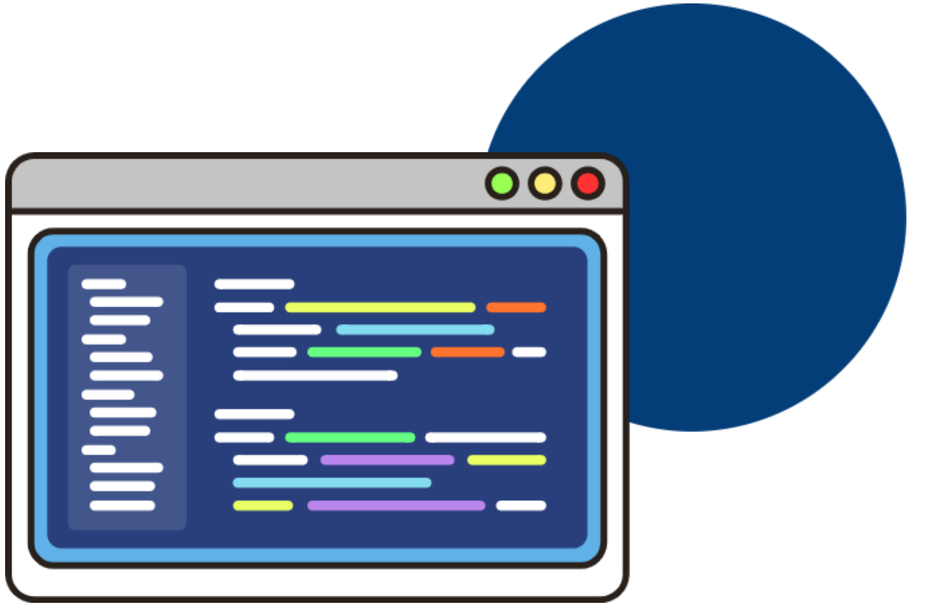
5빠 한번 믿어봐

띵유

명지대 웹 프로젝트



60222561 김예린
60231999 김채희
60212187 박성식
60222143 정유찬

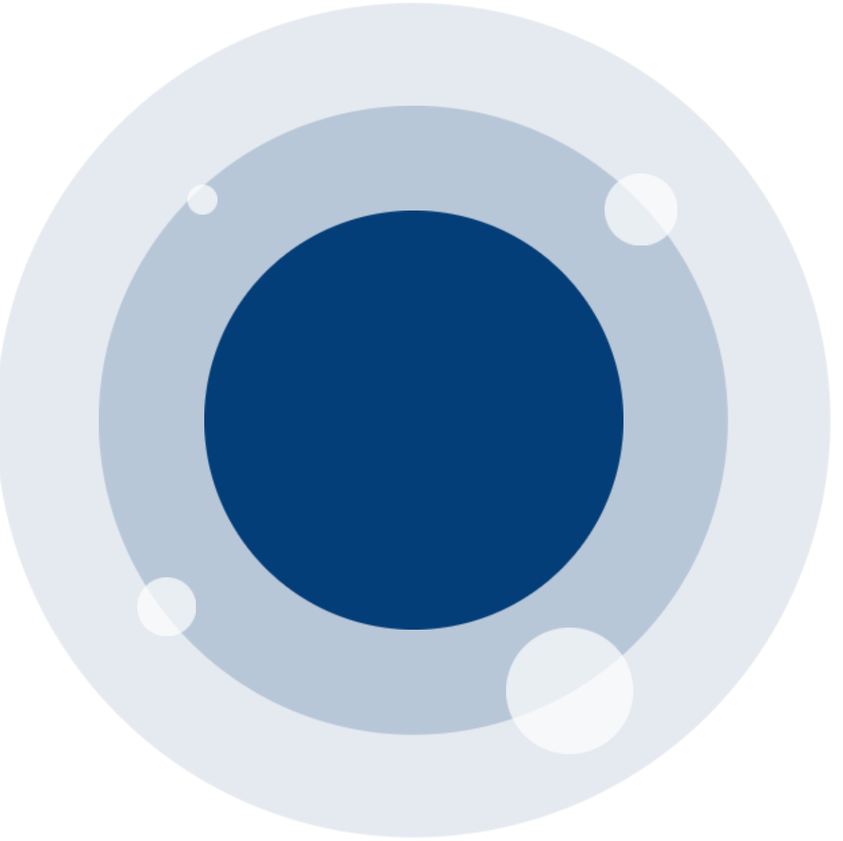
띵유

목차

WEP PROJECT

1.

프로젝트 개요



1. 프로젝트 개요

역할

정유찬

팀장 및 서버,
데이터 베이스,
백엔드 기능 구현 등
총괄 디렉터

김채희

UI 구현 담당,
프론트엔드 개발 담당

김예린

서버 담당,
백엔드 기능 개발

박성식

UI 구현 담당,
프론트엔드 개발 담당

1. 프로젝트 개요

주제 선정 배경

에브리타임 사용 중 겪은 불편함	난이도	시장성	경쟁력	차별성	총점 (100)	최종 순위
	30	30	20	20		
명지대 셔틀, 택시모임	6	10	9	6	31	3
명지대 동아리 홍보	15	13	7	7	42	1
학식 친구 구하기	7	15	8	4	34	2

→ 1순위 명지대 동아리와 2순위 학식 친구 구하기 비교

- 개발 가능 여부 : 한정된 시간 안에 개발이 가능한가?
- 자료구조 : 자료구조 사용 가능한가?
- 범용성 : 프로그램을 사용해서 다른 것에 응용 가능한가?

