전화번호 관리 프로그램

2조 발표 주제는 전화번호를 이용하여 회원을 관리하는 통신사 "GDH" 입니다.

팀원: 김도형(조장), 나현석, 이건민

스케줄

1
2024/12/24
프로젝트 기획 및 설계 단계

2
2024/12/26
기능구현

3
2024/12/27
기능구현

4
2024/12/30
데이터베이스 추가+기능에 연결

5
2024/12/31
프로그램 개발 및 테스트

6
2025/01/02
최종 마무리 및 발표 준비

7
2025/01/03
발표!

통신사를 채택한 이유!

전화번호 활용

관리자	회원
1.전화번호를 활용하여 회원을 효율적으로 관리 2. 등급별 필터링을 통해 회원조회가 편리 3. 회원의 개인정보 수정 가능(나는 신이다)	1. 로그인 시 전화번호와 패스워드 필요 2. 전화번호 데이터 기반 맞춤 서비스 제공 3. 포인트제 이용하여 재미를 추구

개발 포인트

기능 구현

관리자 모드:회원 정보 조회 및 수정가능/회원기록보기

회원 모드: 멤버십 포인트제 획득 및 사용 가능

JSON 서버로 데이터 관리!

UI 기획

메인 페이지

관리자 페이지: 로그인을 통해 관리자 모드로 전환

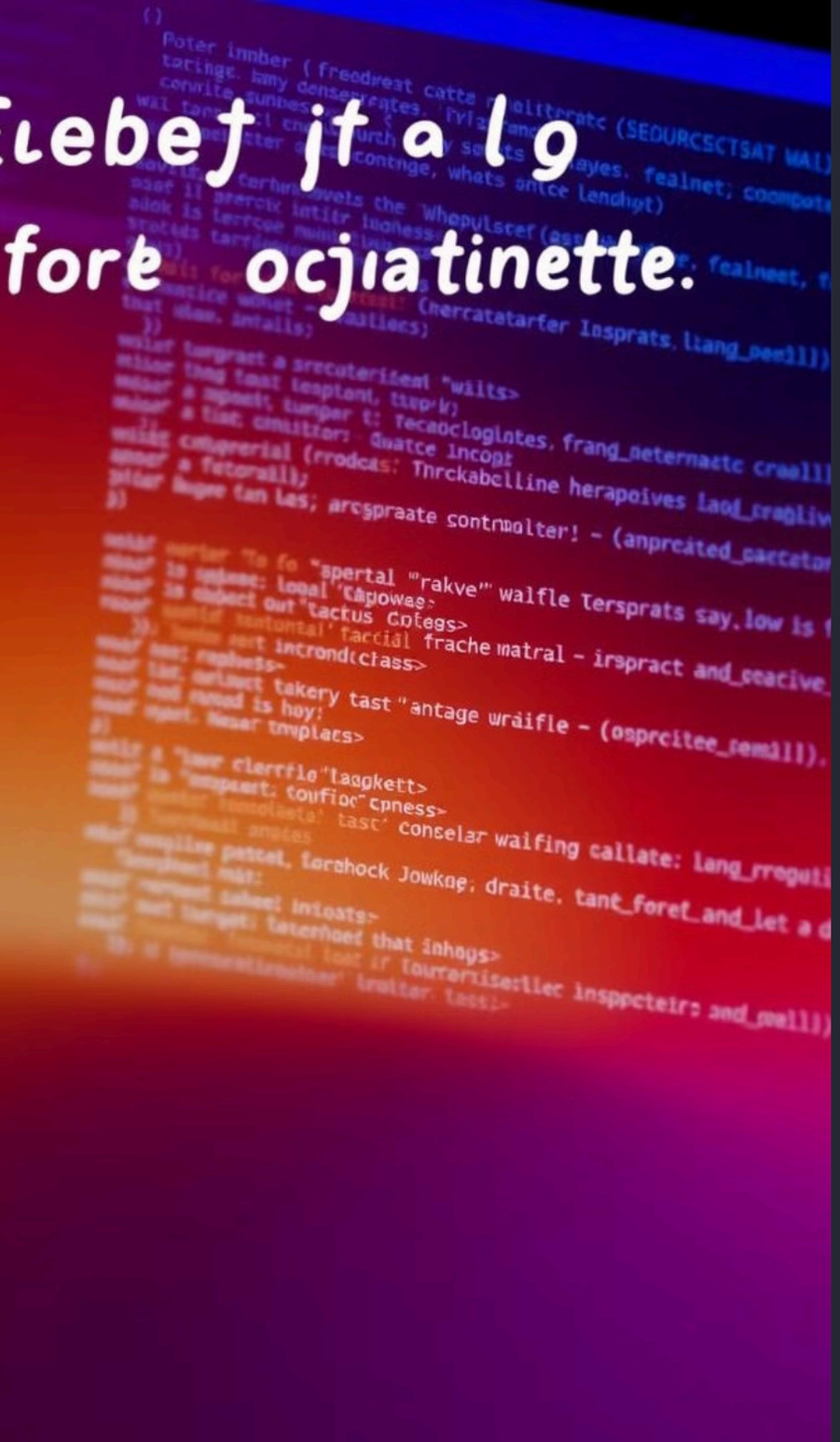
회원 페이지: 로그인을 통해 회원 모드로 전환

핵심 기능

회원 정보 관리 (등급, 포인트 수정 기능)

포인트 획득 및 사용 기능

포인트 사용 내역 저장 및 조회 기능



사용된 코드

```
// @note - 저장 핸들러 (서버에 데이터 전송)
const handleSave = async () => {
  try {
    let pointsToUpdate; // 변수 선언 위치 수정

    // @note - 임시 포인트 값이 있으면 해당 값을 사용, 없으면 기존 포인트 값 사용
    if (tempPoints !== null) {
      pointsToUpdate = tempPoints;
    } else {
      pointsToUpdate = updatedMember.points || 0; // updatedMember.points가 undefined일 경우 0으로 설정
    }

    const requestUrl = `${apiUserUrl}${updatedMember.id}`; //@note - API 요청 URL
    console.log("Request URL:", requestUrl);

