



라인컴퓨터아트학원

안산시 도시환경 계측관리 대시보드

장승우 배석찬 이소윤 이정민 전영진 최종윤

Table of Contents

01	Overview	04 - 05
02	Schedule	07
03	Design	09 - 15
04	Code	17 - 32
05	Review	34 - 39

01

Overview

안산시의 **도시환경 데이터**를 효과적으로
모니터링하고 관리하기 위해 **대시보드**를
구축합니다.



대시보드란?

업무에 필요한 각종 정보를 도표나 그래프 등으로
한 눈에 파악할 수 있도록 디자인 된 페이지입니다.

장점

데이터의 추세와 패턴을 빠르게 파악할 수 있습니다.

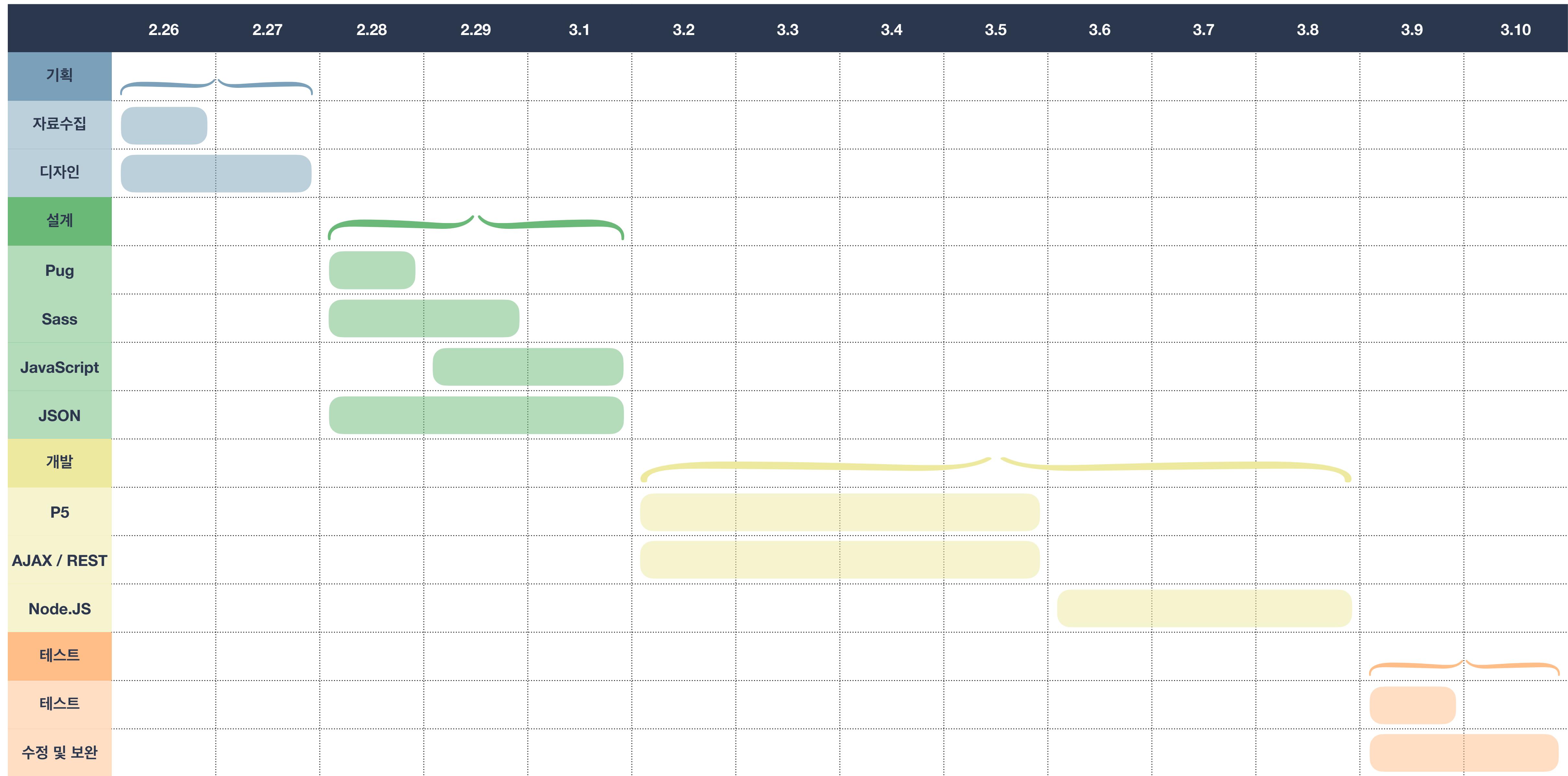
실시간으로 도시환경 데이터를 모니터링 하여 문제가
발생했을 때 빠르게 대응이 가능합니다.

대시보드를 통해 수집된 데이터를 분석하고 시각화하여,
정책 수립 및 의사결정에 도움이 됩니다.



02

Schedule



03

Design



“간결한 디자인”

대시보드의 핵심 내용을 집중을 위한 불필요한 요소를 제거하여 시각적 혼란을 최소화합니다.

“일관성 있는 구획”

각 요소들이 일관되고 조화롭게 배치되어 있어 사용자 맞춤형 정보를 제공합니다.



744px



393px

“반응형 디자인”

다양한 디바이스에서도 일관된 사용 경험을 제공하기 위해 반응형으로 제작하였습니다.

Desktop, Tablet, Mobile 기기 모두 깔끔하고 잘 정리된 레이아웃을 제공합니다.

