

# System Analysis & Design

## 1. Problem Statement & Objectives

### 1.1 Problem Statement

Football fans worldwide struggle to access match results, league standings, and interactive prediction games in a single, user-friendly platform. While several sports websites provide live scores, they often lack an engaging experience that allows users to predict match results, compete with others, and track their success over time.

Additionally, some platforms require expensive subscriptions for detailed match data, limiting accessibility for casual fans. There is a need for an interactive football news and prediction platform that provides reliable match data, user-friendly interactions, and a competitive prediction game.

### 1.2 Project Objectives

The goal of **MatchMind** is to develop an engaging football news platform that allows users to:

- 1) View match scores, results, and league standings in an intuitive way.
- 2) Participate in a prediction game, earning points for accurate match outcomes.
- 3) Create user accounts to save match predictions and track progress.
- 4) Enable admin management for manually updating match results and moderating user activity.
- 5) Provide an accessible and mobile-friendly experience for users worldwide.

## Use Case Diagram:

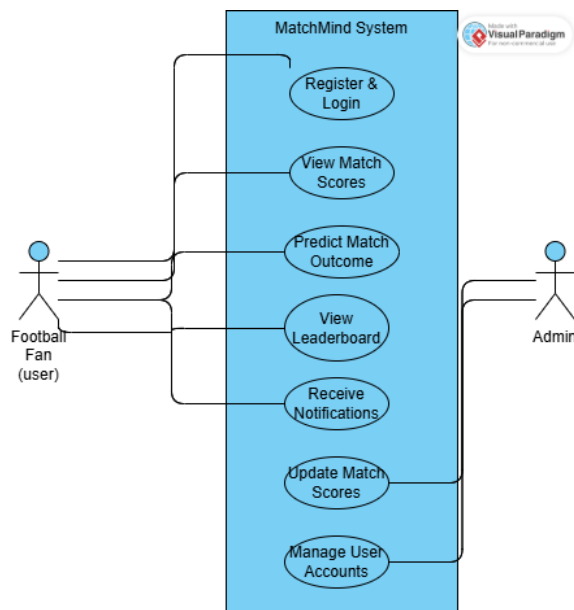
### Overview

The Use Case Diagram illustrates the main functionalities of the MatchMind System, an interactive football news platform that allows users to view match scores, predict match outcomes, and track their performance on a leaderboard. The system includes two primary actors:

1. Football Fan (User) – The main user who interacts with the system.
2. Admin – Responsible for managing user accounts and updating match scores.

The MatchMind System facilitates football fans worldwide to create accounts, receive notifications, and engage in match prediction games, while

administrators ensure data accuracy and system management.



## Functional Requirements (System Capabilities):

### User Management

Users must be able to register an account with an email and password.  
Users must be able to log in and log out securely.  
Users must be able to update their profile information (e.g., username, email).

### Match & Score Features

Users must be able to view live and past match scores.  
Admins must be able to manually update match scores after a game ends.

### Prediction Game Features

Users must be able to predict match outcomes before the match starts.  
The system must store user predictions and compare them with actual results.  
The system must assign points to users based on correct predictions.  
Users must be able to view a leaderboard ranking the best predictors.

### Notifications & Alerts

Users must receive notifications for match updates and leaderboard changes.  
Users must receive an alert if their prediction is successful.

### Admin Features

Admins must be able to update match results.  
Admins must be able to manage user accounts (suspend or delete users if necessary).

## **Non-Functional Requirements (System Constraints & Performance Criteria):**

### Performance & Reliability

The system must load match data within 3 seconds.  
The system must be available 24/7 with 99.9% uptime.

### Security & Data Protection

User data (passwords, emails) must be encrypted for security.  
The system must prevent unauthorized access to admin functionalities.  
The system must have data backup to restore lost information.

### Usability & Accessibility

The website must be mobile-friendly and responsive.  
The user interface must be simple and easy to navigate.  
The system should support multiple languages (if needed for global users).

### Scalability & Maintainability

The system should be scalable to handle thousands of users.  
The system should allow for future feature upgrades without major downtime.

## **Software Architecture:**

### Architecture Style

The MatchMind system follows the Model-View-Controller (MVC) architecture.  
This structure separates the system into three key layers:

- Model (Data Layer) – Manages data storage and retrieval.
- View (Presentation Layer) – Handles user interface and interactions.
- Controller (Logic Layer) – Processes requests, manages business logic, and connects the View to the Model.

