Profiler 구현 보고서

과목명 : 웹 응용기술 (002)

교수명 : 강영명 교수님

팀원 : 20230815 권오현

20230884 홍민경

20230003 김민규

20230878 천병권

목차

1. 프로그램 개요

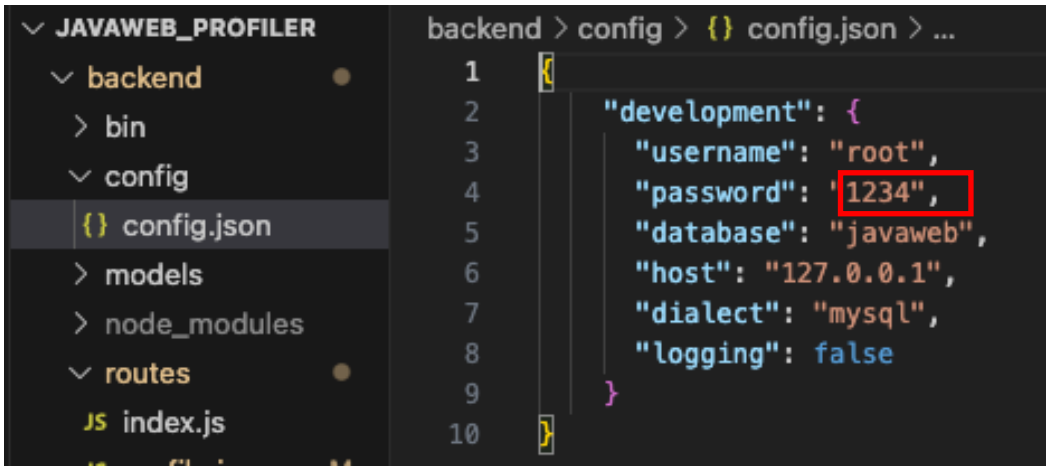
현대의 컴퓨터 작업 처리 속도는 CPU 의 성능에 크게 의존하며, 특히 Core 단위의 처리 능력은 전체 시스템의 효율성과 직결된다. 그러나 물리적으로 제한된 수의 코어를 최적으로 활용하기 위해서는, 각 코어가 얼마만큼의 Task 를 수행하고 있는지를 정량적으로 분석하고 시각화할 수 있는 도구가 필요하다.

본 프로젝트는 이러한 목적을 달성하기 위해 CPU Core 별, Task 수행량을 분석하고 시각화하는 성능 프로파일링 웹 애플리케이션을 구현하였다. 사용자가 업로드한 .xlsx 형식의 성능 데이터를 기반으로, 각 Task 가 어떤 Core 에서 얼마나 실행되었는지 수집하고, 이를 바탕으로 통계 데이터를 계산한다. 이후 최소(MIN), 최대(MAX), 평균(AVG) 사용량과 같은 지표를 도출하여 시각적으로 직관적인 차트로 표현함으로써 사용자는 Core 단위의 처리 편중 현상, 병목 구간 등을 쉽게 파악할 수 있다.

이 프로젝트는 2025 년 1 학기 '웹응용기술' 수업의 팀 프로젝트 과제로 진행되었으며, 백엔드는 Node.js 기반의 Express 프레임워크로 구성되었다. 프론트엔드는 React 를 기반으로 개발하였고, Bootstrap 및 Chart.js 를 활용하여 반응형 UI 와 동적 차트 시각화를 구현하였다. 데이터는 업로드 후 파싱 과정을 거쳐 MySQL 에 저장되며, 이후 다양한 기준(Task, Core)에 따라 필터링 및 분석이 가능하도록 설계하였다.

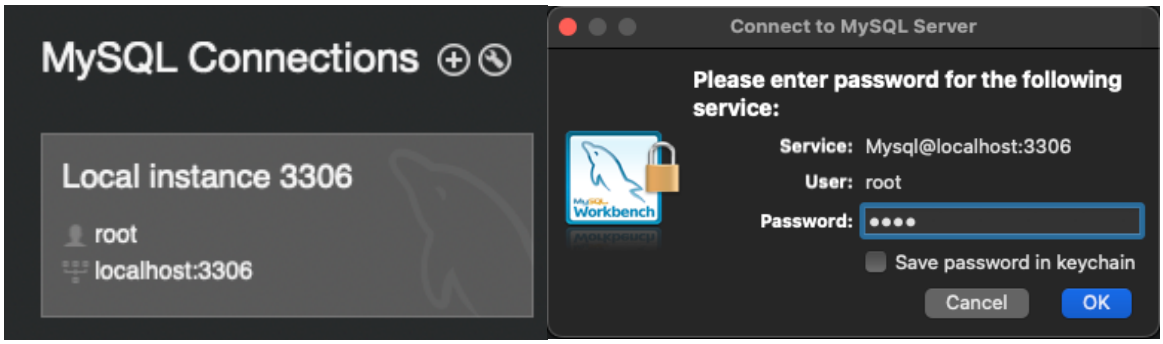
2. 프로그램 수행 절차

2.1 실행 방법



[그림 1] MYSQL DB 비밀번호 수정

프로젝트의 javaweb_profiler/backend/config/config.json 파일로 이동하여 입력되어 있는 password 를 본인 컴퓨터에 설치되어 있는 mysql root 계정의 비밀번호로 설정한다.



