

자동차리콜 로지스터회귀분석

주제에 따른 데이터 수집 방법과 분석, 데이터 변환 및 저장, 정제된 데이터 시각화 처리

발표일 : **2025.05.26**

4차프로젝트 : Recall_final

목차

Contents

빅데이터검색 플랫폼 프로젝트
Logistic Regression

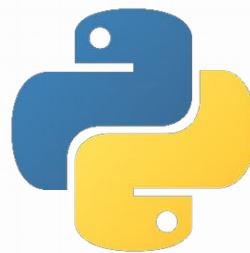
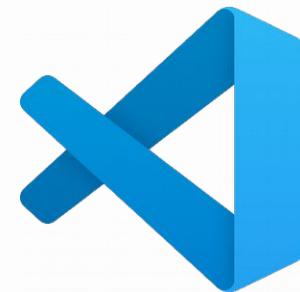
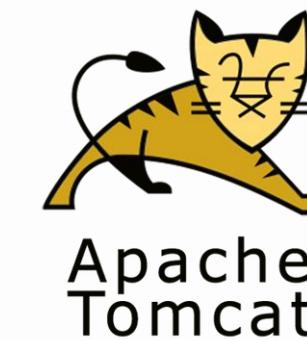
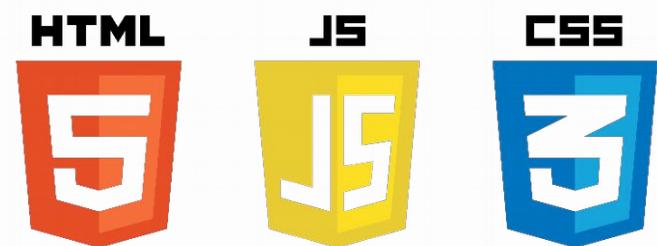
- 01 프로젝트 개발환경
- 02 개발 문서
- 03 팀 구성 및 역할
- 04 프로젝트 일정 및 설계
- 05 주소, 화면시연 , 로직
- 06 수행결과 및 자체평가

개발 환경

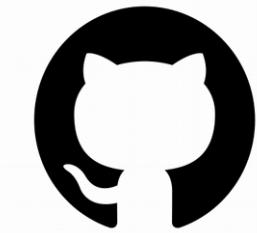
Project Overview

빅데이터검색 플랫폼 프로젝트

Logistic Regression



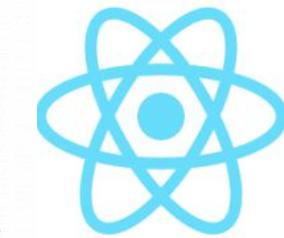
형상관리



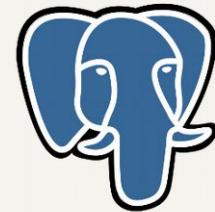
협업툴



프론트엔드



배포



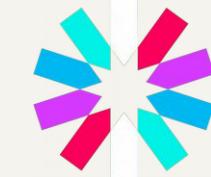
PostgreSQL



render



인증처리



JWT

토큰사용.

API 테스트



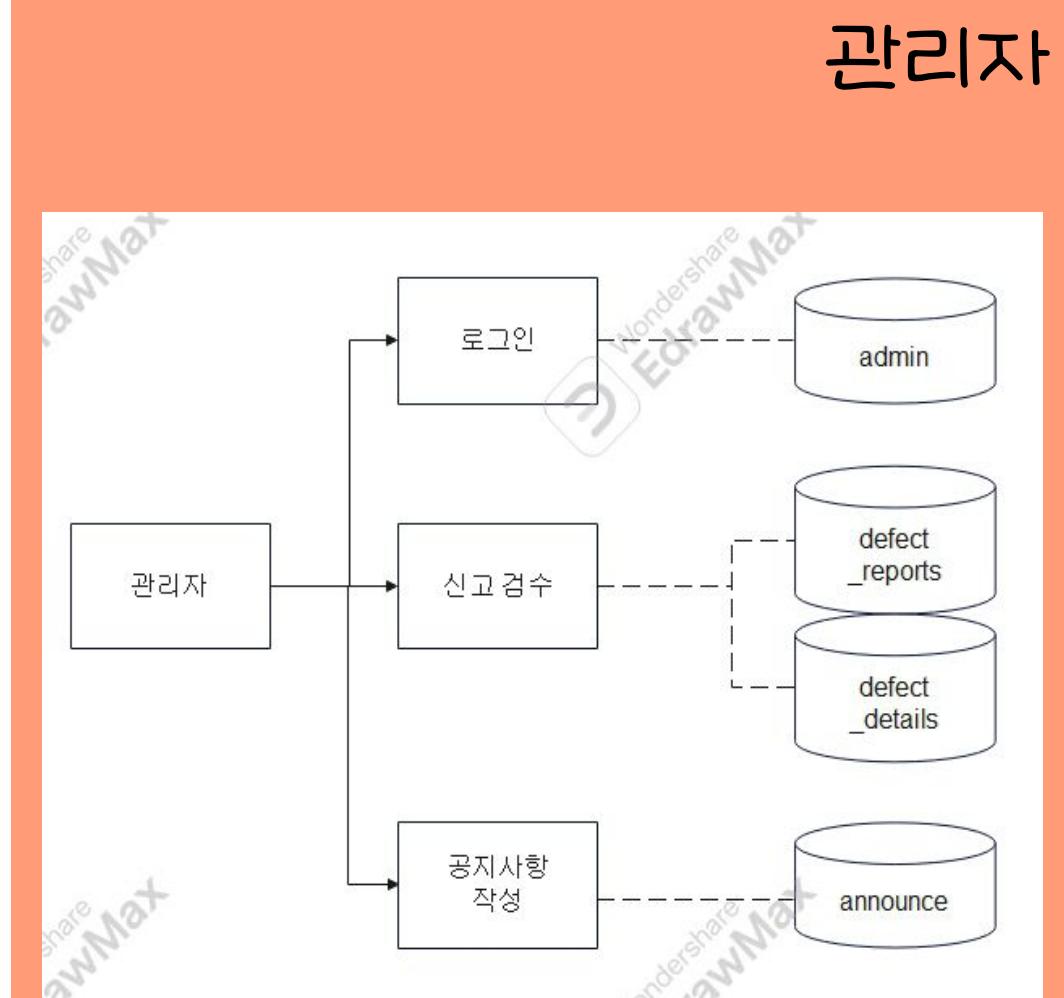
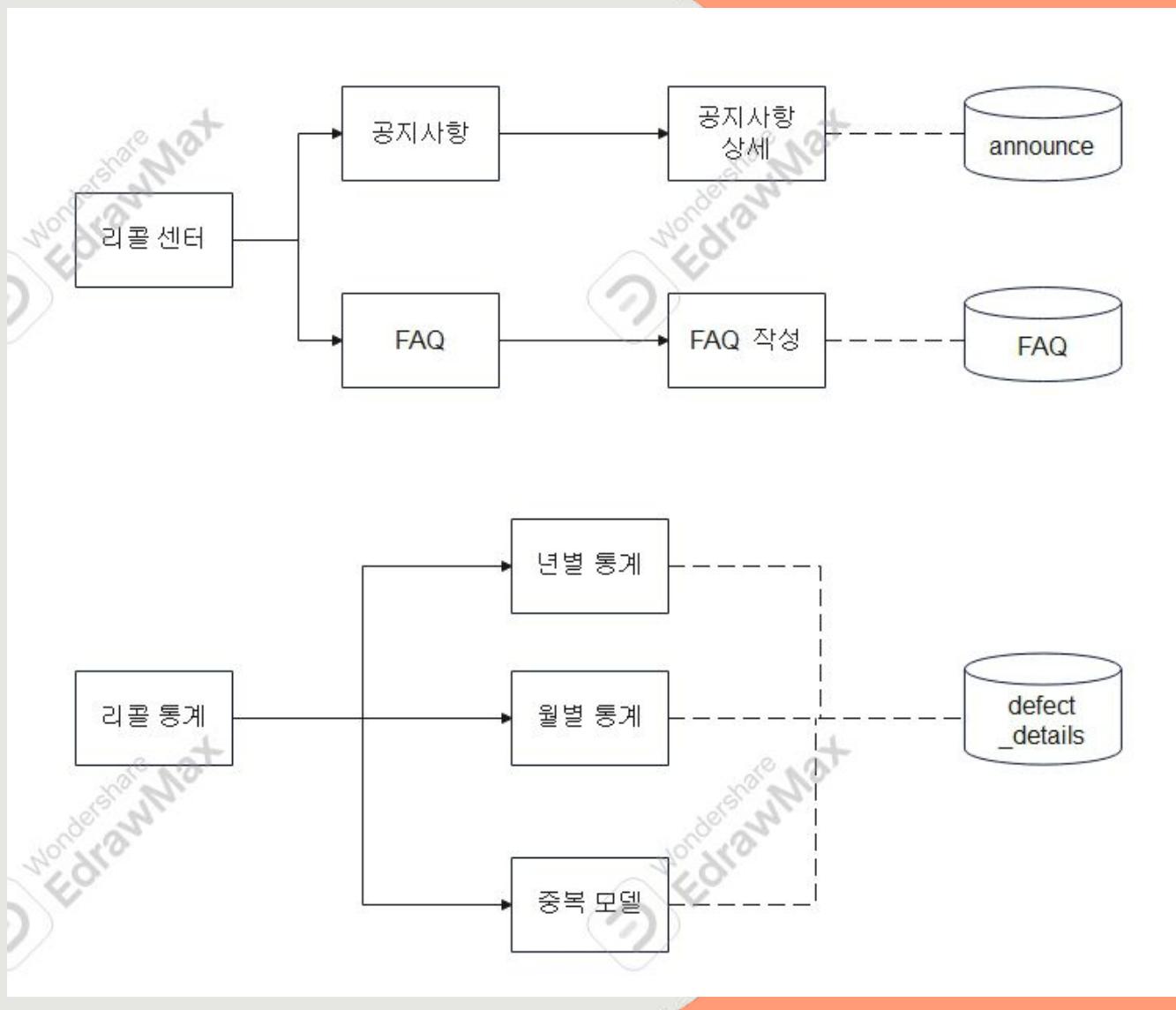
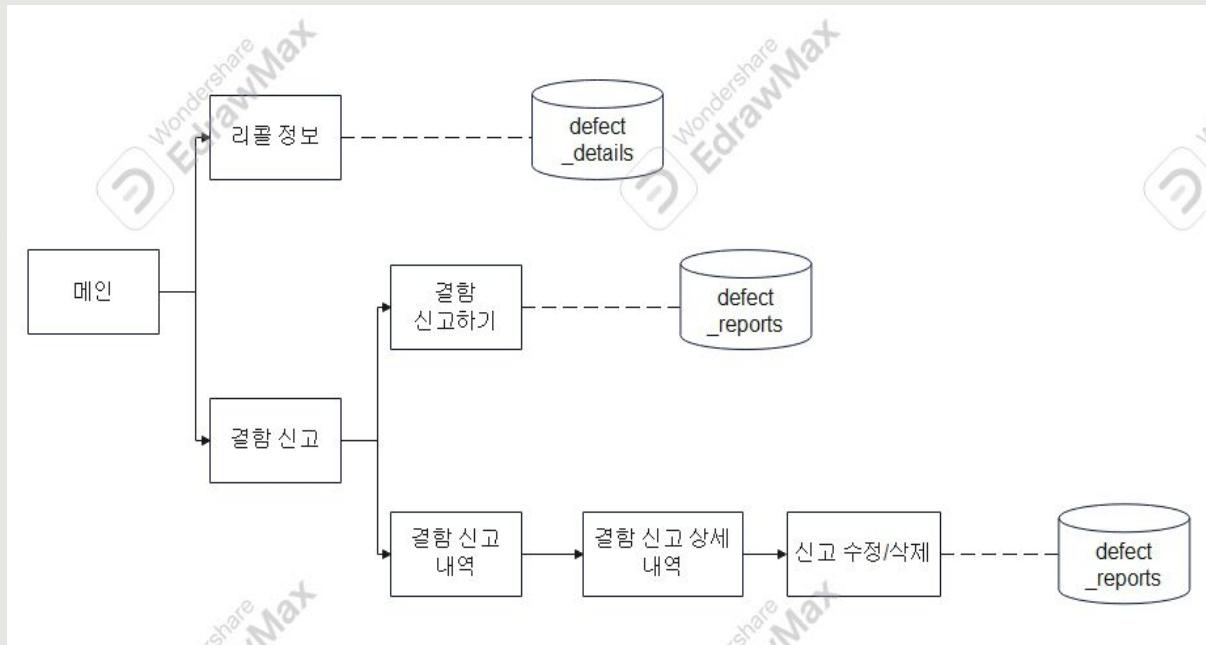
POSTMAN

업무흐름도

Workflow Diagram

빅데이터검색 플랫폼 프로젝트
Logistic Regression

리콜정보, 결함신고



리콜센터, 리콜통계

관리자

메뉴구조도

Menu Structure Diagram

			시스템명	자동차 리콜 데미터 정보					
			문서명	메뉴구조도					
			프로그램 그룹명	/da/an/qa /ad//신고/자동차/데이터/공지/질의/					
프로그램 기능 (○ : 점검 완료 / △ : 추가 점검 및 협의 / X : 기능구현 불필요)									
* 프로그램 ID 네이밍 작성 표준 : 대분류 (r : 관리자, c : 신청자), 중분류 (1,2,3,4/카테고리 번호), 소분류 (00), 세부분류(000)									
중분류	소분류	소분류별메뉴	리포트명	프로그램 ID	프로그램유형	기능확인			
결합신고				rac100001CR	CRU				
	아이디			ra100100CR	CRU				
	신고인			ra100101CR	CRU				
	생년월일			ra100102CR	CRU				
	휴대폰번호			ra100103CR	CRU				
	전화번호			ra100104CR	CRU				
	주소			ra100105CR	CRU				
	비밀번호			ra100106CR	CRU				
	공개여부			ra100107CR	CR				
	신고날짜			ra100108CR	CR				
신고유형			ra100109CR	CR					
자동차등록번호			ca100100CR	CRU					
자동차모델명			ca100101CR	CRU					
자동차제조사			ca100102CR	CRU					
제조일자			ca100103CR	CR					
결합신고 조회				rarc100002CR	CRUD				
	아이디			ra100100CR	R				
	신고인			ra100101CR	RUD				
	신고유형			ra100109CR	RUD				
	자동차모델명			ca100101CR	RUD				
	신고날짜			ra100108CR	RUD				
제조일자			ca100103CR	RU					
리콜정보관리				dar100001RP	CRU				
	아이디			da100101RP	R				
	리콜정보			da100102RP	CRU				
	자동차제조사			da100103RP	RU				
	기간			da100104RP	RU				
	자동차모델명			da100105RP	RU				
	리콜형식			da100106RP	R				
회사(대표번호)			da100107RP	RU					
상세결함			da100108RP	CRU					

공지사항			anc100001CP	CRUD				
	아이디		an100100CP	R				
	제목		an100101CP	CRUD				
	내용		an100102CP	CRUD				
리콜통계			darc100002P	R				
	년별통계		darc101001P	R				
	년		dal01109RP	R				
	자동차제조사		dal01103RP	R				
	데이터 수		dal01110RP	R				
월별통계			darc102001P	R				
	년		dal02109RP	R				
	달		an102111RP	R				
	자동차제조사		da102103RP	R				
	데이터 수		da102110RP	R				
종목모델목록			darc103001P	R				
	자동차모델명		dal03105RP	R				
	데이터 수		dal03110RP	R				
F&Q			qarc100001C	CRUD				
	아이디		qa100100CR	R				
	질문		qa100101CR	CRUD				
	답변		qa100102CR	CRUD				
관리자			adr100001CP	CRU				
	리콜정보검수		adr101001CP	CRU				
	아이디		ad101101CR	CR				
	자동차모델명		ad101102CP	CR				
	자동차제조사		ad101103CP	CR				
	기간		ad101104CR	CRU				
	상세결함		ad101105CR	CRU				
공지사항작성			adr102001C	C				
	아이디		ad102101C	C				
	이메일		ad102102C	C				
	제목		ad102103C	C				
	내용		ad102104C	C				
관리자로그인			adr103001R	R				
	아이디		ad103101R	R				
	비밀번호		ad103102R	R				

/ra/ca/da/an/qa /ad//신고/자동차/데이터/공지/질의/관리자

빅데이터검색 플랫폼 프로젝트

Logistic Regression

ERD

Entity-Relationship Diagram

빅데이터검색 플랫폼 프로젝트

Logistic Regression

defect_reports		announce		qna		defect_details	
id	INT	id	INT	id	INT	id	INT
reporter_name	VARCHAR(100) NN	title	VARCHAR(255) NN	question	VARCHAR(255) NN	product_name	TEXT NN
birth_date	CHAR(8) NN	content	TEXT NN	answer	TEXT NN	manufacturer	TEXT NN
mobile_number	VARCHAR(15) NN	created_at	TIMESTAMP	created_at	TIMESTAMP	manufacturing_period	VARCHAR(255)
phone_number	VARCHAR(15)					model_name	TEXT NN
address	VARCHAR(255)					recall_type	VARCHAR(255)
password	VARCHAR(255) NN	admin				contact_info	VARCHAR(255)
visibility	BOOLEAN	id	VARCHAR(50)			additional_info	TEXT
report_date	TIMESTAMP	password	VARCHAR(100)			hash_code	VARCHAR(64)
report_type	VARCHAR(100) NN	name	VARCHAR(50)				
car_registration_number	VARCHAR(50) NN						
car_model	VARCHAR(100)						
car_manufacturer	VARCHAR(100)						
car_manufacturing_date	DATE						

defect_reports / announce / admin / faqs / qna / defect_details

팀 구성 및 역할

Team Introduction

빅데이터검색 플랫폼 프로젝트

Logistic Regression



우주연 **TL (BE + FE)**

자동 리포트 생성 기능 (PDF)
전 화면 프론트적용(React)
챗봇기능



권준우 **BE (Auth)**

유사리콜 추천기능
JWT 인증 처리



성유리 **BE (Deploy)**

리콜 Csv , Excel 다운로드 기능
백엔드 서버 배포 및 관리



김채윤 **BE + FE
(Deploy/Integrate)**

중복리콜 확인기능
프론트 서버 배포 및 관리
앱 연동

수행 절차 및 방법

Procedure and Method

빅데이터검색 플랫폼 프로젝트

Logistic Regression

기획일정	1W				
	5/19	5/20	5/21	5/22	5/23
아이디어 회의					
JSP->REACT 마이그레이션					
BE 기능 개발					
서버 배포					
버그 퍽스, 마무리 작업					

협업툴 - jira

collaboration tool

빅데이터검색 플랫폼 프로젝트
Logistic Regression

The screenshot shows the Jira project dashboard for 'khproject3'. The top navigation bar includes links for '내 작업', '프로젝트', '필터', '대시보드', '팀', '계획', and '앱'. A '만들기' button is also present. The dashboard features a '12일 남음' timer and a search bar.

프로젝트 / khproject3

요약

18개 완료함 (지난 7일간)

35개 업데이트함 (지난 7일간)

26개 만듦 (지난 7일간)

0개 마감 예정 (다음 7일 이내)

상태 개요
Get a snapshot of the status of your work items. [View all work items](#)

총 업무 항목: 68

해야 할 일: 37

진행 중: 0

완료: 31

최근 활동
프로젝트 전반에서 일어나는 최신 정보를 파악하세요.

2일 전: 주연 님이 SCRUM-95: 통계Api데이터_DB전송,동기화 완료에서 "Rank" 필드를 업데이트했습니다.

2일 전: 주연 님이 SCRUM-99: 웹페이지 요청,서버 처리 완료에서 "Rank" 필드를 업데이트했습니다.

2일 전: 주연 님이 SCRUM-99: 웹페이지 요청,서버 처리 완료의 담당자를 성유리 님으로 변경했습니다.

작업 목록

- > SCRUM-54 보안 기능 강화 (JWT & Spring Security)
- > SCRUM-83 자동 리포트 생성 기능 (PDF)
- > SCRUM-84 결함 원인 키워드 분석
- > SCRUM-85 CSV/Excel 다운로드
- > SCRUM-86 리콜 반복 모델 탐지 기능
- > SCRUM-87 통계Api데이터_DB전송,동기...
- > SCRUM-96 React
- > SCRUM-97 AWS,app배포
- > SCRUM-104 버그리포트

완료

협업툴 - slack

collaboration tool

빅데이터검색 플랫폼 프로젝트

Logistic Regression

The screenshot shows the Slack interface for the 'khproject_3' workspace. The left sidebar lists various channels and users. The main window displays the '#지라' channel. A message from 'Automation for Jira' at 4:40 PM indicates that work has started. Another message from 'Automation for Jira' at 10:37 AM announces the start of a sprint on May 21, 2025. The message content includes links to a page request and server handling. The bottom of the screen shows the Slack message input field.

The screenshot shows the Slack interface for the '#깃허브' channel. A message from 'GitHub' at 5:46 PM indicates a new commit pushed to 'main' by 'kjo5191'. The commit details mention fixes for a bug and changes in the application properties file. Below this, another message from 'GitHub' at 5:46 PM shows a merge branch 'main' from 'https://github.com/Wjyuy/Recall_Final'. The bottom of the screen shows the Slack message input field.

