A PROJECT REPORT

On

**Premier Zone Fantasy**

*Submitted by*

Mr. AKSHAY K BABURAJ

*in partial fulfilment for the award of the degree*

*of*

**BACHELOR OF SCIENCE**

in

**COMPUTER SCIENCE**

*under the guidance of*

**prof. Pallavi Awate**

**Department of Computer Science**

****

**SIES (Nerul) College of Arts, Science and Commerce (Autonomous)**

**(Sem VI)**

**(2024 – 2025)**

****

**SIES (Nerul) College of Arts, Science and Commerce (Autonomous)**

**Sri Chandrasekarendra Saraswathy Vidyapuram, Plot 1-C, Sector V, Nerul, Navi Mumbai-400 706.**

**Department of Computer Science**

**CERTIFICATE**

This is to certify that Mr. AKSHAY K BABURAJ of T.Y.B.Sc. **(Semester VI)** class has satisfactorily completed the Project **Premier Zone Fantasy** , to be submitted in the partial fulfilment for the award of **Bachelor of Science** in **Computer Science** during the academic year 2024 – 2025.

Seat no:

**Date of Submission:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Project Guide Head of Department**

**(prof. Pallavi Awate) ( prof. Dr. Sheeja K. )**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**College seal Signature of Examiner**

**DECLARATION**

I, Mr. **AKSHAY K BABURAJ**, hereby declare that the project entitled “**Premier Zone Fantasy**” submitted in the partial fulfilment for the award of **Bachelor of Science in Computer** Science during the academic year **2024 – 2025** is my original work and the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles**.**

**Signature of the Student:**

**Place:**

**Date:**

**ACKNOWLEDGEMENT**

In addition to my own efforts, a lot of other people's support and direction are crucial to the project's success. I would want to take this opportunity to thank everyone who has contributed to the successful completion of this project.

With great appreciation, I would like to thank my project guide, Professor Pallavi Awate, for her friendly assistance, direction, and supervision. She also provided the necessary information about the project and helped me finish it.

I would also like to express my gratitude and appreciation to the administrators of my college's computer lab for their assistance in getting the necessary materials to me in time for my assignment.

Lastly, I would want to express my sincere gratitude to my parents, friends, and well-wishers for supporting me and being there for me during the project**.**

**PREFACE**

**Premier Zone Fantasy** is a full-stack web application dedicated to everything related to the Premier League. The project aims to provide an immersive and interactive experience for football enthusiasts, offering comprehensive data, real-time updates, and engaging features.

Built using React for the frontend and Spring Boot for the backend, with PostgreSQL as the database, this platform integrates various functionalities to enhance user experience. It aggregates live match data, player statistics, team standings, and historical records while allowing user interaction through comments and head-to-head comparisons.

This document serves as a comprehensive guide to the development, architecture, and functionalities of **Premier Zone Fantasy**. It details the project structure, technologies used, core features, and implementation methodologies. Whether you are a developer contributing to the project or an end-user seeking insights into its functionality, this document will provide essential information.

The application employs secure authentication using JWT tokens, with user registration and login functionalities backed by PostgreSQL. A welcome email feature is integrated to enhance user engagement. Data is efficiently managed using caching mechanisms like Spring Caffeine to optimize performance. Unit testing and API validation are handled through JUnit and Mockito, ensuring a robust and error-free system.

The platform is hosted on Render for seamless deployment, with version control managed via GitHub. Additional tools such as Postman for API testing, Canva for design elements, and pgAdmin for database management are utilized.

Through this document, we aim to provide a structured reference for understanding the various components, modules, and integrations that make up **Premier Zone Fantasy**. This ensures consistency, maintainability, and future scalability of the project.

**Happy coding and enjoy exploring the world of the Premier League through Premier Zone Fantasy!**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **SR.NO** | **CONTENTS** | **PAGE NO** |
| **1.** | **TITLE** | **1** |
| **2.** | **ABSTRACT** | **2** |
| **3.** | **INTRODUCTION** | **3** |
| **4.** | **TOOLS AND TECHNOLOGIES** | **4-5** |
| **5.** | **Software and Hardware Requirements** | **6-7** |
| **6.** | **FEASIBILITY STUDY** | **8-10** |
| **7.** | **SYSTEM ANALYSIS** | **11-12** |
| **8.** | **Gantt Chart** | **13** |
| **9.** | **System Design**   * FLOWCHART * USE CASE DIAGRAM * CLASS DIAGRAM * ER DIAGRAM * SEQUENCE DIAGRAM | **14-18** |
| **10.** | **DATA DICTIONARY** | **19** |
| **11.** | **CODE IMPLEMENTATION** | **20-61** |
| **12.** | **RESULTS** | **62-64** |
| **13.** | **CONCLUSION** | **65** |
| **14.** | **FUTURE SCOPE** | **66** |
| **15.** | **PLAGIARISM REPORT** | **67** |
| **16.** | **REFERENCES** | **68** |

TITLE

It provides me a great opportunity to present this project on the topic

**“Premier Zone Fantasy!”**

*(Your home for everything Premier League related)*





**ABSTRACT**

**Premier Zone Fantasy** is a full-stack web application designed to provide football enthusiasts with an all-in-one platform for everything related to the English Premier League. The project utilizes a React-based frontend and a Spring Boot backend, with PostgreSQL serving as the primary database for storing user-related data, comments, and player information.

The application features multiple components, including real-time match updates, team standings, player statistics, and historical records. Data for matches, news, and team comparisons are fetched dynamically from external APIs, ensuring users receive up-to-date information. Users can search for specific players or teams within the database, and additional filtering options allow for categorization based on position, nationality, and club affiliation.

A personalized dashboard aggregates key insights such as today’s matches, upcoming fixtures, top scorers, and breaking news, offering users quick navigation to relevant sections. The system supports user authentication via JWT tokens, and newly registered users receive a welcome email. A comment system is integrated within match detail sections, allowing for interactive discussions among users.

The project employs modern development tools and frameworks, including VS Code for frontend development, IntelliJ IDEA for backend development, PostgreSQL managed through pgAdmin and SQL Shell, Maven as the build tool, and Postman for API testing. Deployment is handled via Render, ensuring a scalable and secure hosting environment.

Testing is conducted using JUnit and Mockito, while Spring Caffeine is implemented for caching to optimize performance. Spring Mail facilitates email notifications, and FontAwesome enhances UI design with intuitive icons. The project embodies a seamless blend of functionality and aesthetics, making it a comprehensive resource for Premier League fans worldwide.

**INTRODUCTION**

In the digital age, football enthusiasts seek comprehensive platforms that provide real-time updates, in-depth statistics, and engaging content related to their favourite leagues and teams. The **Premier League**, being one of the most followed football leagues worldwide, has a massive fan base that constantly craves accurate and detailed insights. Whether it’s match results, player statistics, team standings, or historical records, fans demand a centralized hub that caters to all aspects of the league.

**Premier Zone Fantasy** is a full-stack web application designed to fulfil this need by offering a feature-rich platform where users can access a wide range of Premier League data. Built using React for the frontend and SpringBoot for the backend, the application is supported by PostgreSQL for database management. The project integrates real**-**timedatafetching from APIs for live match updates, news, head-to-head comparisons, team standings, and top scorers while also maintaining a database-driven approach for users, comments, and fantasy records.

To enhance user experience, the platform includes several key components such as a dashboard that provides a summary of current and upcoming matches, quick access to records, and direct links to different sections. Users can explore playerstatistics based on their position, teams, and nationality, enabling them to find specific information effortlessly. The search feature allows users to look up teams and players from the database, making navigation intuitive and efficient. Additionally, each match has a comment section where users can engage in discussions, with all comments stored in the database.

For security and authentication, the application utilizes JWT tokens, ensuring secure access for registered users. The cache mechanism, implemented using Spring Caffeine, improves performance by optimizing data retrieval. Moreover, JUnit and Mockito have been used for rigorous backend testing, ensuring reliability and stability.

The project is deployed using Render, providing a seamless hosting solution for both the frontend and backend. Tools like Postman were used for API testing, while GitHub ensures proper version control and collaboration. The overall goal of **Premier Zone Fantasy** is to create an engaging and interactive platform for Premier League enthusiasts, combining real-time data, user participation, and a well-structured interface for an immersive football experience.

**TOOLS AND TECHNOLOGIES**

* **Frontend Technologies**
* **React.js** – Used for building the user interface, ensuring a responsive and dynamic user experience.
* **FontAwesome** – Provides high-quality icons for UI elements.
* **NPM (Node Package Manager)** – Manages dependencies and packages in the React project.
* **SCSS** – Enhances styling and responsiveness.
* **Canva** – Used for designing logos and graphical assets.
* **Backend Technologies**
* **Spring Boot** – Powers the backend with a robust and scalable framework for handling business logic.
* **Java 23** – The programming language used for backend development, ensuring performance and reliability.
* **Spring Security (JWT Authentication)** – Secures the application by implementing user authentication and authorization.
* **Spring Mail** – Handles email functionality, such as sending welcome emails to new users.
* **Spring Caffeine Cache** – Improves performance by caching frequently accessed data.
* **JUnit & Mockito** – Used for writing and executing unit and integration tests to ensure backend reliability.
* **Database & API**
* **PostgreSQL** – The relational database used to store user data, comments, and other application-specific records.
* **pgAdmin & SQL Shell (PL/pgSQL)** – Tools used for managing and querying the PostgreSQL database.
* **REST APIs (Fetched Data)** – Integrated to fetch real-time match updates, team standings, player statistics, news, and head-to-head comparisons.
* **Development & Testing Tools**
* **IntelliJ IDEA Community Edition** – The primary IDE used for backend development.
* **Visual Studio Code (VS Code)** – Used for frontend development.
* **Postman** – Used for testing APIs and ensuring proper backend responses.
* **Git & GitHub** – Version control system used for tracking changes and collaborating on the project.
* **Browser developer Tools (Chrome)** – built-in utilities in web browsers that help developers inspect, debug, and optimize web applications in real time.
* **Deployment & Hosting**
* **Render** – Cloud platform used for deploying the full-stack application (frontend, backend, and database).
* **Docker (if used)** – Helps in containerizing the application for smoother deployment and scalability.

**Software and Hardware Requirements**

**Software Requirements:**

### **Frontend Development**

* **Operating System:** Windows 10/11, macOS
* **Development Tools:**
  + Visual Studio Code (VS Code)
  + Node.js & NPM (for React.js dependencies)
  + Git (for version control)
* **Browser:** Chrome, Firefox, Edge (for testing and debugging)
* **Backend Development**
* **Operating System:** Windows 10/11, macOS
* **Development Tools:**
  + IntelliJ IDEA Community Edition (for Spring Boot development)
  + Java Development Kit (JDK 23)
  + Maven (for dependency management)
  + Postman (for API testing)
* **Backend Framework & Libraries:**
  + Spring Boot
  + Spring Security (for authentication using JWT)
  + Spring Mail (for email notifications)
  + JUnit & Mockito (for testing)
  + Spring Caffeine Cache (for caching)
* **Database Management**
* **Database:** PostgreSQL
* **Database Management Tools:**
  + - * pgAdmin
      * SQL Shell (PL/pgSQL)
* **Deployment & Hosting**
* **Cloud Platform:** Render (for backend, frontend, and database hosting)
* **Version Control:** GitHub (for code management and collaboration)
* **API Integration:** Third-party APIs for fetching match data, news, standings, and statistics

**Hardware Requirements:**

* For Development Environment

Processor Intel Core i5 (8th Gen) / AMD Ryzen 5

RAM 8GB DDR4 or higher

Storage 256GB SSD / HDD or higher

Graphics Integrated GPU/ Dedicated GPU (for better performance)

Internet Broadband Connection (10 Mbps)

* For End-Users (Web Application Users)

**Device:** PC, Laptop, Tablet, or Smartphone

**Browser Compatibility:** Chrome, Firefox, Edge, Safari

**Internet Connection:** Stable broadband or mobile data

**FEASIBILITY STUDY**

The **feasibility study** for **Premier Zone Fantasy** assesses its practicality, viability, and sustainability in terms of technical implementation, economic factors, operational efficiency, and legal compliance. The study helps determine whether the project is achievable within the given constraints and resources.

**1. Technical Feasibility**

The project is built using **React.js** for the frontend and **Spring Boot** for the backend, with **PostgreSQL** as the database. These technologies are widely used, well-documented, and support high performance, scalability, and security.

* **Frontend:** React.js ensures a dynamic and responsive UI.
* **Backend:** Spring Boot provides a powerful and scalable REST API with built-in security features.
* **Database:** PostgreSQL is a reliable and efficient relational database system.
* **Hosting & Deployment:** The project will be deployed on **Render**, ensuring continuous integration and deployment.
* **Match Data, News, Standings, and Statistics**: Retrieved using third-party APIs to ensure real-time updates.
* **Authentication & Security**: JWT (JSON Web Token) is used for authentication and authorization.
* **Caching Mechanism**: Spring Caffeine Cache is implemented to improve performance.

**2. Operational Feasibility**

**User Accessibility & Experience**

* **Easy Navigation:** Intuitive dashboard with direct access to matches, standings, records, and more.
* **Real-time Updates:** Fetching live match data ensures an engaging experience.
* **User Authentication:** Secure login and registration with email verification.
* **Commenting System:** Users can interact and discuss match details.

**Scalability & Maintenance**

* **Database Optimization:** PostgreSQL ensures efficient storage and retrieval of player data, user comments, and match records.
* **Continuous Deployment:** GitHub and Render allow seamless updates without downtime.
* **Future Enhancements:** Additional features like fantasy leagues, team-building tools, or AI-based predictions can be incorporated.

Operationally, the system is **user-friendly, scalable, and easy to maintain**.

**3. Economic Feasibility**

**Cost Estimation:**

While the project is designed to be **free to use**, some costs are associated with development and deployment.

| **Component** | **Estimated Cost** |
| --- | --- |
| Development Tools | Free (Open-source tools like VS Code, IntelliJ Community, Postman) |
| Hosting (Render) | Free-tier initially, may require upgrades (~$5-$10/month) |
| Domain Name | Optional (~$10-$15 per year if custom domain is used) |
| API Services | Free-tier available, but premium APIs may cost extra |

**Total Cost Estimate**: **Minimal cost for initial deployment, with scalability options.**

**Revenue Model (Future Considerations)**

* Advertisements and sponsorships from football-related platforms.
* Premium membership for exclusive statistics or analysis.
* Affiliate marketing for football merchandise, tickets, etc.

The project is economically viable, with **minimal costs** and **potential monetization strategies**.

**4. LEGAL & ETHICAL FEASIBILITY**

**Data Privacy & Security**

* **GDPR Compliance**: User data (email, login credentials) is securely stored and encrypted.
* **No Unauthorized Content**: APIs used for match data and news ensure that content is licensed and valid.

**Third-Party API Usage**

* Adhering to API rate limits and licensing agreements is crucial.

**Copyright & Branding Considerations**

* The project does **not use** official **Premier League logos or branding** to avoid copyright violations.
* All images and assets are **either self-designed (Canva) or sourced legally**.

The project complies with **data protection laws and intellectual property rights**, ensuring no legal conflicts.

**SYSTEM ANALYSIS**

The **System Analysis** phase is critical in understanding the requirements, functionality, and workflow of the proposed website. The analysis focuses on gathering information about the system's features, data flow, and architecture, which provides the foundation for designing and developing a scalable, secure, and efficient application.

**1. Existing System Overview**

In the realm of football-related platforms, several websites and applications provide match details, team standings, and player statistics. However, most existing systems come with the following limitations:

* Lack of Centralized Data: Many platforms do not integrate all aspects of football statistics, such as match results, standings, top scorers, and records, into a single user-friendly interface.
* Limited Interactivity: Users often cannot engage with content beyond passive reading; there is no interactive element such as a comment section or personalized recommendations.
* Static Data Presentation: Some systems only display basic statistics without advanced search features or categorization.
* Security Concerns: Existing platforms often have weak authentication mechanisms, increasing the risk of data breaches.
* Scalability Issues: Some applications are unable to handle large amounts of real-time football data efficiently, leading to slow loading times and performance issues.
* Limited Customization: Users have minimal control over content presentation or filtering, making it difficult to find specific information easily.

The proposed system aims to address these challenges by providing a solution that is both **scalable** and **secure**, with a focus on **real-time performance** and **user-friendly design**.

**2. Proposed System Features**

The proposed messaging application is designed to overcome the limitations of existing systems while offering the following features:

#### **User Management & Authentication**

* Secure **user registration and login** with **JWT authentication**.
* **Email verification** upon user registration via Spring Mail.
* **User roles and permissions** to ensure secure access control.

#### **Interactive Features**

* **User Comments & Engagement:** Users can comment on match pages, stored securely in the **PostgreSQL database**.
* **Dashboard Overview:** Quick access to **today’s matches, upcoming derbies, top scorers, latest news, and quick links to various pages**.
* **Quick Link Cards:** Allows users to navigate easily to head-to-head, records, and match pages.

#### **Security & Data Protection**

* **JWT-based authentication** for secure API access.
* **Data encryption and secure access policies** for user information.

**Deployment & Maintenance**

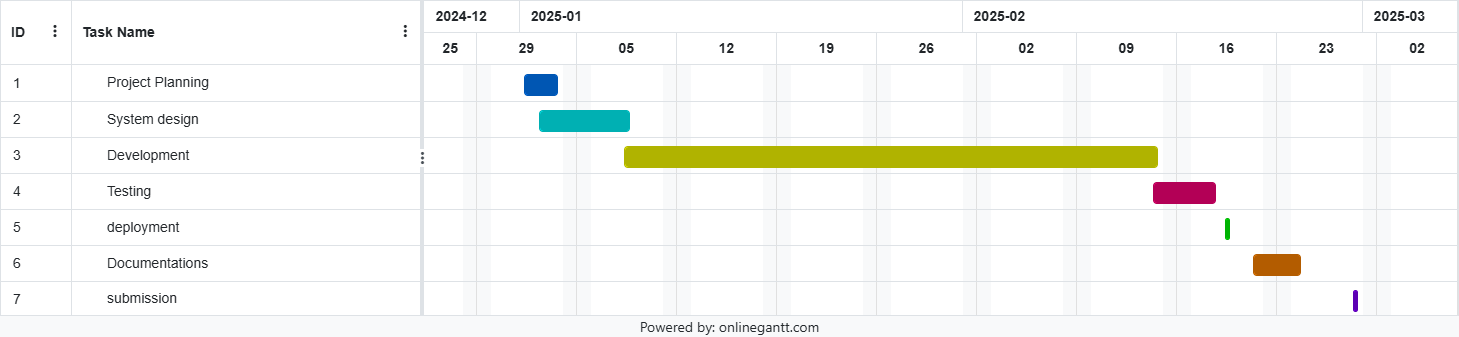
* Hosted on **Render**, supporting **continuous integration and deployment**.
* **Version control via GitHub** for tracking changes and collaborative development.

