**2023 Web Programming – MiniProject**

**19101990 박한빈**

**\* eClass 종합 과제 관리 시스템**

**JSON 파일을 데이터베이스 용도로 사용하고 이를 수정하는 방식으로 진행함**

**텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명**

**Index는 순서**

**Subject는 과목명**

**Professor는 교수명**

**Details는 상세 내역**

**Deadline은 마감일**

**Submit은 제출여부**

**Submit\_details는 제출내용이다.**

**Submit의 기본값은 N.**

**Submit\_details는 ‘’ 값이다.**

**기본 UI 생성 함수**

// 과제에 따른 UI 생성

function createAssignmentItem(item, ul, i) {

    //  각 과제마다 li이 생성하도록 설정

    const li = document.createElement('li');

    li.classList.add('content');

    // 기본적으로 생성되는 ui를 보여주는 div 생성

    const defaultView = document.createElement('div');

    defaultView.classList.add('default\_view');

    //  상세 관련 내용이 있는 div 생성

    const detailText = document.createElement('div');

    detailText.classList.add('detail\_text');

    detailText.setAttribute('id', `detailText${i}`);

    detailText.textContent = item.Details

    // 작성 관련 div 생성

    const postPart = document.createElement('div');

    postPart.classList.add('post\_part');

    postPart.setAttribute('id', `post\_part${i}`);

    postPart.textContent = ""

    // 과제 작성 textarea 생성

    const textarea = document.createElement('textarea');

    textarea.setAttribute('id', `textarea${i}`);

    textarea.classList.add('textarea');

    textarea.placeholder = '과제를 작성하세요';

    // 과제명 span 생성

    const taskName = document.createElement('span');

    taskName.classList.add('task-name');

    taskName.textContent = item.taskName;

    // 과제 제출 버튼 생성

    const submit\_button = document.createElement('button');

    submit\_button.setAttribute('id', `submit\_button${i}`);

    submit\_button.classList.add('submit\_button');

    submit\_button.textContent = "제출"

// 과제 제출 시 json 파일의 submit 값과 submit details 값을 변경해주도록 함

    submit\_button.addEventListener('click', () => {

  const content = textarea.value;

  const submit\_data = {

    submit: "Y",

    submit\_details: content

  };

  const taskName = item.taskName;

  const url = `http://localhost:3000/taskName/${encodeURIComponent(taskName)}`;

  patchData(url, 'PATCH', submit\_data);

});

//  화면 우측 버튼 및 문자 div 생성

    const detail = document.createElement('div');

    detail.classList.add('detail');

    const detailWrapper = document.createElement('div');

    detailWrapper.classList.add('detail-wrapper');

    // 버튼을 모아놓은 div 생성

    const buttonWrapper = document.createElement('div');

    buttonWrapper.classList.add('button-wrapper');

        // 상세 버튼 생성

    const detailButton = document.createElement('button');

    detailButton.classList.add('button');

    detailButton.setAttribute('id', `detailButton${i}`);

    detailButton.textContent = '상세';

    // 상세 버튼 클릭시 상세 내역이 나올수있도록 해줌

    detailButton.addEventListener('click', () => {

    detailText.style.display = detailText.style.display === 'none' ? 'block' : 'none';

});

// 작성 버튼 생성

    const postButton = document.createElement('button');

    postButton.classList.add('button');

    postButton.setAttribute('id', `postButton${i}`);

    postButton.textContent = '작성';

        // 작성 버튼 클릭시 작성창이 나올수있도록 해줌

    postButton.addEventListener('click', () => {

        textarea.style.display = textarea.style.display === 'none' ? 'block' : 'none';

        submit\_button.style.display = submit\_button.style.display === 'none' ? 'block' : 'none';

});

// 수업명

    const subject = document.createElement('span');

    subject.classList.add('subject');

    subject.textContent = item.subject;

// 교수명

    const professor = document.createElement('span');

    professor.classList.add('professor');

    professor.textContent = item.professor;