리액트 - npm install

빅데이터검색 플랫폼 프로젝트
Logistic Regression

01

npm install http-proxy-middleware –save

React 개발 서버에 프록시 설정
(프론트엔드 요청-> 개발 서버가 백엔드 API로 전달-
> 동일한 도메인에서 발생한 것처럼 보이므로 CORS(
Cross-Origin Resource Sharing) 해결



02

npm install axios –save

HTTP 클라이언트 (GET, POST, PUT, DELETE 등)



03

npm install react-router-dom

React는 단일 페이지 애플리케이션(SPA) 프레임워크
클라이언트 측 라우팅을 위해 react-router-dom 라
이브러리 사용(Link, useParams 등 사용)



04

npm install recharts

차트 라이브러리



리액트 디렉토리 구조



- node_modules: React 의존성 설치
- App.js: 어플리케이션의 메인 컴포넌트(라우팅)
- Index.js: React 진입점
- setupProxy.js: API 요청 프록시하는 설정

-----Src 아래 -----

- Components: 재사용하는 UI 컴포넌트
- Hooks: 비즈니스 로직을 재사용
- Layout: header, footer, MainLayout
- Pages: 라우팅되는 페이지 컴포넌트
- Services: api 호출하는 서비스 함수

빅데이터검색 플랫폼 프로젝트
Logistic Regression

```
RECALL_FINAL
  - gradle
  - images
  - node_modules
  - src
    - main
      - frontend
        - node_modules
        - public
    - src
      - assets
      - components
      - hooks
      - layout
      - pages
      - services
      - styles
  - App.js
  - App.test.js
  - appbackup.js
  - index.css
  - index.js
  - logo.svg
  - reportWebVitals.js
  - setupProxy.js
  - setupTests.js
  - .env.development
  - .env.production
  - .gitignore
  - package-lock.json
  - package.json
  - README.md
```

리액트 - 화면구성 예시 1

빅데이터검색 플랫폼 프로젝트

Logistic Regression

```
src > main > frontend > src > App.js > ...  
1 // App.js (일부 수정)  
2 import React, { useState, useEffect } from 'react';  
3 import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';  
4 import MainLayout from './layout/MainLayout';  
5 import HomePage from './pages/HomePage';  
6 import UserProfilePage from './pages/UserProfilePage';  
7 import AnnouncePage from './pages/AnnouncePage'; // AnnouncePage import  
8 import AnnounceviewPage from './pages/AnnounceViewPage'; // AnnounceViewPage import  
24 import JwtTestPage from './pages/JwtTestPage';  
25 import { fetchTestData } from './services/api';  
26 import './styles/App.css';  
27 import './styles/main.css';  
28  
Windsurf: Refactor | Explain | Generate JSDoc | X  
function App() {  
  const [backendData, setBackendData] = useState(null);  
  
  useEffect(() => {  
    Windsurf: Refactor | Explain | Generate JSDoc | X  
    const loadTestData = async () => {  
      const data = await fetchTestData();  
      setBackendData(data);  
    };  
  
    loadTestData();  
  }, []);  
  
  return (  
    <Router>  
      <MainLayout>  
        <Routes>  
          <Route path="/" element={<HomePage />} />  
          <Route path="/profile" element={<UserProfilePage />} />  
          <Route path="/announce" element={<AnnouncePage />} />  
          <Route path="/announce_view/:id" element={<AnnounceViewPage />} />  
          <Route path="/defect_pwcheck/:id" element={<DefectPasswordCheck />} />  
          <Route path="/defect_modify/:id" element={<DefectModify />} />  
          <Route path="/recall_list" element={<RecallList />} />  
          <Route path="/recall_detail/:id" element={<RecallDetail />} />  
          <Route path="/defect_details_check" element={<DefectDetailsCheckPage />} />  
          <Route path="/recall_statics_year" element={<RecallStaticsYearPage />} />  
          <Route path="/recall_statics_month" element={<RecallStaticsMonthPage />} />  
          <Route path="/recall_statics_year/pdf" element={<PdfDownloadPage />} />  
          <Route path="/RecallCountPage" element={<RecallCount />} />  
          <Route path="/login" element={<LoginPage />} />  
          <Route path="/jwt-test" element={<JwtTestPage />} />  
        </Routes>  
      </MainLayout>  
    </Router>  
  );  
}  
  
export default App;
```

Index.js(시작점)

```
src > main > frontend > src > index.js > ...> layout > Header.js  
1 import React from 'react';  
2 import ReactDOM from 'react-dom/client';  
3 import './index.css';  
4 import App from './App';  
5 import reportWebVitals from './reportWebVitals';  
6  
7 const root = ReactDOM.createRoot(document.getElementById('root'));  
8 root.render(  
9   <React.StrictMode>  
10     <App />  
11   </React.StrictMode>  
12 );  
13  
14 // If you want to start measuring performance in your app, pass a function  
15 // to log results (for example: reportWebVitals(console.log))  
16 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals  
17 reportWebVitals();  
18
```

```
src > main > frontend > src > App.js > ...  
1 // App.js (일부 수정)  
2 import React, { useState, useEffect } from 'react';  
3 import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';  
4 import MainLayout from './layout/MainLayout';  
5 import HomePage from './pages/HomePage';  
6 import UserProfilePage from './pages/UserProfilePage';  
7 import AnnouncePage from './pages/AnnouncePage'; // AnnouncePage import  
8 import AnnounceviewPage from './pages/AnnounceViewPage'; // AnnounceViewPage import  
24 import JwtTestPage from './pages/JwtTestPage';  
25 import { fetchTestData } from './services/api';  
26 import './styles/App.css';  
27 import './styles/main.css';  
28  
Windsurf: Refactor | Explain | Generate JSDoc | X  
function App() {  
  const [backendData, setBackendData] = useState(null);  
  
  useEffect(() => {  
    Windsurf: Refactor | Explain | Generate JSDoc | X  
    const loadTestData = async () => {  
      const data = await fetchTestData();  
      setBackendData(data);  
    };  
  
    loadTestData();  
  }, []);  
  
  return (  
    <Router>  
      <MainLayout>  
        <Routes>  
          <Route path="/" element={<HomePage />} />  
          <Route path="/profile" element={<UserProfilePage />} />  
          <Route path="/announce" element={<AnnouncePage />} />  
          <Route path="/announce_view/:id" element={<AnnounceViewPage />} />  
          <Route path="/defect_pwcheck/:id" element={<DefectPasswordCheck />} />  
          <Route path="/defect_modify/:id" element={<DefectModify />} />  
          <Route path="/recall_list" element={<RecallList />} />  
          <Route path="/recall_detail/:id" element={<RecallDetail />} />  
          <Route path="/defect_details_check" element={<DefectDetailsCheckPage />} />  
          <Route path="/recall_statics_year" element={<RecallStaticsYearPage />} />  
          <Route path="/recall_statics_month" element={<RecallStaticsMonthPage />} />  
          <Route path="/recall_statics_year/pdf" element={<PdfDownloadPage />} />  
          <Route path="/RecallCountPage" element={<RecallCount />} />  
          <Route path="/login" element={<LoginPage />} />  
          <Route path="/jwt-test" element={<JwtTestPage />} />  
        </Routes>  
      </MainLayout>  
    </Router>  
  );  
}  
  
export default App;
```

<APP />태그
=컴포넌트

App.js

Index에 있는<App />자리에
얘가 들어가는 겁니다!

```
src > main > frontend > src > index.js > ...> layout > MainLayout.js  
1 import React from 'react';  
2 import ReactDOM from 'react-dom/client';  
3 import './index.css';  
4 import App from './App';  
5 import reportWebVitals from './reportWebVitals';  
6  
7 const root = ReactDOM.createRoot(document.getElementById('root'));  
8 root.render(  
9   <React.StrictMode>
```

App.js

```
10   return (  
11     <Router>  
12       <MainLayout>  
13         <Routes>  
14           <Route path="/" element={<HomePage />} />  
15           <Route path="/profile" element={<UserProfilePage />} />  
16           <Route path="/announce" element={<AnnouncePage />} />  
17           <Route path="/announce_view/:id" element={<AnnounceViewPage />} />  
18           <Route path="/defect_pwcheck/:id" element={<DefectPasswordCheck />} />  
19           <Route path="/defect_modify/:id" element={<DefectModify />} />  
20           <Route path="/recall_list" element={<RecallList />} />  
21           <Route path="/recall_detail/:id" element={<RecallDetail />} />  
22           <Route path="/defect_details_check" element={<DefectDetailsCheckPage />} />  
23           <Route path="/recall_statics_year" element={<RecallStaticsYearPage />} />  
24           <Route path="/recall_statics_month" element={<RecallStaticsMonthPage />} />  
25           <Route path="/recall_statics_year/pdf" element={<PdfDownloadPage />} />  
26           <Route path="/RecallCountPage" element={<RecallCount />} />  
27           <Route path="/login" element={<LoginPage />} />  
28           <Route path="/jwt-test" element={<JwtTestPage />} />  
29         </Routes>  
30       </MainLayout>  
31     </Router>  
32   );  
33 }  
34  
35 export default App;
```

Index.js app아래

```
36 // If you want to start measuring performance in your app, pass a function  
37 // to log results (for example: reportWebVitals(console.log))  
38 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals  
39 reportWebVitals();  
40
```

리액트 - 화면구성 예시 1

빅데이터검색 플랫폼 프로젝트

Logistic Regression

App.js

```
src > main > frontend > src > JS App.js > ...
1 // App.js (일부 수정)
2 import React, { useState, useEffect } from 'react';
3 import { BrowserRouter as Router, Routes, Route, Link } from 'react-router-dom';
4 import MainLayout from './layout/MainLayout';
5 import HomePage from './pages/HomePage';
6 import UserProfilePage from './pages/UserProfilePage';
7 import AnnouncePage from './pages/AnnouncePage'; // AnnouncePage import
8 import AnnounceViewPage from './pages/AnnounceViewPage'; // AnnounceViewPage import

...
40 return (
41   <Router>
42     <MainLayout>
43       <Routes>
44         <Route path="/" element={<HomePage />} />
45         <Route path="/profile" element={<UserProfilePage />} />
46         <Route path="/announce" element={<AnnouncePage />} />
47         <Route path="/announce_view/:id" element={<AnnounceViewPage />} />
48         ...
49         <Route path="/defect_pwcheck/:id" element={<DefectPasswordCheck />} />
50         <Route path="/defect_modify/:id" element={<DefectModify />} />
51         <Route path="/recall_list" element={<RecallList />} />
52         <Route path="/recall_detail/:id" element={<RecallDetail />} />
53         <Route path="/defect_details_check" element={<DefectDetailsCheckPage />} />
54         <Route path="/recall_statics_year" element={<RecallStaticsYearPage />} />
55         <Route path="/recall_statics_Month" element={<RecallStaticsMonthPage />} />
56         <Route path="/recall_statics_year/pdf" element={<PdfDownloadPage />} />
57         <Route path="/RecallCountPage" element={<RecallCount />} />
58         <Route path="/login" element={<LoginPage />} />
59         <Route path="/jwt-test" element={<JwtTestPage />} />
60       </Routes>
61     </MainLayout>
62   </Router>
63 );
64
65
66
67
68
69
70
71
72
73
74 export default App;
75
```

파일명-임포트
아름매핑

pages
JS AnnouncePage.js
JS AnnounceViewPage.js
JS AnnounceWritePage.js
JS DefectDetail.js
JS DefectDetailsCheckPage.jsx
JS DefectModify.js
JS DefectPasswordCheck.js
JS DefectReportList.js
JS FaqPage.js
JS FaqWritePage.js
JS HomePage.js
JS JwtTestPage.jsx
JS LoginPage.jsx
JS RecallCountPage.js
JS RecallDetail.js
JS RecallList.js
JS RecallStaticsMonthPage.jsx
JS RecallStaticsYearPage.jsx
JS ReportDefectPage.js
JS UserProfilePage.js

주소 매핑
(http://.../login)

<Route>랑
import 페이지명 해서
클라이언트 측 라우팅으로 페이지전환

리액트 - 화면구성 예시 2

빅데이터검색 플랫폼 프로젝트

Logistic Regression

Announcepage.js

컴포넌트는 위에서 임포트!

src / main / ui / uiComponents / src / pages / .. / Announcepage.js / App.js

```
1 // pages/AnnouncePage.js
2 // 공지사항 리스트 출력되는 페이지 입니다
3 import React, { useEffect, useState, useCallback } from 'react';
4 import { useSearchParams, Link } from 'react-router-dom'; // URL 쿼리 파라미터 읽기
5 import AnnounceList from '../components/AnnounceList';
6 import AnnounceSearch from '../components/AnnounceSearch';
7 import Pagination from '../components/Pagination';
8 import { fetchAnnouncements } from '../services/announceService';
9
10 return (
11   <section id="starter-section" className="starter-section section">
12     <div className="container" data-aos="fade-up">
13       <div className="section-title text-center">
14         <h2 className="title">공지사항</h2>
15       </div>
16       <div className="widgets-container detail-widgets-container">
17         <AnnounceSearch
18           onSearch={handleSearch}
19           initialType={pageMaker.cri.type}
20           initialKeyword={pageMaker.cri.keyword}
21         />
22         <AnnounceList
23           announcements={announcements}
24           total={pageMaker.total}
25         />
26       </div>
27       <br />
28       <Pagination
29         pageMaker={pageMaker}
30         onPageChange={handlePageChange}
31       />
32     </div>
33   </section>
34 )
```

AnnounceSearch.js

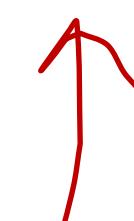
```
return (
  <form onSubmit={handleSubmit} className="search-form" style={{ marginBottom: '20px', display: 'flex', gap: '10px', alignItems: 'center' }}>
    <select name="type" value={type} onChange={e => setType(e.target.value)}>
      <option value="">전체</option>
      <option value="T">제목</option>
      <option value="C">내용</option>
    </select>
    <input
      type="text"
      name="keyword"
      value={keyword}
      onChange={e => setKeyword(e.target.value)}
      placeholder="검색어를 입력하세요"
      style={{ flexGrow: 1, padding: '8px', borderRadius: '4px', border: '1px solid #ccc' }}
    />
    <button type="submit">검색</button>
  </form>
```

AnnounceList.js

```
return (
  <table className="table-custom">
    <thead>
      <tr>
        <th>번호</th>
        <th>제목</th>
        <th>작성시간</th>
      </tr>
    </thead>
    <tbody>
      {announcements.map(({ id, title, created_at }) => (
        <tr key={id} onClick={() => navigate(`/announce_view/${id}?total=${total}`)}>
          <td>{id}</td>
          <td>{title}</td>
          <td>{formatDateTime(created_at)}</td>
        </tr>
      ))}
    </tbody>
  </table>
```

AnnounceList.js

```
return (
  <section id="blog-pagination" className="blog-pagination section">
    <div className="container">
      <div className="d-flex justify-content-center">
        <div className="div_page">
          <ul>
            {prev &&
              <li className="paginate_button">
                <button type="button" onClick={() => handlePageClick(startPage - 1)} className="link-button">
                  <i className="bi bi-chevron-left">/i
                </button>
              </li>
            }
            {Array.from({ length: endPage - startPage + 1 }, (_, i) => startPage + i).map((num) => (
              <li key={num} className={`paginate_button${cri.pageNum === num ? ' active' : ''}`}>
                <button type="button" onClick={() => handlePageClick(num)} className="link-button">
                  {num}
                </button>
              </li>
            ))
            {next &&
              <li className="paginate_button">
                <button type="button" onClick={() => handlePageClick(endPage + 1)} className="link-button">
                  <i className="bi bi-chevron-right">/i
                </button>
              </li>
            }
          </ul>
        </div>
      </div>
    </div>
  </section>
);
```



리액트 - 컴포넌트화의 장점

빅데이터검색 플랫폼 프로젝트

Logistic Regression

재사용성이 매우 좋습니다

예시: <pagination /> 사용

defectlist.js

```
onSearchChange={handleSearchChange}
onSearchSubmit={handleSearchSubmit}
/>


| 번호 | 신고자 | 신고유형 | 모델명 | 신고일 |
|----|-----|------|-----|-----|
|----|-----|------|-----|-----|


<Pagination pageMaker={pageMaker} onPageChange={handlePageChange} />
```

Announcepage.js

```
return (
  <section id="starter-section" className="starter-section section">
    <div className="container" data-aos="fade-up">
      <div className="section-title text-center">
        <h2 className="title">공지사항</h2>
      </div>
      <div className="widgets-container detail-widgets-container">
        <AnnounceSearch
          onSearch={handleSearch}
          initialType={pageMaker.cri.type}
          initialKeyword={pageMaker.cri.keyword}
        />
        <AnnounceList announcements={announcements} total={pageMaker.total} />
      </div>
      <br />
      <Pagination pageMaker={pageMaker} onPageChange={handlePageChange} />
    </div>
  </section>
)
```

FAQ.js

```
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
</div>
</section> /* /Faq Section */
<div className='detail-widgets-container'>
  <AnnounceSearch
    onSearch={handleSearch}
    initialType={pageMaker.cri.type}
    initialKeyword={pageMaker.cri.keyword}
  />
  <div style={{ textAlign: 'right', marginTop: '20px' }}>
    <Link to="/notice_write" >질문 작성하기</Link>
  </div>
</div>
/* Blog Pagination Section (AnnouncePage에서 재사용) */
<Section id="blog-pagination" className="blog-pagination section">
  <Pagination pageMaker={pageMaker} onPageChange={handlePageChange} />
</Section> /* /Blog Pagination Section */

</div>
</section> /* /Starter Section Section */
</main>
```

RecallList.js

```
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
<button
  id="downloadCsvBtn"
  onClick={handleDownloadCsv}
>
  CSV 전체 다운로드
</button>
<button
  id="excelDownloadBtn"
  onClick={handleDownloadExcel}
>
  엑셀 전체 다운로드
</button>
</div>
/* Pagination 컴포넌트 재사용 */
<Pagination pageMaker={pageMaker} onPageChange={handlePageChange} />
</div>
</section>
</main>
```

리액트 - 프론트 > 백엔드 호출

빅데이터검색 플랫폼 프로젝트
Logistic Regression

AnnouncePage.js

```
44 const loadAnnouncements = useCallback(async (params) => {
45   setLoading(true);
46   setError(null);
47   try {
48     const data = await fetchAnnouncements(params);
49     setAnnouncements(data.list);
50     setPageMaker(data.pageMaker);
51
52     // URL 쿼리 파라미터 업데이트는 여기서 한 번만
53     setSearchParams(new URLSearchParams(params).toString());
54
55   } catch (err) {
56     setError('공지사항을 불러오는 데 실패했습니다.');
57     console.error(err);
58   } finally {
59     setLoading(false);
60   }
61 }, [setSearchParams]); // loadAnnouncements의 의존성에는 setSearchParams만!
```

announceService.js

```
11 // 공지사항전체
12 export const fetchAnnouncements = async (params) => {
13   try {
14     // 예시: /api/announcements?pageNum=1&amount=10&type=T&keyword=제목
15     const response = await axios.get(`${API_BASE_URL}/announce`, { params });
16     return response.data; // 서버에서 받은 데이터 (list와 pageMaker 정보 포함)
17   } catch (error) {
18     console.error('공지사항 데이터를 가져오는 데 실패했습니다:', error);
19     throw error;
20   }
21 };
22
```

Async = 비동기적 동작
(params) = 받을 인자(백엔드 DTO(pageNum,type 등))
axios.get() : HTTP GET 요청
\${API_BASE_URL}/announce =
http://localhost:8485/api/announce (로컬) URL 요청
Await로 비동기작업 완료 후 response반환
response.data에 JSON 응답을 JavaScript 객체로 자동 파싱

챗봇기능

빅데이터검색 플랫폼 프로젝트
Logistic Regression

- 드래그해서 위치이동가능
- 프롬프트를 미리 설정해둬서
자동차리콜 관련 질문에만 대답

The screenshot displays the Recall Center interface. At the top, there is a dark header bar with the text "Recall center" on the left and navigation items "리콜정보", "결합신고", "리콜센터", "리콜통계", and "관리자" on the right.

The main area is titled "RECALL-CENTER". It features a sidebar on the left labeled "챗봇 상담" (Chatbot Support) containing a message: "안녕하세요! 자동차 결함 및 리콜에 대해 궁금한 점을 질문해주세요." Below this is a text input field "메시지를 입력하세요..." and a send button "전" with an upward arrow icon.

The central part of the screen is titled "주요 서비스" (Main Services) and contains four service cards:

- 리콜 정보**: Represented by a car icon. Description: "최신 차량 리콜 정보를 확인하세요."
- 결합 신고**: Represented by a pencil icon. Description: "차량 결함 신고 및 내역을 조회합니다."
- 리콜 센터**: Represented by an info icon. Description: "공지사항 및 FAQ를 확인합니다."
- 리콜 통계**: Represented by a line graph icon. Description: "연도별, 월별 리콜 통계를 확인하세요."

On the right side, there is a "챗봇 상담" (Chatbot Support) window with a message: "안녕하세요! 자동차 결함 및 리콜에 대해 궁금한 점을 질문해주세요." It includes a text input field "메시지를 입력하세요...", a send button "전" with an upward arrow icon, and a message icon.

