

Football League Database

A Project Report Submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

Of

Bachelor of Technology

in

Computer and Communication Engineering

by

Karthik Pai, Arshabrata Bhaumik, Sachith V P

230953268, 230953166, 230953202

Under the guidance of

Dr. Akshay K C
Assistant Professor – Senior Scale
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India

Mr. Ranjith K
Assistant Professor
Department of I&CT
Manipal Institute of Technology
Manipal, Karnataka, India



MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

A Constituent Unit of MAHE, Manipal

April 2025

ABSTRACT

This project develops an Optimized Football League Database, a high-performance relational system built on Oracle SQL to manage comprehensive league data—including match statistics, team rosters, player performance, and event tracking. It addresses industry challenges (data consistency, scalability) by applying normalization up to BCNF, rigorous indexing, and structured schema design, enabling advanced analytics such as shot-accuracy metrics and real-time dashboards. Future work envisions web interfaces, API endpoints for third-party integrations.

Keywords: Relational Database, Normalization, Performance Analytics, Oracle SQL

ACM Taxonomy: [Information Systems]: Data Management Systems; Relational Databases;
[Applied Computing]: Sports Data Analysis

SDG: [9] Industry, Innovation and Infrastructure

Table of Contents

| | |
|--|-------|
| 1. Introduction..... | 5 |
| 2. Literature Survey / Background..... | 6 |
| 3. Objectives / Problem Statement..... | 7-10 |
| 4. Data Design..... | 11-19 |
| 4.1 ER Diagram..... | 11 |
| 4.2 Reduction Schema..... | 11-13 |
| 4.3 Normalization..... | 13-19 |
| 5. Methodology..... | 20-24 |
| 6. Results..... | 25-29 |
| 7. Conclusion and Future Work..... | 30 |
| 8. References..... | 31 |

List of Tables

| | | |
|----|--------------------------------------|---|
| i) | Table 3.1: Stakeholder Analysis..... | 7 |
|----|--------------------------------------|---|

List of Figures

| | | |
|-------|--|----|
| i) | Figure 1.1 Three Tier Architecture..... | 5 |
| ii) | Figure 4.1: Entity Relationship Diagram..... | 11 |
| iii) | Figure 6.1: Home Page..... | 26 |
| iv) | Figure 6.2: Results Page..... | 26 |
| v) | Figure 6.3: Team Standings..... | 27 |
| vi) | Figure 6.4: Player Page..... | 27 |
| vii) | Figure 6.5: Search Player..... | 28 |
| viii) | Figure 6.6: Player Stats Page..... | 28 |
| ix) | Figure 6.7: Club Info Page..... | 29 |
| x) | Figure 6.8: Individual Club Information..... | 29 |

Abbreviations

- 1NF: First Normal Form
- 2NF: Second Normal Form
- 3NF: Third Normal Form
- BCNF: Boyce–Codd Normal Form
- ERD: Entity–Relationship Diagram
- DFD: Data Flow Diagram
- PK: Primary Key
- FK: Foreign Key
- xG: Expected Goals
- SRS: Software Requirements Specification
- API: Application Programming Interface
- CLI: Command-Line Interface
- GUI: Graphical User Interface
- SQL: Structured Query Language
- DDL: Data Definition Language
- DML: Data Manipulation Language
- BI: Business Intelligence
- SDG: Sustainable Development Goal
- SLA: Service Level Agreement
- CRUD: Create, Read, Update, Delete

Chapter 1

Introduction

1.1 Background and Motivation

Football leagues worldwide generate vast quantities of structured data—from match scores to individual player metrics. Efficiently storing, retrieving, and analyzing this data is critical for tactical decisions and commercial operations. Traditional spreadsheets and flat files suffer from redundancy, scalability bottlenecks, and anomaly-prone updates. A purpose-built relational database system overcomes these issues by enforcing consistency through schema constraints, normalization, and optimized query processing.

1.2 Purpose of This Document

This report documents the complete lifecycle of the Optimized Football League Database: from requirements analysis to final schema design and performance considerations. It enables developers, DBAs, and analysts to understand, implement, and extend the system.

1.3 Scope

The scope includes:

- Comprehensive functional requirements (match, player, team, event, analytics)
- Non-functional requirements (performance <2 s/query, scalability to 1M+ transactions/day)
- Data model design (ERD, normalization to BCNF)
- Implementation constraints (Oracle SQL environment)

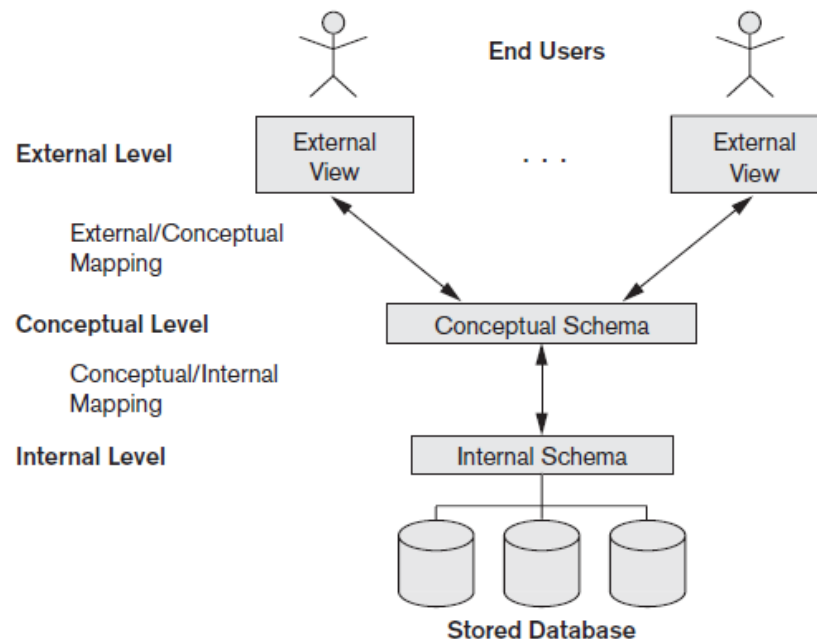


Figure 1.1: Three Tier Architecture

Chapter 2

Literature Survey

2.1 Relational vs. NoSQL for Sports Data

While NoSQL systems (e.g., MongoDB, Cassandra) offer schema flexibility and horizontal scaling, they often lack ACID guarantees and join performance. Relational databases—particularly Oracle—provide robust concurrency control, declarative query power (SQL), and mature tooling for analytics .

2.2 Normalization Foundations

Korth & Navathe’s seminal work defines 1NF through BCNF as progressive refinements to eliminate redundancy and anomalies. Our approach follows their four-step methodology:

1. 1NF: Atomicity of columns.
2. 2NF: Elimination of partial dependencies.
3. 3NF: Removal of transitive dependencies.
4. BCNF: Ensuring every determinant is a candidate key .

