

웹 애플리케이션 개발 프로젝트

추가 적용할만한 정보

✓ 한글 폰트 세팅

- index.html 수정
 - Google Fonts -> Noto Sans Korean
 - Regular 400 사이즈



```
<head>
  <meta charset="UTF-8" />
  <link rel="icon" href="/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
    rel="stylesheet"
    integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSm07SnpJef0486qhLnuZ2cdeRh002iuK6FUUVM"
    crossorigin="anonymous"
  />
  <link rel="preconnect" href="https://fonts.googleapis.com" />
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
  <link
    href="https://fonts.googleapis.com/css2?family=Noto+Sans+KR&display=swap"
    rel="stylesheet"
  />
  <title>Vite App</title>
</head>
```

추가 적용할만한 정보

✓ Google Fonts Icons 세팅

- https://developers.google.com/fonts/docs/material_symbols?hl=ko
- CDN 추가


머티리얼 기호 사용

웹에서 사용

머티리얼 기호 글꼴은 머티리얼 기호를 웹 프로젝트에 통합하는 가장 쉬운 방법입니다.

아이콘은 단일 글꼴로 패키징되므로 웹 개발자가 코드 몇 줄만으로 이러한 아이콘을 쉽게 통합할 수 있습니다.

Google Fonts의 정적 글꼴

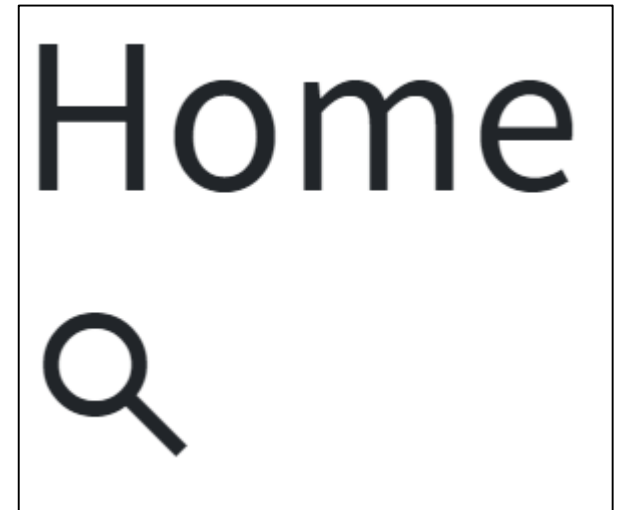
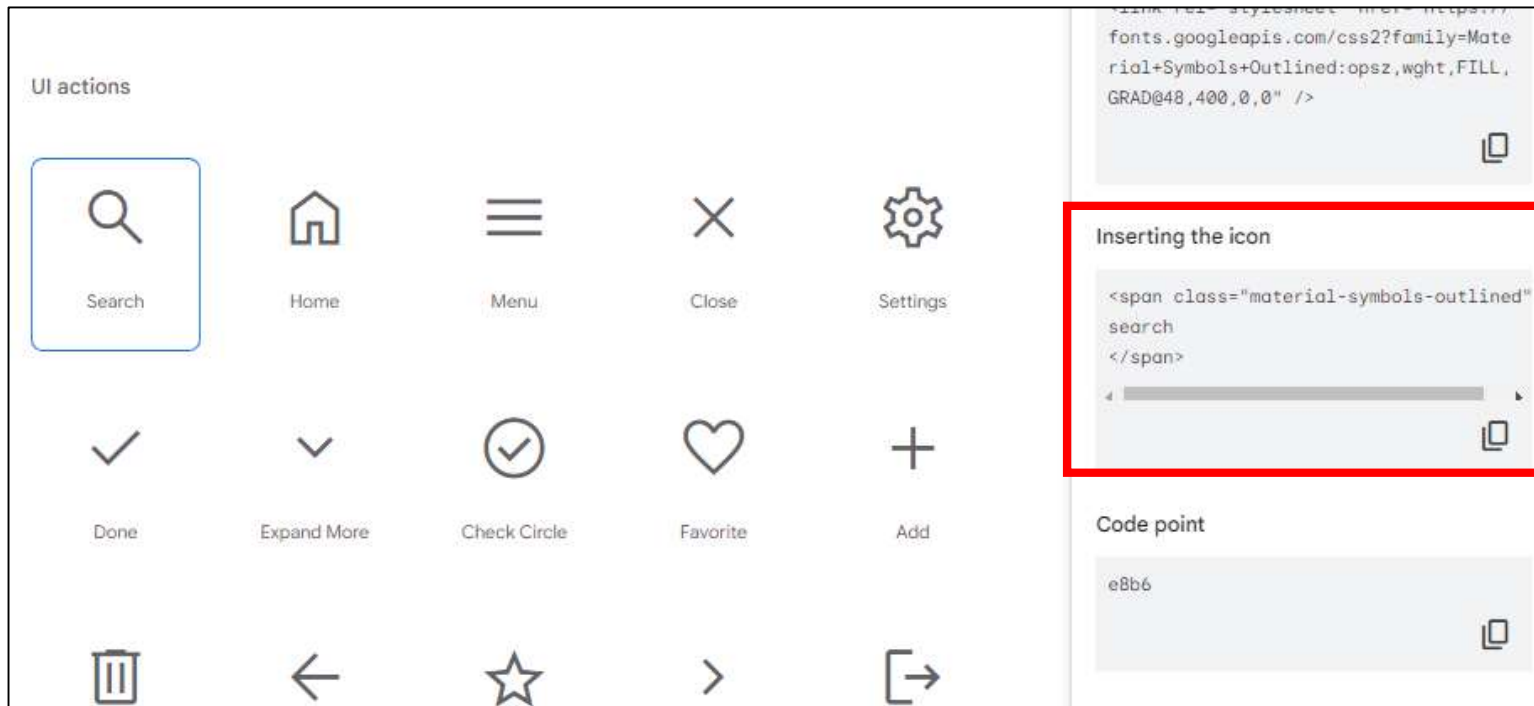
웹페이지에서 사용할 아이콘 글꼴을 설정하는 가장 쉬운 방법은 [Google Fonts](#) 를 사용하는 것입니다. 다음 HTML 한 줄을 포함합니다.

```
<link href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined" rel="stylesheet" />
```