### High-Level System Components

#### 1. Frontend (View Layer - Client Side)

Technology: HTML, CSS, JavaScript (React.js or Vue.js)  
Purpose: Handles user interactions and displays match data.

Interaction: The frontend communicates with the backend via RESTful APIs or GraphQL.

#### 2. Backend (Controller Layer - Application Logic)

Technology: Node.js (Express.js) / Django / Spring Boot

Purpose: Handles all business logic, processes user requests, and interacts with the database.

Interaction: Receives requests from the frontend and interacts with the database and external APIs if needed.

### 3. Database (Model Layer - Data Storage)

Technology: MySQL / PostgreSQL / MongoDB

Purpose: Stores all user data, match information, and prediction results.

Interaction: The backend retrieves and updates data as needed.

### System Interaction Flow (How Components Work Together)

The interaction between components follows this workflow:

User opens the website → Frontend sends a request to the backend.

User logs in / registers → Backend verifies credentials using the database.

User views match scores → Backend fetches match data from the database.

User makes a prediction → Backend stores the prediction in the database.

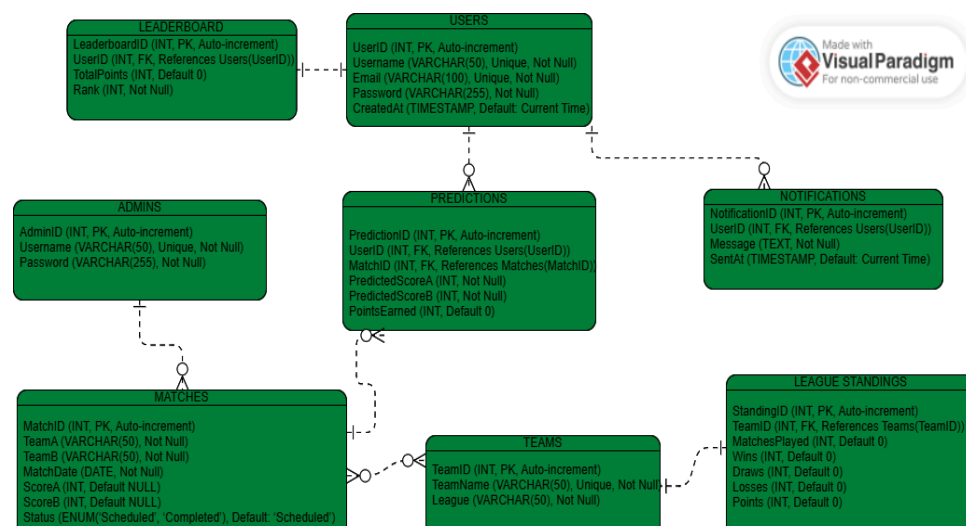
Admin updates match results → Backend updates scores in the database.

Leaderboard updates automatically → Backend recalculates rankings based on prediction accuracy.

Notifications are sent to users → Backend triggers alerts for results & ranking changes.

### **Database Design & Data Modeling:**

• ER Diagram:



• Logical Schema:

Table 1 : User

Column Name	Data Type	Constraints
UserID	INT (PK)	AUTO_INCREMENT, NOT NULL
Username	VARCHAR(50)	UNIQUE, NOT NULL
Email	VARCHAR(100)	UNIQUE, NOT NULL
Password	VARCHAR(255)	NOT NULL
CreatedAt	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Table 2 :Matches

Column Name	Data Type	Constraints
MatchID	INT (PK)	AUTO_INCREMENT, NOT NULL
TeamA	VARCHAR(50)	NOT NULL
TeamB	VARCHAR(50)	NOT NULL
MatchDate	DATE	NOT NULL
ScoreA	INT	DEFAULT NULL
ScoreB	INT	DEFAULT NULL
Status	ENUM('Scheduled', 'Completed')	DEFAULT 'Scheduled'

Table 3 : Predictions

Column Name	Data Type	Constraints
PredictionID	INT (PK)	AUTO_INCREMENT, NOT NULL
UserID	INT (FK)	REFERENCES Users(UserID)
MatchID	INT (FK)	REFERENCES Matches(MatchID)
PredictedScoreA	INT	NOT NULL
PredictedScoreB	INT	NOT NULL
PointsEarned	INT	DEFAULT 0

Table 4: Leaderboard

Column Name	Data Type	Constraints
LeaderboardID	INT (PK)	AUTO_INCREMENT, NOT NULL
UserID	INT (FK)	REFERENCES Users(UserID)
TotalPoints	INT	DEFAULT 0
Rank	INT	NOT NULL

Table 5 : Admin

Column Name	Data Type	Constraints
AdminID	INT (PK)	AUTO_INCREMENT, NOT NULL
Username	VARCHAR(50)	UNIQUE, NOT NULL
Password	VARCHAR(225)	NOT NULL

### Physical Schema

Normalization Goals:

1NF (First Normal Form): No duplicate data; each field holds atomic values.

