Databases, Network and the Web Coursework for Midterm:

MySmartHome

Introduction

Create a new dynamic web application which allows users to monitor and control devices in their smart home. The app is just a prototype so it doesn't need interface with actual hardware, but it should provide the following functionality:

- Users can add a range of smart home devices to their dashboard. For example:
    - cooling and heating systems,
    - curtains and blinds,
    - lighting, security,
    - audio- visual systems such as TV, radio, music players,
    - kitchen appliances such as fridge-freezer, hob, oven, microwave
- Users can organise their devices by giving them custom names
- Users can monitor the status for each device (Eg. heating - room temperature, on/off, lighting on or off, curtains open/closed)
- Users can control each of these devices (Eg. turn lighting on or off, increase or decrease heating temperature etc)

The full requirements are set out in detail in the specification below.

Working Environment

For the purpose of this assignment, we have set up a working directory called "mid-term" inside the "topic7" folder of your labs. The folder contains a dummy "index.js" file which creates a web server on PORT 8089. You can run the application with the following two Terminal commands:

- cd topic7/mid-term
- node index.js

PLEASE NOTE THAT WE STRONGLY RECOMMEND YOU WORK ON YOUR LOCAL DEVELOPMENT ENVIRONMENT AS THE VIRTUAL LABS ARE NOT RELIABLE. YOU CAN FIND THE MIDTERM FOLDER INSIDE THE STARTER FILES THAT YOU PREVIOUSLY DOWNLOADED FOR THIS COURSE.

Specification

Base features:

Implement these basic features correctly to achieve a passing grade.

Home & About pages:

- Name and branding for the web application
- Provide brief information about the web application including your name as the developer.
- A menu with links to all other relevant pages in the application

Add Device page:

- Form to add a new device to the database with the following fields:
    - Name of device
    - Type of device - to be selected from a predefined list
        - Include at least 5 types
    - Other input fields such as on/off, open/close, temperature, volume, etc.
    - Submit button
- Implement client-side form validation.
    - Make sure all form data required from the user is filled and valid.
    - If required fields are empty or data is not valid, re-display the form to the user with appropriate message to fill it again.
        - For example, entering a string or a value of 400 is not valid for the temperature of a ‘heating’ device.

Add Device server-side functionality:

- Add a POST action to the form which collects the form data and passes it to the server
- Add a corresponding POST endpoint which stores the device name, type and its corresponding initial status data (eg. on/off, open/close, temperature, volume, etc) in the database.

Device Status page:

- A device selector input which allows the user to select a device from a list of created devices
- A status viewer which, for the selected device, will display the status data found in the database (eg. on/off, temperature, etc).

Device Status page server-side functionality:

- A GET action which collects the form data and passes it to the server
- A GET endpoint which uses the device name to find and retrieve the relevant device information from the database and serve it to the client

Control device page:

- A device selector input which allows the user to select a device from a list of created devices

- A form with inputs to control the chosen device's settings (eg. on/off, temperature, etc).
- A submit button to change the settings

Control device server-side functionality:
- A POST action which collects the form data and passes it to the server
- A POST endpoint which uses the device name to update the device with the new settings

Delete device page
- A device selector input which allows the user to select a device from a list of created devices
- A delete button
- Extension: Improve the design by asking for the user's confirmation with a confirm modal before the delete operation.

Delete device - server-side functionality
- A POST action which collects the form data and passes it to the server
- A POST endpoint which uses the device name to delete the device and related data from the database

Navigation:
- The site should be fully navigable.
- The home page should have a menu leading to all other pages
- Each other page should display at the very least a home button

Extensions:

Optionally implement some or all of these features to achieve a higher grade.
DON'T FORGET TO WRITE THESE UP IN YOUR COMMENTARY

Success feedback:
- Each POST action should result in a message to the client showing that the action was successful.
- Depending on your preference this might be a simple text message or a redirect to a different page.

Hot reloading:
- Go one step further and implement hot reloading to show changes as and when the user makes them. For example,
    - Reload the Control Device page once a device has been selected populating the input fields with values from the current state of the chosen device
    - Reload the Delete Device page once a device has been deleted to show that it is no longer in the list of devices.

Disable non-applicable fields:
- Not all fields are relevant to each device type.

