FIRE SENSOR MONITORING APPLICATION

1. IT18140194@my.sliit.lk - M.R.M. Tharik
2. IT18149036@my.sliit.lk - V.M. Siriwardena
3. IT18153132@my.sliit.lk - L.M.D. Perera
4. IT18109122@my.sliit.lk - S. Sharanjan

Date 4/5/2020

Group details

Distributed system

ASSIGNMENT 2

**Introduction**

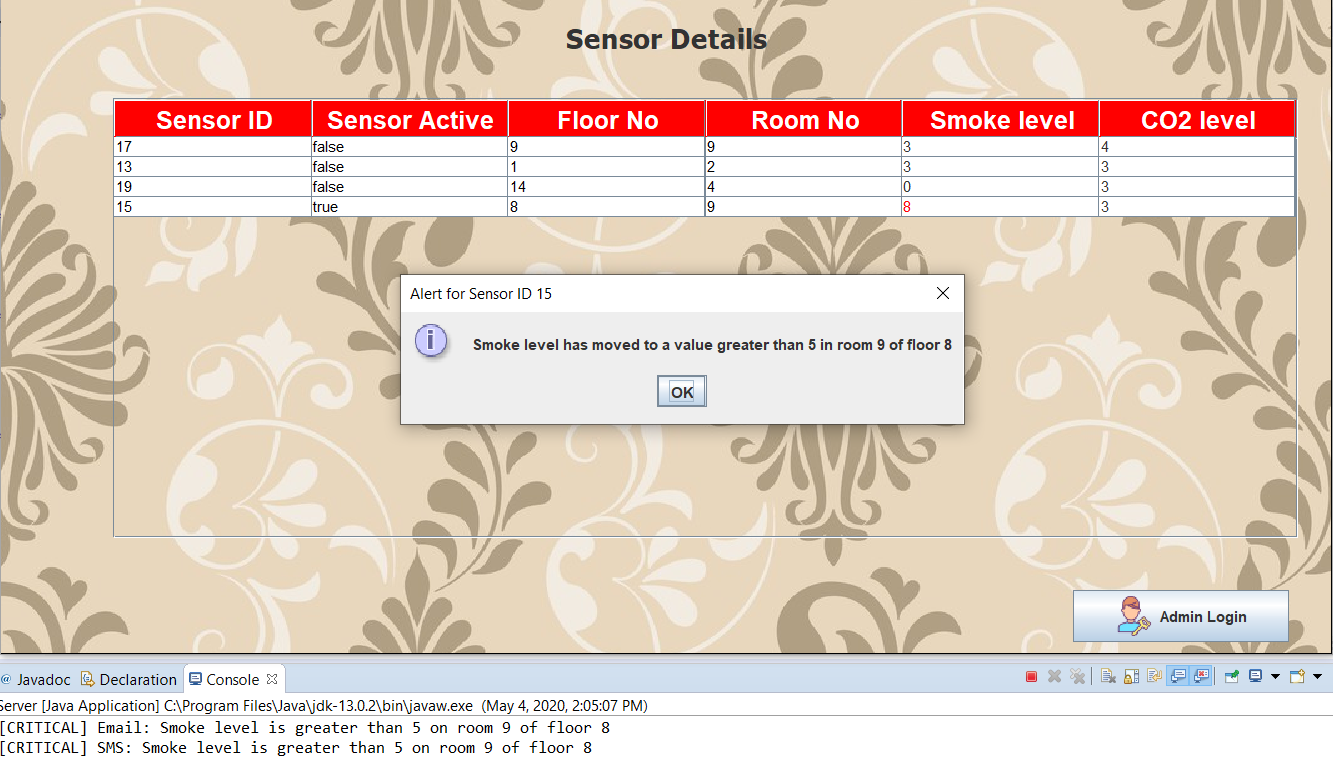
This project is based on a fire alarm monitoring system. The fire alarm monitoring system developed by our team consists of four main components namely desktop application, web client, fire sensor application and a Rest API.