챗봇기능

빅데이터검색 플랫폼 프로젝트

Logistic Regression

1ChatbotPopup.js

```
93 const handleMessage = async (e) => {
94   e.preventDefault();
95   if (inputMessage.trim() === '') return;
96
97   const userMsg = { text: inputMessage, sender: 'user' };
98   setMessages([...prevMessages, userMsg]);
99   setInputMessage('');
100  setIsTyping(true);
101
102 try {
103   const botResponse = await askChatbot(userMsg.text);
104   setMessages([...prevMessages, { text: botResponse, sender: 'bot' }]);
105 } catch (error) {
106   console.error('Error fetching chatbot response:', error);
107   setMessages([...prevMessages, { text: '죄송합니다. 챗봇 응답을 가져오지 못했습니다.' }]);
108 } finally {
109   setIsTyping(false);
110 }
111};
```

1 챗봇팝업 컴포넌트에서 유저가 작성한
text를 파라미터로 AskChatbot 함수 사용

~~폼형식이기 때문에, 전송버튼 클릭시 handleSendMessage 실행~~
`<form onSubmit={handleSendMessage}>`

2chatbotApi.js

```
4 export const askChatbot = async (question) => {
5   try {
6     const response = await fetch(`[${API_BASE_URL}]/askchatbot`, {
7       method: 'POST',
8       headers: {
9         'Content-Type': 'application/json',
10      },
11      body: JSON.stringify({ question }),
12    });
13
14    if (!response.ok) {
15      // HTTP 오류 상태 코드 처리
16      const errorText = await response.text();
17      throw new Error(`HTTP error! status: ${response.status}, message: ${errorText}`);
18    }
19
20    const data = await response.text(); // 응답이 순수 텍스트이므로 .text() 사용
21    return data;
22  } catch (error) {
23    console.error('챗봇 API 호출 중 오류 발생!', error);
24    // 사용자에게 보여줄 에러 메시지를 반환하거나 처리
25    return '죄송합니다. 챗봇과 연결하는 중 오류가 발생했습니다.';
26  }
27};
```

```
유저가 보낸 정보로 객체 저장  
Const userMsg = { text: inputMessage, sender: 'user' };  
메시지 보낸것은 바로 화면에 출력(... 있어야 바로 변함)
```

3Controller

```
7 @PostMapping("/askchatbot")
8 public ResponseEntity<String> askChatbot(@RequestBody ChatRequest request) {
9     String userQuestion = request.getQuestion();
10    String predefinedPrompt = "당신은 자동차 결함 및 리콜 전문가 챗봇입니다. "
11
12    *프롬프트*
13    (당신은리콜 전문가 챗봇입니다.)
14    (묻는말에만 대답하십시오.)
15
16    String fullQuestion = predefinedPrompt + userQuestion;
17
18    System.out.println("Gemini에 보낼 전체 질문: " + fullQuestion); // 디버깅 용도
19
20    String geminiAnswer = geminiService.askGemini(fullQuestion);
21    return ResponseEntity.ok(geminiAnswer);
22 }
```

3. 매팅 찾아서 /askchatbot으로 받아서
Gemini service에 질문 - 대답 형식으로 넣어서 대답을 전송

챗봇기능

빅데이터검색 플랫폼 프로젝트

Logistic Regression

5ChatbotPopup.JS

```
93 const handleSendMessage = async (e) => {
94   e.preventDefault();
95   if (inputMessage.trim() === '') return;
96
97   const userMsg = { text: inputMessage, sender: 'user' };
98   setMessages((prevMessages) => [...prevMessages, userMsg]);
99   setInputMessage('');
100  setIsTyping(true);
101
102 try {
103   const botResponse = await askChatbot(userMsg.text);
104   setMessages((prevMessages) => [...prevMessages, { text: botResponse, sender: 'bot' }]);
105 } catch (error) {
106   console.error('Error fetching chatbot response:', error);
107   setMessages((prevMessages) => [...prevMessages, { text: '죄송합니다. 챗봇 응답을 가져오지 못했습니다.', sender: 'bot' }]);
108 } finally {
109   setIsTyping(false);
110 }
111 }
```

```
5 setMessages((prevMessages) =>
[...prevMessages, { text: botResponse, sender: 'bot' }]);
봇이보낸 메시지 객체 생성 { text: botResponse, sender: 'bot' }
```

4chatbotApi.JS

```
4 export const askChatbot = async (question) => {
5   try {
6     const response = await fetch(`${API_BASE_URL}/askchatbot`, {
7       method: 'POST',
8       headers: {
9         'Content-Type': 'application/json',
10      },
11      body: JSON.stringify({ question }),
12    });
13
14    if (!response.ok) {
15      // HTTP 오류 상태 코드 처리
16      const errorText = await response.text();
17      throw new Error(`HTTP error! status: ${response.status}, message: ${errorText}`);
18    }
19
20    const data = await response.text(); // 응답이 순수 텍스트이므로 .text() 사용
21    return data;
22  } catch (error) {
23    console.error('챗봇 API 호출 중 오류 발생:', error);
24    // 사용자에게 보여줄 에러 메시지를 반환하거나 처리
25    return '죄송합니다. 챗봇과 연결하는 중 오류가 발생했습니다.';
26  }
27 }
```

4전송한대답을 받아서 data로 리턴

```
132 <div className="chatbot-messages">
133 {messages.map((msg, index) => (
134   <div key={index} className={`message ${msg.sender}`}>
135     {msg.text}
136   </div>
137 ))
138 {isTyping && <div className="message bot typing-indicator">...</div>}
139 <div ref={messagesEndRef} />
140 </div>
```

6 {messages.map(...)}으로 채팅 메시지 업데이트

챗봇기능

빅데이터검색 플랫폼 프로젝트
Logistic Regression

www.BANDICAM.com

Recall center

주요 서비스



리콜 정보

최신 차량 리콜 정보를 확인하세요.



리콜 통계

연도별, 월별 리콜 통계를 확인하세요.



결함 신고

차량 결함 신고 및 내역을 조회합니다.



신고 내역

차량 결함 신고 및 내역을 조회합니다.



주소 - <https://recall-final-front.onrender.com/>

빅데이터검색 플랫폼 프로젝트
Logistic Regression

recall-final-front.onrender.com

Recall center

리콜정보 결합신고 리콜센터 리콜통계 관리자

RECALL-CENTER

주요 서비스



리콜 정보

최신 차량 리콜 정보를 확인하세요.



리콜 통계

연도별, 월별 리콜 통계를 확인하세요.









The screenshot shows a web browser window with the URL www.BANDICAM.com in the address bar. The browser title is "React App" and the address is "localhost:3000". The page content is the "Recall center" homepage.

Recall center (Header)

리콜정보 **결함신고** ▾ **리콜센터** ▾ **리콜통계** ▾ **관리자** ▾ (Header menu)

RECALL-CENTER

주요 서비스

리콜 정보

최신 차량 리콜 정보를 확인하세요.

리콜 통계

연도별, 월별 리콜 통계를 확인하세요.

CSV/ Excel 다운로드

빅데이터검색 플랫폼 프로젝트

Logistic Regression

#	d	manufacturer	model_name
1	1	기아자동차(주)	일렉시티(CV)
2	2	(주)기아차 어전트로보 코리아	다렌
3	3	3비엠디모토코리아(주)	BMW 320i, BMW 320i Touring, BMW 420 Convertible, BMW 420i Coupe, BMW 420i Gran Coupe, BMW 520i, BMW 523d, BMW 523d xDrive
4	4	4비엠디모토코리아(주)	BMW X3 20i Drive, BMW X3 20d xDrive
5	5	5비엠디모토코리아(주)	MINI
6	6	6비엠디모토코리아(주)	AMG S 63 E Performance, AMG GT 63 S E Performance
7	7	7비엠디모토코리아(주)	E 300 4MATIC, E 300 4MATIC, E 200, E 450 4MATIC
8	8	8비엠디모토코리아(주)	BMW
9	9	9비엠디모토코리아(주)	ACCORD, ACCORD HYBRID, CR-V, ODYSSEY, PILOT, CIVIC, CROSSTOUR, CR-Z, HR-V, LEGEND
10	10	(주)풀타운자동차코리아	S90, V90CC, S90, V60CC, XC60, XC90, XC40, C40
11	11	11폭스바겐그룹코리아	주식회사
12	12	12폭스바겐그룹코리아	TGS, TGX, TGM, TGL
13	13	13비르센데스스펜스코리아(주)	G
14	14비르센데스스펜스코리아(주)	GV70 JK1 PE, GV70 ElectrifiedJK1 EV PE, 아이오닉9OME1	
15	15	15비르센데스스펜스코리아(주)	주식회사
16	16	16폭스바겐그룹코리아	주식회사
17	17	17폭스바겐그룹코리아	주식회사
18	18	18폭스바겐그룹코리아	주식회사
19	19	19비르센데스스펜스코리아(주)	R 18 R, 18 CLASSIC, R 18 B, R 18 TANSCONTINENTAL, R 18 ROCTANE
20	20	20폭스바겐그룹코리아	주식회사
21	21	21한국쉘피유자동차부품(주)	로터리 시에나 하이브리드 2WD, 토아타 시에나 하이브리드 AWD
22	22	22한국쉘피유자동차부품(주)	BMW R 12, BMW R 12 mInET
23	23	23한국쉘피유자동차부품(주)	BMW 218i, BMW 218d, 더 뉴 레이인지로버 D350 SWB, 더 뉴 레이인지로버 P350 LWB, 더 뉴 레이인지로버 P350 SWB
24	24	24한국쉘피유자동차부품(주)	S90, XC60, XC90
25	25	25비르센데스스펜스코리아(주)	주식회사
26	26비르센데스스펜스코리아(주)	주식회사	
27	27비르센데스스펜스코리아(주)	카이로	
28	28	28비르센데스스펜스코리아(주)	S 390 S 400 4MATIC, S 450 4MATIC, S 500, S 500 4MATIC, S 580 4MATIC, Maybach S 580 4MATIC, Maybach S 580 4MATIC
29	29	29비르센데스스펜스코리아(주)	AMG GT 63 S E Performance, AMG S 63 E Performance, AMG GT 63 S E Performance
30	30	30비르센데스스펜스코리아(주)	주식회사
31	31	31비르센데스스펜스코리아(주)	BMW 118d, BMW X1 x200 A., BMW X1 xdrive20d, BMW X1 xdrive18d, BMW 335i Convertible
32	32	32비르센데스스펜스코리아(주)	주식회사
33	33	33비르센데스스펜스코리아(주)	온다
34	34	34비르센데스스펜스코리아(주)	BMW
35	35	35주식회사	모빌리티네트웍스
36	36	36(주)기아차 어전트로보 코리아	디펜더 110 300, 디펜더 110 400
37	37	37주식회사	모빌리티네트웍스
38	38	38비르센데스스펜스코리아(주)	E 300 4MATIC, E 200, E 350 4MATIC
39	39	39(주)현대차	다나고
40	40	40현대차	다나고 사용량
41	41	41현대차	E 220 4MATIC, E 300 4MATIC, CLE 200
42	42	42현대차	유한회사
43	43	43스텔란티스스코리아	주식회사
44	44	44(주)스즈키코리아	스즈키
45	45	45아이지이빌리티	주식회사(KG)
46	46	46비르센데스스펜스코리아(주)	S 390 4MATIC, Maybach S 580 4MATIC
47	47	47(주)현대차	현대차
48	48	48(주)현대차	현대차
49	49	49(주)현대차	현대차

ID	Product Name	Manufacturer	Model Name
2	[현대] 일렉시 수소전기버스 - 수소 배출구 보호마개 관련 리플	현대자동차(주)	일렉시(EV) 수소전기버스
3	[제국수련트로디] 디펜더 110 D250 - 엔진 고장 및 헤드루프 외도필터 관련 리플	(주)제국수련트로디코리아	디펜더 110 D250
3	[비엠더블유] BMW X3 20 dDrive 등 2종 - 48 스타터 발전기 관련 리플	비엠더블유코리아(주)	BMW X3, BMW 320d Touring, BMW X3 20 dDrive, BMW X3 d20d
5	[비엠더블유] MINI Cooper SE - 고전압 배터리 하우징 관리 리플	비엠더블유코리아(주)	MINI Cooper SE
6	[현대] AMG G 63 E Performance 등 2종 - 고전압 시동 발전기 관련 리플	AMG G 63 E Performance, AMG GT E 220 d 4MATIC 등 2종	AMG G 63 E Performance, AMG GT E 220 d 4MATIC, E 300 4MATIC, CLA 250 d 4MATIC
7	[현대] E 220 d 4MATIC 등 2종 - 48V 시동 엔진 풀보트 관리 리플	현대자동차(주)	BMW 5 45d
9	[비엠더블유] BMW 7 50d xDrive 등 2종 - 고전압 배터리 관리 리플	BMW 7 50d xDrive	ACCORD, ACCORD HYBRID, CR-V, SGR V60CC, S60, V60CC, XC60, XC90
10	[포르쉐] 999 911 992 등 2종 - 외부 배터리 관리 리플	포르쉐(독일)	A5 40 TFSI, A5 45 TFSI, Q5 45 TFSI
10	[현대] S99 1000 1000 등 2종 - 고장기 진단장치 관리 리플	현대모비스(주)	TGS, TOX, TGM, GL
11	[폭스스킨 그룹] AS 40 TFSI quattro 등 4종 - 동승자석 어에브 모듈 관리 리플	폭스스킨 그룹코리아(주)	G 400 d
12	[마린티] TGS 등 4종 - 시가스관 관리 리플	마린티(주)	G700 HKI REE, G70 Electrified,LK1 미안, 막간 4, 미간 4, 미간 터보
13	[현대] G 400 d - 연진 퀸트 페인트 관리 리플	현대자동차(주)	15 AT FTSI vs. Premium
14	[현대] G 700 700 등 2종 - 배터리 관리 리플	현대모비스(주)	E 18, R 18 CLASSIC, R 18 B, R 18 T
14	[현대] G 700 700 등 2종 - 제어기판 관리 리플	현대모비스(주)	Q5 40 TFSI vs. Premium, Q5 35 TDI vs. Premium
15	[포르쉐] ME 400 등 4종 - 우방기판 리소프트웨어 관리 리플	포르쉐코리아(주)	AD 40 TD vs. 312종
16	[폭스스킨 그룹] A7 55 TFSI vs. Premium - 고전압 배터리 관리 리플	폭스스킨 그룹코리아(주)	토토로 시내 하이웨이 2WD, 토토로 시내 하이웨이 4WD
16	[폭스스킨 그룹] A7 55 TFSI vs. Premium - 고전압 배터리 관리 리플	폭스스킨 그룹코리아(주)	BMW R 12, BMW R 12 nineT
17	[폭스스킨 그룹] OS e-tron performance - 우방기판 리소프트웨어 관리 리플	폭스스킨 그룹코리아(주)	더 뉴 레인지로버 D350 LWB, 더 뉴
18	[비엠더블유] R 18 등 5종 - 차우트 조향장치 관리 리플	비엠더블유코리아(주)	S 500 4MATIC
19	[폭스스킨 그룹] OS 40 TFSI vs. Premium 등 3종 - 차우트 - 커스텀 퀸트 유닛 소프트웨어 관리 리플	폭스스킨 그룹코리아(주)	K아모로 (Camaro)
20	[폭스스킨 그룹] OS 40 TDS vs. Premium 등 3종 - 제어기판 관리 리플	폭스스킨 그룹코리아(주)	S 350 d, S 400 d 4MATIC, S 450 4M
21	[포드오토] POI 시리즈 110 110 등 2종 - 차우트 - 차체 통신 모듈 관리 리플	포드오토(주)	AMG G 63 S E Performance, AMG
22	[비엠더블유] RMW 12 12 등 2종 - 고전압 배터리 관리 리플	비엠더블유코리아(주)	제이엠아시아피리케이션부문 ATS, CTS
22	[비엠더블유] RMW 12 12 등 2종 - 고전압 배터리 관리 리플	비엠더블유코리아(주)	BMW 118d, BMW X1 X20d A., BMW
23	[제국수련트로디] 더 뉴 레인지로버 D350 LWB 등 4종 - 우방기판 관리 리플	(주)제국수련트로디코리아	온라인 GT, RS e-tron GT
24	[파운드] S99 300 4MATIC - 푸스 박스 관리 리플	(주)파운드차트코리아	온다 AV750L
25	[현대] KIA 500 4MATIC - 푸스 박스 관리 리플	현대자동차(주)	BMW 750e xDrive
26	[한국기아] 카메로 - 전진식 조향장치 관리 리플	한국기아(주)	SE-24전
27	[현대] S 350 d 4MATIC 등 2종 - 전진 브레이크 이호스 관리 리플	현대모비스(주)	디펜더 110 P300, 디펜더 110 P400
28	[현대] AMG GT 63 S E Performance 등 2종 - 고전압 시동 발전기 관련 리플	현대모비스(주)	SE-24전
30	[지엠코리아] ATS 등 2종 - 고전압 퀸트 조향장치 관리 리플	지엠코리아(사)피리케이션부문 ATS, CTS	E 300 4MATIC, E 200, E 350 e 4MATIC
30	[비엠더블유] BMW 118d 등 5종 - 제어기판 관리 리플	비엠더블유코리아(주)	chn고 쟁
31	[폭스스킨 그룹] e-tron GT 등 2종 - 고전압 배터리 모듈 관리 리플	폭스스킨 그룹코리아(주)	C40
32	[현대] AD 750L vs. Premium vs. 시 원동기 관리 및 고장 개오 관리 리플	현대모터스(주)	E 220 d 4MATIC, E 300 4MATIC, CLA
33	[비엠더블유] BMW 750e xDrive - 통제장치 관리 리플	비엠더블유코리아(주)	Model Y, Model 3
34	[모빌리티네트워크] SE-24전 - 동승석 차체 관리 리플	주식회사 모빌리티네트워크	2021~2023년 디젤 블랙터 PHEV, 2022
35	[제국수련트로디] 디펜더 110 P300 등 2종 - 스노우 페인트 이하이드 키 관련 리플	(주)제국수련트로디코리아	스즈키 AN400A
36	[모빌리티네트워크] SE-24전 - 제어기판 관리 리플	주식회사 모빌리티네트워크	카이보모빌리티 주식회사(KG Mot
37	[현대] E 300 4MATIC 등 2종 - 차우트 헤드 챠저 캐리어 관리 리플	모빌리티네트워크(주)	트리플러(G15), 포란도(G15)
38	[후루타] CT40 - 차우트 - 전진 브레이크 장치 관리 리플	(주)후루타	S 580 4MATIC, Maybach S 580 4M
39	[더릉원사람] G40 - 컨트롤러 관리 리플	더릉원사람	내 차 GTR
40	[현대] E 220 d 4MATIC 등 2종 - MUBX 펫리미디어 시스템 관리 리플	현대모비스(주)	
41	[데일레이] Model Y 등 2종 - 바이어 공기길 유지장치 관리 리플	데일레이(주)	
42	[스텔란티스] SUV 플랫폼 EVH 등 2종 - 고전압 배터리 관리 리플	스텔란티스(주)리아우주식회사	
43	[아모레퍼시픽] 스즈키 AN400A - 연료 페인트 관리 리플	(주)아모레퍼시픽	
44	[현대] 차우트 헤드 챠저 캐리어 관리 리플	차우트 헤드 챠저 캐리어	
45	[현대] S 580 4MATIC 등 2종 - 연진 퀸트 페인트 관리 리플	현대모비스(주)	
46	[현대] 차우트 헤드 챠저 캐리어 관리 리플	차우트 헤드 챠저 캐리어	
47	[현대] 차우트 헤드 챠저 캐리어 관리 리플	차우트 헤드 챠저 캐리어	
Recall List			

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

리콜 내역

검색 유형 선택 ▾

검색

리콜 ID	제품명	제조사	제조기간	기타정보	모델명	리콜유형	연락처
850	타이어	Tesla		타이어 3개 터짐으로 수정	K5	자발적리콜	[벤츠코리아] 대...
849	[현대] 일렉시티 수소전...	현대자동차(주)	20190515~2025...	수소 배출구 보호마개 개선...	일렉시티(CY) 수소전기버스	자발적리콜	현대자동차 대표...
848	[재규어랜드로버] 레인지...	(주)재규어랜드로...	2017-05-02 ~ 20...		레인지로버 벨라	자발적리콜	재규어랜드로버 ...
847	[지엠코리아] 캐딜락 STS...	지엠코리아주식...	2004-09-17 ~ 20...		캐딜락 STS	자발적리콜	지엠코리아 고객...
846	[토요타] 토요타 시에나 ...	한국토요타자동...	2017-10-18 ~ 20...		토요타 시에나 2WD, 토요타 시에나...	자발적리콜	한국토요타자동...
845	[이탈로모토(유)] BEVERL...	이탈로모토(유)	2016-06-01 ~ 20...		BEVERLY 350 SPORT TOURING ABS	자발적리콜	이탈로모토 대표...
844	[화창상사] ROADMASTER...	화창상사(주)	2017-04-20 ~ 20...		ROADMASTER ELITE, ROADMASTER...	자발적리콜	화창상사 고객센...
843	[혼다코리아] ODYSSEY -...	혼다코리아(주)	2013-11-07 ~ 20...		ODYSSEY	자발적리콜	혼다코리아 고객...
842	[비엠더블유] 550i 등 13...	비엠더블유코리...	2008-05-08 ~ 20...		550i, 550i xDrive, 650i Convertible,...	자발적리콜	비엠더블유 커뮤...
841	[비엠더블유] Ghost - 터...	비엠더블유코리...	2010-11-18 ~ 20...		Rolls-Royce Ghost	자발적리콜	롤스로이스 서울 ...