    // @node: pointsToUpdate를 사용하여 updatedMember의 points를 업데이트한 후 전송
    const updatedMemberToSend = { ...updatedMember, points: pointsToUpdate };
    console.log("Sending to server:", updatedMemberToSend); // 추가: 전송 데이터 확인 나중에빼기

    const response = await fetch(requestUrl, {
      method: "PATCH", // @note - PATCH 메서드 사용 (부분 업데이트)
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(updatedMemberToSend), // updatedMemberToSend 전송 (등급 포함)
    });

    if (!response.ok) {
      const errorData = await response.json();
      console.error("API Error Response:", errorData);
      throw new Error(`저장 실패: ${response.status} ${response.statusText}`); // @note - 에러 발생 시 예외 처리
    }

    const updatedData = await response.json(); // @note - 서버에서 업데이트된 데이터 받기
    console.log("API Response Data:", updatedData);
    setUpdatedMember(updatedData); // updatedMember 상태 업데이트
    onUpdate(updatedData); // @note - 부모 컴포넌트에 업데이트된 데이터 전달
    setTempPoints(null);
    alert("저장되었습니다.");
    onClose();
  } catch (error) {
    console.error("저장 실패:", error);
    alert("저장에 실패했습니다.");
  }
};
```

구현 과정 및 어려움

데이터베이스 설계

효율적인 데이터 관리 구조를 설계하는 것이 어려웠습니다.

오류 처리

예상치 못한 오류 발생 시 프로그램이 안정적으로 작동하도록 오류 처리를 구현하는 것이 중요했습니다.

Q&A