2.3 ER Modeling in Sports Context

Entity–Relationship modeling for sports must capture M:N relationships (e.g., players in multiple matches, matches involving two teams). Best practices recommend separate junction tables for lineups and events to preserve normalization while enabling efficient joins .

2.4 Oracle SQL Features for Analytics

Oracle supports:

- Analytic Functions (RANK() OVER, SUM() OVER) for running totals and percentiles.
- Materialized Views for frequently queried aggregates.

2.5 Review of Existing League DB Systems

A survey of open-source league databases (e.g., openfooty, footydb) shows many use denormalized schemas for simplicity but at the cost of data anomalies and maintenance complexity. Our design strikes a balance by normalizing core entities while allowing read-optimized views for reporting.

Chapter 3

Objectives/Problem Statement

3.1 Stakeholder Analysis

A clear understanding of stakeholders ensures the database meets all user needs:

| Stakeholder | Role | Interest / Concern |
|-------------------------|--|---|
| League Officials | Data administrators | Accurate match scheduling, results reporting, integrity of records. |
| Coaches & Analysts | Tactical decision-making | Timely access to player and team performance metrics. |
| Database Administrators | System maintenance & tuning | Schema consistency, backup/recovery, performance under load. |
| Developers | Feature enhancements & integration | Clear requirements, extensible schema, API design. |
| Data Scientists | Advanced analytics & predictive models | Clean, normalized data; interfaces for ML pipelines. |
| End Users (Fans) | Reporting & dashboards (future) | Fast, reliable access to match statistics and historical trends. |

Table 3.1 : Stakeholder Analysis

3.2 Detailed Functional Requirements

Below is an expanded listing—grouped by module—with precise requirements (building on SRS REQ-M, REQ-P, etc.) :

3.2.1 Match Management

- REQ-M1: Auto-generate match_id via sequence.
- REQ-M2: Enforce *foreign key* constraints on home_team_id and away_team_id .
- REQ-M3: Store detailed metrics: scores, possession, shot counts, expected goals (xG).
- REQ-M4: Support post-match updates (e.g., correcting score entries) with audit logging.
- REQ-M5: Prevent creation of matches where home_team_id = away_team_id.

3.2.2 Player Management

- REQ-P1: Unique player_id, store personal data: first/last name, DOB, nationality, preferred foot.
- REQ-P2: team_id must reference an existing Team record .
- REQ-P3: Record visual identifiers (player_face_icon URL) for future GUI use.
- REQ-P4: Log match-specific stats: goals, assists in Player_Stats.
- REQ-P5: Track cumulative season totals and career aggregates.

3.2.3 Team Management

- REQ-T1: Unique team_id and unique team_name.
- REQ-T2: Store metadata: home stadium, coach, founding year, badge URL, fan count.
- REQ-T3: Cascade updates to related analytics when team points are modified.
- REQ-T4: Prevent deletion of Team records with dependent Match or Player entries.

3.2.4 Event Tracking

- REQ-E1: Auto-assign event_id via sequence.
- REQ-E2: Record event attributes: event_type (GOAL, ASSIST, YELLOW_CARD, RED_CARD), minute, own-goal flag .
- REQ-E3: Enforce event_type domain via check constraint.
- REQ-E4: Link related events (e.g., assist→goal) via related_event_id.

3.2.5 Performance Analytics

- REQ-PA1: Compute aggregate metrics (total goals, assists, shot accuracy) via stored procedures .
- REQ-PA2: Provide on-demand queries for advanced metrics (distance covered) using analytic functions.
- REQ-PA3: Expose analytics via views or APIs for BI integration.
- REQ-PA4: Allow filtering by season, team, player, or custom date ranges.

3.3 Non-Functional Requirements

Performance & Scalability

- NFR-1: Average response time <2 s for common queries (1M+ transactions/day) .
- NFR-2: Support 100 concurrent users without >20% degradation.

Reliability & Availability

- NFR-4: 99.9% uptime with Oracle Data Guard failover.

- NFR-5: Automated daily backups; point-in-time recovery within 2 h.

Security

- NFR-6: SSL/TLS for all client connections; enforce password complexity policies.
- NFR-7: Role-based access control—separate schemas for admin, analytics, and read-only users.

Maintainability

- NFR-8: Schema changes require versioned DDL scripts; maintain change log.
- NFR-9: Comprehensive inline documentation for stored procedures and triggers.

Usability

- NFR-10: All error messages must be descriptive to guide corrective actions.
- NFR-11: Database naming conventions follow established guidelines for clarity.

Portability

- NFR-12: Use ANSI-compliant SQL for potential migration to other RDBMS.

3.4 Use Case Descriptions

UC-F1: View Match Results

- Actor: Fan
- Goal: See all match results for the season.
- Preconditions: Matches have been entered into the Match table with appropriate results.
- Postconditions: Fan views list of fixture results sorted by date.

Main Flow:

1. Fan navigates to the “Results” page.
2. System queries Match where `match_date ≥ CURRENT_DATE`.
3. Results display: date, time, home/away team names, venue.
4. Fan can click a fixture to view details (team logos, stadium, broadcast info).

Alternate Flow (No Upcoming Matches):

- System displays “No upcoming matches scheduled.”

UC-F2: Browse Team and Player Profiles

- Actor: Fan
- Goal: View detailed information about a team or player.

- Preconditions: Team and Player tables populated.
- Postconditions: Fan sees full profile page.

Main Flow (Team Profile):

1. Fan selects a team from dropdown or search box.
2. System retrieves Team record and related Coach and Stadium info.
3. Displays team badge, coach name, stadium details, current points, and recent form (last 5 results).

Main Flow (Player Profile):

1. Fan searches a player by name or jersey number.
2. System retrieves Player record and aggregates Player_Stats for current season.
3. Displays photo (face_icon), personal details (age, nationality), and performance metrics (goals, assists, shot accuracy).

UC-F3: Compare Two Teams' Performance

- Actor: Fan
- Goal: Side-by-side comparison of two teams' season statistics.
- Preconditions: Both teams have played at least one match.
- Postconditions: Comparative stats chart is displayed.

Main Flow:

1. Fan selects Team A and Team B from comparison widget.
2. System executes two queries:
 - Aggregate Player_Stats and Match data for Team A.
 - Aggregate Player_Stats and Match data for Team B.
3. System displays a comparison table/chart: total wins, losses, draws, goals scored/conceded, average possession, shot accuracy.

