TERM PROJECT REPORT

CSCI6370 Database Management

Astha Jain, Sanskruti Reddy Donthi, Preeti Chatterjee

College Football Dashboard

12.16.2022

TITLE

American College Football Dashboard - A dashboard to view all the college football teams, the games played by them, player information, win-to-loss ration, and view teams that played games in same stadiums.

ORIGINAL CONTRIBUTIONS (WHAT IS NEW?)

For fans who follow sports everyday and while we use the regular streaming apps, we always wondered if it needed a little improvement. We live in a world right now where content is not the issue, streaming is at a high speed, data caps are not an issue anymore. The only issue is the way the content is portrayed to the user and we designed a fun and intelligent way to interact with college football statistics.

BROADER IMPACT (HOW IS IT USEFUL TO SOCIETY?)

The American College Football Dashbaord is a one-stop solution to view football match data over a period of 18 years in detail. Users will not only be able to view the game details, but also player information, win-to-loss ration, and many more interesting insights.

This project opens up many possibilities to work with trending frameworks like Spring Boot tied up with persistent APIs faster than ever before. Web application development has become crucial to design an intelligent way to interact with huge datasets. Although we did this project for Sports & Entertainment domain, we think the knowledge that we acquired can help us compete with the cream of the latest software developers.

SYSTEM ARCHITECTURE

Backend:

We developed the backend using JavaSpring Boot framework. To ingest the data from CSV files to MySQL database, we have used Spring batch and Spring JPA repositories.

We used Spring Mode-View-Controller architecture to design the web application. MVC design pattern allows features like dependency injection, inversion of control, etc.

Database:

We have worked on a huge dataset that we downloaded from kaggle in the form of CSV. We pre processed the data to get rid of redundant data and stored it in tables in MySQL database.
We have worked extensively by performing query operations on the dataset, using complex queries like one degree of separation.

Frontend:

We have created an interactive frontend and integrated our queries in backend with frontend for effective user interaction. For Frontend, we have used React.js.

SAMPLE QUERY SCREENSHOTS

Game Repository:

One Degree of Separation:

This query takes the two team names that participated in a game and the stadium where the game was played as parameters and gives all the combinations of teams similar to team 1 but different from team 2 that also played in the same stadium.

```
@Query(value = "SELECT DISTINCT g1.home_team_code as team1, g1.visit_team_code as team2, g1.stadium_code, g2.home_team_code as team3, g2.visit_team_code as team4 FROM game g1 "
        + "INNER JOIN game g2 ON g1.home_team_code = g2.home_team_code AND g1.visit_team_code != g2.visit_team_code AND g1.stadium_code = g2.stadium_code "
        + "WHERE g1.home_team_code = ?1 AND g2.visit_team_code = ?2 AND g1.stadium_code = ?3 LIMIT 100", nativeQuery = true)
public List<List<Object>> getOneDegreeOfSeparationOfStadiumsBetweenTeams(@Param("homeTeamCode") Team team1, @Param("visitTeamCode") Team team2, @Param("stadiumCode") Stadium stadium);
```

Figure 1: One Degree of Separation between two teams who played in same stadium

This query gives us the game details based on the team codes ordered in descending order of dates.

```
@Query("select g from Game g where (g.HomeTeamCode = :teamCode1 or g.VisitTeamCode = :teamCode2) order by Date desc")
List<Game> getByTeam1OrTeam2OrderByDateDesc(Team teamCode1, Team teamCode2, Pageable pageable);
```

Figure 2: Fetch all games reverse sorted by dates

This query gives us the games played by a team between particular dates in descending order.

```
@Query("select g from Game g where (g.HomeTeamCode = :teamCode or g.VisitTeamCode = :teamCode) and g.Date between :dateStart and :dateEnd order by Date desc")
List<Game> getGamesByTeamBetweenDates(
    @Param("teamCode") Team teamCode,
    @Param("dateStart") LocalDate dateStart,
    @Param("dateEnd") LocalDate dateEnd
);
```

Figure 3: Fetch all games between two dates

This query finds the latest games played by a team. We've used pagination to work with this query.

```
default List<Game> findLatestGamesbyTeam(Team teamCode, int count) {
    return getByTeam1OrTeam2OrderByDateDesc(teamCode, teamCode, PageRequest.of(page: 0, count));
}
```

Figure 4: Fetch latest games in decreasing order of dates

Team Repository:

This query gives us all the details of a team when a particular team name is selected.

```
@Query("SELECT t from Team t where t.Name = ?1")
public Team findByName(String name);
```

Figure 5: Fetch teams by their name

This query gives us all the details of a team when team code is selected.

```
@Query("SELECT t from Team t where t.TeamCode = ?1")
public Team findTeamByTeamCode(Long teamCode);
```

Figure 6: Fetch teams by their team code

Stadium Repository:

This query gives us the stadium details when a stadium code is entered.

```
@Query("SELECT s from Stadium s where s.StadiumCode = ?1")
public Stadium findStadiumByStadiumCode(Long stadiumCode);
```

Figure 7: Fetch stadium by their code

```
@Query("SELECT s from Stadium s where s.Name = ?1")
public Stadium findStadiumByStadiumName(String name);
```

Figure 8: Fetch stadium by their name

Conference Repository:

This query gives us the conference details given the conference code.

```
@Query("select c from Conference c where c.ConferenceCode = ?1")
List<Conference> getByConferenceCode(Long conferenceCode);
```

Figure 9: Fetch conference information by the conference code

Player Repository:

This query gives all the player details when a player's team code is entered.

```
@Query("select p from Player p where p.PlayerTeamCode = :teamCode")
List<Player> getByTeamCode(Team teamCode);
```

Figure 10: Fetch player information by their team code

TEAM MEMBERS AND RESPONSIBILITIES

1. Astha Jain
    - Developed the backend of the project using Java Spring Boot framework.
    - Ingested data from CSV files to MySQL database using Spring Batch and Spring JPA.
    - Formulated queries to fetch data from the database and feed to REST APIs.
    - Developed the frontend components using React JS.
    - Extracted the schema of college football database using MySQL.
    - Contributed in drafting the project report.
    - Created and updated the README file.
    - Recorded the demo video.

2. Sanskruti Reddy Donthi
    - Contributed in setting up the database by integration data from 2 different datasets on Kaggle.
    - Worked on the missing data that was present in the dataset by searching online.
    - Contributed towards the project report.

3. Preeti Chatterjee
    - Wrote the basic structure of frontend.
    - Contributed towards the project report.

CONCLUSION

We used Java to code our backend using frameworks like spring boot, Java persistent APIs, Spring MVC APIs, JPA repositories. We used Spring Batch/JPA to connect Backend to the database. We also had to improve the data set quality by combining data from various sites that made us learn about data cleaning and pre-processing.

This project made us explore all the above mentioned frameworks and get used to how we can build a full stack web application from scratch.

REFERENCES

1. https://www.kaggle.com/datasets/jeffgallini/college-football-team-stats-2019
2. https://www.kaggle.com/datasets/mhixon/college-football-statistics
3. https://www.kaggle.com/datasets/jeffgallini/college-football-attendance-2000-to-2018
4. https://www.ncaa.com/stats/football/fbs
5. https://spring.io/guides/gs/batch-processing/