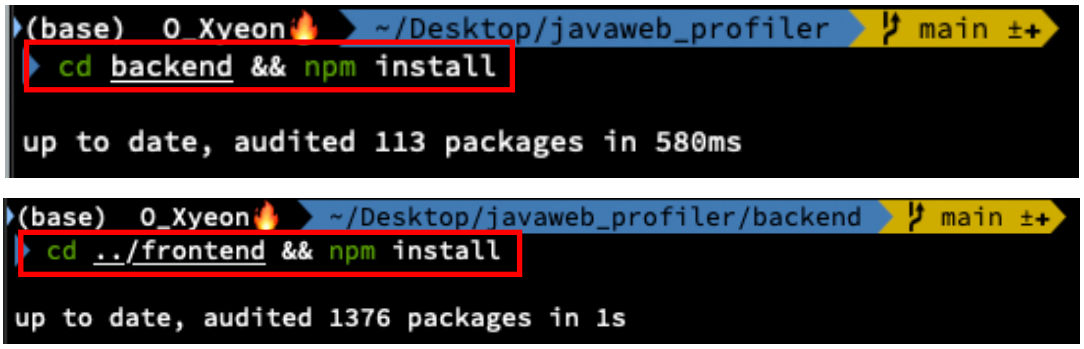
[그림 2] mysql workbench 접속

mysql workbench 를 열어 root 계정으로 비밀번호를 입력하고 들어간다.

[그림 3] DB 생성 및 지정

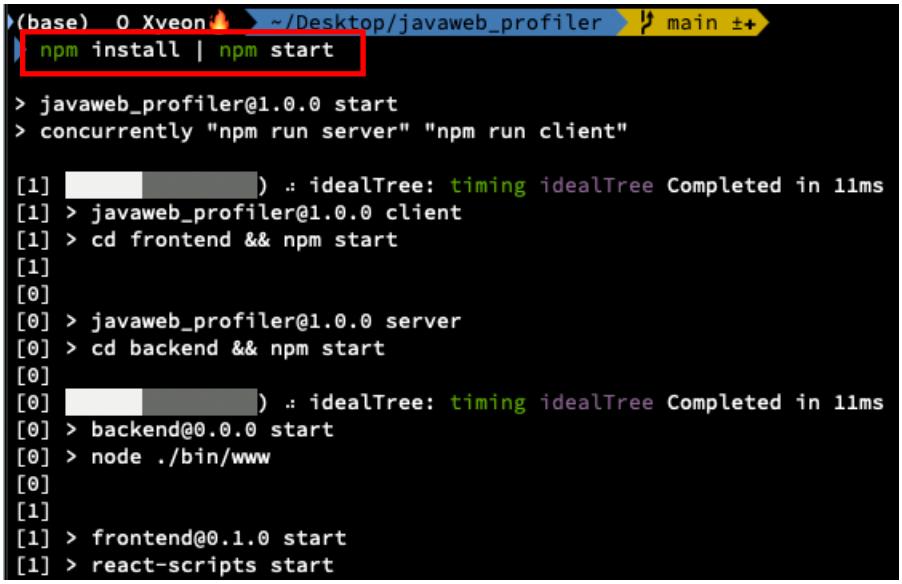
쿼리문을 입력하는 창이 뜨면 CREATE DATABASE javaweb; 과 USE javaweb; 을 실행시켜 데이터 베이스를 생성하고 작업을 진행할 데이터베이스를 지정해준다.

2.2 프로그램 시작



[그림 4] 필요 패키지 설치

backend 와 frontend 디렉토리의 node_modules 폴더에 있는 패키지를 다운받기 위해 터미널을 열고 /javaweb_profiler 디렉토리에서 cd backend && npm install 명령을 통해 backend 에 필요한 패키지를 설치해주고, cd ../frontend && npm install 명령을 통해 frontend 에 필요한 패키지를 설치해준다.



[그림 5] npm 설치 및 프로젝트 실행

javaweb_profiler 디렉토리에 필요한 패키지를 다운받고 실행하기 위해 npm install | npm start 명령을 통해 프로젝트를 실행시킨다.

[그림 6] URL 접속

터미널에 출력되는 주소에 마우스 커서를 올리고 Crtl + Click (MAC : Command + Click)을 하여 페이지에 접속한다.