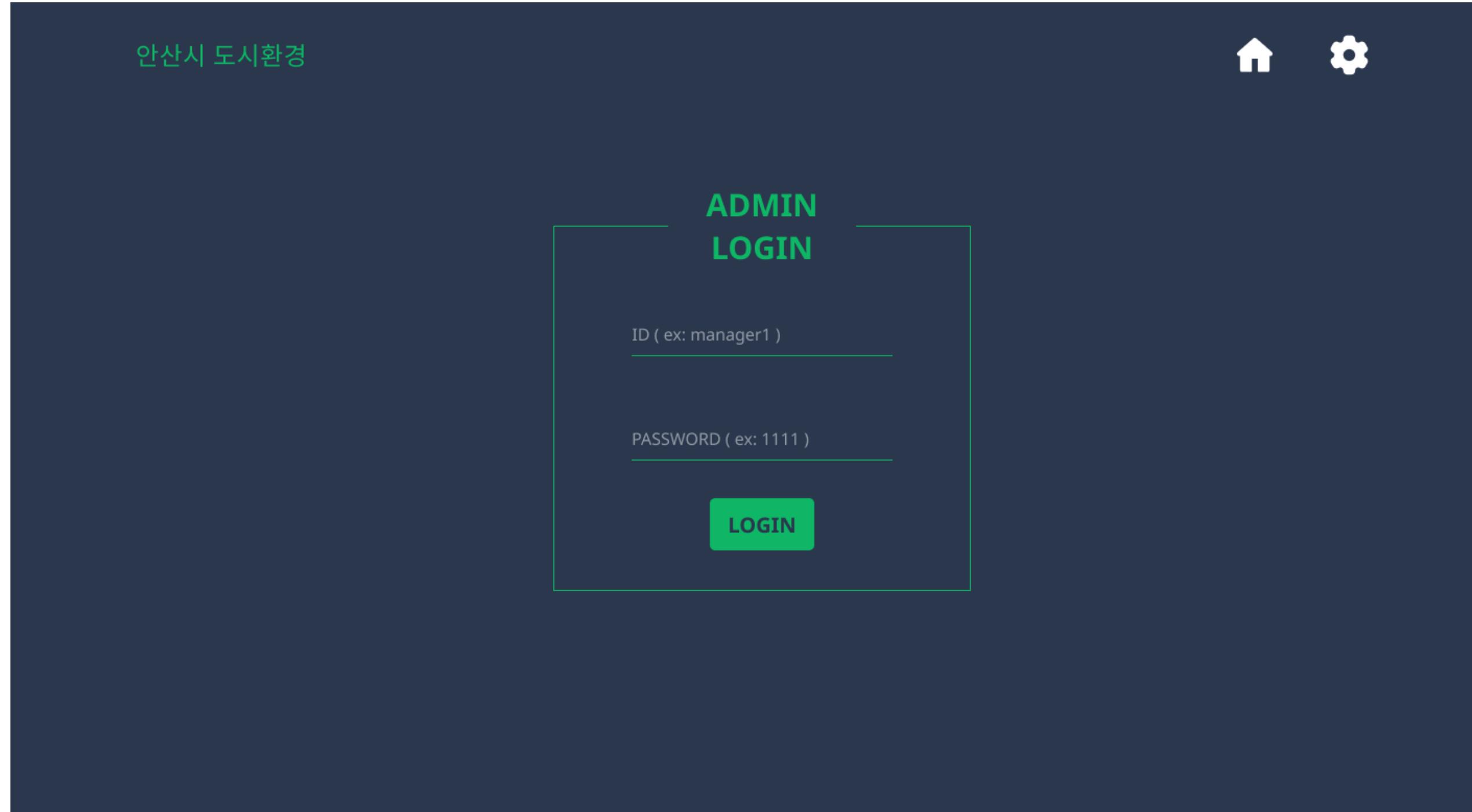


The screenshot shows a dark-themed dashboard for environmental monitoring. At the top left is the title '안산시 도시환경'. In the top center is the date and time '01-01 12:30:00' with a refresh icon. On the top right is a home icon. Below the header are four buttons: '온도 습도 미세먼지' (highlighted in white), '소음', '쓰레기', and '교통'. A '수정' (Edit) button is located at the bottom right of the header area. The main content area displays a table with six rows of data. The columns are labeled '시간' (Time), '온도(°C)' (Temperature), '습도(%)' (Humidity), and '미세먼지(PM10)' (PM10). The data shows constant values across all rows: 30:00, -10, 2, 10; 30:01, -10, 2, 10; 30:02, -10, 2, 10; 30:03, -10, 2, 10; 30:04, -10, 2, 10; 30:05, -10, 2, 10.

시간	온도(°C)	습도(%)	미세먼지(PM10)
30:00	-10	2	10
30:01	-10	2	10
30:02	-10	2	10
30:03	-10	2	10
30:04	-10	2	10
30:05	-10	2	10

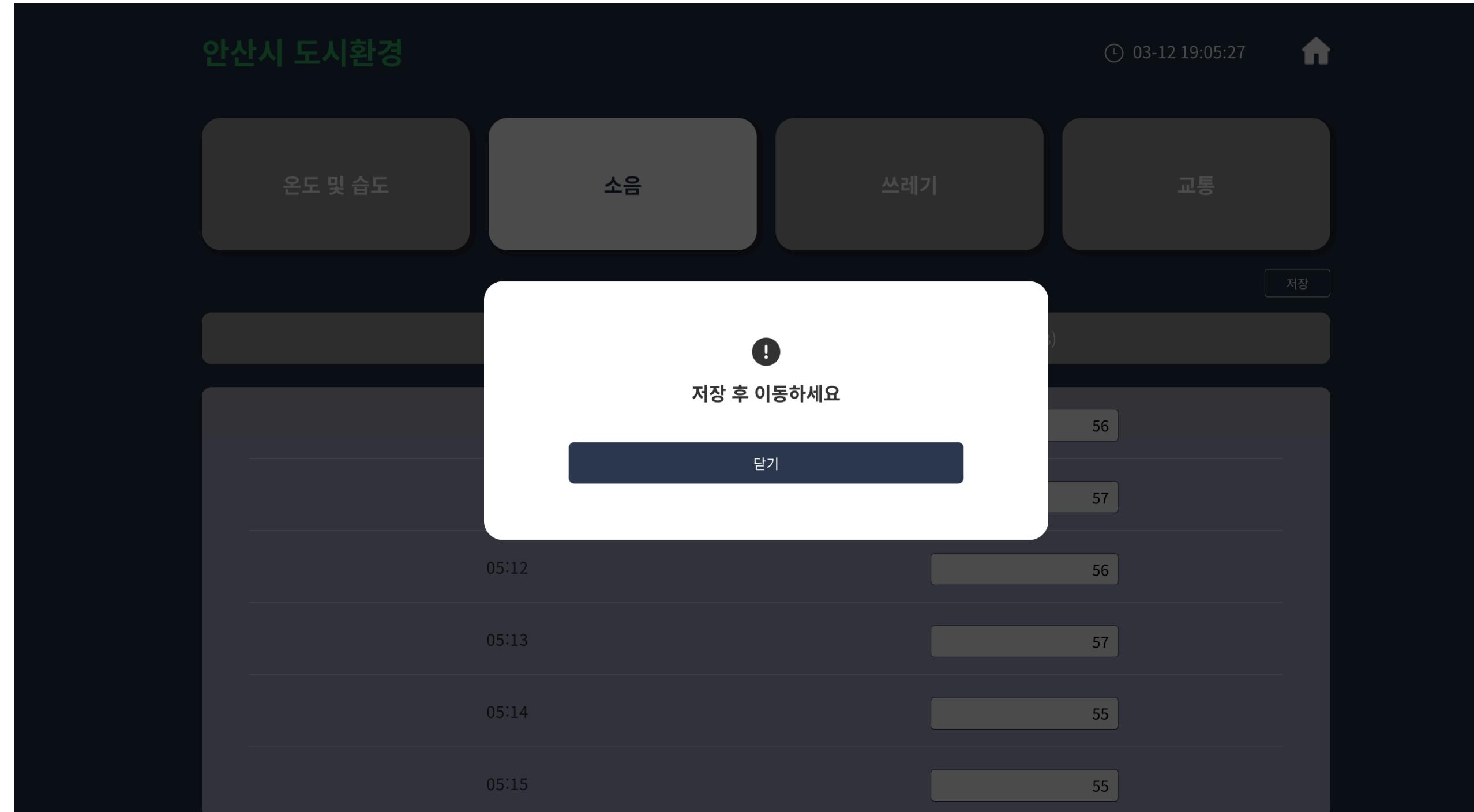
“직관적인 인터페이스”

관리자가 데이터를 쉽게 관리할 수 있게
불필요한 요소 및 디자인적 장식을 최소화하며,
페이지 레이아웃을 명확하게 구조화합니다.



