visual content (pictures and icons).

Database design

The database “smartHomey” consists of only 1 table named “devices”, which has 6 fields in total. The device name is the primary key. The fields “temperature”, “volume”, and “speed” are NULL as they are not applicable to certain device types. For example, devices of type “Fan” should not have a volume setting. Device name, type, and status are of type VARCHAR, while the temperature, volume, and speed are type INT.

```
create database smartHomey;

create table devices(
    device_name VARCHAR(50) NOT NULL,
    device_type VARCHAR(30) NOT NULL,
    status VARCHAR(3) NOT NULL,
    temperature INT NULL,
    volume INT NULL,
    speed INT NULL,
    PRIMARY KEY(device_name)
);
```

(sql query to create the database and table)

Field	Type	Null	Key	Default	Extra
device_name	varchar(50)	NO	PRI	NULL	
device_type	varchar(30)	NO		NULL	
status	varchar(3)	NO		NULL	
temperature	int	YES		NULL	
volume	int	YES		NULL	
speed	int	YES		NULL	

(‘devices’ table information)

device_name	device_type	status	temperature	volume	speed
Corridor Light	Light	Off	NULL	NULL	NULL
Daikin AC	Air Conditioner	Off	18	NULL	NULL
Kitchen Fridge	Refrigerator	On	-4	NULL	NULL
Living Room Fan	Fan	Off	NULL	NULL	6
Pet Camera	Camera	On	NULL	NULL	NULL
PRISM TV	Television	Off	NULL	30	NULL
Sony Speaker	Speaker	Off	NULL	56	NULL

(‘devices’ table with 7 rows of data)