기준 / 아이디어	명지대 동아리 홍보	학식 친구 찾기
개발 가능 여부	+	-
자료구조	+	-
범용성	-	+

→ 명지대 동아리 만들기로 결정

1. 프로젝트 개요

목표

- **동아리 목록 제공**

  카테고리를 통해 관심 분야를 나눈 후 동아리 이름, 대표 사진, 동아리 소개와 같은 정보를 제공

- **보안 기능**

  학생들의 개인정보를 보호하기 위해 해싱 작업으로 데이터를 보관 및 TLS 암호화를 통해 데이터 전송 중 보안 유지

- **검증 기능**

  학생들이 제출하는 신청서는 동아리 페이지 개설자에 의해 수동으로 검토되어 신회성 있는 동아리 가입 절차를 보장

- **사용자 친화적인 인터페이스**

  학생들이 쉽게 웹 어플리케이션을 탐색할 수 있도록 직관적인 사용자 인터페이스를 제공

- **동아리 신청 기능**

  학생들이 관심 있는 동아리에 직접 온라인으로 신청할 수 있는 기능을 구현

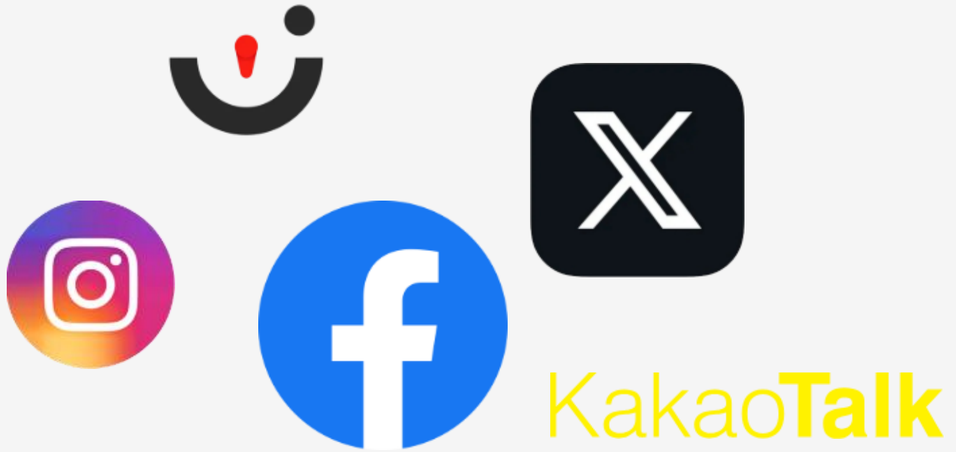
- **동아리 평점 및 리뷰 기능**

  사용자는 동아리에 대한 평점과 후기를 남길 수 있으며, 이를 통해 다른 학생들은 해당 동아리에 대한 정보를 더욱 객관적으로 파악

2.

프로젝트 성공 시의 기대 효과 및 개선 효과

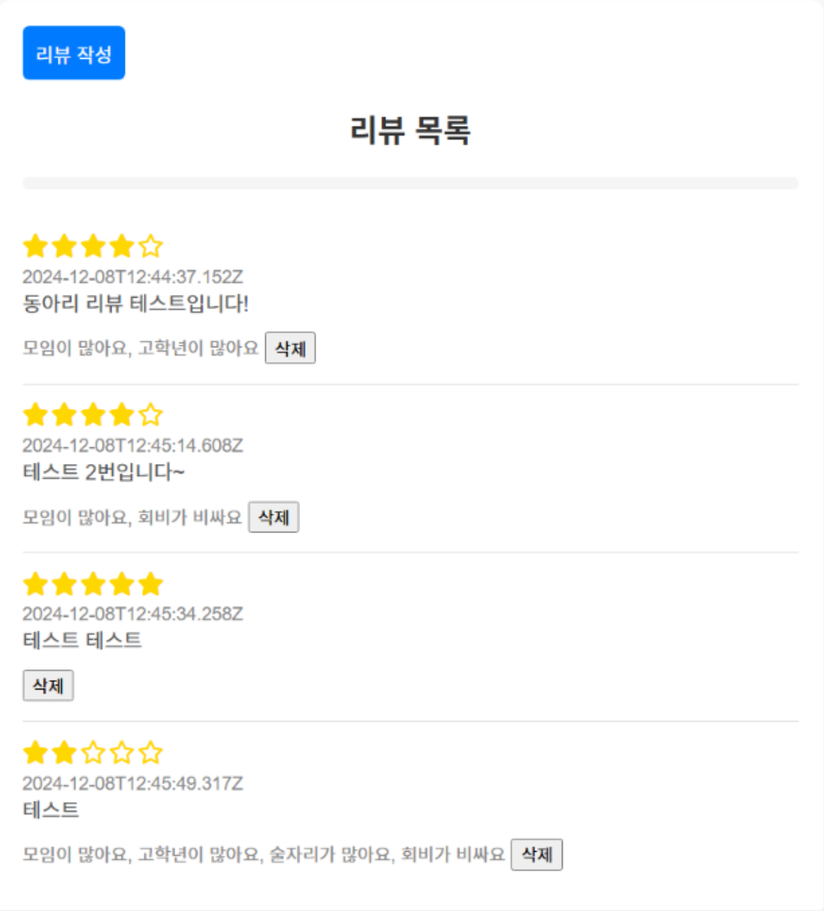
2. 프로젝트 성공 시의 기대 효과 및 개선 효과



동아리 정보 찾기 어려움

동아리를 찾아 볼 수 있는
하나의 웹

2. 프로젝트 성공 시의 기대 효과 및 개선 효과



- **동아리 별점 및 후기 작성**

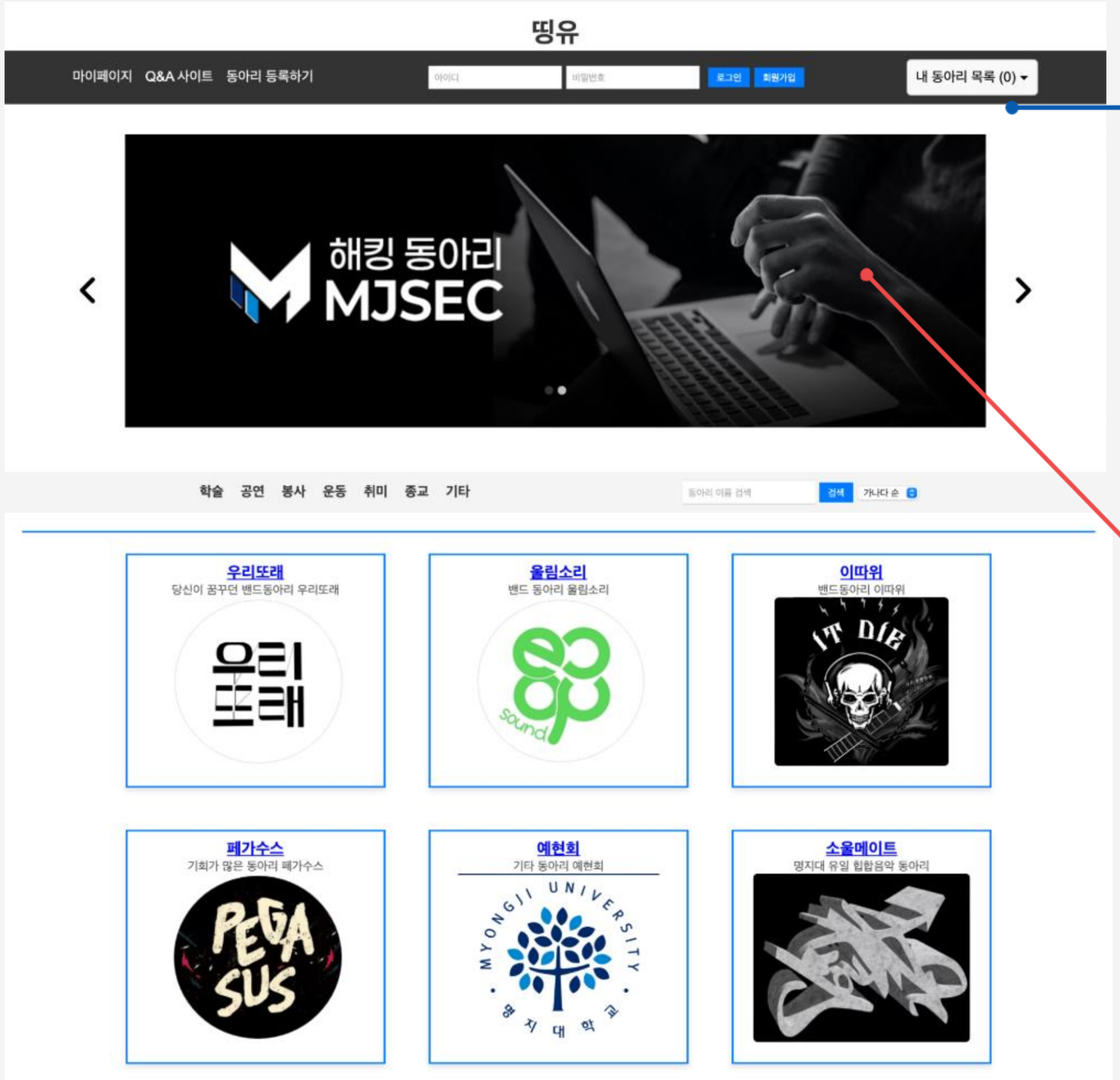
  - 동아리 가입을 원하는 이용자들이 별점과 후기를 통해 동아리의 장단점, 실제 경험을 확인 가능
    → 자신에게 더 적합한 동아리 선택을 할 수 있음