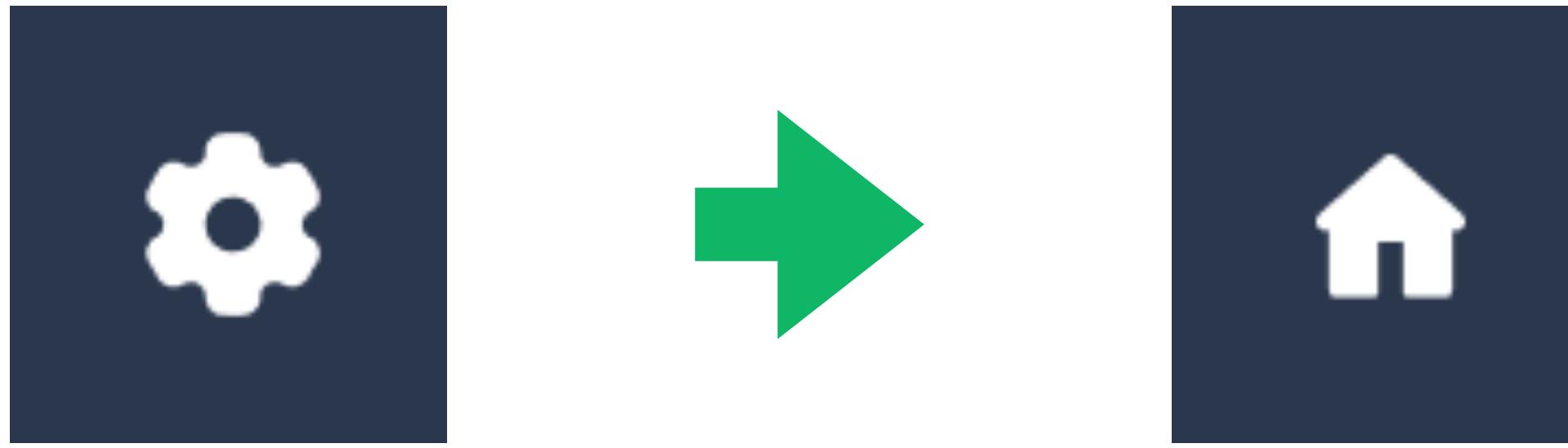
“일관된 디자인 요소”

디자인 요소들이 일관성 있게 배치되어 있어 사용자가 쉽게 로그인 폼과 관련된 요소들을 찾을 수 있도록 배치합니다.



“시각적인 안내와 오류 처리”

사용자가 정보를 잘못 입력했을 때
적절한 안내와 오류 메시지를 표시하여
사용자가 실수를 수정하고 다시
시도할 수 있도록 요청합니다.



“일관된 UI/UX”

대시보드 페이지와 관리자 페이지 간의 일관된 디자인과 사용자 경험을 제공하여 사용자들이 각 페이지를 이동하더라도 혼란을 최소화합니다.

Font

Noto Sans

읽기 쉽고 명확하여 가독성을 높이고
제공되는 정보를 편리하게 이해할 수 있습니다.

Color

Main Color

차분한 색상으로 대시보드
전반적으로 안정적인
느낌을 제공합니다.



#2A374D

Font Color

배경과 대조되어
가독성이 향상됩니다.



#FFFFFF

Point Color1

사용자의 주의를 집중시키고
중요한 정보를 강조합니다.



#0EB665

Point Color2

Point Color1과 대조를 활용하여
데이터를 분류하고 명확한 정보를
전달합니다.