© 2019 Pearson Education, Inc.

© 2019 Pearson Education, Inc.

CSV/ Excel 다운로드

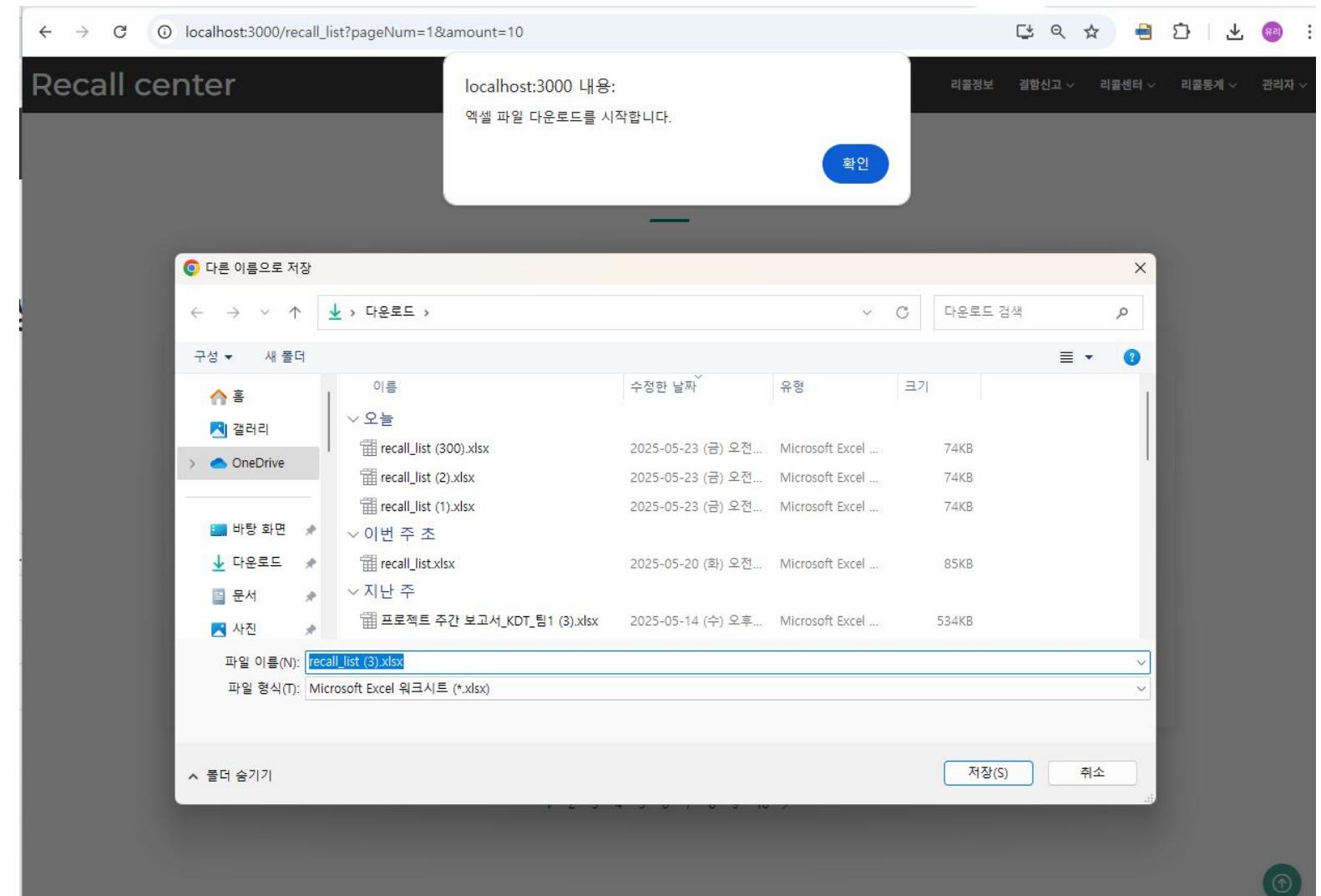
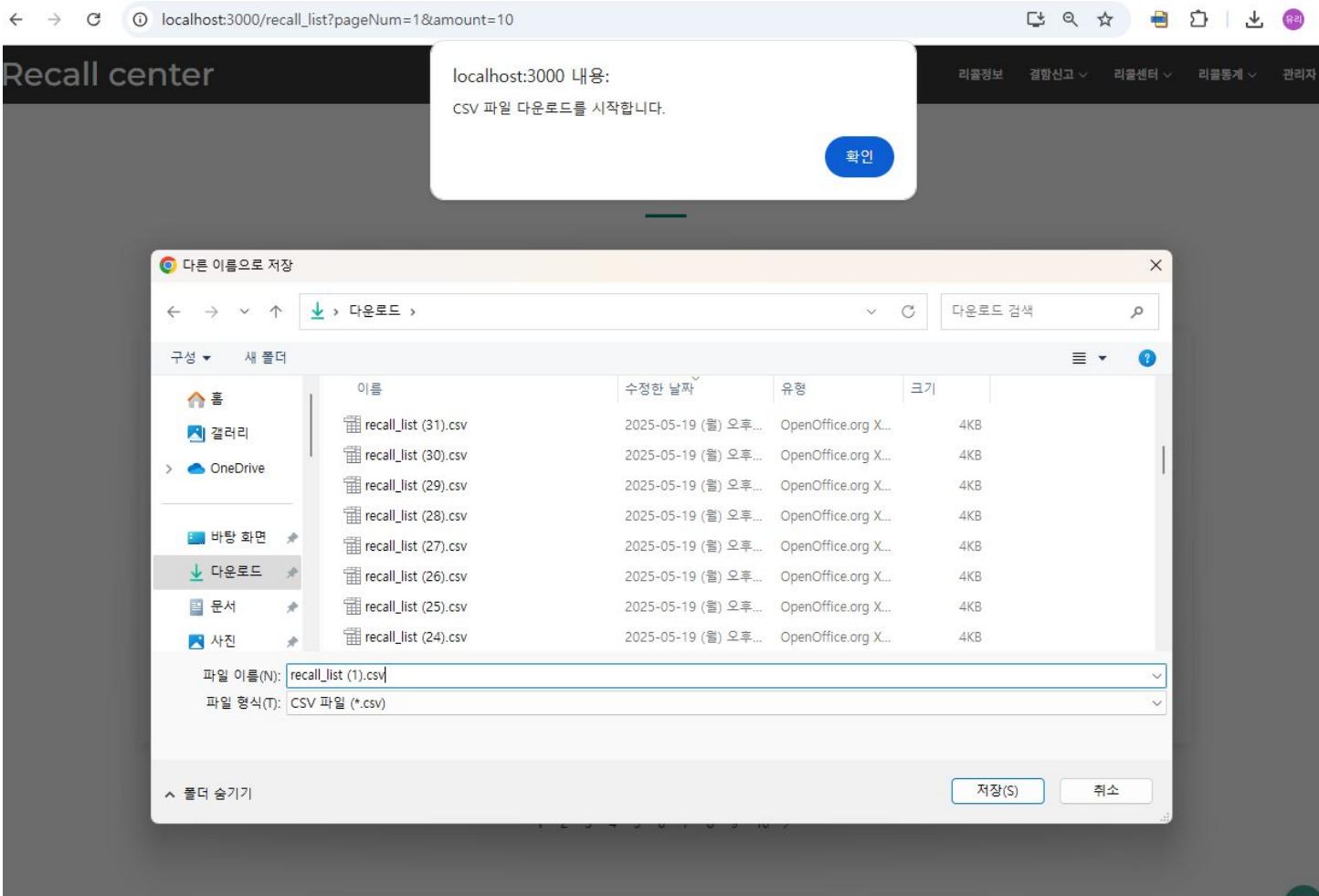
빅데이터검색 플랫폼 프로젝트
Logistic Regression

```
// CSV 전체 다운로드 API
@GetMapping("/recall/downloadCsv")
public ResponseEntity<byte[]> downloadRecallCsv() {
    try {
        byte[] csvBytes = recallService.generateCsvReport(); // 서비스에서 CSV 데이터 생성
        return ResponseEntity.ok()
            .header("Content-Disposition", "attachment; filename=\"recall_list.csv\"")
            .contentType(org.springframework.http.MediaType.parseMediaType("text/csv"))
            .body(csvBytes);
    } catch (IOException e) {
        log.error("@# CSV 파일 생성 또는 다운로드 중 오류 발생: {}", e.getMessage(), e);
        return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
    }
}

// 엑셀 전체 다운로드 API
@GetMapping("/recall/downloadExcel")
public ResponseEntity<byte[]> downloadRecallExcel() {
    try {
        byte[] excelBytes = recallService.generateExcelReport(); // 서비스에서 엑셀 데이터 생성
        return ResponseEntity.ok()
            .header("Content-Disposition", "attachment; filename=\"recall_list.xlsx\"")
            .contentType(org.springframework.http.MediaType.parseMediaType("application/vnd.openxmlformats-officedocument.spreadsheetml.sheet"))
            .body(excelBytes);
    } catch (IOException e) {
        log.error("@# 엑셀 파일 생성 또는 다운로드 중 오류 발생: {}", e.getMessage(), e);
        return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

CSV/ Excel 다운로드

빅데이터검색 플랫폼 프로젝트
Logistic Regression



CSV/ Excel 다운로드

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

☰

검색 유형 선택 ▾ 검색어를 입력하세요 검색

리콜 ID	제품명	제조사	제조기간	기타정보
851	계기판 관련	Chevrolet	2025-05-07 ~...	ddd
850	타이어	Tesla		타이어 3개 터짐으로 수정
849	[현대] 일렉시티 수소...	현대자동차(주)	20190515~20...	수소 배출구 보호마개 ...
848	[재규어랜드로버] 레...	(주)재규어랜드...	2017-05-02 ~...	
847	[지엠코리아] 캐딜락 ...	지엠코리아주...	2004-09-17 ~...	
846	[토요타] 토요타 시에...	한국토요타자...	2017-10-18 ~...	
845	[이탈로모토(유)] BEV...	이탈로모토(유)	2016-06-01 ~...	
844	[화창상사] ROADMA...	화창상사(주)	2017-04-20 ~...	
843	[혼다코리아] ODYSS...	혼다코리아(주)	2013-11-07 ~...	
842	[비엠더블유] 550i 등...	비엠더블유코...	2008-05-08 ~...	

◀ ▶

CSV 전체 다운로드 엑셀 전체 다운로드

1 2 3 4 5 6 7 8 9 10 >

↑ 💬

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

리콜 상세 정보

리콜 번호	847
제품명	[지엠코리아] 캐딜락 STS - 리어서스펜션 토우링크 관련 리콜
제조사	지엠코리아주식회사
제조기간	2004-09-17 ~ 2004-10-25
기타정보	
모델명	캐딜락 STS
리콜유형	자발적리콜
연락처	지엠코리아 고객센터 080-3000-5000

유사 리콜 추천

748번 리콜 보기 747번 리콜 보기 797번 리콜 보기 457번 리콜 보기 190번 리콜 보기

목록으로

1. 리콜 유사도 기반 추천 기능

사용자가 특정 리콜 상세 페이지(`/recall_detail/{id}`)를 조회할 때,
해당 리콜과 유사한 리콜들을 자동으로 추천하는 기능.

이 기능은 Flask로 배포된 Python 서버에서 [TF-IDF + 코사인 유사도] 방식으로 구현됨.

TF-IDF 벡터화란?

TF-IDF는 문서에서 중요한 단어를 수치화해주는 통계적 방법으로, 텍스트를 벡터로 변환하는데 사용됨.

- **TF** (Term Frequency)
 - 특정 단어가 문서에 얼마나 자주 등장하는지
 - 단순히 빈도만 고려
 - **IDF** (Inverse Document Frequency)
 - 전체 문서에서 얼마나 희귀한 단어인지
 - 너무 흔한 단어(예: "the", "is")는 중요도가 낮아짐
- 두 값을 곱해서 ‘TF-IDF 점수’ 계산
- 문서를 수치화된 벡터로 표현 가능

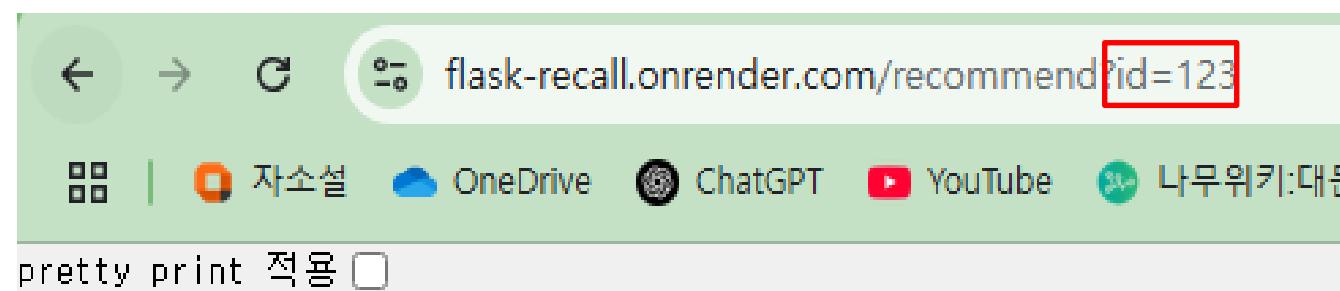
유사리콜추천

빅데이터검색 플랫폼 프로젝트
Logistic Regression

```
1  from flask import Flask, request, jsonify
2  import pandas as pd
3  from sklearn.feature_extraction.text import TfidfVectorizer
4  from sklearn.metrics.pairwise import cosine_similarity
5
6  app = Flask(__name__)
7
8  # 데이터 불러오기
9  df = pd.read_csv('recall.csv') → 1. CSV에서 리콜 데이터 불러옴
10
11 # 추천용 문서 만들기
12 def build_document(row):
13     return f"{row['manufacturer']} {row['model_name']} {row['recall_type']} {row['additional_info']}"
14
15 documents = df.apply(build_document, axis=1)
16 vectorizer = TfidfVectorizer()
17 tfidf_matrix = vectorizer.fit_transform(documents) → 2. TF-IDF 벡터화
18 print(f"[INFO] 전체 문서에서 추출된 고유 단어 수 (벡터 차원 수): {len(vectorizer.get_feature_names_out())}")
19
20 @app.route('/recommend', methods=['GET'])
21 def recommend():
22     target_id = int(request.args.get('id'))
23     if target_id not in df['id'].values:
24         return jsonify([])
25
26     idx = df.index[df['id'] == target_id][0]
27     target_vec = tfidf_matrix[idx]
28     similarities = cosine_similarity(target_vec, tfidf_matrix).flatten() → 3. 코사인 유사도로 유사도 계산
29
30     # 자기 자신 제외
31     similarities[idx] = -1
32
33     top_indices = similarities.argsort()[:-1][:5] → 4. 가장 유사한 리콜 ID 5개 반환
34     result_ids = df.iloc[top_indices]['id'].tolist()
35     return jsonify(result_ids)
36
37 if __name__ == '__main__':
38     app.run(debug=True, port=5000)
39
```

유사리콜추천

빅데이터검색 플랫폼 프로젝트
Logistic Regression



[180, 73, 365, 462, 426] → ID=123 기준 유사도 추천 ID 5개

```
@Override
public List<Integer> getSimilarRecallIds(Long targetId) {
    String apiUrl = flaskBaseUrl + "/recommend?id=" + targetId;

    RestTemplate restTemplate = new RestTemplate();

    ResponseEntity<Integer[]> response = restTemplate.getForEntity(apiUrl, Integer[].class);
    Integer[] ids = response.getBody();

    return Arrays.asList(ids);
}
```

RestTemplate 를 사용해 Flask 서버에 HTTP GET 요청을 보냄
→ 응답을 Integer[] 로 받아서 List<Integer> 로 변환

유사리콜추천

빅데이터검색 플랫폼 프로젝트
Logistic Regression

이후 List 데이터를 RecallDetail.js 에서 구현

```
const data = await fetchRecallDetail(id);

setRecall(data.recall); // 리콜 DTO (혹은 `Defect_DetailsDTO`)
setSimilarIds(data.similarIds || []); // 유사 리콜 ID 배열

