웹과 아두이노를 이용한 스마트팜 제어

이미지 강혜미 송승준

정윤미 차정석

목차

스마트팜이란?

농 · 림 · 축 · 수산물의 생산, 가공, 유통 단계에서
정보 통신 기술(ICT)을 접목하여 지능화된 농업 시스템

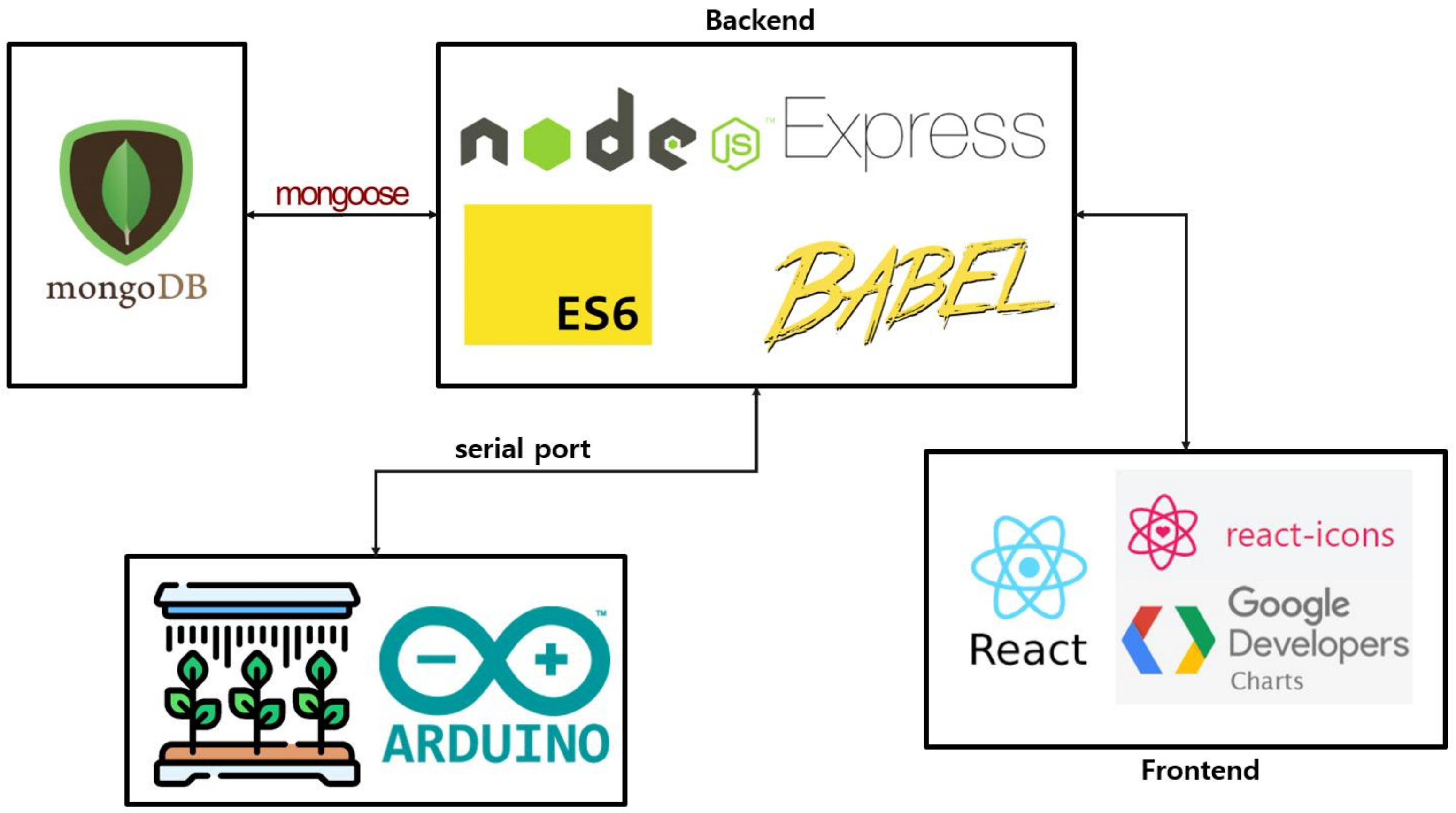
사물 인터넷, 빅데이터, 인공 지능 등의
기술을 이용하여 농작물, 가축 및 수산물 등의
생육 환경을 적정하게 유지 · 관리하고, PC와 스마트폰
등으로 원격에서 자동 관리할 수 있어, 생산의 효율성 뿐만
아니라 편리성도 높일 수 있다.

ICT 기술을 활용한 스마트팜 기술을 통해
환경 정보(온도 · 상대습도 · 광량 · 이산화탄소 · 토양 등) 및
생육 정보에 대한 정확한 데이터를 기반으로 생육 단계별 정밀한
관리와 예측 등이 가능하여 수확량, 품질 등을 향상시켜
수익성을 높일 수 있다.

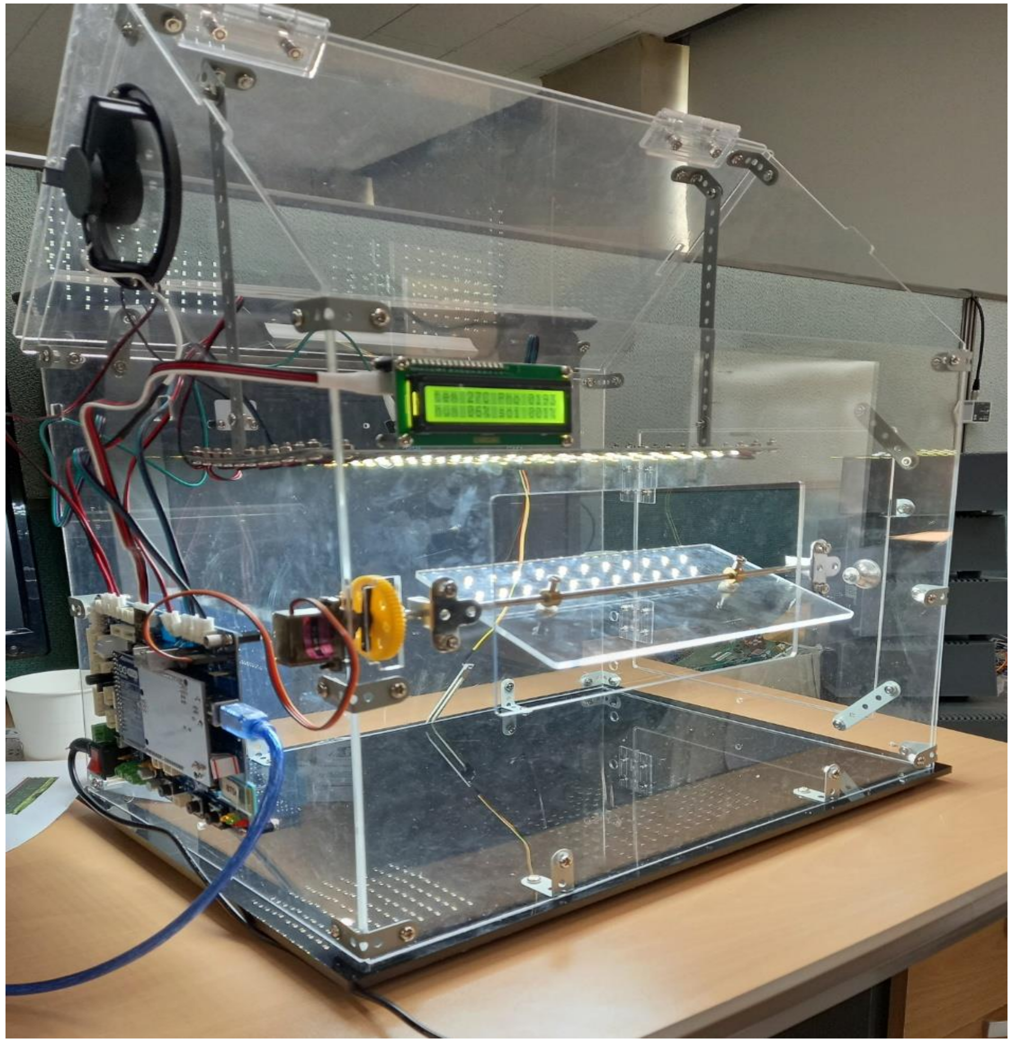
또한, 노동력과 에너지를 효율적으로 관리함으로써
생산비를 절감할 수 있다.

예를 들면, 기존에는 작물에 관수할 때
직접 밸브를 열고 모터를 작동해야 했다면, 스마트 팜에서는
전자밸브가 설정값에 맞춰 자동으로 관수를 한다.
또한, 스마트 팜은 농 · 림 · 축 · 수산물의 상세한
생산 정보 이력을 관리할 수 있어 소비자 신뢰도를 높일 수 있다.

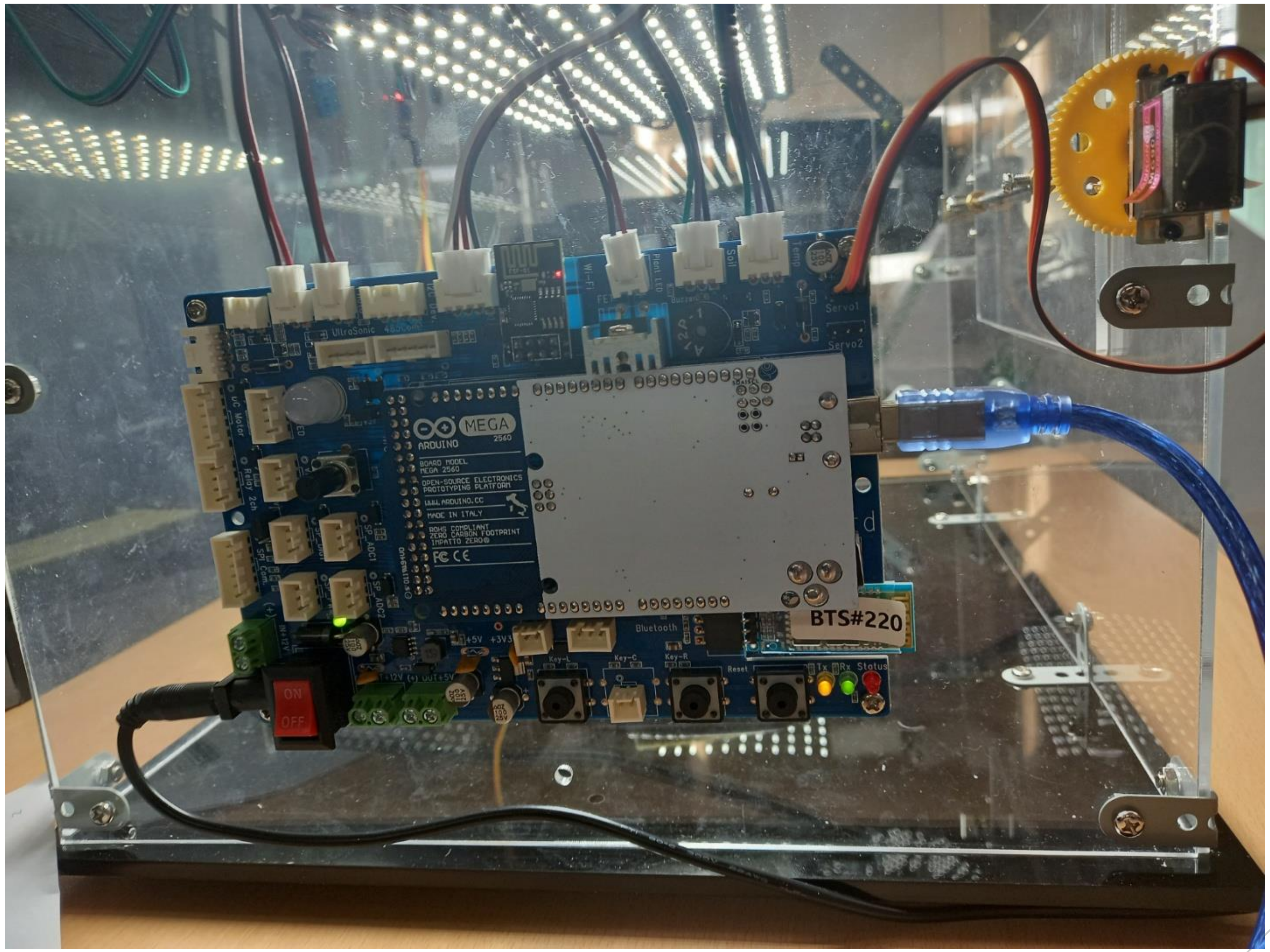
전체 아키텍쳐


Backend
mongoDB
mongoose
node js
Express
ES6
BABEL
serial port
ARDUINO
React
react-icons
Google Developers Charts
Frontend