  - 여러 동아리를 직접 탐방하거나 참여하기 전에 사전 검토가 가능함
    → 시간과 에너지 절약

3.

프로젝트 최종 성과

3. 프로젝트 최종 성과



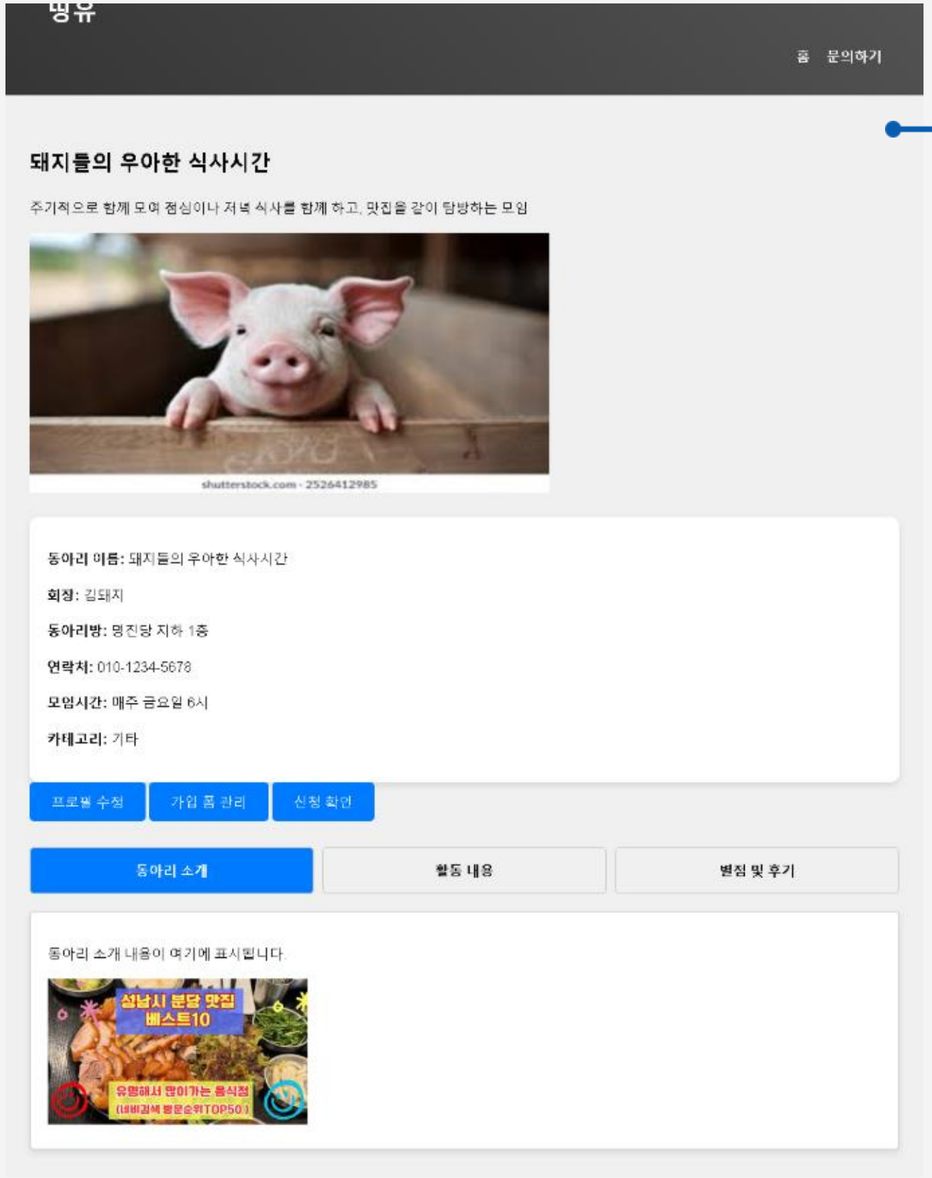
메인페이지

- 유저는 동아리 장르별로 동아리 목록을 볼 수 있다.
- 유저는 동아리 검색을 할 수 있다.
- 유저는 로그인과 회원가입을 할 수 있다.
- 유저는 동아리 목록 정렬순을 변경할 수 있다.
- 유저는 내 동아리 목록을 확인할 수 있다.