추가 적용할만한 정보

- ✓ Google Fonts Icons 세팅
 - <https://fonts.google.com/icons>
 - 접속 후, 원하는 아이콘 선택해 테스트



추가 적용할만한 정보

src/assets/main.css 수정

- 모든 태그의 폰트 Noto Sans KR 로 변경
- , 사용 시, 맨 앞에 붙는 . 기호와 패딩값 삭제



```
1  * {  
2    font-family: "Noto Sans KR", sans-serif;  
3    list-style: none;  
4  }  
5  ul {  
6    padding-left: 0;  
7  }
```

부품관리 시스템 개요

- ✓ 차량 부품 관리 웹 기반 시스템은 차량에 사용되는 다양한 부품 정보를 등록, 조회, 삭제할 수 있는 효율적이고 직관적인 관리 플랫폼입니다. 부품 정보의 중복 등록을 방지하고, 체계적인 데이터 관리를 통해 차량 유지보수의 효율성과 신뢰성을 높입니다.

부품관리 시스템 개요

✓ 목표

- 차량에 사용되는 부품 정보를 등록·조회·삭제할 수 있는 웹 기반 관리 시스템 구축

✓ 핵심 기능

- 부품 등록 (이름/가격)
- 부품 목록 조회
- 부품 삭제
- 예외 처리 (중복 부품 등록 금지)

DB 모델링

- ✓ 테이블명 : part
- ✓ 컬럼구성
 - id: PK, 자동 증가
 - name: 부품명 (중복 불가)
 - price: 가격 (정수형)
 - created_at: 생성일시 (자동 기록)
 - updated_at: 수정일시 (자동 갱신)

Method	Endpoint	기능
GET	<code>/api/v1/parts</code>	부품 전체 조회
GET	<code>/api/v1/parts/add</code>	부품 등록
GET	<code>/api/v1/parts/delete/{id}</code>	부품 삭제

백엔드 API 설계

- ✓ 에러 처리
 - 동일 이름 등록 시 예외 발생 (IllegalStateException)
- ✓ 응답 구조
 - 등록: { id: Long }
 - 삭제: { id: Long }
 - 조회: [{ id, name, price }]

라우팅 구조

경로	설명
/parts	부품 목록 페이지
/parts/add	등록 폼 (모달 또는 페이지)

컴포넌트 구조

컴포넌트	설명	포함 관계
PartListPage	부품 목록 페이지의 메인 컴포넌트	루트 컴포넌트
└ PartTable	부품 데이터를 테이블로 표시	PartListPage의 하위
└ AddPartModal	부품 추가를 위한 모달 폼	PartListPage의 하위 (모달 또는 페이지)
└ DeleteConfirmDialog	삭제 확인 다이얼로그	PartListPage의 하위 (다이얼로그)

- ✓ 여러분은 '**스마트 차량 대시보드 모니터링 시스템**' 개발자가 됩니다.
- ✓ 이 프로젝트의 목표는 실제 차량(또는 가상의 차량 데이터)에서 발생하는 다양한 정보를 수집하여, 운전자나 관리자가 한눈에 차량 상태를 파악하고, 문제 발생 시 즉각적으로 인지할 수 있도록 **직관적이고 사용자 친화적인 대시보드 웹 애플리케이션**을 구현하는 것입니다.
- ✓ 아래의 DB모델 설계와 API는 기존의 프로젝트입니다.
- ✓ 팀별로 자유롭게 주제안에서 **변경**하시면 됩니다

팀 구성 및 가이드라인 수립

이름	역할	담당 영역
홍길동	팀장	일정 관리, 코드 리뷰, 통합
김영희	프론트엔드	UI/UX 구현, React/Vue
이철수	백엔드	API 개발, DB 설계

팀 구성 및 가이드라인 수립

- ✓ 일정 및 협업 방식
 - 주간 스크럼: 매주 월.금 온라인 회의 (10~15분)
 - GitHub 관리: 모든 작업은 PR → 리뷰 → 병합
- ✓ 브랜치 전략
 - main: 배포용 (절대 직접 push 금지)
 - dev: 통합 개발 브랜치
 - feature/기능명: 기능 단위 작업 브랜치
 - 예: feature/create-post

팀 구성 및 가이드라인 수립

✓ 커뮤니케이션 툴

항목	도구
코드 관리	GitHub
실시간 소통	Discord / Slack
회의 기록	Notion / Google Docs
일정 공유	Google Calendar