smartfarm 구조

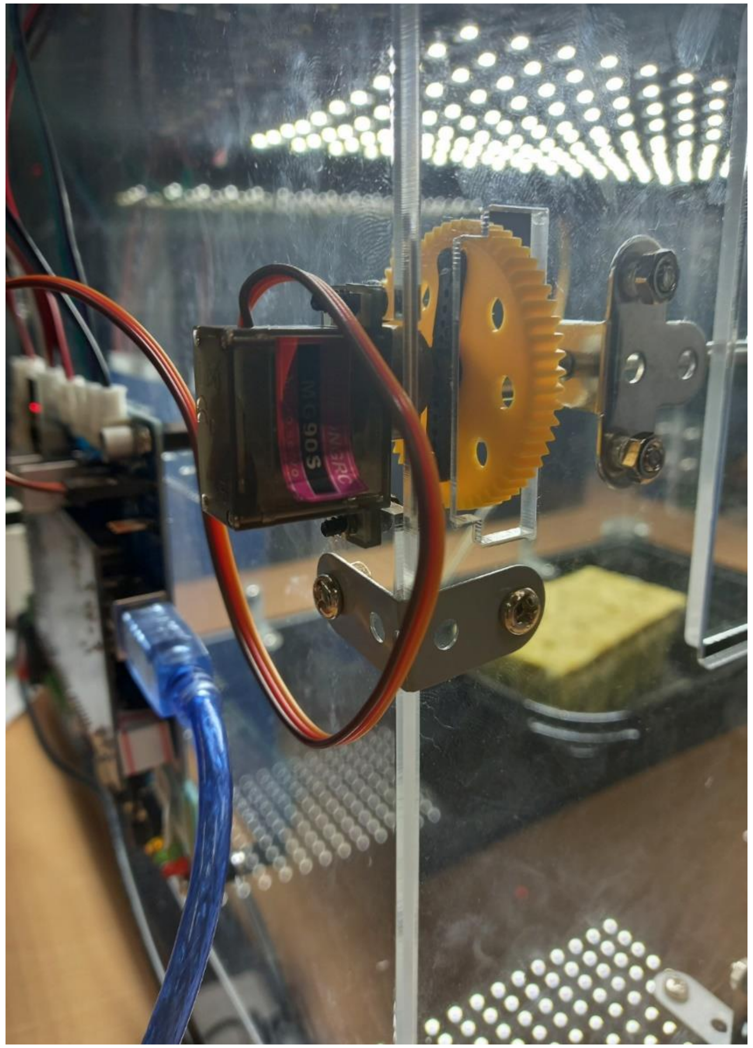


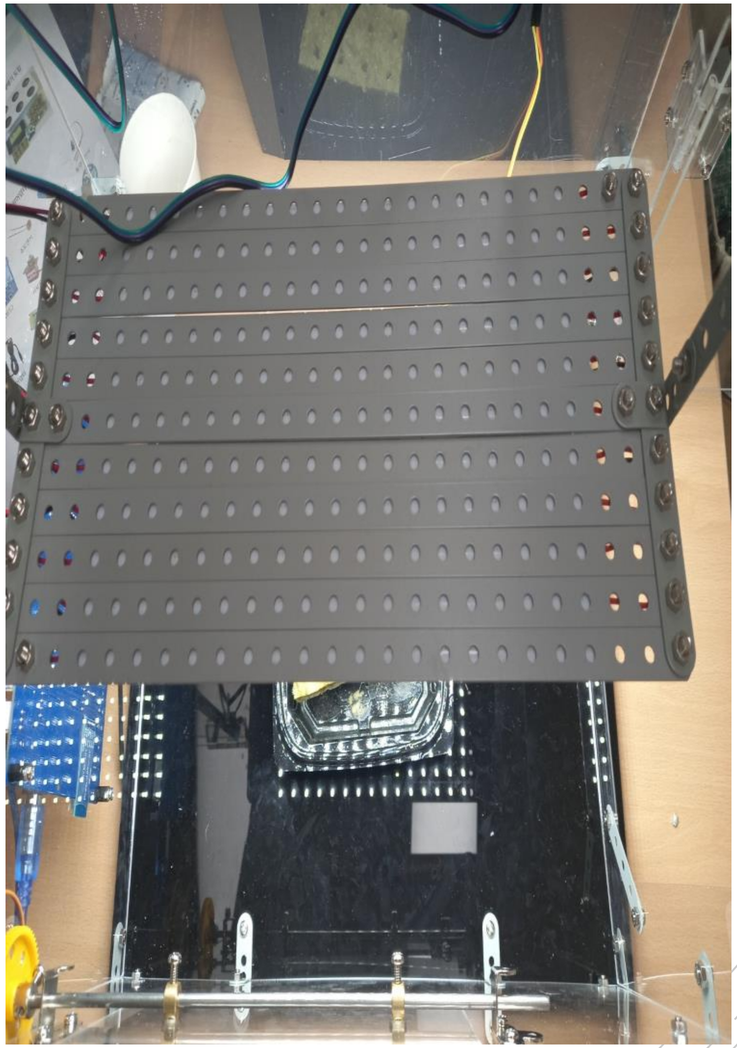
smartfarm 구조



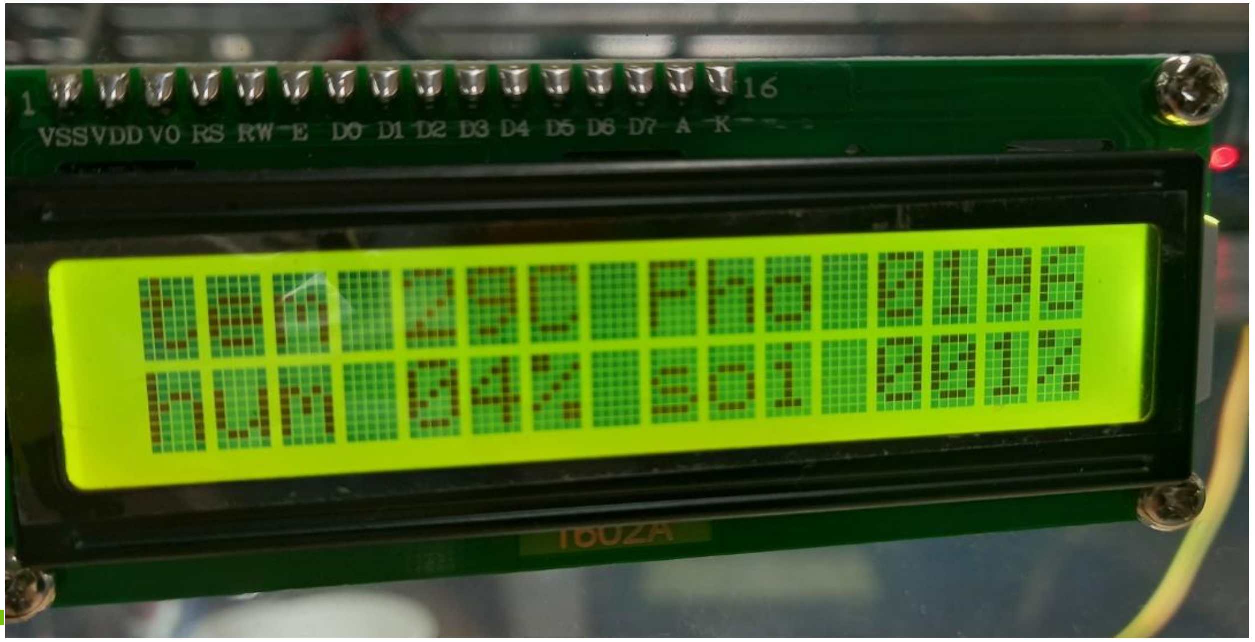
Arduino mega board 2560

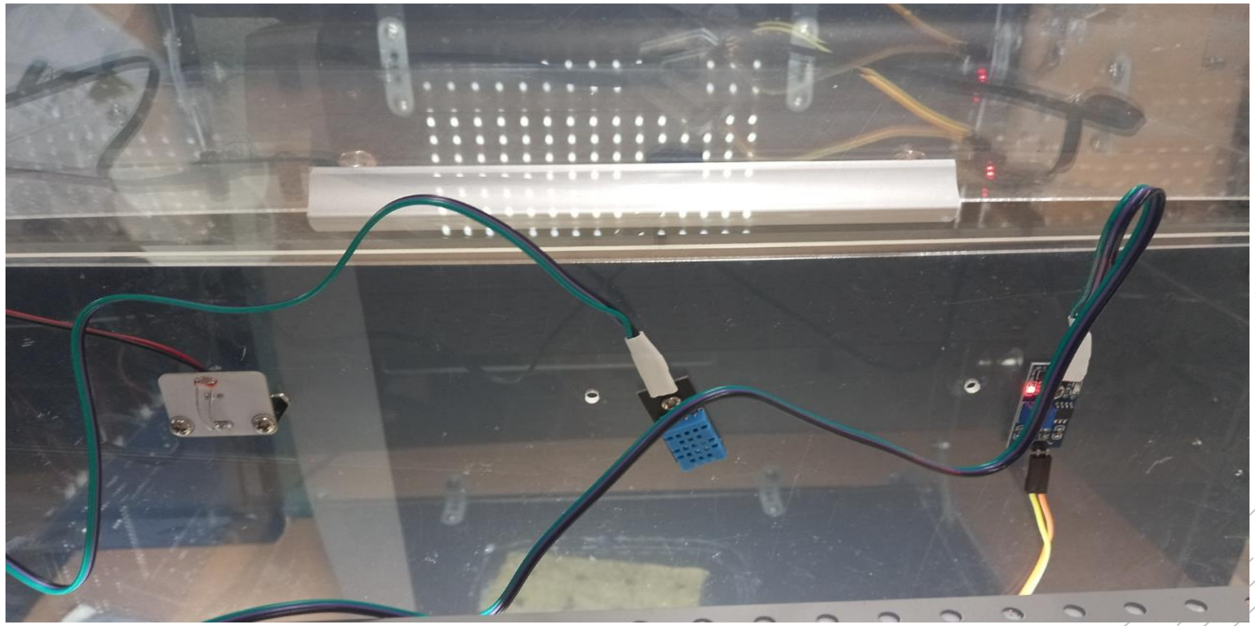
smartfarm 구조


서보모터


LED

smartfarm 구조



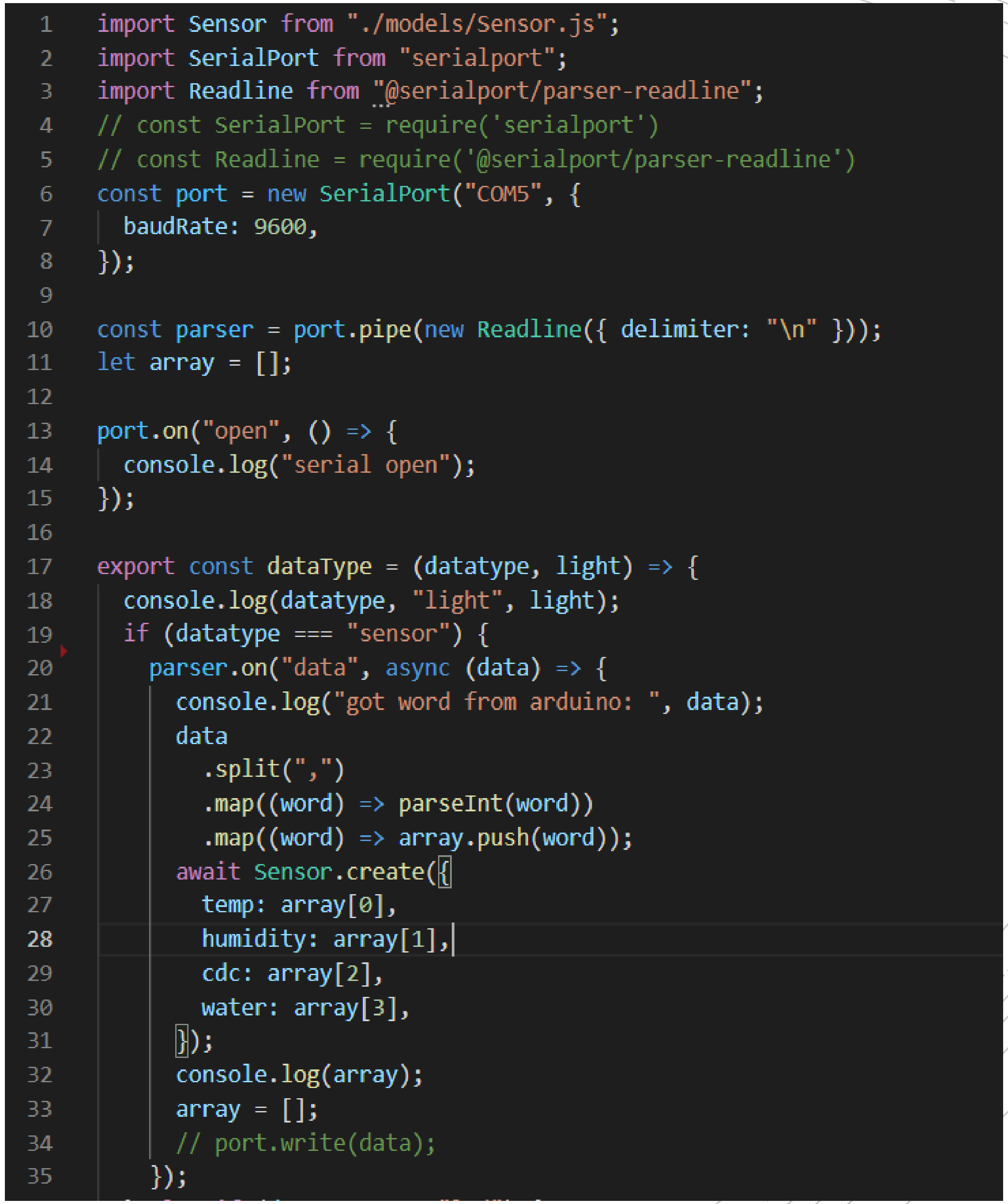


Arduino code

```
#include <Servo.h>

#include <DHT.h>

#include <Wire.h>

#include <LiquidCrystal_I2C.h>
```

Arduino code

```
Serial.print(humidity);

Serial.print(",");

Serial.print(temperature);

Serial.print(",");

Serial.print(cdcValue);

Serial.print(",");

Serial.println(waterValue);
```



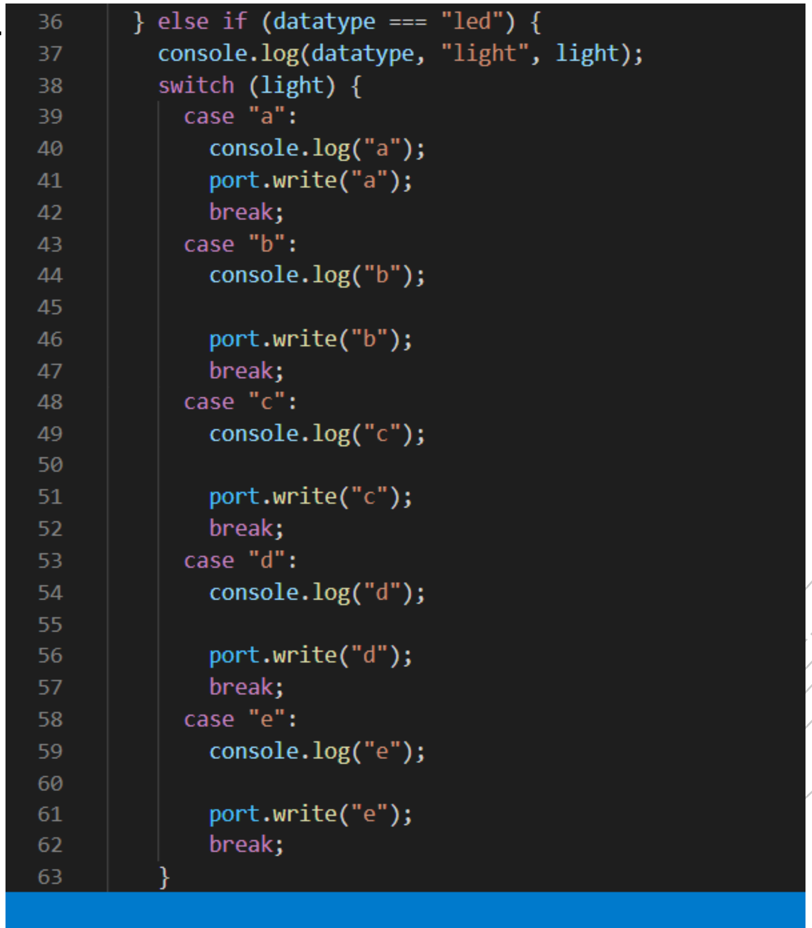
```
import Sensor from "./models/Sensor.js";
import SerialPort from "serialport";
import Readline from "@serialport/parser-readline";
// const SerialPort = require('serialport')
// const Readline = require('@serialport/parser-readline')
const port = new SerialPort("COM5", {
  baudRate: 9600,
});

const parser = port.pipe(new Readline({ delimiter: "\n" }));
let array = [];

port.on("open", () => {
  console.log("serial open");
});

export const dataType = (datatype, light) => {
  console.log(datatype, "light", light);
  if (datatype === "sensor") {
    parser.on("data", async (data) => {
      console.log("got word from arduino: ", data);
      data
        .split(",")
        .map((word) => parseInt(word))
        .map((word) => array.push(word));
      await Sensor.create({
        temp: array[0],
        humidity: array[1],
        cdc: array[2],
        water: array[3],
      });
      console.log(array);
      array = [];
      // port.write(data);
    });
```

Arduino code

```
//servo
 if(waterValue<50){
   myservo.write(90);
   delay(1000);
 }

 //LED
 if(Serial.available()>0){
   char light;
   light=(char) Serial.read();
   LEDControl(light);
 }
```

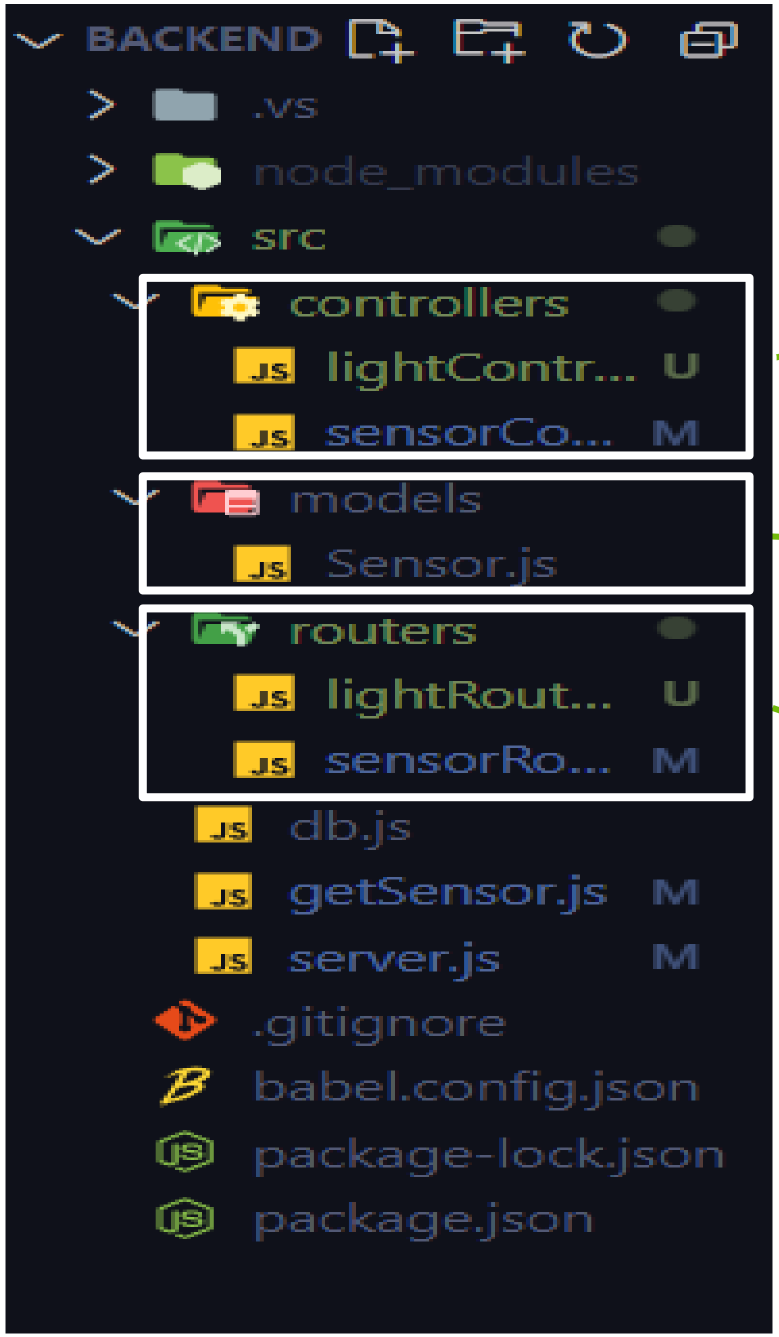
Arduino code

```
void LEDControl(char chr){
 if(chr=='a'){
  analogWrite(4,0);
  delay(1000);
 }
 if(chr=='b'){
  analogWrite(4,50);
  delay(1000);
 }
 if(chr=='c'){
  analogWrite(4,100);
  delay(1000);
 }
 if(chr=='d'){
  analogWrite(4,150);
  delay(1000);
 }
 if(chr=='e'){
  analogWrite(4,255);
  delay(1000);
 }
}
```



```
  } else if (datatype === "led") {
    console.log(datatype, "light", light);
    switch (light) {
      case "a":
        console.log("a");
        port.write("a");
        break;
      case "b":
        console.log("b");

        port.write("b");
        break;
      case "c":
        console.log("c");

        port.write("c");
        break;
      case "d":
        console.log("d");

        port.write("d");
        break;
      case "e":
        console.log("e");

        port.write("e");
        break;
    }
```