2NF (Second Normal Form): All non-key attributes depend on the primary key.

3NF (Third Normal Form): No transitive dependencies; separate tables for users, matches, predictions, etc.

Indexing Considerations:

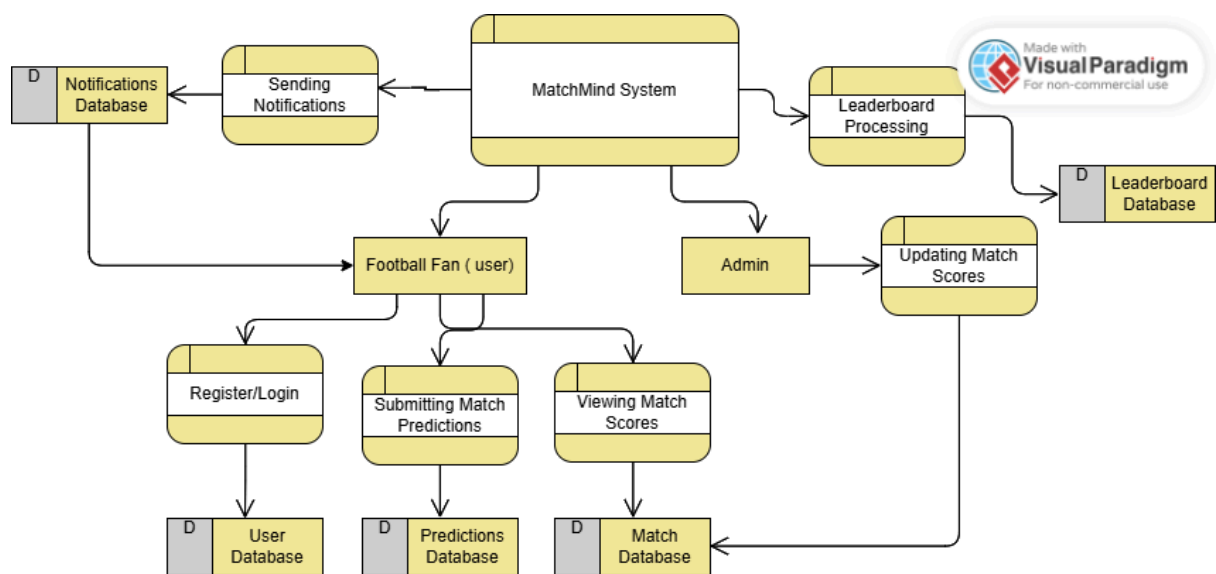
Primary keys (PKs) automatically have indexes.

Foreign keys (FKs) should be indexed for faster query performance.

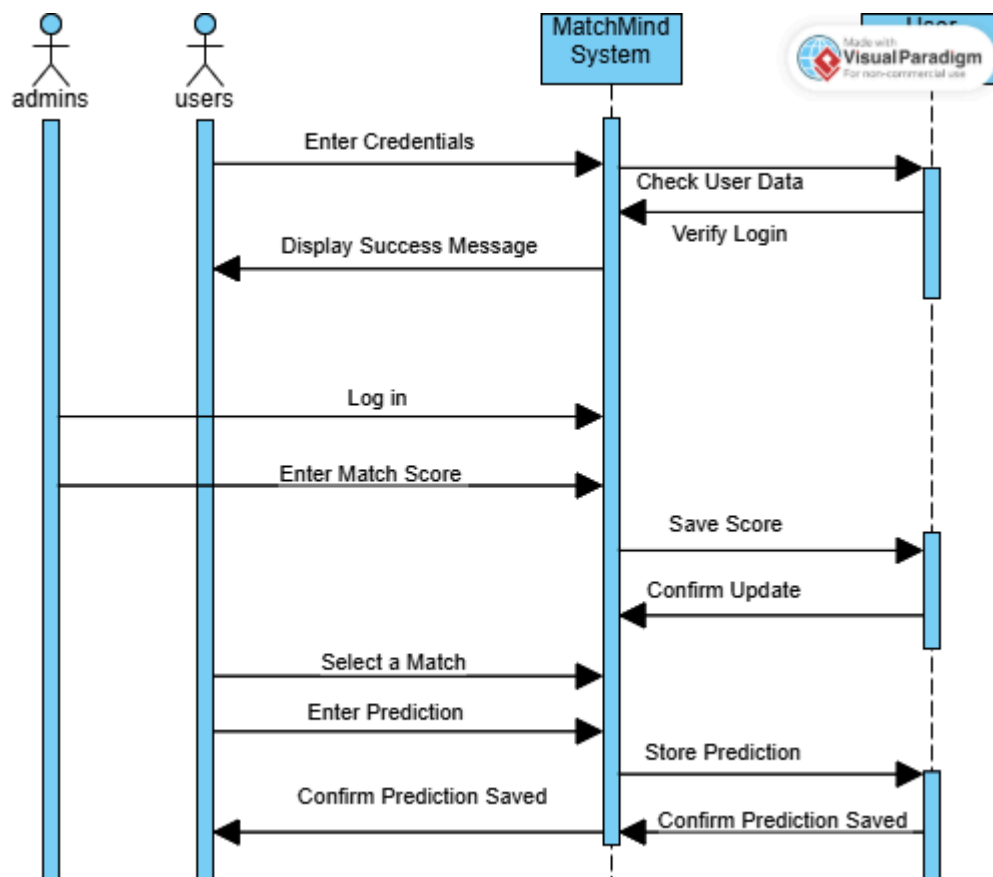
Indexes on frequently searched columns (e.g., MatchDate, TotalPoints).