/* 유사 리콜 추천 */
<div style={{ marginTop: '30px' }}>
  <div className="section-title text-center">
    <h3 className="title">유사 리콜 추천</h3>
  </div>
  <div className='center-group-container'>
    {similarIds.length > 0 ? (
      similarIds.map(sid => (
        <Link
          key={sid}
          to={`/recall_detail/${sid}`}
          style={{
            display: 'inline-block', // 인라인 블록 요소로 만들어 크기 조절 가능하게
            padding: '8px 15px', // 패딩으로 버튼처럼 보이게
            backgroundColor: '#f0f0f0', // 연한 회색 배경
            color: '#333', // 진한 글씨색
            textDecoration: 'none', // 밑줄 제거
            borderRadius: '5px', // 모서리 둥글게
            border: '1px solid #ddd', // 얇은 테두리
            fontSize: '0.9rem', // 글씨 크기 조절
            transition: 'background-color 0.3s ease', // 호버 효과
          }}
          onMouseOver={e => e.currentTarget.style.backgroundColor = '#e0e0e0'}
          onMouseOut={e => e.currentTarget.style.backgroundColor = '#f0f0f0'}
        >
          {sid}번 리콜 보기
        </Link>
      )
    ) : (
      <p>해당하는 유사 리콜이 없습니다.</p>
    )}
  </div>
</div>
```

리콜 번호	123
제품명	[볼보트럭] FL 카고 - 엔진 컨트롤 유닛 소프트웨어 관련 리콜
제조사	볼보트럭코리아(주)
제조기간	20150511~20160421
기타정보	엔진 컨트롤 유닛 소프트웨어 업데이트
모델명	FL 카고
리콜유형	자발적리콜
연락처	볼보트럭코리아 대표번호 080-038-1000

유사 리콜 추천

180번 리콜 보기 73번 리콜 보기 365번 리콜 보기 462번 리콜 보기 426번 리콜 보기

목록으로

The screenshot shows a web browser window titled "Recall center" from the URL "localhost:3000/recall_list?pageNum=1&amount=10". The interface includes a search bar with dropdown options for "검색 유형 선택" and a text input for "검색어를 입력하세요", followed by a green "검색" button. Below the search bar is a table listing 10 recall entries. The table columns are: 리콜 ID, 제품명, 제조사, 제조기간, 기타정보, 모델명, 리콜유형, and 연락처. Each row contains a link to the specific recall details. On the right side of the table, there are two teal circular buttons with white icons: an upward arrow and a speech bubble.

리콜 ID	제품명	제조사	제조기간	기타정보	모델명	리콜유형	연락처
849	계기판 리콜	Renault Samsung	2023-01-23 ~ 2...	결함입니다	Model 3	자발적리콜	[마세라티] 대표...
848	[재규어랜드로버] 레인...	(주)재규어랜드...	2017-05-02 ~ 2...		레인지로버 벨라	자발적리콜	재규어랜드로버...
847	[지엠코리아] 캐딜락 S...	지엠코리아주식...	2004-09-17 ~ 2...		캐딜락 STS	자발적리콜	지엠코리아 고...
846	[토요타] 토요타 시에...	한국토요타자동...	2017-10-18 ~ 2...		토요타 시에나 2WD, 토요타 시에...	자발적리콜	한국토요타자동...
845	[이탈로모토(유)] BEVE...	이탈로모토(유)	2016-06-01 ~ 2...		BEVERLY 350 SPORT TOURING ...	자발적리콜	이탈로모토 대...
844	[화창상사] ROADMAS...	화창상사(주)	2017-04-20 ~ 2...		ROADMASTER ELITE, ROADMAS...	자발적리콜	화창상사 고객...
843	[혼다코리아] ODYSSEY...	혼다코리아(주)	2013-11-07 ~ 2...		ODYSSEY	자발적리콜	혼다코리아 고...
842	[비엠더블유] 550i 등 1...	비엠더블유코리...	2008-05-08 ~ 2...		550i, 550i xDrive, 650i Convertib...	자발적리콜	비엠더블유 커...
841	[비엠더블유] Ghost - ...	비엠더블유코리...	2010-11-18 ~ 2...		Rolls-Royce Ghost	자발적리콜	롤스로이스 서...
840	[한불모터스] Citroen ...	한불모터스(주)	2015-04-14 ~ 2...		Citroen Grand C4 Picasso 1.6 BI...	자발적리콜	한불모터스 대...

결함신고-신고자

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

정보 입력

신고자 정보

신고인	<input type="text"/>
생년월일	예: 19990101
휴대번호	예: 010-1234-5678
전화번호	예: 02-1234-5678
주소	<input type="text"/> 우편번호
기본주소	<input type="text"/>
공개여부	<input checked="" type="radio"/> 공개 <input type="radio"/> 비공개
비밀번호	<input type="password"/>

비밀번호는 9~15자의 영문/숫자/특수문자(~, !, @, #, \$, *, (), =, +, _ , .) 혼용만 가능합니다.

자동차 정보

신고유형	<input checked="" type="radio"/> 자동차 <input type="radio"/> 이륜
자동차 등록번호	예 :서울00나0000
자동차 모델명	<input type="text"/>
자동차 제조사	<input type="text"/>
제조일자	연도-월-일 0 선택

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

주소

주소	<input type="text"/> 기본주소
공개여부	<input checked="" type="radio"/> 공개 <input type="radio"/> 비공개
비밀번호	<input type="password"/> 선택

비밀번호는 9~15자의 영문/숫자/특수문자(~, !, @, #, \$, *, (), =, +, _ , .) 혼용만 가능합니다.

자동차 정보

신고유형	<input checked="" type="radio"/> 자동차 <input type="radio"/> 이륜
자동차 등록번호	예 :서울00나0000
자동차 모델명	<input type="text"/>
자동차 제조사	<input type="text"/>
제조일자	연도-월-일 0 선택

결합 신고하기

© 2025 My App

Recall center

리콜정보 결함신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

신고 내역

전체	키워드를 입력하세요	연도-월-일 0	검색	
번호	신고자	신고유형	모델명	신고일
1003	이용염	자동차	볼보트럭	2025-05-23
1002	신고자999	제조결함	Wrangler	2004-02-09
1001	신고자998	제조결함	i4	2020-02-22
1000	신고자997	제조결함	Sportage	2019-01-23
999	신고자996	제조결함	Equinox	2015-12-09
998	신고자995	제조결함	Ray	2021-02-13
997	신고자994	제조결함	Macan	2008-03-08
996	신고자993	제조결함	Genesis	2004-12-08
995	신고자992	제조결함	S60	2015-06-11
994	신고자991	제조결함	Stinger	2022-01-06

신고 상세 내역

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

신고 상세 내역

번호	1003	신고자	이용연	신고일	2025-05-23
----	------	-----	-----	-----	------------

신고유형	자동차
모델명	볼보트럭
제조사	볼보트럭
제조일자	2025-05-20

수정

목록보기

결함신고 상세 내역 수정시 비밀번호 확인

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

비밀번호 확인

결함신고시 등록한 비밀번호를 입력하세요.

확인

돌아가기

신고 내역 수정

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

신고 내역 수정

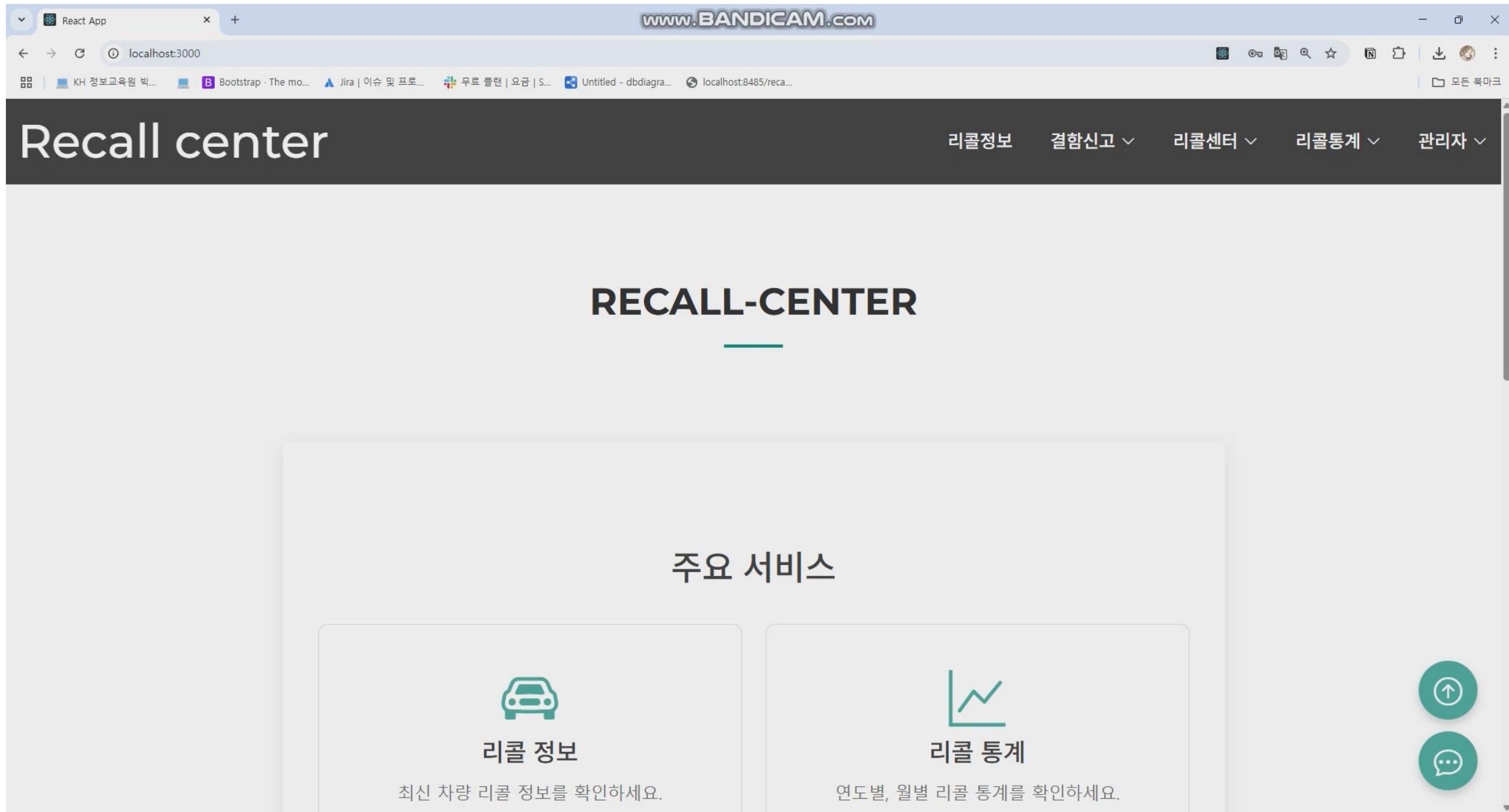
번호	1002	신고자	신고자999	신고일	2004-02-09
				차량 제조일자	2022-07-26

신고유형	제조결함
모델명	Wrangler
제조사	Mercedes-Benz

[수정](#) [삭제](#) [목록보기](#)

결함신고 상세 내역 수정시 비밀번호 확인

빅데이터검색 플랫폼 프로젝트
Logistic Regression



Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

공지사항

전체 ▾

검색어를 입력하세요

검색

번호	제목	작성시간
6	안녕하세요 테스트중입니다	25-05-23-오전10:26
5	dd	25-05-22-오후05:14
4	테스트 공지사항 추가	25-05-21-오후04:03
3	○ ○ ○	25-05-21-오후04:01
2	lemon prince	25-05-21-오후01:49
1	○ ○	25-05-21-오후03:10

공지사항 - 출력 / 이전글 다음글 이동

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

공지사항

안녕하세요 테스트중입니다

작성일: 25-05-23-오전10:26

테스트

이전글 다음글

목록으로 돌아가기

F&Q – 출력 / 전체,제목,내용별 검색 / 페이지

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

자주 묻는 질문들

Q. 안녕하세요 궁금한 것이 있습니다. ▼

오늘은 빠른 퇴근이 가능할까요?

Q. 자동차 관리시 주의점 >

Q. ○○ >

전체 ▾ 검색어를 입력하세요 검색

[질문 작성하기](#)

F&Q 작성

빅데이터검색 플랫폼 프로젝트
Logistic Regression



FAQ 작성

고객	2025. 5. 23. 오후 2:14:29
질문	
내용	

FAQ 작성하기

목록으로

The screenshot shows a web browser window with the URL localhost:3000 in the address bar. The page title is "Recall center". The top navigation bar includes links for "리콜정보", "결합신고", "리콜센터", "리콜통계", and "관리자". The main heading on the page is "RECALL-CENTER". Below it, there is a section titled "주요 서비스" (Main Services) featuring two cards: "리콜 정보" (Recall Information) with a car icon and "리콜 통계" (Recall Statistics) with a chart icon. To the right of these cards are two teal circular icons with white symbols: an upward arrow and a speech bubble.

www.BANDICAM.com

localhost:3000

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

RECALL-CENTER

주요 서비스

리콜 정보

최신 차량 리콜 정보를 확인하세요.

리콜 통계

연도별, 월별 리콜 통계를 확인하세요.

리콜 통계 - 연도별 확인

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

리콜정보 결함신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

리콜 통계

연도별 리콜 통계 신고 현황

시작 연도 종료 연도

-- 선택 -- -- 선택 -- 조회

결함 신고 요약 통계(2000 ~ 2025 전체)

해당 연도	국산자동차		수입자동차		계	
	차종	대수	차종	대수	차종	대수
계	22	37	21	43	26	80

자세히 보기 +

리콜 통계 - 연도별 국산 / 수입 막대그래프 시각화

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

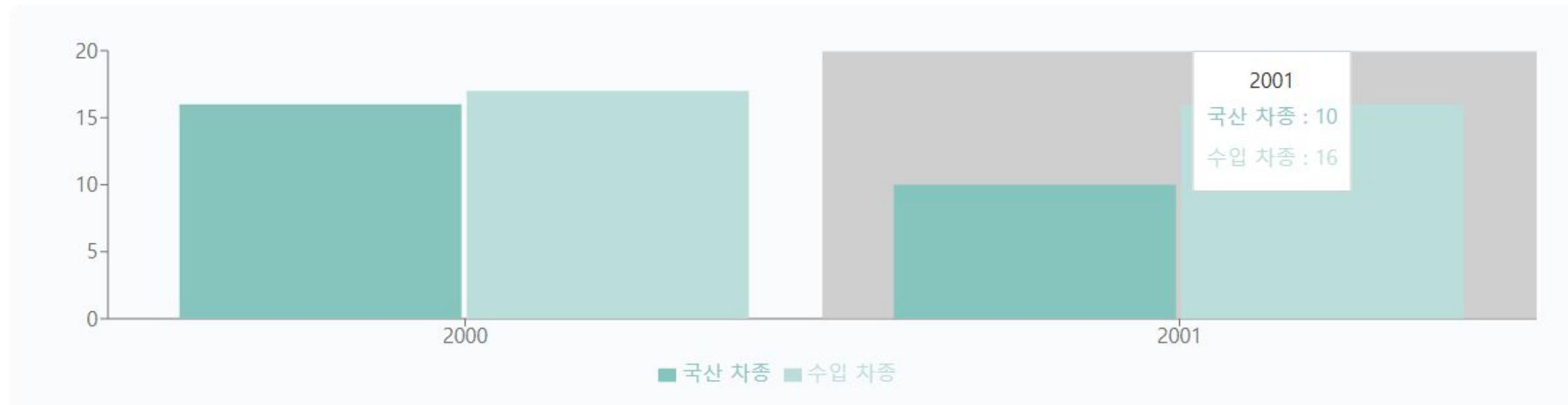
리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

자세히 보기 ▾

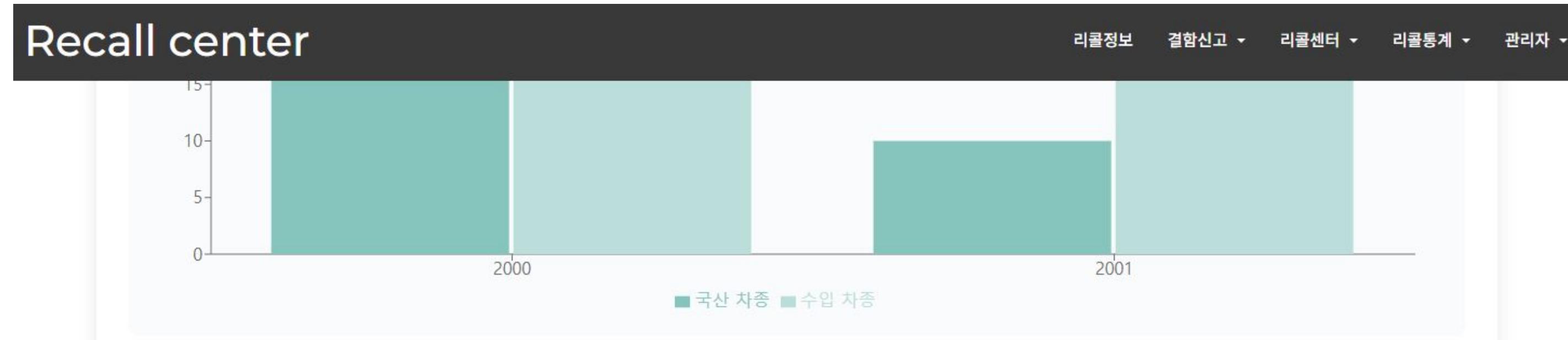
연도별 리콜 현황

해당 연도	국산자동차		수입자동차		계	
	국산 차종	국산 대수	수입 차종	수입 대수	전체 차종	전체 대수
2000	16	22	17	22	24	44
2001	10	15	16	21	20	36

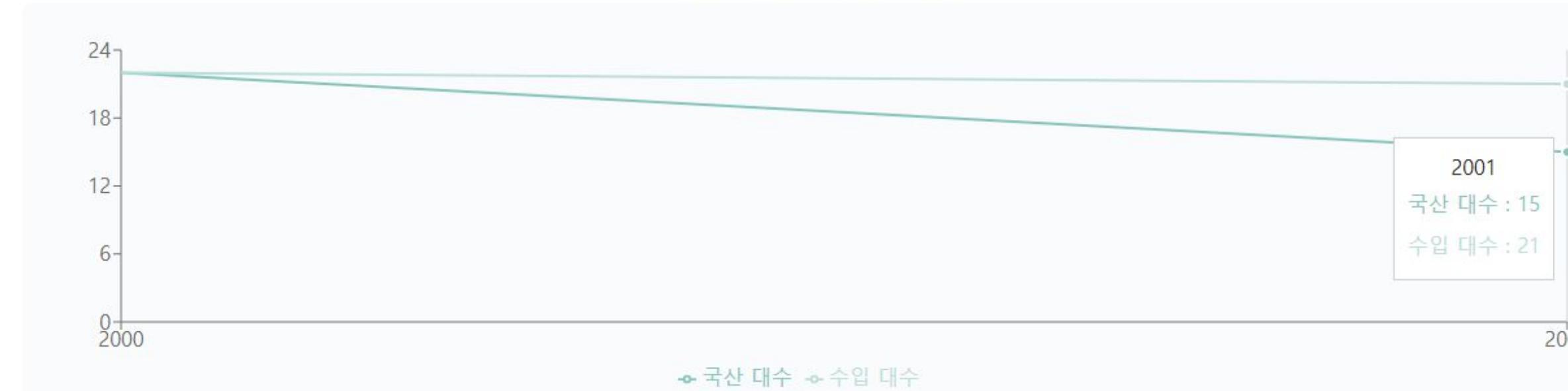
연도별 리콜 차종



리콜 통계 - Ai Gemini 분석된 PDF 다운로드



연도별 리콜 대수



PDF 다운로드

·리콜 통계 - **Ai Gemini** 분석된 **PDF** 다운로드

로컬에서만 가능합니다 !

- Pdf다운받기 위한 wkhtmltopdf 로컬에 설치

```
#generate pdf  
wkhtmltopdf.executable=C:/Program Files/wkhtmltopdf/bin/wkhtmltopdf.exe
```

리콜 통계 - Ai Gemini 분석된 PDF 다운로드

리액트 pdf버튼 핸들러

```
const PdfDownloadButton = ({ pdfBaseUrl, startYear, endYear, startMonth, endMonth }) => [
  Windsurf: Refactor | Explain | Generate JSDoc | X
  const handleDownload = () => {
    const params = [];
    if (startYear) params.push(`startYear=${startYear}`);
    if (endYear) params.push(`endYear=${endYear}`);
    if (startMonth) params.push(`startMonth=${startMonth}`);
    if (endMonth) params.push(`endMonth=${endMonth}`);
    // 최종 PDF를 생성할 API URL (예: http://localhost:8485/pdf/recall_statics_summaryList)
    const dataUrl = `${pdfBaseUrl}?${params.join('&')}`;
    // PDF 생성을 요청할 서버 엔드포인트 (실제 PDF 파일을 내려주는 역할)
    // 이 URL은 서버에서 `dataUrl`을 받아서 PDF로 변환 후 응답해야 합니다.
    const generatePdfEndpoint = 'http://localhost:8485/api/generatePdfFromUrl';
    // 동적으로 form 생성하여 PDF 다운로드 트리거
    const form = document.createElement('form');
    form.method = 'GET';
    form.action = generatePdfEndpoint;
    form.target = '_self'; // 현재 창에서 다운로드 (새 창을 원하면 '_blank'로 변경)
    const input = document.createElement('input');
    input.type = 'hidden';
    input.name = 'url';
    input.value = dataUrl;
    form.appendChild(input);
    document.body.appendChild(form);
    form.submit();
    document.body.removeChild(form); // form 제출 후 제거
  };
]
```

pdfBaseUrl: PDF로 변환할 웹 페이지의 기본 URL
 Pdf로 만들기 위한 페이지가 존재
 Ex> http://localhost:8485/api/recall_statics_summaryList

이 페이지 뒤에 쿼리스트링으로 파라미터를 보낸다
 input.name = 'url'; url 이름으로 보내는 text가 이 링크임!
 Ex> http://localhost:8485/api/recall_statics_summaryList?&startyear=@@@&endyear=@@@

Url 넣으면 그페이지를 pdf로 만들어주는 형식이기 때문에
 그 주소로 form action 해야함

const form = document.createElement('form');에서 보내는 endpoint 링크
 const generatePdfEndpoint = 'http://localhost:8485/api/generatePdfFromUrl'

form.submit();
 하고 폼 제출

리콜 통계 - Ai Gemini 분석된 PDF 다운로드

컨트롤러

```
@GetMapping("/generatePdfFromUrl")
public void generatePdfFromUrl(@RequestParam("url") String url, HttpServletResponse response) throws IOException, Interrupt
    log.info("generatePdfFromUrl");
    File pdfFile = pdfGenerationService.generatePdfFromUrl(url, "generated_from_url.pdf");
    sendFileToClient(pdfFile, response, "generated_from_url.pdf");
}
```

- 그래서 /generatedPdfFromUrl에서
- Pdf로 생성할 url(<http://localhost:8485/api/recallStaticsSummaryList?startYear=@@@&endYear=@@@>)을 받아서 서비스 호출해서 pdf를 만듭니다

서비스

```
public File generatePdfFromUrl(String url, String outputFileName) throws IOException, InterruptedException {
    File outputFile = new File(outputFileName);
    List<String> command = new ArrayList<>();
    command.add(wkhtmltopdfPath);
    command.add(url);
    command.add(outputFile.getAbsolutePath());

    ProcessBuilder processBuilder = new ProcessBuilder(command);
    Process process = processBuilder.start();
    int exitCode = process.waitFor();

    if (exitCode != 0) {
        throw new IOException("wkhtmltopdf exited with error code: " + exitCode);
    }

    return outputFile;
}
```

이거는 jsp
그래서 이 pdf링크를 탈때
Gemini로 데이터를 분석해달라고 부탁



```
Windsurf: Refactor | Explain | Generate Javadoc | X
@RequestMapping("/pdf/recall_statics_summaryList")
public String recall_statics_pdf(
    @RequestParam(required = false) Integer startYear,
    @RequestParam(required = false) Integer endYear,
    Model model) {

    Map<String, Object> paramMap = new HashMap<>();
    if (startYear == null || startYear == 0) {
        startYear = 2000;
    }
    if (endYear == null || endYear == 0) {
        endYear = 2025;
    }
    paramMap.put("start_year", startYear);
    paramMap.put("end_year", endYear);

    //리콜현황
    DefectReportSummaryDTO summary = recallService.getDefectReportSummary(po
    model.addAttribute("summary", summary);
    List<DefectReportSummaryDTO> summaryList = recallService.getDefectRepor
    model.addAttribute("summaryList", summaryList);