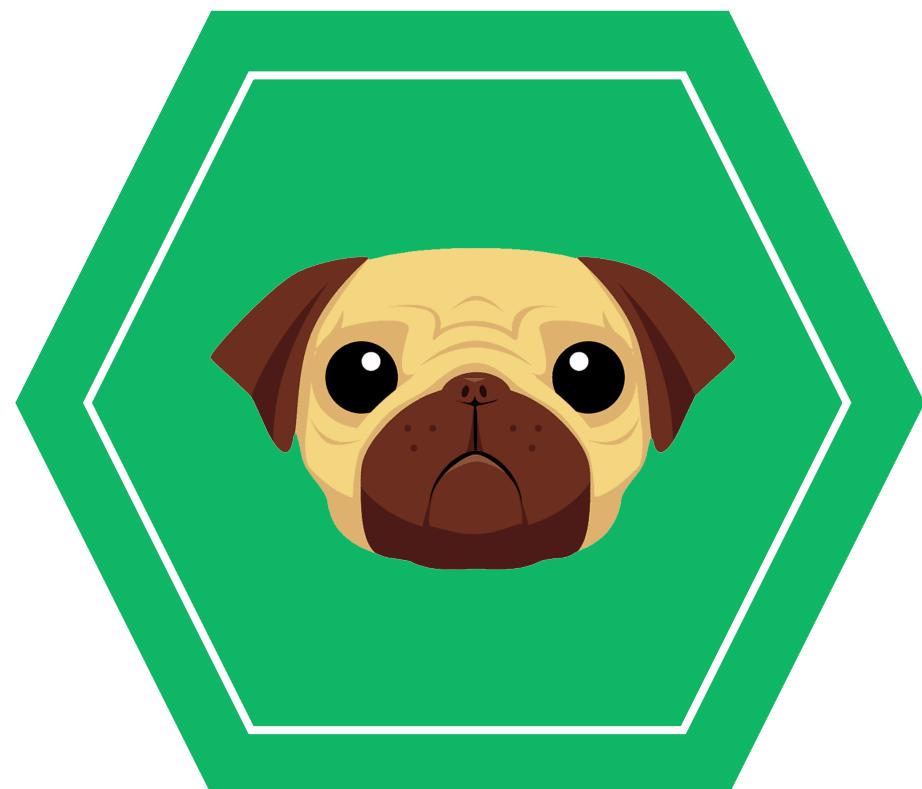


#DC8677

04

Code

Skills



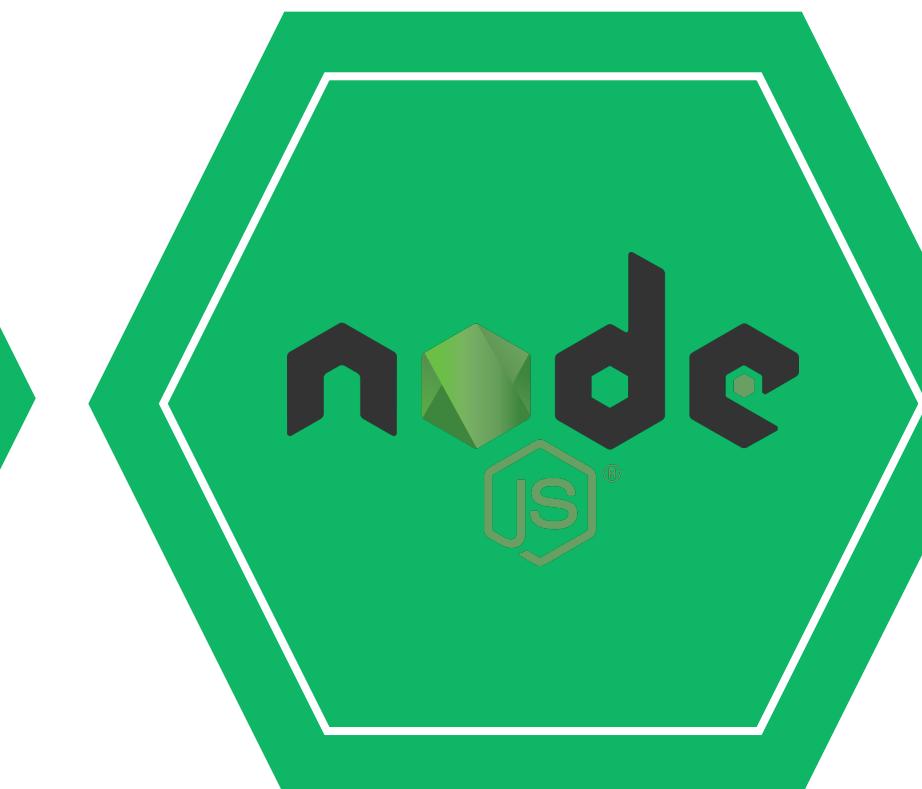
Pug



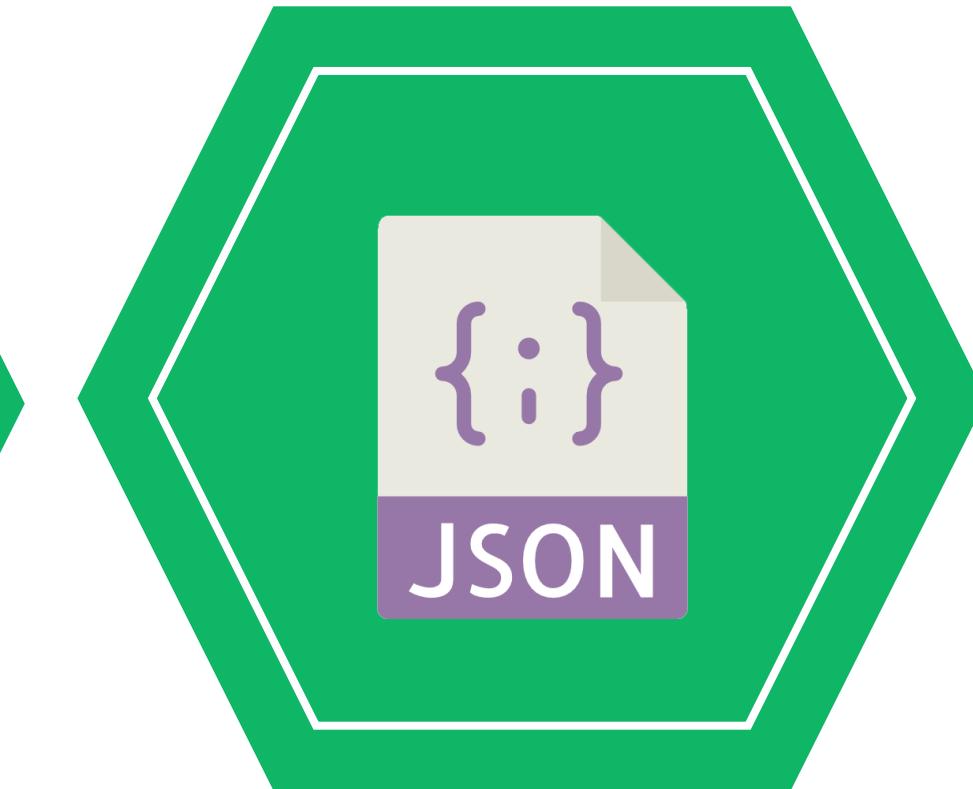
Sass



JavaScript



Node.js



JSON



그래프 확대 함수

그래프 각각의 요소를 객체로 받아와 사용자가 클릭한 그래프를 확대하여 표시하고 축소하는 기능을 제공합니다.

```

const expandCanvas = () => {
  if (!isExpanded) {
    graphs.forEach(graph => {
      document.getElementById(graph.id).onclick = () => {
        if (parseInt(window.getComputedStyle(document.querySelector('.wrapper')).getPropertyValue('width')) > 900) {
          document.getElementById("expandContent").style.display = "block";
          isExpanded = true;
          const multipleCanvas = (multiple) => { // 선택한 그래프 width, height 의 1.5배 만큼 확대
            document.getElementById("expandCanvas").style.width = (parseInt(window.getComputedStyle(document.getElementById(graph.id)).getPropertyValue('width')) * multiple) + "px";
            document.getElementById("expandCanvas").style.height = (parseInt(window.getComputedStyle(document.getElementById(graph.id)).getPropertyValue('height')) * multiple) + "px";
            document.getElementById("expandCanvas").style.padding = (parseInt(window.getComputedStyle(document.getElementById(graph.id)).getPropertyValue('padding')) * multiple) + "px";
          }
          multipleCanvas(1.5);
          document.getElementById("expandCanvas").classList.add(graph.class);
          document.getElementById("expandCanvas").innerHTML += `<h2>온도·습도</h2><p class = "graph_txt">시간(분:초)</p>`;
          new p5(graph.sketch, "expandCanvas");
        }
      }
    })
  }
}

```





```
printTime() {  
    document.getElementById("presentTime").innerHTML = `  
        ${this.month}-${this.date} ${this.hours}:${this.minutes}:${this.seconds}`  
};  
  
timeInfo() {  
    const time = new Date();  
    this.month = String((time.getMonth() + 1) < 10 ? "0" + (time.getMonth() + 1) : (time.getMonth() + 1));  
    this.date = String(time.getDate() < 10 ? "0" + (time.getDate()) : (time.getDate()));  
    this.hours = String(time.getHours() < 10 ? "0" + (time.getHours()) : (time.getHours()));  
    this.minutes = String(time.getMinutes() < 10 ? "0" + (time.getMinutes()) : (time.getMinutes()));  
    this.seconds = String(time.getSeconds() < 10 ? "0" + (time.getSeconds()) : (time.getSeconds()));  
};  
  
updateTime() {  
    this.printTime();  
    this.timeInfo();  
    setTimeout(() => {  
        this.updateTime();  
    }, 990);  
};
```

실시간 시간 구현 클래스

이 코드는 시계 클래스를 정의하고,
현재 시간을 HTML 요소에 업데이트하는
기능을 제공합니다.