APPENDIX: USER INTERFACE

Figure 11 & 12: Dashboard Home Page

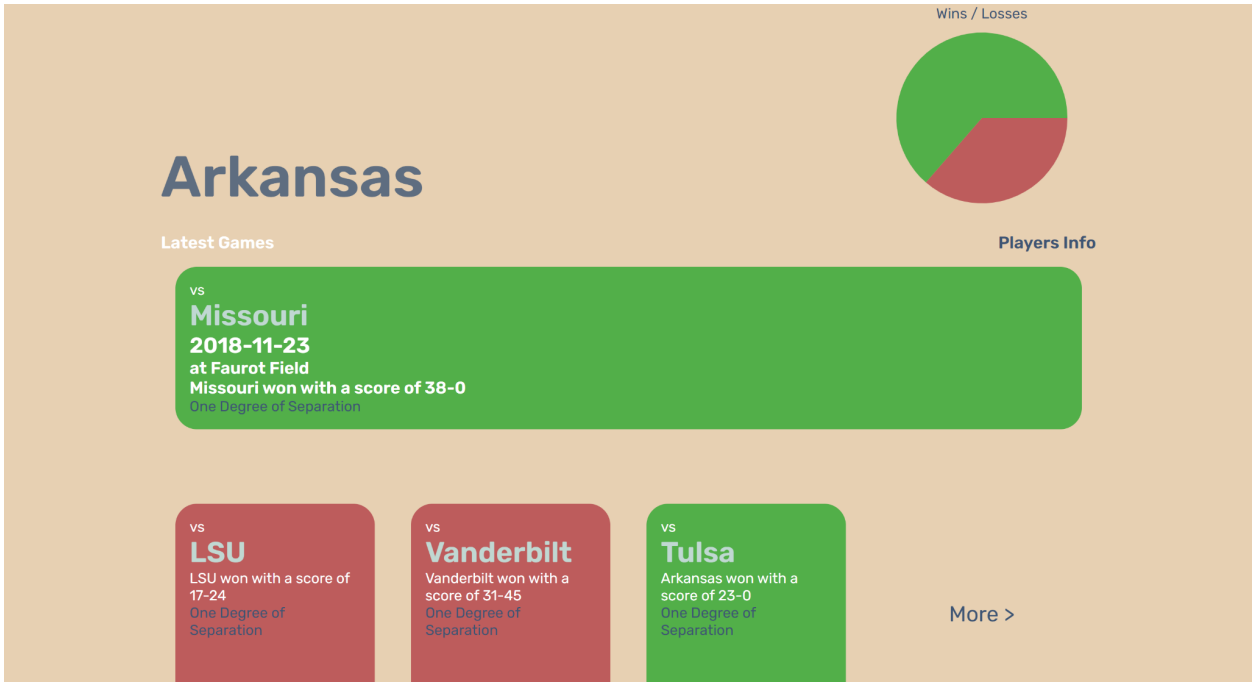

Figure 12: Dashboard Team and Match Details Page

Figure 13: Dashboard Game History Page

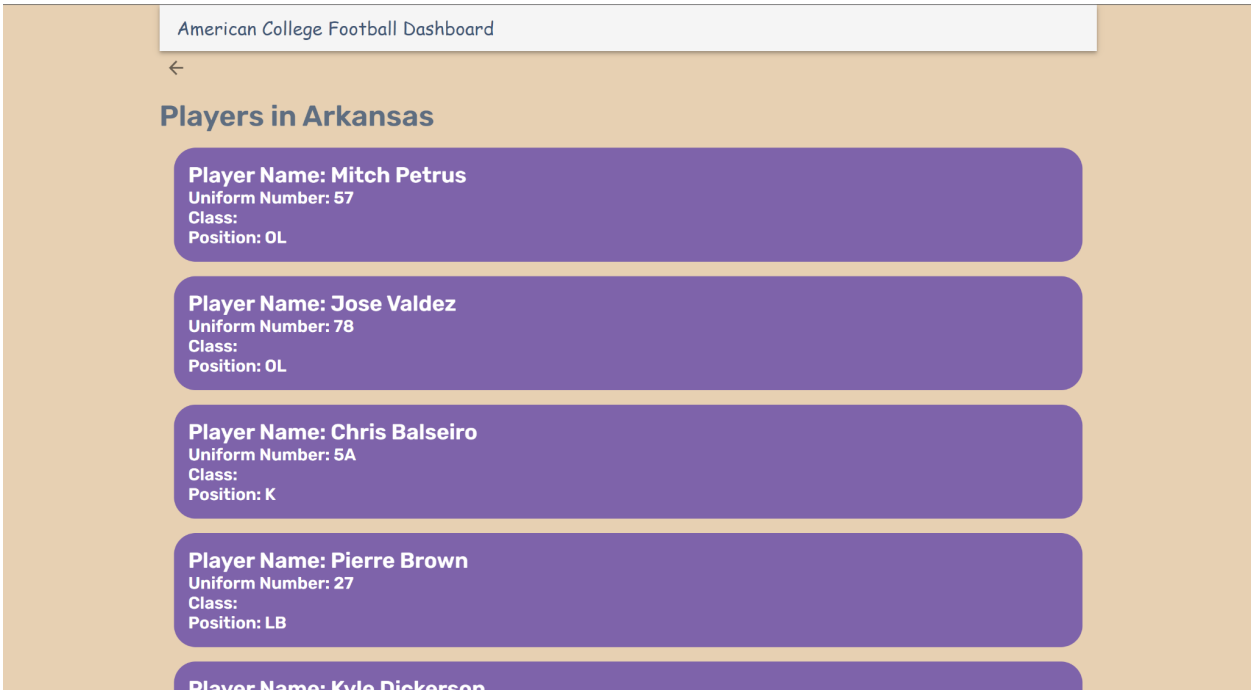

Figure 14: Dashboard Player Details Page

Figure 15: Dashboard One Degree of Separation Page