    String predefinedQuestion = "이 자료들은 연도별 리콜현황에 관한 자료인데
    + summaryList;
    // Gemini API를 호출하여 predefinedQuestion에 대한 답변 얻기
    String geminiAnswer = geminiService.askGemini(predefinedQuestion);
    model.addAttribute("answer", geminiAnswer);

    return "pdf/recall_statics_summaryList";
}
```

리콜 통계 - 연도별 PDF

빅데이터검색 플랫폼 프로젝트
Logistic Regression

연도별 리콜 통계 신고 현황

결함 신고 요약 통계

해당 연도	국산 자동차		수입 자동차		계	
	차종	대수	차종	대수	차종	대수
계	28	2486	28	1561	29	4047

연도별 리콜현황

해당 연도	국산 자동차		수입 자동차		계	
	국산 차종	국산 대수	수입 차종	수입 대수	전체 차종	전체 대수
2000	24	110	25	79	27	189
2001	27	123	27	74	28	197
2002	25	117	26	71	28	188
2003	25	123	26	85	27	208
2004	25	141	26	72	28	213
2005	23	101	23	69	26	170
2006	20	103	25	77	27	180
2007	23	118	26	79	27	197
2008	22	137	26	72	28	209
2009	24	122	23	55	25	177
2010	23	111	26	92	27	203
2011	22	110	27	67	29	177
2012	23	101	28	73	28	174
2013	22	129	25	83	26	212
2014	22	119	27	76	28	195
2015	24	123	27	71	28	194
2016	23	120	28	77	28	197
2017	24	123	27	70	27	193
2018	23	125	28	68	28	193
2019	26	120	23	83	27	203
2020	21	110	25	68	27	178

Gemini's summarize

**html

국산 및 수입 차 리콜 위험 점수:

리콜 건수와 차종 수를 바탕으로 위험 점수를 평가할 수 있습니다. 국산 차의 경우 리콜 건수는 수입 차보다 높게 나타나는 경향이 있으며, 이는 국내 자동차 시장 규모와 생산량, 그리고 운행 대수가 수입 차보다 많기 때문일 수 있습니다. 하지만 수입 차의 경우, 차종당 리콜 건수가 높다면 특정 차종의 결함이 광범위하게 발생하고 있을 가능성이 있습니다. 따라서, 단순히 리콜 건수뿐만 아니라 차종 수를 함께 고려하여 위험 점수를 산출해야 합니다. 예를 들어, 특정 연도에 수입 차 리콜 차종 수는 적지만 리콜 건수가 높다면, 해당 차종에 대한 집중적인 관리가 필요합니다.

주요 리콜 국산 및 수입 현황:

제공된 데이터를 살펴보면, 국산 차는 매년 20~27개 차종에서 101~141건의 리콜이 발생하고 있습니다. 수입 차는 23~28개 차종에서 55~92건의 리콜이 발생합니다. 전반적으로 국산 차의 리콜 건수가 더 많지만, 차종 수를 고려하면 수입 차도 결코 낮은 수치는 아닙니다. 특정 연도에 리콜 건수가 급증하는 경우, 해당 연도에 출시된 신차 또는 특정 부품의 결함 발생 가능성을 의심해 볼 수 있습니다. 따라서, 연도별 리콜 현황을 지속적으로 모니터링하고, 특정 차종 또는 부품에서 반복적으로 결함이 발생하는지 파악하는 것이 중요합니다.

자동차 리콜의 중요성:

자동차 리콜은 제조사의 결함으로 인해 발생할 수 있는 안전 문제를 예방하고 소비자 안전을 확보하는 데 매우 중요합니다. 리콜은 잠재적인 사고를 미연에 방지하고, 소비자의 생명과 재산을 보호하며, 자동차 산업의 신뢰성을 유지하는데 필수적인 절차입니다. 리콜 정보는 소비자에게 투명하게 공개되어야 하며, 제조사는 리콜 대상 차량 소유자에게 신속하게 통지하고, 무상 수리를 제공해야 합니다. 또한, 정부는 리콜 제도를 강화하고, 제조사의 책임성을 높여 안전한 자동차 운행 환경을 조성해야 합니다.

Contact Us

부산광역시 부산진구 중앙대로 672 2

상비빌딩

2F, 12F

Phone: 010-1234-5678

Email: contact@example.com

© KH정보교육원 KH리콜안전공단 자동차안전연구원

Designed by team KH리콜안전공단

리콜 통계 – 제조사별 리콜 요약 통계

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

제조사별 리콜 요약 통계

제조사명	계
Volvo Cars	4
Stellantis	10
Kia	7
Chevrolet	8
Tesla	3
Porsche	6
Mercedes-Benz	6
Genesis	11
Hyundai	11
BMW	6
Renault Samsung	8

자세히 보기 +

리콜 통계 - 제조사별 연도별 리콜 상세

빅데이터검색 플랫폼 프로젝트
Logistic Regression

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

자세히 보기 -

제조사별 연도별 리콜 상세

해당 연도	제조사명	계
합산 +	Volvo Cars	7
합산 +	Stellantis	5
합산 +	Kia	4
합산 +	Chevrolet	11
합산 +	Tesla	7
합산 +	Porsche	4
합산 +	Mercedes-Benz	7
합산 +	Genesis	10
합산 +	Hyundai	12
합산 +	BMW	4
합산 +	Renault Samsung	4

제조사별 리콜 대수 비율



리콜 통계 – 제조사별 Ai Gemini 분석된 PDF 다운로드

Recall center

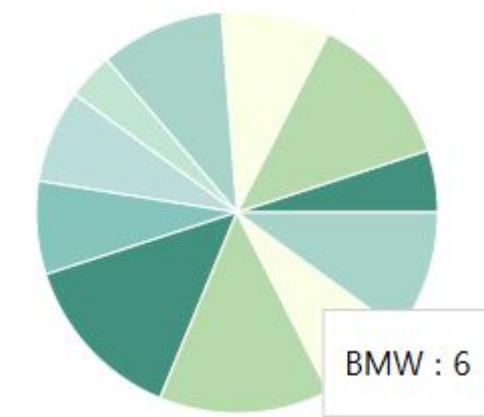
리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

합산 +

Renault Samsung

8

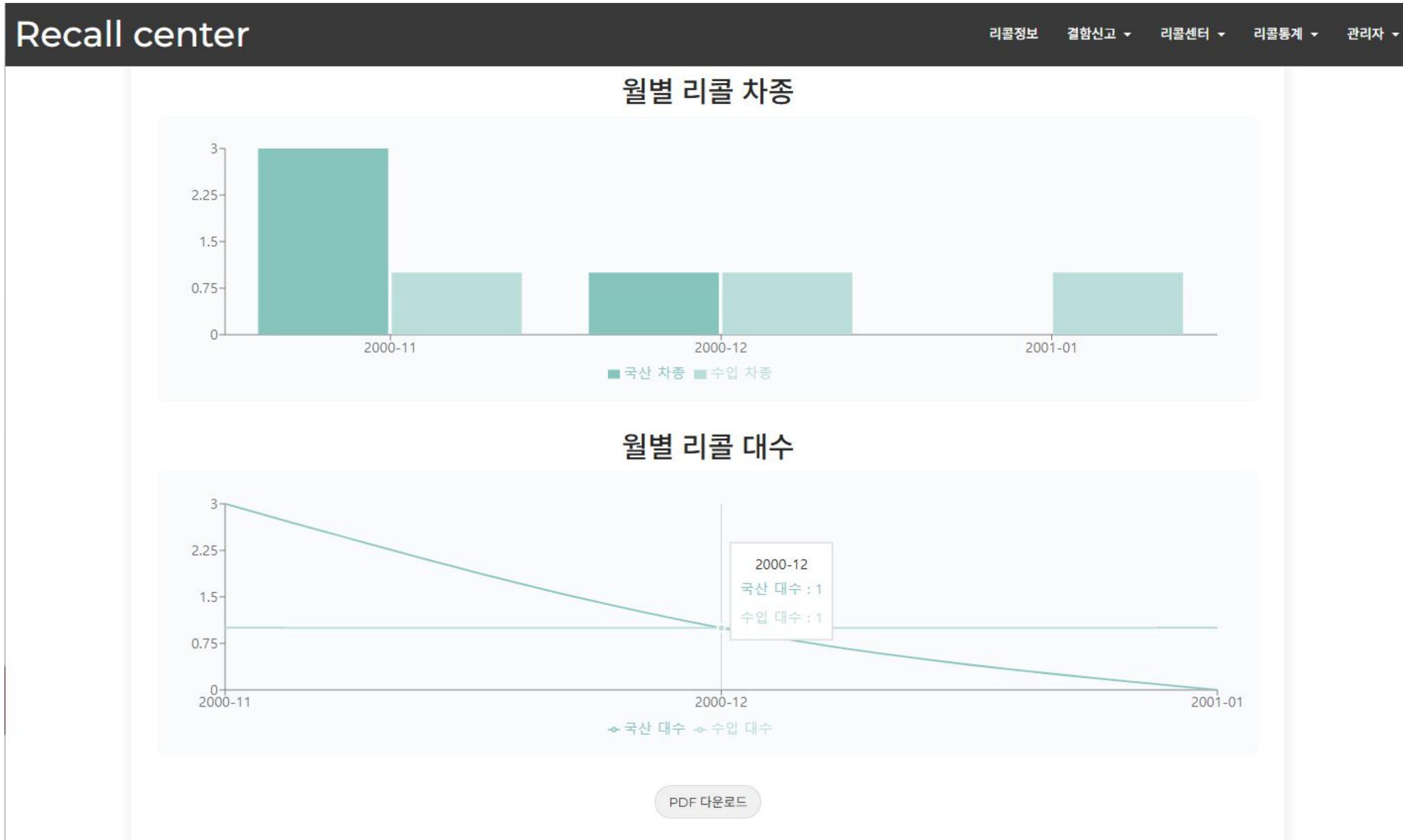
제조사별 리콜 대수 비율



● Volvo Cars ● Stellantis ● GM ● Chevrolet ● Tesla ● Porsche ● Mercedes-Benz ● Genesis ● Hyundai ● Kia ● Renault Samsung

PDF 다운로드

리콜 통계 – 월별 리콜통계 차트 출력



리콜 통계 – 월별 제조사별 출력

Recall center

[리콜정보](#) [결함신고](#) [리콜센터](#) [리콜통계](#) [관리자](#)

월별 리콜 현황 - 제조사별

해당 연도-월	제조사명	계
합산 +	Volvo Cars	4
합산 +	Stellantis	4
합산 +	Kia	2
합산 +	Chevrolet	3
합산 +	Tesla	4
합산 +	Porsche	3
합산 +	Mercedes-Benz	4
합산 +	Genesis	3
합산 +	Hyundai	8
합산 +	BMW	3
합산 +	Renault Samsung	2

리콜 통계 – 제조사별 리콜대수 비율 / PDF 다운로드

Recall center

- 리콜정보
- 결함신고
- 리콜센터
- 리콜통계
- 관리자

합산 +	Hyundai	8
합산 +	BMW	3
합산 +	Renault Samsung	2

제조사별 리콜 대수 비율

제조사	비율
Volvo Cars	1
Stellantis	2
Chevrolet	3
Tesla	3
Porsche	4
Mercedes-Benz	4
Genesis	4
Hyundai	8
Renault Samsung	9

PDF 다운로드

리콜 통계 - 중복 리콜모델 목록 / 모델및 횟수 확인

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

중복 리콜 모델 목록

모델명	횟수
GLE 450 4MATIC	14회
BMW 530i xDrive	12회
S 350 d 4MATIC	12회
GLC 220 d 4MATIC	12회
S 350 d	11회
E 220 d 4MATIC	11회
GLE 300 d 4MATIC	11회
BMW 520i	10회
S 450 4MATIC	9회
타이칸 4S	9회
카이엔	9회
Model Y	9회
타이칸	9회
ODYSSEY	9회
Maybach S 580 4MATIC	9회
S 580 4MATIC	9회
S 400 d 4MATIC	8회
TGS 카고	8회
BMW 523d	8회

·리콜 통계

The screenshot shows a web browser window titled "Recall center" at "localhost:3000". The page features a dark header bar with the title "Recall center" on the left and navigation items "리콜정보", "결합신고", "리콜센터", "리콜통계", and "관리자" on the right. Below the header are four main sections arranged in a 2x2 grid:

- 리콜 정보**: Represented by a car icon. Description: "최신 차량 리콜 정보를 확인하세요.".
- 리콜 통계**: Represented by a line graph icon. Description: "연도별, 월별 리콜 통계를 확인하세요."
- 결합 신고**: Represented by a pencil icon. Description: "차량 결합 신고 및 내역을 조회합니다."
- 신고 내역**: Represented by a document icon. Description: "차량 결합 신고 및 내역을 조회합니다."

In the bottom right corner, there are two teal circular icons: one with an upward arrow and another with a speech bubble containing three dots.

·리콜 통계 - 중복 리콜모델 목록 / 모델및 횟수 확인

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Recall center". The main content area displays a heading "중복 리콜 모델 목록" and a table with the following data:

모델명	횟수
GLE 450 4MATIC	14회
BMW 530i xDrive	12회
S 350 d 4MATIC	12회
GLC 220 d 4MATIC	12회
S 350 d	11회
E 220 d 4MATIC	11회
GLE 300 d 4MATIC	11회

On the right side of the table, there are two teal-colored circular icons: one with an upward arrow and another with a speech bubble.

·관리자- 관리자 로그인



로그인 페이지

관리자 로그인

· 관리자- 관리자 로그인 / jwt 토큰 인증

[1] 로그인 성공 시, JWT 토큰 발급

↓
[2] 클라이언트가 받은 토큰을 Authorization 헤더에 담아 모든 요청에 포함

↓
[3] JwtInterceptor에서 헤더를 검사하고 토큰 유효성 확인

↓
[4] 통과 시 controller 로직 실행, 실패 시 401 반환

The screenshot shows a web browser window with two main parts. On the left is the 'Recall center' login page, which has a dark header with tabs like '리콜정보', '결합신고', '리콜센터', '리콜통계', and '관리자'. Below the header is a large button labeled '관리자 페이지'. Underneath the button, there's a message: 'KH 리콜정보 관리자 시스템에 오신 것을 환영합니다. 관리자 님'. At the bottom of the button are three buttons: 'API 저장', 'API 동기화', and '로그아웃'. On the right side of the browser is the 'Chrome DevTools' interface, specifically the 'Application' tab. The 'Storage' section is expanded, showing 'Local storage' with an item for the origin 'https://recall-final-backend.onrender.com'. This item contains a key 'jwt_token' with a long value starting with 'eyJhbGciOiJIUzI1NiJ9.eyJzdWl...'. The 'Origin' section also lists 'https://recall-final-backend.onrender.com'. The 'Key' and 'Value' columns are clearly visible.

· 관리자- 관리자 로그인 / jwt 토큰 인증

```
package com.boot.security;

import javax.servlet.http.HttpServletRequest;
@Log4j2
public class JwtInterceptor implements HandlerInterceptor {

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

        // ✅ OPTIONS 요청은 통과
        if ("OPTIONS".equalsIgnoreCase(request.getMethod())) {
            return true;
        }

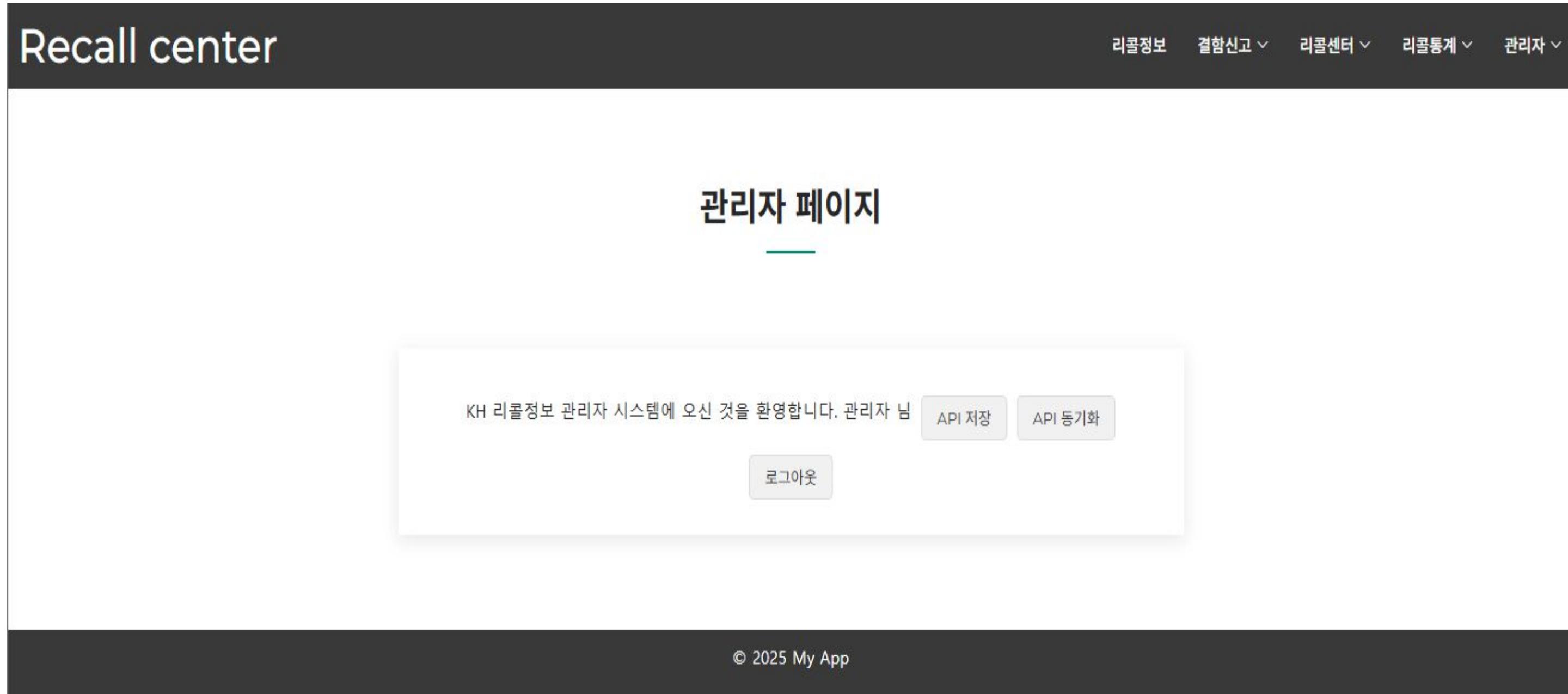
        String authHeader = request.getHeader("Authorization");

        if (authHeader != null && authHeader.startsWith("Bearer ")) {
            String token = authHeader.substring(7); // "Bearer " 이후만 잘라냄
            String adminId = JwtUtil.validateToken(token);

            if (adminId != null) {
                request.setAttribute("adminId", adminId);

                Log.info("Authorization 헤더: {}", authHeader);
                return true;
            }
        }
    }
}
```

·관리자 – api저장 및 동기화



· 관리자 - api저장 및 동기화

```

// 2. API -> DB 전체 저장용
@GetMapping("/save_all")
public ResponseEntity<Map<String, Object>> saveAllToDb() throws Exception {
    String cntntsId = "0301";
    int perPage = 100;

    Criteria cri = new Criteria(1, perPage);
    String firstXml = recallService.fetchXmlFromApi(cri, cntntsId);
    int total = XmlParserUtil.getTotalCount(firstXml);
    int totalPages = (int) Math.ceil((double) total / perPage);

    int savedCount = 0;

    for (int page = 1; page <= totalPages; page++) {
        Criteria pageCri = new Criteria(page, perPage);
        String xml = recallService.fetchXmlFromApi(pageCri, cntntsId);
        List<Defect_DetailsDTO> list = XmlParserUtil.parseToList(xml);
        recallService.saveApiDataToDB(list); _____
        savedCount += list.size();

        Log.info(">> {}페이지 처리 완료 ({})건", page, list.size());
    }

    Map<String, Object> response = new HashMap<>();
    response.put("message", "전체 저장 완료");
    response.put("savedCount", savedCount);
    response.put("totalPages", totalPages);

    return ResponseEntity.ok(response);
}

```

1. 공공 데이터를 xml형식으로 받음
2. xml을 List로 변환
3. DB에 저장

· 관리자 - api저장 및 동기화

```
// 3. 전체 동기화
@GetMapping("/sync_all")
public ResponseEntity<Map<String, Object>> syncAllToDb() throws Exception {
    String cntntsId = "0301";
    int perPage = 100;

    Criteria cri = new Criteria(1, perPage);
    String firstXml = recallService.fetchXmlFromApi(cri, cntntsId);
    int total = XmlParserUtil.getTotalCount(firstXml);
    int totalPages = (int) Math.ceil((double) total / perPage);

    int inserted = 0, updated = 0, skipped = 0;

    for (int page = 1; page <= totalPages; page++) {
        Criteria pageCri = new Criteria(page, perPage);
        String xml = recallService.fetchXmlFromApi(pageCri, cntntsId);
        List<Defect_DetailsDTO> list = XmlParserUtil.parseToList(xml);

        SyncDTO result = recallService.syncApiDataWithDB(list);
        result = recallService.syncApiDataWithDB(list);

        inserted += result.getInserted();
        updated += result.getUpdated();
        skipped += result.getSkipped();

        Log.info("{}페이지 완료: insert {}, update {}, skip {}", page, result.getInserted(), result.getUpdated(), result.getSkipped());
    }

    Map<String, Object> response = new HashMap<>();
    response.put("message", "전체 동기화 완료");
    response.put("inserted", inserted);
    response.put("updated", updated);
    response.put("skipped", skipped);

    return ResponseEntity.ok(response);
}
```

API 데이터를 저장때와 마찬가지로 List 형태로 DTO에 담음
→ DB에 담겨있는 정보와 비교
→ 다른 경우에만 Update

· 관리자 - api저장 및 동기화

```
// API -> DB 등기화
@Override
public SyncDTO syncApiDataWithDB(List<Defect_DetailsDTO> apiList) {
    RecallStaticDAO dao = sqlSession.getMapper(RecallStaticDAO.class);
    SyncDTO result = new SyncDTO();
    int inserted = 0, updated = 0, skipped = 0;

    for (Defect_DetailsDTO dto : apiList) {
        dto.setHash_code(dto.generateHashCode());
```

