**Software Requirements and Design Document**

**For**

**Group 8**

Version 1.0

**Authors**:

Megan Achurra

Dylan Giesler

Daniel Lukish

Miguel Perez Ojito

Thomas Rudolph

# Overview

*Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).*

Our project is a fantasy soccer website that is supposed to cater to newer players. It will have all the normal features expected from a fantasy sports site, such as drafts, trading, and of course a points system. Since we are attempting to also cater to newer players, we are adding in things like tool tips and suggestions that will make the system more accessible to those not familiar with fantasy sports or soccer.

Our drafts will allow the user and their friends to pick from the top players and assemble an all-star team based on informed decisions thanks to our player information system. Trades are also a very important part of any fantasy system, and ours will definitely have one. Knowing which data points are relevant and not is also important for a newcomer, and so we provide explanations for each along with how this can affect them. Additionally, while most fantasy sites use point systems that have all sorts of partial points, negative points, and blocked points, ours uses a simple point system. If your player scores a point, you gain a point. For every assist your player gets, they gain a point.

# Functional Requirements

*List the* ***functional requirements*** *in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

1. Create Leagues – High Priority

The ability to create a league and invite friends to it. Basically, each league will be a group of people competing with each other to see whose players can score more points. This is also how league owners are created.

1. Invite To League – High Priority

League owners should be able to invite any user to join their league.

1. Drafts – High Priority

League owners should be able to begin a draft when ready. This will begin a snake draft where the users are randomly assigned an order and given equal opportunity to choose players.

1. Point System – High Priority

The ability to gain points based on a user’s player’s performance. This is the primary metric by which players will compete.

1. Trading – Medium Priority

Users should be able to trade their players with other users in their league.

1. New Player Assistance – Low Priority

Users will be provided tools such as newsfeeds and annotated stats to make better decisions.

# Non-functional Requirements

*List the* ***non-functional requirements*** *of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

# Use Case Diagram

*This section presents the* ***use case diagram*** *for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*

# Class Diagram and/or Sequence Diagrams

*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

*If the main* ***paradigm*** *used in your project is* ***Object Oriented*** *(i.e., you have classes or something that acts similar to classes in your system), then draw the* ***Class Diagram******of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main* ***paradigm*** *in your system is* ***not Object Oriented*** *(i.e., you* ***do not*** *have classes**or anything similar to classes in your system) then only draw* ***Sequence Diagrams****,* ***but for all the use cases of your system.*** *In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams*** *show the* ***fundamental objects/classes*** *that must be modeled with the system to satisfy its requirements and* ***the relationships*** *between them. Each class rectangle on the diagram* ***must also include the attributes and the methods of the class*** *(they can be refined between increments). All the* ***relationships between classes and their multiplicity*** *must be shown on the class diagram.*

*A* ***Sequence Diagram*** *simply depicts* ***interaction******between objects*** *(or* ***functions -*** *in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*

# Operating Environment

*Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.*

# Assumptions and Dependencies

*List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.*

One dependency is the use of embedded twitter frames. While it is unlikely for twitter to change the way their frames work since it would break so many sites, it is still something we need to be aware of. Our home page also currently uses a recent games frame from sofascore.com. This could be something we do on our own in the future once our game database is working, but for now it works fine. Also on the home page, we currently have a stand-in chat system provided by minnit.chat, which is just standing in for the chat system we will make in the future.