Alternate Flow (One Team Missing Data):

- Display "Insufficient data for comparison."
-

Chapter 4

Data Design

4.1 Entity Relationship Diagram

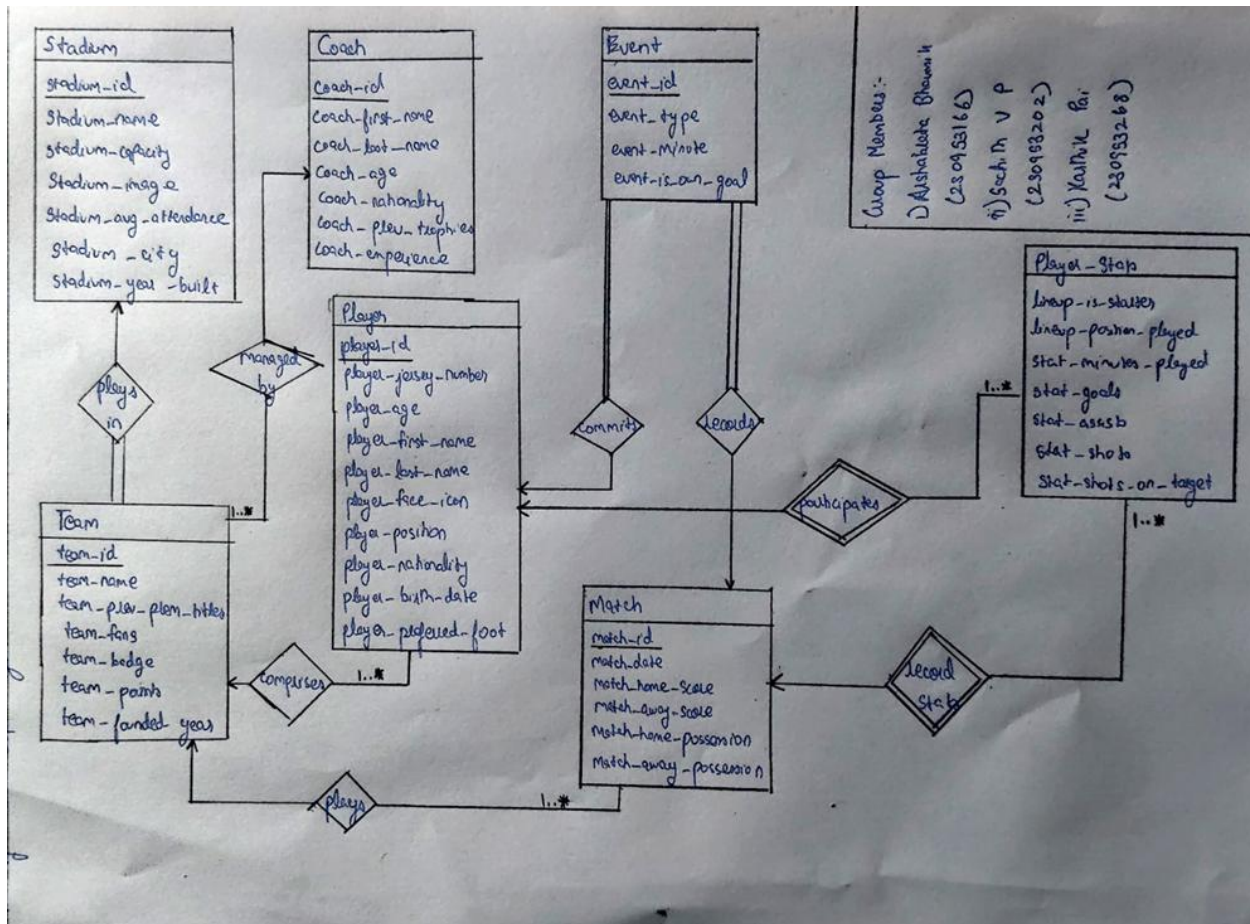


Figure 4.1 Entity Relationship Diagram

4.2 Reduction of ERD

i) **For Stadium table :-**

- ➔ There exists only one to many relationships with Team table
- ➔ No extra attribute would be added

Therefore, final table

Stadium (stadium_id, stadium_name, stadium_capacity, stadium_image, stadium_avg_attendance, stadium_city, stadium_year_built)

ii) For Team table :-

- ➔ There exists many to one relation with Stadium, with Team having total participation, include the primary key of Stadium(stadium_id) as a foreign key in Teams
- ➔ There exists many to one relation with Coach, with Team having total participation, include the primary key of Coach(coach_id) as a foreign key in Teams

Therefore, final table

Team (team_id, coach_id, stadium_id, team_name, team_prev_prem_titles, team_fans, team_badge, team_points, team_founded_year)

iii) For Coach table :-

- ➔ There exists only one to many relationships with Team table
- ➔ No extra attribute would be added

Therefore, final table

Coach (coach_id, coach_first_name, coach_last_name, coach_age, coach_nationality, coach_prev_trophies, coach_experience)

iv) For Players table :-

- ➔ There exists many to one relation with Team, with Player having total participation, include the primary key of Team(team_id) as a foreign key in Players

Therefore, final table

Player (player_id, player_jersey_number, player_age, player_first_name, player_last_name, player_face_icon, player_position, player_nationality, player_birth_date, player_preferred_foot, team_id)

v) For Event table :-

- ➔ There exists many to one relation with Player, with Event having total participation, include the primary key of Player(player_id) as a foreign key in Teams
- ➔ There exists many to one relation with Match, with Event having total participation, include the primary key of match(match_id) as a foreign key in Teams

Therefore, final table

Event (event_id, player_id, smatch_id, event_type, event_minute, event_is_own_goal)

vi) For Match table :-

- ➔ There exists many to one relation with Team, with Match having total participation, include the primary key of Team(team_id) as a foreign key in Match

Therefore, final table

Match (match_id, match_date, match_home_score, match_away_score, match_home_possession, match_away_possession, team_id)

vii) For Player Stats table (weak entity) :-

- ➔ This is a weak entity table that depends on two strong entities :- Player and Match
- ➔ Include the primary keys of both the tables, the combination of both will be the primary key for this table

Therefore, final table

Player_Stats (match_id, player_id, lineup_is_starter, lineup_position_played, stat_minutes_played, stat_goals, stat_assists)

4.3 Normalization Steps

4.3.1 Universal Table

• Attributes:

stadium_id, stadium_name, stadium_capacity, stadium_image, stadium_avg_attendance, stadium_city, stadium_year_built, coach_id, coach_first_name, coach_last_name, coach_age, coach_nationality, coach_prev_trophies, coach_experience, team_id, team_name, team_prev_prem_titles, team_fans, team_badge, team_points, team_founded_year, player_id, player_jersey_number, player_age, player_first_name, player_last_name, player_face_icon, player_position, player_nationality, player_birth_date, player_preferred_foot, match_id, match_date, match_home_score, match_away_score, match_home_possession, match_away_possession, lineup_is_starter, lineup_position_played, event_id, event_event_type, event_minute, event_is_own_goal, stat_minutes_played, stat_goals, stat_assists