**Performance Enhancements**

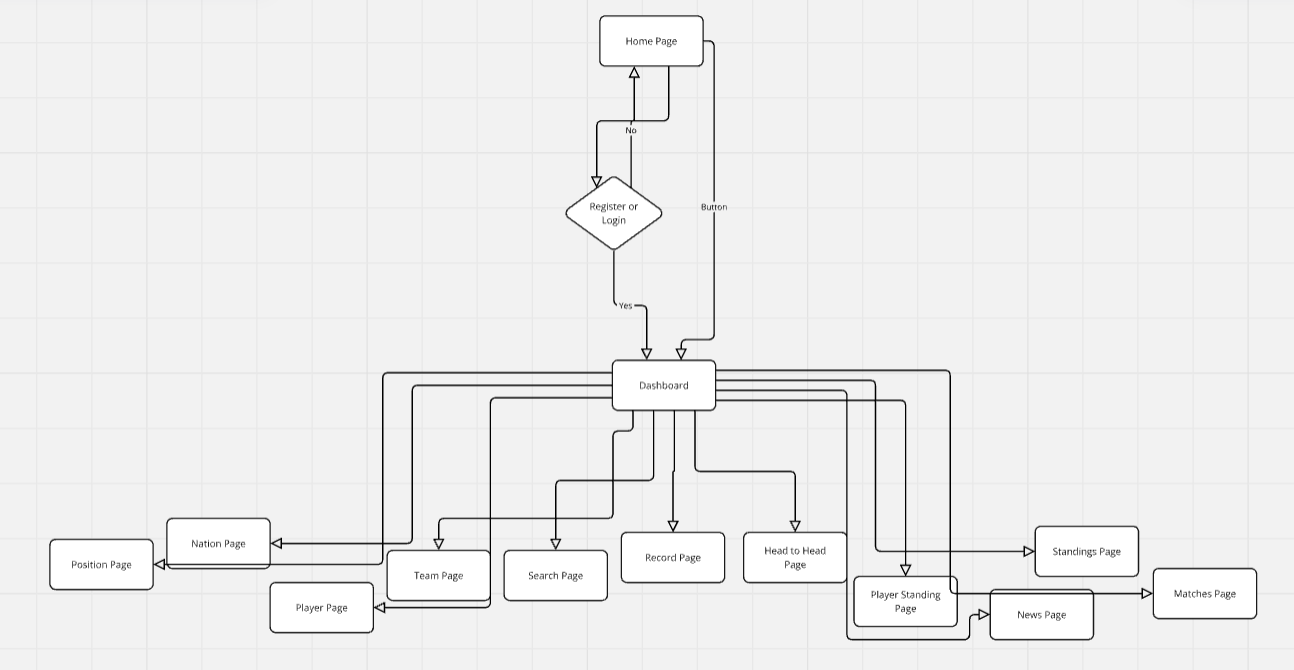
* **Caching with Spring Caffeine Cache** for faster data retrieval and reduced API calls.
* **Efficient database indexing** in PostgreSQL to optimize query performance.
* **Asynchronous processing** for handling real-time data updates.

The Premier Zone Fantasy project is designed to be a scalable, interactive, and user-friendly platform. By integrating multiple football data sources, ensuring high security, and providing a rich user experience, it fills the gap left by existing football statistics websites and applications. The proposed system offers real-time data updates, advanced search functionalities, user engagement features, and strong security measures, making it a comprehensive solution for football enthusiasts worldwide.

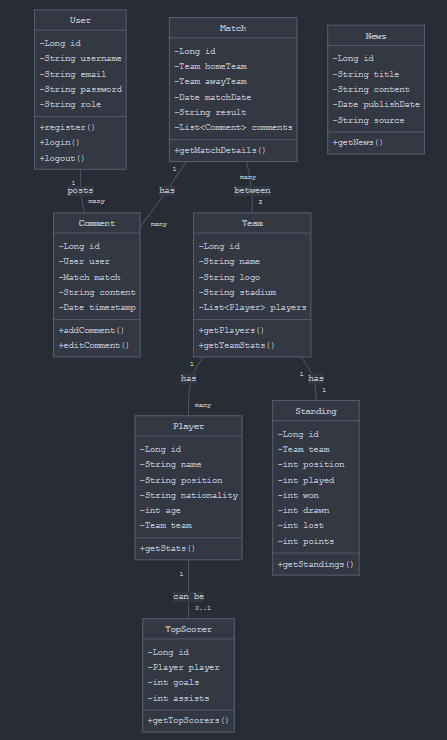
**Gantt Chart**

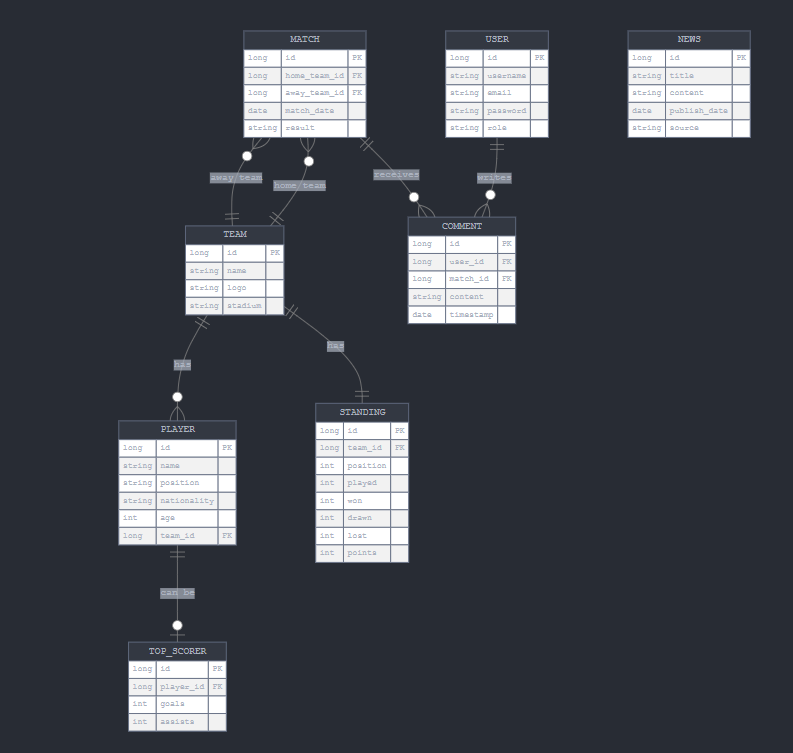
****

**FLOW CHART**

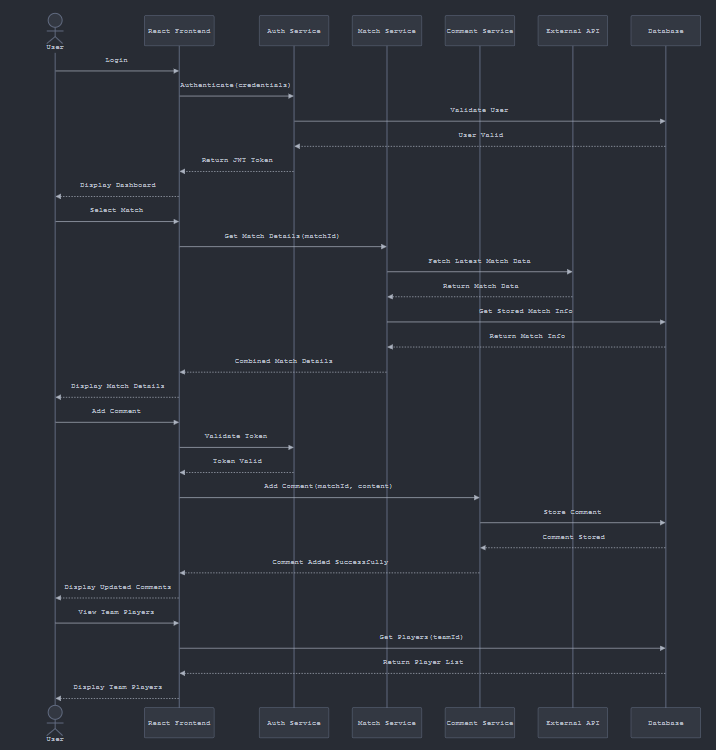


**CLASS DIAGRAM**

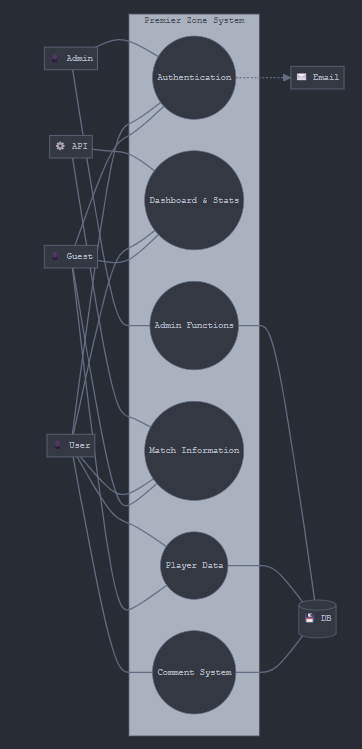
****

**ENTITY RELATIONSHIP DIAGRAM**

**SEQUENCE DIAGRAM**

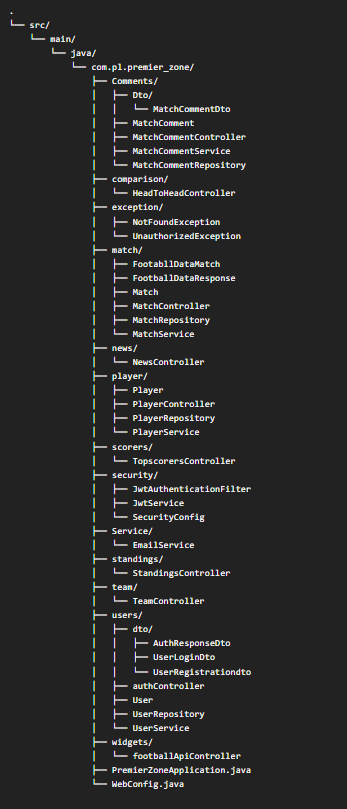
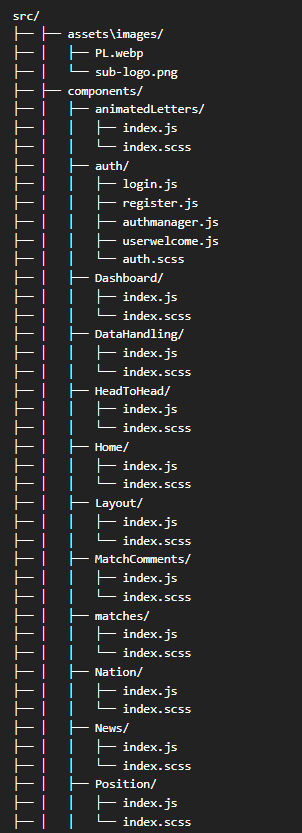
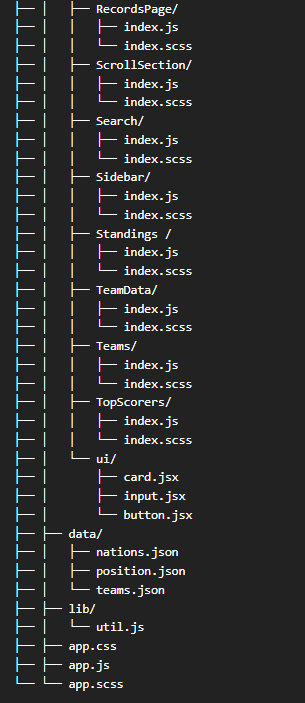
****

**USECASE DIAGRAM**



**DATA DICTIONARY**

|  |  |  |  |
| --- | --- | --- | --- |
| **SR.NO** | **Field Name** | **Data Type** | **Description** |
| **1.** | id | UUID | Unique identifier for each user. |
| **2.** | email | VARCHAR (100) | User's registered email. |
| **3.** | username | VARCHAR (50) | User's chosen username. |
| **4.** | passwordHash | VARCHAR(255) | Hashed password for security. |
| **5.** | created\_at | TIMESTAMP | Timestamp when user registered.. |
| **6.** | |  | | --- | | Updated\_at |  |  | | --- | |  | | TIMESTAMP | Timestamp when user Updates |
| **7.** | match\_id | UUID | Unique match identifier.. |
| **8.** | home\_team | UUID | Home team ID. |
| **9.** | away\_team | UUID | Away team ID. |
| **10.** | match\_date | TIMESTAMP | Date and time of the match. |
| **11.** | content | TEXT | Comment text. |

**CODE IMPLEMENTATION**

.

**PremierZoneApplication.java**

package com.pl.premier\_zone;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cache.annotation.EnableCaching;  
import org.springframework.scheduling.annotation.EnableScheduling;  
@SpringBootApplication  
@EnableCaching  
@EnableScheduling  
public class PremierZoneApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(PremierZoneApplication.class, args);}}

**WebConfig.java**

package com.pl.premier\_zone;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.client.RestTemplate;  
import org.springframework.web.servlet.config.annotation.CorsRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
@Configuration  
public class WebConfig implements WebMvcConfigurer {  
 @Bean  
 public RestTemplate restTemplate() {  
 return new RestTemplate();}  
 @Override  
 public void addCorsMappings(CorsRegistry registry) {  
 registry.addMapping("/\*\*")  
 .allowedOrigins("http://localhost:3000") // Allow frontend origin  
 .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS");}}

**FootballApiController.java**

package com.pl.premier\_zone.widgets;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.client.RestTemplate;  
import org.springframework.http.HttpEntity;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.HttpMethod;  
import org.springframework.http.ResponseEntity;  
@RestController  
@RequestMapping("/api/football")  
@CrossOrigin(origins = "http://localhost:3000")  
public class FootballApiController {  
 @Value("${api.football.key}")  
 private String apiKey;  
 private String baseUrl = "https://v3.football.api-sports.io";  
 private final RestTemplate restTemplate = new RestTemplate();  
 private HttpHeaders createHeaders() {  
 HttpHeaders headers = new HttpHeaders();  
 headers.set("x-rapidapi-key", apiKey);  
 headers.set("x-rapidapi-host", "v3.football.api-sports.io");  
 return headers;}  
 @GetMapping("/matches")  
 public ResponseEntity<String> getMatches() {  
 HttpEntity<String> entity = new HttpEntity<>(createHeaders());  
 return restTemplate.exchange(  
 baseUrl + "/fixtures?league=39&season=2024",  
 HttpMethod.*GET*,  
 entity,  
 String.class); }  
 @GetMapping("/champions")  
 public ResponseEntity<String> getChampions() {  
 HttpEntity<String> entity = new HttpEntity<>(createHeaders());  
 return restTemplate.exchange(  
 baseUrl + "/leagues/seasons?id=39",  
 HttpMethod.*GET*,  
 entity,  
 String.class);}}

**AuthResponseDto.java**

package com.pl.premier\_zone.user.dto;  
public class AuthResponseDto {  
 private String token;  
 private String username;  
 public AuthResponseDto(String token, String username) {  
 this.token = token;  
 this.username = username;}  
 public String getToken() {  
 return token;}  
 public void setToken(String token) {  
 this.token = token;}  
 public String getUsername() {  
 return username; }  
 public void setUsername(String username) {  
 this.username = username; }}

**UserLoginDto.java**

package com.pl.premier\_zone.user.dto;  
public class UserLoginDto {  
 private String username;  
 private String password;  
 public String getUsername() {  
 return username; }  
 public void setUsername(String username) {  
 this.username = username; }

public String getPassword() {  
 return password; }  
 public void setPassword(String password) {  
 this.password = password; }}

package com.pl.premier\_zone.user.dto;

**UserRegistrationDto.java**  
public class UserRegistrationDto {  
 private String username;  
 private String email;  
 private String password;  
 public String getUsername() {  
 return username; }  
 public void setUsername(String username) {  
 this.username = username; }  
 public String getEmail() {  
 return email; }  
 public void setEmail(String email) {  
 this.email = email; }  
 public String getPassword() {  
 return password; }  
 public void setPassword(String password) {  
 this.password = password; }}

**AuthController.java**

package com.pl.premier\_zone.user;  
import com.pl.premier\_zone.user.dto.UserRegistrationDto;  
import com.pl.premier\_zone.user.dto.UserLoginDto;  
import com.pl.premier\_zone.user.dto.AuthResponseDto;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
@RestController  
@RequestMapping("/api/auth")  
@CrossOrigin(origins = "http://localhost:3000") // Adjust this to match your React frontend   
public class AuthController {  
 private final UserService userService;  
 public AuthController(UserService userService) {  
 this.userService = userService; }  
 @PostMapping("/register")  
 public ResponseEntity<AuthResponseDto> register(@RequestBody UserRegistrationDto registrationDto) { try {  
 AuthResponseDto response = userService.registerUser(registrationDto);  
 return ResponseEntity.*ok*(response);  
 } catch (IllegalArgumentException e) {  
 return ResponseEntity.*badRequest*().build(); } }  
 @PostMapping("/login")  
 public ResponseEntity<AuthResponseDto> login(@RequestBody UserLoginDto loginDto) { try {  
 AuthResponseDto response = userService.loginUser(loginDto);  
 return ResponseEntity.*ok*(response);  
 } catch (Exception e) { return ResponseEntity.*badRequest*().build(); } }}

**User.java**

package com.pl.premier\_zone.user;  
import jakarta.persistence.\*;  
import java.time.LocalDateTime;  
@Entity  
@Table(name = "users")  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Integer id;  
 @Column(unique = true, nullable = false)  
 private String username;  
 @Column(unique = true, nullable = false)  
 private String email;  
 @Column(name = "password\_hash", nullable = false)  
 private String passwordHash;  
 @Column(name = "created\_at")  
 private LocalDateTime createdAt;  
 @Column(name = "updated\_at")  
 private LocalDateTime updatedAt;  
 public Integer getId() {  
 return id; }  
 public void setId(Integer id) {  
 this.id = id; }  
 public String getUsername() {  
 return username; }  
 public void setUsername(String username) {  
 this.username = username; }  
 public String getEmail() {  
 return email; }  
 public void setEmail(String email) {  
 this.email = email; }  
 public String getPasswordHash() {  
 return passwordHash; }  
 public void setPasswordHash(String passwordHash) {  
 this.passwordHash = passwordHash; }  
 public LocalDateTime getCreatedAt() {  
 return createdAt; }  
 public void setCreatedAt(LocalDateTime createdAt) {  
 this.createdAt = createdAt; }  
 public LocalDateTime getUpdatedAt() {  
 return updatedAt; }  
 public void setUpdatedAt(LocalDateTime updatedAt) {  
 this.updatedAt = updatedAt; }  
 @PrePersist  
 protected void onCreate() {  
 createdAt = LocalDateTime.*now*();  
 updatedAt = LocalDateTime.*now*(); }  
 @PreUpdate  
 protected void onUpdate() {  
 updatedAt = LocalDateTime.*now*(); }}