중간 발표 feedback 1번: 동아리 선택의 편의를 위해, 관심사 입력을 통한 추천시스템이 있었으면 좋을 것 같다

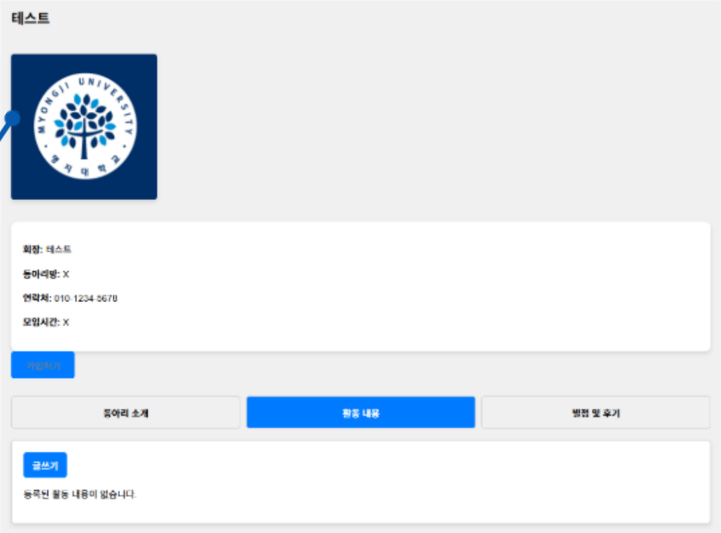
→개발 역량 부족으로 구현하지 못함

3. 프로젝트 최종 성과



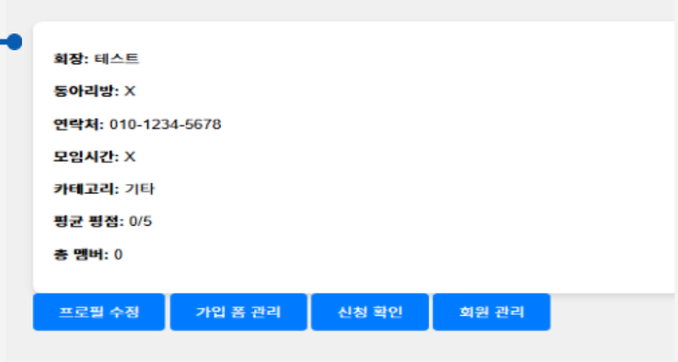
동아리 페이지

- 유저는 동아리 페이지에 접속 할 수 있다.
- 유저는 동아리 소개, 활동 내용, 별점 및 후기를 볼 수 있다.
- 유저는 동아리 가입 신청을 할 수 있다.



동아리 관리자 페이지

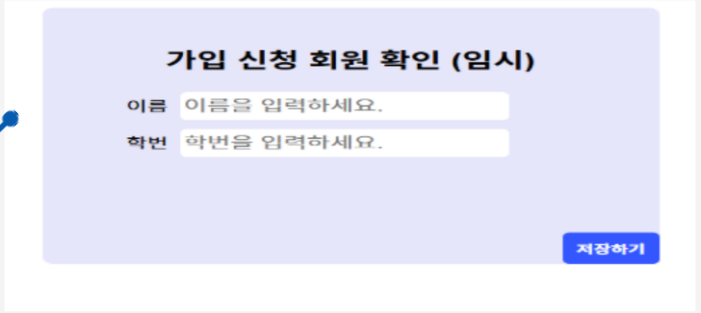
- 동아리 관리자 유저에게는 일반 유저와는 별도의 페이지에 접속할 수 있다.
- 동아리 관리자 유저는 동아리 프로필 수정, 가입 폼 관리 및 회원 관리가 가능하다.



3. 프로젝트 최종 성과

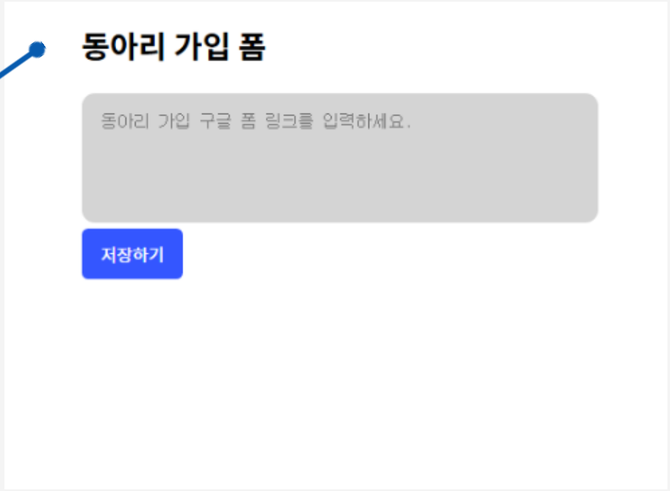
동아리 수정 페이지

- 동아리 관리자 유저는 수정 페이지에 접속할 수 있다.
- 동아리 관리자 유저는 동아리 정보 수정이 가능하다.



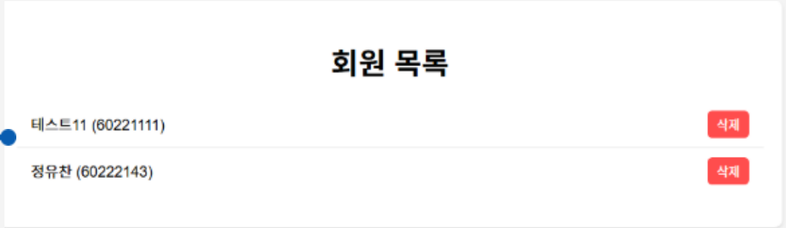
가입 신청 회원 확인 페이지

- 동아리 관리자 유저는 동아리 정보 수정이 가능하다.



동아리 가입 폼 페이지

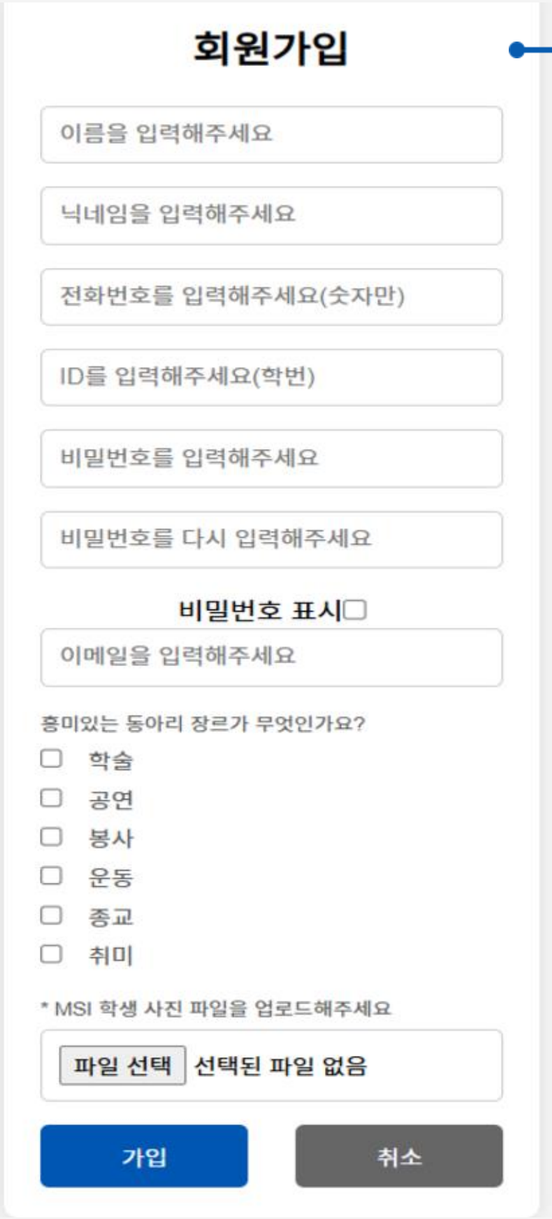
- 동아리 관리자 유저는 동아리 가입 구글 폼 링크를 기입할 수 있다.