BACKEND 구조



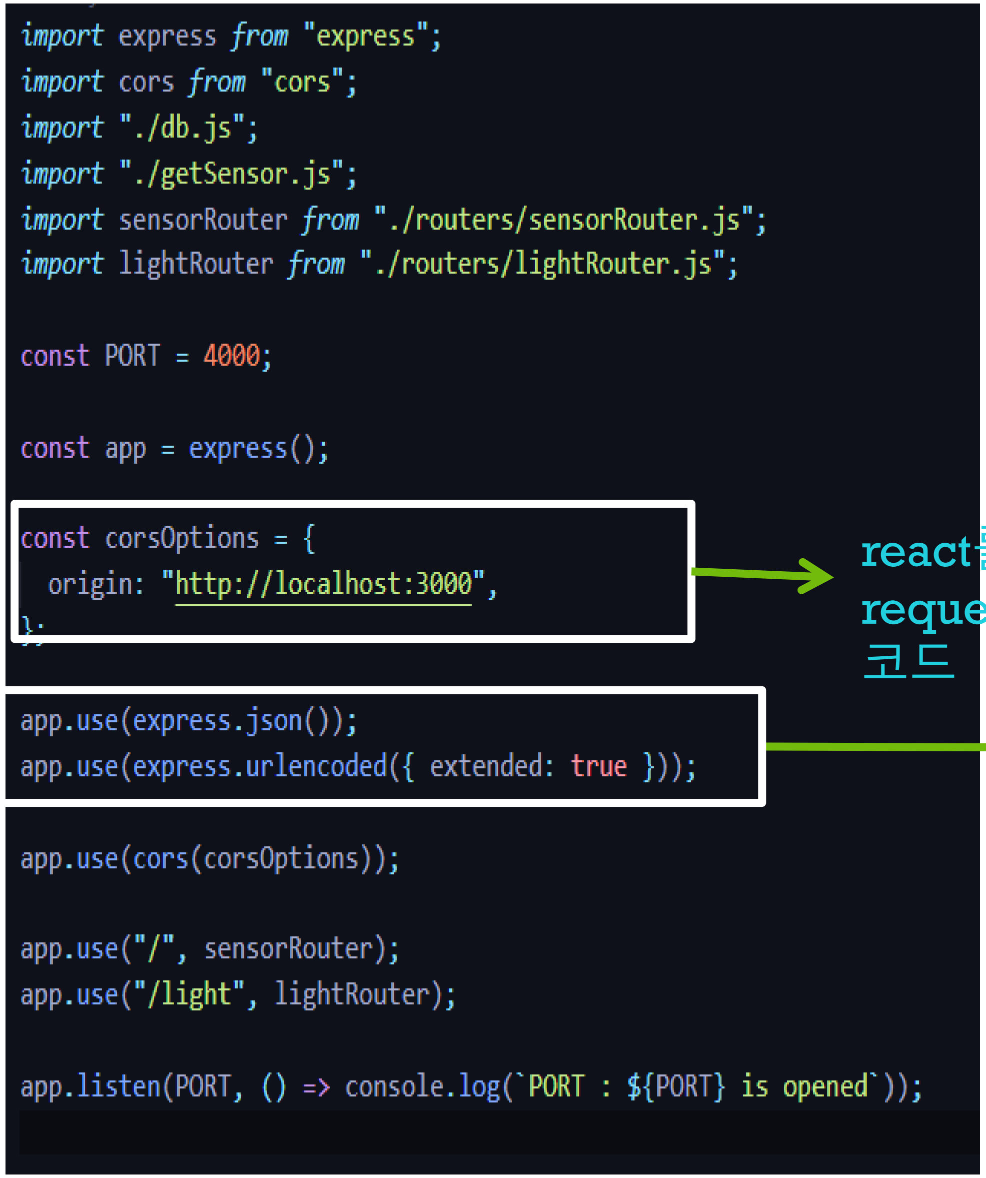
```
BACKEND
  > .vs
  > node_modules
  v src
    v controllers
        lightContr...   U
        sensorCo...     M
    v models
        Sensor.js
    v routers
        lightRout...    U
        sensorRo...     M
      db.js
      getSensor.js      M
      server.js         M
    .gitignore
    babel.config.json
    package-lock.json
    package.json
```

request에 대한 처리 담당

센서 데이터 저장을 위한 **Model** 생성

url 관리를 위한 라우터 생성

Server.js

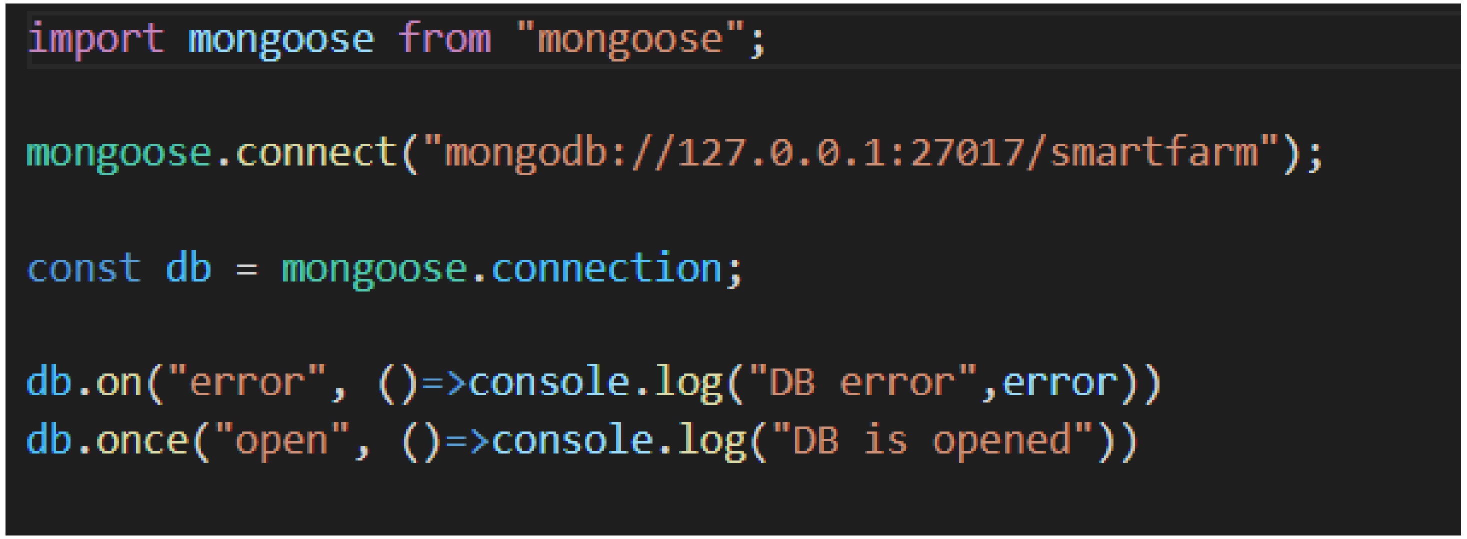


```
import express from "express";
import cors from "cors";
import "./db.js";
import "./getSensor.js";
import sensorRouter from "./routers/sensorRouter.js";
import lightRouter from "./routers/lightRouter.js";

const PORT = 4000;

const app = express();

const corsOptions = {
  origin: "http://localhost:3000",
};

app.use(express.json());
app.use(express.urlencoded({ extended: true }));

app.use(cors(corsOptions));

app.use("/", sensorRouter);
app.use("/light", lightRouter);

app.listen(PORT, () => console.log(`PORT : ${PORT} is opened`));
```

react를 사용하는 프론트의 request 요청을 받겠다는 코드

body-parser 사용

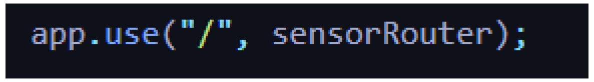
db.js



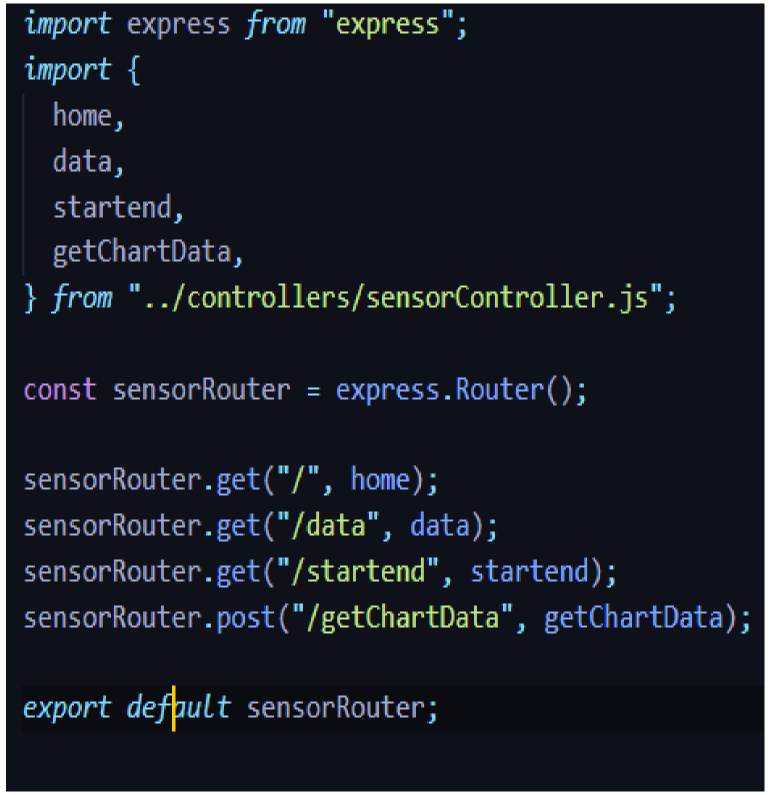
```
import mongoose from "mongoose";

mongoose.connect("mongodb://127.0.0.1:27017/smartfarm");

const db = mongoose.connection;

db.on("error", ()=>console.log("DB error",error))
db.once("open", ()=>console.log("DB is opened"))
```

Routers

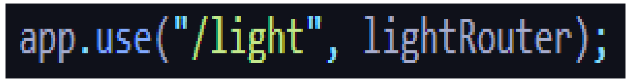
-sensorRouter
-lightRouter



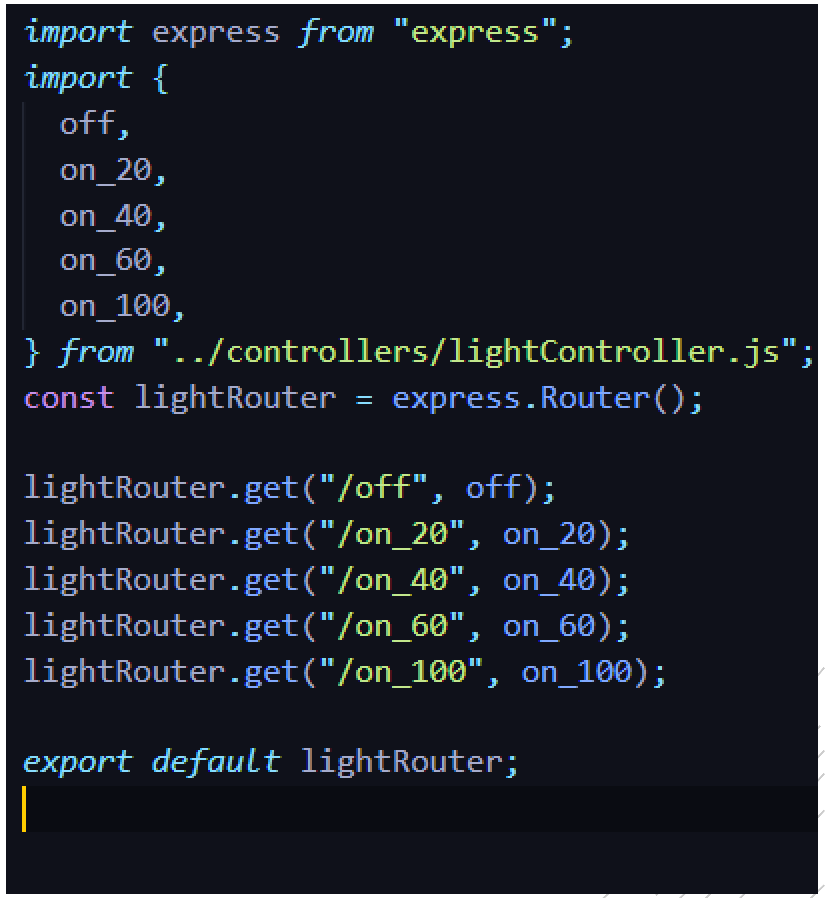
```
app.use("/", sensorRouter);
```



```
import express from "express";
import {
  home,
  data,
  startend,
  getChartData,
} from "../controllers/sensorController.js";

const sensorRouter = express.Router();

sensorRouter.get("/", home);
sensorRouter.get("/data", data);
sensorRouter.get("/startend", startend);
sensorRouter.post("/getChartData", getChartData);

export default sensorRouter;
```

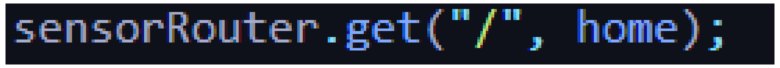


```
app.use("/light", lightRouter);
```

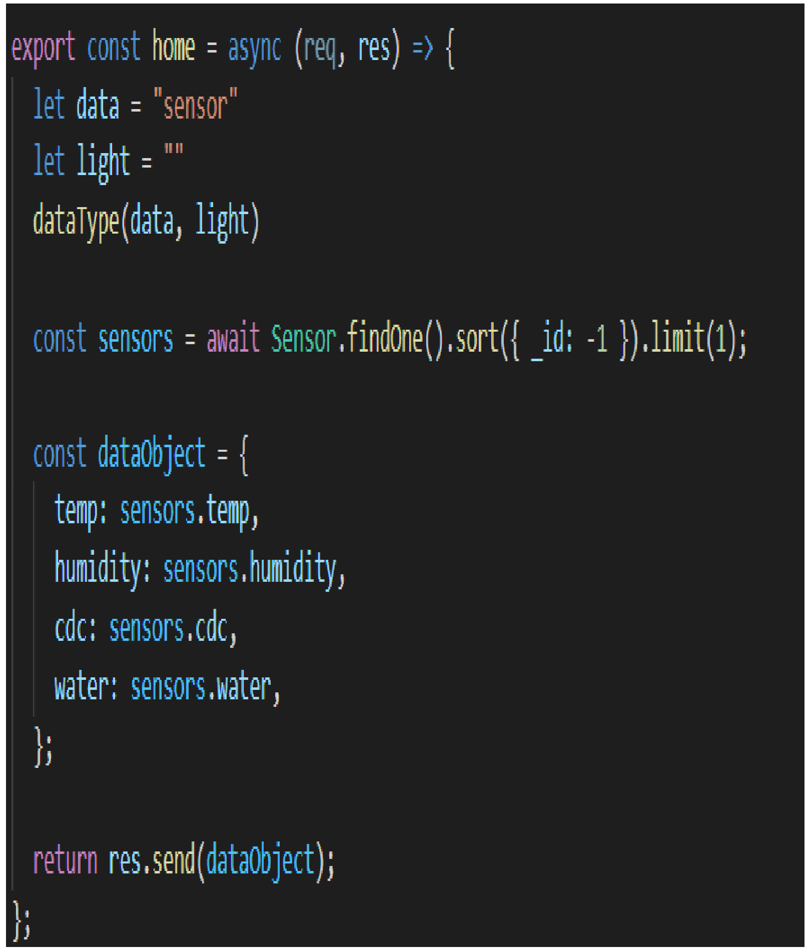


```
import express from "express";
import {
  off,
  on_20,
  on_40,
  on_60,
  on_100,
} from "../controllers/lightController.js";
const lightRouter = express.Router();

lightRouter.get("/off", off);
lightRouter.get("/on_20", on_20);
lightRouter.get("/on_40", on_40);
lightRouter.get("/on_60", on_60);
lightRouter.get("/on_100", on_100);

export default lightRouter;
```

Sensor Controller .js

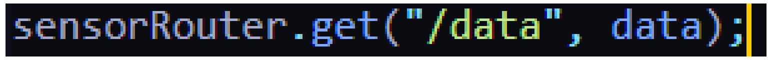


```
sensorRouter.get("/", home);
```

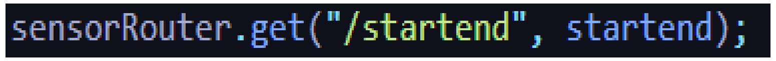


```
export const home = async (req, res) => {
  let data = "sensor"
  let light = ""
  dataType(data, light)

  const sensors = await Sensor.findOne().sort({ _id: -1 }).limit(1);

  const dataObject = {
    temp: sensors.temp,
    humidity: sensors.humidity,
    cdc: sensors.cdc,
    water: sensors.water,
  };

  return res.send(dataObject);
};
```

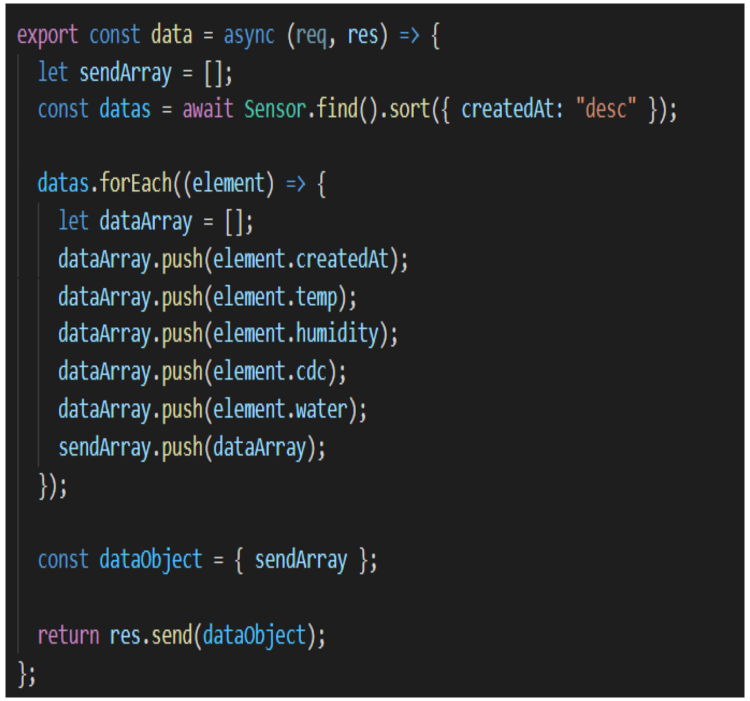
Sensor Controller.js



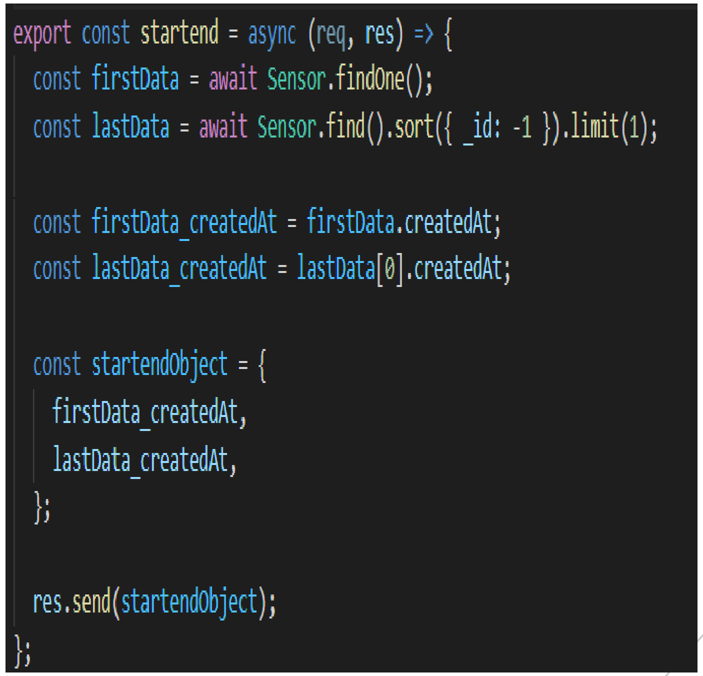
```
sensorRouter.get("/data", data);
```



```
sensorRouter.get("/startend", startend);
```