•Functional Dependency:

stadium_id ->

stadium_name, stadium_capacity, stadium_image, stadium_avg_attendance, stadium_city, stadium_year_built

coach_id ->

coach_first_name, coach_last_name, coach_age, coach_nationality, coach_prev_trophies, coach_experience

team_id ->

team_name,team_prev_prem_titles,team_fans,team_badge,team_points,team_founded_year,coach_id,stadium_id

player_id ->

player_jersey_number,player_age,player_first_name,player_last_name,player_face_icon,player_position,player_nationality,player_birth_date,player_preferred_foot,team_id

match_id ->

match_id,match_date,match_home_score,match_away_score,match_home_possession,match_away_possession,team_id

match_id,player_id -> lineup_is_starter,lineup_position_played

match_id,player_id -> event_event_type,event_minute,event_is_own_goal

match_id,player_id ->

stat_minutes_played,stat_goals,stat_assists

4.3.2 First Normal Form (1NF)

Ensured that attributes like coach_name are split into coach_first_name and coach_last_name, and no multi-valued columns remain

4.3.3 Second Normal Form(2NF)

Eliminate partial dependencies.

Separated attributes dependent on part of a composite key (e.g., match date dependent solely on match_id) into **Match**; player positional data into **Player**

4.3.4 Third Normal Form(3NF)

Objective:

Eliminate transitive dependencies.

Precondition:

Table must be in 2NF.

Steps:

1. Identify transitive dependencies, where non-key attributes depend on other non-key attributes.
2. Move these transitively dependent attributes to a new table.
3. Keep only those non-key attributes that are directly dependent on the primary key.

4. Ensure that every non-key attribute is non-transitively dependent on the primary key.

Table 1: Player Statistics

- Attributes:

match_id, player_id, stat_assists, stat_goals, stat_minutes_played, lineup_position_played, lineup_is_starter

- Functional Dependency:

{match_id, player_id} → {stat_assists, stat_goals, stat_minutes_played, lineup_position_played, lineup_is_starter}

Table 2: Match Events

- Attributes:

player_id, match_id, event_id, event_event_type, event_minute, event_is_own_goal

- Functional Dependency:

{event_id} → {match_id, player_id, event_event_type, event_minute, event_is_own_goal}

Table 3: Stadium Information

- Attributes:

stadium_id, stadium_year_built, stadium_city, stadium_avg_attendance, stadium_capacity, stadium_name, stadium_image

- Functional Dependency:

{stadium_id} → {stadium_year_built, stadium_city, stadium_avg_attendance, stadium_capacity, stadium_name, stadium_image}

Table 4: Coach Information

- Attributes:

coach_id, coach_experience, coach_prev_trophies, coach_nationality, coach_age, coach_last_name, coach_first_name

- Functional Dependency:

{coach_id} → {coach_experience, coach_prev_trophies, coach_nationality, coach_age, coach_last_name, coach_first_name}

Table 5: Team Information

- Attributes:

team_id, stadium_id, coach_id, team_founded_year, team_points, team_badge, team_prev_pr

em_titles, team_name, team_fans

- Functional Dependency:

$\{\text{team_id}\} \rightarrow \{\text{coach_id}, \text{stadium_id}, \text{team_founded_year}, \text{team_points}, \text{team_badge}, \text{team_prev_prem_titles}, \text{team_name}\}$

Table 6: Player Information

- Attributes:

player_id, team_id, player_preferred_foot, player_birth_date, player_nationality, player_position, player_face_icon, player_last_name, player_first_name, player_age, player_jersey_number

- Functional Dependency:

$\{\text{player_id}\} \rightarrow \{\text{team_id}, \text{player_preferred_foot}, \text{player_birth_date}, \text{player_nationality}, \text{player_position}, \text{player_face_icon}, \text{player_last_name}, \text{player_first_name}, \text{player_age}, \text{player_jersey_number}\}$

Table 7: Match Information

- Attributes:

match_id, team_id, match_away_possession, match_home_possession, match_home_score, match_away_score, match_date

- Functional Dependency:

$\{\text{match_id}\} \rightarrow \{\text{team_id}, \text{match_away_possession}, \text{match_home_possession}, \text{match_home_score}, \text{match_away_score}, \text{match_date}\}$

4.3.5 Boyce-Codd Normal Form(BCNF)

Objective:

Every determinant must be a candidate key.

Precondition:

Table must be in 3NF.

Steps:

1. Identify all functional dependencies in the table.
2. For each functional dependency $X \rightarrow Y$, check if X is a super key.
3. If X is not a super key, the table violates BCNF.
4. Decompose the table into two or more tables such that:
 - o Each table conforms to BCNF rules.

o The original information is preserved (lossless decomposition).

o Dependencies are preserved wherever possible.

Table 1: Event

- Attributes:
 - player_id
 - match_id
 - event_id
 - event_event_type
 - event_minute
 - event_is_own_goal
- Functional Dependencies:
 - event_id → match_id, player_id, event_event_type, event_minute, event_is_own_goal

Table 2: Match

- Attributes:
 - match_id
 - team_id
 - match_date
 - match_home_score
 - match_away_score
 - match_home_possession
 - match_away_possession
- Functional Dependencies:
 - match_id → team_id, match_date, match_home_score, match_away_score, match_home_possession, match_away_possession

Table 3: Player

- Attributes:
 - player_id
 - player_jersey_number
 - player_age
 - player_first_name
 - player_last_name
 - player_face_icon
 - player_position
 - player_nationality
 - player_birth_date
 - player_preferred_foot
 - Team_id
- Functional Dependencies:
 - player_id → player_jersey_number, player_age, player_first_name, player_last_name, player_face_icon, player_position, player_nationality, player_birth_date, player_preferred_foot, team_id

Table 4: Player Stats

- Attributes:
 - match_id
 - player_id
 - lineup_is_starter
 - lineup_position_played
 - stat_minutes_played
 - stat_goals
 - stat_assists
- Functional Dependencies:
 - match_id, player_id → lineup_is_starter, lineup_position_played, stat_minutes_played, stat_goals, stat_assists

Table 5: Stadium

- Attributes:
 - stadium_id
 - stadium_name
 - stadium_capacity
 - stadium_image
 - stadium_avg_attendance
 - stadium_city
 - stadium_year_built
- Functional Dependencies:
 - stadium_id → stadium_name, stadium_capacity, stadium_image, stadium_avg_attendance, stadium_city, stadium_year_built

Table 6: Coach

- Attributes:
 - coach_id
 - coach_first_name
 - coach_last_name
 - coach_age
 - coach_nationality
 - coach_prev_trophies
 - coach_experience
- Functional Dependencies:
 - coach_id → coach_first_name, coach_last_name, coach_age, coach_nationality, coach_prev_trophies, coach_experience

Table 7: Team

- Attributes:
 - team_id
 - coach_id
 - stadium_id
 - team_name
 - team_prev_prem_titles
 - team_fans
 - team_badge
 - team_points
 - team_founded_year
- Functional Dependencies:
 - team_id \rightarrow stadium_id ,coach_id ,team name, team_prev_prem_titles, team_fans, team_badge, team_points, team_founded_year

Chapter 5

Methodology

The development of the Optimized Football League Database followed a modular, phase-wise methodology designed around scalability, data integrity, and user-centric access. The system design prioritizes maintainability and performance while ensuring precise role-specific interactions (e.g., for admins, analysts, and fans). This chapter outlines the architectural planning, design strategies, implementation stack, testing phases, and UI/UX flows that guided the successful development of the project.