**UserService.java**

package com.pl.premier\_zone.user;  
import com.pl.premier\_zone.security.JwtService;  
import com.pl.premier\_zone.service.EmailService;  
import com.pl.premier\_zone.user.dto.UserRegistrationDto;  
import com.pl.premier\_zone.user.dto.UserLoginDto;  
import com.pl.premier\_zone.user.dto.AuthResponseDto;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.stereotype.Service;  
import org.springframework.security.authentication.BadCredentialsException;  
@Service  
public class UserService {  
 private final UserRepository userRepository;  
 private final PasswordEncoder passwordEncoder;  
 private final JwtService jwtService;  
 private final EmailService emailService;  
 public UserService(UserRepository userRepository,  
 PasswordEncoder passwordEncoder,  
 JwtService jwtService,  
 EmailService emailService) {  
 this.userRepository = userRepository;  
 this.passwordEncoder = passwordEncoder;  
 this.jwtService = jwtService;  
 this.emailService = emailService;  
 }  
 public AuthResponseDto registerUser(UserRegistrationDto registrationDto) {  
 if (userRepository.existsByUsername(registrationDto.getUsername())) {  
 throw new IllegalArgumentException("Username already exists"); }  
 if (userRepository.existsByEmail(registrationDto.getEmail())) {  
 throw new IllegalArgumentException("Email already exists"); }  
 User user = new User();  
 user.setUsername(registrationDto.getUsername());  
 user.setEmail(registrationDto.getEmail());  
 user.setPasswordHash(passwordEncoder.encode(registrationDto.getPassword()));  
  
 userRepository.save(user);  
 emailService.sendWelcomeEmail(user.getEmail(), user.getUsername());  
 String token = jwtService.generateToken(user.getUsername());  
 return new AuthResponseDto(token, user.getUsername()); }  
 public AuthResponseDto loginUser(UserLoginDto loginDto) {  
 User user = userRepository.findByUsername(loginDto.getUsername());  
 if (user == null || !passwordEncoder.matches(loginDto.getPassword(), user.getPasswordHash())) {  
 throw new BadCredentialsException("Invalid username or password"); }  
 String token = jwtService.generateToken(user.getUsername());  
 return new AuthResponseDto(token, user.getUsername()); }}

**TeamController.java**

package com.pl.premier\_zone.team;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.http.HttpEntity;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.HttpMethod;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.client.RestTemplate;  
@RestController  
@RequestMapping("/api/v4")  
@CrossOrigin(origins = "http://localhost:3000")  
public class TeamController {  
 private final RestTemplate restTemplate = new RestTemplate();  
 @Value("${football-data.api-key}")  
 private String apiKey;  
 public ResponseEntity<?> getPremierLeagueTeams() {  
 String url = "http://api.football-data.org/v4/competitions/PL/teams";  
 try {  
 HttpHeaders headers = new HttpHeaders();  
 headers.set("X-Auth-Token", apiKey);  
 HttpEntity<String> entity = new HttpEntity<>("parameters", headers);  
 ResponseEntity<Object> response = restTemplate.exchange(  
 url,  
 HttpMethod.*GET*,  
 entity,  
 Object.class );  
 return ResponseEntity.*ok*(response.getBody());  
 } catch (Exception e) {  
 return ResponseEntity.*internalServerError*()  
 .body("Error fetching teams: " + e.getMessage()); } }}

**StandingsController.java**

package com.pl.premier\_zone.standings;  
  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.http.ResponseEntity;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.HttpEntity;  
import org.springframework.web.client.RestTemplate;  
import org.springframework.cache.annotation.Cacheable;  
@RestController  
@RequestMapping("/api/v4")  
@CrossOrigin(origins = "http://localhost:3000")  
public class StandingsController {  
 @Value("${football-data.api-key}")  
 private String apiKey;  
 private final String PREMIER\_LEAGUE\_ID = "PL"; // Premier League competition ID  
 private final RestTemplate restTemplate = new RestTemplate();  
 @GetMapping("/standings")  
 @Cacheable("standings")  
 public ResponseEntity<?> getStandings() {  
 String url = "http://api.football-data.org/v4/competitions/" + PREMIER\_LEAGUE\_ID + "/standings";  
 try {  
 HttpHeaders headers = new HttpHeaders();  
 headers.set("X-Auth-Token", apiKey);  
 HttpEntity<String> entity = new HttpEntity<>("parameters", headers);  
 ResponseEntity<Object> response = restTemplate.exchange(  
 url,  
 org.springframework.http.HttpMethod.*GET*,  
 entity,  
 Object.class );  
 return ResponseEntity.*ok*(response.getBody());  
 } catch (Exception e) {  
 return ResponseEntity.*internalServerError*()  
 .body("Error fetching standings: " + e.getMessage()); } }}

**EmailService.java**

package com.pl.premier\_zone.service;  
import jakarta.mail.internet.MimeMessage;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.core.io.ClassPathResource;  
import org.springframework.mail.javamail.JavaMailSender;  
import org.springframework.mail.javamail.MimeMessageHelper;  
import org.springframework.stereotype.Service;  
@Service  
public class EmailService {  
 @Autowired  
 private JavaMailSender mailSender;  
  
 public void sendWelcomeEmail(String to, String username) {  
 try {  
 MimeMessage mimeMessage = mailSender.createMimeMessage();  
 MimeMessageHelper helper = new MimeMessageHelper(mimeMessage, true, "ut8");  
 helper.setTo(to);  
 helper.setSubject("Welcome to Premier Zone!");  
 String htmlContent = createWelcomeEmailHtml(username);  
 helper.setText(htmlContent, true); // true indicates this is HTML  
 ClassPathResource logoResource = new ClassPathResource("static/images/img.png");  
 helper.addInline("logo", logoResource);

mailSender.send(mimeMessage);  
 } catch (jakarta.mail.MessagingException e) {  
 throw new RuntimeException(e); } }  
 private String createWelcomeEmailHtml(String username) {  
 return "<!DOCTYPE html>\n" +  
 "<html>\n" +  
 "<head>\n" +  
 " <meta charset=\"UTF-8\">\n" +  
 " <meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">\n" +  
 " <title>Welcome to Premier Zone</title>\n" +  
 "</head>\n" +  
 "<body style=\"margin: 0; padding: 0; font-family: Arial, sans-serif; background-color: #ade8f4;\">\n" +  
 " <table border=\"0\" cellpadding=\"0\" cellspacing=\"0\" width=\"100%\">\n" +  
 " <tr>\n" +  
 " <td style=\"padding: 20px 0;\">\n" +  
 " <table align=\"center\" border=\"0\" cellpadding=\"0\" cellspacing=\"0\" width=\"600\" style=\"border-collapse: collapse; background-color: #ffffff; border-radius: 12px; box-shadow: 0 4px 6px rgba(0,0,0,0.1);\">\n" +  
 " <tr>\n" +  
 " <td align=\"center\" style=\"padding: 40px 0; background: linear-gradient(to right, #0077b6, #00b4d8); border-top-left-radius: 12px; border-top-right-radius: 12px;\">\n" +  
 " <img src=\"cid:logo\" alt=\"Premier Zone Logo\" width=\"120\" style=\"display: block;\">\n" +  
 " <h1 style=\"color: white; margin-top: 15px; font-size: 28px;\">Premier Zone</h1>\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " <tr>\n" +  
 " <td style=\"padding: 40px 30px;\">\n" +  
 " <table border=\"0\" cellpadding=\"0\" cellspacing=\"0\" width=\"100%\">\n" +  
 " <tr>\n" +  
 " <td style=\"color: #1a202c; font-size: 24px; font-weight: bold;\">\n" +  
 " Welcome to Premier Zone, " + username + "!\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " <tr>\n" +  
 " <td style=\"padding: 20px 0; color: #4a5568; font-size: 16px; line-height: 24px;\">\n" +  
 " We're thrilled to have you join our community of Premier League fans and fantasy football enthusiasts. With Premier Zone, you'll get access to comprehensive statistics, player insights, and fantasy football analytics.\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " <tr>\n" +  
 " <td style=\"padding: 10px 0; color: #4a5568; font-size: 16px; line-height: 24px;\">\n" +  
 " What you can do with Premier Zone:\n" +  
 " <ul style=\"padding-left: 20px;\">\n" +  
 " <li>Track player and team statistics</li>\n" +  
 " <li>Get data-driven insights for your fantasy team</li>\n" +  
 " <li>Stay updated with the latest Premier League news</li>\n" +  
 " <li>Connect with other football fans</li>\n" +  
 " </ul>\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " <tr>\n" +  
 " <td style=\"padding: 20px 0;\">\n" +  
 " <table border=\"0\" cellpadding=\"0\" cellspacing=\"0\" width=\"100%\">\n" +  
 " <tr>\n" +  
 " <td align=\"center\">\n" +  
 " <a href=\"http://localhost:3000/dashboard\" style=\"background: linear-gradient(to right, #0077b6, #00b4d8); border-radius: 6px; color: #ffffff; display: inline-block; font-size: 16px; font-weight: bold; padding: 12px 24px; text-decoration: none; transition: transform 0.3s ease;\">GET STARTED</a>\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " </table>\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " </table>\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " <tr>\n" +  
 " <td style=\"padding: 30px; background: linear-gradient(to right, #0077b6, #00b4d8); border-bottom-left-radius: 12px; border-bottom-right-radius: 12px;\">\n" +  
 " <table border=\"0\" cellpadding=\"0\" cellspacing=\"0\" width=\"100%\">\n" +  
 " <tr>\n" +  
 " <td style=\"color: #ffffff; font-size: 14px; text-align: center;\">\n" +  
 " <p style=\"margin: 0;\">© 2024 Premier Zone. All rights reserved.</p>\n" +  
 " <p style=\"margin: 10px 0 0 0;\">Your home for everything Premier League related!</p>\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " </table>\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " </table>\n" +  
 " </td>\n" +  
 " </tr>\n" +  
 " </table>\n" +  
 "</body>\n" +  
 "</html>"; }}

**JwtAuthenticationFilter.java**

package com.pl.premier\_zone.security;  
import jakarta.servlet.FilterChain;  
import jakarta.servlet.ServletException;  
import jakarta.servlet.http.HttpServletRequest;  
import jakarta.servlet.http.HttpServletResponse;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;  
import org.springframework.stereotype.Component;  
import org.springframework.web.filter.OncePerRequestFilter;  
import java.io.IOException;  
import java.util.ArrayList;  
@Component  
public class JwtAuthenticationFilter extends OncePerRequestFilter {  
 private final JwtService jwtService;  
 public JwtAuthenticationFilter(JwtService jwtService) {  
 this.jwtService = jwtService; }  
 @Override  
 protected void doFilterInternal(  
 HttpServletRequest request,  
 HttpServletResponse response,  
 FilterChain filterChain  
 ) throws ServletException, IOException {  
 final String authHeader = request.getHeader("Authorization");  
 if (authHeader == null || !authHeader.startsWith("Bearer ")) {  
 filterChain.doFilter(request, response);  
 return; }  
 final String jwt = authHeader.substring(7);  
 final String username = jwtService.validateTokenAndGetUsername(jwt);  
 if (username != null && SecurityContextHolder.*getContext*().getAuthentication() == null) {  
 UsernamePasswordAuthenticationToken authToken = new UsernamePasswordAuthenticationToken(  
 username,  
 null,  
 new ArrayList<>() // empty authorities );  
 authToken.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));  
 SecurityContextHolder.*getContext*().setAuthentication(authToken); }  
 filterChain.doFilter(request, response); }}

**JwtService.java**

package com.pl.premier\_zone.security;  
import java.util.Date;  
import java.util.Base64;  
import io.jsonwebtoken.Jwts;  
import io.jsonwebtoken.SignatureAlgorithm;  
import io.jsonwebtoken.Claims;  
import io.jsonwebtoken.JwtException;  
import org.springframework.stereotype.Service;  
@Service  
public class JwtService {  
 private final String SECRET\_KEY = "yourSecretKeyHereMakeItLongAndSecureAtLeast32Chars";  
 private static final long *EXPIRATION\_TIME* = 864\_000\_000; // 10 days  
 private byte[] getSigningKey() {  
 return Base64.*getEncoder*().encode(SECRET\_KEY.getBytes()); }  
 public String generateToken(String username) {  
 Date now = new Date();  
 Date expiryDate = new Date(now.getTime() + *EXPIRATION\_TIME*);  
 return Jwts.*builder*()  
 .setSubject(username)  
 .setIssuedAt(now)  
 .setExpiration(expiryDate)  
 .signWith(SignatureAlgorithm.*HS256*, getSigningKey())  
 .compact(); }  
 public String validateTokenAndGetUsername(String token) {  
 try { Claims claims = Jwts.*parser*()  
 .setSigningKey(getSigningKey())  
 .parseClaimsJws(token)  
 .getBody();  
 return claims.getSubject();  
 } catch (JwtException e) {  
 return null; } }}

**SecurityConfig.java**

package com.pl.premier\_zone.security;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;  
import org.springframework.security.crypto.password.PasswordEncoder;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.web.SecurityFilterChain;  
import org.springframework.security.config.http.SessionCreationPolicy;  
importorg.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;  
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
 private final JwtAuthenticationFilter jwtAuthFilter;  
 public SecurityConfig(JwtAuthenticationFilter jwtAuthFilter) {  
 this.jwtAuthFilter = jwtAuthFilter; }  
 @Bean  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
 .securityMatcher("/api/\*\*")  
 .csrf(csrf -> csrf.disable())  
 .cors(cors -> cors.configure(http))  
 .sessionManagement(session ->  
 session.sessionCreationPolicy(SessionCreationPolicy.*STATELESS*))  
 .authorizeHttpRequests(auth -> auth  
 .requestMatchers(new AntPathRequestMatcher("/api/auth/\*\*")).permitAll()  
 .requestMatchers(new AntPathRequestMatcher("/api/v4/matches/\*\*")).permitAll()  
 .requestMatchers(new AntPathRequestMatcher("/api/v4/teams/\*\*")).permitAll()  
 .requestMatchers(new AntPathRequestMatcher("/api/v1/news/\*\*")).permitAll()  
 .requestMatchers(new AntPathRequestMatcher("/api/v4/standings/\*\*")).permitAll()  
 .requestMatchers(new AntPathRequestMatcher("/api/v4/scorers/\*\*")).permitAll()  
 .requestMatchers(new AntPathRequestMatcher("/api/v4/comparison/{team1Id}/{team2Id}/\*\*")).permitAll()  
 .requestMatchers(new AntPathRequestMatcher("/api/v1/player/\*\*")).permitAll()  
 .requestMatchers(new AntPathRequestMatcher("/api/v4/matches/\*/comments", "GET")).permitAll()  
 .anyRequest().authenticated() )  
 .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);  
 return http.build(); }  
 @Bean  
 public PasswordEncoder passwordEncoder() {  
 return new BCryptPasswordEncoder(); }}

**TopScorersController.java**

package com.pl.premier\_zone.scorers;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.http.ResponseEntity;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.HttpEntity;  
import org.springframework.web.client.RestTemplate;  
import org.springframework.cache.annotation.Cacheable;  
@RestController  
@RequestMapping("/api/v4")  
@CrossOrigin(origins = "http://localhost:3000")  
public class TopScorersController {  
  
 @Value("${football-data.api-key}")  
 private String apiKey;  
 private final String PREMIER\_LEAGUE\_ID = "PL";  
 private final RestTemplate restTemplate = new RestTemplate();  
 @GetMapping("/scorers")  
 @Cacheable("scorers")  
 public ResponseEntity<?> getTopScorers() {  
 String url = "http://api.football-data.org/v4/competitions/" + PREMIER\_LEAGUE\_ID + "/scorers";  
 try {  
 HttpHeaders headers = new HttpHeaders();  
 headers.set("X-Auth-Token", apiKey);  
 HttpEntity<String> entity = new HttpEntity<>("parameters", headers);  
 ResponseEntity<Object> response = restTemplate.exchange(  
 url,  
 org.springframework.http.HttpMethod.*GET*,  
 entity,  
 Object.class );  
 return ResponseEntity.*ok*(response.getBody());  
 } catch (Exception e) {  
 return ResponseEntity.*internalServerError*()  
 .body("Error fetching top scorers: " + e.getMessage()); } }}

**NewsController.java**

package com.pl.premier\_zone.news;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.web.bind.annotation.CrossOrigin;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.client.RestTemplate;  
import org.springframework.http.ResponseEntity;  
import org.springframework.cache.annotation.Cacheable;  
@RestController  
@RequestMapping("/api/v1")  
@CrossOrigin(origins = "http://localhost:3000")  
public class NewsController {  
 @Value("${newsapi.key}")  
 private String apiKey;  
 private final RestTemplate restTemplate = new RestTemplate();  
 @GetMapping("/news")  
 @Cacheable("footballNews")  
 public ResponseEntity<?> getFootballNews() {  
 String url = "https://newsapi.org/v2/everything?" +  
 "q=(premier+league+OR+EPL)+AND+(transfer+OR+injury+OR+news)" +  
 "&language=en" +  
 "&sortBy=publishedAt" +  
 "&apiKey=" + apiKey;  
 try { return ResponseEntity.*ok*(  
 restTemplate.getForObject(url, Object.class) );  
 } catch (Exception e) {  
 return ResponseEntity.*internalServerError*()  
 .body("Error fetching news: " + e.getMessage()); } }  
}

