CM2040 MIDTERM REPORT

D3) List of requirements:

R1: Home page:	
R1A: Display the name of the web application.	When the browser makes a request to the localhost:8089 (initial load), routing script renders the home page – index.html (main.js line 14); The home page template has a header that includes application name (index.html line 13).
R1B: Display links to other pages or a navigation bar that contains links to other pages.	Same page template includes navigation menu (index.html lines 15-21) that contains links to other pages like “About”, “List” or “Add device”.
R2: About page:	
R2A: Display information about the web application including your name as the developer. Display a link to the home page or a navigation bar that contains links to other pages.	Routing script handles requests to the about page on the line 20, and responds with render of the about.html page. As every other page it contains header, navigation menu, but also contains additional information about the web application, like purpose, information about tools used and author name.
R3: Add device page:	
R3A: Display a form to users to add a new device to the database. The form should consist of the following items: name of the device to be selected from a pre-defined set of devices and other input fields such as on/off, open/close, temperature, volume, etc. Depending on the chosen device other input fields might be applicable or not applicable. The minimum number of devices in the	When the user make an initial request to the “Add device page” (no params) the routing script makes a query to the database and renders “deviceAdd.html” with information about all available device types and information about properties of the initial type (main.js line 96). Page template (deviceAdd.html) holds input field for the device name (line 26), and uses template engine to parse information about device type into the table of radio inputs (lines 38-52). There are 26 pre-built device type in this application. The rest of the form is rendered by the template engine on the line 56 → the data itself is the HTML string, that is rendered on the fly in the asynchronous mode depending

pre-defined list is 20. Display a link to the home page or a navigation bar that contains links to other pages.	on the user input. This approach allows user smooth experience, because there is no need to reload the page every time he want to choose another device type. This is dynamic page, because we need to show user different information based on the input. But dynamic approach requires logic and the we handle it on the back-end, and return user only information he require (i.e. only applicable input fields). Function “dataToForm” (function.js line 35) creates wrapper, iterates over all properties of the selected device type, and add required input. Function “createInput” (function.js line 167) holds logic, decides what input should be prepared and handles additional options, like input ranges, max and min values, etc. Navigation menu is pre-built in the page template. (deviceAdd.html line 15).
R3B: Collect form data to be passed to the back-end (database) and store the device name and its corresponding initial status in the database. Each device item consists of input fields such as name, on/off, open/close, temperature, volume, etc. Some fields might not be applicable for some devices. As an example, the open/close field is not related to a heating device but applies to a blind or a curtain, or a door. As another example volume does not apply to the heating system, but the temperature does. Display a message indicating that add device operation has been done successfully.	When the user fills the form and clicks the submit button application makes a POST request to the server with data, attached to the requests body. Routing script handles post request, sanitize user input and makes a query to add a new device to the database (main.js line 141 – for request, line 153 for the query). To prevent SQL injections, the back-end uses prepared MySQL statement with sanitized data, while form validation prevent the user from entering wrong data on the front-end. To handle non applicable fields “devices” table has default values for almost all settings, so if we input required fields, the rest will be automatically filled with NULL values. Depending on the status of the request, user will be redirected with the service message. We cant send data while redirecting, but, to work around this limitation and also to prevent user from querying directly, we use cookies to send some data to the user’s browser, to check that message request is valid, before executing it. If everything is OK, we redirect user to the list page to see the new device with success message (main.js line 164). On the other hand, if there is a problem with the Insert operation, we return him to the main page with the error message (main.js line 164). To handle redirects we use custom function, that prepares message, writes cookies to the user’s browser and redirect him with the system message parameter (functions.js line 403).
R3C: Improve R3B by automatically assigning ‘not applicable’ initial value to the fields	The input form for the add Device page will be based on the settings associated with the required device. We use table “properties” to store information about each device

that do not apply to the chosen device. It is even better to disable or hide those input fields from users when adding a new device. As an example, if a user adds a new device called 'heating' then the minimum required input fields to be displayed and initialised by the user are: on/off and temperature and all other non-related input fields must be initialised to NA (not applicable) status.	type and applicable settings. Also, I cheated a bit, and used column names to store information about which input type to show user. The data inside the "properties" table sets different options for this settings. For example: different devices can have different number of pre-built programs, or temperature ranges. Using this two tables, we can create specialized forms that include only applicable settings and limit input values: so the user doesn't have to deal with unnecessary settings or input fields.
R3D: Form validation, make sure all required form data is filled and also valid data is entered by user. If required fields are empty or data is not valid, re-display the form to the user with appropriate message to fill it again. As an example, entering a string or a value of 400 is not valid for the temperature of a 'heating' device.	As I stated before, all validation is handled on the front-end, using HTML built in tools, like different input types and prefixes (for example, deviceAdd.html line 26 show the use of prefixes 'required', 'pattern', 'maxlength', line 45 uses input type radio with 'checked' prefix for the first type, etc.). Using this tools we can guarantee that all required fields are filled before submission, and that all data are indeed in valid form.