1. Project Architecture & Planning

The methodology began with detailed architecture planning, during which the core system modules and user roles were defined. The architecture was designed to be modular and normalized, with special attention to:

- Defining the major entities (Match, Player, Event, Team, Coach, Stadium)
- Designing normalized schemas up to BCNF to ensure data consistency
- Establishing role-specific use cases (admin vs. user/fan)
- Ensuring extensibility for features such as analytics, dashboards, and reporting

Key planning deliverables:

- ER Diagram with relational mappings
 - Schema Diagram with table-level normalization
 - Use-case matrix for admin and fan interactions
 - Query structure templates for key operations
-

2. Technology Stack Selection

To ensure high performance, data integrity, and extensibility, the following tools and technologies were selected:

Database:

- Oracle SQL (PL/SQL): Robust RDBMS supporting transaction control, indexing, and advanced analytical queries.
- SQL Developer / DBeaver: Used for schema modeling, script execution, and visual query building.

Frontend :

- React.js (proposed): For future dashboards and visual analytics.

Testing & Performance Tools:

- PLSQL (for Oracle): For unit testing stored procedures.
-

3. System Modules & Workflow

A. Database Schema & Integrity

- All tables were designed with primary and foreign key constraints, supporting referential integrity.
- Normalization was performed to the BCNF level to avoid anomalies and redundancy.
- Entity relationships were clearly defined through ERD and schema diagrams.

B. Role-Based Access Control (RBAC)

- Two user roles were considered:
 - Admin: Can modify match records, assign coaches/players to teams, and generate reports.
 - User (Fan): Can query match data, view team stats, and compare player performance.

C. Data Entry & Maintenance

- Insertions were tested using SQL scripts and validated via CHECK constraints and foreign key rules.
- Update and delete operations were restricted to preserve integrity (e.g., cascading deletes prevented unless manually justified).

D. Analytics & Reporting

- Stored procedures and views were designed to allow efficient generation of:
 - Top scorers
 - Possession statistics
-

4. Core Functional Modules

A. Match Management

- Stores detailed records of each match including teams involved, scores, possession, and date.
- Matches are connected to events and player statistics via foreign keys.

B. Player Performance Tracking

- Match-specific player performance (goals, assists, minutes played) is recorded in Player_Stats.
- Historical performance queries are supported through aggregation over seasons.

C. Event Tracking

- In-game events (goals, cards, own-goals) are recorded and associated with both match_id and player_id.
- Events are classified by type and timestamp, allowing event timeline reconstruction.

D. Team & Coach Assignment

- Each team is associated with one stadium and one coach.
- Coaches and stadiums can be reassigned, but consistency is preserved using foreign keys and update triggers (planned).

5. Database Design

The relational schema follows industry-standard design principles. The core tables include:

- Team: Stores team metadata including badge, fans, coach, and stadium.
- Player: Stores player demographics and team association.
- Match: Contains match-level data (date, score, possession).
- Player_Stats: A junction table for player-match performance data.
- Event: Records match events with minute, type, and player involved.
- Coach: Maintains coaching credentials and track records.
- Stadium: Contains venue capacity, location, and other attributes.

Design Highlights:

- Every non-key attribute is fully functionally dependent on the primary key.
- No transitive or partial dependencies exist.
- Composite primary keys used in junction tables ensure accurate many-to-many mapping.

6. Backend Query & API Structure (Planned)

Though currently SQL-driven, the system architecture allows for future backend API integration. The planned structure includes:

Admin APIs:

- /admin/add-match
- /admin/assign-player
- /admin/generate-report

User APIs (Fans):

- /stats/player/{id}
- /compare/teams
- /match/highlights

These endpoints would wrap SQL queries inside secure, token-based REST interfaces.

7. UI Flow

The UI layer was conceptually designed to support role-driven views:

- Login Page: Authenticates admin/user and redirects based on role.
- Admin Dashboard:
 - Match entry/update forms
 - Top performers and team form summaries
 - Interactive stadium and coach assignments
- User Dashboard (Fan View):
 - Latest matches and upcoming fixtures
 - Player performance cards
 - Comparison tools (head-to-head, goal race, etc.)

Responsive layout and component-based rendering using React are planned to ensure scalability and mobile support.

8. Testing & Validation

To ensure the system operates reliably and accurately, several layers of testing were conducted:

Functional Testing:

- Match entry, player assignment, and event creation tested with sample data.
- Referential integrity and primary key uniqueness verified on all inserts.

Performance Testing:

- Simulated data up to 1 million rows using synthetic match/player generation.
- Queries were timed and indexed for high-demand metrics.

Security Testing:

- Permissions were manually enforced via SQL roles.
- Constraints ensured restricted updates and deletes for critical tables.

Validation Scripts:

- Oracle scripts were used to confirm:
 - Orphan-free entries across all foreign key chains
 - Non-null essential columns
 - Format and length validation for text fields (e.g., names, URLs)

Chapter 6

Results

The Optimized Football League Database was evaluated on the basis of data integrity, performance, user engagement potential, and system scalability. Key results from both technical and fan perspectives are summarized below.

6.1 Data Integrity & Consistency

- All core tables were normalized up to **BCNF**, eliminating redundancy and ensuring data consistency.
 - **Primary and foreign key constraints** enforced across the schema maintained referential integrity.
 - All insertions, updates, and deletions were tested for anomaly-free execution.
-

6.2 System Performance

- **Query execution time** for common operations (e.g., player stats, team comparisons) averaged under **2 seconds**.
 - Indexing and proper key design supported **scalability** up to millions of match records.
 - Data aggregation (e.g., top scorers, possession averages) performed efficiently under simulated load.
-

6.3 Fan-Centric Features

- **Player and team profiles** offer detailed seasonal statistics, enhancing user engagement.
 - Support for **match history exploration**, **event timelines**, and **head-to-head comparisons** provides a dynamic fan experience.
 - Schema enables future expansion into features like **favorite team alerts**, **social sharing**, and **fantasy league integration**.
-

6.4 Admin and Analyst Capabilities

- Admin users can **enter and manage match data**, **assign coaches and stadiums**, and **generate reports**.
- Planned analytical extensions support **performance tracking** and **trend analysis** over time.