**HeadToHeadController.java**

package com.pl.premier\_zone.comparison;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.http.ResponseEntity;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.HttpEntity;  
import org.springframework.web.client.RestTemplate;  
import org.springframework.cache.annotation.Cacheable;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import java.util.Map;  
@RestController  
@RequestMapping("/api/v4")  
@CrossOrigin(origins = "http://localhost:3000") // Add this if your React app runs on port 3000  
public class HeadToHeadController {  
 private static final Logger *logger* = LoggerFactory.*getLogger*(HeadToHeadController.class);  
 @Value("${football-data.api-key}")  
 private String apiKey;  
 private final String PREMIER\_LEAGUE\_ID = "PL";  
 private final RestTemplate restTemplate = new RestTemplate();  
 @GetMapping("/teams")  
 @Cacheable("teams")  
 public ResponseEntity<?> getTeams() {  
 String url = "http://api.football-data.org/v4/competitions/" + PREMIER\_LEAGUE\_ID + "/teams";  
 try { HttpHeaders headers = new HttpHeaders();  
 headers.set("X-Auth-Token", apiKey);  
 HttpEntity<String> entity = new HttpEntity<>("parameters", headers);  
 ResponseEntity<Object> response = restTemplate.exchange(  
 url,  
 org.springframework.http.HttpMethod.*GET*,  
 entity,  
 Object.class );  
 return ResponseEntity.*ok*(response.getBody());  
 } catch (Exception e) { return ResponseEntity.*internalServerError*()  
 .body("Error fetching teams: " + e.getMessage()); } }  
 @GetMapping("/comparison/{team1Id}/{team2Id}")  
 @Cacheable("comparison")  
 public ResponseEntity<?> getComparison(  
 @PathVariable String team1Id,  
 @PathVariable String team2Id ) {  
 String url = String.*format*(  
 "http://api.football-data.org/v4/teams/%s/matches?limit=10&status=FINISHED",  
 team1Id );  
 try { HttpHeaders headers = new HttpHeaders();  
 headers.set("X-Auth-Token", apiKey);  
 HttpEntity<String> entity = new HttpEntity<>("parameters", headers);  
 ResponseEntity<Object> response = restTemplate.exchange(  
 url,  
 org.springframework.http.HttpMethod.*GET*,  
 entity,  
 Object.class );  
 String url2 = String.*format*(  
 "http://api.football-data.org/v4/teams/%s/matches?limit=10&status=FINISHED",  
 team2Id );  
 ResponseEntity<Object> response2 = restTemplate.exchange(  
 url2,  
 org.springframework.http.HttpMethod.*GET*,  
 entity,  
 Object.class );  
 Map<String, Object> result = Map.*of*(  
 "team1Matches", response.getBody(),  
 "team2Matches", response2.getBody() );  
 return ResponseEntity.*ok*(result);  
 } catch (Exception e) {return ResponseEntity.*internalServerError*()  
 .body("Error fetching comparison data: " + e.getMessage()); } }}

**Player.java**

package com.pl.premier\_zone.player;  
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
import java.io.Serializable;  
@Entity  
@Table(name = "player\_statistic")  
public class Player {  
 @Id@Column(name = "player\_name", unique = true)  
 private String name;  
 private String nation;  
 private String position;  
 private Integer age;  
 private Integer matches\_played;  
 private Integer starts;  
 private Double minutes\_played;  
 private Double goals;  
 private Double assists;  
 private Double penalties\_scored;  
 private Double yellow\_cards;  
 private Double red\_cards;  
 private Double expected\_goals;  
 private Double expected\_assists;  
 private String team\_name;  
 public Player(String name) {  
 this.name = name; }  
 public Player(String name, String nation, String position, Integer age, Integer matches\_played, Integer starts, Double minutes\_played, Double goals, Double assists, Double penalties\_scored, Double yellow\_cards, Double red\_cards, Double expected\_goals, Double expected\_assists, String team\_name) {  
 this.name = name;  
 this.nation = nation;  
 this.position = position;  
 this.age = age;  
 this.matches\_played = matches\_played;  
 this.starts = starts;  
 this.minutes\_played = minutes\_played;  
 this.goals = goals;  
 this.assists = assists;  
 this.penalties\_scored = penalties\_scored;  
 this.yellow\_cards = yellow\_cards;  
 this.red\_cards = red\_cards;  
 this.expected\_goals = expected\_goals;  
 this.expected\_assists = expected\_assists;  
 this.team\_name = team\_name; }  
 public Player() { }  
 public String getName() {  
 return name; }  
 public void setName(String name) {  
 this.name = name; }  
 public String getNation() {  
 return nation != null ? nation : "unknown"; }  
 public void setNation(String nation) {  
 this.nation = nation; }  
 public String getPosition() {  
 return position != null ? position : "unknown"; }  
 public void setPosition(String position) {  
 this.position = position; }  
 public Serializable getAge() {  
 return age != null ? age : "unknown"; }  
 public void setAge(Integer age) {  
 this.age = age; }  
 public Serializable getMatches\_played() {  
 return matches\_played != null ? matches\_played : "unknown"; }  
 public void setMatches\_played(Integer matches\_played) {  
 this.matches\_played = matches\_played; }  
 public Serializable getStarts() {  
 return starts != null ? starts : "unknown"; }  
 public void setStarts(Integer starts) {  
 this.starts = starts; }  
 public Serializable getMinutes\_played() {  
 return minutes\_played != null ? minutes\_played : "unknown"; }  
 public void setMinutes\_played(Double minutes\_played) {  
 this.minutes\_played = minutes\_played; }  
 public Serializable getGoals() {  
 return goals!= null ? goals : "unknown"; }  
 public void setGoals(Double goals) {  
 this.goals = goals; }  
 public Serializable getAssists() {  
 return assists!= null ? assists : "unknown"; }  
 public void setAssists(Double assists) {  
 this.assists = assists; }  
 public Serializable getPenalties\_scored() {  
 return penalties\_scored != null ? penalties\_scored : "unknown"; }  
 public void setPenalties\_scored(Double penalties\_scored) {  
 this.penalties\_scored = penalties\_scored; }  
 public Serializable getYellow\_cards() {  
 return yellow\_cards != null ? yellow\_cards : "unknown"; }  
 public void setYellow\_cards(Double yellow\_cards) {  
 this.yellow\_cards = yellow\_cards; }  
 public Serializable getRed\_cards() {  
 return red\_cards != null ? red\_cards : "unknown"; }  
 public void setRed\_cards(Double red\_cards) { this.red\_cards = red\_cards; }

public Serializable expected\_goals() {  
 return expected\_goals != null ? expected\_goals : "unknown"; }  
 public void setExpected\_goals(Double expected\_goals) {  
 this.expected\_goals = expected\_goals; }  
 public Double getExpected\_assists() {  
 return expected\_assists; }  
 public void setExpected\_assists(Double expected\_assists) {  
 this.expected\_assists = expected\_assists; }  
 public String getTeam\_name() {  
 return team\_name; }  
 public void setTeam\_name(String team\_name) {  
 this.team\_name = team\_name; }}

**PlayerController.java**

package com.pl.premier\_zone.player;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import java.util.List;  
@RestController@RequestMapping(path = "api/v1/player")  
public class PlayerController {  
 private final PlayerService playerService;  
 @Autowired  
 public PlayerController(PlayerService playerService) {  
 this.playerService = playerService; }  
 @GetMapping  
 public List<Player> getPlayers(  
 @RequestParam(required = false) String team\_name,  
 @RequestParam(required = false) String name,  
 @RequestParam(required = false) String position,  
 @RequestParam(required = false) String nation) {  
 if (team\_name != null && position != null) {  
 return playerService.getPlayersByTeamAndPosition(team\_name, position); } else if (team\_name != null) {  
 return playerService.getPlayersFromTeam(team\_name); } else if (name != null) {  
 return playerService.getPlayersByName(name); } else if (position != null) {  
 return playerService.getPlayersByPos(position); } else if (nation != null) {  
 return playerService.getPlayersByNation(nation); } else {  
 return playerService.getPlayers(); } }  
 @PostMapping  
 public ResponseEntity<Player> addPlayer(@RequestBody Player player){  
 Player createdPlayer = playerService.addPlayer(player);  
 return new ResponseEntity<>(createdPlayer, HttpStatus.*CREATED*); }  
 @PutMapping  
 public ResponseEntity<Player> updatePlayer(@RequestBody Player updatedPlayer) {  
 Player resultPlayer = playerService.updatePlayer(updatedPlayer);  
 if (resultPlayer != null) {  
 return new ResponseEntity<>(resultPlayer, HttpStatus.*OK*); } else {  
 return new ResponseEntity<>(HttpStatus.*NOT\_FOUND*); } }  
 @DeleteMapping("/{playerName}")  
 public ResponseEntity<String> deletePlayer(@PathVariable String playerName) {  
 playerService.deletePlayer(playerName);  
 return new ResponseEntity<>("Player deleted successfully", HttpStatus.*OK*); }

**PlayerService.java**

package com.pl.premier\_zone.player;  
import jakarta.transaction.Transactional;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Component;  
import java.util.List;  
import java.util.Optional;  
import java.util.stream.Collectors;  
@Component  
public class PlayerService {  
 private final PlayerRespository playerRespository;  
 @Autowired  
 public PlayerService(PlayerRespository playerRespository){  
 this.playerRespository = playerRespository; }  
 public List<Player> getPlayers() {  
 return playerRespository.findAll(); }  
 public List<Player> getPlayersFromTeam(String teamName) {  
 return playerRespository.findAll().stream()  
 .filter(player -> teamName.equals(player.getTeam\_name()))  
 .collect(Collectors.*toList*()); }  
 public List<Player> getPlayersByName(String searchText) {  
 return playerRespository.findAll().stream()  
 .filter(player -> player.getName().toLowerCase().contains(searchText.toLowerCase()))  
 .collect(Collectors.*toList*()); }  
 public List<Player> getPlayersByPos(String searchText) {  
 return playerRespository.findAll().stream()  
 .filter(player -> player.getPosition().toLowerCase().contains(searchText.toLowerCase()))  
 .collect(Collectors.*toList*()); }  
 public List<Player> getPlayersByNation(String searchText) {  
 return playerRespository.findAll().stream()  
 .filter(player -> player.getNation().toLowerCase().contains(searchText.toLowerCase()))  
 .collect(Collectors.*toList*()); }  
 public List<Player> getPlayersByTeamAndPosition(String team, String position){  
 return playerRespository.findAll().stream()  
 .filter(player -> team.equals(player.getTeam\_name()) && position.equals(player.getPosition()))  
 .collect(Collectors.*toList*()); }  
 public Player addPlayer(Player player) {  
 playerRespository.save(player);  
 return player; }  
 public Player updatePlayer(Player updatedPlayer) {  
 Optional<Player> existingPlayer = playerRespository.findByName(updatedPlayer.getName());  
  
 if (existingPlayer.isPresent()) {  
 Player playerToUpdate = existingPlayer.get();  
 playerToUpdate.setName(updatedPlayer.getName());  
 playerToUpdate.setTeam\_name(updatedPlayer.getTeam\_name());  
 playerToUpdate.setPosition(updatedPlayer.getPosition());  
 playerToUpdate.setNation(updatedPlayer.getNation());  
 playerRespository.save(playerToUpdate);  
 return playerToUpdate; }  
 return null; }  
 @Transactional  
 public void deletePlayer(String playerName) {  
 playerRespository.deleteByName(playerName); }}

**FootballDataMatch.java**

package com.pl.premier\_zone.match;  
import com.fasterxml.jackson.annotation.JsonProperty;  
import java.time.ZonedDateTime;  
import java.util.List;  
public class FootballDataMatch {  
 private Long id;  
 private Team homeTeam;  
 private Team awayTeam;  
 private Score score;  
 @JsonProperty("utcDate")  
 private ZonedDateTime utcDate;  
 private String status;  
 private List<Referee> referees;  
 private Integer matchday;   
 public Integer getMatchday() {  
 return matchday; }  
 public void setMatchday(Integer matchday) {this.matchday = matchday; }  
 public List<Referee> getReferees() { return referees; }  
 public void setReferees(List<Referee> referees) { this.referees = referees; }  
 public Long getId() { return id; }  
 public void setId(Long id) { this.id = id; }  
 public Team getHomeTeam() { return homeTeam; }  
 public void setHomeTeam(Team homeTeam) { this.homeTeam = homeTeam; }  
 public Team getAwayTeam() { return awayTeam; }  
 public void setAwayTeam(Team awayTeam) { this.awayTeam = awayTeam; }  
 public Score getScore() { return score; }  
 public void setScore(Score score) { this.score = score; }  
 public ZonedDateTime getUtcDate() { return utcDate; }  
 public void setUtcDate(ZonedDateTime utcDate) { this.utcDate = utcDate; }  
 public String getStatus() { return status; }  
 public void setStatus(String status) { this.status = status; }   
 public static class Referee { public Long getId() { return id; }  
 public void setId(Long id) { this.id = id; }  
 public String getName() { return name; }  
 public void setName(String name) { this.name = name; }  
 public String getType() { return type; }  
 public void setType(String type) { this.type = type; }  
 public String getNationality() { return nationality; }  
 public void setNationality(String nationality) { this.nationality = nationality; }  
 private Long id;

private String name;  
 private String type;  
 private String nationality; }  
 public static class Team {  
 private String name;  
 @JsonProperty("crest")  
 private String crest;  
 public String getName() ; return name; }  
 public void setName(String name) { this.name = name; }  
 public String getCrest() { return crest; }  
 public void setCrest(String crest) { this.crest = crest; } }  
 public static class Score {  
 @JsonProperty("fullTime")  
 private FullTime fullTime;  
 @JsonProperty("halfTime")  
 private FullTime halfTime;  
 private String winner;  
 public FullTime getFullTime() { return fullTime; }  
 public void setFullTime(FullTime fullTime) { this.fullTime = fullTime; }  
 public FullTime getHalfTime() { return halfTime; }  
 public void setHalfTime(FullTime halfTime) { this.halfTime = halfTime; }  
 public String getWinner() { return winner; }  
 public void setWinner(String winner) { this.winner = winner; }  
 public static class FullTime {  
 private Integer home;  
 private Integer away;  
 public Integer getHome() { return home; }  
 public void setHome(Integer home) { this.home = home; }  
 public Integer getAway() { return away; }  
 public void setAway(Integer away) { this.away = away; } } }}

**Match.java**

package com.pl.premier\_zone.match;  
import java.time.LocalDateTime;  
import java.util.List;  
import jakarta.persistence.\*;  
@Entity@Table(name = "matches")  
public class Match { @Id  
 private Long id;  
 @Column(name = "home\_crest") private String homeCrest;  
 @Column(name = "away\_crest") private String awayCrest;  
 @Column(name = "home\_team", nullable = false) private String homeTeam;  
 @Column(name = "away\_team", nullable = false) private String awayTeam;  
 @Column(name = "home\_score") private Integer homeScore;  
 @Column(name = "away\_score") private Integer awayScore;  
 @Column(name = "match\_date", nullable = false) private LocalDateTime matchDate;  
 @Column(nullable = false) private String status;  
 @Embedded private Score score; @ElementCollection  
 @CollectionTable(name = "match\_referees", joinColumns = @JoinColumn(name = "match\_id"))  
 private List<MatchReferee> referees;  
 private Integer matchday;  
 @Column(name = "created\_at") private LocalDateTime createdAt;  
 @Column(name = "updated\_at") private LocalDateTime updatedAt;  
 public Match() {}  
 public LocalDateTime getCreatedAt() { return createdAt; }  
 public void setCreatedAt(LocalDateTime createdAt) { this.createdAt = createdAt; }  
 public LocalDateTime getUpdatedAt() { return updatedAt; }  
 public void setUpdatedAt(LocalDateTime updatedAt) { this.updatedAt = updatedAt; }  
 public Integer getMatchday() { return matchday; }  
 public void setMatchday(Integer matchday) { this.matchday = matchday; }  
 public List<MatchReferee> getReferees() { return referees; } public void setReferees(List<MatchReferee> referees) { this.referees = referees; }  
 public String getHomeCrest() { return homeCrest; }  
 public void setHomeCrest(String homeCrest) { this.homeCrest = homeCrest; }  
 public String getAwayCrest() { return awayCrest; }  
 public void setAwayCrest(String awayCrest) { this.awayCrest = awayCrest; }  
 public Long getId() { return id; }  
 public void setId(Long id) { this.id = id; }  
 public String getHomeTeam() { return homeTeam; }  
 public void setHomeTeam(String homeTeam) { this.homeTeam = homeTeam; }  
 public String getAwayTeam() { return awayTeam; }  
 public void setAwayTeam(String awayTeam) { this.awayTeam = awayTeam; }  
 public Integer getHomeScore() { return homeScore; }  
 public void setHomeScore(Integer homeScore) { this.homeScore = homeScore; }  
 public Integer getAwayScore() { return awayScore; }  
 public void setAwayScore(Integer awayScore) { this.awayScore = awayScore; }  
 public LocalDateTime getMatchDate() { return matchDate; }  
 public void setMatchDate(LocalDateTime matchDate) { this.matchDate = matchDate; }  
 public String getStatus() { return status; }  
 public void setStatus(String status) { this.status = status; }  
 public Score getScore() { return score; }  
 public void setScore(Score score) { this.score = score; }  
 @Embeddable  
 public static class MatchReferee { private String name;  
 public String getName() { return name; }  
 public void setName(String name) { this.name = name; } }  
 public static class Score {  
 private Integer homeHalfTime;  
 private Integer awayHalfTime;  
 private Integer homeFullTime;  
 private Integer awayFullTime;  
 private String winner;  
 public Integer getHomeHalfTime() { return homeHalfTime; }  
 public void setHomeHalfTime(Integer homeHalfTime) { this.homeHalfTime = homeHalfTime; }  
 public Integer getAwayHalfTime() { return awayHalfTime; } public void setAwayHalfTime(Integer awayHalfTime) { this.awayHalfTime = awayHalfTime; }  
 public Integer getHomeFullTime() { return homeFullTime; }  
 public void setHomeFullTime(Integer homeFullTime) { this.homeFullTime = homeFullTime; }  
 public Integer getAwayFullTime() {  
 return awayFullTime; }

public void setAwayFullTime(Integer awayFullTime) { this.awayFullTime = awayFullTime }  
 public String getWinner() { return winner; }  
 public void setWinner(String winner) { this.winner = winner; } }  
}