팀 구성 및 가이드라인 수립

✓ 커밋 컨벤션

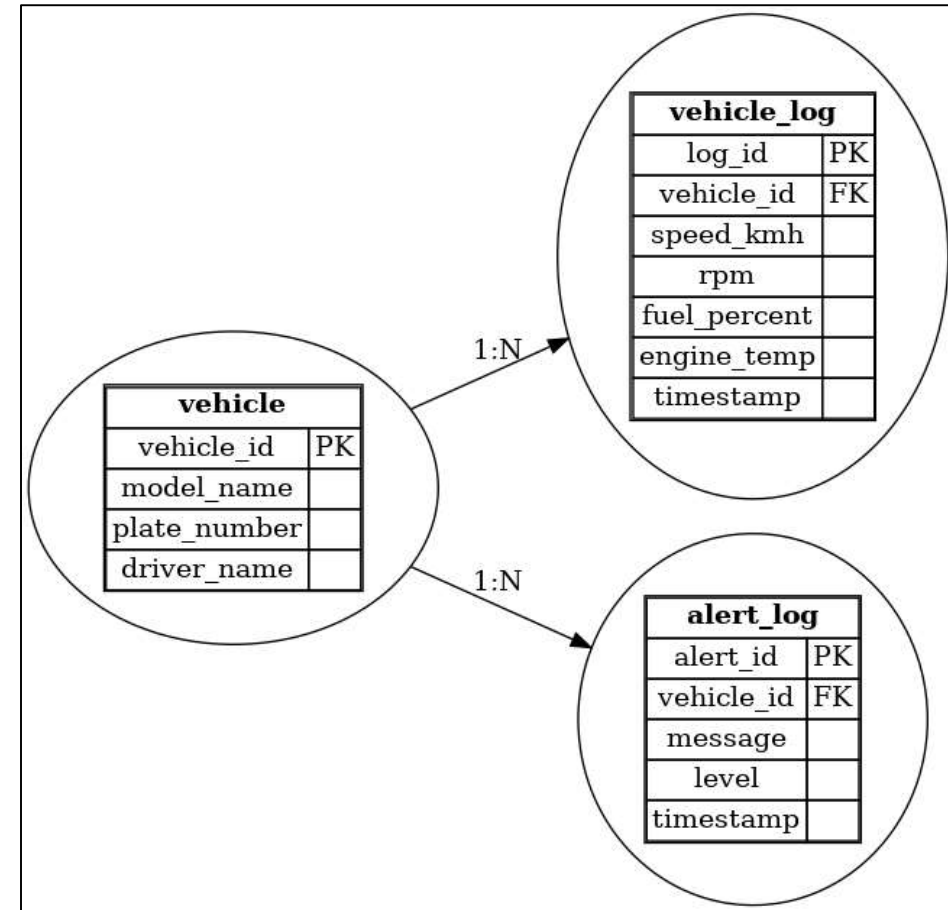
- 형식: 타입: 설명
 - 예시:
 - feat: 게시글 작성 API 추가
 - fix: 게시글 수정 시 예외 처리
 - 타입 종류:
 - feat: 새로운 기능
 - fix: 버그 수정
 - refactor: 리팩토링
 - docs: 문서 수정
 - style: 스타일/포맷

