Web and Mobile Application development

Submitted by

Dharmalingam Dhayananth

Mines Saint Etienne

Université Jean Monnet

Introduction

This report provides information about developed mobile and web server application in order to manage heater and window IoT devices remotely. Different web technologies and mobile application development technologies were used in order to implement this project.

This report feature explanation of what has been done in the project and how far the features are functional.

Web Application

Web application has been developed using Java Spring boot framework. This application is developed by following REST architecture.

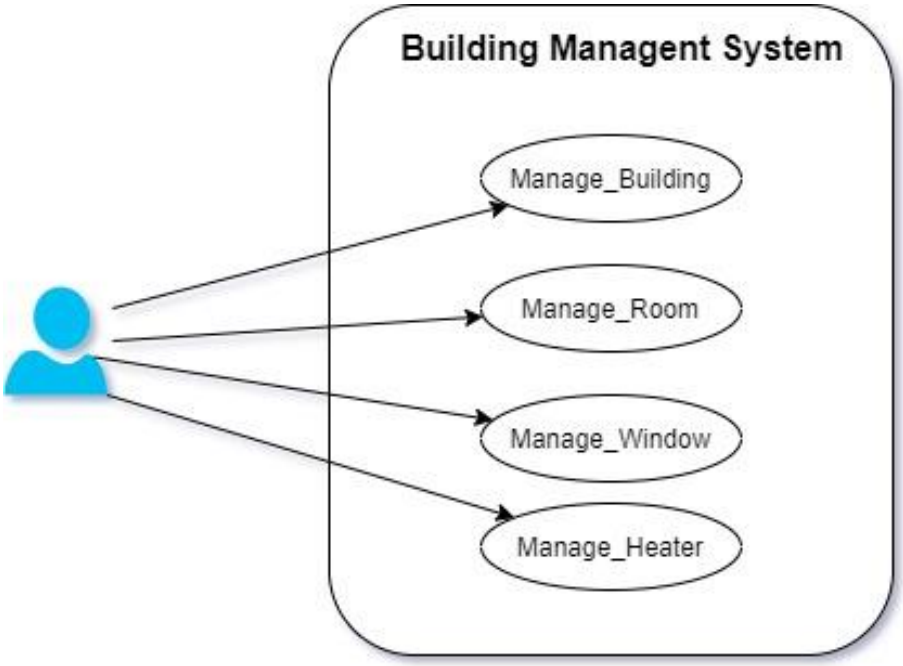

Figure 1: Use case diagram

Above diagram shows Use case diagram of developed web application. The application has different entity-based functionalities that allows to perform CRUD operations on selected entity. The functionalities are as follow:

Manage_Building:

- Create Building

  User can able to create new building, that data will be stored to the database.

- Update Building

  User can able to update an existing building (Ex: name of the building, outdoor temperature)

- Delete Building

User can able to delete a building record.

- Switch all **Heater** status of all Room in a building {ON | OFF}

User can able to **turn off** or **turn on** all the heaters of a specific room. That will change status of heaters within all the rooms in a building.

- Switch all **Window** status of all Room a building {CLOSE |OPEN}

User can able to **open** or **close** all the windows of a specific building. That will change status of windows within all the rooms in a building.

Note: Deleting a building will delete all corresponding **Rooms** of that building and relating **Heaters** and **Windows**.

- Get all building

User can able to get all existing Building data from the system

- Get one specific building

User also able to get one specific building by passing specific building-id to the system.

Manage_Room:

- Create Room

User can able to create new room that belongs to a specific building. The data will be stored to the database. It is mandatory to pass corresponding building-id to associate new room data to its belonging building.

- Update Room

User can able to update an existing room (Ex: name of the room, floor, target temperature etc)

- Delete Room

User can able to delete an existing Room.

Note: Deleting a Room will delete all corresponding **Heaters** and **Windows**.

- Get all Rooms

User can able to get all existing Room data from the system

- Get one specific Room

User also able to get one specific Room by passing specific room-id to the system.

- Switch all **Heaters** status of a Room {ON | OFF}

User can able to **turn off** or **turn on** all the heaters of a specific room. That will change status of heaters within all the given room.

- Switch all **Windows** status of a Room {CLOSE |OPEN}

User can able to **open** or **close** all the windows of a specific room. That will change status of windows within the given room.

Manage_Heater:

- Create Heater

User can able to create new heater that belongs to a specific room. The data will be stored to the database. It is mandatory to pass corresponding room-id to associate new heater data to its belonging room.

- Delete Heater

User can able to delete an existing heater.

- Get all Heater

User can able to get all existing heater data from the system

- Get one specific Heater

User also able to get one specific heater by passing specific heater-id to the system.

