📌 Week 1: Web Development Overview & JavaScript Fundamentals

📌 1주차: 웹 개발 개요 및 환경 세팅 및 JavaScript 기초

🎯 Learning Objectives | 강의 목표

- 웹 개발의 기본 개념 이해 (프론트엔드 & 백엔드 개념)
- HTML, CSS의 기초 개념 및 기본적인 웹 페이지 작성
- JavaScript의 기본 문법 및 데이터 타입 이해
- 변수 선언 (let, const, var) 및 기본 연산

🛠️ Exercises | 실습

✅ Understanding Web Development Basics

- 웹사이트가 어떻게 동작하는지 개념 학습 (클라이언트-서버 모델)
- 프론트엔드(HTML, CSS, JavaScript) vs. 백엔드 개념 비교

✅ Creating a Simple Static Web Page

- HTML 기본 태그(<h1>, <p>, , , <a> 등) 실습
- CSS를 활용한 간단한 스타일링 (color, font-size, margin, padding 등)

✅ JavaScript Basics

- console.log()를 사용하여 출력 테스트
- 변수 선언 및 데이터 타입 (string, number, boolean, array, object)
- 연산자 (+, -, *, /, %, ++, --) 실습

📌 Assignment | 과제

- HTML과 CSS를 활용하여 개인 프로필 웹 페이지 만들기

📌 Week 2: JavaScript Fundamentals 2 & TypeScript

📌 2주차: JavaScript 기초 2 및 TypeScript

🎯 Learning Objectives | 강의 목표

- 조건문 (if, switch), 반복문 (for, while)
- 기본 함수 및 이벤트 리스너 (addEventListener) 학습
- TypeScript 기본 문법 및 JavaScript와의 차이 이해

🛠️ Exercises | 실습

✅ Control Flow & Loops

- if-else, switch 문을 활용한 조건문 작성
- for, while 문을 사용한 반복문 연습

✅ Functions & Event Listeners

- 기본적인 함수 (function myFunction() {}) 및 화살표 함수 (() => {}) 학습
- addEventListener()를 사용하여 버튼 클릭 시 이벤트 처리

✅ Manipulating the DOM (Document Object Model)

- document.querySelector()를 사용하여 HTML 요소 선택
- innerText, innerHTML, style 속성 변경

✅ TypeScript Basics

- let vs const vs var 비교
- 기본 데이터 타입 (number, string, boolean, object, array, any, unknown)
- interface와 type을 활용한 데이터 구조 정의

📌 Assignment | 과제

- 버튼을 클릭하면 텍스트가 변경되는 간단한 웹 페이지 만들기
- for 또는 while 문을 활용하여 배열 데이터를 출력하는 코드 작성
- 사용자 입력을 받아 if-else 문을 활용해 조건에 따라 다른 메시지를 출력하는 프로그램 작성
- JS 프로그램을 TS로 바꿔보기

📌 Week 3: API Basics (Vanilla JavaScript) & Development Environment

📌 3주차: API 기초 (Vanilla JS 실습) 및 개발 환경 설정

🎯 Learning Objectives | 강의 목표

- API의 개념과 RESTful API 작동 방식 이해
- Vanilla JavaScript에서 fetch() 및 Axios를 사용한 API 요청 실습
- 개발 환경 구성 (VS Code, Node.js, Git/GitHub)
- 기본적인 CLI(Command Line Interface) 명령어 학습

🛠️ Exercises | 실습

✅ Making API Requests with fetch()

- JSONPlaceholder API에서 데이터 가져오기 (https://jsonplaceholder.typicode.com/users)
- fetch()를 사용하여 GET 요청 보내고 응답 데이터를 콘솔에 출력

✅ Handling API Responses & Errors

- try/catch 구문을 활용한 오류 처리
- async/await을 활용한 비동기 API 호출

✅ Using Axios for API Calls

- Axios 설치 및 GET, POST 요청 실습
- API 응답 데이터를 HTML에 동적으로 출력

✅ Setting Up the Development Environment

- Node.js, npm/yarn, VS Code 설치 및 설정
- Git 설치 및 GitHub 계정 생성
- 웹페이지를 브라우저에서 실행하는 방법 익히기

