USER REQUIREMENT SPECIFICATION



**Start Date:** 24th- April – 2022

**End Date:** 10th-June-2022

**Location:** Rachelsmolen R10, 5623 PE Eindhoven

**Made by:**Georgi Zhizgov

**Tutor:**Marcel Boelaars

Table of Contents

[1. Introduction 3](#_Toc105542518)

[2. Functional requirements 4](#_Toc105542519)

[3. Use cases 11](#_Toc105542520)

[4. Use case Diagram 15](#_Toc105542521)

# Introduction

The company DuelSys inc. wants a software solution to allow their customers (sport associations) to manage their sport tournaments. For now, the software must support a round-robin tournament system for badminton, but DuelSys inc. also wants the software to have the potential to support other types of tournament systems and sports. A tournament has multiple players competing in badminton games to determine who is the best (e.g. gold, silver and bronze medal). To determine this, the purpose of the software is to register all the results of each game. This software solution will be used by sport association staff (staff) to organize tournaments and by players to find information about the tournament(s) they want to participate in.

# Functional requirements

The requirements

Below you can find the requirements for this assignment, and they are divided into three categories. The submission must contain all the *Core requirements*, at least one (1) of the *Major requirements* and at least one (1) of the *Minor requirements;*

In order to prioritize the requirements, the best way to describe the importance of each one and show which one I would like to implement a MoSCoW table is the solution:

- Must(M):

**Core requirements**

**Non-functional requirements**

• NFR-01: Maintainable and extendable

Proper OO principles must be applied to ensure good maintainability and extensibility of the

code base.

• NFR-02: Bug free system

Appropriate testing techniques must be used when implementing the system to ensure proper functioning.

• NFR-03: Secure software

Only authorized people may make use of the system and can only access data they are authorized for. Passwords and user input must also be handled appropriately.

**Functional Requirements**

• FR-01: Manage Tournaments

Staff must be able to manage (CRUD operations) the tournaments.

This requirement must be implemented in a desktop application.

• FR-02: Support registering players

When a player is interested in participating in a tournament, they can visit the sport association website, retrieve the list of available tournaments and register themselves for it.

This requirement must be implemented in a web application.

• FR-03: Support generating tournament schedule

Staff must be able to generate the tournament’s schedule.

This requirement must be implemented in a desktop application.

• FR-04: Support registering the results of the games

When a game between two players is finished, the results must be registered in the system by staff.

You can decide whether this should be done in a web or desktop application.

• FR-05: Support showing tournament information and results

Any interested party (e.g. a sport enthusiast, a player) must be able to retrieve information about any given tournament.

This requirement must be implemented in a web application.

**Major requirements**

**Functional requirements**

FR-06: Support multiple tournament systems

Extend the software solution to also support different tournament systems. It should be possible, for a staff member, to specify what tournament system should be used when creating a new tournament. For now, at least one of the following tournament systems (in addition to round-robin) is required:

o Single-elimination

o Double-elimination

o Double round-robin

**Minor requirements Functional requirements**

FR-10: Generate player profile

Extend the software solution to also show player profile information. The information should include player’s general information, information related to participation in different tournaments with the ranking, and also can include the individual matches (games) played against different opponents.

Should(S):

FR-07: Support multiple sport types

Extend the software solution to also support different sport types (e.g. basketball, tennis, quidditch, league of legends, chess, etc.). It should be possible, for a staff member, to specify which sport type when creating new tournament. Make sure that when registering the result of a game the official scoring rules is followed.

FR-09: Support challenge games

Extend the software solution to also support challenge games. It should be possible for a player to challenge another player for a game. When the challenge is created, the opponent can either accept or reject the challenge. If it is accepted, the result of the game can be registered by one of the players.

- Could (C):

FR-11: Handle ties

At the end of a tournament, when all matches are played that the software only have three winners. The software handles any ties, i.e., two or more players have won the same number of games/matches, so that is always only one player per for the first, the second and the third places. As an example given in Figure 3 of Appendix A, Kento Momota and Viktor Axelsen won two (2) games/matches each, while the other opponents just won one (1). The system decides that Kento Momota should score the first place, because he won both games played against Viktor Axelsen. For handling ties, you can re-use and extend the strategy given in the example above or define your own strategy.

- Would (W):

FR-09: Support leader board

Extend the software solution to also support a leader board. When there is an ongoing tournament, any interested party (e.g. a sport enthusiast, a player) can retrieve the list of players participating in the tournament, ordered based on their current position/rank in the tournament.