동아리 회원 목록 페이지

- 동아리 관리자 유저는 동아리 회원 목록 열람이 가능하다.

3. 프로젝트 최종 성과

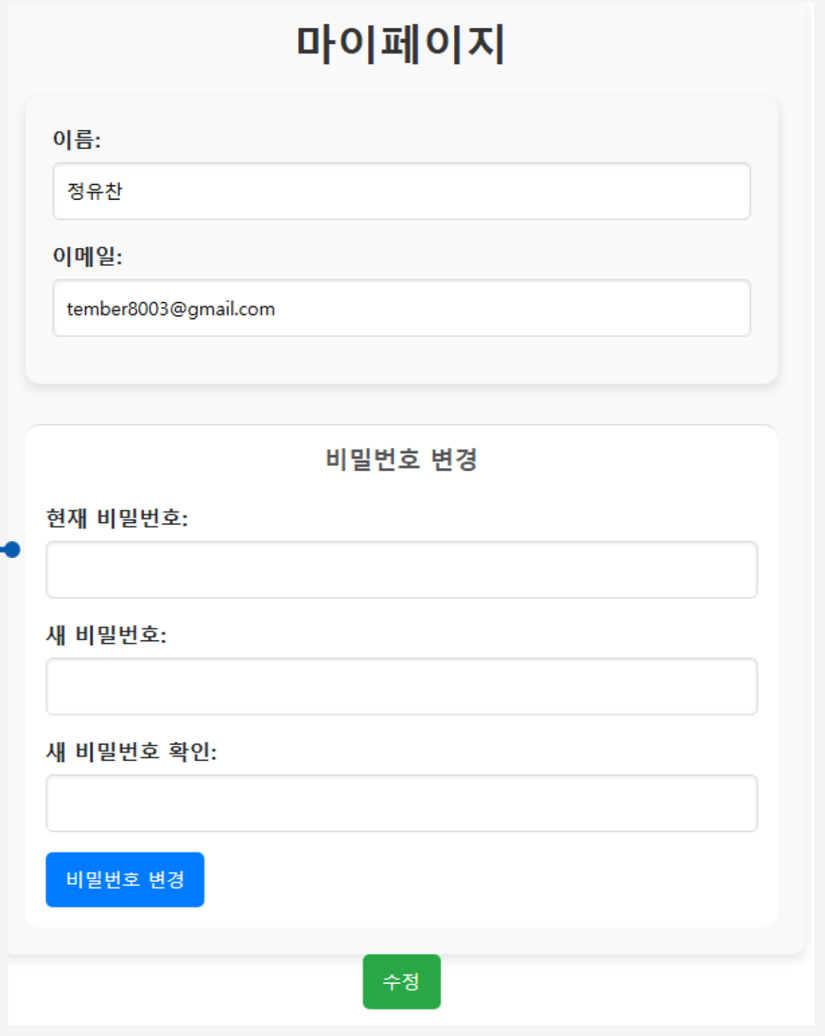


회원가입

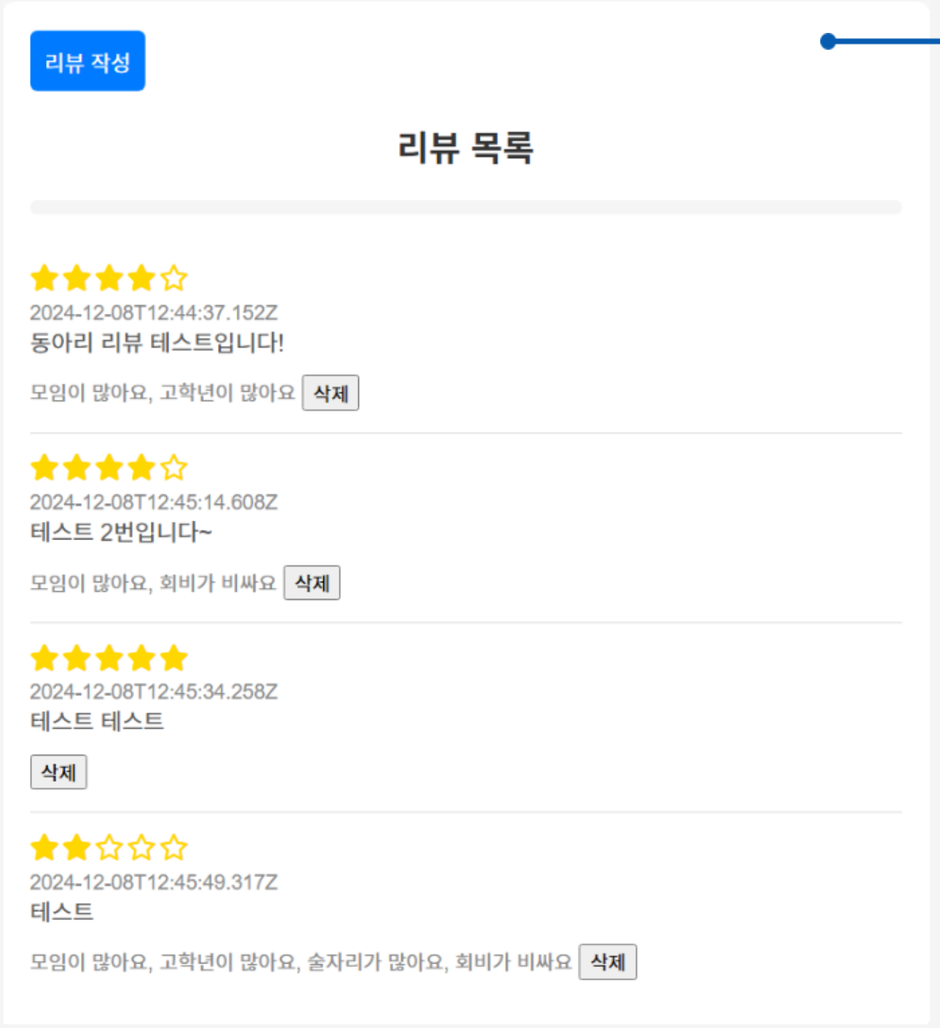
- 유저는 회원가입 페이지에 접속 할 수 있다.
- 유저는 회원 가입 시 흥미있는 동아리 장르를 선택 할 수 있다.
- 유저는 회원 가입 시 MSI 사진 파일을 업로드하여 명지대 재학생임을 인증한다.