조건

- ✓ 수업에 배운 기술 스택을 활용 + 추가 스택 적용 가능

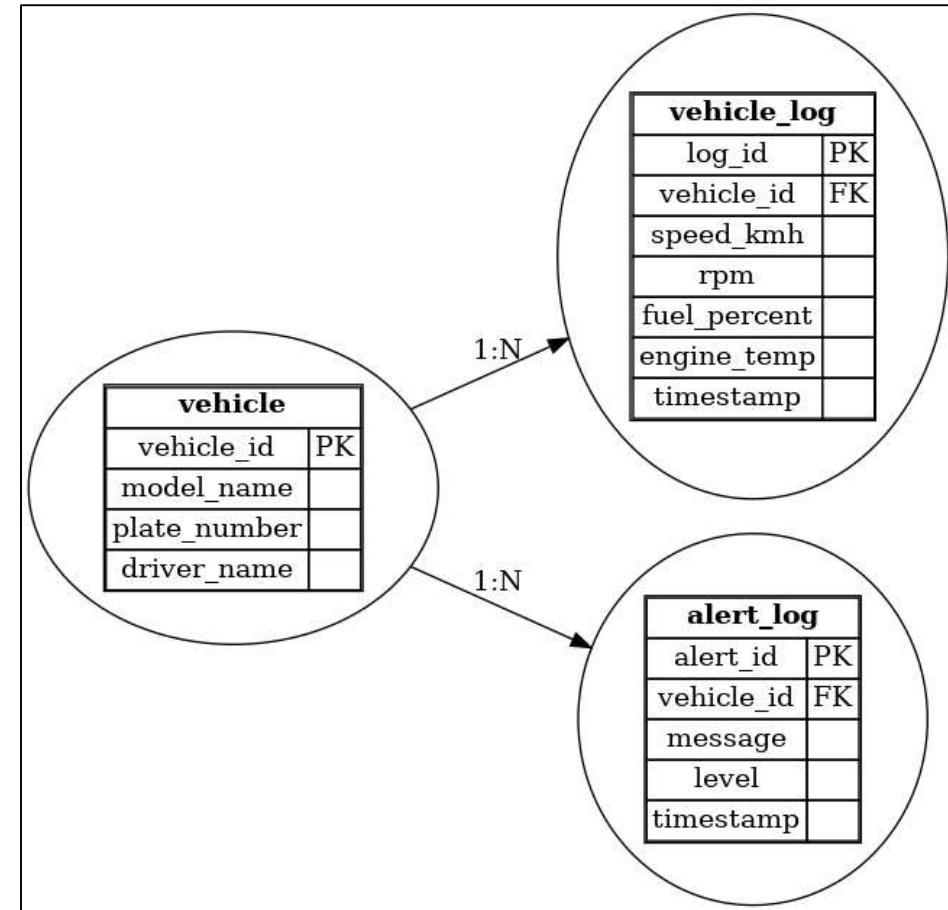
✓ Vehicle Table

- vehicle_id: 기본 키(PK)로, 각 차량을 고유하게 식별
 - PK (Primary Key)
- model_name: 차량 모델을 나타냄
 - VARCHAR
- plate_number: 차량 번호판을 나타냄
 - VARCHAR
- driver_name: 운전자 이름을 나타냄
 - VARCHAR