FR-08: Support matches in a tournament

Extend the software solution to support multi-game matches in a tournament. For now, every match will consist of three (3) games that must be played by the same players and the player with most one games has won the match.

# Use cases

|  |  |
| --- | --- |
| **UC-01** | Log in |
| **Actor** | Any user |
| **Precondition** | Web App must be started |
| **Main success scenario** | 1. A user inputs his credentials 2. The system processes the user’s credentials 3. The system informs that the user was successfully logged in 4. The system moves from the log in page to the home page 5. End of use case |
| **Extensions**: | 1.1 No data was input  1. A message informs the user that there was no data entered  2. End of use case  1.2 Wrong data input  1. A message shows that the input data was in the wrong format  2. End of use case |

|  |  |
| --- | --- |
| **UC-02** | Register |
| **Actor** | User |
| **Precondition** | Web App must be started |
| **Main success scenario** | 1. User inputs the information needed about the new registration.  2. The system processes user’s information  3. Program verifies if the data is correctly input.  4. The system confirms that the new registration was successfully created  5. End of use case |
| **Extensions:** | 3.1 User with the same data   1. The system prompts that there is an employee with the same data(email). 2. End of use case   3.2 There is missing or wrongly input information  1.The system informs the user that the input data is not in the correct format.  2.End of use case |

|  |  |
| --- | --- |
| **UC-03** | Add tournament |
| **Actor** | Admin |
| **Precondition** | Desktop App should be started |
| **Main success scenario** | 1. Admin inputs the information needed about the new registration.  2. System verifies if the data is correctly input.  3. The system confirms that the new tournament was successfully created  4. End of use case |
| **Extensions:** | * 1. Tournament with the same data   1. The system prompts that there is a tournament with the same data (same start date and location).  2.End of use case  2.2 There is missing or wrongly input information  1. The system informs the user that the input data is not in the correct format.  2. End of use case |

|  |  |
| --- | --- |
| **UC-04** | Cancel a tournament(upcoming) |
| **Actor** | Admin |
| **Precondition** | Desktop App must be started |
| **Main success scenario** | 1. Admin selects the tournament.  2. Admin submits cancelation request.  3. Program verifies if the data is correctly input.  4. The system confirms that the deletion was successful  5. End of use case |
| **Extensions:** | 3.1. Tournament is already scheduled/canceled/over date   1. The action is canceled 2. End of use case |

|  |  |
| --- | --- |
| **UC-05** | Update tournament |
| **Actor** | Admin |
| **Precondition** | Desktop App must be started |
| **Main success scenario** | 1. Admin selects the tournament.  2. Admin submits the new information about the tournament.  3. Program verifies if the data is correctly input.  4. The system confirms that the update was successful  5. End of use case |
| **Extensions:** | 3.1. The tournament doesn’t exist   1. The action is canceled 2. End of use case |

|  |  |
| --- | --- |
| **UC-06** | Create tournament schedule |
| **Actor** | Admin |
| **Precondition** | Desktop App must be started |
| **Main success scenario** | 1. Admin selects to which tournament should the schedule be created tournament.  2. The system verifies if the number minimum count of competitors is reached and if there is already less than a week to the start of the tournament.  3. The system confirms that the schedule has already been created  4. End of use case |
| **Extensions:** | 3.1. Minimum competitors has not been reached   1. The action is canceled 2. End of use case   3.2 The tournament is still open for registration(there is more than a week until the start) |

|  |  |
| --- | --- |
| **UC-07** | Set score to a match |
| **Actor** | Admin |
| **Precondition** | Desktop App must be started , tournament should be scheduled |
| **Main success scenario** | 1. Admin selects the match of the tournament and the round.  2. Admin submits the result information.  3. Program verifies if the data is correctly input.  4. The system confirms that the update of the match was successful  5. End of use case |
| **Extensions:** | 3.1. The score information was not in the correct format.   1. The action is canceled 2. End of use case   3.2 The score of the current match has already been set.   1. The action is canceled 2. End of use case |

|  |  |
| --- | --- |
| **UC-08** | Sign in for a tournament |
| **Actor** | User |
| **Precondition** | User should be registered in the website, should be logged in and should have not signed in before for this tournament |
| **Main success scenario** | 1. User selects the tournament to sign in for.  2. User submits the sign in request.  3. Program verifies if the data is correctly input.  4. The system redirects to profile page.  5. End of use case |
| **Extensions:** | 3.1. The user has already signed in for the tournament   1. The action is canceled 2. The system redirects to index page 3. End of use case |

# Use case Diagram