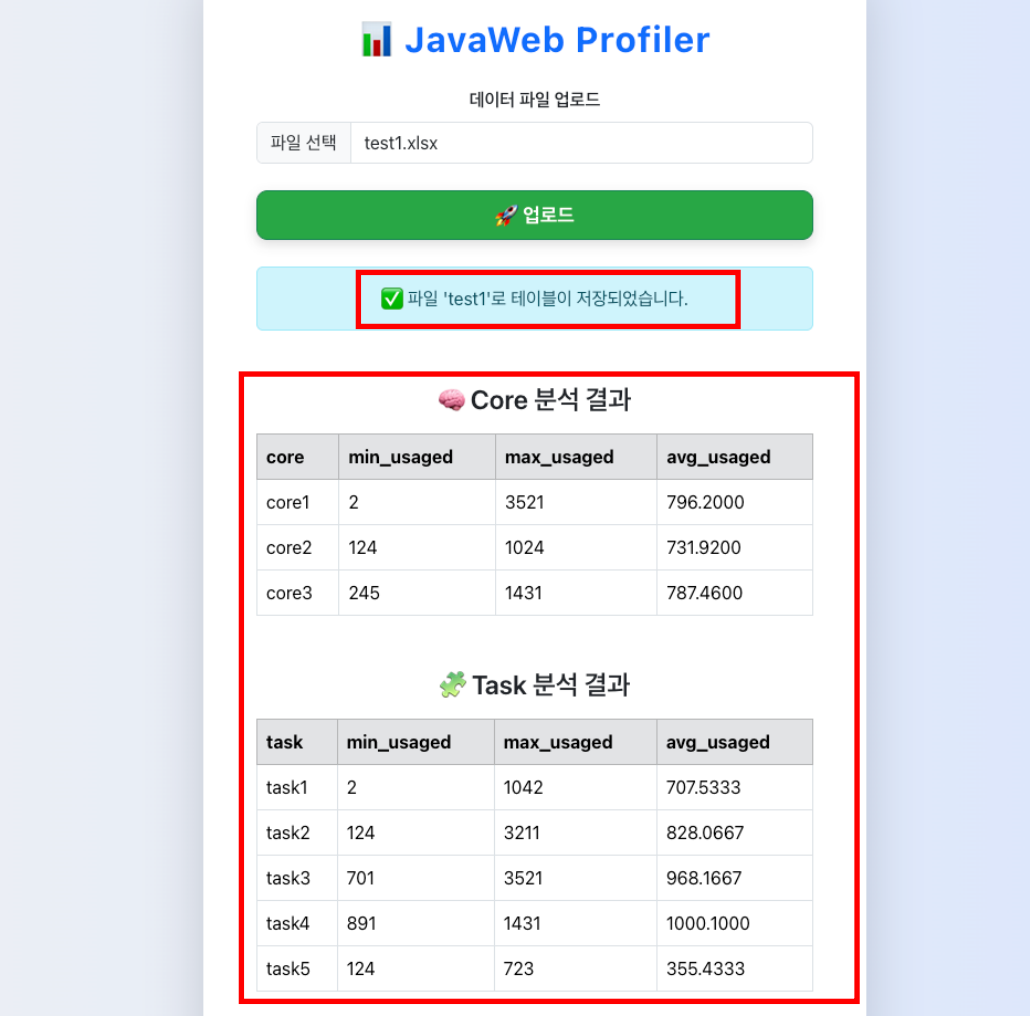
2.3 데이터 파일 업로드

[그림 7] 메인 페이지

메인 페이지가 뜨면 파일 업로드를 눌러 파일 업로드 페이지로 이동한다.

[그림 8] 파일 업로드 페이지 업로드 절차

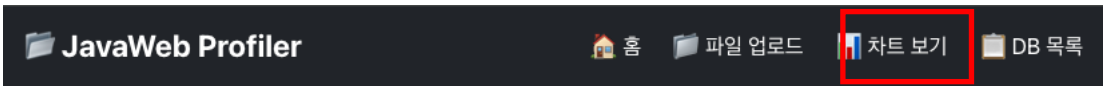
업로드 화면이 뜨면 파일 선택 버튼을 누르고 profiling 을 원하는 데이터 파일(.xlsx)을 선택 후 업로드 한다.



