A picture containing drawing

Description automatically generated

Sri Lanka Institute of Information Technology

Fire Alarm Monitoring System

**SE3020**

Group Report

|  |  |
| --- | --- |
| **IT number** | **Name with initials** |
| IT18063288 | Samarasekara H. D. K. |
| IT18050240 | Rajapaksha S. D. D. |
| IT18049114 | Hansaka H. M. P. |
| IT18058338 | Tharaka W.A.D.G. |

Introduction

We developed a fire alarm monitoring system that refreshes every 40 seconds and give the status of the co2 level / smoke level. It has an administer login where admin can add/register new fire alarm systems. It also have a web application where users can see the fire alarm sensors. We used React to develop the web application and API was developed using Node & Express. Desktop client and server was developed using NetBeans and mongo DB was used as the database.

High Level Architectural Diagram

A close up of a map

Description automatically generated

Figure 1: High level architectural diagram

Sequence Diagram for Desktop Client

A screenshot of a cell phone

Description automatically generated

Figure : Sequence diagram for desktop client

Sequence Diagram for Web Client

A close up of a map

Description automatically generated

Figure : Sequence diagram for web client

Sequence Diagram for Sensor

A screenshot of a social media post

Description automatically generated

Figure :Sequence diagram for sensor

Flow Chart for Desktop Application

A close up of text on a white background

Description automatically generated

Figure : Flow chart for desktop

Flow Chart for Web Application

A close up of text on a white background

Description automatically generated

Figure : Flow chart of web application

Code

* In the Client Desktop we used the method to get sensor details from DB.

public String getSensorData() throws MalformedURLException, IOException {

String newfinaltext2 = null;

URL link = new URL("http://localhost:5000/infos/");

System.out.println("link fetch complete");

URLConnection conn = link.openConnection();

System.out.println("4");

try (BufferedReader buff = new BufferedReader(new InputStreamReader(conn.getInputStream()))) {

String line;

while ((line = buff.readLine()) != null) {

finalText = line;

}

System.out.println(finalText);

String newfinaltext = finalText.replace("[", "");

newfinaltext2 = newfinaltext.replace("]", "");

System.out.println(newfinaltext2);

System.out.println("completed");

} catch (IOException ex) {

ex.printStackTrace();

System.out.println(ex.getMessage());

}

return newfinaltext2;

}

* We used these methods to insert Data.

public String getSensorData() throws MalformedURLException, IOException {

String newfinaltext2 = null;

URL link = new URL("http://localhost:5000/infos/");

System.out.println("link fetch complete");

URLConnection conn = link.openConnection();

System.out.println("4");

try (BufferedReader buff = new BufferedReader(new InputStreamReader(conn.getInputStream()))) {

String line;

while ((line = buff.readLine()) != null) {

//System.out.println(line);

finalText = line;

}

System.out.println(finalText);

String newfinaltext = finalText.replace("[", "");

newfinaltext2 = newfinaltext.replace("]", "");

System.out.println(newfinaltext2);

System.out.println("completed");

} catch (IOException ex) {

ex.printStackTrace();

System.out.println(ex.getMessage());

}

return newfinaltext2;

}

public void postDetails(String postData) {

try {

URL urls = new URL("http://localhost:5000/infos/add");

//String postData = "foo1=bar1&foo2=bar2";

HttpURLConnection conn = (HttpURLConnection) urls.openConnection();

conn.setRequestMethod("POST");

conn.setDoOutput(true);

conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

conn.setRequestProperty("Content-Length", Integer.toString(postData.length()));

conn.setUseCaches(false);

try (DataOutputStream dos = new DataOutputStream(conn.getOutputStream())) {

dos.writeBytes(postData);

}

try (BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()))) {

String line;

while ((line = br.readLine()) != null) {

System.out.println(line);

}

}

} catch (Exception e) {

e.printStackTrace();

System.out.println(e.getMessage());

}

}

public void postSensorDetails(String postSensorData) {

try {

URL urls = new URL("http://localhost:5000/sensor/add");

//String postData = "foo1=bar1&foo2=bar2";

HttpURLConnection conn = (HttpURLConnection) urls.openConnection();

conn.setRequestMethod("POST");

conn.setDoOutput(true);

conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

conn.setRequestProperty("Content-Length", Integer.toString(postSensorData.length()));

conn.setUseCaches(false);

try (DataOutputStream dos = new DataOutputStream(conn.getOutputStream())) {

dos.writeBytes(postSensorData);