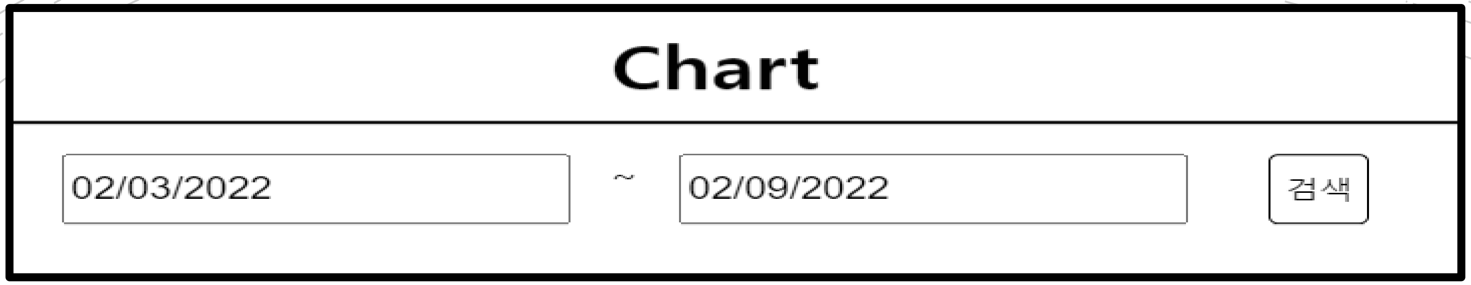


```
export const data = async (req, res) => {
  let sendArray = [];
  const datas = await Sensor.find().sort({ createdAt: "desc" });

  datas.forEach((element) => {
    let dataArray = [];
    dataArray.push(element.createdAt);
    dataArray.push(element.temp);
    dataArray.push(element.humidity);
    dataArray.push(element.cdc);
    dataArray.push(element.water);
    sendArray.push(dataArray);
  });

  const dataObject = { sendArray };

  return res.send(dataObject);
};
```



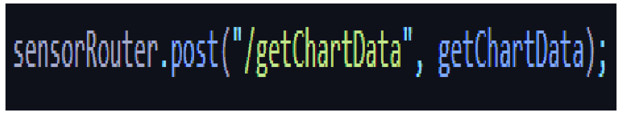
```
export const startend = async (req, res) => {
  const firstData = await Sensor.findOne();
  const lastData = await Sensor.find().sort({ _id: -1 }).limit(1);

  const firstData_createdAt = firstData.createdAt;
  const lastData_createdAt = lastData[0].createdAt;

  const startendObject = {
    firstData_createdAt,
    lastData_createdAt,
  };

  res.send(startendObject);
};
```

Sensor Controller.js

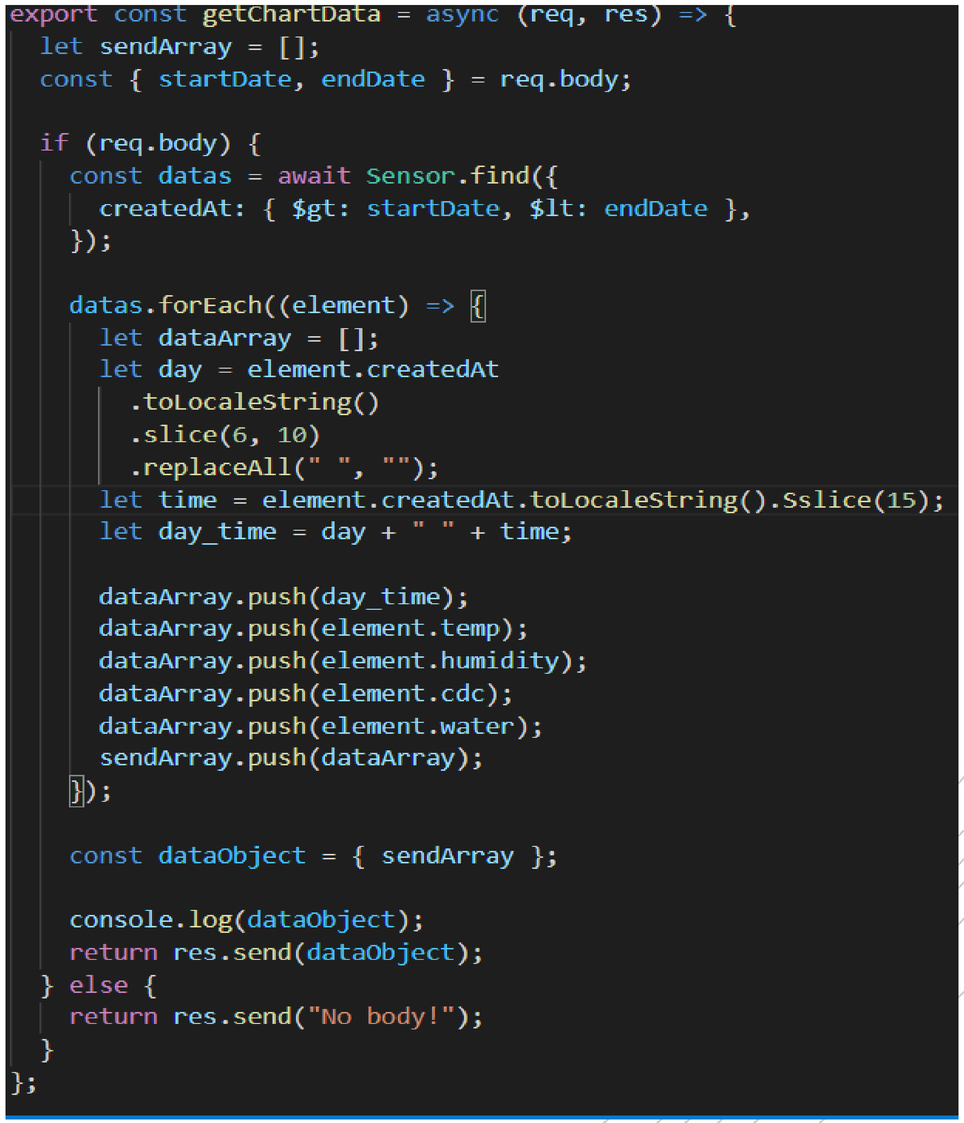


Chart

02/03/2022 ~ 02/09/2022 검색

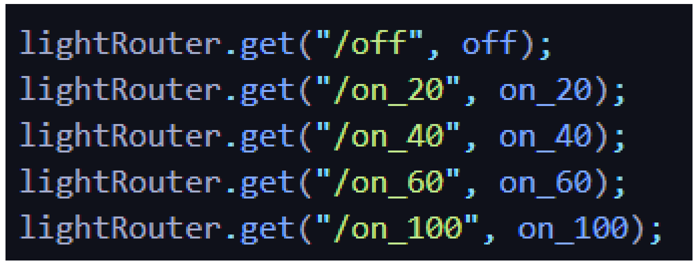


```
sensorRouter.post("/getChartData", getChartData);
```

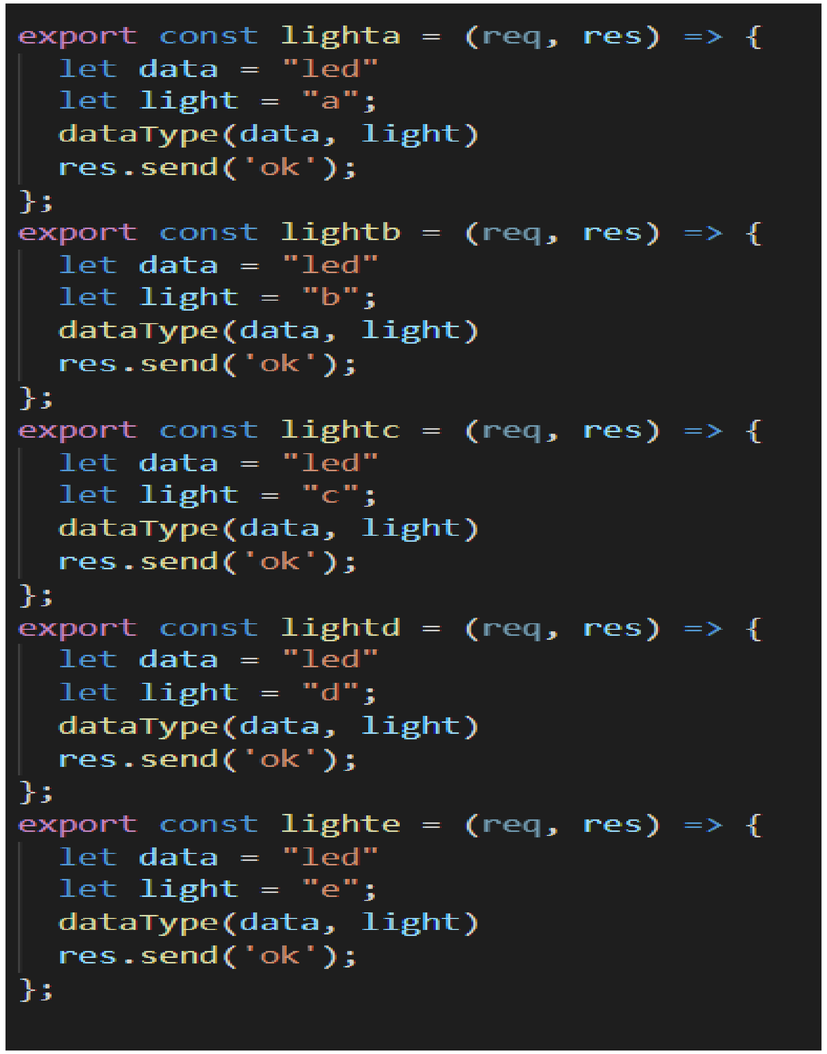


```
export const getChartData = async (req, res) => {
  let sendArray = [];
  const { startDate, endDate } = req.body;

  if (req.body) {
    const datas = await Sensor.find({
      createdAt: { $gt: startDate, $lt: endDate },
    });

    datas.forEach((element) => {
      let dataArray = [];
      let day = element.createdAt
        .toLocaleString()
        .slice(6, 10)
        .replaceAll(" ", "");
      let time = element.createdAt.toLocaleString().Sslice(15);
      let day_time = day + " " + time;

      dataArray.push(day_time);
      dataArray.push(element.temp);
      dataArray.push(element.humidity);
      dataArray.push(element.cdc);
      dataArray.push(element.water);
      sendArray.push(dataArray);
    });

    const dataObject = { sendArray };

    console.log(dataObject);
    return res.send(dataObject);
  } else {
    return res.send("No body!");
  }
};
```

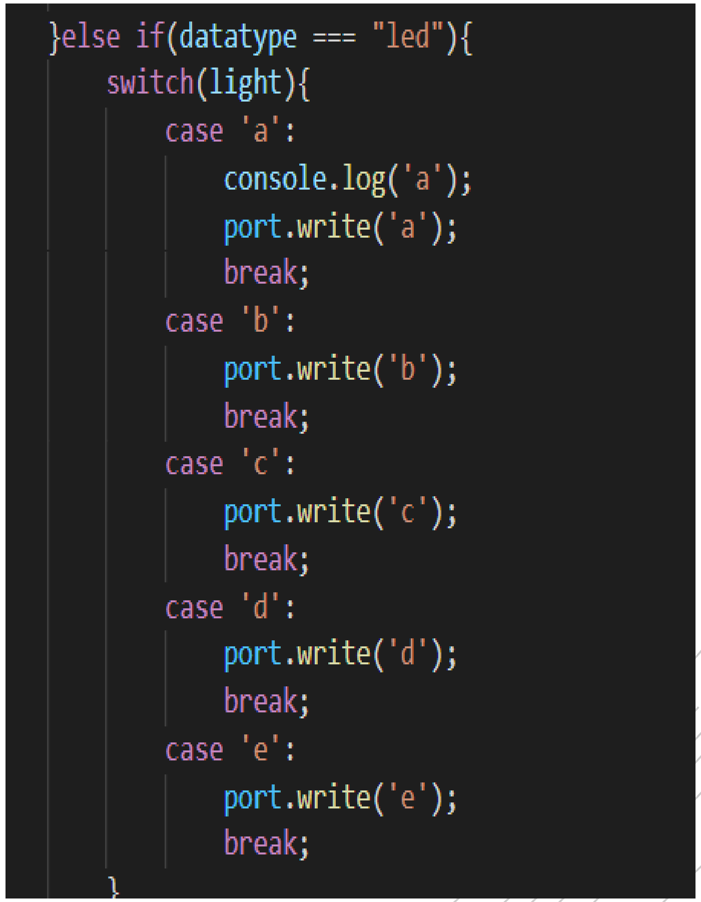
lightController.js



```
lightRouter.get("/off", off);
lightRouter.get("/on_20", on_20);
lightRouter.get("/on_40", on_40);
lightRouter.get("/on_60", on_60);
lightRouter.get("/on_100", on_100);
```

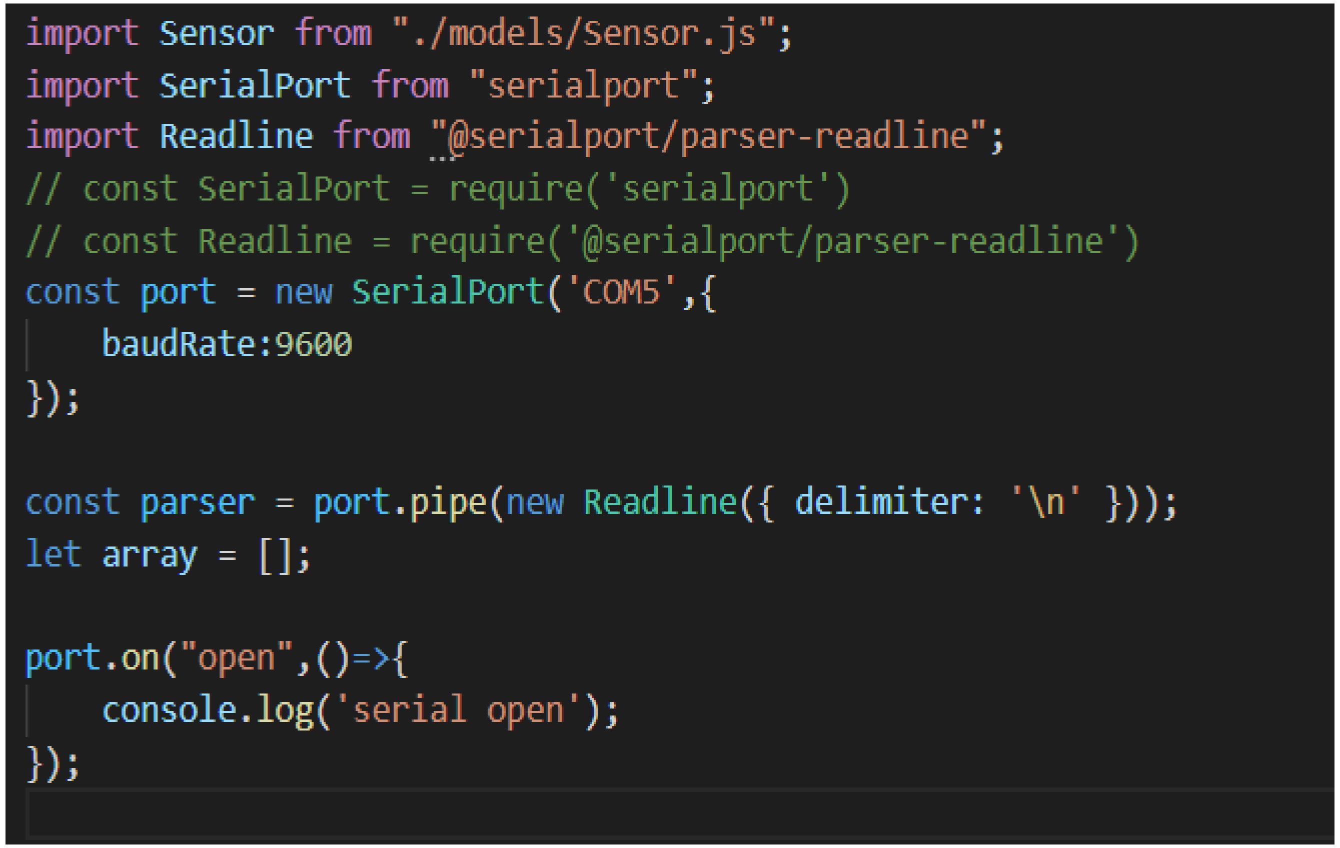


```
export const lighta = (req, res) => {
  let data = "led"
  let light = "a";
  dataType(data, light)
  res.send('ok');
};
export const lightb = (req, res) => {
  let data = "led"
  let light = "b";
  dataType(data, light)
  res.send('ok');
};
export const lightc = (req, res) => {
  let data = "led"
  let light = "c";
  dataType(data, light)
  res.send('ok');
};
export const lightd = (req, res) => {
  let data = "led"
  let light = "d";
  dataType(data, light)
  res.send('ok');
};
export const lighte = (req, res) => {
  let data = "led"
  let light = "e";
  dataType(data, light)
  res.send('ok');
};
```



```
}else if(datatype === "led"){
    switch(light){
        case 'a':
            console.log('a');
            port.write('a');
            break;
        case 'b':
            port.write('b');
            break;
        case 'c':
            port.write('c');
            break;
        case 'd':
            port.write('d');
            break;
        case 'e':
            port.write('e');
            break;
    }
```