✅ Introduction to Git & GitHub

- git init, git add, git commit, git push 명령어 실습
- GitHub에 첫 번째 저장소 생성 및 코드 업로드

📌 Assignment | 과제

- fetch()를 사용하여 JSONPlaceholder API에서 사용자 목록을 가져와 웹페이지에 출력
- try/catch를 활용하여 API 요청 실패 시 오류 메시지 표시
- HTML 입력 폼을 만들어 사용자가 입력한 데이터를 API로 전송하는 기능 추가

Week 4: React Basics & API Integration

📌 4주차: React 기초 및 API 연동

🎯 Learning Objectives | 강의 목표

- React의 기본 개념 (컴포넌트, props, state) 이해
- React에서 API 데이터를 가져와 화면에 출력하는 방법 학습
- useState, useEffect를 활용한 상태 및 비동기 데이터 관리

🛠️ Exercises | 실습

✅ Setting Up a React Project
- create-react-app 또는 Vite를 사용하여 React 프로젝트 생성
- 프로젝트 디렉토리 구조 및 기본 설정 익히기

✅ Building a Simple Component
- props를 사용하여 컴포넌트 간 데이터 전달 실습
- useState를 활용한 간단한 상태 변경 예제 (ex: Counter App)

✅ Fetching API Data in React
- useEffect를 사용하여 API 호출 (fetch() 또는 Axios)
- JSONPlaceholder API에서 사용자 목록을 가져와 렌더링

✅ Handling API Responses & Errors in React
- API 호출 시 로딩 상태 (isLoading) 및 에러 처리 (error) 구현

📌 Assignment | 과제

- JSONPlaceholder API에서 사용자 데이터를 가져와 리스트로 출력하는 React 컴포넌트 제작
- 검색 창을 추가하여 사용자 목록을 필터링
- API 요청 중 로딩 상태 표시

📌 Week 5: Next.js Basics & Server-Side Rendering (SSR)

📌 5주차: Next.js 기초 및 서버 사이드 렌더링(SSR)

🎯 Learning Objectives | 강의 목표

- 클라이언트 사이드 렌더링(CSR)과 서버 사이드 렌더링(SSR)의 차이 이해
- Next.js의 라우팅 및 페이지 렌더링 방식 학습
- getServerSideProps()를 활용한 서버 사이드 데이터 패칭

🛠️ Exercises | 실습

✅ Setting Up a Next.js Project
- create-next-app을 사용하여 Next.js 프로젝트 생성
- 프로젝트 폴더 구조 및 주요 파일 (pages/, public/, styles/) 이해

✅ Understanding Next.js Routing
- pages/index.tsx, pages/about.tsx 등 파일 기반 라우팅 실습
- Link 컴포넌트를 활용한 클라이언트 사이드 내비게이션

✅ Fetching API Data with SSR (getServerSideProps)
- getServerSideProps()를 사용하여 서버에서 API 데이터를 가져오기
- JSONPlaceholder API에서 사용자 목록을 불러와 페이지에서 렌더링

✅ Comparing SSR with CSR (useEffect)
- 같은 API를 useEffect를 사용하여 CSR 방식으로 가져와 SSR과 비교

📌 Assignment | 과제

- 4주차에서 만든 React API 호출 컴포넌트를 Next.js 페이지에서 getServerSideProps()를 활용하여 SSR 방식으로 변환
- Next.js Link를 활용하여 페이지 간 내비게이션 구현
- ~~CSR과 SSR의 속도 차이를 비교하고 정리~~

📌 Week 6: Next.js API Routes & Backend Development

📌 6주차: Next.js API Routes 및 백엔드 개발

🎯 Learning Objectives | 강의 목표

- Next.js API Routes의 개념과 사용법 이해
- Next.js에서 백엔드 API 엔드포인트를 생성하고 처리하는 방법 학습
- Next.js API Routes를 활용한 CRUD 기능 구현

🛠️ Exercises | 실습

✅ Setting Up API Routes in Next.js

- /pages/api/hello.ts를 생성하고 API 응답 테스트
- res.json({ message: "Hello from API" })를 활용하여 JSON 응답 반환

✅ Creating a Simple API Endpoint

- /pages/api/users.ts에서 사용자 데이터 반환 API 구현
- GET 요청을 처리하여 가짜 사용자 목록 반환
- API 호출 테스트: fetch('/api/users')를 사용하여 API 데이터 가져오기

✅ Handling HTTP Methods (GET, POST, PUT, DELETE)

