High-Level Design- European League Football Gaming App

PREPARED FOR

Calmin Scarlett (Executive Sponsor)

VTI Lecturer & Associates

PREPARED BY

VTI Software Engineering Group 2023

Contents

1. Introduction

1.1. Why this High-Level Design Document?

This High-Level Design (HLD) Document serves as a blueprint for the high-level design of the Fantasy Football League application. It outlines the overall structure, functionality, and key considerations for the application's development.

1.2. Scope

The scope of this document encompasses the design aspects of the application, including its general description, design details, and various architectural considerations. It does not cover specific implementation details.

1.3. Definitions

- HLD: High Level Design
- FFL: Fantasy Football League, the name of the application being developed.
- Fantasy League: A virtual football league where users can create and manage their own teams, competing against other players based on real-life football statistics.
- AI: Artificial Intelligence, referring to computer-controlled opponents for players who haven't signed up or completed their registration.
- As

1.4. Overview

This HLD provides an overview of the high-level design of the FFL application.
It outlines the:

i) Product Perspective
ii) Tools Used
iii) General constraints
iv) Assumptions
v) Special Design Aspects
vi) Design details

, necessary for the successful implementation of the FFL application.

○

2. General Description

2.1. Product Perspective

Fantasy Football League is a mobile and web app that brings the world of football to life through a fantasy league based on the English Premier League. Players sign up for an account to join the league, It provides an immersive and realistic experience, simulating matches, managing teams, and tracking league standings.

2.2. Tools Used

The development of the application utilizes Java as the programming language, Spring or Java EE as web frameworks, HTML, CSS, and JavaScript for front-end development, and PostgreSQL as the database management system.

2.3. General Constraints

General constraints include scalability, security, maintenance, compatibility, performance optimisation, user feedback and support, legal and regulatory compliance, monetization and revenue generation, competitive landscape, and evolving user expectations.

2.4. Assumptions

This project is based on the model of Fantasy Football Competitions thus the assumptions include user availability, stable internet connection, availability of real-life football statistics, fair play and rule adherence, data accuracy, performance requirements, compliance with privacy regulations, third-party integrations, user engagement and retention, and commitment to continuous improvement.

2.5. Special Design Aspects

Special design aspects include implementing realistic match simulations, managing player substitutions, handling penalties, injuries, and cards, displaying match results, and utilizing algorithms to determine winners based on statistics and training.

3. Design Details

3.1. Main Design Features

The main design features include seven (7) major parts:

- **User Interface (UI):** The application should have an intuitive and user-friendly interface to facilitate easy navigation and interaction. This includes screens for various functionalities such as team registration, match schedules, player statistics, and league standings.

- **User Registration and Authentication**: Users should be able to create accounts and log in securely. Authentication mechanisms like email verification or two-factor authentication can enhance security.

- **Team Management:** Teams should be able to register and manage their information, including team name, logo, players, coaches, and other relevant details. This module may include features for adding, removing, and updating team information.

- **Fixture Management:** The application should provide the functionality to create and manage fixtures, including match dates, venues, and opponent teams. The system should be able to generate a schedule that considers factors like team availability and fairness.

- **Match Recording and Statistics**: The application should allow the recording and tracking of match results, including scores, goals, assists, yellow and red cards, substitutions, and other relevant statistics. This information can be used for generating player and team performance reports.

- **League Standings:** The system should calculate and display league standings based on match results. It should consider factors like points earned, goal differences, goals scored, and other tie-breaking criteria. Standings can be presented in a tabular format with filters and search options.

- **Player Profiles and Statistics:** The application should have player profiles that contain player information, such as name, age, position, and performance statistics. Users should be able to view individual player statistics and compare them with others.

- **Notifications and Communication:** The application can include features for sending notifications to teams and players regarding match schedules, results, and other important updates. It may also facilitate communication between teams, coaches, and league administrators.

- **Admin Panel:** An administration panel should be available to manage the overall system. This includes user management, content management, configuration settings, and access controls.

- **Integration with External Services:** The application may integrate with external services, such as payment gateways for team registration fees or third-party APIs for real-time score updates and news feeds.

- **User Registration and Authentication:** Users should be able to create accounts and log in securely to access the gaming application. Proper authentication mechanisms, such as email verification or social media login, should be implemented.

- **Team Creation and Management**: Users should be able to customize the team, manage players (add, remove, edit), and assign player roles and positions.

- **Match Gameplay:** The application should provide a simulated gameplay experience where users can compete against other teams in virtual football matches. The gameplay should include passing, shooting, dribbling, defending, and goalkeeping. Realistic physics and game mechanics can enhance the overall gaming experience.

- **Matchmaking:** The application should have a matchmaking system that pairs teams with similar skill levels for fair and competitive gameplay. The matchmaking algorithm should consider factors like team ratings, win-loss records, and player skill levels.