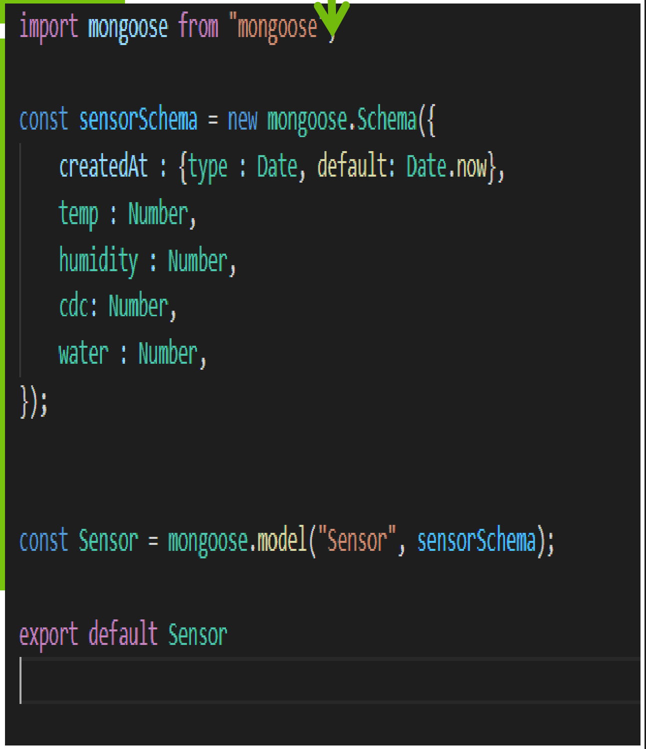
getSensor.js



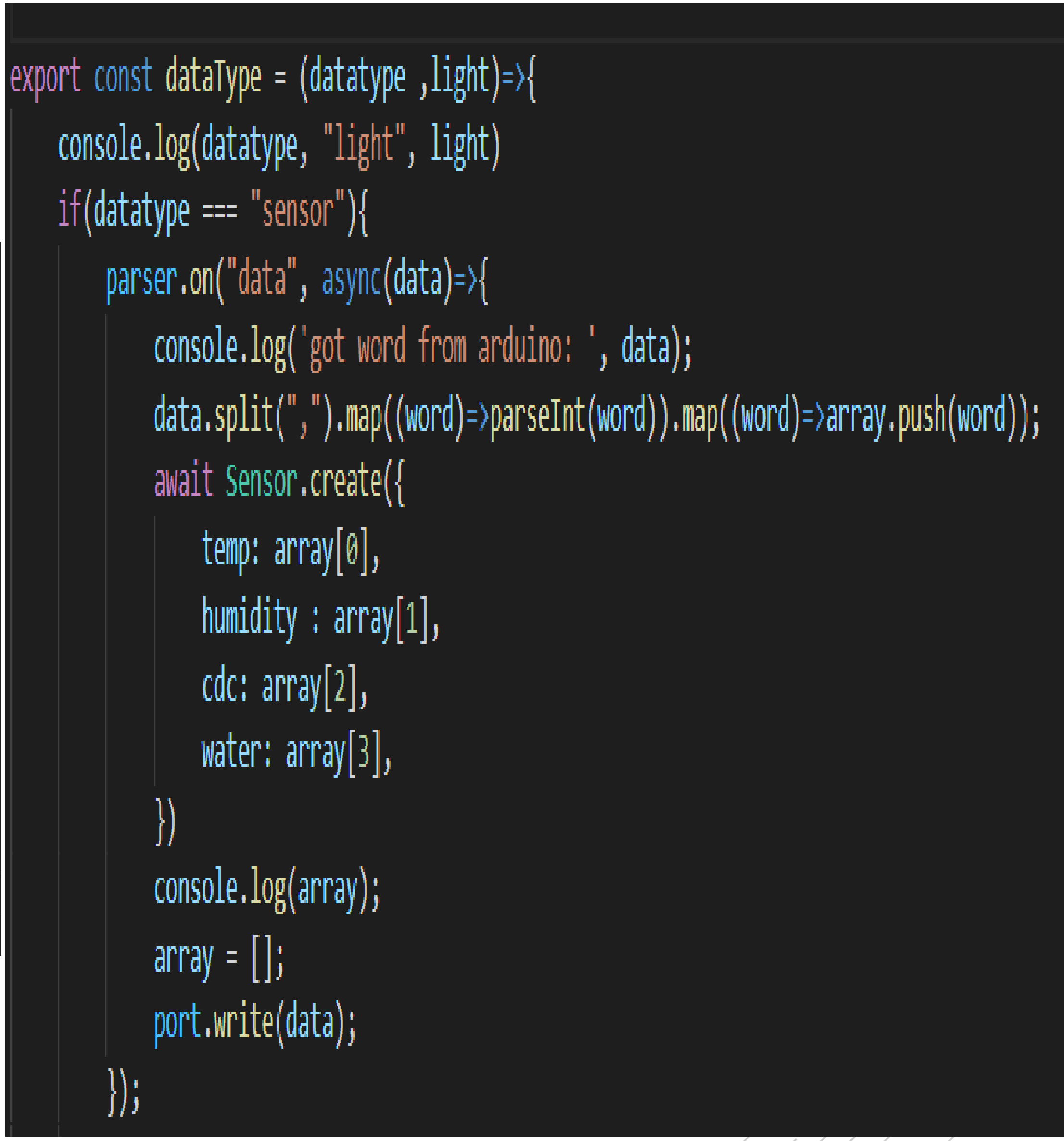
```
import Sensor from "./models/Sensor.js";
import SerialPort from "serialport";
import Readline from "@serialport/parser-readline";
// const SerialPort = require('serialport')
// const Readline = require('@serialport/parser-readline')
const port = new SerialPort('COM5',{
    baudRate:9600
});

const parser = port.pipe(new Readline({ delimiter: '\n' }));
let array = [];

port.on("open",()=>{
    console.log('serial open');
});
```

getSensor.js

센서 모델

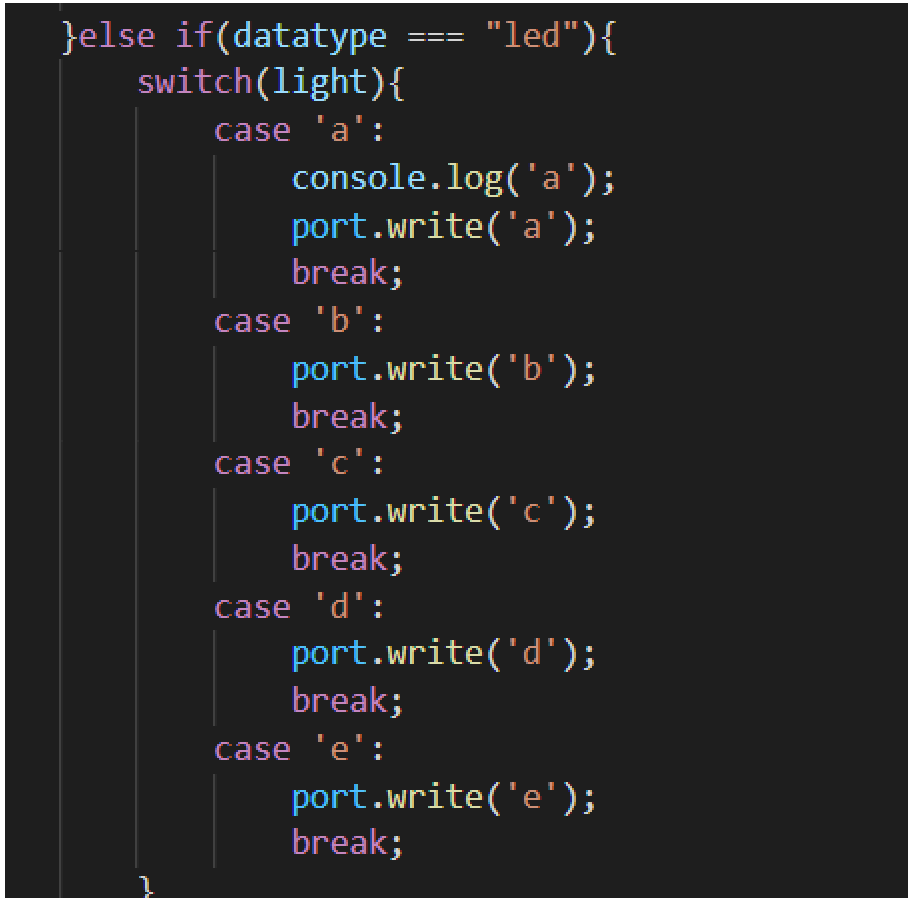


```
import mongoose from "mongoose"

const sensorSchema = new mongoose.Schema({
    createdAt : {type : Date, default: Date.now},
    temp : Number,
    humidity : Number,
    cdc: Number,
    water : Number,
});


const Sensor = mongoose.model("Sensor", sensorSchema);

export default Sensor
```



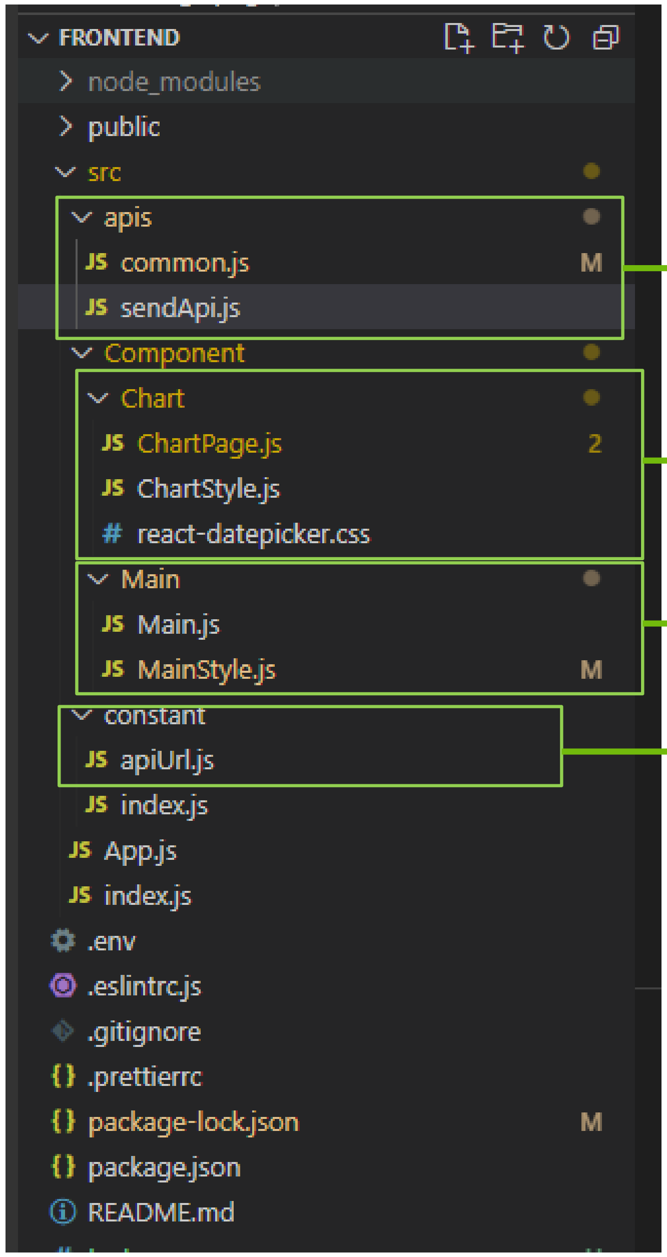
```
export const dataType = (datatype ,light)=>{
    console.log(datatype, "light", light)
    if(datatype === "sensor"){
        parser.on("data", async(data)=>{
            console.log('got word from arduino: ', data);
            data.split(",").map((word)=>parseInt(word)).map((word)=>array.push(word));
            await Sensor.create({
                temp: array[0],
                humidity : array[1],
                cdc: array[2],
                water: array[3],
            })
            console.log(array);
            array = [];
            port.write(data);
        });
```

getSensor.js



```
}else if(datatype === "led"){
    switch(light){
        case 'a':
            console.log('a');
            port.write('a');
            break;
        case 'b':
            port.write('b');
            break;
        case 'c':
            port.write('c');
            break;
        case 'd':
            port.write('d');
            break;
        case 'e':
            port.write('e');
            break;
    }
```

프론트 구조



```
FRONTEND
  > node_modules
  > public
  v src
    v apis
      JS common.js          M
      JS sendApi.js
    v Component
      v Chart
        JS ChartPage.js     2
        JS ChartStyle.js
        # react-datepicker.css
      v Main
        JS Main.js
        JS MainStyle.js     M
    v constant
      JS apiUrl.js
      JS index.js
    JS App.js
    JS index.js
  .env
  .eslintrc.js
  .gitignore
  {} .prettierrc
  {} package-lock.json      M
  {} package.json
  README.md
```

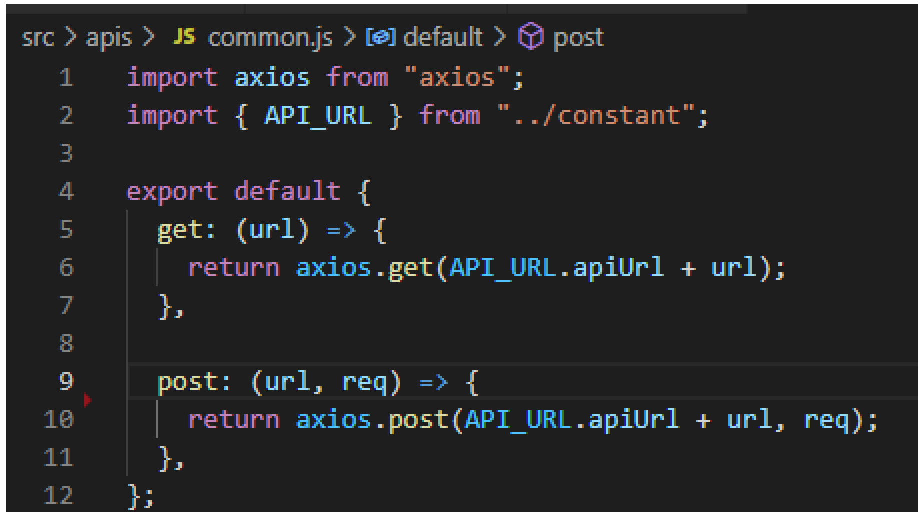
API 통신 처리

Chart 모달 창 component

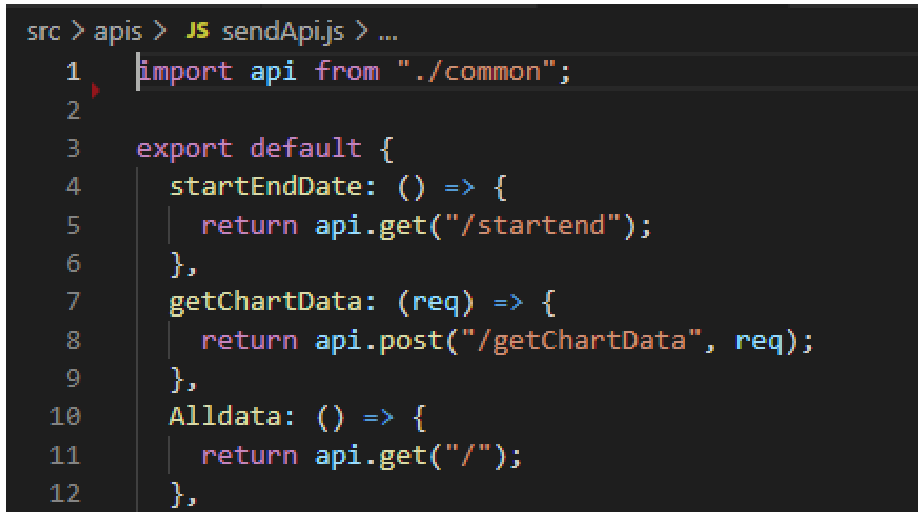
Main 화면 component

.env 파일로부터 백엔드
서버 URL 받아옴

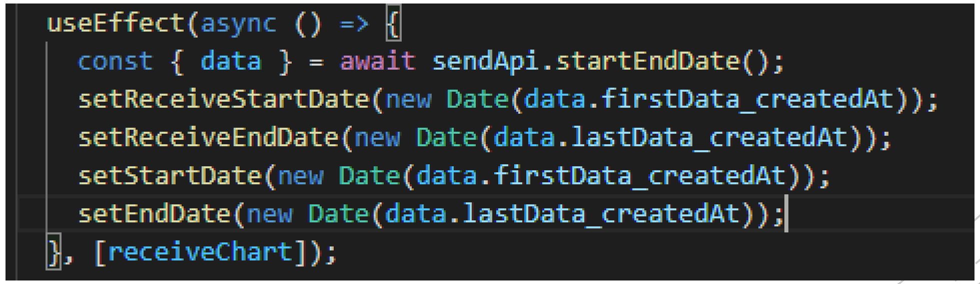
프론트 구조 통신 API



```
import axios from "axios";
import { API_URL } from "../constant";

export default {
  get: (url) => {
    return axios.get(API_URL.apiUrl + url);
  },

  post: (url, req) => {
    return axios.post(API_URL.apiUrl + url, req);
  },
};
```