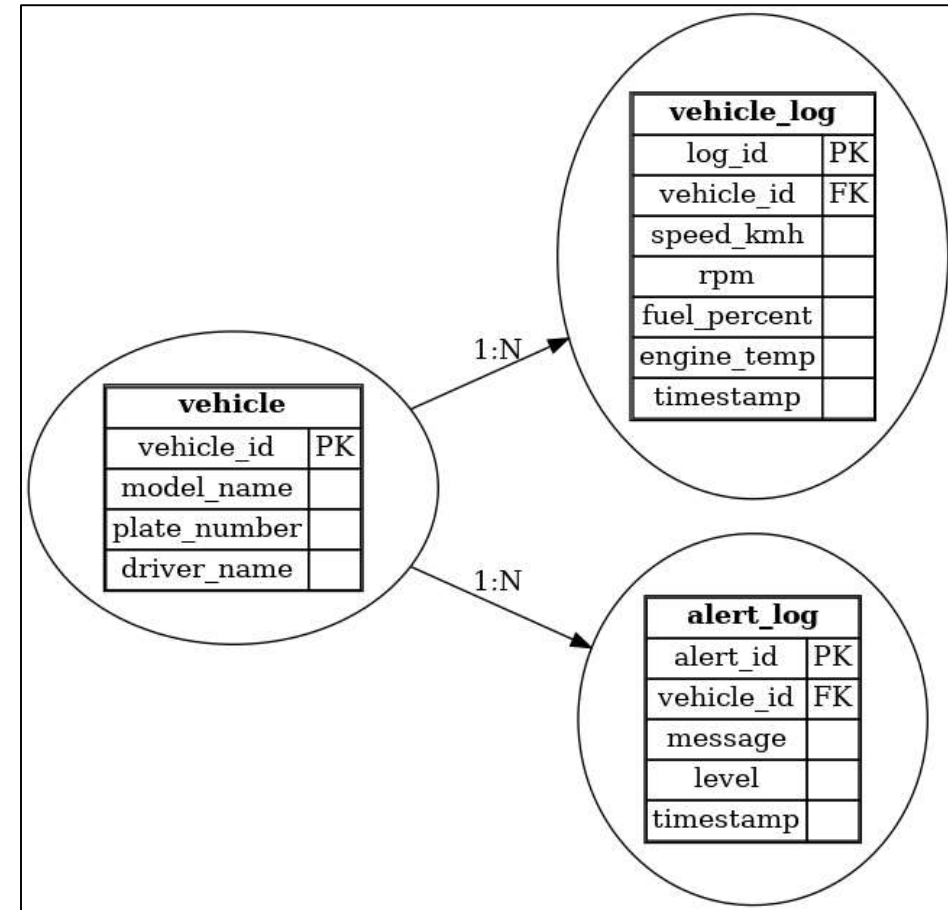
✓ Vehicle_Log Table

- log_id: 기본 키(PK)로, 각 로그 항목을 고유하게 식별
 - PK
- vehicle_id: 외래 키(FK)로, Vehicle 테이블의 vehicle_id를 참조
 - FK (Foreign Key)
- speed_kmh: 차량의 속도를 km/h 단위로 나타냄
 - INT/FLOAT
- rpm: 엔진 회전수를 나타냄
 - INT
- fuel_percent: 연료 잔량을 퍼센트로 나타냄
 - FLOAT
- engine_temp: 엔진 온도를 나타냄
 - FLOAT
- timestamp: 로그가 기록된 시간을 나타냄
 - DATETIME



✓ Alert_Log Table

- ✓ alert_id: 기본 키(PK)로, 각 알림 로그 항목을 고유하게 식별
 - ✓ PK
- ✓ vehicle_id: 외래 키(FK)로, Vehicle 테이블의 vehicle_id를 참조
 - ✓ FK (Foreign Key)
- ✓ message: 알림 메시지를 나타냅니다.
 - ✓ VARCHAR
- ✓ level: 알림의 심각도(INFO/WARN/CRITICAL)를 나타냅니다.
 - ✓ ENUM
- ✓ timestamp: 알림이 기록된 시간을 나타냅니다.
 - ✓ DATETIME



HTTP	URL	설명
GET	/api/vehicles	등록된 차량 목록 조회
GET	/api/vehicles/{id}	특정 차량 기본 정보 조회
GET	/api/vehicles/{id}/logs?start=xxx&end=xxx	특정 차량의 센서 로그 조회
GET	/api/vehicles/{id}/alerts	특정 차량의 알람 로그 조회
POST	/api/vehicles/{id}/logs	(선택) 센서 데이터 수신용 API
POST	/api/vehicles	차량 등록
GET	/api/dashboard/summary	전체 차량 상태 요약 (통계용)

컴포넌트 구조 및 라우터 설계

카테고리	파일명/컴포넌트	주요 역할 및 설명
페이지 (/pages)	DashboardPage.vue	전체 차량 실시간 현황(대시보드) 시각화
	VehicleDetailPage.vue	특정 차량의 상세 로그 및 알림 그래프
	AlertLogPage.vue	전체 차량 알림 로그(리스트 및 필터)
	AddVehiclePage.vue	새로운 차량 추가 화면 (차량 등록 기능)
	NotFoundPage.vue	존재하지 않는 경로 접근 시 표시
차트 컴포넌트	charts/SpeedLineChart.vue	차량 속도(시계열) 라인 차트
	charts/FuelGaugeChart.vue	연료 잔량 게이지 차트
	charts/TempBarChart.vue	엔진 온도 막대 차트
	charts/AlertPieChart.vue	알림/이상 유형별 비율 파이 차트
레이아웃	layout/AppHeader.vue	상단 헤더(로고, 네비게이션 등 공통 UI)
	layout/AppSidebar.vue	좌측(또는 우측) 사이드바(메뉴/네비)
공통 컴포넌트	common/VehicleCard.vue	차량 카드(리스트/간략 정보용)
	common/AlertItem.vue	알림/이상 이벤트 단건 표시 카드
	common/LogTable.vue	각종 로그 테이블(운행, 센서, 알림 등)