## Data Flow & System Behavior:

- Data Flow Diagram:

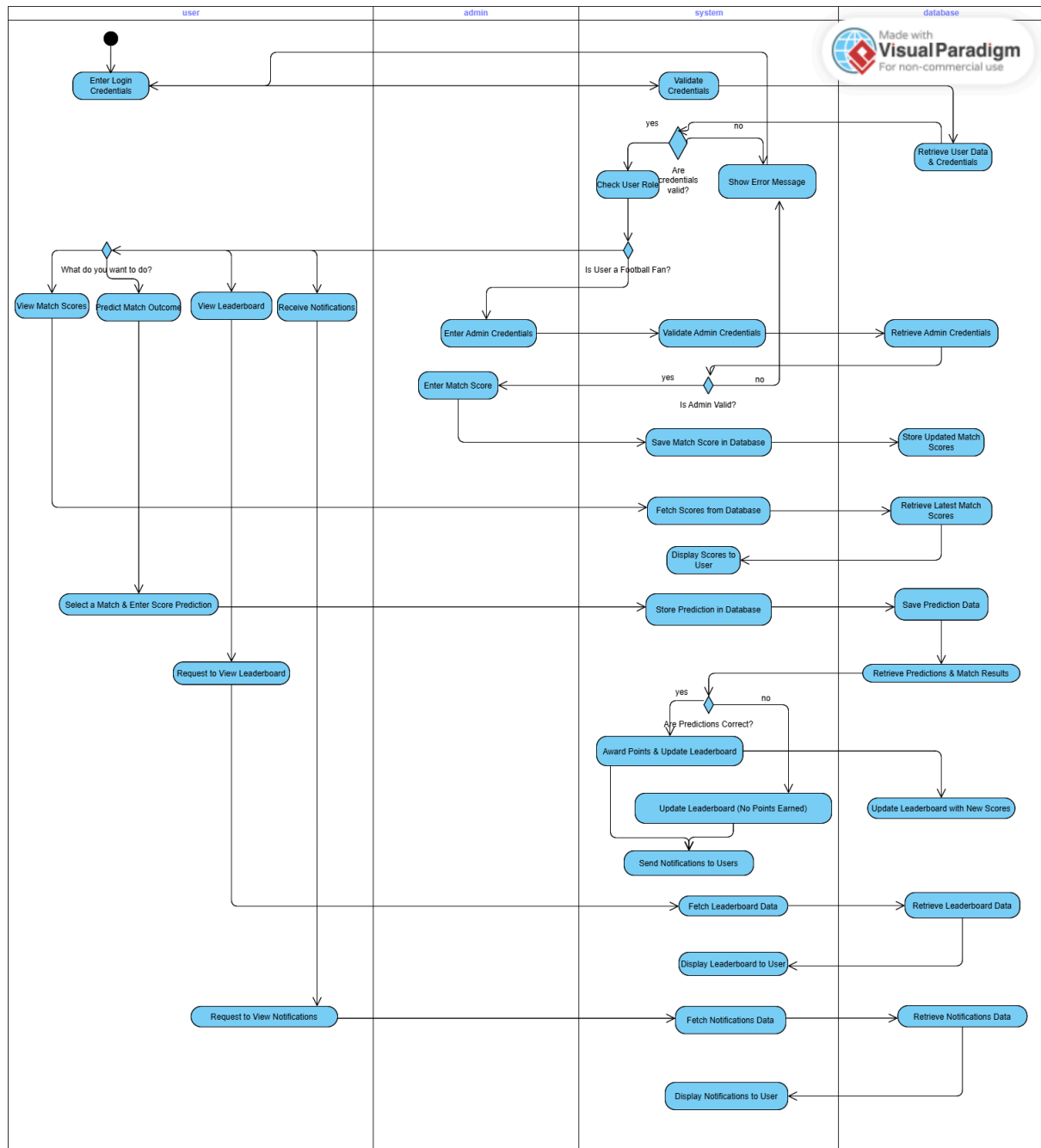


• Sequence Diagram:

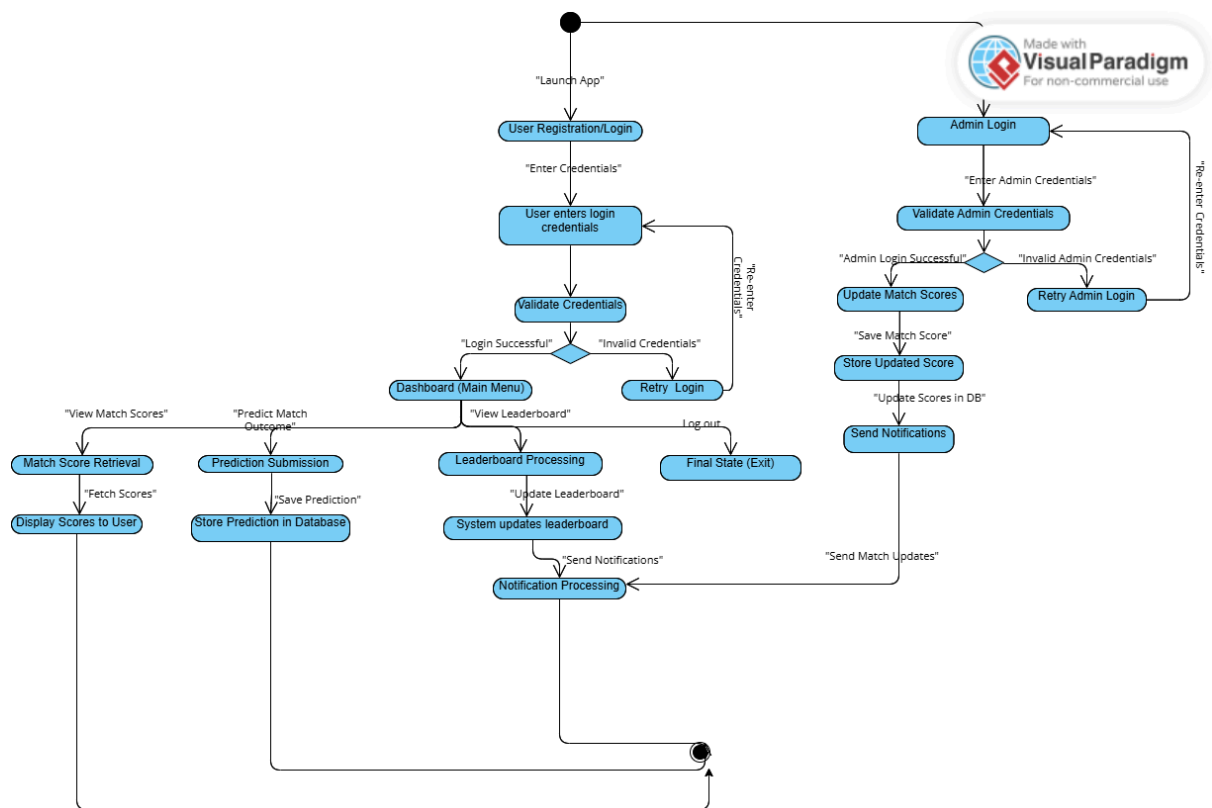