마이페이지

- 유저는 마이페이지에 접속 할 수 있다.
- 유저는 마이페이지에서 개인 정보를 수정 할 수 있다.
- 유저는 마이페이지에서 작성한 후기를 볼 수 있다.



3. 프로젝트 최종 성과

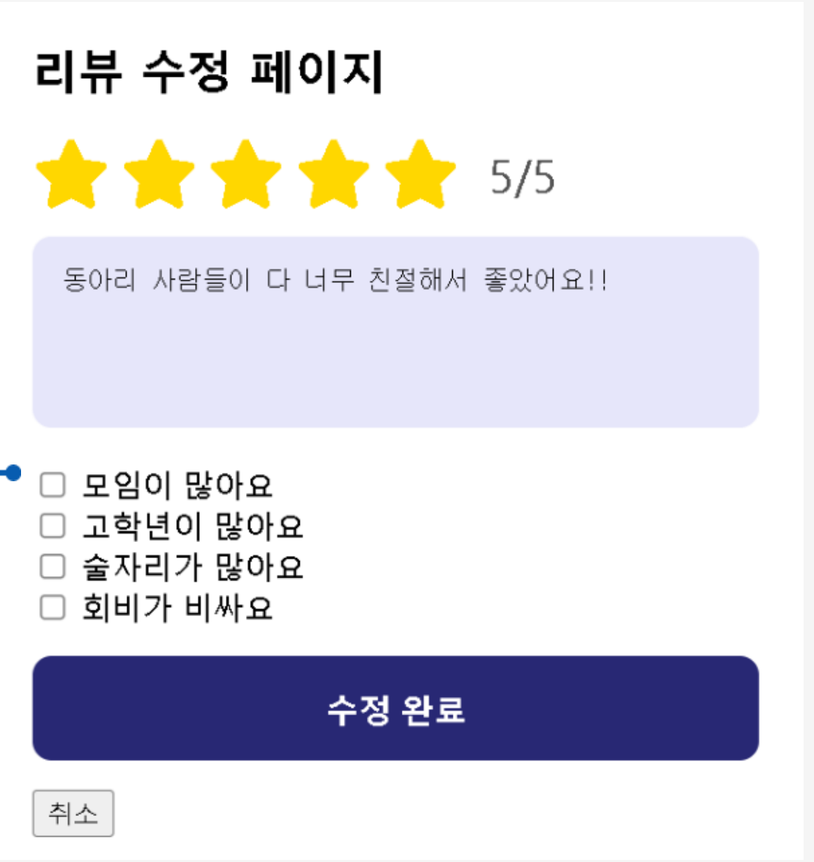


동아리 평점 및 리뷰

- 유저는 동아리 평점 및 리뷰를 볼 수 있다.
- 유저는 동아리 평점 및 리뷰를 작성 및 수정, 삭제를 할 수 있다.

리뷰 수정 페이지

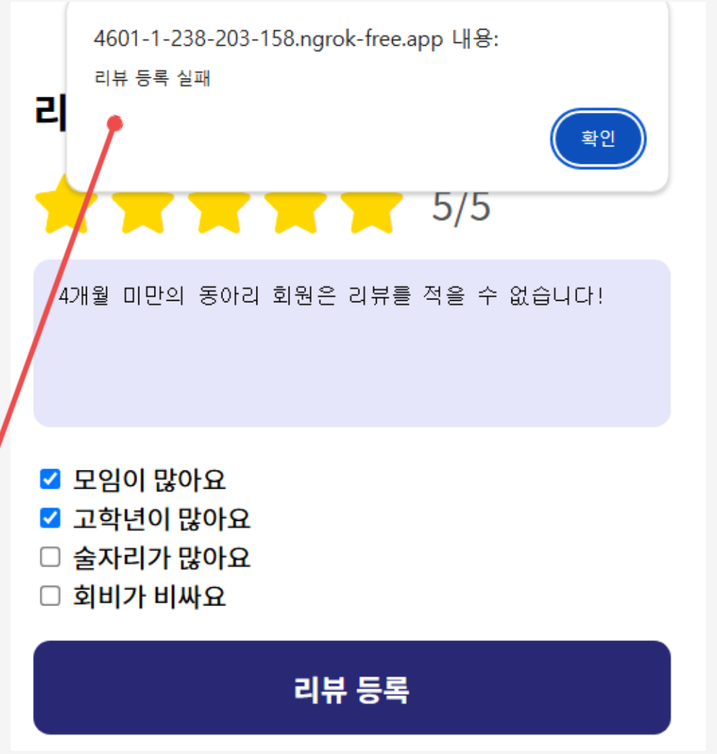
- 유저는 동아리 평점 및 리뷰 수정 페이지에 접속할 수 있다.
- 유저는 동아리 평점 및 리뷰를 수정할 수 있다.



3. 프로젝트 최종 성과

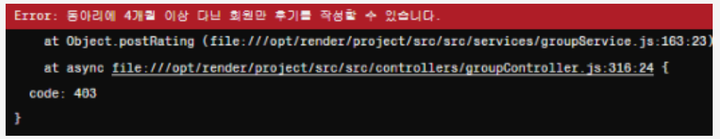
리뷰 작성 페이지

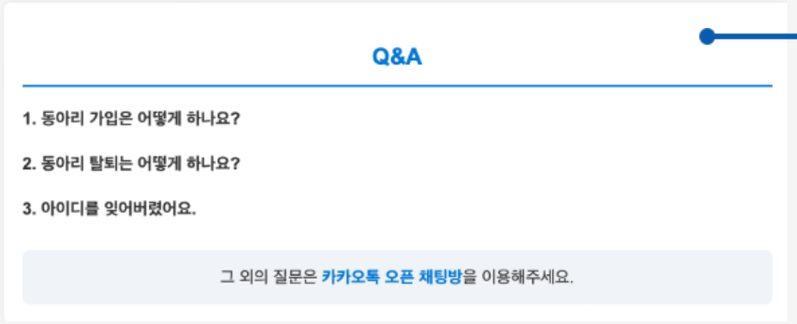
- 유저는 리뷰 작성 페이지에 접속할 수 있다.
- 유저는 동아리 평점 및 리뷰를 작성할 수 있다.