- API에서 POST 요청을 받아 새로운 사용자 추가
- PUT 요청을 사용하여 사용자 정보 업데이트
- DELETE 요청을 사용하여 특정 사용자 삭제

✅ Connecting API Routes to React Components

- Next.js 페이지에서 /api/users API 호출하여 데이터 표시
- useState 및 useEffect를 활용하여 사용자 목록 렌더링

📌 Assignment | 과제

- Next.js API Routes를 활용하여 간단한 CRUD API 만들기
  - GET /api/users: 사용자 목록 가져오기
  - POST /api/users: 새로운 사용자 추가
  - PUT /api/users/:id: 특정 사용자 정보 수정
  - DELETE /api/users/:id: 특정 사용자 삭제
- Next.js 페이지에서 CRUD 기능을 연동하여 UI에서 사용자 추가/수정/삭제 기능 구현

📌 Example Code: Simple Next.js API Route (/pages/api/users.ts)

JavaScript

```
import { NextApiRequest, NextApiResponse } from "next";

let users = [
  { id: 1, name: "Alice" },
  { id: 2, name: "Bob" },
];

export default function handler(req: NextApiRequest, res: NextApiResponse) {
  if (req.method === "GET") {
    res.status(200).json(users);
```

```
  } else if (req.method === "POST") {
    const newUser = { id: users.length + 1, name: req.body.name };
    users.push(newUser);
    res.status(201).json(newUser);
  } else if (req.method === "DELETE") {
    const { id } = req.query;
    users = users.filter((user) => user.id !== Number(id));
    res.status(200).json({ message: "User deleted" });
  } else {
    res.status(405).json({ message: "Method Not Allowed" });
  }
}
```

📌 Week 7: Database Integration with Prisma & Supabase

📌 7주차: Prisma 및 Supabase를 활용한 데이터베이스 연동

🎯 Learning Objectives | 강의 목표

- Supabase 개념 이해 및 프로젝트 생성
- Prisma를 활용한 Next.js에서의 데이터베이스 연동
- Supabase API를 활용한 데이터 읽기/쓰기
- Next.js API Routes에서 Supabase를 사용한 CRUD 구현

🛠️ Exercises | 실습

✅ Setting Up Supabase & Connecting Prisma

- Supabase 회원가입 및 새로운 프로젝트 생성
- Supabase에서 데이터베이스 테이블(User) 생성
- @supabase/supabase-js 패키지를 Next.js 프로젝트에 설치
- Prisma 설치 및 prisma/schema.prisma 설정

✅ Creating a User Model in Prisma

- Supabase와 연결된 Prisma 모델 정의
- DATABASE_URL을 Supabase에서 제공하는 PostgreSQL URL로 설정
- npx prisma migrate dev 실행하여 테이블 생성

```
Unset
generator client {
  provider = "prisma-client-js"
}
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
model User {
  id    Int     @id @default(autoincrement())
  name  String
  email String  @unique
}
```

✅ Connecting API Routes to Supabase

- /api/users.ts API Routes에서 Supabase를 사용하여 데이터 조회
- GET, POST, PUT, DELETE 요청을 처리하는 API 구현

✅ Fetching and Displaying Data in Next.js

- useEffect()를 사용하여 Supabase API에서 데이터를 가져와 화면에 출력

📌 Assignment | 과제

- Supabase 테이블 추가: posts 테이블을 만들고, 사용자와 연결
- Next.js 페이지에서 CRUD 연동: Supabase API를 활용하여 사용자 추가/삭제 기능 구현
- API Routes에서 Prisma & Supabase 동시 사용 실습: PrismaClient를 사용하여 Supabase 데이터 관리

📌 Week 8: Authentication & Role-Based Access Control (RBAC)

📌 8주차: NextAuth.js 및 Supabase를 활용한 사용자 인증 및 역할 기반 접근 제어 (RBAC)

🎯 Learning Objectives | 강의 목표

- Supabase와 NextAuth.js를 활용하여 OAuth 로그인 (Google, GitHub) 구현
- Supabase를 활용하여 이메일 & 비밀번호 로그인 구현
- Supabase 테이블에 role 컬럼을 추가하고 RBAC(역할 기반 접근 제어) 적용
- 관리자(Admin)와 일반 사용자(User) 역할을 구분하여 접근 권한 제어