// 마감일

    const deadline = document.createElement('span');

    deadline.classList.add('deadline');

    deadline.textContent = item.deadline;

    detailWrapper.appendChild(subject);

    detailWrapper.appendChild(professor);

    detailWrapper.appendChild(deadline);

    buttonWrapper.appendChild(detailButton);

    buttonWrapper.appendChild(postButton);

    postPart.appendChild(textarea);

    postPart.appendChild(submit\_button);

    detail.appendChild(buttonWrapper);

    detail.appendChild(detailWrapper);

    defaultView.appendChild(taskName);

    defaultView.appendChild(detail);

    li.appendChild(defaultView);

    li.appendChild(detailText);

    li.appendChild(postPart);

    ul.appendChild(li);

}

1. **현재 부여된 전체 미해결 과제 목록 조회 (10점)**

**프론트**

데이터 로드를 위한 함수

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명빈 값의 ul을 생성 후 data를 받아서 createAssignmentItem을 통해 각 과제 하나하나씩 li로 생성해서 넣어줌

백엔드

// 미제출 조회

app.get("/", (req, res) => {

    fs.readFile(

        \_\_dirname + "/data/UnslovedAssignment.json",

        "utf8",

        (err, data) => {

            if (err) {

                console.error(err);

                res.status(500).send("Internal Server Error");

                return;

            }

            let jsonData = JSON.parse(data);

            const filteredData = jsonData.filter((item) => item.submit === "N");

            res.json(filteredData);

        }

    );

});

1. **과제 필터 기능**

**프론트**

// 검색

document.getElementById('inquiry').addEventListener('click', () => {

    const selectedOption = document.querySelector('input[name="Select\_opt"]:checked').value;

    const searchText = document.getElementById('inquiry\_text').value;

    const url = `http://localhost:3000/UnslovedAssignment?selectedOption=${selectedOption}&searchText=${searchText}`;

    loadData(url);

});

**백엔드**

// 선택 조회

app.get("/UnslovedAssignment", (req, res) => {

    fs.readFile(

        \_\_dirname + "/data/UnslovedAssignment.json",

        "utf8",

        (err, data) => {

            if (err) {

                console.error(err);

                res.status(500).send("Internal Server Error");

                return;

            }

            let jsonData = JSON.parse(data);

            let selectedOption = req.query.selectedOption;

            let searchText = req.query.searchText;

            let filteredData = [];

            if (selectedOption === "Subject") {

                filteredData = jsonData.filter((item) =>

                    item.subject.includes(searchText)

                );

            } else if (selectedOption === "Task\_Name") {

                filteredData = jsonData.filter((item) =>

                    item.taskName.includes(searchText)

                );

            }

            res.json(filteredData);

        }

    );

});

1. **과제 조회 화면에서 특정 과제 내용 작성 및 제출 기능 (35점)**

**프론트**

// 데이터 수정

function patchData(url, method, data) {

  const options = {

    method: method,

    headers: {'Content-Type': 'application/json'},

    body: JSON.stringify(data)

};

fetch(url, options)

    .then((response) => response.json())

    .then((data) => {

    })

    .catch((error) => {

    //   console.error("오류가 발생했습니다:", error);

    });

}

**백엔드**

let assignments = [];

app.patch("/taskName/:taskName", (req, res) => {

    const taskName = req.params.taskName;

    const updatedAssignment = req.body;

    try {

        const data = fs.readFileSync("data/UnslovedAssignment.json", "utf8");

        assignments = JSON.parse(data);

        const assignment = assignments.find(

            (assignment) => assignment.taskName === taskName

        );

        if (!assignment) {

            res.status(404).send(`(${taskName}) is not find`);

        } else {

            Object.assign(assignment, updatedAssignment);

            fs.writeFileSync(

                "data/UnslovedAssignment.json",

                JSON.stringify(assignments, null, 2)

            );

            res.send(`(${taskName}) is updated`);

        }

    } catch (err) {

        console.error(err);

        res.status(500).send("Internal Server Error");

    }

});

**4) 제출 완료된 과제 목록 조회 (20점)**

🡪 제출이 완료된 과제 목록이 조회되어야 함

\* 조회 결과에는 교과목명, 담당교수명, 과제명, “상세” 버튼이 표시되어야 함 (10점)

\* “상세” 버튼을 클릭 시, **학생이 제출한 과제 내용**이 표시되어야 함 (10점)

**5) 보너스 (기본 과제를 50%이상 달성 후 진행해야 보너스 부여)**

(1) 서비스를 종료 후 재시작해도 기존 미해결 과제 및 과제 제출 기록 조회 가능 +30점

\* express 서버의 종료 및 재시작을 의미함