- **League Structure:** The gaming application can have a league structure where teams compete against each other in a series of matches. The league can have multiple divisions or tiers based on team performance, allowing teams to progress or be relegated accordingly. – Further Modeled off the English Premier League

- **Player Progression and Development:** Users should be able to improve their players' skills and attributes over time through training,

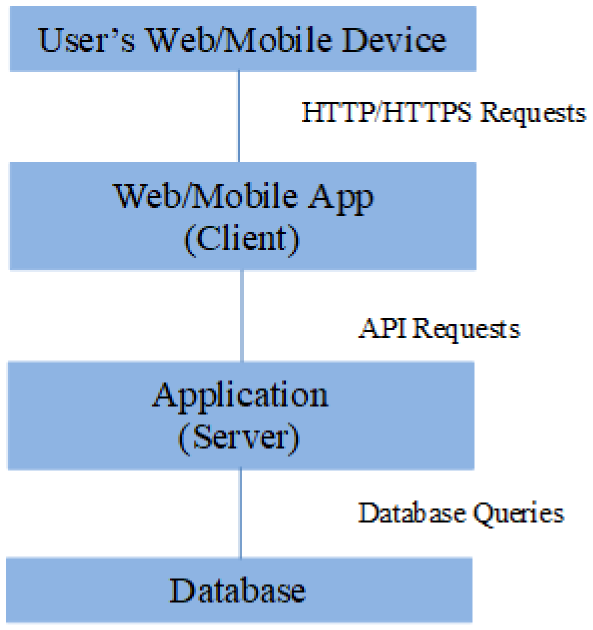
- **In-Game Currency and Rewards:** The application may include a virtual currency system that users can earn or purchase. Users can spend this currency to unlock new features, customize their teams, or acquire virtual items. Rewards can be given for winning matches, completing achievements, or participating in events.

- **Live Matches and Spectator Mode:** The application can provide the option for users to watch live matches between other teams. A spectator mode can allow users to observe ongoing matches, chat with other spectators, and learn from different playing styles.

- **Social Features:** Leaderboards and ranking systems can encourage healthy competition among users.

3.2. Application Architecture

The application follows a client-server architecture, where the client interacts with the server through API endpoints for various operations, such as user authentication, team management, and match simulations.



3.3. Technology Architecture

3.3.1. Web Application Architecture

The web application architecture consists of a presentation layer, data access layer, and business logic layer. It follows a model-view-controller (MVC) pattern to separate concerns and enhance maintainability.

3.3.2. Presentation Layer

Handles the user interface, allowing users to interact with the application. It utilizes HTML, CSS, and JavaScript for web interfaces and corresponding frameworks for mobile platforms.

3.3.3. Data Access Layer

The data access layer facilitates communication with the PostgreSQL database. It handles database operations, data retrieval, and data manipulation using Java Database Connectivity (JDBC) or an Object-Relational Mapping (ORM) framework.

3.3.4. Tools Used

See section 2.2 for tools used in the design of this project.

3.4. Standards

The application adheres to industry standards and best practices for software development, including coding conventions, naming conventions, and design patterns. It also complies with relevant security standards and data privacy regulations.

3.5. Database design

The design follows a relational model, utilizing tables to store user information, team data, match details, and league standings. Proper indexing and normalization techniques are employed to ensure efficient data retrieval and integrity.

3.6. Files

This application may involve file handling for storing user avatars, team logos, or other related media. The design includes file storage mechanisms and appropriate file formats to support these features.

3.7. User Interface

The user interface design focuses on providing a user-friendly and intuitive experience. It incorporates responsive design principles to ensure compatibility across different devices and screen sizes. The UI elements and layout are designed to support the desired functionality and enhance user engagement.

3.8. Reports

The application may include reporting features to generate league standings, player statistics, or other relevant data. The design includes appropriate reporting tools or libraries to facilitate the generation and presentation of reports.

3.9. Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything that falls outside the normal and intended usage.

3.10. Interfaces

The application may integrate with external systems or APIs for features such as real-time match updates, player statistics, or payment gateways (if applicable). The design includes the definition and specifications of these interfaces, ensuring seamless integration and data exchange.

3.11. Help

The application provides a help system or documentation to assist users in understanding its features and functionality. It includes user guides, tooltips, or contextual help within the user interface to provide assistance and address common queries.

3.12. Performance

Performance is going to be very important for this project. For everything to run smoothly for this project, the modules will have to be able to update data on the database and refresh the points table before it is supposed to do so again. This is likely to be the most processor-intensive aspect of the project. The database server will need to keep up with all database requests and transactions.

3.13. Security

Because security is not the prime focus of this project, only the minimal aspects of security will be implemented. A username and password will be required to log into an administrative interface and database. For now, all data will be sent in plain text. Verification of user to IP or MAC address is also outside the scope of this project. For now, there will also be no log of failed attempts of an administrator logging in. Caching, query optimization, and load testing are performed to identify and address performance bottlenecks.

3.14. Reliability

The application is designed to be reliable and available for users. Measures such as fault tolerance, data backups, and disaster recovery plans are implemented to minimize downtime and ensure uninterrupted access to the system.

3.15. Maintainability

The application is designed with maintainability in mind, allowing for easy bug fixes, enhancements, and future updates. It follows modular and well-structured coding practices, includes documentation, and incorporates version control to facilitate ongoing maintenance.

3.16. Portability

The application is designed to be platform-independent and compatible with different operating systems and web browsers. It adheres to web standards and avoids dependencies on specific hardware or software configurations.

3.17. Reusability

The design promotes code reusability by following modular and component-based architectures. Common functionalities and modules are identified, making them reusable across different parts of the application to improve development efficiency.

3.18. Application compatibility

The application is tested for compatibility across different devices, screen resolutions, and browsers to ensure a consistent user experience. Responsive design techniques and browser compatibility testing are employed to address compatibility challenges.

3.19. Resource utilization

The application is designed to efficiently utilize system resources such as memory, CPU, and network bandwidth. Techniques like resource pooling, optimization algorithms, and caching mechanisms are implemented to optimize resource usage.

3.20. Major Classes

The design identifies the major classes and their relationships within the application. This includes classes for user management, team management, match simulations, league standings, data access, and other key functionalities. The design ensures proper encapsulation, modularity, and extensibility through class hierarchies and inheritance where applicable.