 03-19 09:38:45





레스트용 객체 데이터 메소드

Express.js를 사용하여 여러 디렉토리에 있는 정적 파일을 제공하고, 각 파일의 확장자에 따라 적절한 Content-Type을 설정합니다.

```
allData(type) { // 레스트용 객체 데이터 메소드()
  if (type === 'weatherData') {
    let inputValue = [];
    let dataInput = document.querySelectorAll('.dataP > input');
    dataInput.forEach(v => inputValue.push(v.value.replace(/\s/g, '')));
    let itor = inputValue[Symbol.iterator]();
    let ~~~~
    let weatherData = { ... }
    return weatherData
  } else if (type === 'noiseLevel') {
    let inputValue = [];
    let dataInput = document.querySelectorAll('.dataP > input');
    dataInput.forEach(v => inputValue.push(v.value.replace(/\s/g, '')));
    let itor = inputValue[Symbol.iterator]();
    let noiseLevel = { ... }
    return noiseLevel
  } else if (type === 'wasteType') {
    let inputValue = [];
    let dataInput = document.querySelectorAll('.dataP > input');
    dataInput.forEach(v => inputValue.push(v.value.replace(/\s/g, '')));
    let itor = inputValue[Symbol.iterator]();
    let wasteType = { ... }
    return wasteType
  } else if (type === 'trafficVolume') {
    let inputValue = [];
    let dataInput = document.querySelectorAll('.dataP > input');
    dataInput.forEach(v => inputValue.push(v.value.replace(/\s/g, '')));
    let itor = inputValue[Symbol.iterator]();
    let trafficVolume = { ... }
    return trafficVolume
  }
}
```



```

let intervalId;
app.get('/', (req, res) => {
  res.render('index'); // admin.pug 파일을 렌더링하여 클라이언트에게 전송
  clearInterval(intervalId)
  intervalId = setTimeout(interval, 1000);
});

function interval() {
  const A_Data = db.get("Ansan").value();

  // weatherData 데이터 처리
  A_Data[0].weatherData = processData(A_Data[0].weatherData, 0);

  // noiseLevel 데이터 처리
  A_Data[1].noiseLevel = processData(A_Data[1].noiseLevel, 1);

  // wasteType 데이터 처리
  A_Data[2].wasteType = processData(A_Data[2].wasteType, 2);

  // trafficVolume 데이터 처리
  A_Data[3].trafficVolume = processData(A_Data[3].trafficVolume, 3);
}

// 파일에 다시 쓰기
db.unset("Ansan").write();
db.set("Ansan", A_Data).write();
intervalId = setTimeout(interval, 1000);
}

// 서버가 종료되면 타임아웃을 취소합니다.
process.on('SIGINT', () => {
  clearTimeout(intervalId);
  process.exit();
});

```

실시간 데이터 업데이트 메소드

각 데이터 유형에 대해 처리 함수를 호출하고, 처리된 데이터를 다시 파일에 씁니다.
또한, 서버가 종료될 때 타임아웃을 취소하여 정리 작업을 수행합니다.

시간	온도(°C)	습도(%)	미세먼지(PM10)	풍향	풍속
39:16	4	4	2	3	5
39:17	4	4	2	3	5
39:18	4	4	2	3	5
39:19	4	4	2	3	5
39:20	3	50	149	50	5
39:21	3	49	149	63	5



```
jsonData() { // JSON 파일 로드
  const xhr = new XMLHttpRequest();
  xhr.onreadystatechange = () => {
    if (xhr.readyState === XMLHttpRequest.DONE) {
      if (xhr.status === 200) {
        const jsonData = JSON.parse(xhr.responseText);
        this.resetArr();
        this.dataType(jsonData);
        this.windDirection();
        this.windSpeed();
      } else {
        console.log('에러');
      }
    }
  }
  xhr.open('GET', this.dataLink, true);
  xhr.send();
}
```

JSON 파일 로드 메소드

이 함수는 XMLHttpRequest를 사용하여 JSON 파일을 로드합니다. 파일이 성공적으로 로드되면 JSON 데이터를 파싱하고 다양한 데이터 처리 메소드를 호출합니다. 그러나 파일 로드에 실패하면 에러 메시지를 출력합니다.



```

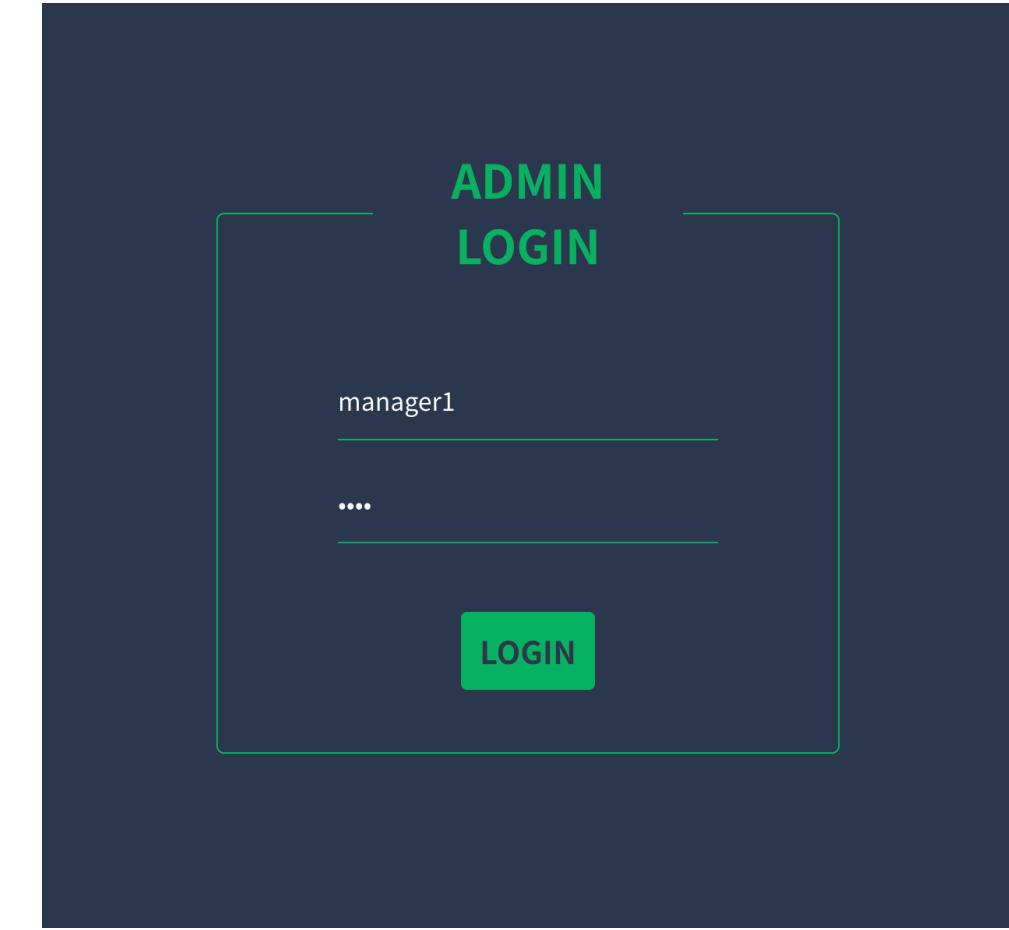
app.get('/login', (req, res) => {
  res.render('login'); // admin.pug 파일을 렌더링하여 클라이언트에게 전송
});

// index, login 페이지 추가 0304

// admin.pug를 렌더링하여 클라이언트에게 전송
const myAuth = new Auth('myAuth'); //Auth 클래스의 인스턴스 생성
app.get('/admin', myAuth.checkAuth, (req, res) => {
  res.render('admin'); // admin.pug 파일을 렌더링하여 클라이언트에게 전송
  clearInterval(intervalId)
});

app.post('/login', function (req, res) {
  const {
    username,
    password
  } = req.body;
  // console.log("Received login request:", username, password);
  // 사용자 정보 검색
  const user = l_Data.loginData.find(item => item.userName === username && item.userPassword === password);
  // console.log("Found user:", user);
  if (user) {
    // 로그인 성공
    req.session.user = user;
    res.status(200).redirect('/admin');
  } else {
    // 로그인 실패
    res.status(401).send("로그인 실패: 잘못된 사용자 이름 또는 비밀번호");
  }
});