🛠️ Exercises | 실습

✅ Setting Up OAuth Login (Google, GitHub) in NextAuth.js

- next-auth 및 @supabase/supabase-js 패키지 설치
- pages/api/auth/[...nextauth].ts에서 Google, GitHub OAuth Provider 설정
- 로그인 후 useSession()을 사용하여 사용자 정보 가져오기

✅ Setting Up Email & Password Authentication in Supabase

- Supabase 대시보드에서 Email & Password 인증 활성화
- signUp() 및 signInWithPassword() 메서드를 활용하여 로그인 구현
- NextAuth.js의 CredentialsProvider를 활용하여 이메일 & 비밀번호 로그인 추가

✅ Implementing Role-Based Access Control (RBAC) in Supabase

- Supabase users 테이블에 role 컬럼 추가 (예: admin, user)
- middleware.ts를 사용하여 특정 역할이 필요한 페이지 보호
- 관리자만 접근할 수 있는 Admin Dashboard 페이지 구현

✅ Setting Up Supabase Row-Level Security (RLS)

- Supabase에서 Row-Level Security(RLS) 정책을 활성화
- 특정 사용자는 본인의 데이터만 조회할 수 있도록 제한 (auth.uid() 사용)
- 관리자(role = 'admin')는 모든 데이터를 볼 수 있도록 정책 추가

📌 Assignment | 과제

- Google OAuth 및 Kakao Talk OAuth 로그인을 구현하고 사용자 정보를 Supabase에 저장
- Supabase에서 이메일 & 비밀번호 로그인 구현하고 NextAuth.js CredentialsProvider로 연결
- role 기반 접근 제어를 적용하여 일반 사용자와 관리자 권한을 분리
- Supabase RLS 정책을 설정하여 사용자가 본인의 데이터만 조회하도록 제한

📌 Example Code: NextAuth.js API Route (`/pages/api/auth/[...nextauth].ts`)

```javascript
import NextAuth from "next-auth";
import GoogleProvider from "next-auth/providers/google";
import KakaoProvider from "next-auth/providers/kakao";
import CredentialsProvider from "next-auth/providers/credentials";
```

```
import { createClient } from "@supabase/supabase-js";

const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL!,
  process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!
);

export default NextAuth({
  providers: [
    GoogleProvider({
      clientId: process.env.GOOGLE_CLIENT_ID!,
      clientSecret: process.env.GOOGLE_CLIENT_SECRET!,
    }),
    KakaoProvider({
      clientId: process.env.KAKAO_CLIENT_ID!,
      clientSecret: process.env.KAKAO_CLIENT_SECRET!,
    }),
    CredentialsProvider({
      name: "Email & Password",
      credentials: {
        email: { label: "Email", type: "text" },
        password: { label: "Password", type: "password" },
      },
      async authorize(credentials) {
        const { email, password } = credentials;
        const { data, error } = await supabase.auth.signInWithPassword({
          email,
          password,
        });

        if (error) throw new Error(error.message);
        return data.user;
      },
    }),
  ],
  callbacks: {
    async session({ session, user }) {
      const { data } = await supabase.from("users").select("role").eq("email",
session.user.email);
      session.user.role = data?.[0]?.role || "user";
      return session;
    },
  },
});
```

- Google OAuth를 사용하여 로그인 구현
- 로그인한 사용자의 정보를 Supabase DB에 저장

📌 Week 9: Advanced API Handling & Data Fetching

📌 9주차: 고급 API 핸들링 및 데이터 패칭 최적화

🎯 Learning Objectives | 강의 목표

- Next.js에서 서버 사이드 데이터 패칭(SSR, SSG) 최적화
- Supabase API와 Next.js의 useSWR() 또는 React Query를 활용한 캐싱 및 성능 최적화
- 무한 스크롤(Infinite Scroll) 및 페이지네이션(Pagination) 구현
- API 요청 최적화(중복 요청 방지, Debouncing, Prefetching)

🛠️ Exercises | 실습

✅ Optimizing Data Fetching with SSR & SSG
- getServerSideProps() vs getStaticProps() 비교 및 활용
- revalidate를 활용하여 정적 페이지 데이터 업데이트

✅ Using SWR or React Query for API Caching
- useSWR()를 활용한 Supabase API 데이터 캐싱
- stale-while-revalidate 패턴을 사용하여 API 요청 최적화