중간 발표 feedback 7번: 동아리 후기를 아무나 작성하면 별점 테러 우려 -> 실제 동아리를 일정기간 이상(6개월, 1년...) 활동한 사람들만 후기를 작성하면 해결 가능할 듯

- 별점 테러 방지를 위하여 채택 및 구현



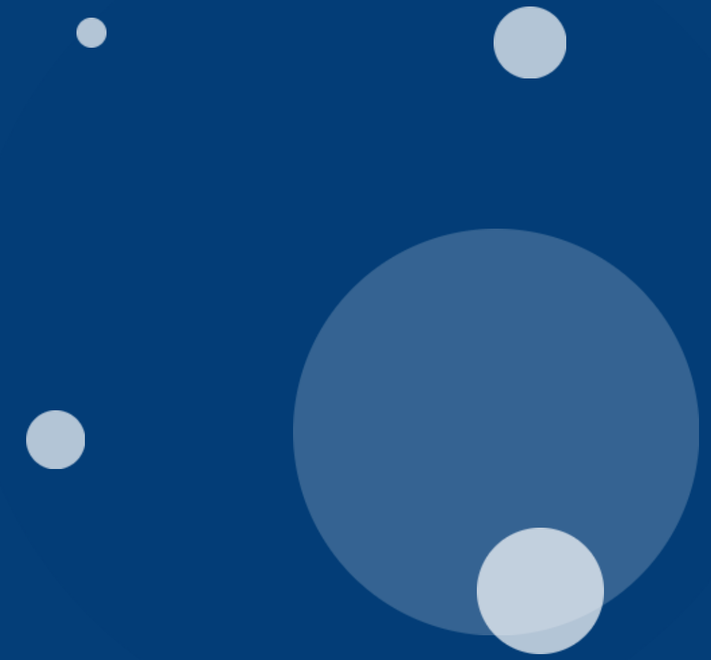


Q&A 페이지

- 유저는 Q&A 페이지에 접속할 수 있다.
- 유저는 자주 묻는 질문과 답변을 확인할 수 있다.

4.

Demonstration



4. Demonstration

5.

프로젝트 진행 시
어려웠던 점과 극복 과정

5. 프로젝트 진행 시 어려웠던 점 및 극복 과정 (프론트엔드)

1. 문제 상황

동아리 등록 페이지에서 '등록하기' 버튼을 누른 후 데이터가 서버로 전달되지 않고 에러 발생하여 동아리 등록 실패 메시지가 출력됨 또는 아무 반응이 없음

2. 정보 파악

파일 확인

- HTML 파일(club_reigster.html): 등록 폼 설정과 제출 버튼
- JavaScript 파일(main.js): 폼 제출 이벤트 처리
- Python 서버 파일(server.py): /register_club 엔드포인트 처리

→ 클라이언트와 서버 간 데이터 전송 과정에서 오류 가능성 발견

5. 프로젝트 진행 시 어려웠던 점 및 극복 과정 (프론트엔드)

3. 문제 격리

- 클라이언트 측 JavaScript 문제

  → JSON 포맷으로 데이터를 전송하고 있으나, 서버는 `request.form`과 파일 업로드(`request.files`)를 사용

- 서버 측 /register_club 엔드포인트 문제

  → server.py에서 request.form으로 데이터를 수신하지만, 클라이언트는 Content-Type: application/json으로 데이터를 전송

  → 파일 업로드(request.files) 처리 누락 또는 불일치

- API 호출

  → JAPI 호출 성공 여부 및 응답 확인

  → API URL과 인증 토큰 확인

5. 프로젝트 진행 시 어려웠던 점 및 극복 과정 (프론트엔드)

4. 수정 후 테스트 및 확인

- 클라이언트와 서버 간 데이터 포맷 일치

  → main.js: FormData 객체 사용으로 전환

```
/ 기존 JSON 데이터 전송을 FormData로 변경
document.getElementById("club-register-form").addEventListener("submit", async function
(event) {
  event.preventDefault();

  const formData = new FormData(this); // FormData 객체 생성

  try {
    const response = await fetch("/register_club", {
      method: "POST",
      body: formData // FormData를 그대로 전송
    });
    const result = await response.json();
    if (result.success) {
      alert("동아리 등록 성공!");
      window.location.href = "/";
    } else {
      alert(result.message || "동아리 등록 실패!");
    }
  } catch (error) {
    console.error("서버 오류:", error);
    alert("서버와 통신 중 문제가 발생했습니다.");
  }
});
```

5. 프로젝트 진행 시 어려웠던 점 및 극복 과정 (프론트엔드)

4. 수정 후 테스트 및 확인

- 서버에서 파일 처리 로직 검증

  → server.py: request.files와 FormData가 올바르게 수신되는지 디버깅

- API 호출

  → Node.js 백엔드에서 API 응답 확인 및 문제 수정 후 정상적으로 데이터 전송 및 응답 처리 확인

```
@app.route('/register_club', methods=['POST'])
@authenticate_token
def register_club():
    # Form 데이터 수신
    club_name = request.form.get('club_name')
    club_leader = request.form.get('club_leader')
    group_image = request.files.get('GroupImage')  # 파일 업로드

    if not club_name or not club_leader or not group_image:
        return jsonify({"success": False, "message": "필수 데이터가 누락되었습니다."}), 400

    # Node.js API로 데이터 전송
    files_to_send = {
        'GroupImage': (group_image.filename, group_image.stream, group_image.content_type)
    }
    form_data = {
        "club_name": club_name,
        "club_leader": club_leader,
        "category": request.form.get('category'),
    }
    try:
        response = requests.post(
            'https://teamproject-jv72.onrender.com/api/group_form',
            data=form_data,
            files=files_to_send
        )
        response.raise_for_status()
        return jsonify({"success": True, "message": "동아리 등록 완료!"}), 201
    except requests.exceptions.RequestException as e:
        print(f"API 호출 오류: {e}")
        return jsonify({"success": False, "message": "백엔드 오류"}), 500
```

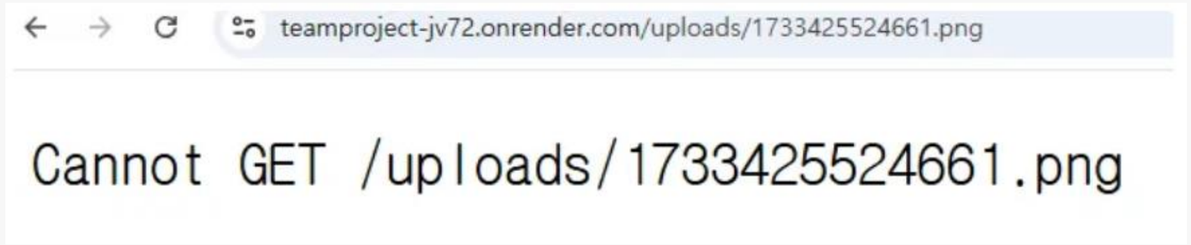
5. 프로젝트 진행 시 어려웠던 점 및 극복 과정 (백엔드)

1. 문제 상황

- 백엔드 Render로 코드 배포시 이미지 파일 확인 불가 문제

Render에 배포된 백엔드 서버에서 /uploads 폴더에 업로드된 이미지를 클라이언트가 확인할 수 없음. 이 문제는 클라이언트가 업로드한 파일의 접근 경로를 제공하는 기능이 제대로 작동하지 않아 사용자 경험에 큰 영향을 미침. 특히, 404 Not Found 에러로 인해 이미지 URL을 통한 GET 요청이 실패하는 상황이 발생함