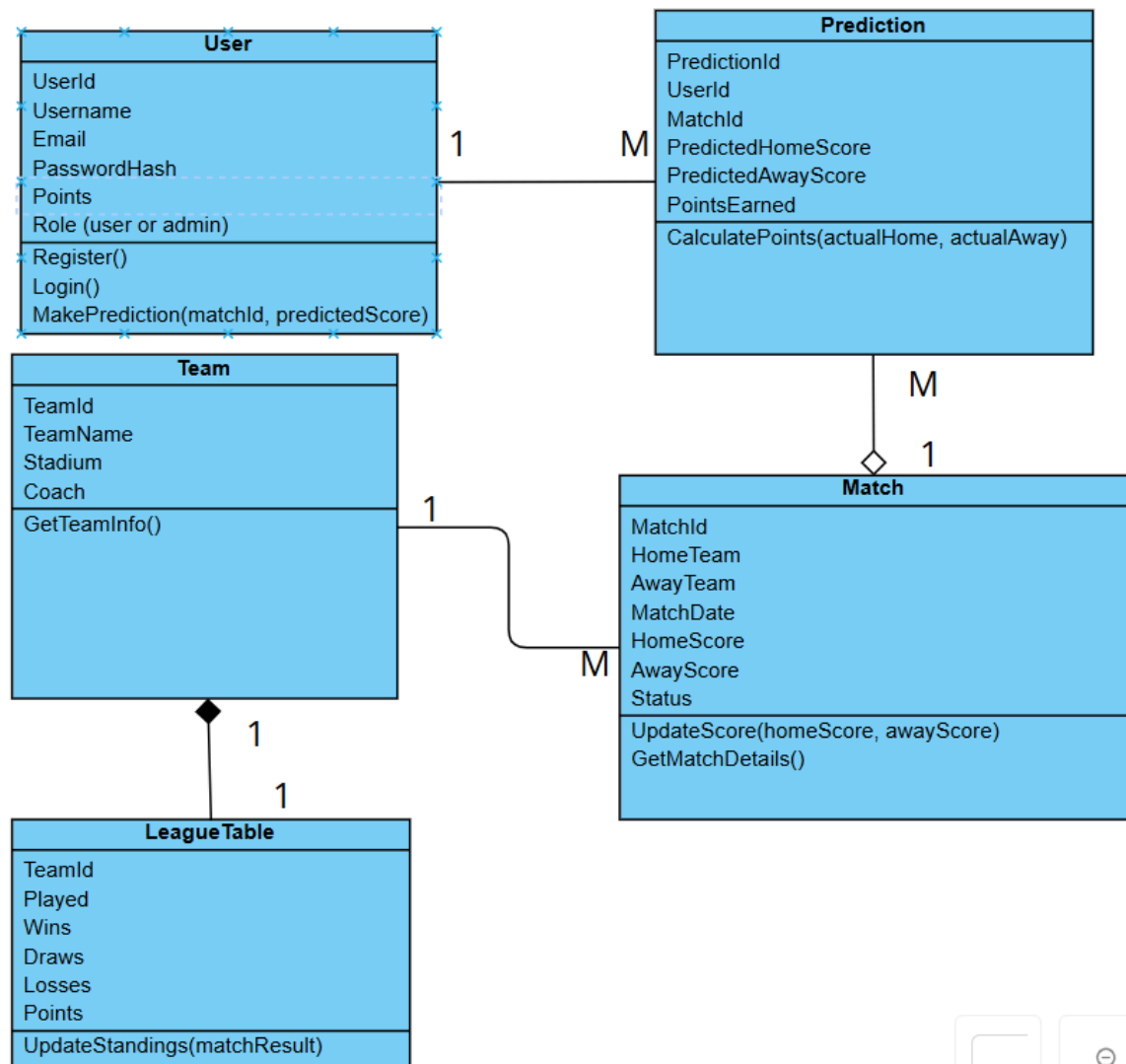
• Activity Diagram:



• State Diagram:

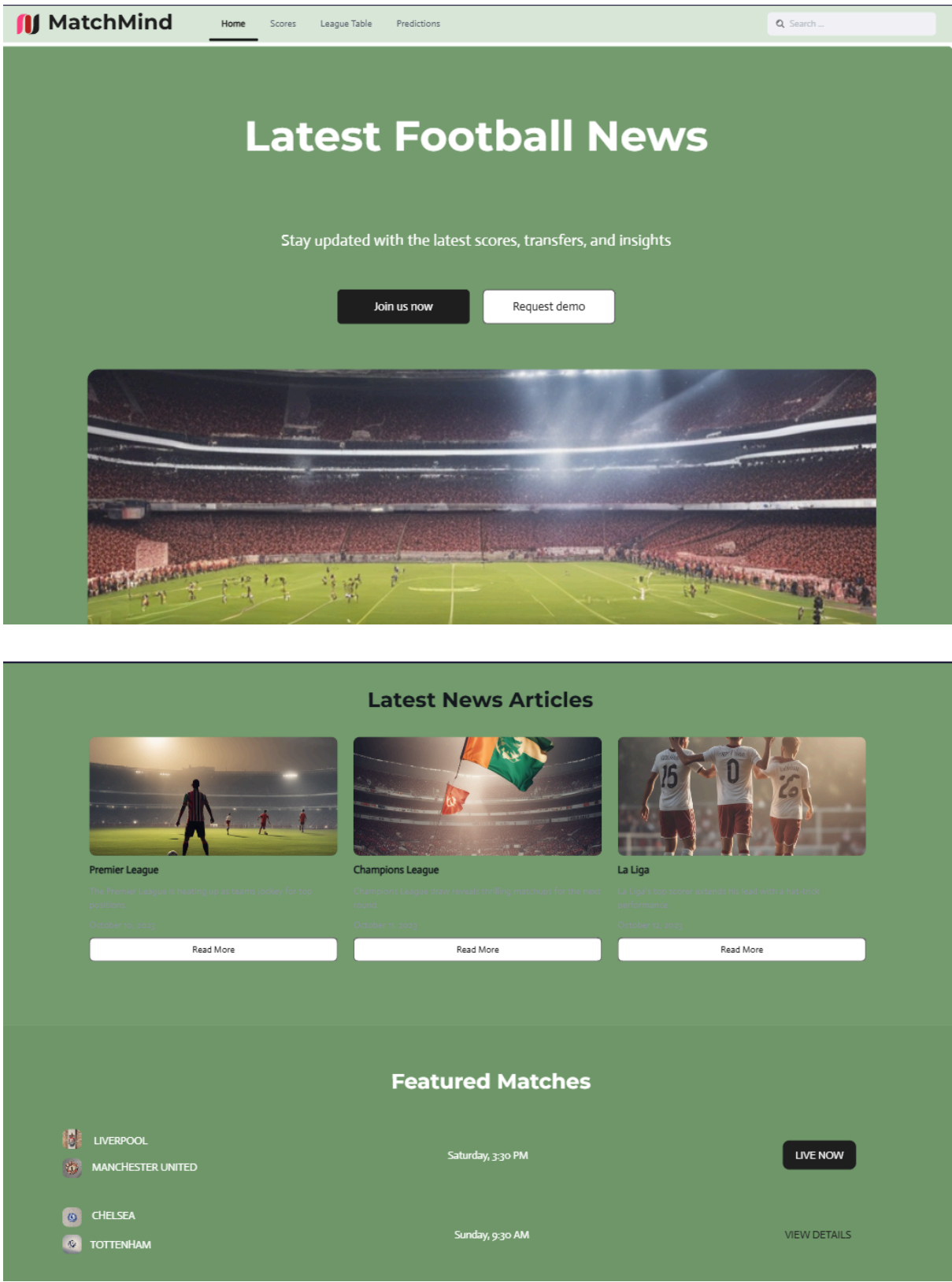







• Class Diagram:




# UI/UX Design & Prototyping:

- Wireframes & Mockups:



 LIVERPOOL	Saturday, 3:30 PM	<a href="#">LIVE NOW</a>
 MANCHESTER UNITED		
 CHELSEA	Sunday, 9:30 AM	<a href="#">VIEW DETAILS</a>
 TOTTENHAM		
 NEWCASTLE UNITED	Monday, 9:30 AM	<a href="#">LIVE NOW</a>
 ARSENAL		



Subscribe to our newsletter

[Subscribe](#)

Product

Features

Pricing

Resources

Blog

User guides

Webinars

Company

About us

Contact us

Plans & Pricing





Personal

Start up

Organization

English

© 2024 Brand, Inc. [Privacy](#) [Terms](#) [Sitemap](#)



Upcoming Matches

Team A vs B

Stadium: X

Date: 2023-11-15, 18:00

3 days left

Team C vs D

Stadium: Y

Date: 2023-11-18, 20:00

6 days left


Team E vs F

Stadium: Z

Date: 2023-11-21, 17:30

9 days left


Team Results



Team A

Won 3-1 against Team B


Date: Oct 5, 2023



Team C

Lost 0-2 to Team D

Date: Oct 7, 2023



Team E

Won 1-0 against Team F

Date: Oct 8, 2023

### Score Prediction

<input type="text" value="Team A"/>	<input type="text" value="Team B"/>
<input type="text" value="o"/>	<input type="text" value="o"/>
<input type="text" value="Team C"/>	<input type="text" value="Team D"/>
<input type="text" value="o"/>	<input type="text" value="o"/>
<input type="text" value="Team E"/>	<input type="text" value="Team F"/>
<input type="text" value="o"/>	<input type="text" value="o"/>