```



Login 처리 메소드

사용자가 제공한 사용자 이름과 비밀번호를 검사하여, 해당 정보가 로그인 데이터베이스에 있는지 확인 후 로그인을 처리 합니다.

```
// 폼 데이터 가져오기
const formData = new FormData(form);

// AJAX 요청 보내기
var xhr = new XMLHttpRequest();
xhr.open('POST', '/login', true);
xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
xhr.onreadystatechange = function () {
  if (xhr.readyState === XMLHttpRequest.DONE) {

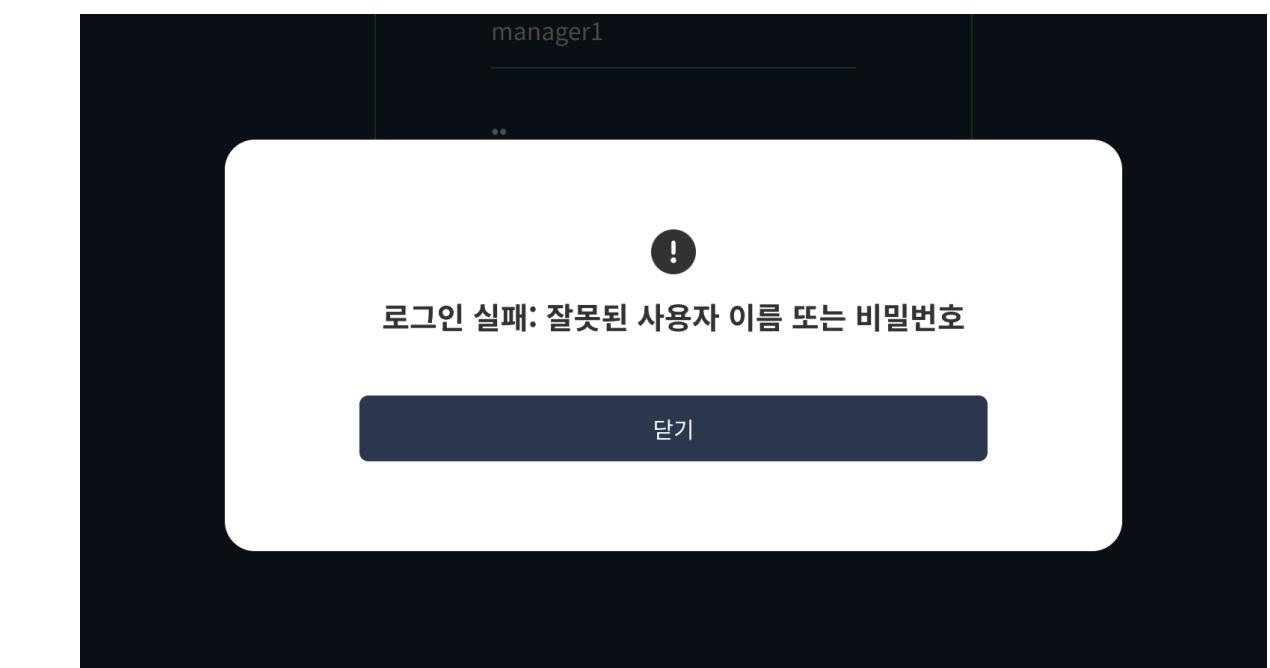
    // 응답 처리
    if (xhr.status === 200) {
      // 로그인 성공
      // 이동하거나, 추가 작업을 수행할 수 있음
      window.location.href = '/admin';
    } else {
      // 로그인 실패
      popupWrap.style.display = 'block';
      popup.style.display = 'block';
      popupText.innerText = xhr.responseText; // 서버 응답을 popupText에 표시
    }
  }
};

xhr.send(new URLSearchParams(formData).toString());
});
```

Login 처리 메소드

폼 데이터를 FormData 객체를 사용하여 수집하고,
이를 URL 인코딩하여 서버로 보냅니다.

AJAX 요청에 대한 응답을 받으면, 응답을 처리합니다.
응답이 성공하면 관리자 페이지로 이동하고, 실패하면
오류 메시지를 팝업으로 표시합니다.





레스트 업데이트 메소드

데이터를 JSON 문자열로 변환하여 서버로 전송하며, 요청의 상태를 확인하여 응답을 처리합니다.

```
modifyData(id, data) { // 레스트 업데이트 메소드()
    const dataObj = JSON.stringify(data); // 객체를 JSON 문자열로 변환

    const xhr = new XMLHttpRequest();
    xhr.open('PUT', `${this.dataLink}/${id}`);
    xhr.setRequestHeader('Content-Type', 'application/json'); // JSON 데이터를 전송하기 위해 헤더 설정
    xhr.send(dataObj);

    xhr.onreadystatechange = function (e) {
        if (xhr.readyState !== XMLHttpRequest.DONE) return;
        if (xhr.status === 200) {
            console.log("저장완료!");
        } else {
            console.log("Error!");
        }
    }
}
```



```
parseData(data, dataType, dataArr, dataName) { // 데이터 파싱 매소드()
    let someData = [];
    for (let i = 0; i < data.length; i++) {
        for (let key in data[i][dataType]) {
            let time = data[i][dataType][key]["시간"];
            someData.push({
                "시간": time,
                dataName: data[i][dataType][key]["data"][dataName]
            });
        }
        dataArr.push(someData);
        // console.log(dataArr); // 파싱 결과 출력
    }
}
```