- Switch **Heater** status {ON | OFF}

User can able to **turn off** or **turn on** the heaters of a specific room.

Manage_Windows:

- Create Window

  User can able to create new window that belongs to a specific room. The data will be stored to the database. It is mandatory to pass corresponding window -id to associate new window data to its belonging room.

- Delete Window

  User can able to delete an existing window data.

- Get all Window

  User can able to get all existing window data from the system

- Get one specific Window

  User also able to get one specific window by passing specific window-id to the system.

- Switch **Window** status {CLOSE |OPEN}

  User can able to **open** or **close** all the window of a specific room.

End-point and Response

Above mentioned functions associated to single end-point in REST application. The API-Swagger document gives detailed view of each end points and expecting input args and response sample. (can be accessed at : https://rest-api.cleverapps.io/swagger-ui/index.html)

Error response

The application designed to handle all possible errors and provide proper response based on the system error. For example, the system will return “404 building not found” when trying to edit a building with wrong building id. In this case, the given building id was not associated with any database record.

Testing

The application was tested properly using Junit testing framework in order to assure the reliability of functionalities.

Manage_Heater:

- Create Heater

User can able to create new room that belongs to a specific building. The data will be stored to the database. It is mandatory to pass corresponding building-id to associate new room data to its belonging building.

- Update Heater

User can able to update an existing room (Ex: name of the room, floor, target temperature etc)

- Delete Heater

User can able to delete a existing Room.

Note: Deleting a Room will delete all corresponding **Heaters** and **Windows**.

- Get all Heater

User can able to get all existing Room data from the system

- Get one specific Heater

User also able to get one specific Room by passing specific room-id to the system.

Mobile Application

Mobile application has been developed in order to provide user friendly interaction to the system. This application has been developed using Kotlin. Following screenshot if the pages gives better explanation about functionalities.

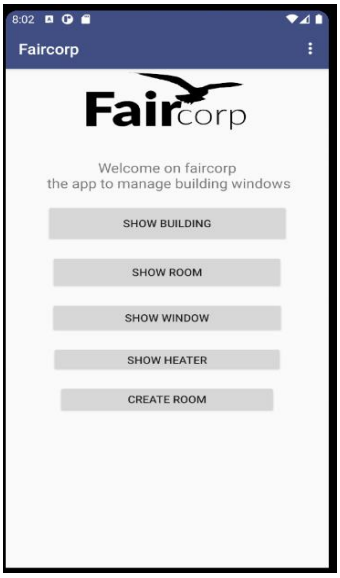

Figure 2: Home Screen

Above figure shows the home screen of the mobile application. User can able to click any of the button to navigate to specific screen. Show building button will bring all the buildings thing the system. Then user can able to select a specific building to get its rooms data. This screen is shown below.

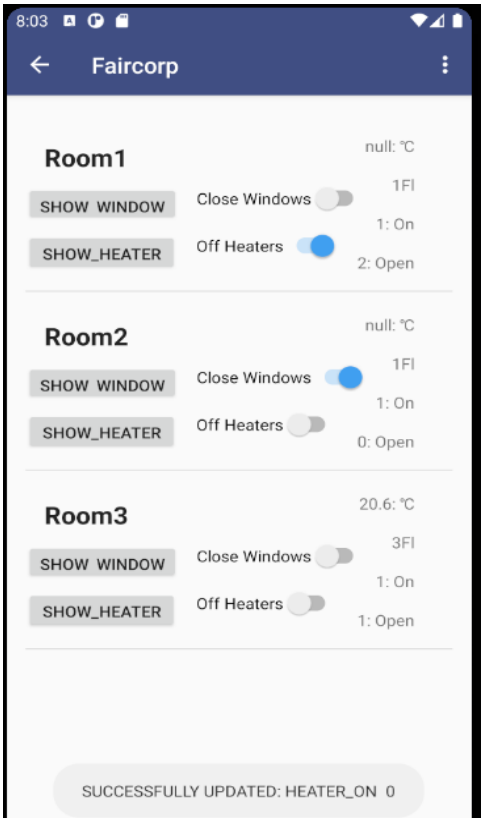

Figure 3: Manage Room screen

Above image shows the room management screen. This screen will get all the rooms of a selected building. User can able to close all the windows with the small “switch-control”. User can either CLOSE or OPEN all the windows of the specific room by changing this “switch-control” value.

In the same manner, user can able to switch heater status by changing the “switch-control” of heaters. User can either turn OFF or ON all the heaters of a specific room. Each room has this both switch controls.