When talking about the technologies used to develop these components, the desktop application was built using JAVA swing whereas the web application was built using React JS. Laravel PHP framework was used to build the REST API with the help of the passport plugin. The RMI server which stands in between the desktop application and the REST API was built using the JAVA programming language. GSON was used to convert the JSON data coming from the REST API to an object. The sensor application which is used to mimic the functionality of an actual sensor was built using electron.

This system can be used by two types of users namely a normal user and an admin. Both the admin and the normal user are able to observe the alerts and receive emails and SMS notifications when the CO2 level or smoke level goes up. Several security mechanisms have been implemented to ensure that only an admin is able to add, edit and delete sensors. These security mechanisms and the other validations implemented will be broadly explained in the subsequent chapters.

**REPORT**

In the desktop client, a table consisting of all sensor details is displayed to the user. The data in the table is refreshed every 15 seconds and an alert will pop up when the CO2 level or Smoke level moves to a value greater than 5. If the CO2 level or Smoke level is greater than 5, they are displayed in red and an email and sms notification is sent in such an occasion as shown below in the diagram.

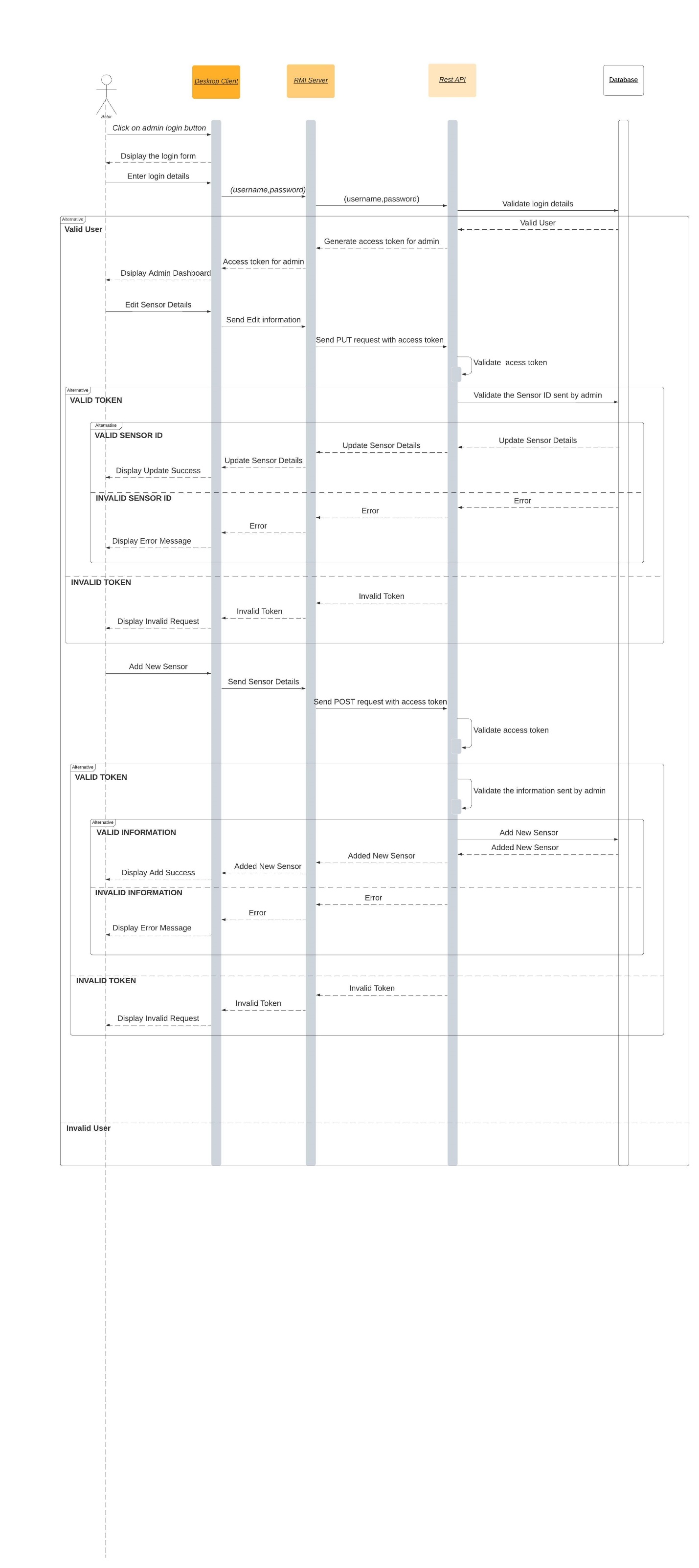


By clicking on the “Admin Login” button, the user is provided with a login form where the user has to enter the username and password in order to login as an administrator. After validating the user, the REST API generates an access token for admin. Afterwards, the user is provided with an admin dashboard where the user is given the administrative privileges to add, edit and delete the sensor details. Admin has the authority to edit the floor number and room number of the sensors. The admin dashboard also consists of a table consisting of all the sensor details which is refreshed every 15 seconds.

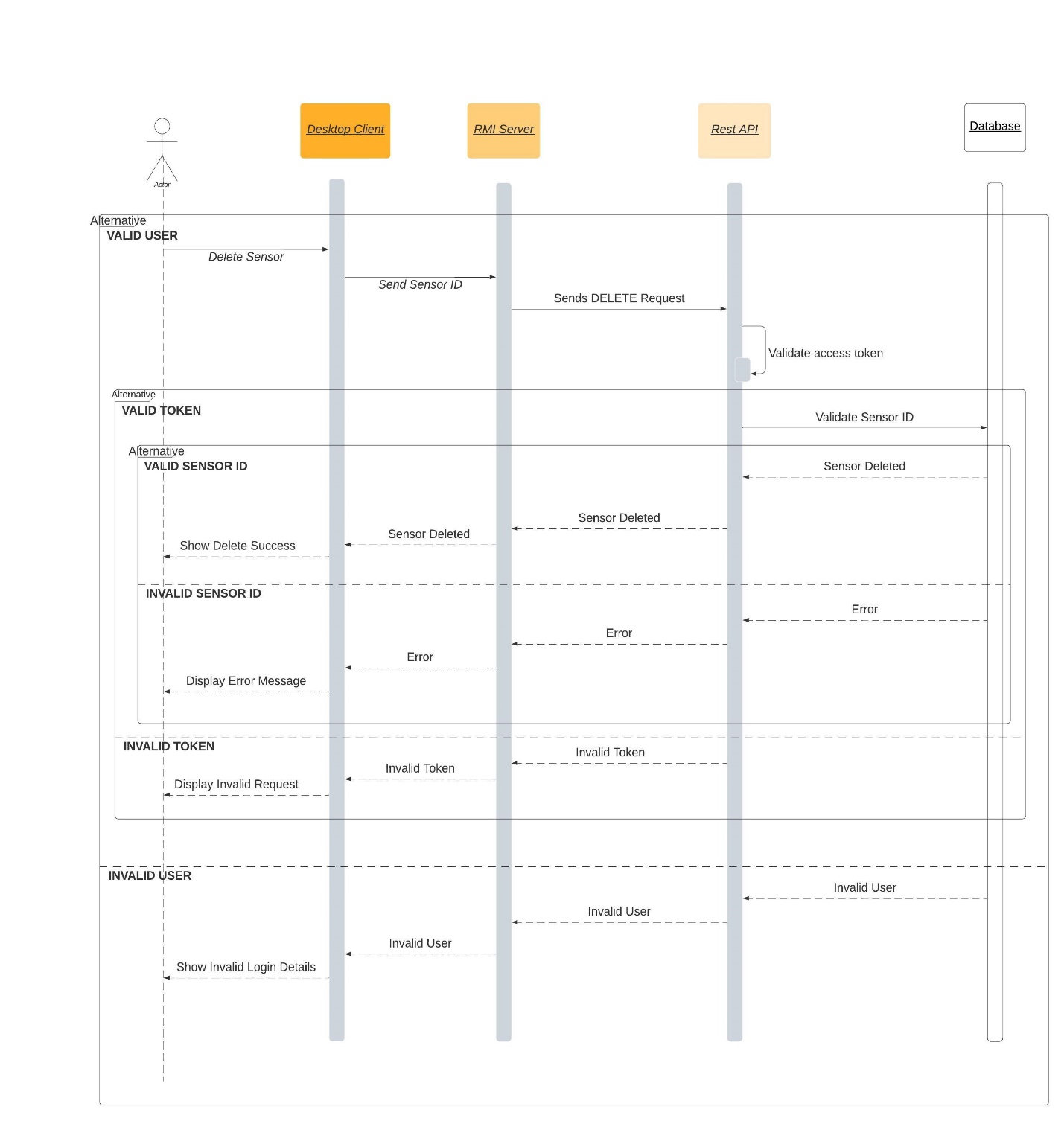
All the forms related to login, add, edit and delete are validated so that the user can submit only valid data. For example, when adding and editing sensors, the admin is only able to input numerical values to the floor number and room number input fields. Validations have also been applied to ensure that empty forms cannot be submitted.