R4: Show device status page

R4A: Display a list of already added devices and then let the user choose a device from the list. Display a link to the home page or a navigation bar that contains links to other pages.	If user queries device status, device update or device delete page without parameter that specifies the id of the device, we redirect him to the list page (main.js lines 179 – device status → list, lines 274 and 376 – device update → list and line 456 device delete → list). Device status and device update have their own page templates which include navigation menu (deviceStatus.html and deviceUpdate.html line 16)
R4B: Display data related to the chosen device found in the database to users including name, and related settings. As an example, if TV is selected related settings would be on/off setting, volume, and channel. If the heating system is chosen minimum settings, are on/off and temperature. Display a message to the user if not found.	When user queries this page with ID parameter, routing scripts checks database to confirm that such device exists (main.js line 188). If ID is valid, then the server get information about device settings from the database, prepare HTML form and render it on the template (line 243). If device ID is not valid → user is redirected to the main page with error message (line 215).
R4C: Improve R4B by automatically hiding input fields that do not apply to the chosen device.	Function cleanQuery (functions.js line 8) skips NULL settings, and later we create user form ONLY FOR applicable settings. As for user friendly visuals → input form types provide more intuitive interface with buttons,

R4D: Going beyond by improving R4B and R4C, by a graphical display of the device settings like a dashboard.	range sliders, color pickers and time schedules (functions.js lines 179 - 277).

R5: Update device status page

R5A: Display a list of already added devices and then let the user choose a device from the list. Display a link to the home page or a navigation bar that contains links to other pages.	Same as R4A.
R5B: Display data related to the chosen device found in the database to users including name, and related settings. As an example, if TV is selected related settings would be on/off setting, volume, and channel. If the heating system is chosen minimum settings, are on/off and temperature, so users can update each field. Collect form data to be passed to the back-end (database) and store updated device status in the database. Display a message indicating the update operation has been done.	Same as R4B. The only difference is that form created for this operation allow data modification and re-submission (achieved through optional flag for the createInput function (main.js line 231 – locked form, line 330 – unlocked one). In the end, again, redirect with service message, depending on the result.
R5C: Improve R5B by automatically hiding input fields that do not apply to the chosen device.	Same as R4C and R4D
R5D: Going beyond by improving R5B and R5C. Improvement could be achieved by letting the user update device status through a graphical display of the device settings like a dashboard.	

R6: Delete device page

R6A: Display a list of already added devices and then let the user choose a	Same as R4A. I made delete page, but later decided not to use it, because

device from the list to be deleted. Display a link to the home page or a navigation bar that contains links to other pages.	we don’t need to show anything unique (list page already exists) worth full page. Delete is mostly operation, and we can use another pages to show status message. Rather than delete page, I created List page, that handles different actions and also has navigation menu there (list.html line 15).
R6B: It is always a good practice to ask for the user’s confirmation before the delete operation. Ask the user 'Are you sure?' then if confirmed, delete the chosen device and related data from the database by collecting form data related to the device name to be passed to the back-end (database) and performing delete operation on the database. Display a message indicating the delete operation has been done successfully.	To prevent accidents, I used JS confirm function, to prompt the user (list.html line 52). If user confirms, the link makes request “deviceDelete” request with appropriate parameter (line 51). This operation ends with redirect and service message – depending on the operation status (success or error).