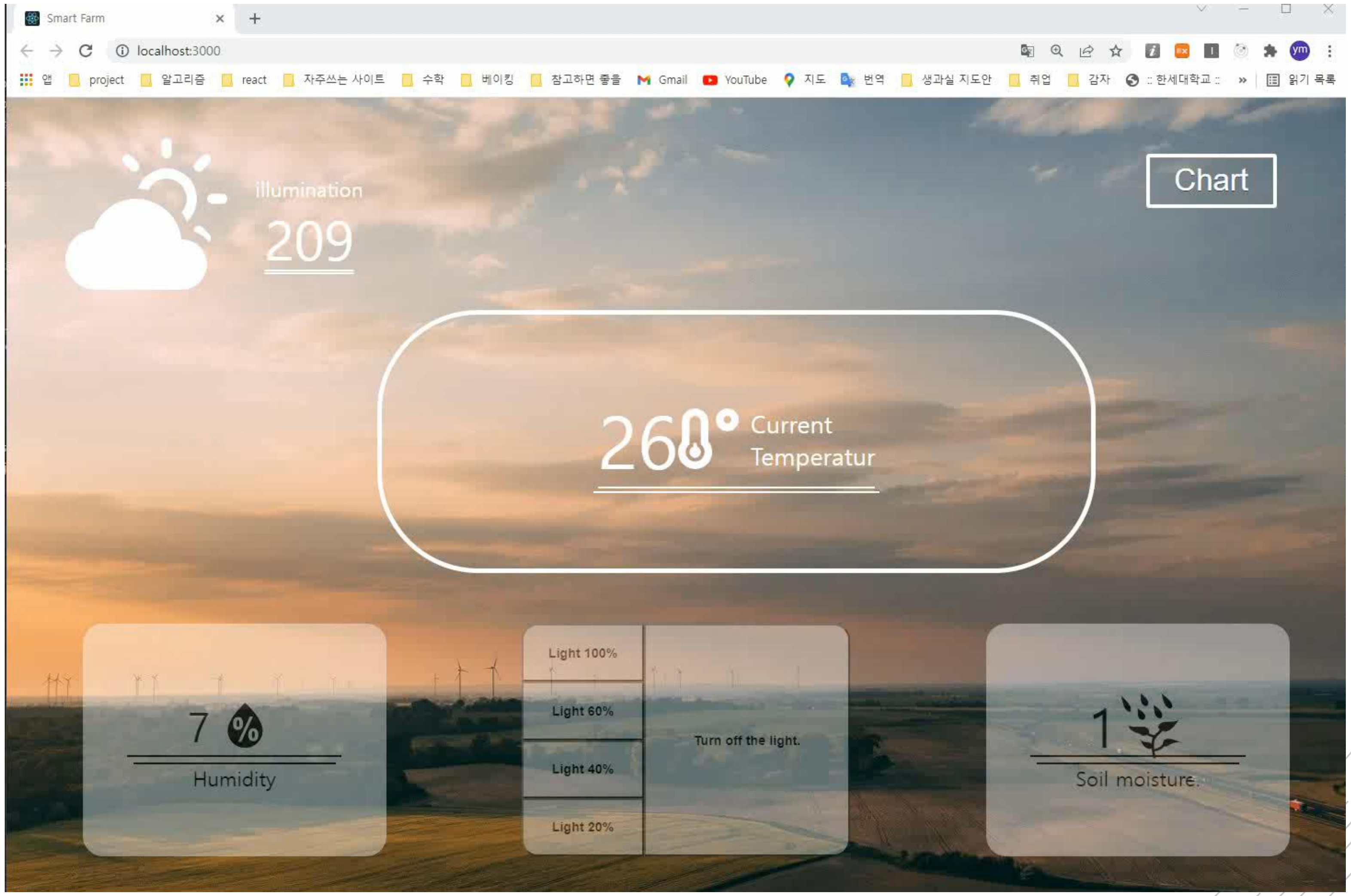


```
import api from "./common";

export default {
  startEndDate: () => {
    return api.get("/startend");
  },
  getChartData: (req) => {
    return api.post("/getChartData", req);
  },
  Alldata: () => {
    return api.get("/");
  },
```



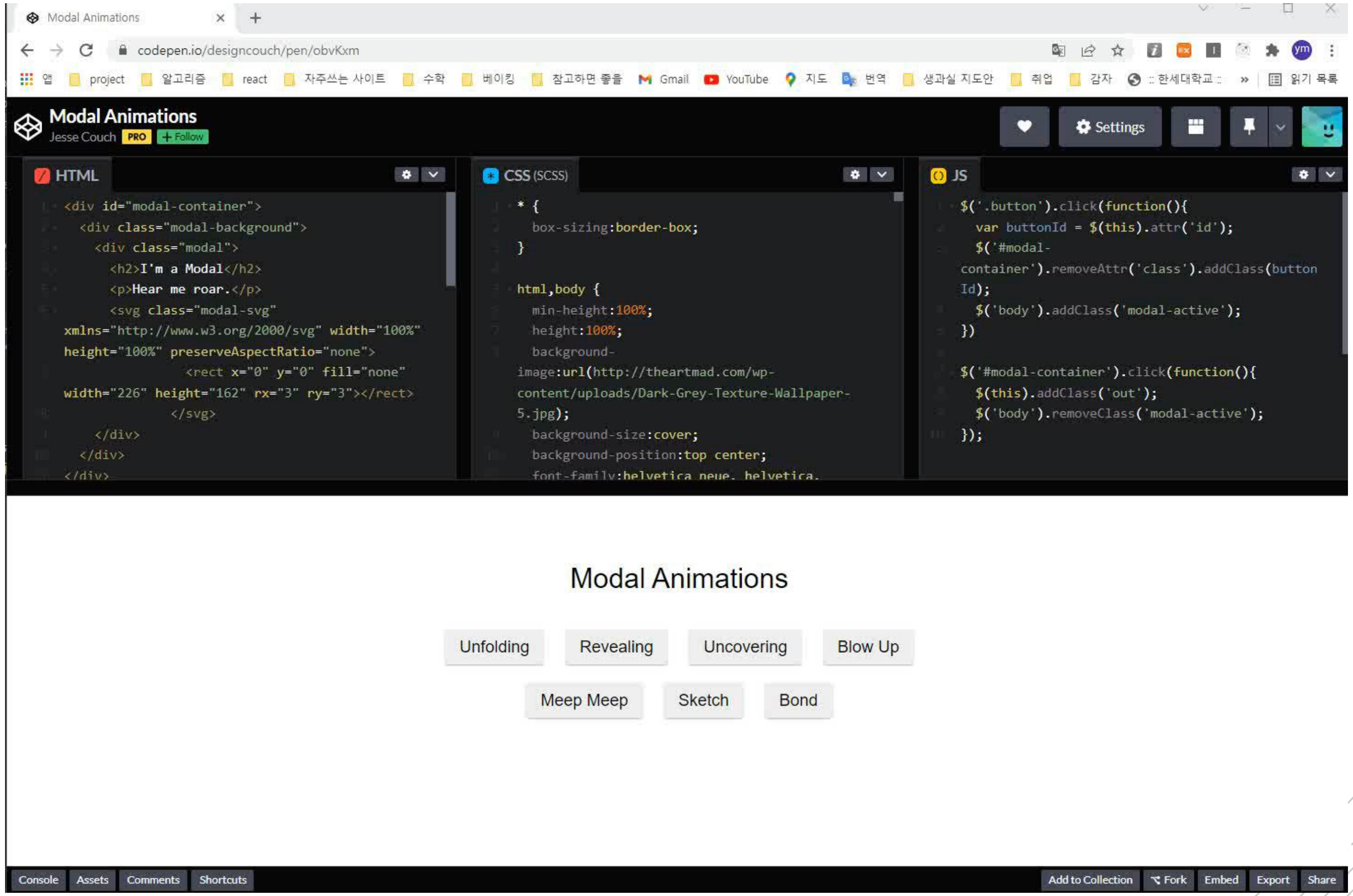
```
useEffect(async () => {
  const { data } = await sendApi.startEndDate();
  setReceiveStartDate(new Date(data.firstData_createdAt));
  setReceiveEndDate(new Date(data.lastData_createdAt));
  setStartDate(new Date(data.firstData_createdAt));
  setEndDate(new Date(data.lastData_createdAt));
}, [receiveChart]);
```

프론트 구조
Chart Page



프론트 구조 Chart Page

- https://codepen.io/designcouch/pen/obvKxm

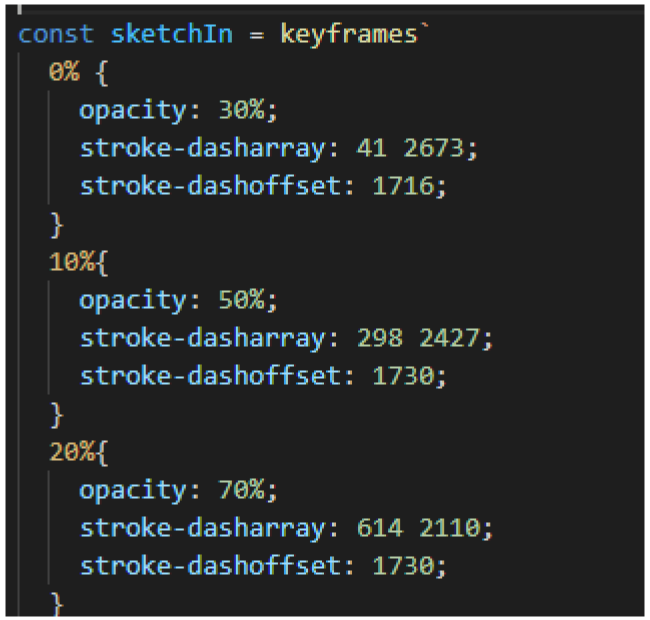


```html
<div id="modal-container">
  <div class="modal-background">
    <div class="modal">
      <h2>I'm a Modal</h2>
      <p>Hear me roar.</p>
      <svg class="modal-svg"
xmlns="http://www.w3.org/2000/svg" width="100%"
height="100%" preserveAspectRatio="none">
                <rect x="0" y="0" fill="none"
width="226" height="162" rx="3" ry="3"></rect>
              </svg>
    </div>
  </div>
</div>
```

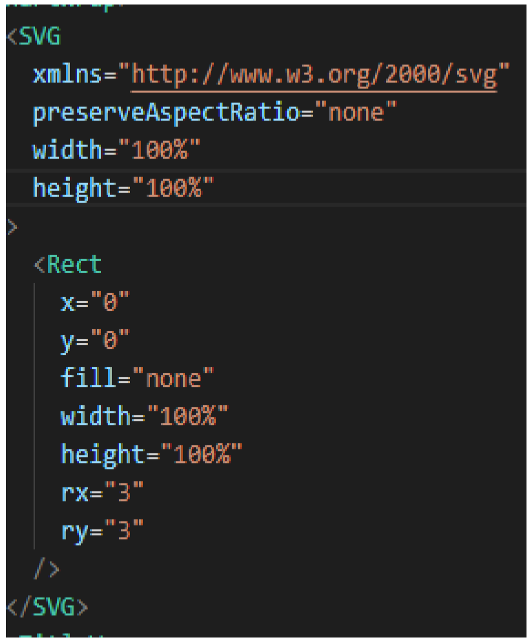
```scss
* {
  box-sizing:border-box;
}

html,body {
  min-height:100%;
  height:100%;
  background-
image:url(http://theartmad.com/wp-
content/uploads/Dark-Grey-Texture-Wallpaper-
5.jpg);
  background-size:cover;
  background-position:top center;
  font-family:helvetica neue, helvetica,
```

```js
$('.button').click(function(){
  var buttonId = $(this).attr('id');
  $('#modal-
container').removeAttr('class').addClass(button
Id);
  $('body').addClass('modal-active');
})

$('#modal-container').click(function(){
  $(this).addClass('out');
  $('body').removeClass('modal-active');
});
```

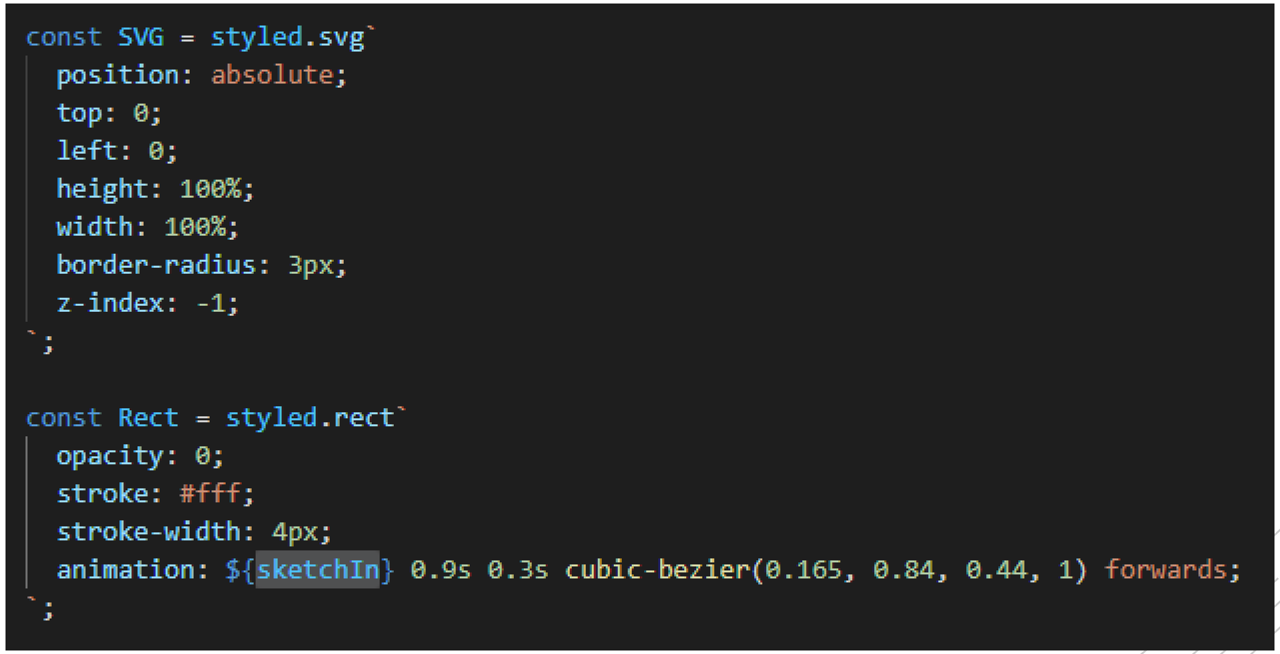
프론트 구조
Chart Page



```
const sketchIn = keyframes`
  0% {
    opacity: 30%;
    stroke-dasharray: 41 2673;
    stroke-dashoffset: 1716;
  }
  10%{
    opacity: 50%;
    stroke-dasharray: 298 2427;
    stroke-dashoffset: 1730;
  }
  20%{
    opacity: 70%;
    stroke-dasharray: 614 2110;
    stroke-dashoffset: 1730;
  }
```



```
<SVG
  xmlns="http://www.w3.org/2000/svg"
  preserveAspectRatio="none"
  width="100%"
  height="100%"
>
  <Rect
    x="0"
    y="0"
    fill="none"
    width="100%"
    height="100%"
    rx="3"
    ry="3"
  />
</SVG>
```

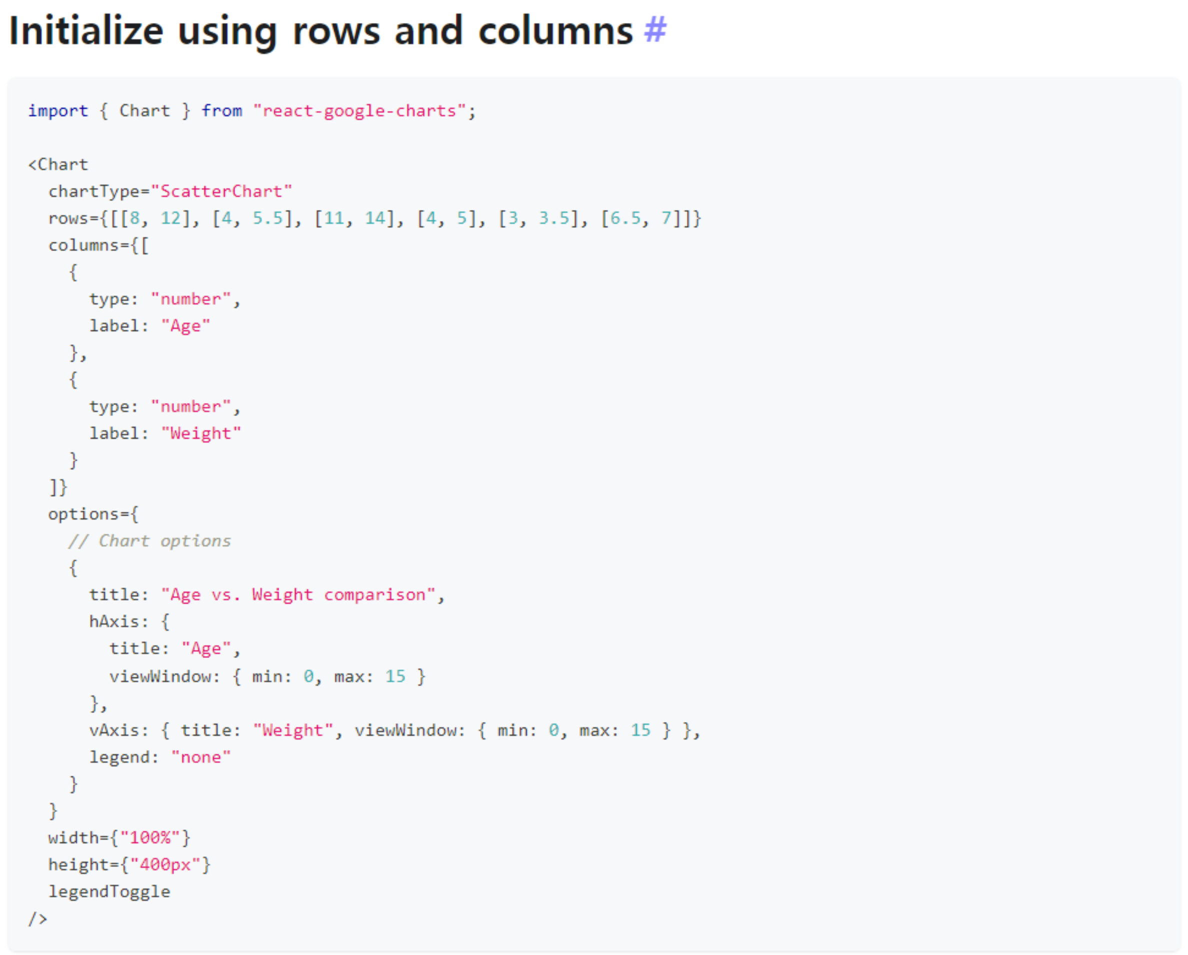


```
const SVG = styled.svg`
  position: absolute;
  top: 0;
  left: 0;
  height: 100%;
  width: 100%;
  border-radius: 3px;
  z-index: -1;
`;

const Rect = styled.rect`
  opacity: 0;
  stroke: #fff;
  stroke-width: 4px;
  animation: ${sketchIn} 0.9s 0.3s cubic-bezier(0.165, 0.84, 0.44, 1) forwards;
`;
```