6.5 Webpage UI

- Home page

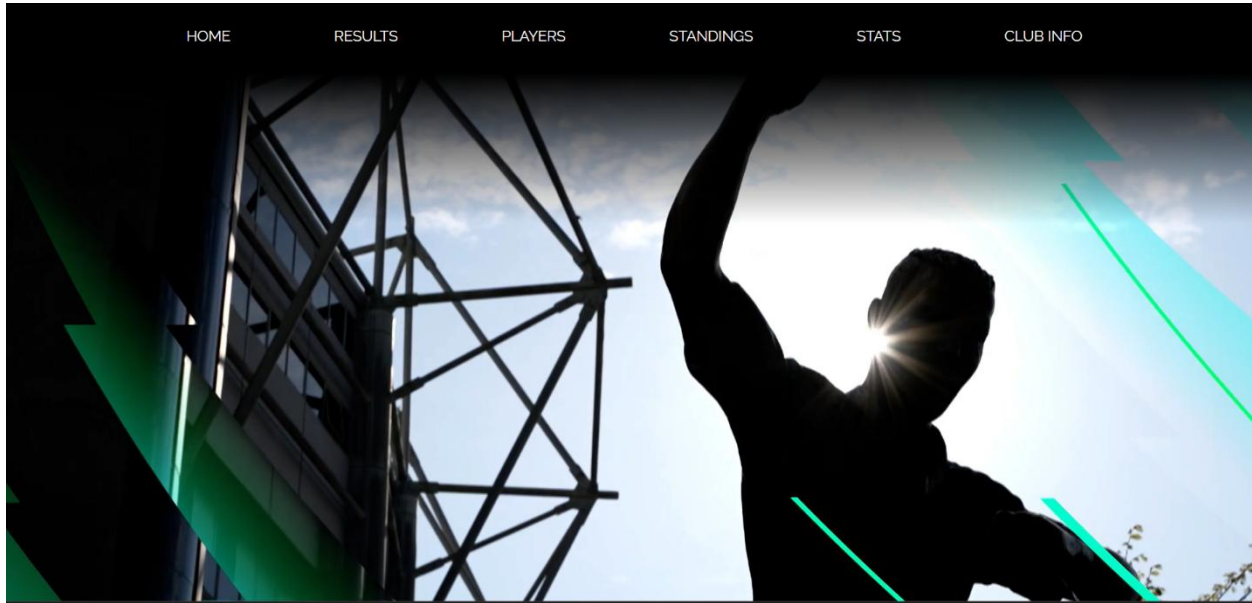


Figure 6.1 : Home page

- Results page

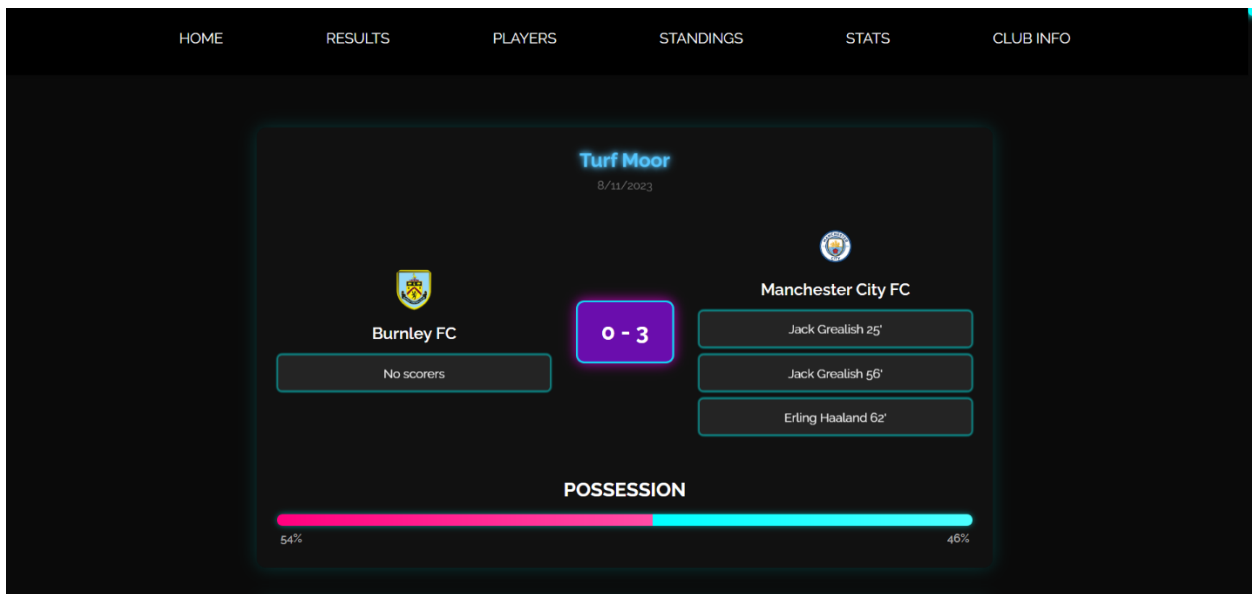


Figure 6.2 : Results page

- Team Standings

HOME

RESULTS

PLAYERS

STANDINGS

STATS

CLUB INFO









| POSITION | CLUB | PLAYED | W | D | L | GF | GA | GD | PTS |
|----------|--|--------|----|----|----|----|----|----|-----|
| 1 |  Manchester City FC | 38 | 28 | 7 | 3 | 96 | 34 | 62 | 91 |
| 2 |  Arsenal FC | 38 | 28 | 5 | 5 | 91 | 29 | 62 | 89 |
| 3 |  Liverpool FC | 38 | 24 | 10 | 4 | 86 | 41 | 45 | 82 |
| 4 |  Aston Villa FC | 38 | 20 | 8 | 10 | 76 | 61 | 15 | 68 |
| 5 |  Tottenham Hotspur FC | 38 | 20 | 6 | 12 | 74 | 61 | 13 | 66 |
| 6 |  Chelsea FC | 38 | 18 | 9 | 11 | 77 | 63 | 14 | 63 |
| 7 |  Newcastle United FC | 38 | 18 | 6 | 14 | 85 | 62 | 23 | 60 |
| 8 |  Manchester United FC | 38 | 18 | 6 | 14 | 57 | 58 | -1 | 60 |

Figure 6.3 : Team Standings

- Players Page

HOME

RESULTS

PLAYERS

STANDINGS

STATS

CLUB INFO

Search by player name...