**MatchController.java**

package com.pl.premier\_zone.match;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import java.time.LocalDate;  
import java.util.\*;  
@RestController  
@RequestMapping("/api/v4/matches")  
@CrossOrigin(origins = "http://localhost:3000")  
public class MatchController {  
 private final Logger logger = LoggerFactory.*getLogger*(MatchController.class);  
 private final MatchService matchService;  
 public MatchController(MatchService matchService) { this.matchService = matchService; }  
 @GetMapping  
 public ResponseEntity<Map<String, List<Match>>> getMatches() {  
 try { Map<String, List<Match>> matches = matchService.getAllMatches();  
 return ResponseEntity.*ok*(matches);  
 } catch (Exception e) {  
 return ResponseEntity.*internalServerError*().build(); } }  
 @GetMapping("/date")  
 public ResponseEntity<List<Match>> getMatchesByDate(@RequestParam("date") String dateString) {  
 try { LocalDate specificDate = LocalDate.*parse*(dateString);  
 List<Match> matches = matchService.getMatchesByDate(specificDate);  
 return ResponseEntity.*ok*(matches);  
 } catch (Exception e) {  
 return ResponseEntity.*badRequest*().build(); } }  
 @GetMapping("/{id}/details")  
 public ResponseEntity<Map<String, Object>> getMatchDetails(@PathVariable Long id) {  
 try { Optional<Match> match = matchService.findById(id);  
 if (match.isEmpty()) {  
 return ResponseEntity.*notFound*().build(); }  
 Map<String, Object> details = new HashMap<>();  
 details.put("match", match.get());  
 details.put("streaming", getStreamingInfo(match.get()));  
 return ResponseEntity.*ok*(details);  
 } catch (Exception e) {  
 return ResponseEntity.*internalServerError*().build(); } }  
 private Map<String, String> getStreamingInfo(Match match) {  
 Map<String, String> streamingInfo = new HashMap<>();  
 streamingInfo.put("global", "https://www.premierleague.com/broadcast-schedules");  
 streamingInfo.put("india", "https://www.hotstar.com/in/sports/football/tournaments/premier-league/" + match.getHomeTeam() + match.getAwayTeam());  
 streamingInfo.put("matchUrl", "https://www.hotstar.com/sports/football/match/" + match.getHomeTeam() + match.getAwayTeam());  
 return streamingInfo; }  
 @GetMapping("/current-matchday")  
 public ResponseEntity<Integer> getCurrentMatchday() {  
 try { Integer matchday = matchService.getCurrentMatchday();  
 return ResponseEntity.*ok*(matchday);  
 } catch (Exception e) {  
 return ResponseEntity.*internalServerError*().build(); } }  
 @GetMapping("/today")  
 public ResponseEntity<List<Match>> getTodayMatches() {  
 try { Map<String, List<Match>> matches = matchService.getAllMatches();  
 return ResponseEntity.*ok*(matches.get("today"));  
 } catch (Exception e) {  
 return ResponseEntity.*internalServerError*().build(); } }  
 @GetMapping("/upcoming")  
 public ResponseEntity<List<Match>> getUpcomingMatches() {  
 try { Map<String, List<Match>> matches = matchService.getAllMatches();  
 return ResponseEntity.*ok*(matches.get("upcoming"));  
 } catch (Exception e) {  
 return ResponseEntity.*internalServerError*().build(); } }  
 @GetMapping("/yesterday")  
 public ResponseEntity<List<Match>> getYesterdayMatches() {  
 try {  
 Map<String, List<Match>> matches = matchService.getAllMatches();  
 return ResponseEntity.*ok*(matches.get("yesterday"));  
 } catch (Exception e) {   
 return ResponseEntity.*internalServerError*().build(); } }}

**MatchService.java**

package com.pl.premier\_zone.match;  
import org.springframework.beans.factory.annotation.Value;  
import org.springframework.cache.annotation.Cacheable;  
import org.springframework.http.HttpEntity;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.HttpMethod;  
import org.springframework.http.ResponseEntity;  
import org.springframework.stereotype.Service;  
import org.springframework.web.client.RestTemplate;  
import java.time.\*;  
import java.util.\*;  
import java.util.stream.Collectors;  
@Service  
public class MatchService {  
 private final RestTemplate restTemplate;  
 private final String apiKey;  
 private final String baseUrl = "http://api.football-data.org/v4";  
 private SeasonInfo currentSeason;  
 private final MatchRepository matchRepository;  
 public MatchService(  
 RestTemplate restTemplate,  
 MatchRepository matchRepository,  
 @Value("${football-data.api-key}") String apiKey) {  
 this.restTemplate = restTemplate;  
 this.matchRepository = matchRepository;  
 this.apiKey = apiKey; }  
 public Integer getCurrentMatchday() {  
 return currentSeason != null ? currentSeason.getCurrentMatchday() : null; }  
 @Cacheable(value = "matches", key = "'all'")  
 public Map<String, List<Match>> getAllMatches() {  
 List<Match> allMatches = fetchMatchesFromApi();   
 Map<String, List<Match>> categorizedMatches = new HashMap<>();  
 categorizedMatches.put("today", new ArrayList<>());  
 categorizedMatches.put("yesterday", new ArrayList<>());  
 categorizedMatches.put("upcoming", new ArrayList<>());  
 LocalDate today = LocalDate.*now*();  
 LocalDate yesterday = today.minusDays(1);  
 for (Match match : allMatches) {  
 if (match.getMatchDate() == null) continue;  
 LocalDate matchDate = match.getMatchDate().toLocalDate()   
 if (matchDate.isBefore(today) || matchDate.isEqual(yesterday)) {  
 categorizedMatches.get("yesterday").add(match);  
 } else if (matchDate.isEqual(today)) {  
 categorizedMatches.get("today").add(match);  
 } else {  
 categorizedMatches.get("upcoming").add(match); } }   
 categorizedMatches.get("yesterday").sort((m1, m2) ->  
 m2.getMatchDate().compareTo(m1.getMatchDate()));  
 categorizedMatches.get("today").sort(Comparator.*comparing*(Match::getMatchDate));  
 categorizedMatches.get("upcoming").sort(Comparator.*comparing*(Match::getMatchDate));  
 return categorizedMatches; }  
 private List<Match> fetchMatchesFromApi() {  
 HttpHeaders headers = new HttpHeaders();  
 headers.add("X-Auth-Token", apiKey);  
 HttpEntity<String> entity = new HttpEntity<>(headers);  
 String url = baseUrl + "/competitions/PL/matches?season=2024"; String seasonUrl = baseUrl + "/competitions/PL";  
 try { ResponseEntity<Map> seasonResponse = restTemplate.exchange(  
 seasonUrl,  
 HttpMethod.*GET*,  
 entity,  
 Map.class );  
 if (seasonResponse.getBody() != null) {  
 Map<String, Object> seasonData = seasonResponse.getBody();  
 currentSeason = new SeasonInfo();  
 currentSeason.setId((Integer) seasonData.get("id"));  
 currentSeason.setCurrentMatchday((Integer) seasonData.get("currentMatchday")); }  
 }catch (Exception e) { e.printStackTrace(); }   
 try { ResponseEntity<FootballDataResponse> response = restTemplate.exchange(  
 url,  
 HttpMethod.*GET*,  
 entity,  
 FootballDataResponse.class );  
 if (response.getBody() != null && response.getBody().getMatches() != null) {  
 return convertToMatchEntities(response.getBody().getMatches()); }  
 } catch (Exception e) { e.printStackTrace(); }  
 return new ArrayList<>(); }  
 public List<Match> getMatchesByDate(LocalDate specificDate) {  
 return fetchMatchesFromApi().stream()  
 .filter(match -> match.getMatchDate() != null &&  
 match.getMatchDate().toLocalDate().equals(specificDate))  
 .sorted(Comparator.*comparing*(Match::getMatchDate))  
 .collect(Collectors.*toList*()); }  
 public Optional<Match> findById(Long id) {  
 return fetchMatchesFromApi().stream()  
 .filter(match -> match.getId().equals(id))  
 .findFirst(); }  
 private static class SeasonInfo { public Integer getId() { return id; }  
 public void setId(Integer id) { this.id = id; }  
 private Integer id;  
 private LocalDate startDate;  
 private LocalDate endDate;  
 private Integer currentMatchday;  
 public LocalDate getStartDate() { return startDate; }  
 public void setStartDate(LocalDate startDate) { this.startDate = startDate; }  
 public Integer getCurrentMatchday() { return currentMatchday; }  
 public void setCurrentMatchday(Integer currentMatchday) {his.currentMatchday currentMatchday; }  
 public LocalDate getEndDate() { return endDate; }  
 public void setEndDate(LocalDate endDate) { this.endDate = endDate; } }  
 private List<Match> convertToMatchEntities(List<FootballDataMatch> apiMatches) {  
 return apiMatches.stream().map(apiMatch -> {  
 Match match = new Match();  
 match.setId(apiMatch.getId());  
 match.setHomeTeam(apiMatch.getHomeTeam().getName());  
 match.setAwayTeam(apiMatch.getAwayTeam().getName());  
 Match.Score score = new Match.Score();  
 FootballDataMatch.Score apiScore = apiMatch.getScore();  
 if (apiScore.getFullTime() != null) {  
 score.setHomeFullTime(apiScore.getFullTime().getHome());  
 score.setAwayFullTime(apiScore.getFullTime().getAway()); }  
 if (apiScore.getHalfTime() != null) {  
 score.setHomeHalfTime(apiScore.getHalfTime().getHome());  
 score.setAwayHalfTime(apiScore.getHalfTime().getAway()); }  
 score.setWinner(apiScore.getWinner());  
 match.setScore(score);  
 if (apiMatch.getReferees() != null) {  
 match.setReferees(apiMatch.getReferees().stream()  
 .map(ref -> {  
 Match.MatchReferee referee = new Match.MatchReferee();  
 referee.setName(ref.getName());  
 return referee; }) .collect(Collectors.*toList*())); }  
 FootballDataMatch.Score.FullTime fullTime = apiMatch.getScore().getFullTime();  
 match.setHomeScore(fullTime != null ? fullTime.getHome() : null);  
 match.setAwayScore(fullTime != null ? fullTime.getAway() : null);  
 match.setHomeCrest(apiMatch.getHomeTeam().getCrest());  
 match.setAwayCrest(apiMatch.getAwayTeam().getCrest());  
 ZonedDateTime utcDateTime = apiMatch.getUtcDate();  
 LocalDateTime istTime = utcDateTime.withZoneSameInstant(ZoneId.*of*("Asia/Kolkata")).toLocalDateTime();  
 match.setMatchDate(istTime);  
 match.setStatus(apiMatch.getStatus());  
 match.setMatchday(apiMatch.getMatchday());  
 return match; }).collect(Collectors.*toList*()); }}

**MatchComment.java**

package com.pl.premier\_zone.comments;  
import com.pl.premier\_zone.match.Match;  
import com.pl.premier\_zone.user.User;  
import jakarta.persistence.\*;  
import java.time.LocalDateTime;  
@Entity  
@Table(name = "match\_comments")  
public class MatchComment {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 @Column(nullable = false)  
 private String content;  
 @ManyToOne  
 @JoinColumn(name = "match\_id", nullable = false)  
 private Match match;  
 @ManyToOne  
 @JoinColumn(name = "user\_id", nullable = false)  
 private User user;  
 @Column(name = "created\_at")  
 private LocalDateTime createdAt;  
 @Column(name = "updated\_at")  
 private LocalDateTime updatedAt;  
 @PrePersist  
 protected void onCreate() {  
 createdAt = LocalDateTime.*now*();  
 updatedAt = LocalDateTime.*now*(); }  
 @PreUpdate  
 protected void onUpdate() {  
 updatedAt = LocalDateTime.*now*(); }   
 public Long getId() { return id; }  
 public void setId(Long id) { this.id = id; }  
 public String getContent() { return content; }  
 public void setContent(String content) { this.content = content; }  
 public Match getMatch() { return match; }  
 public void setMatch(Match match) { this.match = match; }  
 public User getUser() { return user; }  
 public void setUser(User user) { this.user = user; }  
 public LocalDateTime getCreatedAt() { return createdAt; }  
 public LocalDateTime getUpdatedAt() { return updatedAt; }}

**MatchCommentController.java**

package com.pl.premier\_zone.comments;  
import com.pl.premier\_zone.comments.dto.MatchCommentDto;  
import com.pl.premier\_zone.exception.NotFoundException;  
import com.pl.premier\_zone.exception.UnauthorizedException;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import java.util.List;  
import java.util.Map;  
@RestController  
@RequestMapping("/api/v4/matches/{matchId}/comments")  
@CrossOrigin(origins = "http://localhost:3000")  
public class MatchCommentController {  
 private final MatchCommentService commentService;  
 public MatchCommentController(MatchCommentService commentService) { this.commentService = commentService; }  
 @GetMapping  
 public ResponseEntity<List<MatchCommentDto>> getComments(@PathVariable Long matchId) {  
 return ResponseEntity.*ok*(commentService.getMatchComments(matchId)); }  
 @PostMapping  
 public ResponseEntity<?> addComment( // Changed return type to handle errors  
 @PathVariable Long matchId,  
 @RequestBody Map<String, String> payload,  
 @RequestHeader("Authorization") String token) {  
 try { String content = payload.get("content");  
 if (content == null || content.trim().isEmpty()) {  
 return ResponseEntity.*badRequest*().body("Content cannot be empty"); }  
 MatchCommentDto comment = commentService.addComment(matchId, content, token);  
 return ResponseEntity.*ok*(comment);  
 } catch (UnauthorizedException e) {  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body(e.getMessage());  
 } catch (NotFoundException e) {  
 return ResponseEntity.*status*(HttpStatus.*NOT\_FOUND*).body(e.getMessage());  
 } catch (Exception e) {  
 e.printStackTrace(); // Add this for debugging  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*)  
 .body("Error adding comment: " + e.getMessage()); } }}

**MatchCommentService.java**

package com.pl.premier\_zone.comments;  
import com.pl.premier\_zone.comments.dto.MatchCommentDto;  
import com.pl.premier\_zone.exception.UnauthorizedException;  
import com.pl.premier\_zone.match.Match;  
import com.pl.premier\_zone.match.MatchRepository;  
import com.pl.premier\_zone.security.JwtService;  
import com.pl.premier\_zone.user.User;  
import com.pl.premier\_zone.user.UserRepository;  
import org.springframework.data.crossstore.ChangeSetPersister;  
import org.springframework.stereotype.Service;  
import java.util.List;  
import java.util.stream.Collectors;  
import com.pl.premier\_zone.exception.NotFoundException;  
import org.springframework.transaction.annotation.Transactional;  
@Service  
@Transactional  
public class MatchCommentService {  
 private final MatchCommentRepository commentRepository;  
 private final MatchRepository matchRepository;  
 private final UserRepository userRepository;  
 private final JwtService jwtService;  
 public MatchCommentService(  
 MatchCommentRepository commentRepository,  
 MatchRepository matchRepository,  
 UserRepository userRepository,  
 JwtService jwtService) {  
 this.commentRepository = commentRepository;  
 this.matchRepository = matchRepository;  
 this.userRepository = userRepository;  
 this.jwtService = jwtService; }  
 public List<MatchCommentDto> getMatchComments(Long matchId) {  
 return commentRepository.findByMatchIdOrderByCreatedAtDesc(matchId)  
 .stream()  
 .map(MatchCommentDto::new)  
 .collect(Collectors.*toList*()); }  
 public MatchCommentDto addComment(Long matchId, String content, String token) throws ChangeSetPersister.NotFoundException {  
 System.*out*.println("Received request - matchId: " + matchId + ", content: " + content);   
 String username = jwtService.validateTokenAndGetUsername(token.substring(7));  
 System.*out*.println("Username from token: " + username); // Add this  
 if (username == null) {  
 throw new UnauthorizedException("Invalid token"); }  
 User user = userRepository.findByUsername(username);   
 Match match = matchRepository.findById(matchId)  
 .orElseThrow(() -> new NotFoundException("Match not found"));   
 MatchComment comment = new MatchComment();  
 comment.setContent(content);  
 comment.setMatch(match);  
 comment.setUser(user);  
 MatchComment savedComment = commentRepository.save(comment);   
 return new MatchCommentDto(savedComment); }}

**index.JS**

import React, { useState, useEffect } from 'react';

import { Card, CardHeader, CardTitle, CardContent } from '../ui/card';

import "./index.scss";

*const* TopScorersComponent = () => {

*const* [scorers, setScorers] = useState([]);

*const* [loading, setLoading] = useState(true);

*const* [error, setError] = useState(null);

  useEffect(() => {

*const* fetchScorers = *async* () => {

      try {  *const* response = *await* fetch('http://localhost:8080/api/v4/scorers');

        if (!response.ok) {

          throw new Error('Network response was not ok');     }

*const* data = *await* response.json();

        setScorers(data.scorers || []);

        setLoading(false);

      } catch (error) {

        console.error('Error fetching scorers:', error);

        setError('Failed to fetch top scorers data');

        setLoading(false);     }  };

    fetchScorers();

  }, []);

  if (loading) {

*return* (

      <div className="scorers-page">

        <div className="min-h-[400px] flex items-center justify-center">

          <div className="text-lg text-gray-600">Loading top scorers...</div>

        </div>

      </div>    );  }

  if (error) {

*return* (

      <div className="scorers-page">

        <div className="min-h-[400px] flex items-center justify-center">

          <div className="text-lg text-red-500">{error}</div>

        </div>

      </div>   );  }

*return* (

    <div className="scorers-page">

      <Card className="scorers-card">

        <CardHeader className="scorers-header">

          <CardTitle>Premier League Top Scorers</CardTitle>

        </CardHeader>

        <CardContent className="scorers-content">

          <div className="scorers-table-container">

            <table className="scorers-table">

              <thead>

                <tr>      <th className="pos-header" data-tooltip="Position">Pos</th>

                  <th className="player-header" data-tooltip="Player">Player</th>

                  <th className="team-header" data-tooltip="Team">Team</th>

                  <th data-tooltip="Goals">Goals</th>

                  <th data-tooltip="Assists">Assists</th>

                  <th data-tooltip="Penalties">Penalties</th>

                  <th data-tooltip="Matches Played">MP</th>      </tr>

              </thead>

              <tbody>

                {scorers.map((scorer, index) => (

                  <tr key={scorer.player.id} className="scorer-row">

                    <td className="position">

                      <span className={`position-indicator ${index < 3 ? 'pos-top' : 'pos-regular'}`}>

                        {index + 1}

                      </span>

                    </td>

                    <td className="player-name">

                      <div className="player-info">

                        <span>{scorer.player.name}</span>

                    </div>                   </td>

                    <td className="team-name">

                      <div className="team-info">

                        <img           src={scorer.team.crest}

                          alt={scorer.team.shortName}

                          className="team-crest"      />

              <span>{scorer.team.shortName}</span>

                      </div> </td>

                    <td className="goals">{scorer.goals}</td>

                    <td>{scorer.assists || 0}</td>

                    <td>{scorer.penalties || 0}</td>

                    <td>{scorer.playedMatches}</td>

                  </tr>  ))}

              </tbody>

            </table>

          </div>

        </CardContent>

      </Card>

    </div>  );};export default TopScorersComponent;