}

try (BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()))) {

String line;

while ((line = br.readLine()) != null) {

System.out.println(line);

}

}

} catch (Exception e) {

e.printStackTrace();

System.out.println(e.getMessage());

}

}

Generate Randum Value

public int getReadingco2(){

int min = 0;

int max = 10;

return (int) ((Math.random() \* ((max - min) + 1)) + min);

}

public int getReadingSmoke(){

int min = 0;

int max = 10;

return int) ((Math.random() \* ((max - min) + 1)) + min);

}

}

* Every 40 second send senor details.

public static void main(String[] args) {

Service s1 = new Service();

try {

Registry reg = LocateRegistry.createRegistry(1099);

reg.rebind("server", new RMIServer());

System.out.println("server started");

} catch (Exception e) {

System.out.println(e);

}

Timer timer = new Timer();

TimerTask task = new TimerTask() {

public void run() {

System.out.println("Every 40 second");

s1.addSensorDetails();

}

};

timer.schedule(task, 0L, 40000L);

System.out.println("After thread");

}

**Web project souse code (not include all source code)**

import React, { Component } from 'react'

import axios from 'axios';

const Info = props =>(

<div>

{props.info.smokeLevel > 5

? ( <div>

<br/>

<div className="card" style={{backgroundColor: "#FF3333",width:"50%",marginLeft:"25%",borderRadius:"10%"}}>

<h5><b>Active : </b>{props.info.activity}</h5>

<br/>

<h5><b>Co2 Level : </b>{props.info.co2Level}</h5>

<br/>

<h5><b>Smoke Level : </b>{props.info.smokeLevel}</h5>

<br/>

<h5><b>Floor No : </b>{props.info.floorNo}</h5>

<br/>

<h5><b>Room No : </b>{props.info.roomNo}</h5>

<br/>

</div>

</div>

)

:(<div>

<br/>

<div className="card" style={{backgroundColor: "#3EFF44",width:"50%",marginLeft:"25%",borderRadius:"10%"}}>

<h5><b>Active : </b>{props.info.activity}</h5>

<br/>

<h5><b>Co2 Level : </b>{props.info.co2Level}</h5>

<br/>

<h5><b>Smoke Level : </b>{props.info.smokeLevel}</h5>

<br/>

<h5><b>Floor No : </b>{props.info.floorNo}</h5>

<br/>

<h5><b>Room No : </b>{props.info.roomNo}</h5>

<br/>

</div>

</div>)

}

</div>

);

export default class InformationView extends Component {

constructor(props) {

super(props);

this.state ={

infoArray: []};

this.infoList = this.infoList.bind(this);

}

componentDidMount() {

axios.get('http://localhost:5000/infos/')

.then(response =>{

this.setState({infoArray:response.data});

})

.catch((error) => {

console.log(error);

})

}

infoList(){

return this.state.infoArray.map(function (currentinfo,i) {

return <Info info = {currentinfo} key={i}/>

});

}

render() {

return (

<div className="App">

<br/>

<div className="container">

<div>{this.infoList()}</div>

</div>

</div>

)

}

}

***Info.model.js***

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const infoSchema = new Schema({

activity:{type:String,required:true},

co2Level:{type: Number,required: true},

smokeLevel:{type: Number,required: true},

floorNo:{type: Number,required: true},

roomNo:{type: Number,required: true},

},{

timeStamp:true,

});

const Info = mongoose.model('Info', infoSchema);

module.exports=Info;

***Sensor.model.js***

const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const sensorSchema = new Schema({

status:{type:Boolean,required:true},

floorNo:{type: Number,required: true},

roomNo:{type: Number,required: true},

});

const sensor = mongoose.model('sensor', sensorSchema);

module.exports=sensor;

***Route > Sensor.js***

const router = require('express').Router();

let Sensor = require('../model/sensor.model');

router.route('/').get((req,res) => {

Sensor.find()

.then(infos => res.json(infos))

.catch(err => res.status(400).json('Error: ' + err));

});

router.route('/add').post((req,res) => {

const status= req.body.status;

const floorNo= req.body.floorNo;

const roomNo= req.body.roomNo;

const sensor = new Sensor({

status,

floorNo,

roomNo,

});

sensor.save()

.then(() => res.json('Info Added'))

.catch(err => res.status(400).json('Error: ' + err));

});

module.exports = router;