✅ Implementing Infinite Scroll & Pagination
- 프론트엔드 무한 스크롤: IntersectionObserver API를 활용하여 데이터 자동 로딩
- 서버 사이드 페이지네이션: Supabase range() 메서드를 사용하여 페이지별 데이터 가져오기

✅ Debouncing & Prefetching for Better UX
- debounce를 활용하여 검색 입력 시 API 호출 최적화
- prefetch를 활용하여 사용자가 방문할 가능성이 높은 데이터 미리 로드

📌 Assignment | 과제

- Next.js의 getServerSideProps()와 getStaticProps()를 비교하고 최적화 적용
- SWR을 활용하여 Supabase API 데이터를 캐싱하고, 새로고침 없이 업데이트 반영
- 무한 스크롤 및 페이지네이션을 구현하여 대량 데이터 처리 최적화
- Debouncing을 활용하여 검색 입력 시 API 요청을 최소화

📌 Example Code: Using SWR for API Caching

JavaScript

```javascript
import useSWR from "swr";
import { createClient } from "@supabase/supabase-js";

const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL!,
  process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!
);

const fetcher = async () => {
  const { data } = await supabase.from("users").select("*");
  return data;
};
```

```
export default function UserList() {
  const { data: users, error } = useSWR("/api/users", fetcher, { revalidateOnFocus: false });

  if (error) return <div>Failed to load users</div>;
  if (!users) return <div>Loading...</div>;

  return (
    <ul>
      {users.map((user) => (
        <li key={user.id}>{user.name}</li>
      ))}
    </ul>
  );
}
```

📌 Example Code: Infinite Scroll with Intersection Observer

JavaScript

```
import { useEffect, useState, useRef } from "react";
import { createClient } from "@supabase/supabase-js";

const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL!,
  process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!
);

export default function InfiniteScrollUsers() {
  const [users, setUsers] = useState([]);
  const [page, setPage] = useState(0);
  const observer = useRef();

  useEffect(() => {
    async function loadUsers() {
      const { data } = await supabase
        .from("users")
        .select("*")
        .range(page * 10, (page + 1) * 10 - 1);
      setUsers((prev) => [...prev, ...data]);
    }
    loadUsers();
  }, [page]);

  const lastUserRef = (node) => {
    if (observer.current) observer.current.disconnect();
    observer.current = new IntersectionObserver((entries) => {
      if (entries[0].isIntersecting) {
        setPage((prev) => prev + 1);
      }
```

```
    });
    if (node) observer.current.observe(node);
  };

  return (
    <ul>
      {users.map((user, index) => (
        <li key={user.id} ref={index === users.length - 1 ? lastUserRef : null}>
          {user.name}
        </li>
      ))}
    </ul>
  );
}
```

📌 Week 10: Deployment & Performance Optimization

📌 10주차: 배포 및 성능 최적화

🎯 Learning Objectives | 강의 목표

- Vercel을 활용한 Next.js 배포 자동화 및 환경 변수 설정
- Supabase와 Vercel을 연결하여 프로덕션 환경에서 데이터 관리
- Next.js의 이미지 최적화, 코드 스플리팅, Lazy Loading 적용
- API 응답 속도 최적화 및 Lighthouse 성능 개선

🛠️ Exercises | 실습

✅ Deploying Next.js to Vercel

- Vercel CLI 설치 및 프로젝트 연결 (vercel login, vercel link)
- Vercel에서 GitHub 연동을 통한 자동 배포 설정
- vercel env를 활용한 환경 변수 설정 (NEXT_PUBLIC_SUPABASE_URL, NEXT_PUBLIC_SUPABASE_ANON_KEY)

✅ Connecting Supabase with Vercel

- Supabase 프로덕션 데이터베이스와 연결하여 API 작동 테스트
- prisma migrate deploy를 사용하여 Supabase DB 마이그레이션 적용

✅ Optimizing Next.js Performance

- Next.js의 next/image를 활용한 이미지 최적화
- getStaticProps()를 활용한 정적 데이터 생성 (SSG) 최적화
- Dynamic Import를 활용하여 코드 스플리팅 적용 (next/dynamic)

✅ Improving API Response Time & Lighthouse Score

- API 요청 최적화를 위한 gzip 압축, HTTP/2 적용
- Lighthouse(구글 웹 성능 분석 도구)를 사용하여 성능 분석
- 불필요한 API 호출 제거 및 로딩 속도 개선