[그림 9] 업로드 결과 및 분석 결과

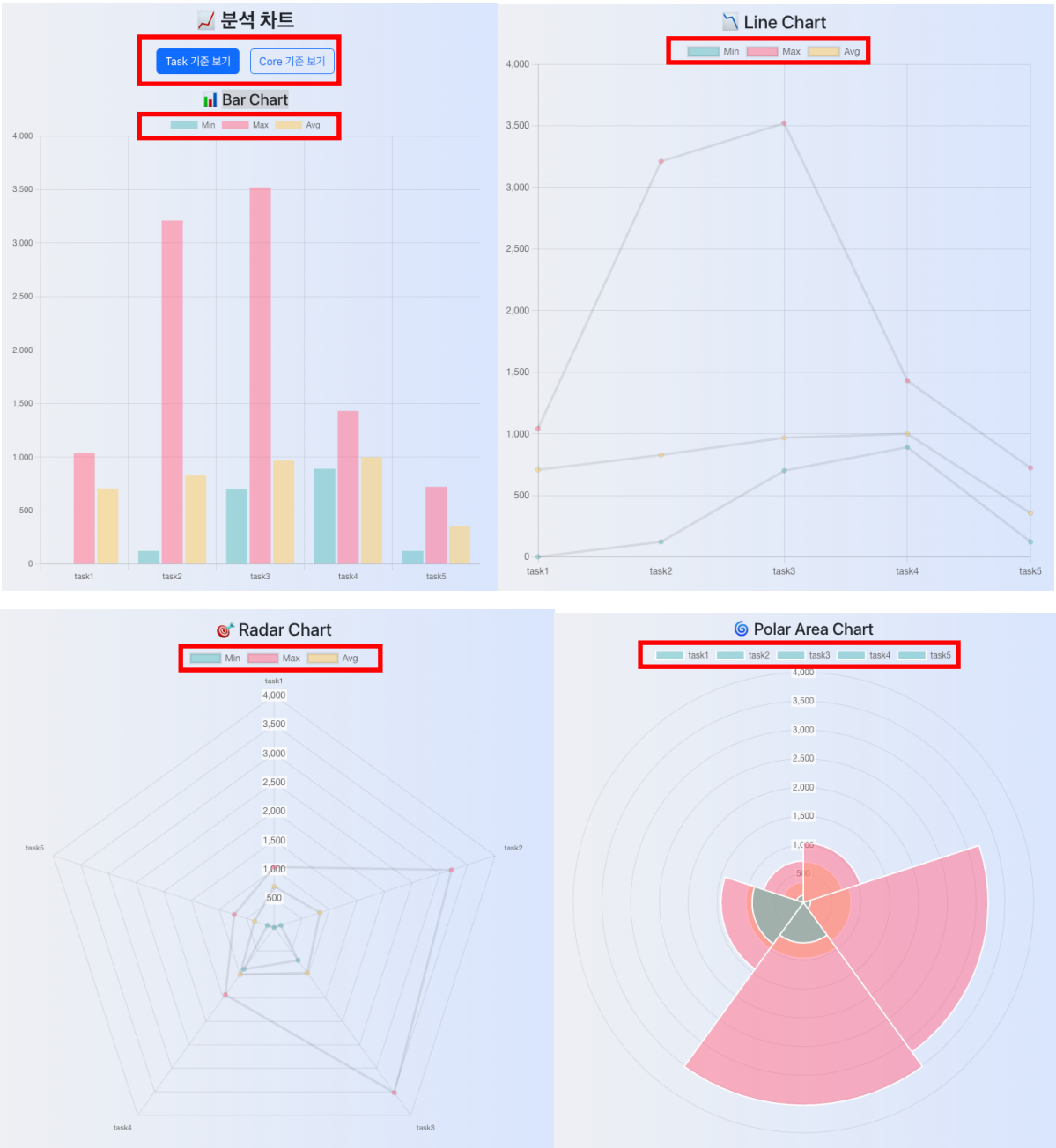
파일을 업로드하면 데이터 베이스에 저장되었는지 결과가 나오고, 그 아래에 Core 의 기준과 Task 의 기준으로 나뉘어 2 차원 테이블로 표현한 데이터 분석 결과가 나온다.

2.4 결과 출력



[그림 10] 페이지 헤더(차트보기)

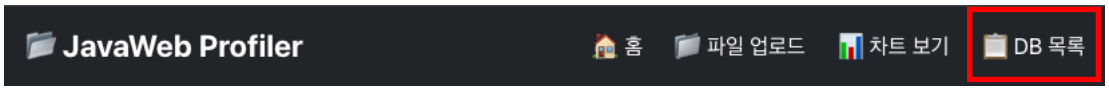
웹 페이지 헤더에서 차트 보기를 누르면 Bar Chart, Line Chart, Radar Chart, Polar Area Chart 로 표현된 분석 결과를 볼 수 있다.



[그림 11] 파일 분석 차트

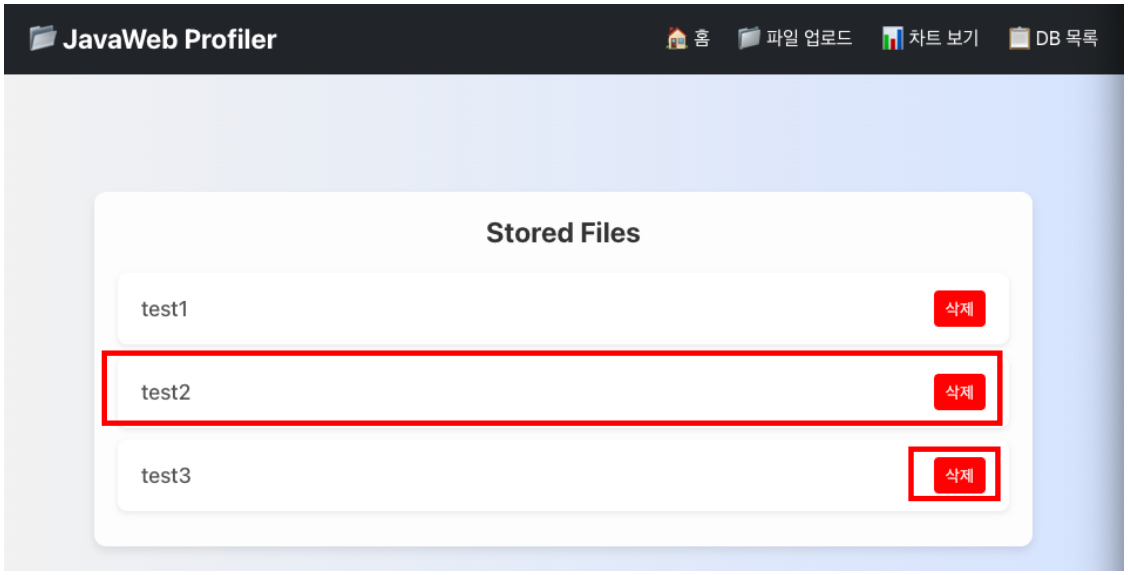
각각의 차트들이 나오고 Task 기준 보기 버튼과 Core 기준 보기 버튼을 통해 각각의 분석 결과를 볼 수 있고 Min, Max, Avg 옆에 있는 색을 클릭하면 그 값을 제외한 나머지 값들의 분석 결과만 볼 수 있다. Polar Area Chart 는 특정 Task 또는 Core 를 제외하고 결과를 볼 수 있다.

