1. **프로젝트**

* **주제**: 토트넘 경기 데이터 기반 웹 대시보드 개발
* **목적:** 축구 경기 데이터를 시각화하여 통계와 통찰을 사용자에게 제공
* **사용 기술 스택:**
  + 백엔드: Flask, SQLAlchemy
  + 데이터베이스: MySQL
  + 프론트엔드: HTML, CSS, JavaScript (Chart.js, jQuery)
  + API 개발: Python

1. **구현 과정**

**2-1. API 설계:**

* /api/matches 엔드포인트를 통해 경기 데이터를 JSON으로 제공
* B.Web-SRC-2315726-인공지능공학부-이가은.js에서 API 데이터를 호출하여 UI를 동적으로 렌더링

**2-2. 프론트엔드 구성**

* + - HTML로 테이블과 필터 UI 구성 (B.Web-SRC-2315726-인공지능공학부-이가은.html 참조)
    - JavaScript를 이용하여 사용자가 선택한 조건에 맞게 데이터를 필터링

**2-3. 데이터 시각화**

* Chart.js를 활용하여 시즌별 승률 데이터를 시각화
* 테이블과 차트를 연동하여 동적 데이터 표시

1. **주요 기능**

* 데이터 필터링
  + 텍스트, 소프트웨어, 번호, 스크린샷이(가) 표시된 사진

    자동 생성된 설명상대팀 이름, 경기 날짜, 경기 장소를 기준으로 데이터 검색

그림 1: 스크린샷 2024-11-17 오후 9.53.31

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그림 2: 스크린샷 2024-11-17 오후 9:53:59

* 경기 결과 시각화
  + 시즌별 승률과 경기 데이터를 차트 및 테이블로 표시

스크린샷, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그림 3: 스크린샷 2024-11-17 오후 9.53.39

1. **코드 설명**

4-1. JavaScript (B.Web-SRC-2315726-인공지능공학부-이가은.js)

기능: API에서 데이터를 가져와 테이블 생성 및 필터링 적용

코드

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그림 4: 스크린샷 2024-11-17 오후 11.14.39

4-2. HTML (B.Web-SRC-2315726-인공지능공학부-이가은.html)

* 구성: 필터링 UI, 경기 결과 테이블, 차트를 포함한 웹페이지 구조 정의
* 코드

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그림 5: 스크린샷 2024-11-17 오후 11.15.57

4-3. 코드 설명

* Football Data API를 활용하여 2022-2024 시즌 동안의 토트넘 경기 데이터를 수집했습니다.
* 주요 데이터 필드
  + - 경기 날짜 (Date)
    - 상대 팀 이름 (Opponent)
    - 장소 (Location)
    - 득점 (Tottenham Goals, Opponent Goals)
* 데이터를 수집한 후 CSV 파일로 저장하고, MySQL 데이터베이스에 업로드했습니다.
* Python 코드:

*import* pandas *as* pd

*from* sqlalchemy *import* create\_engine

*# 데이터 수집*

url = "https://api.football-data.org/v2/competitions/PL/matches"

headers = {"X-Auth-Token": "YOUR\_API\_KEY"}

response = requests.get(url, *headers*=headers)

data = response.json()

*# 데이터프레임으로 변환*

df = pd.json\_normalize(data['matches'])

df.to\_csv('tottenham\_matches.csv', *index*=False)

*# MySQL 업로드*

engine = create\_engine('mysql+pymysql://root:password@localhost/football\_db')

df.to\_sql(*name*='tottenham\_matches', *con*=engine, *if\_exists*='replace', *index*=False)

* API로부터 불필요한 데이터를 제거하고, 필드를 정리했습니다.
* 예: 날짜 형식 변환, 결측값 처리
* 득점 차이: goal\_difference = TottenhamGoals - OpponentGoals
* 경기 결과:
* 승리: TottenhamGoals > OpponentGoals
* 무승부: TottenhamGoals == OpponentGoals
* 패배: TottenhamGoals < OpponentGoals
* Python 코드:

df['goal\_difference'] = df['TottenhamGoals'] - df['OpponentGoals']

df['Result'] = df.apply(

lambda *row*: 'Win' *if* *row*['TottenhamGoals'] > *row*['OpponentGoals']

*else* ('Lose' *if* *row*['TottenhamGoals'] < *row*['OpponentGoals'] *else* 'Draw'), *axis*=1

)

* Flask를 활용하여 MySQL 데이터를 사용자에게 제공합니다.
* 주요 엔드포인트:
  + /api/matches: 모든 경기 데이터를 JSON으로 반환
* Python 코드

*from* flask *import* Flask, jsonify

*import* pandas *as* pd

app = Flask(\_\_name\_\_)

*# 데이터베이스 연결*

df = pd.read\_sql("SELECT \* FROM tottenham\_matches", *con*=engine)

@app.route('/api/matches', *methods*=['GET'])

def get\_matches():

*return* jsonify(df.to\_dict(*orient*="records"))

*if* \_\_name\_\_ == '\_\_main\_\_':

app.run(*debug*=True)