데이터 파싱 매소드

주어진 데이터를 파싱하여
지정된 형식으로 변환하는 매소드입니다.



```
// 데이터 처리 함수
function processData(data, dataIndex) {
  const shiftData = data.shift();
  // 받은 데이터에서 앞부분 끊어서 저장한다.
  let newId = Number(shiftData.id) + 6;
  // 새로운 데이터는 앞 부분에서 끊 id에서 +6 --> 우리 데이터가 6개씩임
  // 시간계산 timestamp로 함
  let newTime = new Date();
  let myMinute = String(newTime.getMinutes());
  let mySecond = String(newTime.getSeconds());
  // 현재 시간 받아와서 분이랑 초를 설정함

  if (myMinute.length < 2) {
    myMinute = "0" + myMinute;
  }
  if (mySecond.length < 2) {
    mySecond = "0" + mySecond;
  }
  // 0분:0초를 01분 01초로 설정
  let myTime = `${myMinute}:${mySecond}`;
  // 시간계산
```

데이터 처리 메소드

데이터를 처리하여 새로운 형식으로 반환합니다.
데이터 배열에서 첫 번째 요소를 제거하고
(`shift` 메소드를 사용하여),
제거된 요소의 `id`를 기반으로 새로운 `id`를
생성합니다.
이 경우에는 6을 더하여 새로운 `id`를 생성합니다.



```
const express = require('express');
const app = express();
const session = require('express-session');
const bodyParser = require('body-parser');
const l_Data = require('./views/json/loginData.json');
const Auth = require('./views/js/auth');
const {
  processData
} = require('./views/js/processData')
const low = require('lowdb');
const FileSync = require('lowdb/adapters/FileSync');
const adapter = new FileSync('./views/json/db.json');
const db = low(adapter);

app.locals.pretty = true;
app.set('view engine', 'pug');
app.set('views', './views');
```

Express를 활용 웹 개발

Express 프레임워크를 사용하여 웹 애플리케이션을 구상하였습니다.

세션, 요청 파싱, 사용자 인증은 각각 express-session, body-parser, Auth 모듈을 사용합니다.

데이터 처리는 processData 모듈을 사용하며, lowdb를 데이터베이스로 활용합니다.

```
// 세션 추가 0305
app.use(session({
  secret: 'secret_key',
  resave: false,
  saveUninitialized: true,
}));

// 바디 파서 추가 0304
app.use(bodyParser.urlencoded({
  extended: true
}));

// 사용자가 로그인되었는지 확인하는 미들웨어
app.use((req, res, next) => {
  if (req.session && !req.path.startsWith('/admin')) {
    delete req.session.user;
  }
  next();
})
```

세션을 활용한 요청

Express 애플리케이션에 세션 미들웨어를 추가하여 세션을 사용합니다. 세션은 클라이언트의 요청 간 상태를 유지하고 저장하기 위해 사용됩니다. 이를 위해 secret 키를 사용하여 세션을 암호화하고, resave 및 saveUninitialized 옵션을 설정하여 세션을 저장하고 초기화합니다.



```
p.clear(); // 캔버스를 프레임마다 초기화  
  
graphData.jsonData(); // jsonData 불러오기  
p.frameRate(100) // 프레임 속도 초당 100초  
  
p.timeValue = []; //시간  
p.data1Value = []; //데이터1  
p.data2Value = []; //데이터2  
  
for (let key in graphData.temperatureCelsius) { // 온도 -> data1Value 시간 -> timeValue 파싱  
    for (let key2 in graphData.temperatureCelsius[key]) {  
        for (let key3 in graphData.temperatureCelsius[key][key2]) {  
            if (key3 === '시간') {  
                p.timeValue.push(graphData.temperatureCelsius[key][key2]['시간']);  
            } else {  
                p.data1Value.push(graphData.temperatureCelsius[key][key2][key3]);  
            }  
        }  
    }  
}
```

JSON 데이터 파싱

프레임마다 캔버스를 초기화하고,
그래프를 그리기 위한 데이터를
jsonData에서 가져와 각 배열에
파싱하는 작업을 수행합니다.



Canvas 생성

부모 div의 너비 높이 만큼의 사이즈로 캔버스를 생성하고 id를 지정 해줍니다.

```
p.setup = function () {
    // 캔버스를 부모 div의 너비 높이 사이즈로 생성
    let canvas = p.createCanvas(document.getElementById('graph1').offsetWidth, document.getElementById('graph1').offsetHeight);
    canvas.id('canvas1'); // 캔버스 id 지정
}
```





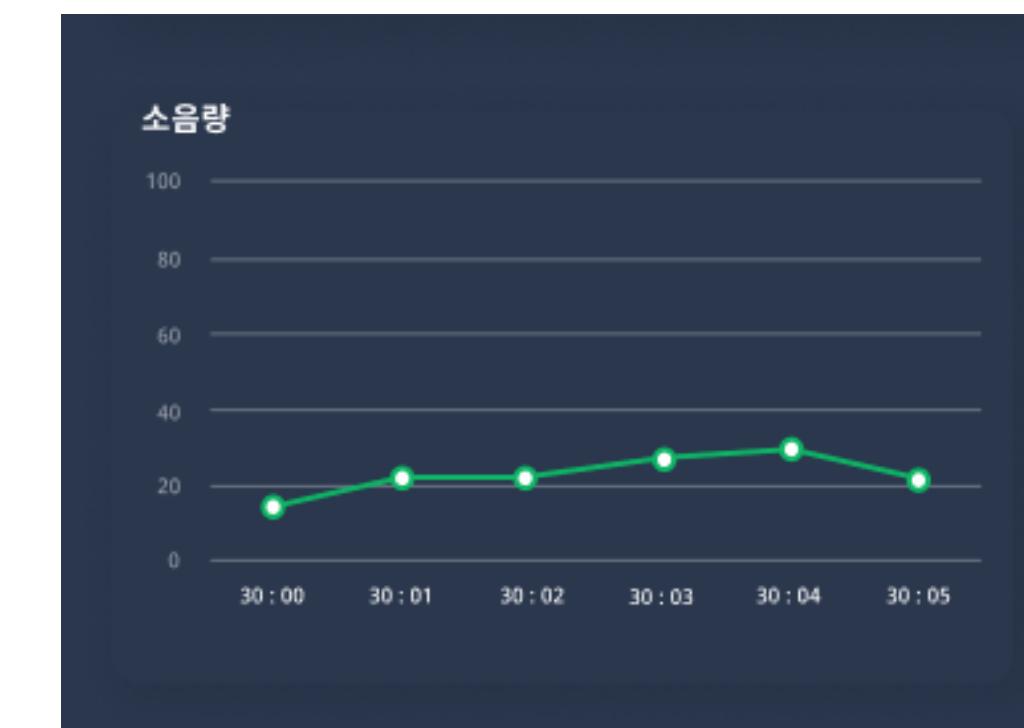
동적으로 조정되는 Canvas

브라우저 크기가 변경될 때마다 새로운 너비와 높이 값을 갖는 캔버스를 생성하고,
이를 부모 div의 사이즈에 맞게 조정합니다.

```
p.windowResized = function() { //창 크기가 변경될때마다 변한 너비 높이값으로 캔버스를 재생성  
    p.resizeCanvas(document.getElementById('graph1').offsetWidth,document.getElementById('graph1').offsetHeight);  
}
```



Tablet version



Mobile version

05

Review

기획, JSON, REST, JS, Node.JS 담당



장승우 - 팀장

이번 팀 프로젝트에서 기획부터 JSON, REST, JavaScript, Node.js를 수행했습니다.