프론트 구조
Chart Page

Google Chart

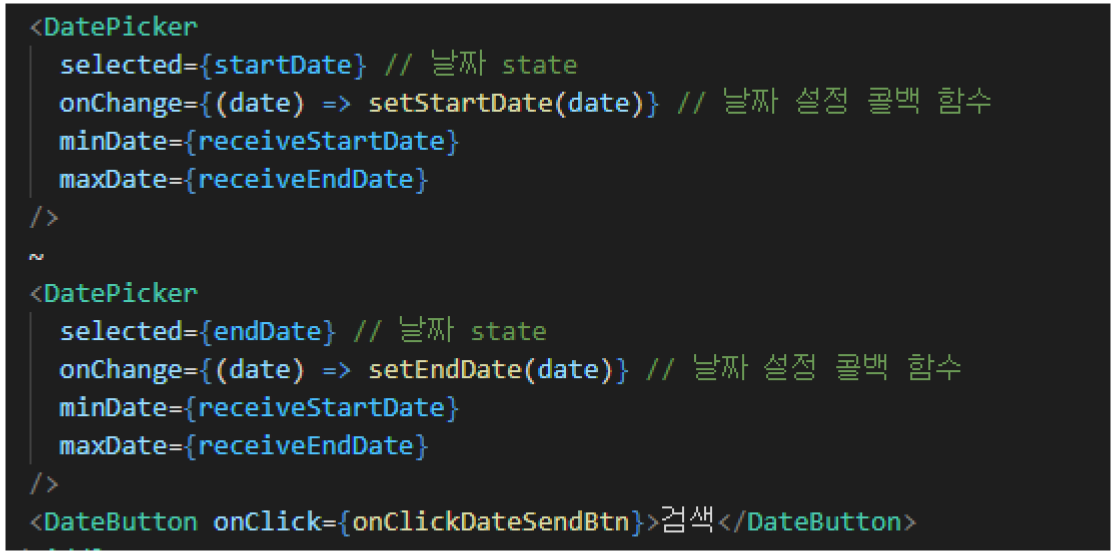
- https://developers.google.com/chart/interactive/docs/gallery/linechart?hl=ko
- https://www.react-google-charts.com/



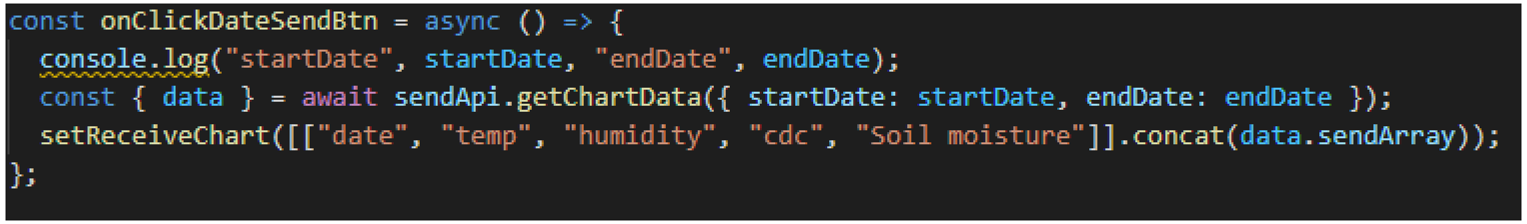
Initialize using rows and columns

```
import { Chart } from "react-google-charts";

<Chart
  chartType="ScatterChart"
  rows={[[8, 12], [4, 5.5], [11, 14], [4, 5], [3, 3.5], [6.5, 7]]}
  columns={[
    {
      type: "number",
      label: "Age"
    },
    {
      type: "number",
      label: "Weight"
    }
  ]}
  options={
    // Chart options
    {
      title: "Age vs. Weight comparison",
      hAxis: {
        title: "Age",
        viewWindow: { min: 0, max: 15 }
      },
      vAxis: { title: "Weight", viewWindow: { min: 0, max: 15 } },
      legend: "none"
    }
  }
  width={"100%"}
  height={"400px"}
  legendToggle
/>
```

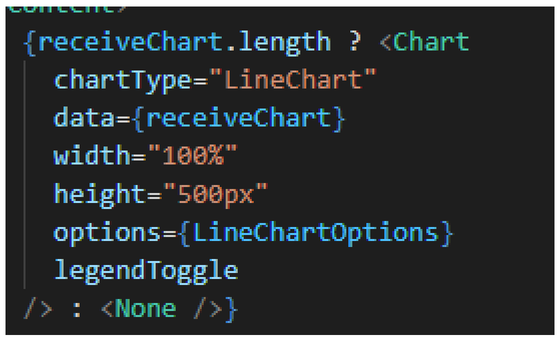
프론트 구조
Chart Page



```
<DatePicker
  selected={startDate} // 날짜 state
  onChange={(date) => setStartDate(date)} // 날짜 설정 콜백 함수
  minDate={receiveStartDate}
  maxDate={receiveEndDate}
/>
~
<DatePicker
  selected={endDate} // 날짜 state
  onChange={(date) => setEndDate(date)} // 날짜 설정 콜백 함수
  minDate={receiveStartDate}
  maxDate={receiveEndDate}
/>
<DateButton onClick={onClickDateSendBtn}>검색</DateButton>
```

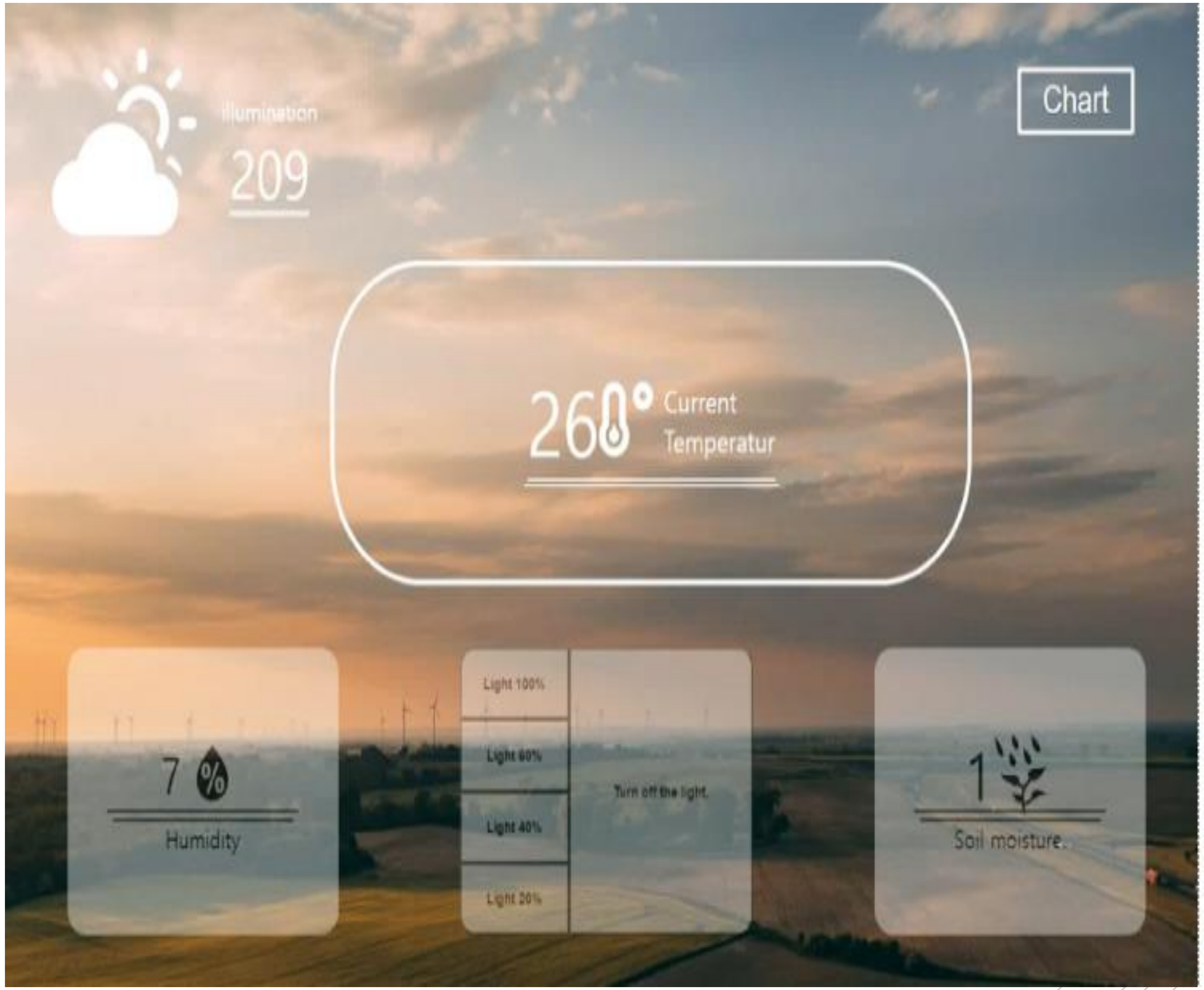


```
const onClickDateSendBtn = async () => {
  console.log("startDate", startDate, "endDate", endDate);
  const { data } = await sendApi.getChartData({ startDate: startDate, endDate: endDate });
  setReceiveChart([["date", "temp", "humidity", "cdc", "Soil moisture"]].concat(data.sendArray));
};
```

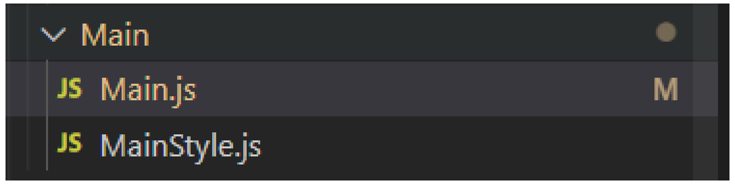


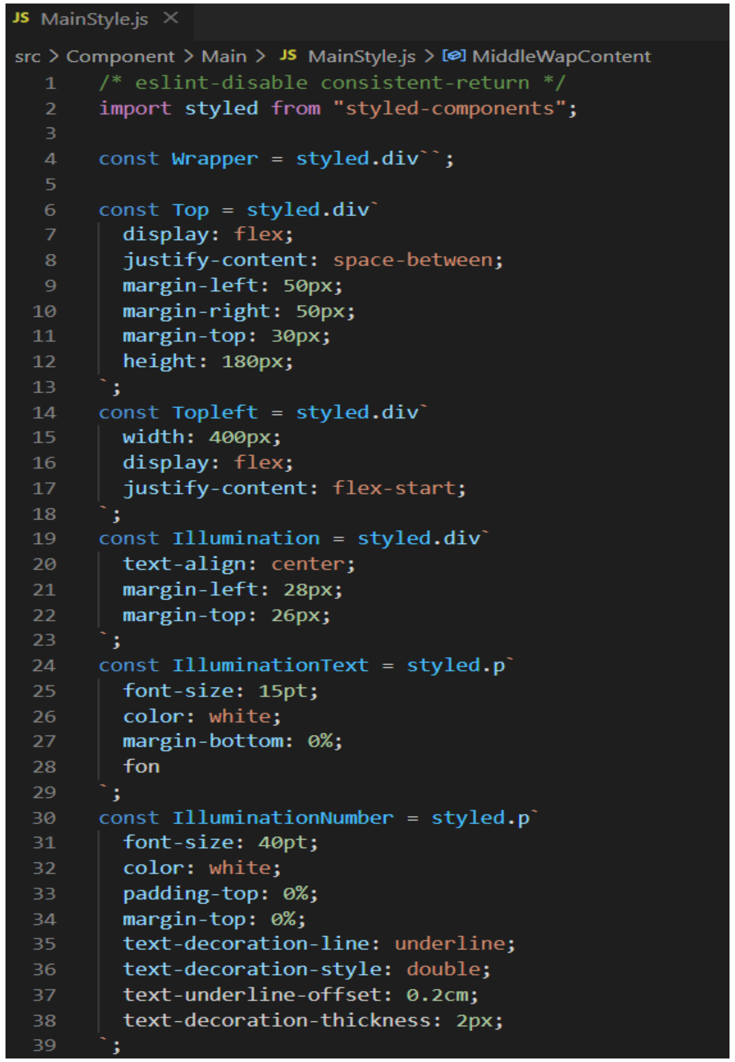
```
{receiveChart.length ? <Chart
  chartType="LineChart"
  data={receiveChart}
  width="100%"
  height="500px"
  options={LineChartOptions}
  legendToggle
/> : <None />}
```

프론트 구조
Main Page

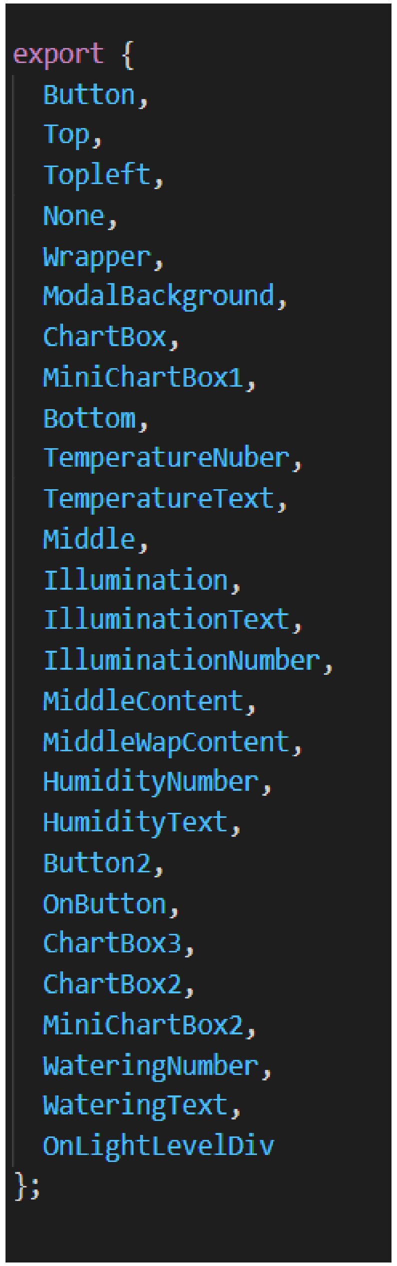


프론트 구조 Main Page

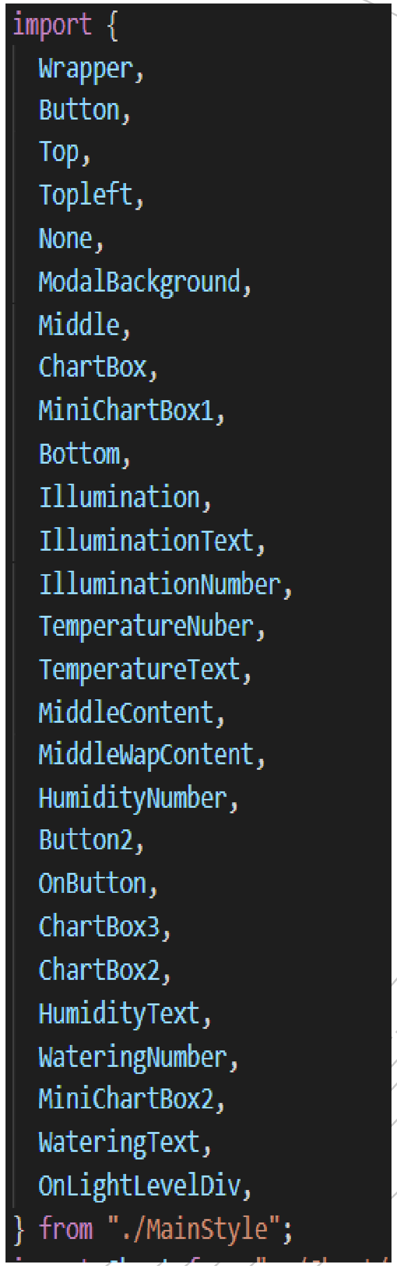




```
/* eslint-disable consistent-return */
import styled from "styled-components";

const Wrapper = styled.div``;

const Top = styled.div`
  display: flex;
  justify-content: space-between;
  margin-left: 50px;
  margin-right: 50px;
  margin-top: 30px;
  height: 180px;
`;
const Topleft = styled.div`
  width: 400px;
  display: flex;
  justify-content: flex-start;
`;
const Illumination = styled.div`
  text-align: center;
  margin-left: 28px;
  margin-top: 26px;
`;
const IlluminationText = styled.p`
  font-size: 15pt;
  color: white;
  margin-bottom: 0%;
  fon
`;
const IlluminationNumber = styled.p`
  font-size: 40pt;
  color: white;
  padding-top: 0%;
  margin-top: 0%;
  text-decoration-line: underline;
  text-decoration-style: double;
  text-underline-offset: 0.2cm;
  text-decoration-thickness: 2px;
`;
```