- 이미지 URL을 통한 GET 조회 불가



5. 프로젝트 진행 시 어려웠던 점 및 극복 과정 (백엔드)

2. 정보 파악

- 환경: Render 배포 환경, Node.js 기반 서버
- 관련 기능: 파일 업로드, 파일 경로 제공, Express.js를 사용한 정적 파일 제공
- 초기 증상
  - 서버에서 파일 업로드는 정상적으로 이루어졌으나, 브라우저로 접근 시 `404 Not Found` 에러 발생
  - 접근 경로: `https://your-render-domain/uploads/<filename>`
- 추가 조사
  - Render에서 기본적으로 정적 파일 제공을 설정하지 않거나 특정 디렉터리의 접근 권한을 명시적으로 처리하지 않을 경우, 정적 파일이 노출되지 않을 수 있음
  - 관련 Express.js 설정 및 Render 환경에 관한 공식 문서 확인

5. 프로젝트 진행 시 어려웠던 점 및 극복 과정 (백엔드)

3. 문제 격리

테스트 범위

- 파일이 서버에 정상적으로 업로드되는지 확인 (파일 경로 및 저장 상태 점검)
- 브라우저에서의 접근 경로가 올바른지 확인
- Express.js의 정적 파일 제공(`express.static`) 설정 확인
- Render에서 `/uploads` 디렉터리의 접근 권한 및 경로 설정 여부 점검
- 회원가입이나 동아리 등록시 파일 경로 확인

확인 결과

- 파일이 서버의 `/uploads` 폴더에 정상적으로 저장됨을 확인
  → 회원가입, 동아리 등록시 이미지 파일 등록 후 데이터 저장되는 것을 확인
- 파일 저장 해당 경로가 destination: '/opt/render/project/src/src/uploads' 인 것을 확인
- 해당 경로는 Render의 임시 저장 폴더 경로임을 확인
  → Render에서는 `/opt/render/project/...` 경로가 기본적으로 임시 디스크로 설정되며, 서버 재시작 시 데이터가 유지되지 않아 이를 해결하기 위해 영구 저장소로 지정된 `/uploads` 디렉터리를 사용해야 함

5. 프로젝트 진행 시 어려웠던 점 및 극복 과정 (백엔드)

4. 테스트 및 확인

```
createMulterStorage(folderName) {
  // 프로젝트 루트 경로에서 src/group 경로로 설정
  const uploadPath = path.join(__dirname, '..', folderName); // '..'을 추가해 상위 폴더로 이동

  // 폴더가 존재하지 않으면 생성
  if (!fs.existsSync(uploadPath)) {
    fs.mkdirSync(uploadPath, { recursive: true });
  }

  return multer.diskStorage({
    destination: function (req, file, callback) {
      console.log("파일 저장 경로:", uploadPath); // 경로 확인
      callback(null, uploadPath);
    },
    filename(req, file, callback) {
      const fileName = Date.now() + path.extname(file.originalname);
      console.log("파일 이름:", fileName); // 파일 이름 확인
      callback(null, fileName);
    }
  });
}
```

새롭게 폴더가 존재하지 않는 경우 생성하는 방식으로 uploads 폴더를 새롭게 만드는 문제 발생

→ 해당 코드는 '상대 경로'로 폴더가 설정되어 있음
→ Render의 임시 폴더로 이동
이를 수정해 절대 경로로 설정하고 폴더를 새롭게 만들지 않도록 수정하며 데이터가 서버 재시작 후에도 유지되도록 보장하는`/uploads` 디렉터리를 사용해야 함

5. 프로젝트 진행 시 어려웠던 점 및 극복 과정 (백엔드)

4. 테스트 및 확인

```
const storage = multer.diskStorage({
    destination: function (req, file, callback) {
        const uploadPath = '/uploads'; // Persistent Disk 경로
        console.log("파일 저장 경로:", uploadPath); // 경로 확인 로그
        callback(null, uploadPath); // 저장 경로 지정
    },
    filename: function (req, file, callback) {
        const sanitizedFilename = Date.now() + '-' +
file.originalname.replace(/[^a-zA-Z0-9.]/g, '-');
        console.log("파일 이름:", sanitizedFilename); // 파일 이름 확인 로그
        callback(null, sanitizedFilename);
    }
});
```

Render 환경에 배포 후, `https://your-render domain/uploads/<filename>`로 접근 테스트

→ 이미지가 잘 나타나는 것을 확인함.

6.

프로젝트 제약 사항 및 향후 추가 개선 방안

6. 프로젝트 제약 사항 및 향후 추가 개선 방안

제약 사항

- **개발 인력의 변경**

  개발 내용과 정보, 진행 상황에 대한 정보를 노션, 깃허브를 통해 기록된 내용으로 공유, 교육

- **개발 기술 부족**

  다른 팀원들의 도움과 강의, 구글링, 공식 정보 등을 활용해 해결

- **개발 과정에서의 각종 예외처리**

  시작품의 결과물 완성 및 테스트, 디버깅 환경설정

- **중요한 상황에서의 인원 공백**

  개발자들 간의 개발 부분의 공유와 상황 공유를 통해 피해 최소화 후 작업 재분배

- **개발 시간 지연**

  노션, 깃허브를 통한 개발 진행 상황 확인 후 업무 재분배

- **한정된 개발 시간**

  적절한 시간 분배 계획 및 가장 중요한 기능의 품질을 우선적으로 개선하거나 추가

6. 프로젝트 제약 사항 및 향후 추가 개선 방안

기술적 부분

- **데이터베이스 성능 개선**

  적절한 인덱스 설정 및 쿼리 최적화

- **반응형 웹 디자인**

  다양한 해상도와 디바이스에서 최적화된 화면 제공

- **개인화된 경험 제공**

  사용자 행동 데이터를 분석해 맞춤형 콘텐츠 제공

- **보안 강화**

  웹페이지 트래픽을 암호화하여 데이터 도난 방지

- **코드의 간결화**

  코드 크기를 줄이고, 불필요한 공백과 주석 제거

- **브라우저 캐싱**

  자주 사용되는 리소스를 캐시에 저장해 재요청 감소.

6. 프로젝트 제약 사항 및 향후 추가 개선 방안

기능적 부분

명지대 재학생 인증 방식

MSI 사진 등록 방식에서 이메일 인증 방식으로 개선

홈페이지 메인 배너(중간고사 feedback 7번)

동아리 맞춤 추천 서비스로 개선

5빠 한번 믿어봐 - 띵유

Thanks for watching!

WEP Project
김예린

WEP Project
김채희

WEP Project
박성식

WEP Project
정유찬