**INDEX.SCSS**

.scorers-page {

    padding: 2rem;

    width: 100%;

    min-height: 100vh;

    background-color: #ade8f4;

    display: flex;

    justify-content: center;

    align-items: flex-start;

    box-sizing: border-box;  }

   .scorers-card {

    width: 100%;

    max-width: 1000px;

    margin: 0 auto;

    background: white;

    border-radius: 12px !important;

    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1) !important;

        &:hover {

      box-shadow: 0 8px 12px rgba(0, 0, 0, 0.15) !important;    }  }

    .scorers-header {

    background: linear-gradient(to right, #0077b6, #00b4d8) !important;

    border-radius: 12px 12px 0 0;

    padding: 1.5rem !important;

        h3 {

      color: white !important;

      text-align: center;

      margin: 0;    }  }

    .scorers-content {

    padding: 1.5rem !important;  }

    .scorers-table-container {

    overflow-x: auto;  }

    .scorers-table {

    width: 100%;

    border-collapse: collapse;

        th, td {

      padding: 1rem;

      text-align: center;

      color: #1a202c;

      font-weight: 500;

            &.player-header, &.team-header {

        text-align: left;      }    }

        th {

      background-color: #f8f9fa;

      font-weight: 700;

      color: #1a202c;

      border-bottom: 2px solid #e2e8f0;    }

        .scorer-row {

      border-bottom: 1px solid #e2e8f0;

        &:hover {

        background-color: #f8fafc;      }

       td {

        font-size: 0.95rem;

        color: #1a202c;      }    }

      .player-name, .team-name {

      text-align: left;

      font-weight: 600;    }

    .team-info, .player-info {

      display: flex;

      align-items: center;

      gap: 1rem;    }      .team-crest {

      width: 24px;

      height: 24px;

      object-fit: contain;    }

      .position {

      width: 40px;  }

      .position-indicator {

      display: inline-flex;

      align-items: center;

      justify-content: center;

      width: 28px;

      height: 28px;

      border-radius: 50%;

      font-weight: 600;

            &.pos-top {

        background-color: #fef3c7;

        color: #92400e;    }

          &.pos-regular {

        background-color: #f1f5f9;

        color: #1a202c;      }    }

        .goals {

      font-weight: 700;

      color: #15803d;    }

    thead tr th {

      font-weight: 700;

      color: #0f172a;

      text-transform: uppercase;

      font-size: 0.85rem;

      letter-spacing: 0.05em;    }  }

    @media (max-width: 768px) {

    .scorers-table {

      th, td {

        padding: 0.75rem 0.5rem;

        font-size: 0.875rem;      }

    .team-crest {

        width: 20px;

        height: 20px;      }    }  }

**index.js**

import React, { useEffect, useState } from "react";

import Loader from "react-loaders";

import { Link } from 'react-router-dom';

import "./index.scss";

import AnimatedLetters from "../AnimatedLetters";

import teamData from "../../data/teams.json";

*const* Teams = () => {

*const* [letterClass, setLetterClass] = useState('text-animate');

*const* [searchQuery, setSearchQuery] = useState('');

*const* [filteredTeams, setFilteredTeams] = useState([]);

  useEffect(() => {

*const* timer = setTimeout(() => {

            setLetterClass("text-animate-hover");       }, 3000);

*return* () => clearTimeout(timer);    });

    useEffect(() => {

*const* filtered = teamData.teams.filter(team =>

            team.title.toLowerCase().includes(searchQuery.toLowerCase())        );

        setFilteredTeams(filtered);

    }, [searchQuery]);

*const* handleSearchChange = event => {

        setSearchQuery(event.target.value);    };

*const* renderTeam = (teams) => {

*return* (           <div className="images-container">

                {teams.map((team, idx) => (

                    <div key={idx} className="image-box">

                        <img

                            src={team.cover}

                            alt={team.title}

                            className="teams-image"                />

                        <div className="content">

                            <p className="title">{team.title}</p>

                            <Link             className="btn"

                                to={`/data?team=${encodeURIComponent(team.title)}`}             >

                                View </Link> </div></div>))}</div);};

*return* (<>

            <div className="nation-page">

                <h1 className="page-title">

                    <AnimatedLetters

                        letterClass={letterClass}

                        strArray={"Teams".split("")}

                        idx={15}           />

                </h1>

                <div className="search-bar">

                    <input             type="text"

                        placeholder="Search for teams"

                        value={searchQuery}

                        onChange={handleSearchChange}       />

                </div>

                <div className="nations-container">

                    <div className="nations-header">

                        <h3>Football Teams</h3>         </div>

                    <div className="nations-content">

                        {renderTeam(filteredTeams)}</div> </div>

            </div>   <Loader type="pacman" />     </>    );};

export default Teams;

index.scss

.teams-page {

    padding: 2rem;

    width: 100%;

    min-height: 100vh;

    background-color: #ade8f4;

    display: flex;

    flex-direction: column;

    align-items: center;

    box-sizing: border-box;

    h1.page-title {

        color: #0077b6;

        font-size: 3.5rem;

        font-weight: bold;

        margin: 2rem 0;

        text-align: center;    }

    .search-bar {

        width: 100%;

        max-width: 600px;

        margin: 2rem auto;

        position: relative;

        padding: 0 20px;

        box-sizing: border-box;

        input {

            width: 100%;

            padding: 15px 45px;

            background: white;

            border: 1px solid #e2e8f0;

            border-radius: 12px;

            color: #1a202c;

            font-size: 1.1rem;

            transition: all 0.3s ease;

            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

            &:focus {

                outline: none;

                border-color: #0077b6;

                box-shadow: 0 0 0 2px rgba(0, 119, 182, 0.2);     }

            &::placeholder {

                color: #64748b;           }        }   }

    .teams-container {

        width: 100%;

        max-width: 1200px;

        background: white;

        border-radius: 12px;

        box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

        margin: 0 auto;

        overflow: hidden;

        .teams-header {

            background: linear-gradient(to right, #0077b6, #00b4d8);

            padding: 1.5rem;

            h3 {

                color: white;

                text-align: center;

                margin: 0;

                font-size: 1.5rem;      }

        .teams-content {

            padding: 1.5rem;     }

        .images-container {

            display: grid;

            grid-template-columns: repeat(4, 1fr);

            gap: 20px;

            padding: 20px 0;

            width: 100%;

            box-sizing: border-box       }

        .image-box {

            position: relative;

            aspect-ratio: 3/2;

            max-height: 200px;

            overflow: hidden;

            border-radius: 10px;

            box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

            transition: all 0.3s ease;

            background: white;

        .teams-image {

                width: 100%;

                height: 100%;

                object-fit: contain;          }

  .content {

                position: absolute;

                width: 100%;

                z-index: 3;

                padding: 15px;

                transition: all 0.3s ease;

                background: linear-gradient(

                    180deg,

                    rgba(0, 119, 182, 0) 0%,

                    rgba(0, 119, 182, 0.8) 50%,

                    rgba(0, 119, 182, 0.95) 100%      );

                bottom: -70px;

                box-sizing: border-box;     }

      .title {

                margin: 0;

                color: white;

                font-size: 1.2rem;

                font-weight: 600;

                text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.3);      }

            .btn {

                display: inline-block;

                margin-top: 10px;

                padding: 8px 20px;

                border: 2px solid white;

                border-radius: 6px;

                font-size: 0.9rem;

                color: white;

                background: transparent;

                text-transform: uppercase;

                font-weight: 600;

                transition: all 0.3s ease;

                text-decoration: none;

                &:hover {

                    background: white;

                    color: #0077b                    }        }    }}

@media screen and (max-width: 1400px) {

    .teams-page .teams-container .images-container {

        grid-template-columns: repeat(3, 1fr);    }}

@media screen and (max-width: 1100px) {

    .teams-page{

        padding: 1.5rem;

    .teams-container .images-container {

        grid-template-columns: repeat(2, 1fr);

        gap: 15px;    }

    h1.page-title{

        font-size: 3rem;    }}}}

**index.js**

import React, { useState, useEffect } from 'react';

import { Card, CardHeader, CardTitle, CardContent } from '../ui/card';

import "./index.scss";

*const* StandingsComponent = () => {

*const* [standings, setStandings] = useState([]);

*const* [loading, setLoading] = useState(true);

*const* [error, setError] = useState(null);

  useEffect(() => {  *const* fetchStandings = *async* () => {   try {

*const* response = *await* fetch('http://localhost:8080/api/v4/standings');

        if (!response.ok) {

          throw new Error('Network response was not ok');    }

*const* data = *await* response.json();

        setStandings(data.standings[0].table || []);

        setLoading(false);

      } catch (error) {

        console.error('Error fetching standings:', error);

        setError('Failed to fetch standings data');

        setLoading(false);      }    };

    fetchStandings();  }, []);

  if (loading) {

*return* (    <div className="standings-page">

        <div className="min-h-[400px] flex items-center justify-center">

          <div className="text-lg text-gray-600">Loading standings...</div>

        </div>      </div>    );  }

  if (error) {

*return* (      <div className="standings-page">

        <div className="min-h-[400px] flex items-center justify-center">

          <div className="text-lg text-red-500">{error}</div>

        </div>     </div>    );  }

*return* (   <div className="standings-page">

      <Card className="standings-card">

        <CardHeader className="standings-header">

          <CardTitle>Premier League Standings</CardTitle>

        </CardHeader>

        <CardContent className="standings-content">

          <div className="standings-table-container">

            <table className="standings-table">

            <thead>         <tr>

                    <th className="pos-header" data-tooltip="Position">Pos</th>

                    <th className="team-header" data-tooltip="Team">Team</th>

                    <th data-tooltip="Matches Played">MP</th>

                    <th data-tooltip="Wins">W</th>

                    <th data-tooltip="Draws">D</th>

                    <th data-tooltip="Losses">L</th>

                    <th data-tooltip="Goals For">GF</th>

                    <th data-tooltip="Goals Against">GA</th>

                    <th data-tooltip="Goal Difference">GD</th>

                    <th data-tooltip="Points">Pts</th>         </tr>

            </thead>

              <tbody>

                {standings.map((team) => (

                  <tr key={team.team.id} className="team-row">

                    <td className="position">

                      <span className={`position-indicator pos-${getPositionClass(team.position)}`}>

                        {team.position}

                      </span>

                    </td>

                    <td className="team-name">

                      <div className="team-info">

                        <img

                          src={team.team.crest}

                          alt={team.team.shortName}

                          className="team-crest"             />

                        <span>{team.team.shortName}</span>    </div>

                    </td>

                    <td>{team.playedGames}</td>

                    <td>{team.won}</td>

                    <td>{team.draw}</td>

                    <td>{team.lost}</td>

                    <td>{team.goalsFor}</td>

                    <td>{team.goalsAgainst}</td>

                    <td className={team.goalDifference >= 0 ? 'positive' : 'negative'}>

                      {team.goalDifference > 0 ? '+' : ''}{team.goalDifference}

                    </td>

                    <td className="points">{team.points}</td>

                  </tr>        ))}

              </tbody>

            </table>   </div>

        </CardContent>

      </Card>   </div>  );};

export default StandingsComponent;

**index.js**

import React, { useEffect, useState } from "react";

import Loader from "react-loaders";

import "./index.scss";

import AnimatedLetters from "../AnimatedLetters";

*const* Search = () => {

*const* [letterClass, setLetterClass] = useState('text-animate');

*const* [searchQuery, setSearchQuery] = useState('');

    useEffect(() => {

*const* timer = setTimeout(() => {

            setLetterClass("text-animate-hover");

        }, 3000);

*return* () => {

            clearTimeout(timer);        }   }, []);

*const* handleSearchChange = event => {

        setSearchQuery(event.target.value);    };

*const* handleGoButtonClick = () => {

        window.location.href = `/data?name=${encodeURIComponent(searchQuery)}`;    };

*return* (      <>

            <div className="container teams-page">

                <h1 className="page-title">

                    <br/>       <br/>

                    <AnimatedLetters letterClass={letterClass} strArray={"Search".split("")} idx={15}/>         </h1>

                <div className="search-bar">

                    <input

              type="text"

                        placeholder="Search for players"

                        value={searchQuery}

                        onChange={handleSearchChange}

                  />      <button onClick={handleGoButtonClick}>Go</button>

              </div>     </div>

       <Loader type="pacman"/>        </>    );}

export default Search;

**index.js**

import React, { useEffect, useState } from "react";

import Loader from "react-loaders";

import { Link } from 'react-router-dom';

import "./index.scss";

import AnimatedLetters from "../AnimatedLetters";

import positionData from "../../data/positions.json";

*const* Positions = () => {

*const* [letterClass, setLetterClass] = useState('text-animate');

*const* [searchQuery, setSearchQuery] = useState('');

*const* [filteredPositions, setFilteredPositions] = useState([]);

    useEffect(() => {

*const* timer = setTimeout(() => {

            setLetterClass("text-animate-hover");

        }, 3000);

*return* () => {

            clearTimeout(timer);       }, []);

    useEffect(() => {

*const* filtered = positionData.positions.filter(position =>

            position.title.toLowerCase().includes(searchQuery.toLowerCase())    );

        setFilteredPositions(filtered);

    }, [searchQuery]);

*const* handleSearchChange = event => {

        setSearchQuery(event.target.value);   };

*const* renderPositions = (positions) => {

*return* (   <div className="images-container">

                {positions.map((position, idx) => (

                    <div key={idx} className="image-box">

                        <img src={position.cover} alt="positions" className="teams-image" />

                        <div className="content">

                            <p className="title">{position.title}</p>

                            <Link

                                className="btn"

                                to={`/data?position=${encodeURIComponent(position.search)}`}       >

                                View     </Link> </div></div>     ))       </div>        )    };    *return* (        <>

            <div className="position-page">

                <h1 className="page-title">

                    <AnimatedLetters

                        letterClass={letterClass}

                        strArray={"Positions".split("")}

                        idx={15}           />

                </h1>

                <div className="search-bar">

                    <input

                        type="text"

                        placeholder="Search for positions"

                        value={searchQuery}

                        onChange={handleSearchChange}             />

                </div>

                <div className="positions-container">

                    <div className="positions-header">

                        <h3>Football Positions</h3>

                    </div>          <div className="positions-content">

                        {renderPositions(filteredPositions)                </div>

                </div>

            </div>

            <Loader type="pacman"/>

        </>    );}

**export default Positions;**

**index.js**

import React, { useState, useEffect } from 'react';

import { Card, CardHeader, CardTitle, CardContent } from '../ui/card';

import { Calendar, ExternalLink } from 'lucide-react';

import "./index.scss";

*const* NewsComponent = () => {

*const* [news, setNews] = useState([]);

*const* [loading, setLoading] = useState(true);

*const* [error, setError] = useState(null);

  useEffect(() => {

*const* fetchNews = *async* () => {

      try {   *const* response = *await* fetch('http://localhost:8080/api/v1/news');

        if (!response.ok) {

          throw new Error('Network response was not ok');       }

*const* data = *await* response.json();

        setNews(data.articles || []);

        setLoading(false);

      } catch (error) {

        console.error('Error fetching news:', error);

        setError('Failed to fetch news data');

        setLoading(false);    }    };

    fetchNews();

*const* interval = setInterval(fetchNews, 15 \* 60 \* 1000);

*return* () => clearInterval(interval)  }, []);

*const* formatDate = (dateString) => {

*return* new Date(dateString).toLocaleDateString('en-US', {

      year: 'numeric',

      month: 'short',

      day: 'numeric'    });  };

  if (loading) {

*return* (     <div className="news-page">

        <div className="min-h-[400px] flex items-center justify-center">

          <div className="text-lg text-gray-800">Loading news...</div>

        </div>

      </div>    );  }

  if (error) {

*return* (      <div className="news-page">

        <div className="min-h-[400px] flex items-center justify-center">

          <div className="text-lg text-red-600">{error}</div>

        </div>    </div>   );  }

*return* (

    <div className="news-page">

      <Card className="news-card">

        <CardHeader className="news-header">

          <CardTitle>Premier League News</CardTitle>

        </CardHeader>

        <CardContent className="news-content">

          <div className="space-y-4">

            {news.map((item, index) => (

              <div key={index} className="news-item">

                <div className="news-item-content">

                  <h3 className="news-title">

                    {item.title}

                  </h3>

                  <p className="news-description">

                    {item.description}          </p>

                  <div className="news-meta">

                    <div className="news-date">

                      <Calendar className="w-4 h-4" />

                      <span>{formatDate(item.publishedAt)}</span>

                    </div>

                    <div className="news-source">

                      Source: {item.source.name}       </div>

                    <a

                      href={item.url}

                      target="\_blank"

                      rel="noopener noreferrer"

                      className="news-link"

                    >          Read more <ExternalLink className="w-4 h-4 ml-1" />          </a>                </div>    </div>

              </div>     ))        </div>

        </CardContent>

      </Card>

    </div> );};

export default NewsComponent;

import React, { useEffect, useState } from "react";

import Loader from "react-loaders";

import { Link } from 'react-router-dom';

import "./index.scss";

import AnimatedLetters from "../AnimatedLetters";

import nationData from "../../data/nations.json";

import { ReactCountryFlag } from 'react-country-flag';

*const* Nations = () => {

*const* [letterClass, setLetterClass] = useState('text-animate');

*const* [searchQuery, setSearchQuery] = useState('');

*const* [filteredNations, setFilteredNations] = useState([]);

    useEffect(() => {

*const* timer = setTimeout(() => {

            setLetterClass("text-animate-hover");

        }, 3000);

*return* () => {

            clearTimeout(timer);        }    });

    useEffect(() => {

*const* filtered = nationData.nations.filter(nation =>

            nation.name.toLowerCase().includes(searchQuery.toLowerCase())      );        setFilteredNations(filtered);

    }, [searchQuery]);

*const* handleSearchChange = event => {

        setSearchQuery(event.target.value);    };

*const* renderCountryFlags = (countries) => {

*return* (            <div className="images-container">

                {countries.map((country, idx) => (

                    <div key={idx} className="image-box">

                        <ReactCountryFlag

                            countryCode={country.code}

                            svg

                            style={{

                                width: '100%',

                                height: '100%',                          }}                      />

                        <div className="content">

                            <p className="title">{country.name}</p>

                            <Link

                                className="btn"

                                to={`/data?nation=${encodeURIComponent(country.search)}`}                      >

                                View

                            </Link>

                      </div>

                  </div>                ))}            </div>        );    };