2.5 데이터 삭제



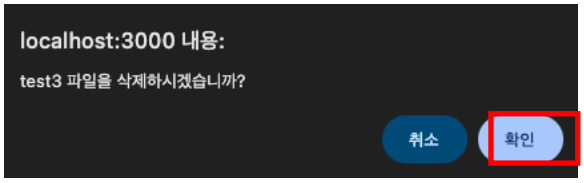
[그림 12] 페이지 헤더(DB 목록)

웹 페이지의 헤더에서 DB 목록을 들어가면 현재 데이터 베이스에 저장되어 있는 파일 목록을 확인할 수 있다.



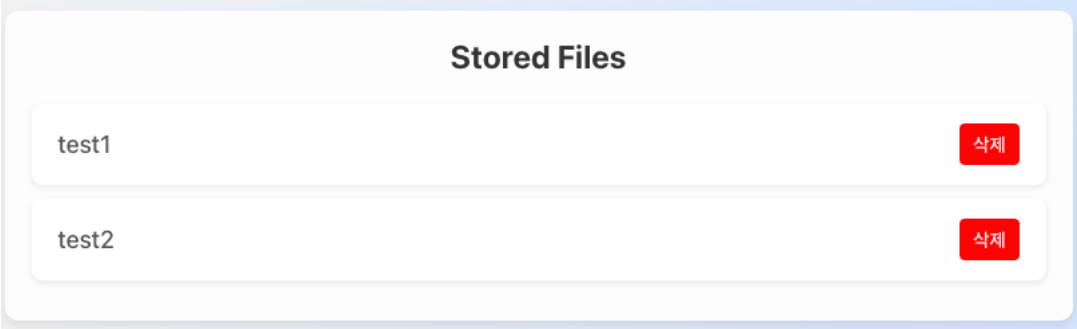
[그림 13] DB 목록 조회 및 데이터 삭제

각 목록을 클릭하면 차트 보기 페이지로 넘어가 각 데이터의 분석 결과 차트를 확인할 수 있다.



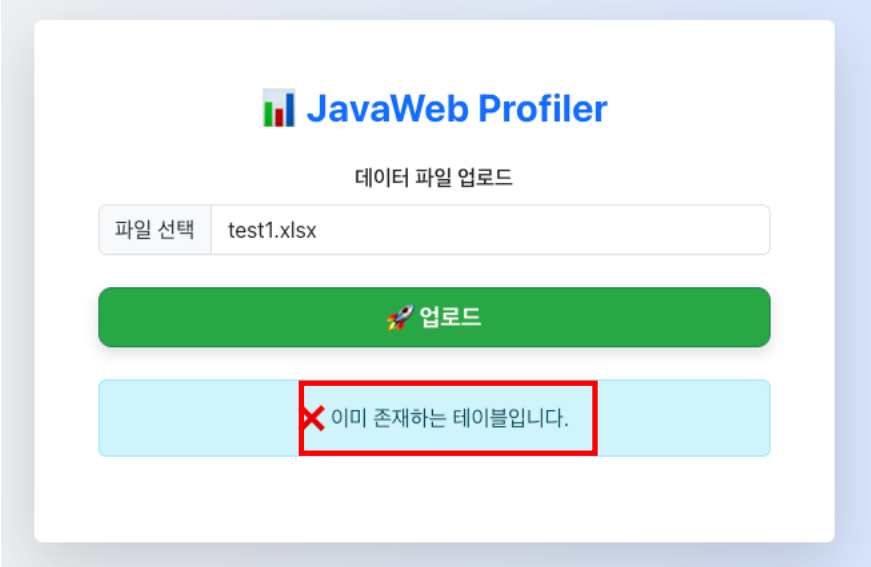
[그림 14] 삭제 시 alert 메세지

삭제 버튼을 누르면 alert 메세지가 뜨고 확인을 누르면 데이터 베이스에서 파일 이름으로 된 테이블이 삭제되는 것을 확인할 수 있다.