    - the open/close field is not related to a heating device but applies to a blind or a curtain, or a door.
    - Volume does not apply to the heating system, but the temperature does.
- Automatically assign null values, (eg. -1, null, “or not applicable”) as the initial value for fields that do not apply to the given deviceType.
- Disable or hide input fields with null values from users.
    - For example, if a user adds a new device of type ‘heating’ then only on/off and temperature should be displayed.

Data Sanitisation:

- Prevent malicious attacks on your server by sanitising the user data in each of your POST endpoints
- Make sure that the device name conforms to a schema (eg. minimum length, avoid special characters)
- Check the other data matches the expected types, ranges and values
- Throw an error if the message fails to meet requirements and make sure that this is returned to the client without crashing the server

Front end styling and GUI:

- Use CSS styling to create a modern professional look and feel to your application
- You might do this using utility frameworks such as Bootstrap or Tailwind, or write the CSS from scratch
- You might go further and implement graphical controls for some of your device inputs. (eg. a turnable knob for a temperature control)

Code Style Requirements:

- Code is organised into JavaScript (.js) files and template files (.html or .ejs or .pug). JavaScript files contain web server code (index.js) and middleware (main.js in routes folder). Template files (.html and .ejs and .pug) are stored in views folder.
- Each route in main.js has comments describing purpose, inputs, and outputs
- Code is laid out clearly with consistent indenting
- Each database interaction has comments describing the purpose, inputs, and outputs
- Functions and variables have meaningful names, with a consistent naming style

Submission

- A video of your web application showing the main functionalities. This should not be longer than 2.5 minutes. We recommend that you capture the video in mp4 format using software such as OBS.
    - This is how we grade the functionality that you have implemented against the rubric so be sure to show all of the functionality within the time limit.
    - We will not watch the video beyond the time limit.

- The source code of your web application in zip format
    - We will not run this code but will look at it to assess the quality of your implementation

- Commentary in PDF format
    - Approx 300 words on your of implementation
        - Highlight achievements.
        - Describe what extensions you implemented and how
        - Describe any aspects that you struggled with and how you eventually overcame them
        - Describe what improvements you would make to the app
        - Use screenshots of your code and the interface to illustrate your points.
    - Documentation of your database design
        - Use the mysql CLI or app to screenshot each of your database tables populated with several rows of data
        - Below each screenshot write a brief description of what the table is for and the rationale for its structure.

Rubric

Home & About pages:
0 - no attempt / not visible in the demo video
1 - partial implementation
2 - complete implementation

Add Device:
0 - no attempt / not visible in the demo video
1 - limited functionality (eg. client side form is implemented but doesn't have functionality)
2 - incomplete functionality (eg. form validation is not demonstrated, device names are not implemented)
3 - all aspects are implemented (functionality fully demonstrated)

Device Status page:
0 - no attempt / not visible in the demo video
1 - partial implementation (eg. client side form is implemented but doesn't have functionality)
2 - complete implementation (functionality fully demonstrated)

Control device page:
0 - no attempt / not visible in the demo video
1 - partial implementation (eg. client side form is implemented but doesn't have functionality)
2 - complete implementation (functionality fully demonstrated)

Delete device page:
0 - no attempt / not visible in the demo video
1 - partial implementation (eg. client side form is implemented but doesn't have functionality)

2 - complete implementation (functionality fully demonstrated)

Navigation:
0 - there is no apparent menu or navigation
1 - partial implementation (menu is visible but not on every page, screen refreshes and back button presses are required)
2 - complete implementation

Extensions:
0 - no extensions attempted
1 - extensions attempted but incomplete
2 - one extension completed
3 - two extensions completed
4 - at least three extensions completed
5 - at least three extensions completed with significant expansion beyond the requirements in some places OR all extensions completed

Code quality:
Mark out of 6
Penalise 1 point for minor faults in a category, 2 points for major faults in a category
NB. marks can't be negative !
Feedback in comments

Database design:
0 - not demonstrated / implemented
1 - disorganised/irrational design showing conceptual misunderstanding
2 - good design implementation with some minor errors or inconsistencies
3 - perfect design

NB. Marks can be deducted if the database design is not fully documented

Presentation/deliverables:
0 - non submission
1 - poor organisation and presentation, multiple items missing
2 - organisation and presentation could be improved, or one item missing
3 - perfect submission