*return* (        <>

            <div className="nation-page">

                <h1 className="page-title">

                    <AnimatedLetters

                        letterClass={letterClass}

                        strArray={"Nations".split("")}

                        idx={15}                 />

                </h1>

                <div className="search-bar">                    <input

                        type="text"

                        placeholder="Search for countries"

                        value={searchQuery}

                        onChange={handleSearchChange}         />

              </div>

                <div className="nations-container">

                    <div className="nations-header">

                        <h3>National Teams</h3>

              </div>            <div className="nations-content">

                        {renderCountryFlags(filteredNations)}

                    </div>             </div>        </div>

            <Loader type="pacman" />      </>    );}

export default Nations;

import { useEffect, useState } from 'react';

import Loader from 'react-loaders';

import { Link } from 'react-router-dom';

import LogoPL from '../../assets/images/PL.webp';

import AnimatedLetters from '../AnimatedLetters';

import ScrollSection from '../ScrollSection';

import './index.scss';

*const* Home = () => {

*const* [letterClass, setLetterClass] = useState('text-animate');

*const* [currentSlide, setCurrentSlide] = useState(0);

*const* [username, setUsername] = useState(null);

*const* nameArray = "Welcome to".split("");

*const* jobArray = "Premier Zone Fantasy!".split("");

*const* backgroundImages = [

        '/premierpic/one.jpg',

        '/premierpic/two.jpg',

        '/premierpic/three.jpg',

        '/premierpic/four.jpg',

        '/premierpic/five.jpg',

        '/premierpic/six.jpg'

    ];

    useEffect(() => {

*const* storedUsername = localStorage.getItem('username');

        if (storedUsername) {

            setUsername(storedUsername);        }

*const* timerId = setTimeout(() => {

            setLetterClass('text-animate-hover');        }, 4000);

*return* () => clearTimeout(timerId);    }, []);

    useEffect(() => {

*const* slideInterval = setInterval(() => {

            setCurrentSlide((prev) =>

                prev === backgroundImages.length - 1 ? 0 : prev + 1       );      }, 5000);

*return* () => clearInterval(slideInterval);

    }, [backgroundImages.length]);

*const* handleLogout = () => {

        localStorage.removeItem('token');

        localStorage.removeItem('username');

        setUsername(null);

    };

*return* (        <>

            <div className="container home-page">

                <nav className="nav-auth">                    {username ? (

                      <div className="user-welcome">

                            <span>Welcome, {username}</span>

                            <button onClick={handleLogout} className="auth-button">LOGOUT</button>

                        </div>

                    ) :                   <>

                            <Link to="/login" className="auth-button">LOGIN</Link>

                            <Link to="/register" className="auth-button">REGISTER</Link>

                        </>                    )}                </nav>

                <div className="slideshow-background">

                    {backgroundImages.map((img, index) => (

                        <div      key={index}

                            className={`slide ${index === currentSlide ? 'active' : ''}`}

                            style={{ backgroundImage: `url(${img})` }}         />             ))}

                    <div className="gradient-overlay" />   </div>

                <div className="text-zone">

                    <h1>

                        <img src={LogoPL} alt="PremierZone" />

                        <AnimatedLetters letterClass={letterClass} strArray={nameArray} idx={12} />

                        <AnimatedLetters letterClass={letterClass} strArray={jobArray} idx={15} />          </h1>

                    <h2>Your home for everything Premier League related!</h2>

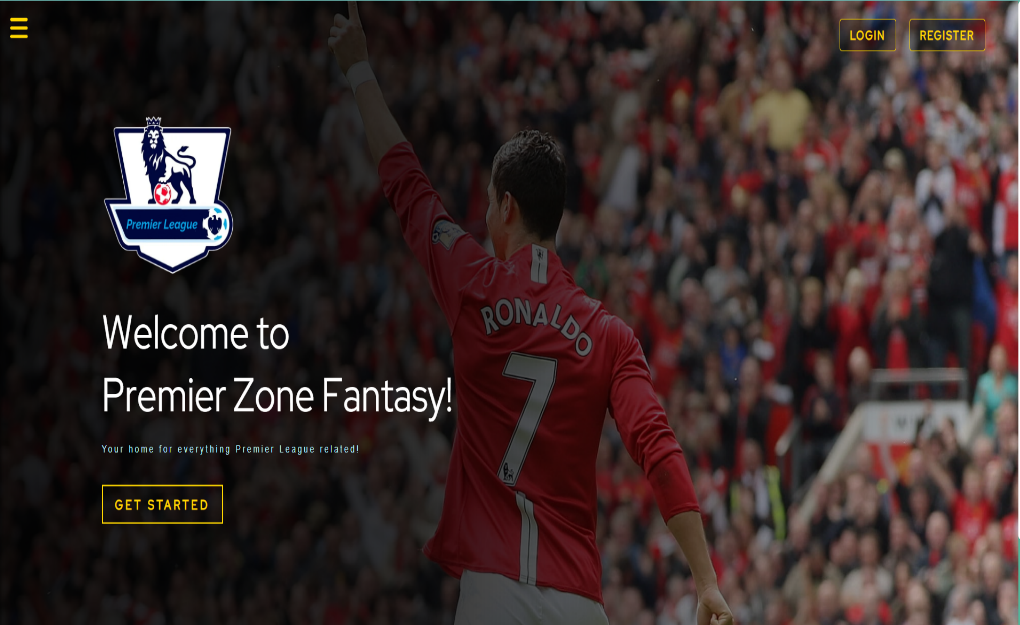
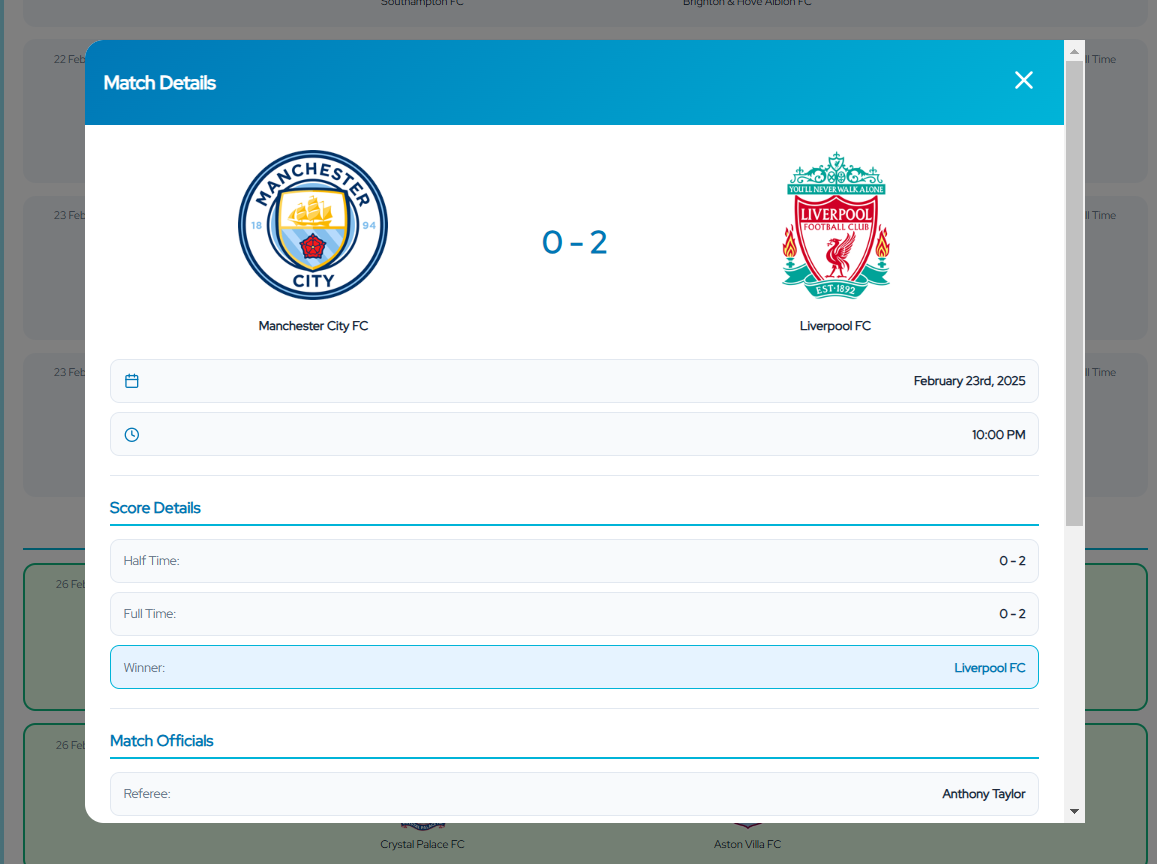
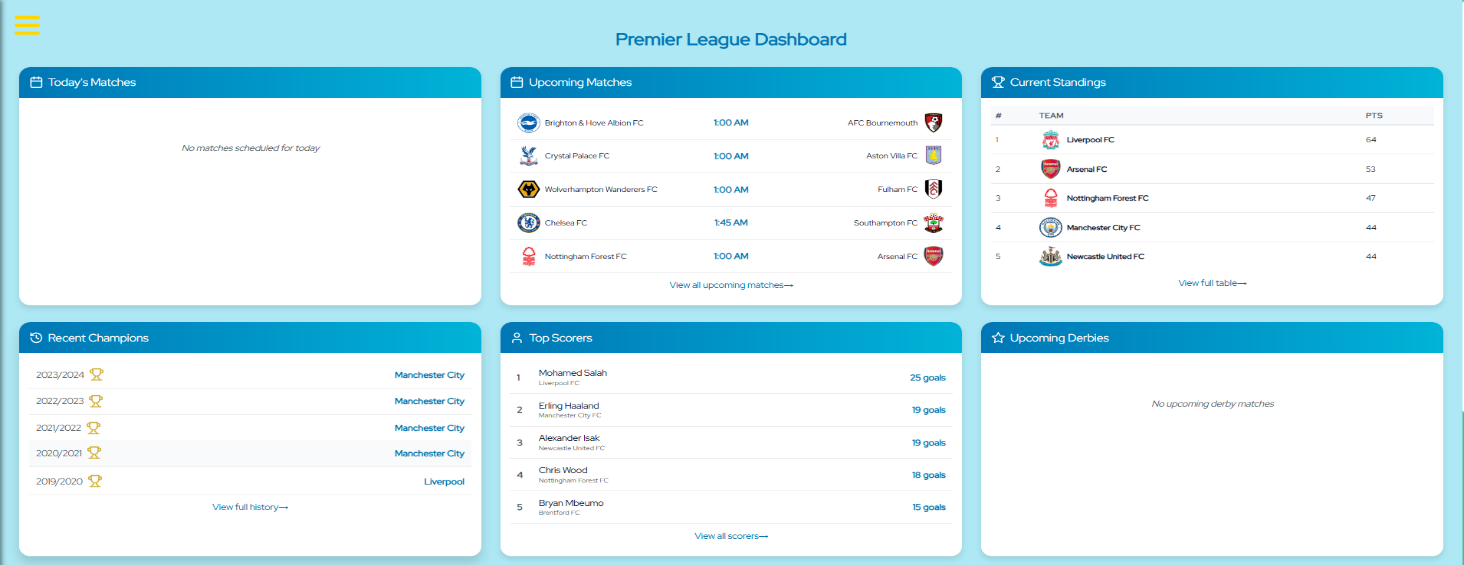
                    <Link to="/dashboard" className="flat-button">GET STARTED</Link>

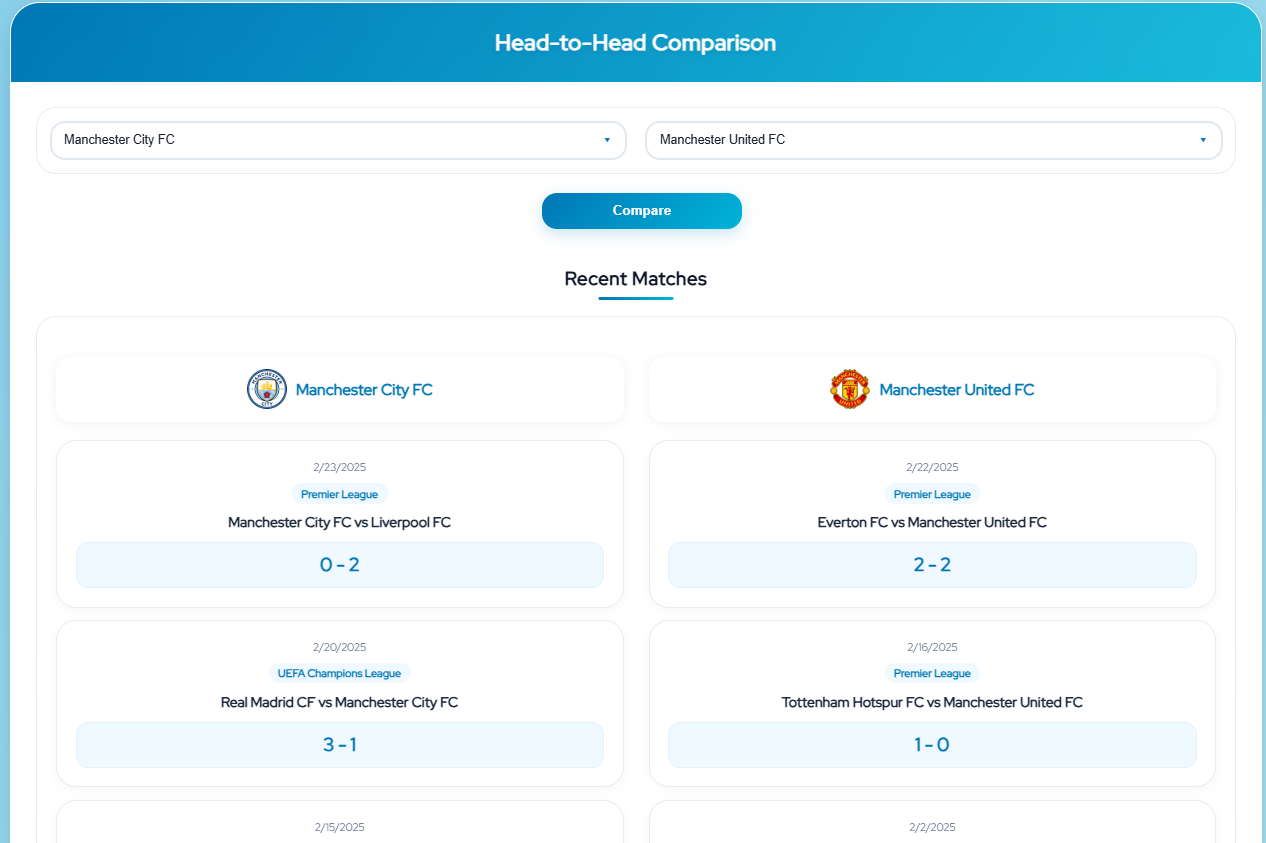
                </div>           </div>

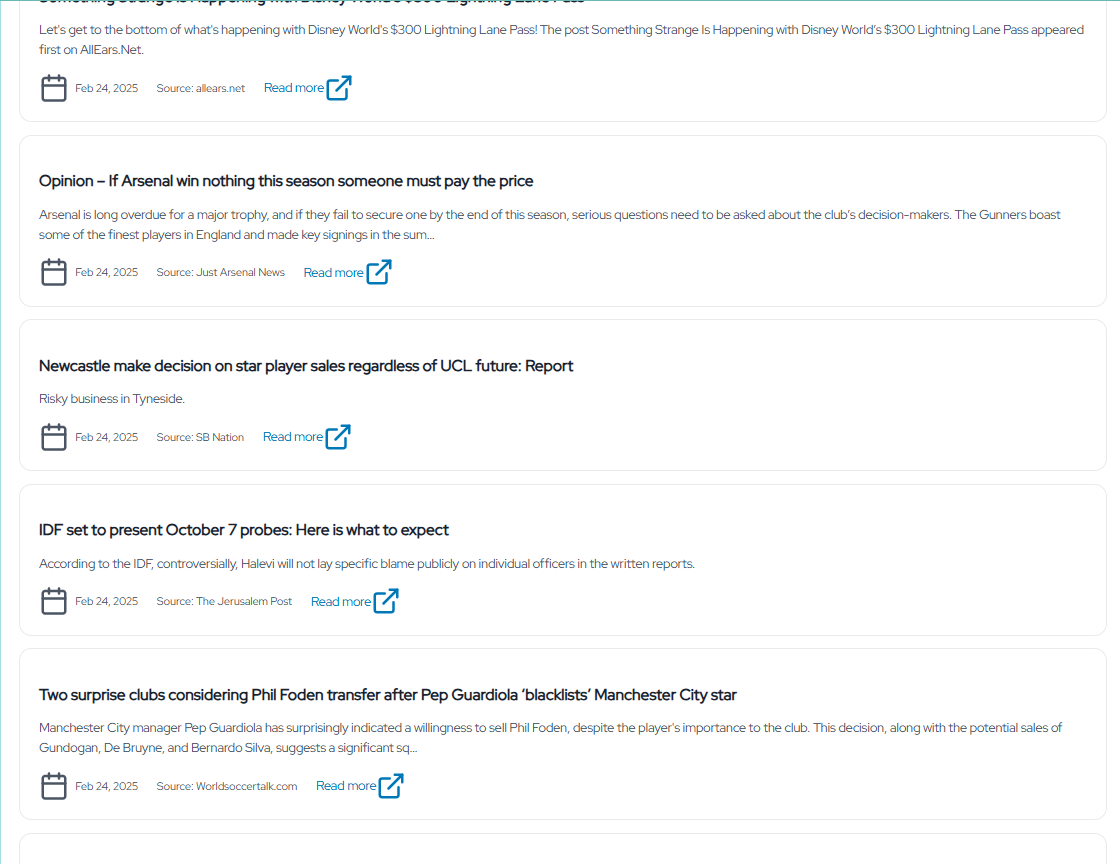
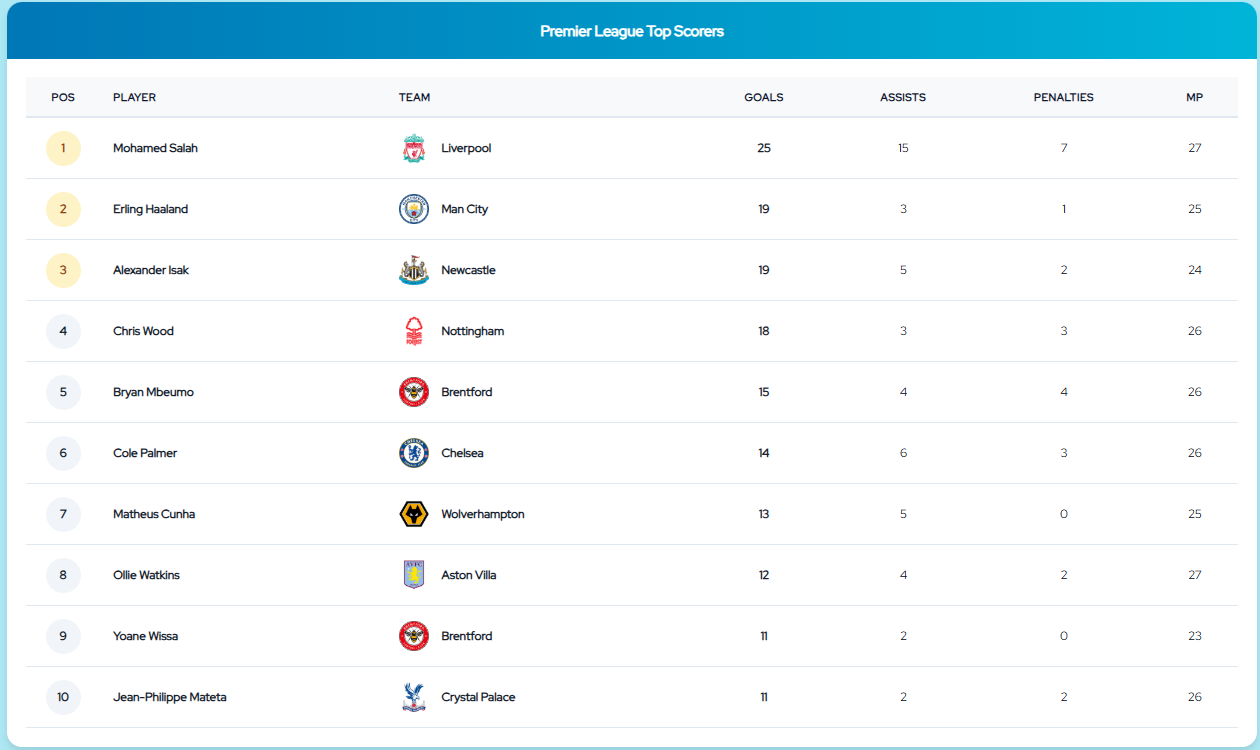
            <ScrollSection />