[그림 15] 데이터 삭제 결과

2.6 오류 처리



[그림 16] 데이터 존재 유무 확인

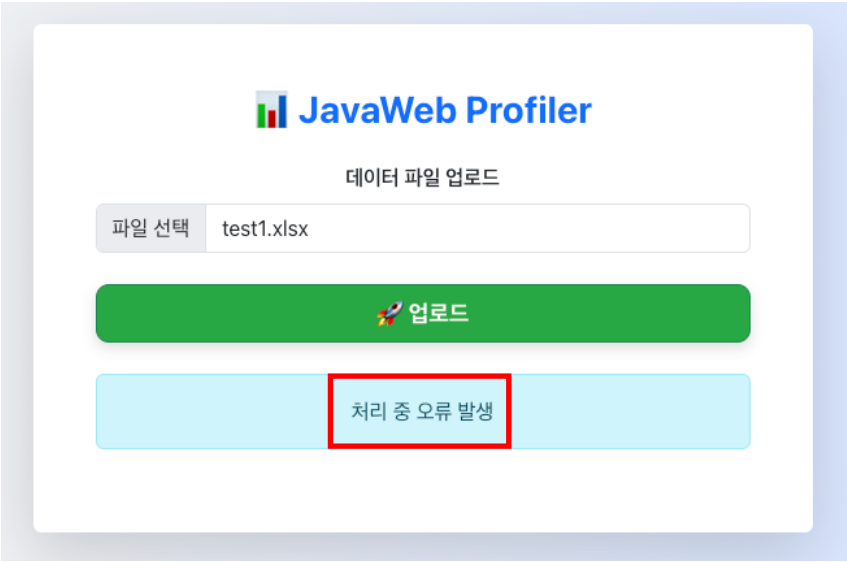
업로드 할 때 데이터 베이스에서 테이블이 존재하는지 확인후 이미 업로드된 파일이라면 다시 저장되지 않도록 만들었다. 오류 메세지를 화면에 출력하여 다른 엑셀 파일을 업로드 할 수 있도록 하였다.

[그림 17] 데이터 파일 확장자 확인

```
error: 엑셀 파일(.xlsx, .xls)만 업로드 가능합니다.
```

파일의 확장자를 검사하여 엑셀 파일(.xlsx, .xls) 이 외의 파일은 업로드 되지 않도록 만들었고 화면의 오류 메세지를 출력하여 엑셀 파일만 업로드 할 수 있도록 처리했다.

또한, 서버의 콘솔에도 오류 메세지를 출력하여 어떤 오류가 발생했는지 알 수 있도록 만들었다.



[그림 18] 서버 오류 메세지

```
POST /api/profile/uploadExcel 500
```

이 외의 서버 오류 발생 시 처리 중 오류 발생이라는 오류 메세지를 화면에 출력하고 데이터 파일 저장을 막는다. 서버의 콘솔에서 오류 메시지를 통해 어떤 API 에서 오류가 발생했는지 확인할 수 있다.

3. 서버 수행 절차 및 로직

3.1 Node 서버 구성

이번 프로젝트의 백엔드는 Node.js 환경에서 Express 프레임워크를 기반으로 구현되었으며, 루트 파일인 app.js 를 통해 서버를 구동합니다.

주요 사용 라이브러리는

1) xpress: 웹 서버 프레임워크
2) sequelize: MySQL 과 연동하기 위한 ORM
3) path: 파일 경로 관련 유틸리티
4) fs: 파일 시스템 접근용 모듈
5) morgan: 개발 중 접속 로그 확인용 미들웨어

폴더명	역할 설명
config/	config.json 파일을 통해 DB(MySQL) 접속 설정 정보를 포함
models/	Sequelize 기반 DB 모델 및 테이블 생성, 삭제 등의 기능을 정의
routes/	클라이언트 요청을 처리하는 라우팅 로직이 들어있는 폴더
node_modules/	프로젝트에 설치된 외부 라이브러리들이 저장

프론트엔드는 React 기반으로 별도 디렉토리(frontend/)에서 구성됩니다.

3.2 데이터 파일 입력

사용자는 웹 화면에서 .xlsx 확장자를 가진 엑셀 파일을 업로드함으로써 데이터를 입력할 수 있다. 업로드 폼은 <input type="file">을 통해 구현되어 있으며, 사용자가 파일을 선택하지 않고 제출을 시도할 경우 "📂 파일을 선택해주세요!"라는 경고 메시지가 브라우저 alert 창을 통해 출력된다.

사용자가 파일을 선택하고 제출 버튼을 클릭하면, JavaScript 에서 FormData 객체를 사용하여 선택한 파일을 file 이라는 키로 담아 POST 방식으로 서버의 /api/profile/uploadExcel 엔드포인트에 전송한다. 이때 파일은 프론트엔드에서 형식 검사 없이 그대로 서버로 전송되며, 백엔드에서 multer 의 fileFilter 옵션을 통해 .xlsx 또는 .xls 파일만 허용되도록 필터링한다. 조건에 맞지 않는 경우 "❌ 엑셀

파일(.xlsx, .xls)만 업로드 가능합니다." 라는 메시지가 JSON 형식으로 응답되고, 프론트엔드에서 이를 받아 사용자에게 알려준다.