📌 Assignment | 과제

- Next.js 프로젝트를 Vercel에 배포하고 Supabase 프로덕션 데이터베이스와 연결
- next/image 및 next/dynamic을 사용하여 프론트엔드 성능 최적화
- Lighthouse를 사용하여 성능 분석 후 최적화 조치 수행 (Lazy Loading, 코드 스플리팅 등)
- API 응답 속도 개선을 위해 revalidate, 캐싱 및 프리페칭 적용

📌 Example Code: Optimizing Images with Next.js (`next/image`)

JavaScript

```
import Image from "next/image";

export default function Profile() {
  return (
    <div>
      <h1>User Profile</h1>
      <Image
```

```
        src="/profile.jpg"
        alt="Profile Picture"
        width={200}
        height={200}
        priority
      />
    </div>
  );
}
```

- `next/image`를 사용하면 자동 압축, 최적화, Lazy Loading 적용

📌 Example Code: Using Dynamic Imports for Code Splitting

JavaScript

```
import dynamic from "next/dynamic";

const HeavyComponent = dynamic(() => import("../components/HeavyComponent"), {
  loading: () => <p>Loading...</p>,
  ssr: false,
});

export default function Home() {
  return (
    <div>
      <h1>Home Page</h1>
      <HeavyComponent />
    </div>
  );
}
```

- next/dynamic을 사용하여 무거운 컴포넌트는 필요할 때만 로드
- `ssr: false` 옵션을 추가하면 서버 사이드 렌더링을 비활성화하여 클라이언트 사이드에서만 로드

📌 Example Code: Vercel & Supabase Deployment (`vercel.json`)

JavaScript

📌 Week 11–12: Full-Stack Project Development (Church Visitor Management App)

📌 11~12주차: 교회 방문자 정보 관리 앱 개발 프로젝트

🎯 Learning Objectives | 프로젝트 목표

- Next.js와 Supabase를 활용하여 교회 방문자 정보 저장 및 조회 기능 구현
- 인증 기능(NextAuth.js)과 역할 기반 접근 제어(RBAC)를 적용하여 관리자만 데이터 관리 가능하도록 설정
- 데이터 패칭 최적화(SWR, React Query) 및 API 성능 개선
- Vercel을 활용한 프로젝트 배포 및 성능 최적화

📌 Week 11: Project Kickoff & Development

📌 11주차: 프로젝트 기획 및 개발 시작

🛠️ Session 1: Project Introduction & Planning

✅ 프로젝트 개요 설명 (교회 방문자 정보 관리 앱)
✅ 요구 사항 정리 및 기능 정의
✅ ERD(Entity Relationship Diagram) 설계 및 Supabase 테이블 생성
✅ UI/UX 디자인 구상 및 Figma 등으로 와이어프레임 작성

🛠️ Development Phase (10 Days)

✅ Next.js 프로젝트 셋업 및 환경 변수 설정
✅ Supabase 데이터베이스 모델링 (방문자 정보 저장 테이블)
✅ NextAuth.js를 활용한 사용자 인증 및 관리자 RBAC 설정
✅ CRUD API 구현: 방문자 정보 저장(Create), 조회(Read), 수정(Update), 삭제(Delete)

📌 Week 12: Project Completion & Presentation

📌 12주차: 프로젝트 마무리 및 발표

🛠️ Final Development & Testing

✅ 데이터 패칭 최적화 (useSWR(), React Query)
✅ Supabase Row-Level Security 설정 검토 및 수정
✅ API 응답 최적화 (Debouncing, Prefetching)
✅ Lighthouse 테스트 및 최적화

🛠️ Session 2: Project Presentation & Review

✅ 프로젝트 시연 및 발표 (각자 만든 기능 설명)
✅ 코드 리뷰 및 피드백 제공
✅ 보완할 부분 개선 및 추가 기능 논의

📌 Summary | 프로젝트 요약

✅ Next.js + Supabase 기반 CRUD 기능 구현 (방문자 정보 저장 및 조회)
✅ NextAuth.js를 활용한 인증 및 역할 기반 접근 제어(RBAC)
✅ SWR/React Query를 활용한 API 데이터 캐싱 및 최적화
✅ Vercel을 활용한 배포 및 성능 테스트 진행