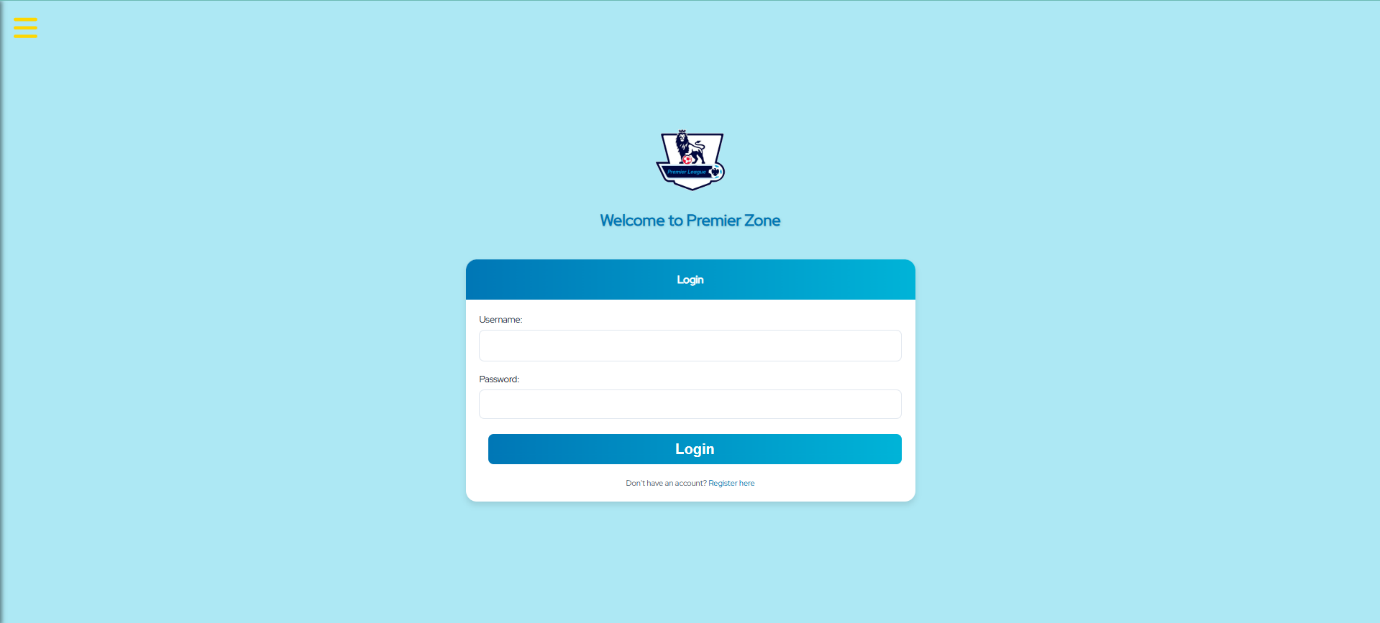
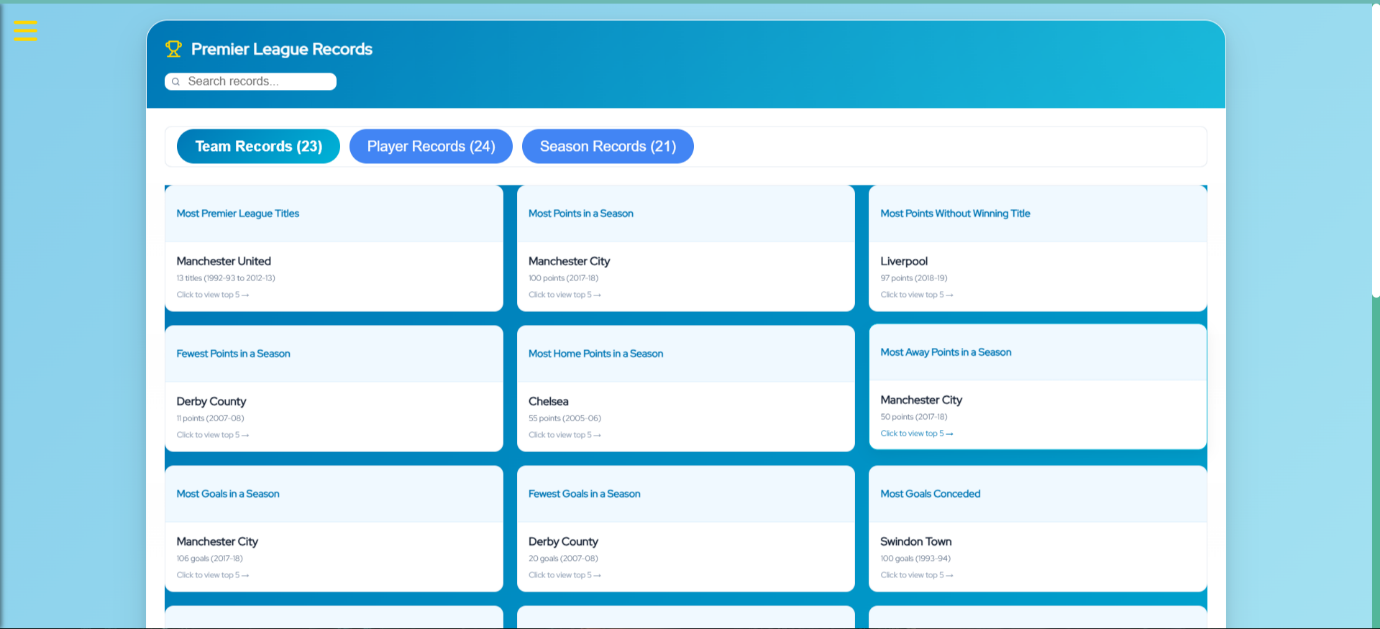
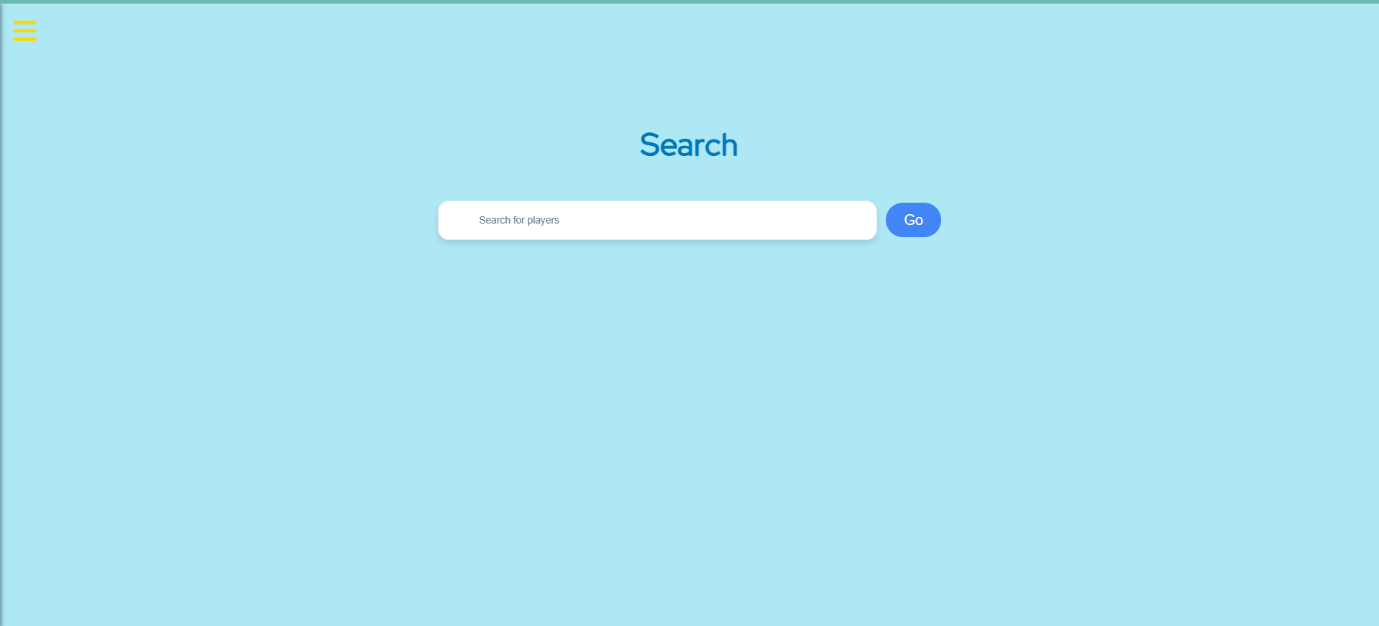
            <Loader type="pacman" />     </>    );};

export default Home;

******RESULTS**





**CONCLUSION**

The **Premier Zone Fantasy** project successfully delivers a full-stack web application dedicated to providing Premier League enthusiasts with an engaging and interactive experience. With a **React-based frontend** and a **Spring Boot backend**, the platform ensures a smooth and responsive user experience while leveraging a **PostgreSQL database** to store crucial data, including users, comments, and player statistics.

The system integrates multiple features, including **real-time match updates, team standings, top scorers, head-to-head comparisons, and historical records**—all fetched dynamically from reliable external APIs. The **search functionality** and **position-based player listings** enhance usability, enabling users to explore teams and players efficiently. Additionally, user authentication with **JWT tokens**, along with email notifications for new registrations, ensures both security and a personalized experience.

The implementation of **Spring Caffeine caching** optimizes performance, reducing server load while keeping data retrieval efficient. Furthermore, rigorous testing using **JUnit and Mockito** ensures system reliability and correctness. The deployment strategy using **Render** provides a scalable and cost-effective hosting solution for both the frontend and backend.

By combining modern web technologies with a user-friendly interface, **Premier Zone Fantasy** stands as a robust and scalable platform for football fans. Future enhancements may include **live match commentary, fantasy league integration, and user-generated content features** to further improve engagement.

This project demonstrates the potential of **full-stack development, API integration, and database management** in creating dynamic, data-driven applications. It serves as both a learning experience in software engineering best practices and a valuable tool for football enthusiasts worldwide.

**FUTURE SCOPE**

The **Premier Zone Fantasy** project has been designed with scalability and future enhancements in mind. While the current system provides a comprehensive Premier League experience, there are several potential improvements and expansions that can further enhance its functionality and user engagement.

**1. Advanced Fantasy League Integration**

* Implement a **fantasy football** system where users can create their own teams, select players, and earn points based on real-life performances.
* Introduce **leaderboards** and competitions to make the platform more interactive.

**2. Live Match Commentary & Analysis**

* Add **real-time commentary** for ongoing matches, providing in-depth analysis and instant updates.
* Integrate AI-based **match predictions** and player performance forecasts.

**3. Enhanced User Engagement Features**

* Introduce a **social discussion forum** where users can share opinions, polls, and match reviews.
* Enable **user-generated content**, such as blogs and articles, to increase community participation.

**4. Multi-League Expansion**

* Extend the platform beyond the Premier League to include other major football leagues like **La Liga, Bundesliga, Serie A, and UEFA competitions**.

**5. Video Highlights & Media Sharing**

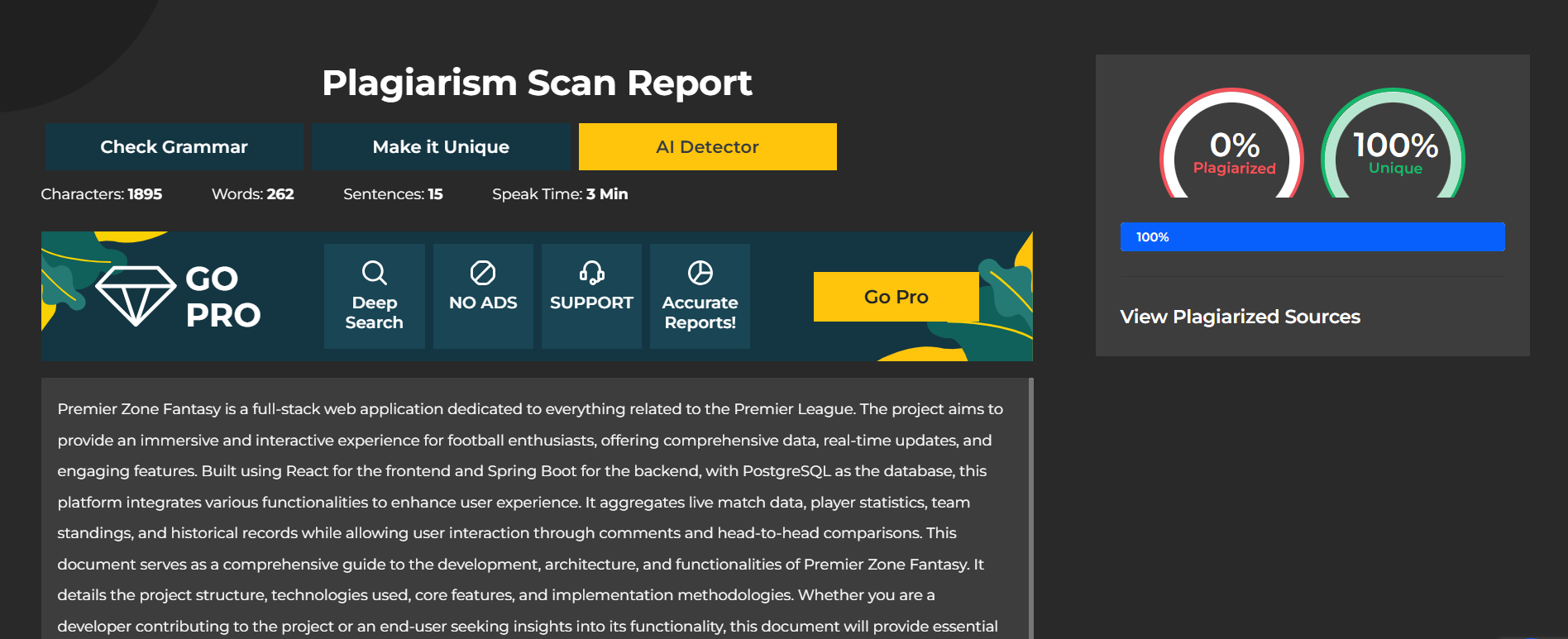
* Integrate **video highlights, goal replays, and key match moments** fetched from official sources.
* Allow users to **share clips, memes, and reactions** related to their favorite teams and matches.

**6. Enhanced Security & Performance Optimization**

* Implement **role-based access control (RBAC)** for better user management.
* Optimize API requests and **database indexing** to improve performance and reduce load times.

By continuously evolving and implementing these enhancements, **Premier Zone Fantasy** can become a leading platform for football enthusiasts, offering a **unique blend of statistics, entertainment, and interactive engagement**.

**PLAGIARISM REPORT**



**REFERENCES**

* [**https://github.com/**](https://github.com/)
* [**https://react.dev/**](https://react.dev/)
* [**https://spring.io/projects/spring-boot**](https://spring.io/projects/spring-boot)
* [**https://www.postgresql.org/docs/**](https://www.postgresql.org/docs/)
* [**https://maven.apache.org/guides/**](https://maven.apache.org/guides/)
* [**https://newsapi.org/**](https://newsapi.org/)
* [**https://www.football-data.org/**](https://www.football-data.org/)
* [**https://render.com/docs**](https://render.com/docs)
* [**https://junit.org/junit5/docs/current/user-guide/**](https://junit.org/junit5/docs/current/user-guide/)
* [**https://site.mockito.org/**](https://site.mockito.org/)
* [**https://jwt.io/introduction/**](https://jwt.io/introduction/)
* [**https://learning.postman.com/**](https://learning.postman.com/)
* [**https://claude.ai/**](https://claude.ai/)
* [**https://www.youtube.com/**](https://www.youtube.com/)
* [**https://stackoverflow.com/**](https://stackoverflow.com/)
* [**https://www.canva.com/**](https://www.canva.com/)
* [**https://www.figma.com/**](https://www.figma.com/)
* [**https://www.lucidchart.com/pages/**](https://www.lucidchart.com/pages/)
* [**https://plagiarismdetector.net/**](https://plagiarismdetector.net/)
* [**https://openai.com/chatgpt/**](https://openai.com/chatgpt/)
* [**https://www.google.com/**](https://www.google.com/)