서버에서 업로드된 파일은 xlsx 모듈을 통해 첫 번째 시트의 데이터를 sheet_to_json 방식으로 2 차원 배열로 파싱된다. 이 배열에서 모든 셀이 빈 값인 줄은 제거하여 유효한 데이터만 필터링한다.

필터링된 데이터는 models/index.js 파일의 createDynamicTable 함수에 전달되어, 각 행을 core, task, usaged 의 형태로 변환하여 새로운 DB 테이블에 삽입된다. 테이블 이름은 업로드된 파일의 원래 이름을 기반으로 자동 설정되며, 이미 동일한 이름의 테이블이 존재하는 경우 중복 방지를 위해 업로드가 거부된다. 이때도 "❌ 이미 존재하는 테이블입니다." 라는 메시지가 사용자에게 반환된다.

업로드가 성공적으로 완료되면, 백엔드는 해당 테이블을 기반으로 분석 API(/analyze/:tableName)를 호출할 수 있도록 클라이언트에 성공 메시지와 함께 테이블 이름을 전달한다. 프론트엔드는 이 정보를 받아 localStorage 에 저장한 후, 자동으로 분석 API 를 호출하여 core/task 별 사용량의 최소값, 최대값, 평균값을 계산한 결과를 받아 시각적으로 표 형태로 출력한다.

이를 통해 사용자는 다수의 코어가 수행한 프로파일 데이터를 손쉽게 업로드하고, 분석 결과를 직관적으로 확인할 수 있다.

3.3 차트 출력 과정

업로드된 데이터가 서버에 저장되고 분석이 완료되면, 사용자는 시각화를 통해 Core 및 Task 기준의 통계 데이터를 다양한 그래프 형태로 확인할 수 있다. 본 프로젝트에서는 Chart.js 를 활용하여 다음과 같은 방식으로 시각화 기능을 구현하였다.

1) 테이블 이름 확인 및 분석 데이터 요청

Chart.jsx 에서는 업로드된 테이블 이름을 location.state 또는 localStorage 로부터 확인한 뒤, 백엔드 API /analyze/:tableName 으로 분석 데이터를 요청한다. 이 API 는 테이블명에 해당하는 데이터에서 core 기준과 task 기준으로 min, max, avg 값들을 각각 계산하여 반환한다.

2) 분석 기준 선택 (Core, Task)

화면에는 두 개의 버튼이 제공되며, 사용자는 Task 기준 보기 또는 Core 기준 보기 중 하나를 선택할 수 있다. 선택된 값(viewType)에 따라 시각화에 사용되는 데이터가 결정된다. 버튼을 클릭하면 useEffect 가 다시 실행되어 해당 기준의 데이터를 받아온다

3) 차트 데이터 구성

서버로부터 받은 데이터는 labels 와 datasets 으로 가공된다. labels 에는 core 또는 task 의 이름이 들어가고, datasets 에는 각각 Min, Max, Avg 값을 나타내는 데이터 배열이 포함된다. 각 데이터셋은 색깔을 다르게 지정해 시각적으로 구분할 수 있도록 설정하였다.

4) 차트 렌더링

구성된 chartData 를 바탕으로, 화면에는 Bar Chart, Line Chart, Radar Chart, Polar Area Chart 가 순차적으로 렌더링된다. 동일한 데이터를 다양한 형태로 시각화함으로써 사용자가 통계 정보를 직관적으로 비교할 수 있도록 하였다.

5) 예외 처리 및 안내 문구

로컬스토리지에 테이블 정보가 없을 경우, 사용자에게 "먼저 FileUpload 에서 데이터를 업로드하세요"라는 경고 문구를 출력하여 사용 흐름을 유도한다.

이와 같은 구성은 단순한 수치 데이터를 그래픽 형태로 전환하여 사용자 경험(UX)을 향상시키며, Core 및 Task 별 성능을 보다 효과적으로 비교하고 분석할 수 있도록 설계되었다.

3.4 DB 목록 보기

사용자는 업로드한 .xlsx 파일이 데이터베이스에 저장되었는지 확인하고, 저장된 데이터를 직접 선택하거나 삭제할 수 있다. 이를 위해 Database.jsx 페이지를 구현하였으며, 다음과 같은 과정을 통해 DB 목록 확인 기능이 제공된다.

1) 데이터베이스 테이블 목록 요청

컴포넌트가 마운트되면 useEffect 를 통해 백엔드 API /api/profile 에 GET 요청을 보낸다. 이 API 는 현재 데이터베이스에 저장되어 있는 테이블 이름 목록을 배열 형태로 반환한다. 반환된 목록은 상태 변수 files 에 저장된다.

2) 목록 출력

요청 결과가 로딩 중일 경우 "파일 목록을 불러오는 중..."이라는 메시지를 출력하고, 저장된 테이블이 없을 경우 "저장된 파일이 없습니다."라는 메시지를 보여준다. 테이블이 존재할 경우에는 각각을 리스트 항목(<li>)으로 출력하며, 항목은 클릭 또는 삭제가 가능하다.