Submit Prediction

### My Points Tracker

142

pts

• UI/UX Guidelines:

### Design Principles:

**Simplicity:** Clean and intuitive layout with easy-to-access features.

**Consistency:** Uniform fonts, colors, and UI elements throughout the site.

**Responsiveness:** Fully functional on mobile, tablet, and desktop devices.

**Fast Load Times:** Optimized images and minimal animations to enhance performance.

### Color Scheme:

The MatchMind color palette is designed for a professional and sporty feel:

**Primary Color:** Green

**Secondary Color:** White

### Typography:

The chosen fonts are modern, clean, and easy to read:

- Primary Font: Montserrat Bold (for headings and titles).
- Secondary Font: Actor (for body text and content).

### Accessibility Considerations:

- Color Contrast: Ensures readability for users with visual impairments.
- Keyboard Navigation: Allows users to navigate using only the keyboard.
- Alt Text for Images: All images have descriptive alternative text for screen readers.

System Deployment & Integration:

• Technology Stack:

Technology	Purpose	Selected Tool
Frontend	User interface & design	Bootstrap (HTML, CSS, JavaScript)
Backend	Server-side logic & API management	C# with ASP.NET Core
Database	Store match data, user info, predictions	SQL Server

**Bootstrap** – Simple, responsive frontend framework with prebuilt UI components.

**C# with ASP.NET Core** – Secure and scalable backend with MVC support.

**SQL Server – Relational database** for structured match data.

• Deployment Diagram:

User (Browser)

|

▼

[Frontend: Bootstrap (HTML, CSS, JS)] (Hosted on Azure / IIS)

|

▼

[Backend: ASP.NET Core + C#] (Hosted on Azure / IIS)

|

▼

[Database: SQL Server] (On-Premise / Azure SQL)

• Component Diagram:

**Frontend (Bootstrap + HTML, CSS, JavaScript):**

- Displays match results & league standings.
- Provides UI for the **prediction game**.
- Handles **user authentication & form submissions**.

**Backend (C# with ASP.NET Core):**

- Processes **match scores, user predictions, and leaderboards**.
- Manages **authentication & authorization** (ASP.NET Identity / JWT).
- Provides **admin functionalities** for score management.

**Database (SQL Server):**

- Stores **match data, user profiles, and prediction history**.
- Maintains **leaderboard rankings and statistics**.



### Additional Deliverables (if applicable):

- API Documentation: no APIs used.
- Testing & Validation:

### Unit Testing:

**Objective:** Test individual components of the system, such as user authentication, match data storage, and prediction processing.

Test Case	Expected Outcome
User can register an account	System successfully creates a new user
User cannot register with an existing email	System prevents duplicate account creation
Match prediction is saved correctly	Prediction data is stored in SQL Server
Admin can update match scores	Score is updated in the database

### Integration Testing:

**Objective:** Ensure that frontend, backend, and database work together smoothly.

Scenario	Expected Result
User logs in and views match scores	System retrieves data from SQL Server and displays correctly
User submits a match prediction	Prediction is saved and reflected in UI
Admin updates a match score	Frontend shows the updated score

## User Acceptance Testing (UAT):

**Objective:** Validate that the system meets user expectations before deployment.

- Test with real users (football fans, testers, stakeholders).
- Gather feedback on usability, speed, and feature functionality.
- Make final adjustments before public launch.

• Deployment Strategy:

## Hosting Environment:

- Frontend (Bootstrap + HTML, CSS, JavaScript) → Hosted on Azure App Services or IIS Server.
- Backend (ASP.NET Core with C#) → Deployed on Azure or an on-premise Windows Server.
- Database (SQL Server) → Hosted on Azure SQL Database or a dedicated on-premise SQL Server.

## Deployment Pipelines:

Pipeline Steps:

Code Commit – Developers push code changes to GitHub / Azure DevOps.

Build & Test – Automated build & unit tests using Azure DevOps Pipelines.

Deployment – Application is deployed to Azure / IIS Server.

Monitoring – Logs & performance checks using Azure Monitor.

---

## Scaling Considerations:

Load Balancing – Distributes traffic between multiple backend servers.

Database Optimization – SQL Server is indexed and optimized for performance.

Caching Mechanisms – Stores frequently accessed data to reduce load times.

Failover Strategy – Backup system in place for disaster recovery.