D4) Database structure:

Table 1 – Properties

Field name	Data type	Purpose
type	varchar(50), PRIMARY KEY	Device type name to identify available list of settings and value ranges
status[radio]	set('on','off'), NOT NULL	Device status: On or Off, all devices have this setting
schedule_start[radio]	set('on','off'), DEFAULT NULL	Schedule activation mode
start_time[datetime]	datetime, DEFAULT NULL	Schedule activation date+time
program[range]	varchar(16), DEFAULT NULL	Pre-built program / work mode
silent_mode[radio]	set('on','off'), DEFAULT NULL	Low noise work mode
ionization_mode[radio]	set('on','off'), DEFAULT NULL	Active ionization work mode
air_conditioning[radio]	set('on','off'), DEFAULT NULL	Air conditioning work mode
flow_temperature,_°C[range]	varchar(16), DEFAULT NULL	Temperature: water, hot air, etc
brightness,_%[range]	varchar(16), DEFAULT NULL	Illumination / screen brightness
lights_color[color]	varchar(7), DEFAULT NULL	Illumination color
speed,_RPM[range]	varchar(16), DEFAULT NULL	Device work speed
resolution[radio]	set('320p','480p','720p','1080p'),	Graphical resolution

	DEFAULT NULL	
night_mode[radio]	set('on','off'), DEFAULT NULL	Low brightness work mode
password[text]	varchar(16), DEFAULT NULL	Device password
force_lock[radio]	set('on','off'), DEFAULT NULL	Force lock device
blinds[radio]	set('open','close'), DEFAULT NULL	Blinds state
flow_rate,_%[range]	varchar(16), DEFAULT NULL	Rate of the work flow: water, air
section_1_temperature,_°C[range]	varchar(16), DEFAULT NULL	Temperature of the specific section
section_2_temperature,_°C[range]	varchar(16), DEFAULT NULL	
section_3_temperature,_°C[range]	varchar(16), DEFAULT NULL	
freezer_temperature,_°C[range]	varchar(16), DEFAULT NULL	
ice_maker[radio]	set('on','off'), DEFAULT NULL	Activate ice maker mode
notes[text]	varchar(255), DEFAULT NULL	Just a simple information note
delicate_mode[radio]	set('on','off'), DEFAULT NULL	Activate delicate work mode
alarm_clock[radio]	set('on','off'), DEFAULT NULL	Activate timed alarm
alarm_time[time]	time(6), DEFAULT NULL	Timed alarm timer
wattage,_W[range]	varchar(16), DEFAULT NULL	Adjust power mode
steam_preheat[radio]	set('on','off'), DEFAULT NULL	Preheat stem in coffee machine
processor_setting[radio]	set('low','high','pulse'), DEFAULT NULL	Food processor work settings
channel[range]	varchar(16), DEFAULT NULL	Radio or TV channel
volume,_%[range]	varchar(16), DEFAULT NULL	Sound volume
schedule_stop[radio]	set('on','off'), DEFAULT NULL	Schedule deactivation mode
stop_time[datetime]	datetime(6), DEFAULT NULL	Schedule deactivation date+time
find/play_track[text]	varchar(64), DEFAULT NULL	Track name and author for autoplay mode
auto_play[radio]	set('on','off'), DEFAULT NULL	Autoplay mode, to find and play tracks automatically
bass_boost[radio]	set('on','off'), DEFAULT NULL	Activate bass boost
activate_camera[radio]	set('on','off'), DEFAULT NULL	Activate device’s video channel
activate_voice_channel[radio]	set('on','off'), DEFAULT NULL	Activate device’s audio channel
portion_size,_g[range]	varchar(16), DEFAULT NULL	Feeder’s portion size
energy_save_mode[radio]	set('on','off'), DEFAULT NULL	Energy save work mode

section_1_firmness[radio]	set('soft','medium','firm'), DEFAULT NULL	Firmness of the specific section
section_2_firmness[radio]	set('soft','medium','firm'), DEFAULT NULL	
track_heartbeat[radio]	set('on','off'), DEFAULT NULL	Activate heartbeat sensor
activate_security_system[radio]	set('on','off'), DEFAULT NULL	Activate security system

It should be noted, that all field names (expect the type) have 2 parts: one – device property name, and other – device property HTML input type. All radio inputs have SET datatype to present several predefined options, specific for each device type. Date, datetime, text and color inputs are also self explanatory. Range inputs are tricky ones: we set their values as a set of 3 coma separated values, to define min, max and step values for this device input.

Table 2 – Devices

Field name	Data type	Purpose
id	Int(4) PRIMARY KEY	Device identity number
name	varchar(50), NOT NULL	Device name
type	varchar(50), NOT NULL	Device type name to identify available list of settings and value ranges
status[radio]	set('on','off'), NOT NULL	Device status: On or Off, all devices have this setting
schedule_start, and other …	SAME AS TABLE 1	SAME AS TABLE 1, the only difference is that field names in this table doesn't have input type part in them

Both tables are connected through the `TYPE` field name in both tables.