3) 테이블 선택 → 차트 페이지 이동

사용자가 리스트 항목을 클릭하면 해당 테이블 이름을 navigate 의 state 로 넘겨 /chart 경로로 이동한다. 이때 Chart.jsx 에서는 전달받은 테이블명을 기반으로 시각화 데이터를 불러온다.

4) 테이블 삭제 기능

리스트 항목 우측에 위치한 삭제 버튼을 누르면 DELETE /api/profile/drop/:tableName API 가 호출되어 해당 테이블이 데이터베이스에서 제거된다. 삭제 전에는 window.confirm()을 통해 사용자에게 확인을 요청하며, 삭제 후에는 상태에서 해당 항목을 제거하여 실시간으로 목록이 갱신된다.

이러한 DB 목록 페이지는 사용자에게 현재까지 저장된 분석 테이블을 확인하고 관리할 수 있는 기능을 제공함으로써, 데이터 탐색과 시각화 흐름을 더욱 직관적이고 효율적으로 연결해주는 중요한 역할을 한다.

4. 프론트 엔드 시각화 및 로직

4.1 폴더 구조

폴더명	역할 설명
src/	React 애플리케이션의 핵심 소스 코드가 위치하는 폴더. 컴포넌트, 페이지, 스타일, 유틸리티 포함.
src/components/	재사용 가능한 UI 요소나 작은 기능 단위 컴포넌트들이 모여있는 폴더
src/pages/	주요 페이지 또는 뷰 단위 컴포넌트가 위치하는 폴더.
node_modules/	프로젝트에 설치된 외부 라이브러리(패키지)들이 저장되는 폴더.

4.2 FileUpload.jsx

FileUpload 컴포넌트에서는 파일 선택과 업로드, 서버와의 비동기 통신, 업로드 결과 확인 및 분석 데이터 가공의 전반적인 흐름이 체계적으로 구현되어 있다. 파일 업로드 성공 및 실패 시 상태 메시지를 즉시 표시하여 사용자에게 현재 진행 상황을 명확히 알리고, 업로드 후에는 백엔드로부터 전달받은 데이터를 바탕으로 Core 및 Task 기준의 분석 결과를 동적으로 표 형태로 출력된다.

4.3 chart.jsx

ChartPage 는 업로드된 데이터의 성능 지표를 시각적으로 확인할 수 있는 페이지로, Chart.js 라이브러리를 활용해 다양한 차트(bar, line, radar, polar area)를 제공한다. 서버로부터 받은 데이터를 Chart.js 형식에 맞게 가공해 차트를 동적으로 렌더링하며, 버튼 클릭만으로 Task 단위와 Core 단위 시각화를 전환할 수 있다. 또한 각 차트에서는 Min, Max, Avg 각각의 버튼을 통해 특정 항목을 선택하거나 제거하여 원하는 데이터만 볼 수 있어, 보다 맞춤형 비교가 가능하다. 차트 전환은 Chart.js 의 데이터 업데이트 기능을 활용해 간단하게 구현되었으며, 데이터가 없을 경우에는 안내 메시지를 통해 사용자 혼선을 방지한다. 이처럼 ChartPage 는 직관적이고 효율적인 데이터 분석을 지원한다.

4.4 Database.jsx

Database 컴포넌트는 서버에서 파일 목록을 불러와 시각적으로 구분되는 리스트 항목 형태로 표시하며, 각 파일을 클릭하면 해당 데이터의 분석 차트 페이지로 이동할 수 있도록 구현했다. 각 항목 옆에는 삭제 버튼을 배치하여 단일 인터페이스 내에서 파일 선택과 삭제를 동시에 처리할 수 있다. 삭제 전 사용자에게 확인을 요청하여 원치 않는 삭제를 막는다.

4.5 공통 레이아웃

공통 레이아웃은 전체 페이지에 일관된 구조와 디자인을 제공하여 사용자에게 통일된 경험을 전달한다. 하단(Footer)에는 팀 이름인 LeGend 와 소스 코드를 확인할 수 있는 GitHub 저장소 링크가 포함되어 있다.

4.6 bootstrap5

프론트엔드 화면 구성에는 Bootstrap 5 유틸리티와 컴포넌트 클래스를 적극적으로 활용했다. 이를 통해, 버튼, 카드, 테이블, 폼 및 그리드 레이아웃 등 시각 요소 대부분을 DOM 에서 즉시 적용 가능한 Bootstrap 의 클래스 조합으로 구현하여 별도의 복잡한 스타일링 없이도 깔끔하면서 반응형이 적용된 사용자 인터페이스를 완성했다. 클래스는 적재적소에 적용되어 레이아웃의 일관성과 시각적 통일감을 높였으며, 각 기능별 인터페이스도 구조에 맞게 직관적으로 구현했다.

6. 프로젝트 기여

- **20230815 권오현 : 25%**
- **20230883 홍민경 : 25%**
- **20230003 김민규 : 25%**
- **20230878 천병권 : 25%**