특히, 프로젝트에서 JSON과 REST를 활용하여 데이터를 교환하고, JavaScript와 Node.js를 사용하여 프론트엔드와 백엔드를 개발하는 경험을 쌓았습니다.

또한, 프로젝트의 초기 단계에서 사용자 요구사항을 분석하고 기획을 수립하여 프로젝트의 목표와 방향을 명확히 하였습니다.

팀장을 맡아 프로젝트 진행 중에는 팀원들의 업무를 조율하고 프로젝트 일정을 관리하여 팀원들 간의 원활한 협업을 돋는 역할을 수행했습니다.

이러한 경험을 통해 팀원과의 협업 능력과 프로젝트 관리 능력을 향상시키고, 앞으로도 지속적으로 기술을 학습하여 더 나은 프로젝트를 이끌어 나갈 것입니다.

JSON, Node.JS 담당



이번 프로젝트는 저에게 많은 배움의 기회를 제공해 주었습니다. 간단한 로그인 기능을 통해 클라이언트와 서버 간의 효과적인 통신 방법을 습득할 수 있었으며, Express 세션과 같은 미들웨어를 사용하는 방법에 대해도 더 많이 이해할 수 있었습니다.

또한, JSON 데이터를 처리하는 방법에 대한 지식을 향상시킬 수 있었는데, 이는 json-server를 통해 데이터를 다루는 과정에서 배웠습니다. 혼자서 진행했다면 헤매고 어려움을 겪었을 부분을 팀원들과의 상의를 통해 더 쉽게 해결할 수 있었습니다.

배석찬

이번 경험을 통해 기획 단계의 중요성과 효과적인 소통이 프로젝트를 성공적으로 이끌어 나갈 수 있다는 것을 깨달았습니다. 앞으로의 프로젝트에서도 이러한 경험을 적극적으로 활용하여 팀원들과 협력하여 더 나은 결과물을 만들어 나가고자 합니다.

기획, 디자인, JavaScript 기능 구현 담당



이소윤

팀 프로젝트를 시작할 때에는 대시보드를 구성하는 요소들과 그 기능에 대한 이해도가 낮아서 처음에는 어려움을 겪었습니다.

하지만 팀원들이 필요한 정보를 설명해주며 함께 문제를 해결하기 위해 적극적으로 노력했습니다.

특히, 대시보드를 구현하는 과정에서 웹 디자인과 JavaScript 효과를 구현하는 역할을 맡았기 때문에, 이에 대한 지식과 기술적인 면에서도 도움을 받았습니다.

이러한 팀원들의 협력 덕분에 프로젝트를 완료할 수 있었습니다.

이 경험을 통해 팀워크의 중요성과 함께, 서로 도와주고 협력하는 것이 프로젝트의 성공에 얼마나 중요한지 알게 되었습니다.

이를 바탕으로 다음 팀 프로젝트에서도 팀 협업의 가치를 생각하며 진행할 것입니다.

디자인, 퍼블리셔, JavaScript 담당



이정민

이번 대시보드 프로젝트를 통해 협업에 대한 중요성을 깨닫게 되었습니다. 여러 명의 다양한 아이디어와 의견을 조율하여 하나의 프로젝트로 완성시키는 것이 쉬운 일이 아니지만, 저희 팀은 그 과정에서 많은 것을 배웠고 성장할 수 있는 기회를 얻을 수 있었다고 생각합니다.

특히, 이번 프로젝트를 통해 기획 단계가 얼마나 중요한지 명확히 이해했습니다. 각자의 아이디어와 목표를 조합하여 더 나은 방향을 찾아가는 것이 중요하다는 것을 깨달았습니다. 다음 프로젝트에서는 이러한 경험을 바탕으로 더 나은 방향을 제시할 수 있을 것이라고 생각합니다.

이번 프로젝트에서 얻은 경험과 지식을 활용하여 다음 프로젝트에서 더 효율적으로 협업하고 더 나은 결과물을 만들어낼 수 있기를 기대하고 있습니다. 함께 성장하고 발전한 이번 경험이 정말 소중합니다.

p5 담당



이번 대시 보드 프로젝트에서 저는 p5.js를 이용해 실시간으로 생성되는 JSON 데이터를 그래프로 시각화 하는 부분을 맡았습니다.

각 항목별로 파싱된 데이터들을 불러와 재파싱하여 각 Canvas에서 그래프로 시각화 하였고 브라우저 크기마다 스케일, 너비, 높이, 간격 등 다르게 조절하여 반응형으로 제작하였습니다.

이전에는 CSS를 통해 반응형 Contents를 제작했다면 이번에는 스크립트 코드를 이용해 제작하다 보니 낯설고 힘들었지만 하나하나 완성되는 모습을 보며 뿌듯함을 느꼈습니다.

전영진

어떤 어려운 프로젝트라도 서로 협업하며 진행하다 보면 잘 마무리할 수 있겠다는 자신감을 얻었고 이번 프로젝트를 통해 개인의 기술, 역량도 중요하지만, 무엇보다 제일 중요한 건 서로 간의 소통이라는 걸 깨닫게 되었습니다.

Server 담당



최종윤

제가 수행한 작업은 실시간으로 서버와 클라이언트간의 원활한 비동기 통신을 위한 서버 코드 수정 및 express 최적화 작업이었습니다.
이전에 php 백엔드 현장에서 일한 경험으로 팀 프로젝트에 많은 도움이 되었다고 생각합니다.

이번 프로젝트로 오랜만에 협업을 진행해 보았고, 많은 느끼는 바가 있었습니다.
현장에서는 이미 위계 질서가 잡혀있는 상황이라 팀원들과 의견 충돌은 자주 발생하지 않는데에 비해 서로 동등한 위치에서 프로젝트를 진행하게 되면 충돌이 발생할 수도 있다는 것을 깨달았고, 이러한 상황이 오랜만이라 신선하게 다가왔습니다.
물론 의견 충돌이 나쁜 점만 있는 것은 아니었습니다.
서로간의 관점 차이를 이해하고, 좀 더 나은 방향으로 프로젝트를 진행될 수 있어, 결과적으로 팀 프로젝트에 긍정적인 영향을 미쳤다고 생각합니다.



감사합니다.

장승우 배석찬 이소윤 이정민 전영진 최종윤