API: https://fire-alarm-api-ds.herokuapp.com

Below sequence diagram explains the workflow of the desktop client.



**…. continuation of the previous sequence diagram**



As shown in the above sequence diagram, when the user submits the login form with the login details, the entered login details are validated. If such an user exists in the database, an access token which is specific to an admin is generated at the REST API and an admin dashboard is displayed to the user giving the administrative privileges to add, edit and delete sensor details. These administrative privileges are given to the user after validating the admin access token, and if it is invalid, the user is unable to add, edit and delete sensor details.

**API Response for Login Request:**

**A screenshot of a cell phone

Description automatically generated**

**Adding a new Sensor**

When a user tries to add a new sensor, the desktop client sends the sensor details entered by the user to the RMI. Thereafter, the RMI sends those information to the webserver which hosts the REST API as a POST request with the admin access token which was received during login. Then on the web server, the admin access token is validated first to ensure that the request is authorized to add a new sensor. If the authorization fails, an error message is sent to the RMI. If the authorization is successful, then the information entered by user is sent for validation. If the validation is successful, then the new sensor details are added to the database and new sensor ID is generated automatically.

A screenshot of a cell phone

Description automatically generated

**Editing a Sensor**

In the event of editing the sensor details, the desktop client sends the information entered by the user to the RMI. Thereafter, the RMI sends those information to the webserver as PUT request with the access token of the admin. Inside the webserver, the admin access token is validated to ensure that the admin has the authorization to send a PUT request. After successfully validating the admin access token, the information entered by user is passed for validation. During the validation, it is checked whether a sensor with the ID sent by the admin exists. If such a sensor exits, the sensor details of that specific sensor is updated. If there is no sensor matching the sensor ID sent by the admin, then an error message is displayed.

A screenshot of a cell phone

Description automatically generated

**Deleting a Sensor**

In case the user wants to delete a sensor, the desktop client sends the id of the sensor that needs to be deleted to the RMI. Afterwards, the RMI sends the information to the web server as a DELETE request with the admin access token. Then in the server, the admin access token is validated to ensure that the request is authorized to delete an existing sensor. If the authorization fails, an error message is sent to the RMI server. If the authorization is successful, the sensor ID that needs to be deleted is sent for validation. Here, the server checks whether a sensor with the given ID exists in the database, and if such a sensor exists, that particular sensor is deleted and a success message is sent back as a response. If such a sensor does not exist, then an error message is sent as a response to the RMI server.

A screenshot of a cell phone

Description automatically generated

**Sensor Application**

Sensor application mimics the functionality of an actual sensor. It sends Sensor Data such as CO2 and Smoke levels, to the REST API every 10 seconds. Before sending the data, the sensor needs to be authenticated as valid sensor. If the sensor is not a sensor which is registered by the admin, then the authentication will fail, but if you get room number or the floor number is wrong, then the sensor will auto correct itself. You will be able to change the data manually.

A screenshot of a cell phone

Description automatically generated

**Response of a server PUT request sent by sensor to update the CO2 and Smoke level.**

**A screenshot of a cell phone

Description automatically generated**

**How the REST API determine if the sensor is active or inactive**

The API records the time stamp of the last request sent by the Sensor along with the levels. Then if there is a time gap more than 10 seconds between the current time and the last sensor data updated time, then the server decides that the sensor is inactive.