```
        Defect_DetailsDTO dbData = dao.findByKey(dto);
        if (dbData == null) {
            dao.insertDefect(dto);
            inserted++;
        } else if (!dto.getHash_code().equals(dbData.getHash_code())) {
            dao.updateDefect(dto);
            updated++;
        } else {
            skipped++;
        }
    }

    result.setInserted(inserted);
    result.setUpdated(updated);
    result.setSkipped(skipped);
    return result;
}
```

모든 컬럼값을 비교하지 않고, 주요 컬럼을 통해 HashCode 생성

→ HashCode 값만 비교하여 변경 여부를 통한 Update

→ 동기화 속도 향상&최적화

·관리자 - 신고검수 / 회사 선택 자동입력 / 리콜통계DB입력

Recall center

신고 검수

아이디	아이디를 입력하세요	검색
리콜정보	선택	
	예) 계기판 관련 리콜	
자동차 제조사	예) 블보 자동차 코리아	
기간	연도-월-일 0	
	~	
	연도-월-일 0	
자동차 모델명		
리콜 형식	자발적리콜	
회사(대표번호)	회사 선택	예) 블보자동차 대표번호 1588-17-
상세 결함		

검수 완료!

© 2025 My App

검수등록

빅데이터검색 플랫폼 프로젝트
Logistic Regression

The screenshot shows a web browser window with the URL localhost:3000/defect_list?pageNum=1&amount=10. The page title is "Recall center". The interface includes a search bar with dropdown filters for "전체" (All), "키워드를 입력하세요" (Input keyword), and "연도-월-일" (Year-Month-Day). A green "검색" (Search) button is located to the right of the search bar. Below the search area is a table with 10 rows of data. The table columns are: 번호 (Number), 신고자 (Reporter), 신고유형 (Report Type), 모델명 (Model Name), and 신고일 (Report Date). The data is as follows:

번호	신고자	신고유형	모델명	신고일
7004	신고자999	제조결함	Model 3	2015-12-21
7003	신고자998	제조결함	Ray	2020-12-28
7002	신고자997	제조결함	Stinger	2013-11-08
7001	신고자996	제조결함	C-Class	2020-11-29
7000	신고자995	제조결함	K5	2008-08-02
6999	신고자994	제조결함	C-Class	2000-08-02
6998	신고자993	제조결함	Tucson	2003-08-05
6997	신고자992	제조결함	S60	2000-09-17
6996	신고자991	제조결함	Sonata	2018-07-01
6995	신고자990	제조결함	Model Y	2012-12-05

On the right side of the table, there are two teal circular icons with white symbols: an upward arrow and a speech bubble.

·관리자 – 공지사항 작성 / 작성시간 자동반영

Recall center

리콜정보 결합신고 ▾ 리콜센터 ▾ 리콜통계 ▾ 관리자 ▾

공지사항 작성

고객

2025. 5. 23. 오후 12:23:29

제목

내용

공지사항 작성하기

목록으로

· 관리자 동기화 및 공지사항

The screenshot shows a web browser window titled "React App" with the URL "localhost:3000". The page is titled "Recall center" and features a dark header bar with navigation links: "리콜정보", "결함신고", "리콜센터", "리콜통계", and "관리자". Below the header are six cards arranged in a grid:

- 리콜 정보**: Represented by a car icon. Description: "최신 차량 리콜 정보를 확인하세요."
- 리콜 통계**: Represented by a line graph icon. Description: "연도별, 월별 리콜 통계를 확인하세요."
- 결함 신고**: Represented by a pencil icon. Description: "차량 결함 신고 및 내역을 조회합니다."
- 신고 내역**: Represented by a list icon. Description: "차량 결함 신고 및 내역을 조회합니다."
- 공지사항**: Represented by an info icon. Description: "관련 정보 및 공지사항을 확인하세요."
- FAQ**: Represented by an info icon. Description: "FAQ 목록을 확인하세요."

On the right side of the page, there are two teal circular icons: one with an upward arrow and another with a speech bubble.

- 리액트 모바일에서도
- 호환 가능하게끔 프론트



빅데이터검색 플랫폼 프로젝트
Logistic Regression

•프론트 서버 배포

The screenshot displays the Render platform interface for a project named "Recall_Final_Front".

Left Sidebar:

- Dashboard
- Recall_Final_Front
- Events** (highlighted in purple)
- Settings
- MONITOR
- Metrics
- MANAGE
- Environment
- Previews
- Redirects/Rewrites
- Headers
- Changelog
- Invite a friend
- Contact support
- Render Status

Main Content Area:

Recall_Final_Front (STATIC SITE)

Wjyuy / Recall_Final main
https://recall-final-front.onrender.com

Logs: May 23, 2025 at 4:46 PM (Live)

```

May 23 04:46:56 PM  file sizes after gzip.
May 23 04:46:56 PM  218.59 kB  build/static/js/main.3f427268.js
May 23 04:46:56 PM  10.67 kB  build/static/css/main.e51c3c00.css
May 23 04:46:56 PM  1.77 kB  build/static/js/453.d855a71b.chunk.js
May 23 04:46:56 PM 
May 23 04:46:56 PM  The project was built assuming it is hosted at /.
May 23 04:46:56 PM  You can control this with the homepage field in your package.json.
May 23 04:46:56 PM 
May 23 04:46:56 PM  The build folder is ready to be deployed.
May 23 04:46:56 PM  You may serve it with a static server.
May 23 04:46:56 PM 
May 23 04:46:56 PM  npm install -g serve
May 23 04:46:56 PM  serve -s build
May 23 04:46:56 PM 
May 23 04:46:56 PM  Find out more about deployment here:
May 23 04:46:56 PM 
May 23 04:46:56 PM https://onrender.com/deployment+

```

Need better ways to work with logs? Try the [Render CLI](#) or set up a [log stream integration](#).

Bottom Right Panel:

Environment

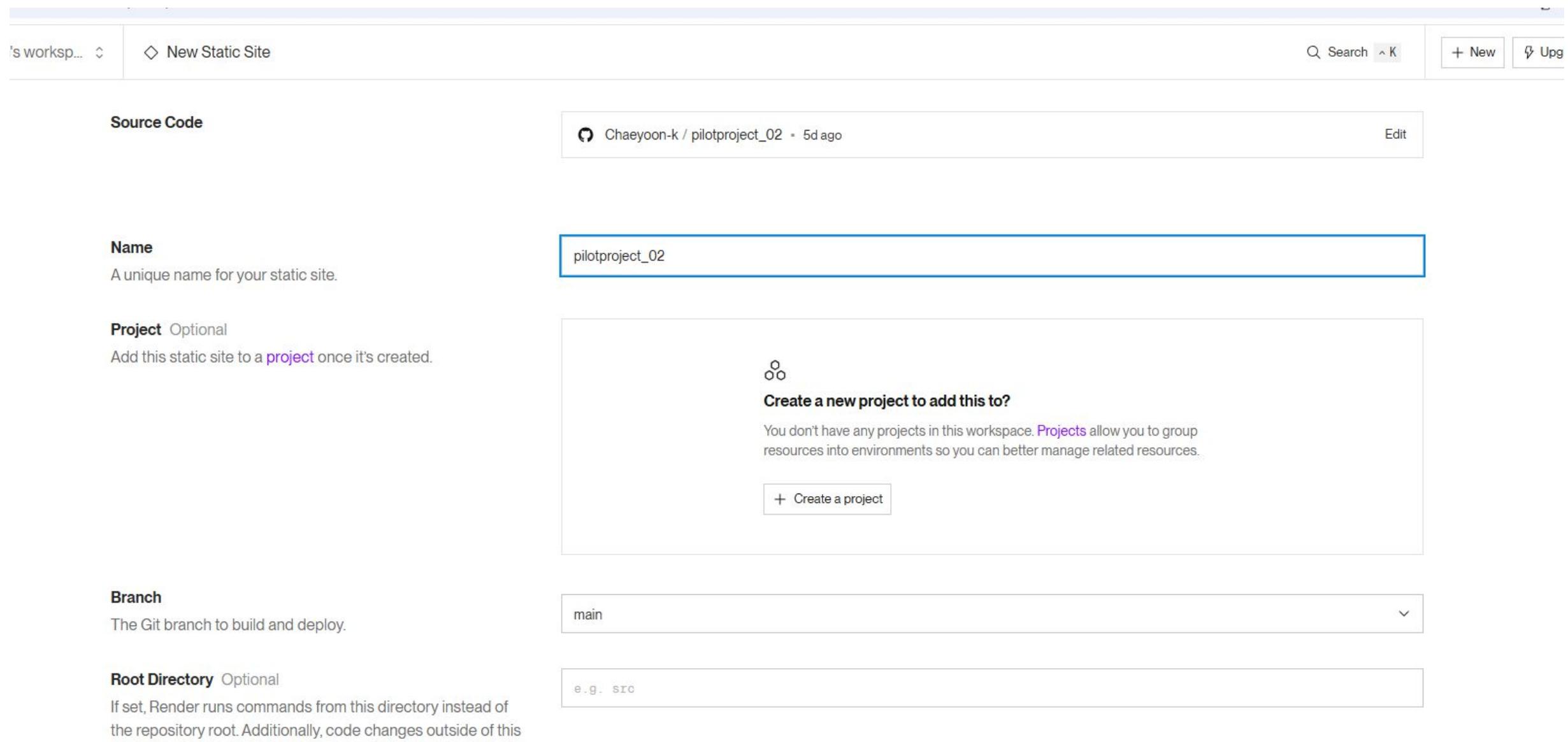
Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

Key	Value
REACT_APP_API_BASE_URL

·프론트 서버 배포

- 프론트 엔드의 경우 static Site를 사용



• 프론트 서버 배포

<한국 문학, 예술>

api.js 파일에서 주소 추가해준다 ,관련한 js 파일 모두 주소로 수정해줌

The screenshot shows a code editor interface with the following details:

- File Menu:** 파일(F), 편집(E), 선택 영역(S), ...
- Search Bar:** frontend
- Toolbar:** Back, Forward, Refresh, Home, Minimize, Maximize, Close.
- Sidebar:** Includes icons for file operations (New, Open, Save, Find, Replace, Copy, Paste, Delete, Undo, Redo) and a tree view of the project structure:
 - Frontend folder
 - public
 - index.html
 - logo192.png
 - logo512.png
 - manifest.json
 - robots.txt
 - src
 - assets
 - components
 - hooks
 - layout
 - pages
 - AnnouncePage.js
 - AnnounceViewPage.js
 - FaqPage.js
 - FaqWritePage.js
 - HomePage.js
 - UserProfilePage.js
 - services
 - announceService.js
 - api.js
 - faqService.js
 - styles
 - App.js
 - App.test.js
 - appbackup.js
 - index.css
 - index.js
 - logo.svg
- Code Editor:** The active tab is "api.js". The code is as follows:

```
src > services > JS api.js > [x] fetchTestData
1 // services/api.js
2 import axios from 'axios';
3
4 const API_BASE_URL = 'https://recall-final-backendtest.onrender.com/api'; // 예시 백엔드 URL
5
6 export const fetchTestData = async () => {
7   console.log("axios 호출 시작");
8   try {
9     // const response = await axios.get(`${API_BASE_URL}/test`);
10    const response = await axios.get(`https://recall-final-backendtest.onrender.com/api/test`);
11    console.log("받은 데이터: ", response.data);
12    return response.data;
13  } catch (error) {
14    console.error('데이터를 가져오는 데 실패했습니다:', error);
15    return null;
16  }
17};
```
- Terminal:** Shows deployment information:
 - 문제, 출력, 디버그 콘솔, 터미널, 포트 tabs.
 - Find out more about deployment here:
<https://cra.link/deployment>
 - C:\develop\Recall_Final\src\main\frontend>
- Bottom Status Bar:** 총 7, 역 20, 공백 2, HTML, CSS, JavaScript, Go Live.

•프론트 서버 배포

<랜더 프론트, 백엔드 웹 연결>

백엔드 RestController에 CrossOrigin에 주소 추가해주지만 WebConfig파일에 주소써주면 CrossOrigin에서는 생략해도 된다.



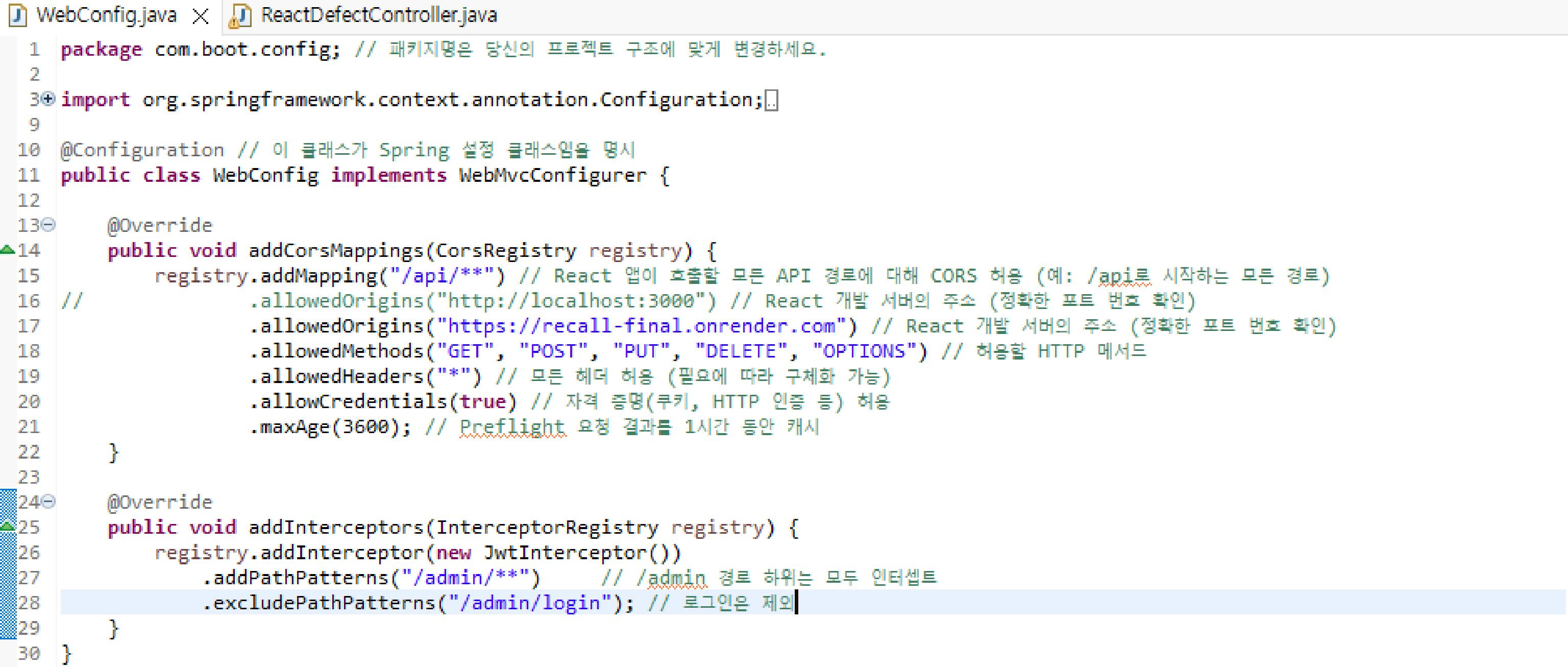
The screenshot shows a code editor window with the tab 'ReactController.java' selected. The code is a Java class named 'ReactController' with annotations for CORS and a single endpoint method 'hello()'.

```
1 package com.boot.controller;
2
3 import org.springframework.web.bind.annotation.CrossOrigin;
4
5
6 @RestController
7 // @CrossOrigin(origins = "http://localhost:3000")
8 @CrossOrigin(origins = "https://recall-final.onrender.com")
9 public class ReactController {
10     @GetMapping("/api/test")
11     public String hello() {
12         return "안녕하세요 백엔드입니다.";
13     }
14 }
15
16 }
```

•프론트 서버 배포

<랜더 프론트, 백엔드 웹 연결>

백엔드 RestController에 CrossOrigin에 주소 추가해주지만 WebConfig파일에 주소써주면 CrossOrigin에서는 생략해도 된다.



```
WebConfig.java × ReactDefectController.java
1 package com.boot.config; // 패키지명은 당신의 프로젝트 구조에 맞게 변경하세요.
2
3+ import org.springframework.context.annotation.Configuration;
9
10 @Configuration // 이 클래스가 Spring 설정 클래스임을 명시
11 public class WebConfig implements WebMvcConfigurer {
12
13@ Override
▲ 14     public void addCorsMappings(CorsRegistry registry) {
15         registry.addMapping("/api/**") // React 앱이 호출할 모든 API 경로에 대해 CORS 허용 (예: /api를 시작하는 모든 경로)
16         // .allowedOrigins("http://localhost:3000") // React 개발 서버의 주소 (정확한 포트 번호 확인)
17         // .allowedOrigins("https://recall-final.onrender.com") // React 개발 서버의 주소 (정확한 포트 번호 확인)
18         // .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS") // 허용할 HTTP 메서드
19         // .allowedHeaders("*") // 모든 헤더 허용 (필요에 따라 구체화 가능)
20         // .allowCredentials(true) // 자격 증명(쿠키, HTTP 인증 등) 허용
21         // .maxAge(3600); // Preflight 요청 결과를 1시간 동안 캐시
22     }
23
24@ Override
▲ 25     public void addInterceptors(InterceptorRegistry registry) {
26         registry.addInterceptor(new JwtInterceptor())
27             .addPathPatterns("/admin/**") // /admin 경로 하위는 모두 인터셉트
28             .excludePathPatterns("/admin/login"); // 로그인은 제외
29     }
30 }
```

•프론트 서버 배포

<랜더 프론트, 백엔드 웹 연결>

React 프론트 → axios 요청(API 요청(HTTP요청)) → Spring 백엔드(요청에 응답)

리액트 페이지 모든 로컬호스트 변경해주면 DB까지 나옴

The screenshot shows a web application titled "My Awesome App" with a dark header bar containing navigation links: 리콜정보, 결합신고, 리콜센터, 리콜통계, and 관리자.

The main content area has a title "리콜내역조회" and a sub-instruction "신고된 자동차 및 건설기계 리콜내역을 조회할 수 있습니다."

Below this, there is a numbered list:

1. 흐
2. 리콜내역조회

Under the second item, there is a section titled "리콜 내역" with a search form. The search form includes a dropdown menu labeled "검색 유형 선택" and a text input field labeled "검색어를 입력하세요". To the right of the input field is a blue "검색" button.

The main table displays a list of recall records with the following columns:

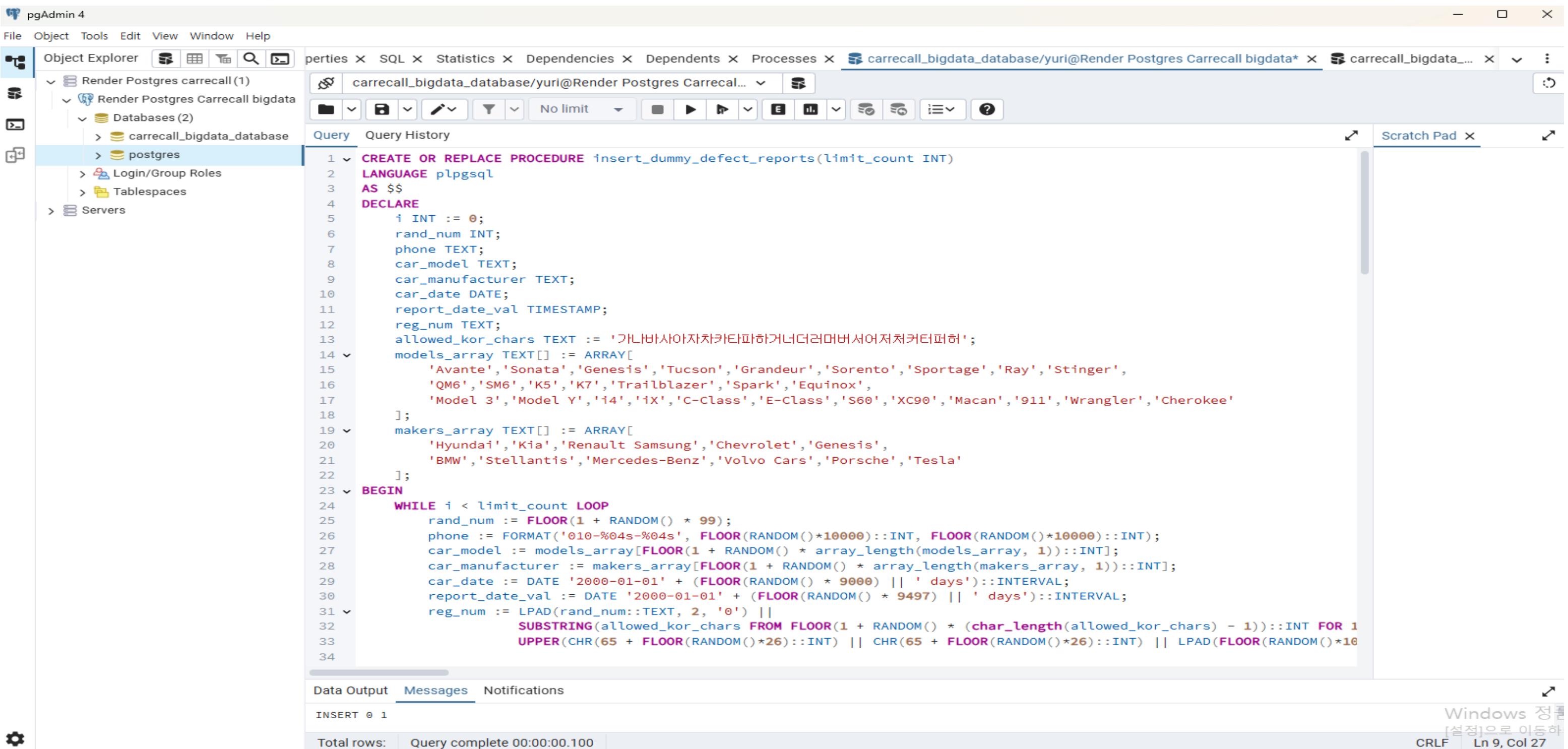
리콜 ID	제품명	제조사	제조기간	기타 정보	모델명	리콜 유형	연락처
848	[재규어랜드로버] 레인지로버 벨라_냉/난방 장치 관련 리콜	(주)재규어랜드로버코리아	2017-05-02 ~ 2017-11-09	레인지로버 벨라		자발적리콜	재규어랜드로버 고객센터 080-337-9696
847	[지엠코리아] 캐딜락 STS - 리어서스펜션 토우링 지엠코리아 크 관련 리콜	지엠코리아 주식회사	2004-09-17 ~ 2004-10-25	캐딜락 STS		자발적리콜	지엠코리아 고객센터 080-3000-5000
846	[토요타] 토요타 시에나 2WD 등 2차종 - 제동장치(진공펌프) 관련 리콜	한국토요타자동차(주)	2017-10-18 ~ 2017-10-19	토요타 시에나 2WD, 토요타 시에나 4WD		자발적리콜	한국토요타자동차 고객지원실 080-525-8255
845	[이탈로모토(유)] BEVERLY 350 SPORT TOURING ABS - 연료탱크 관련 리콜	이탈로모토(유)	2016-06-01 ~ 2018-06-12	BEVERLY 350 SPORT TOURING ABS		자발적리콜	이탈로모토 대표번호 02-502-1946
844	[화창상사] ROADMASTER ELITE 등 6차종 - 제어 스위치 관련 리콜	화창상사(주)	2017-04-20 ~ 2017-11-14	ROADMASTER ELITE, ROADMASTER, CHIEF VINTAGE, SPRINGFIELD, CHIEF CLASSIC, SPRINGFIELD DARK HORSE		자발적리콜	화창상사 고객센터 02-2279-0170
843	[혼다코리아] ODYSSEY - 2열 좌측시트 등받이 고정장치 관련 리콜	혼다코리아(주)	2013-11-07 ~ 2015-06-05	ODYSSEY		자발적리콜	혼다코리아 고객센터 080-360-0505

At the bottom of the page, there is a footer with the text "© 2025 My Awesome App".

•백엔드 서버 배포

- DB mysql ->pgAdmin 4(postgresql)
- 마이그레이션
- 프롬프트 변경후 입력

빅데이터검색 플랫폼 프로젝트
Logistic Regression



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the Object Explorer with a connection to 'carrecall_bigdata_database'. The main area contains a SQL query window with the following PL/pgSQL code:

```
1 CREATE OR REPLACE PROCEDURE insert_dummy_defect_reports(limit_count INT)
2 LANGUAGE plpgsql
3 AS $$
4 DECLARE
5     i INT := 0;
6     rand_num INT;
7     phone TEXT;
8     car_model TEXT;
9     car_manufacturer TEXT;
10    car_date DATE;
11    report_date_val TIMESTAMP;
12    reg_num TEXT;
13    allowed_kor_chars TEXT := '가나뱌쌰아자차카탸퍄하거너더러마벼서어저쳐커터펴허';
14    models_array TEXT[] := ARRAY[
15        'Avante', 'Sonata', 'Genesis', 'Tucson', 'Grandeur', 'Sorento', 'Sportage', 'Ray', 'Stinger',
16        'QM6', 'SM6', 'K5', 'K7', 'Trailblazer', 'Spark', 'Equinox',
17        'Model 3', 'Model Y', 'i4', 'iX', 'C-Class', 'E-Class', 'S60', 'XC90', 'Macan', '911', 'Wrangler', 'Cherokee'
18    ];
19    makers_array TEXT[] := ARRAY[
20        'Hyundai', 'Kia', 'Renault Samsung', 'Chevrolet', 'Genesis',
21        'BMW', 'Stellantis', 'Mercedes-Benz', 'Volvo Cars', 'Porsche', 'Tesla'
22    ];
23 BEGIN
24     WHILE i < limit_count LOOP
25         rand_num := FLOOR(1 + RANDOM() * 99);
26         phone := FORMAT('010-%04s-%04s', FLOOR(RANDOM()*10000)::INT, FLOOR(RANDOM()*10000)::INT);
27         car_model := models_array[FLOOR(1 + RANDOM() * array_length(models_array, 1))::INT];
28         car_manufacturer := makers_array[FLOOR(1 + RANDOM() * array_length(makers_array, 1))::INT];
29         car_date := DATE '2000-01-01' + (FLOOR(RANDOM() * 9000) || ' days')::INTERVAL;
30         report_date_val := DATE '2000-01-01' + (FLOOR(RANDOM() * 9497) || ' days')::INTERVAL;
31         reg_num := LPAD(rand_num::TEXT, 2, '0') ||
32                     SUBSTRING(allowed_kor_chars FROM FLOOR(1 + RANDOM() * (char_length(allowed_kor_chars) - 1))::INT FOR 1
33                     UPPER(CHR(65 + FLOOR(RANDOM()*26)::INT) || CHR(65 + FLOOR(RANDOM()*26)::INT) || LPAD(FLOOR(RANDOM()*10
34
```

The bottom status bar indicates 'Query complete 00:00:00.100'.

•백엔드 서버 배포

- Dockerfile
- 설치 명령

```

FROM openjdk:17-jdk-slim

WORKDIR /app

# Node.js 설치 및 wkhtmltopdf 설치
RUN apt-get update && apt-get install -y curl wkhtmltopdf \
    && curl -fsSL https://deb.nodesource.com/setup_18.x | bash - \
    && apt-get install -y nodejs \
    && apt-get clean

COPY ..

RUN ls -l gradlew

RUN chmod +x gradlew

RUN ls -l gradlew

RUN ./gradlew bootJar --no-daemon --stacktrace --info

RUN cp build/libs/*jar app.jar

CMD ["java", "-jar", "app.jar"]

```

이름	수정한 날짜	유형
.git	2025-05-23 (금) 오후...	파일 콜
.gradle	2025-05-21 (수) 오후...	파일 콜
.settings	2025-05-23 (금) 오후...	파일 콜
bin	2025-05-21 (수) 오후...	파일 콜
build	2025-05-21 (수) 오후...	파일 콜
gradle	2025-05-21 (수) 오후...	파일 콜
images	2025-05-21 (수) 오후...	파일 콜
src	2025-05-21 (수) 오후...	파일 콜
.classpath	2025-05-21 (수) 오후...	CLASS
.gitattributes	2025-05-21 (수) 오후...	Git Atti
.gitignore	2025-05-21 (수) 오후...	Git Ign
.project	2025-05-21 (수) 오후...	PROJE
build.gradle	2025-05-21 (수) 오후...	Gradle
Dockerfile	2025-05-21 (수) 오후...	파일
generated_from_url.pdf	2025-05-23 (금) 오후...	Micro
gradlew	2025-05-21 (수) 오후...	파일
gradlew.bat	2025-05-21 (수) 오후...	Windows
README.md	2025-05-21 (수) 오후...	Markdo
settings.gradle	2025-05-21 (수) 오후...	Gradle

•백엔드 서버 배포

•환경변수지정

```

1 spring.application.name=boot_bigdata_project
2 server.port=${PORT:8485}
3
4 #Server
5 server.servlet.session.timeout=30m
6
7 #Spring MVC
8 spring.mvc.view.prefix=/WEB-INF/views/
9 spring.mvc.view.suffix=.jsp
10
11 #Database config
12 #spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
13 #spring.datasource.url=jdbc:mysql://localhost:3306/atom
14 #spring.datasource.username=bts
15 #spring.datasource.password=1234
16
17 #Render
18 spring.datasource.driver-class-name=org.postgresql.Driver
19 spring.datasource.url=${SPRING_DATASOURCE_URL}
20 spring.datasource.username=${SPRING_DATASOURCE_USERNAME}
21 spring.datasource.password=${SPRING_DATASOURCE_PASSWORD}
22
23 #mybatis config
24 mybatis.config-location=classpath:mybatis-config.xml
25
26 #google.gemini
27 google.gemini.api.key=AIzaSyB6bqfe7rN1E_uU9Qes_DDTqY2f9cnF49g
28
29 #generate pdf
30 wkhtmltopdf.executable=C:/Program Files/wkhtmltopdf/bin/wkhtmltopdf.exe
31
32 #flask
33 flask.api.base-url=http://localhost:5000
34
35 # For branching of common recall statistics grammar
36 #mybatis config dbType=postgresql

```

The screenshot shows the environment configuration for the 'Recall_Final_backend' service. On the left, a sidebar lists various service components: Dashboard, Recall_Final_backend (selected), Events, Settings, MONITOR (Logs, Metrics), MANAGE (Environment, Shell, Scaling, Previews, Disks, Jobs). The main area displays the 'Environment' section with the heading 'Environment Variables'. It contains a table with four rows:

Key	Value
SPRING_DATASOURCE_DRIVER_CLASS...	org.postgresql.Driver
SPRING_DATASOURCE_PASSWORD	FqEPRBW7IBk6v7SsEoX06E7WwUpiL9C1
SPRING_DATASOURCE_URL	jdbc:postgresql://dpg-d0m40jgd13pa73c0bahg-a.singapore...
SPRING_DATASOURCE_USERNAME	yuri

•백엔드 서버 배포

•서버 돌아감

•서버 시작 및 배포완료

빅데이터검색 플랫폼 프로젝트

Logistic Regression

Upgrade now

May 23, 2025 at 5:40 PM In Progress

632fe60 Merge branch 'main' of https://github.com/Wjyuy/Recall_Final

Cancel deploy

All logs Search Live tail GMT+9 ...

```
May 23 05:44:21 PM #15 exporting to docker image format
May 23 05:44:27 PM #15 exporting layers
May 23 05:44:55 PM #15 exporting layers 27.8s done
May 23 05:44:55 PM #15 exporting manifest sha256:cb0c430013efa2e261d7e89b77a7df323
04a1939938e6c830350f94bb129bbf0 done
May 23 05:44:55 PM #15 exporting config sha256:68c41a9d2d862c8f4be61bf00d6967f32e6
69cb70c5c427e4973619cf37b45ef done
May 23 05:45:06 PM #15 DONE 38.5s
May 23 05:45:06 PM
May 23 05:45:06 PM #16 exporting cache to client directory
May 23 05:45:06 PM #16 preparing build cache for export
May 23 05:45:20 PM #16 Pushing image to registry...
May 23 05:45:19 PM #16 writing cache manifest sha256:65eceb72c590f8e4a365a2c346df3
4eac0b0ab7f79db68c6c32518358e9e7403 0.0s done
May 23 05:45:19 PM #16 DONE 13.4s
May 23 05:45:40 PM ==> Deploying...
May 23 05:45:37 PM Upload succeeded
```

ⓘ Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more.

Upgrade now

May 23, 2025 at 5:40 PM In Progress

632fe60 Merge branch 'main' of https://github.com/Wjyuy/Recall_Final

Cancel deploy

All logs Search Live tail GMT+9 ...

```
May 23 05:44:21 PM #15 exporting layers
May 23 05:44:55 PM #15 exporting layers 27.8s done
May 23 05:44:55 PM #15 exporting manifest sha256:cb0c430013efa2e261d7e89b77a7df323
04a1939938e6c830350f94bb129bbf0 done
May 23 05:44:55 PM #15 exporting config sha256:68c41a9d2d862c8f4be61bf00d6967f32e6
69cb70c5c427e4973619cf37b45ef done
May 23 05:45:06 PM #15 DONE 38.5s
May 23 05:45:06 PM
May 23 05:45:06 PM #16 exporting cache to client directory
May 23 05:45:06 PM #16 preparing build cache for export
May 23 05:45:19 PM #16 writing cache manifest sha256:65eceb72c590f8e4a365a2c346df3
4eac0b0ab7f79db68c6c32518358e9e7403 0.0s done
May 23 05:45:19 PM #16 DONE 13.4s
May 23 05:45:20 PM #16 Pushing image to registry...
May 23 05:45:37 PM #16 Upload succeeded
May 23 05:45:40 PM ==> Deploying...
May 23 05:46:16 PM #16 서버 시작!
```

·백엔드 서버 배포

- Env 파일 사용 / 개발 배포 서버 따로이용

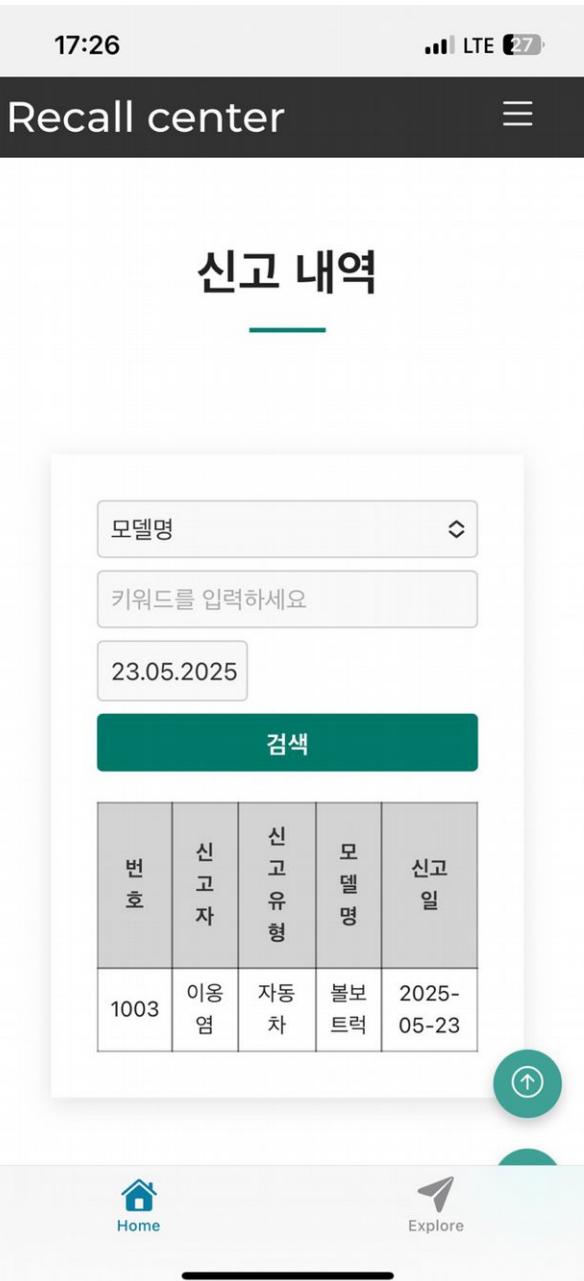
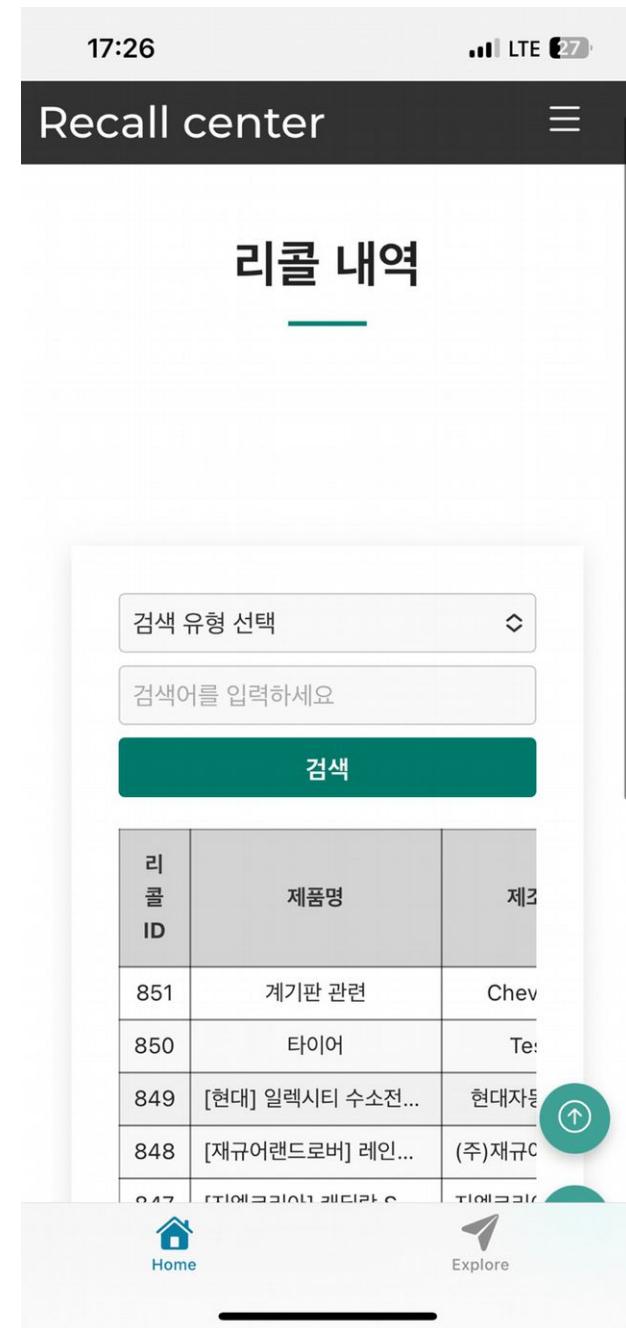
```
bash
복사편집

project-root/
  └── backend/          # Spring Boot (Gradle 기반)
    ├── .env             # 로컬 전용 환경 변수
    └── src/main/resources/application.yml

  └── frontend/          # React
    ├── .env             # 로컬 전용
    └── .env.production  # Render 전용

└── .gitignore
```

·어플 화면 동기화



·어플 화면 동기화

<webview 사용할때 index.tsx>

생성된 폴더에서 App.js 파일을 열어서 수정해준다.(C:\Users\user\recall-app)

vs code에서 수정 - tabs 구조여서 app/tabs/index.tsx 파일을 수정해줬다.

```
import { SafeAreaView } from 'react-native';
import { WebView } from 'react-native-webview';

export default function IndexPage() {
  return (
    <SafeAreaView style={{ flex: 1 }}>
      <WebView
        source={{ uri: 'https://recall-final.onrender.com' }}
        style={{ flex: 1 }}
      />
    </SafeAreaView>
  );
}
```

코드로 들어가면

Tunnel ready.



```
> Metro waiting on exp://plwqvsw-anonymous-8081.expo.direct
> Scan the QR code above with Expo Go (Android) or the Camera app (iOS)

> Web is waiting on http://localhost:8081

> Using Expo Go
> Press s | switch to development build

> Press a | open Android
> Press w | open web
```

·어플 화면 동기화

<webview 사용할때 index.tsx>

그후에 npx expo start 하면 qr 생성된다.

코드로 들어가면



웹화면이 보인다.

자체평가

빅데이터검색 플랫폼 프로젝트

Logistic Regression

01 우주연

이번 프로젝트에서 JSP 프로젝트를 React로 마이그레이션/ 프론트앤드 디자인하고, Google Gemini API를 연동하여 리콜 통계 요약 및 챗봇 기능을 구현하였습니다!

React 컴포넌트 분리 재사용을 사용해보면서, 모듈화의 장점을 크게 느꼈고 마이그레이션하면서 새로 만드는게 빠르겠는데.?라는 생각이 계속 들었지만, 그래도 코드 구조화, 사용자 경험 개선 등의 작업을 해 보면서 많이 배운 것 같습니다! 깔끔하게 서버 배포해주고 기능 구현 똑딱똑딱 해주는 팀원들 덕분에 뿌듯한 프로젝트 결과물이 나왔다고 생각해 만족스럽습니다!

03 성유리

API를 사용하여 자료를 다운받을수 있게끔 구현하고, 서버 배포를 위해 aws를 시도하다, 시간상 render를 사용했습니다. 백엔드와 프론트를 나누어 작업하니 유지보수에 원활하여 좋았던것 같습니다. 뿐만 아니라, 개발과 배포를 나누어 소스트리로 작업하더라도 각자 진행할수 있어 좋았던것 같습니다. 신경써야할 설정이 많았지만, 각자 맡은바를 잘 설명해주어 진행이 잘되었던 생각합니다.

02 권준우

데이터 분석 위주의 프로젝트는 처음인데 좋은 결과가 나온 것 같아 만족합니다. 특히 파이썬 flask 서버에서 구현한 벡터 유사도 측정 기능을 Spring과 연동해 표현하는 작업이 흥미로웠습니다.

처음 예정보다 일정이 상당히 빽빽했는데, 기존에 진행했던 프론트를 React로 마이그레이션 하고, 동시에 처음 하는 배포 작업을 진행하느라 여태 경험하지 못한 많은 문제들을 해결하면서 상황에 대처하는 능력을 기를 수 있었습니다.

04 김채윤

API 사용 외에도 배포, 앱화면 출력 등 다양한 시도를 많이 해볼 수 있었던 시간이었다. 팀원들이 각자 맡은 것들이 프로젝트에 잘 적용되어서 만족스러운 프로젝트였고 배포하면서 이러한 문제들이 발생할 수 있다는 것을 알게 되었다