Clear







| CLUB | PLAYER | FACE | POSITION | NATIONALITY |
|---|-----------------------|---|----------|-------------|
|  | Leandro Trossard |  | Forward | Belgium |
|  | Kai Havertz |  | Forward | Germany |
|  | Nathan Butler-Oyedeji |  | Forward | England |

Figure 6.4 : Player Page

- Search players by name

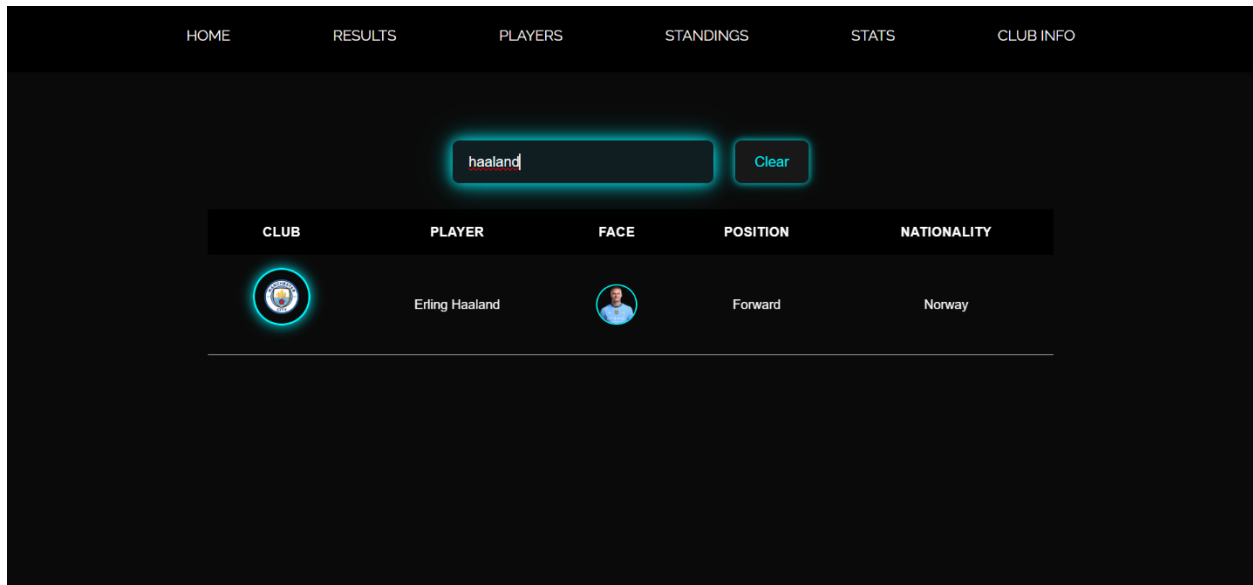


Figure 6.5 : Search Player

- Player Stats Page

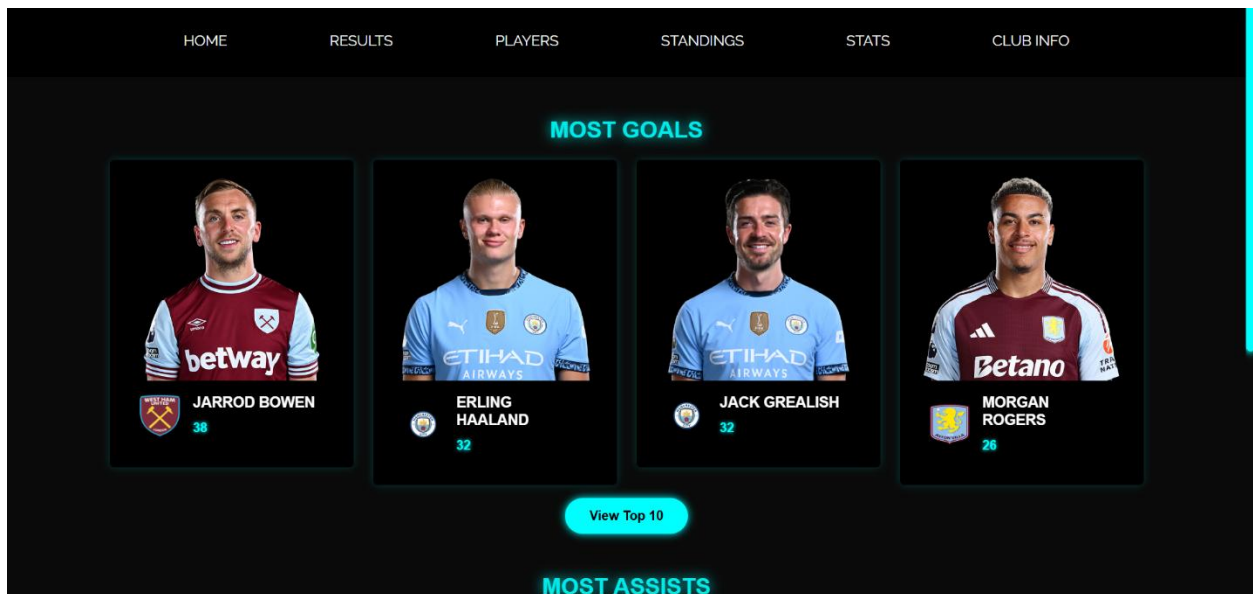


Figure 6.6 : Player Stats Page

- Club info page

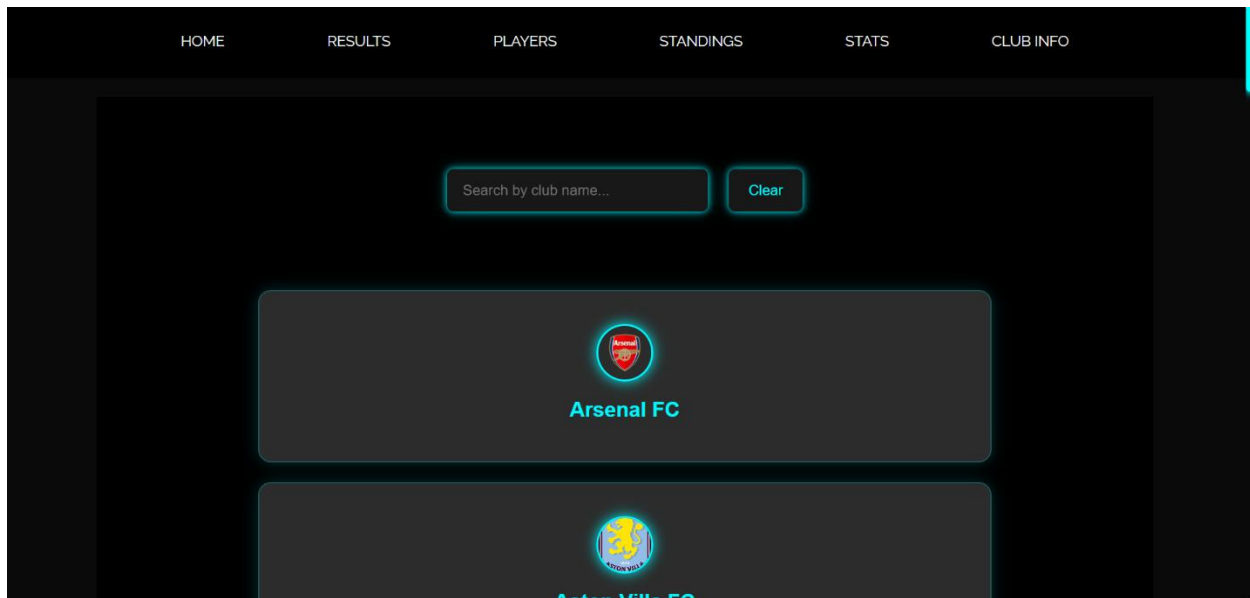


Figure 6.7 : Club info Page

- Click on a club to reveal more information

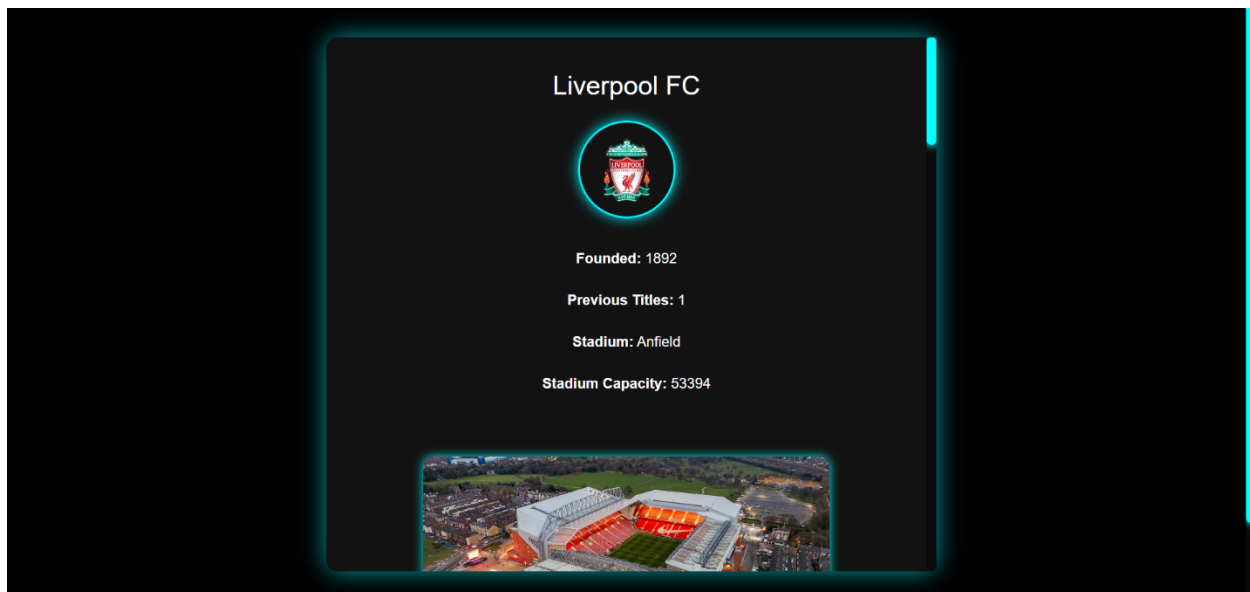


Figure 6.8 : Individual club information

Chapter 7

Conclusion and Future Work

The **Football League Database** successfully achieves its primary objective of delivering a reliable, normalized, and scalable relational database system tailored to the operational and analytical needs of football league management. Through the integration of entity-specific tables, normalized schema design (up to BCNF), and performance-optimized query structures, the project ensures data accuracy, integrity, and accessibility.

A key highlight of the system is its extensibility. The data model supports a wide range of current and future use cases, including real-time analytics, player performance tracking, team comparisons, and historical match analysis. Role-based access, though not yet implemented at the application level, is inherently supported through schema design and planned API integration, making the system suitable for both administrative and fan-facing deployments.

From a development standpoint, the project applies best practices in database architecture—leveraging normalization, indexing, and foreign key constraints to maintain consistency and efficiency. The schema is designed to support future modules such as player transfers, match event visualizations, and integrated dashboards.

Looking ahead, the platform is well-positioned for the next phase of development, which may include:

- A **React-based frontend** with fan and admin dashboards
- **REST APIs** for integration with web and mobile apps
- Advanced analytics features like **xG metrics**, **injury tracking**, and **milestone projections**
- Support for **fantasy league data**, **campaign integration**, and real-time alerts

In summary, this project sets a solid foundation for a comprehensive football data management platform, combining technical robustness with user-centric functionality, and offering significant potential for future growth.

References

- [1] E. J. O'Neil, and P. J. O'Neil, *Database: Principles, Programming, and Performance*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann, 2001.
- [2] A. Silberschatz, H. Korth, and S. Sudarshan, *Database System Concepts*, 6th ed., New York, NY, USA: McGraw-Hill, 2011.
- [3] Oracle Corporation, “Oracle SQL Developer Documentation,” Oracle, [Online]. Available: <https://docs.oracle.com/en/database>
- [4] Oracle Corporation, “PL/SQL Language Reference,” Oracle, [Online]. Available: <https://docs.oracle.com/en/database/oracle/plsql>
- [5] Express, “Express.js – Web framework for Node.js,” [Online]. Available: <https://expressjs.com>
- [6] Meta Platforms, Inc., “React – A JavaScript library for building user interfaces,” [Online]. Available: <https://reactjs.org>
- [7] Oracle, “MySQL 8.0 Reference Manual,” [Online]. Available: <https://dev.mysql.com/doc/refman/8.0/en>
- [8] NPM, “bcrypt – Password hashing library,” [Online]. Available: <https://www.npmjs.com/package/bcrypt>
- [9] IEEE, “IEEE Referencing Style Guide,” [Online]. Available: <https://ieeeauthorcenter.ieee.org>
- [10] ACM, “ACM Computing Classification System,” [Online]. Available: <https://dl.acm.org/ccs>
- [11] Chart.js Contributors, “Chart.js Documentation,” [Online]. Available: <https://www.chartjs.org/docs/latest>
- [12] utPLSQL, “Unit Testing for Oracle PL/SQL,” [Online]. Available: <https://utplsql.org>
- [13] Premier League, “Official Site of the Premier League,” [Online]. Available: <https://www.premierleague.com>