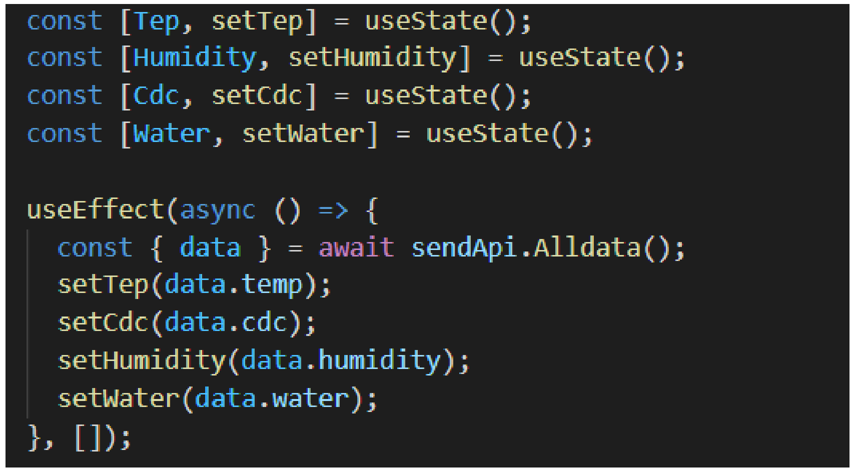


```
export {
  Button,
  Top,
  Topleft,
  None,
  Wrapper,
  ModalBackground,
  ChartBox,
  MiniChartBox1,
  Bottom,
  TemperatureNuber,
  TemperatureText,
  Middle,
  Illumination,
  IlluminationText,
  IlluminationNumber,
  MiddleContent,
  MiddleWapContent,
  HumidityNumber,
  HumidityText,
  Button2,
  OnButton,
  ChartBox3,
  ChartBox2,
  MiniChartBox2,
  WateringNumber,
  WateringText,
  OnLightLevelDiv
};
```

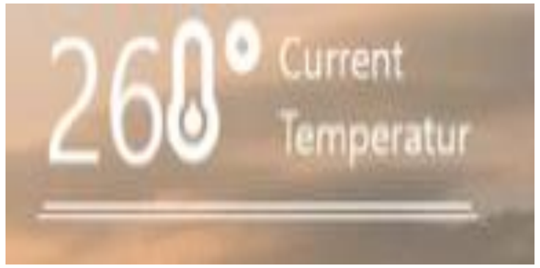


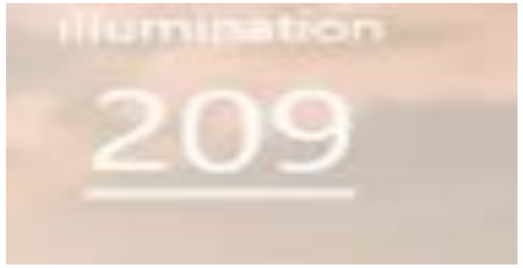
```
import {
  Wrapper,
  Button,
  Top,
  Topleft,
  None,
  ModalBackground,
  Middle,
  ChartBox,
  MiniChartBox1,
  Bottom,
  Illumination,
  IlluminationText,
  IlluminationNumber,
  TemperatureNuber,
  TemperatureText,
  MiddleContent,
  MiddleWapContent,
  HumidityNumber,
  Button2,
  OnButton,
  ChartBox3,
  ChartBox2,
  HumidityText,
  WateringNumber,
  MiniChartBox2,
  WateringText,
  OnLightLevelDiv,
} from "./MainStyle";
```

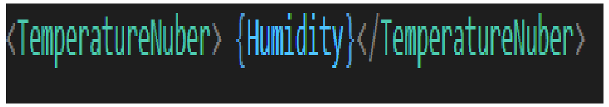
프론트 구조
Main Page



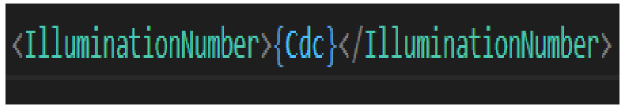
```
const [Tep, setTep] = useState();
const [Humidity, setHumidity] = useState();
const [Cdc, setCdc] = useState();
const [Water, setWater] = useState();

useEffect(async () => {
  const { data } = await sendApi.Alldata();
  setTep(data.temp);
  setCdc(data.cdc);
  setHumidity(data.humidity);
  setWater(data.water);
}, []);
```



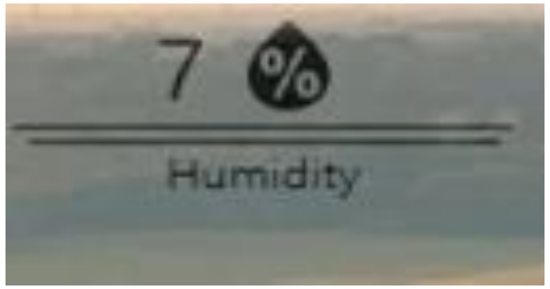


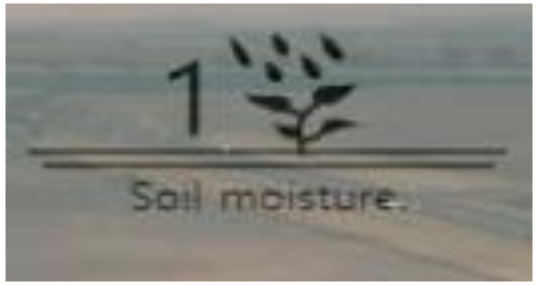


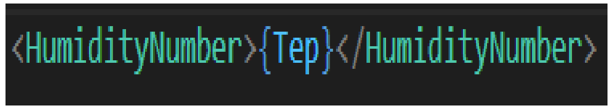
```
<TemperatureNuber> {Humidity}</TemperatureNuber>
```



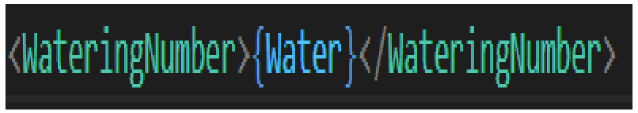
```
<IlluminationNumber>{Cdc}</IlluminationNumber>
```





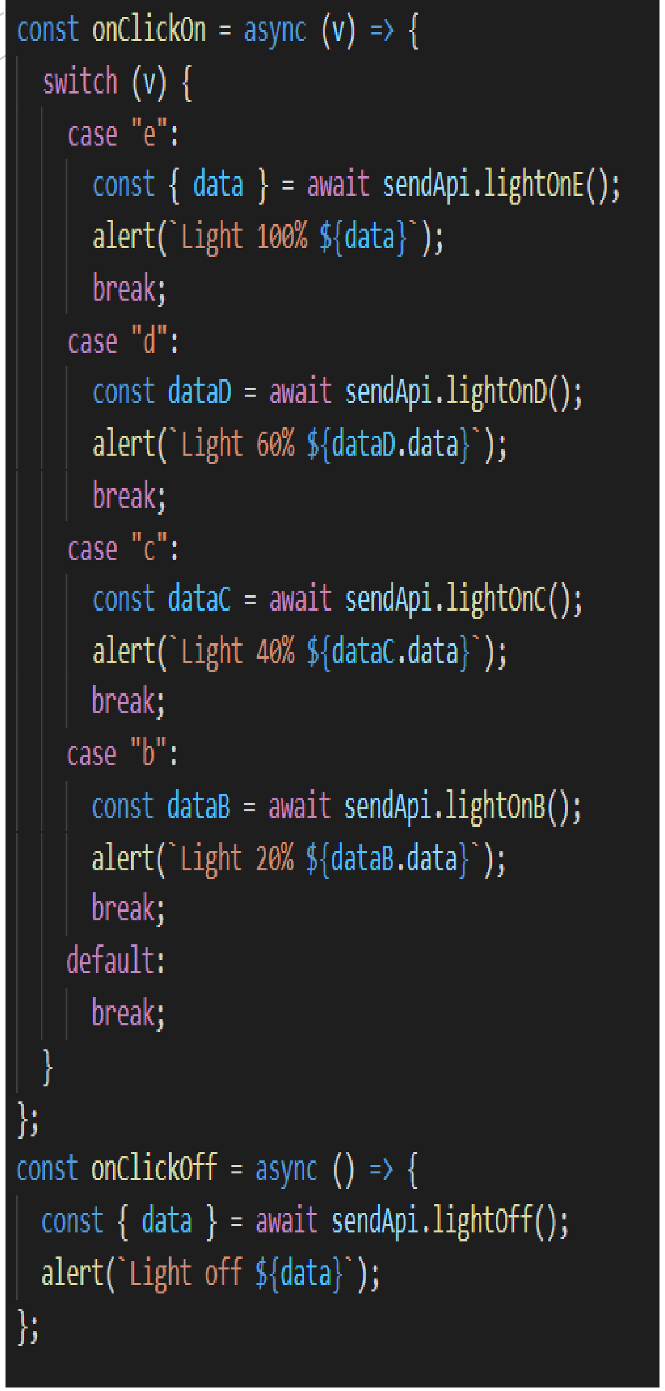


```
<HumidityNumber>{Tep}</HumidityNumber>
```

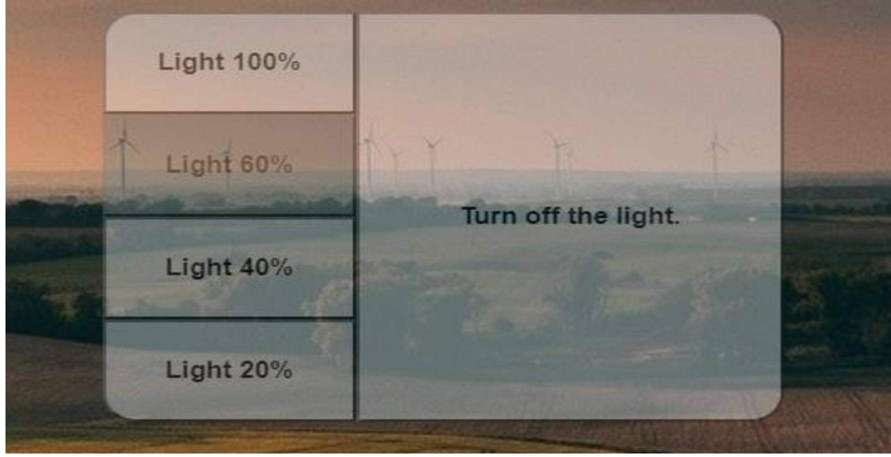


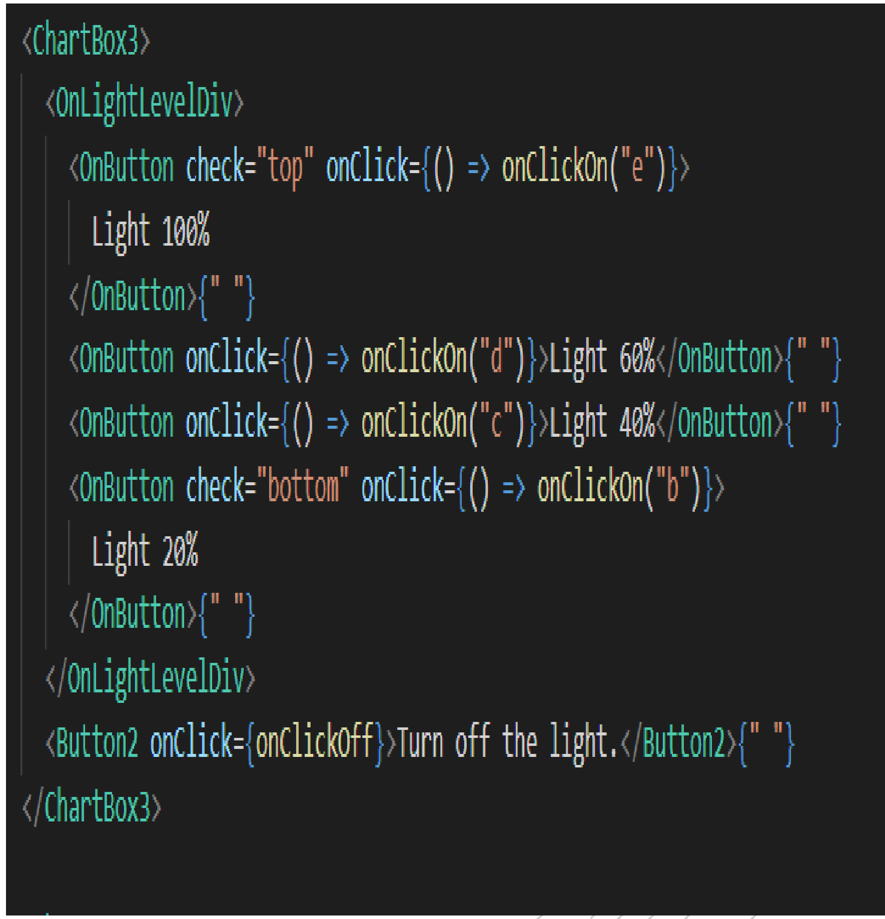
```
<WateringNumber>{Water}</WateringNumber>
```

프론트 구조
Main Page



```
const onClickOn = async (v) => {
  switch (v) {
    case "e":
      const { data } = await sendApi.lightOnE();
      alert(`Light 100% ${data}`);
      break;
    case "d":
      const dataD = await sendApi.lightOnD();
      alert(`Light 60% ${dataD.data}`);
      break;
    case "c":
      const dataC = await sendApi.lightOnC();
      alert(`Light 40% ${dataC.data}`);
      break;
    case "b":
      const dataB = await sendApi.lightOnB();
      alert(`Light 20% ${dataB.data}`);
      break;
    default:
      break;
  }
};
const onClickOff = async () => {
  const { data } = await sendApi.lightoff();
  alert(`Light off ${data}`);
};
```

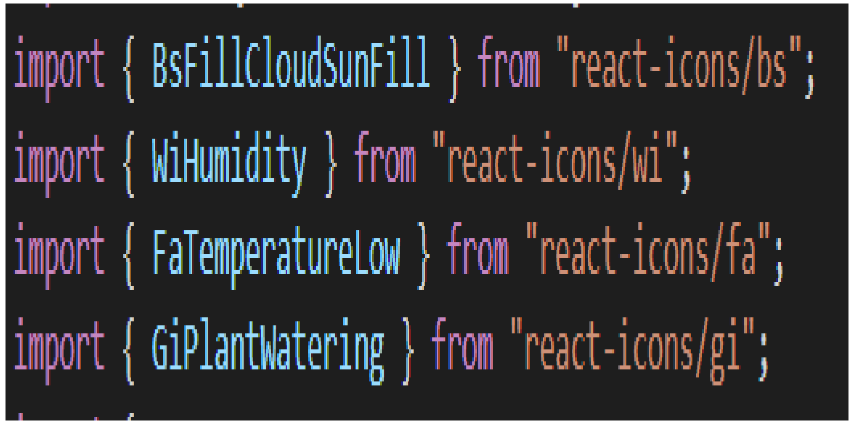




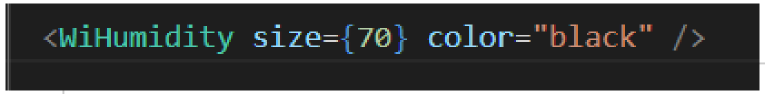
```
<ChartBox3>
  <OnLightLevelDiv>
    <OnButton check="top" onClick={() => onClickOn("e")}>
      Light 100%
    </OnButton>{" "}
    <OnButton onClick={() => onClickOn("d")}>Light 60%</OnButton>{" "}
    <OnButton onClick={() => onClickOn("c")}>Light 40%</OnButton>{" "}
    <OnButton check="bottom" onClick={() => onClickOn("b")}>
      Light 20%
    </OnButton>{" "}
  </OnLightLevelDiv>
  <Button2 onClick={onClickOff}>Turn off the light.</Button2>{" "}
</ChartBox3>
```

프론트 구조
Main Page

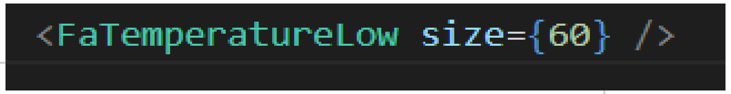
- https://react-icons.github.io/react-icons/



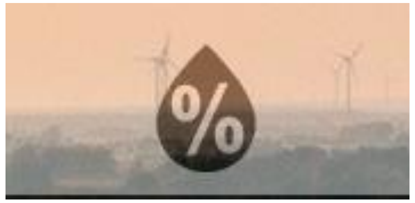
```
import { BsFillCloudSunFill } from "react-icons/bs";
import { WiHumidity } from "react-icons/wi";
import { FaTemperatureLow } from "react-icons/fa";
import { GiPlantWatering } from "react-icons/gi";
```

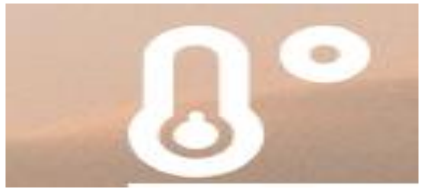


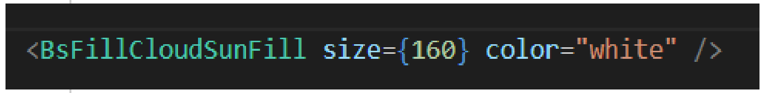
```
<WiHumidity size={70} color="black" />
```



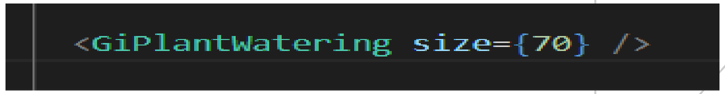
```
<FaTemperatureLow size={60} />
```



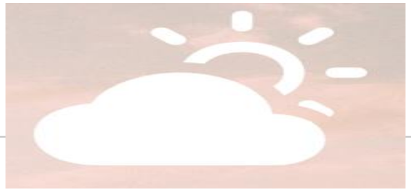


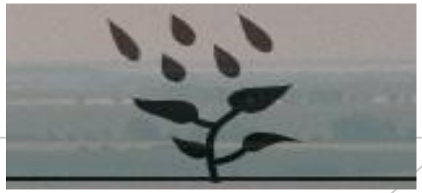


```
<BsFillCloudSunFill size={160} color="white" />
```



```
<GiPlantWatering size={70} />
```





시연

보완할 점 및 더 구현하고 싶은 내용

보완할 점

- Frontend

- 좀 더 인터랙티브한 web
- Chart 구현시 좀 더 깔끔한 디자인

-Backend

- LED의 늦은 반응 해결
- 레거시한 코드 수정

더 구현하고 싶은 내용

- 카메라를 통한 실시간 스마트 팜 관찰

느낀 점

사용한 자료 기록