Additionally, user can able to view all the heaters status by clicking “SHOW_HEATER” button. That will open up another page where user can view and update each heater individually. Similarly, “SHOW_WINDOW” button will bring up window management page, where user can view the status of the window and manage it individually. These two screens are discussed below.

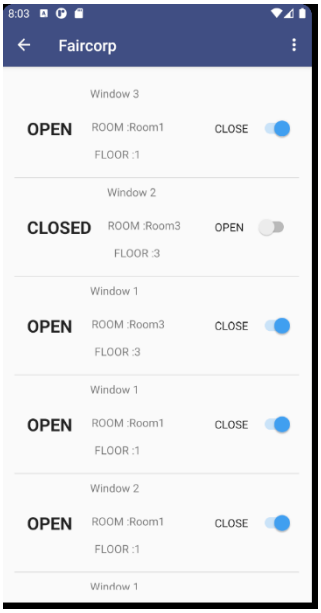

Figure 5: Window management screen

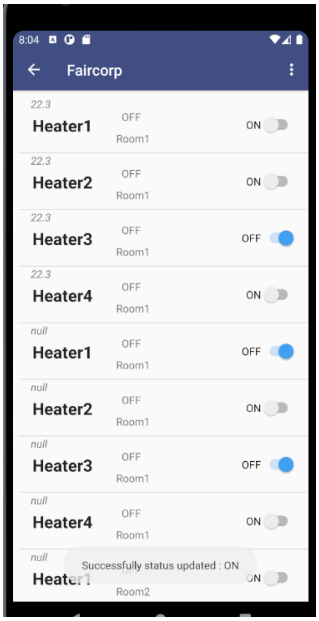

Figure 4: Heater management screen

Above figure shows the screen shot of heater and window management screens. In this screen user can able to see all window status and heater status individually and update the status of it by changing the switch control. User can either CLOSE or OPEN a window by changing that corresponding switch control and ON or OFF the heater with associated switch control in heater management screen.

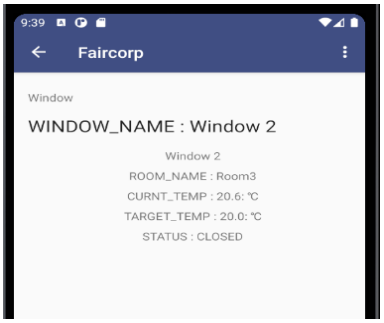

Figure 6: Detailed window info screen

User can able to view detailed view of window by clicking window list element in the shown window management screen. This will bring up information screen as shown in above figure.

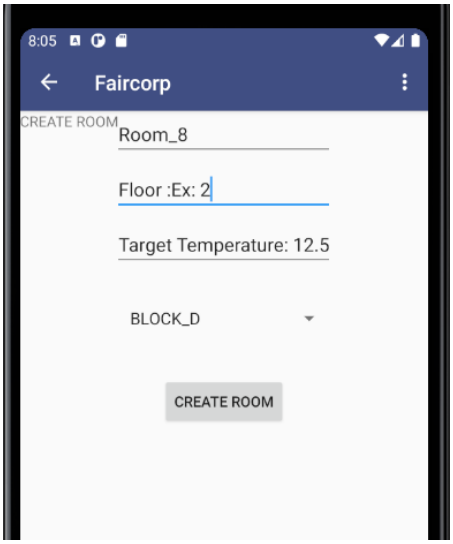

Figure 7: Create room screen

Additionally, user can able to create a room with the create room screen that is shown in above figure. User must enter room name (text), floor (number) and target temperature (decimal number). Also, the user should select one of the building in the given “select box”. This select box will be loaded with all existing building from the system. At last, user should click create room button in order to create the room.

All these addressed functionalities and features are tested and working properly.

Testing configuration

The API application can be tested using INSOMNIA tool, that helps to test API Application. The configuration file will be given along with the project files, which you can able to simply import to your INSOMNIA tool, and instantly start testing the end points with just clicking the buttons. Because each end points are pre-configured with required input-args.

Step 1: import the '*Insomnia_2021-01-13.json'* file to your insomnia workspace (right top corner)

Step 2: select 'remote' in the environment tab.

You can download the tool here : https://insomnia.rest/

You can download the workspace configuration here : https://github.com/ujm-projects/spring-rest/blob/master/Insomnia_2021-01-13.json

Atonally, you can test the application using SWAGGER API document in the following link.

https://rest-api.cleverapps.io/swagger-ui/index.html

Links to the repo

GitHub organization : https://github.com/ujm-projects

Repository-Spring rest application : https://github.com/ujm-projects/spring-rest

Repository-Android application : https://github.com/ujm-projects/android-app