**Requirement Document**

**.1 Introduction**

1.1 **Purpose of Document**

This is a requirement document for a web-based collaborative sports system.

The game will improve the sports experience for anyone who wants to play, by offering him courts suitable for the type of sport chosen that are close to his current place. The system will see how many players are currently in the selected court and if the amount sufficient the game will be appointed to everyone inside the group ,the size of the group must meet a certain number In order for a game to take place.

This document describes the scope and objectives of the new system.

In addition to describing the non-functional requirements, this document designs the functional requirements with use cases, interaction diagrams, and classic models.

1.2 **Project Scope**

The scope of this project is a web-based system that supports in search of existing courts in the system by no-sql queries. user place services, direct communication between users are not part of this project.

2. Actors and Goals

Actors names:Players(Player)-primary,can see all courts that are active, and create or join

to a game that planned on court.

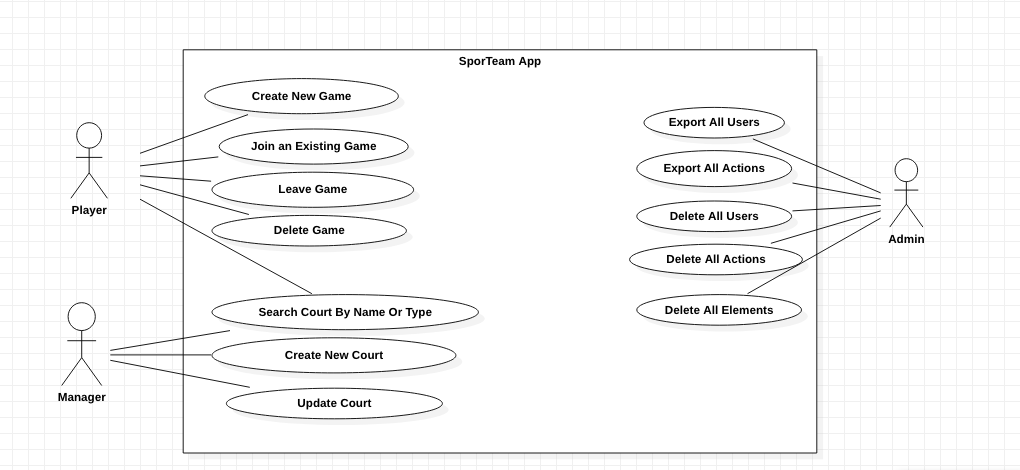
Manager-support , can create and edit new courts to the system.

Admin -Responsible for system administration.

3. Functional Requirements

* Users should be able to create a new game .
* Users should be able to join an existing game .
* Users should be able to leave an existing game .
* Users should be able to delete a game.
* Users should be able to search courts by name or type

3.1 Use Case Diagram



3.2 Use Case Details

|  |  |
| --- | --- |
| Use Case Name: | Create new game |
| Summary: | This use case create new game by the player |
| Basic Flow: | 1. The player login to the system 2. The player defines details about the game ( type, players range, age range, hours..) 3. The system displays related play-courts close by . 4. The player chooses the desired play-court. 5. The System creates a new game group. |
| Alternative Flows: | 3.a There aren’t available courts.  3.b The system will not create the game. |
| Participating Actors | Player. |

|  |  |
| --- | --- |
| Use Case Name: | Join to game |
| Summary: | This use case show how to join to existing game |
| Basic Flow: | 1. The player logs in to the system. 2. The player chooses the type of sport he wants to play. 3. The system displays games with open spots. 4. The player picks up a game. 5. The system updates the game’s saved details. |
| Alternative Flows: | 1. There isn't games with open spot 2. The system notifies the player |
| Participating Actors | Player . |

|  |  |
| --- | --- |
| Use Case Name: | Leave Game |
| Summary: | This use case allows the user to leave an existing game. |
| Basic Flow: | 1.Player registered to a game.  2.Player wants to leave the game that he is registered in.  3.Player press on leave group button.  4.System update the group without this player. |
| Participating Actors | Player. |

|  |  |
| --- | --- |
| Use Case Name: | Delete Game |
| Summary: | This use case allows the user to delete a game only if he is the creator of this game. |
| Basic Flow: | 1.Player created a game.  2.Player wants to delete the game that he created.  3.Player press on delete group button.  4.The system deletes the game from the games list.  5.The system update the new games list |
| Alternative flow | 1a.Player registered to a game.  2a.Player wants to delete the game that he is registered  3a.Player press on delete group button.  4a.The system will not delete the game and notify the user that only the player that has created the game can delete it. |
| Participating Actors | Player . |

|  |  |
| --- | --- |
| Use Case Name: | Search by Type/Name |
| Summary: | This use case is for searching courts by Name or sport Type. |
| Basic Flow: | 1.Player decides to search the court by name or sport type.  2.Player enter the name or the type of the court that he would like to search.  3.The system displays a list of courts that have the same name or type that have been entered by the Player . |
| Alternative flow | 2a.Player enter name or sport type that doesn't exist in the system  3a.The system doesn't display courts list. |
| Participating Actors | Player. |

|  |  |
| --- | --- |
| Use Case Name: | Create new court |
| Summary: | This use case is for creating new court by manager |
| Basic Flow: | 1. Manager decide to create new court  2.Manager press on button create new court  3.Manager enter court details  4. New court created |
| Alternative flow | 3a.Player court details with null name/type  4a.The system doesn't create new court |
| Participating Actors | Manager |

|  |  |
| --- | --- |
| Use Case Name: | Update court |
| Summary: | This use case is for updating court by manager |
| Basic Flow: | 1.Manager decide to update court  2.Manager enter the id of the court  3.Manager enter the details for updating.  4.Court details updated |
| Alternative flow | 2a.Manager enter id that doesn't exist in the system  3a.The system doesn't update the details. |
| Participating Actors | Manager |

|  |  |
| --- | --- |
| Use Case Name: | Delete all Users/Elements/Actions |
| Summary: | This use case is for delete all Users/Elements/Actions by admin |
| Basic Flow: | 1.Admin decide to delete all Users/Elements/Actions  2.Admin press on button delete Users/Elements/Actions  3.The Users/Elements/Actions deleted successfully |
| Alternative flow | 2a. the user role is not admin (player/manager)  3a. The system will not delete the Users/Elements/Actions |
| Participating Actors | Admin |

|  |  |
| --- | --- |
| Use Case Name: | Export all Users/Actions |
| Summary: | This use case is for export all Users/Actions by admin |
| Basic Flow: | 1.Admin decide to export all Users/Actions  2.Admin press on button export Users/Actions  3.The Users/Actions exported successfully |
| Alternative flow | 2a. the user role is not admin (player/manager)  3a. The system will not export the Users/Actions |
| Participating Actors | Admin |

4. Non Functionals Requirements

|  |  |  |
| --- | --- | --- |
| Requirement Type | Requirement Description | Requirement Number |
| Usability | 1. The system would use simplified icons for easy navigation . 2. The system should be efficient to use with minimal clicks as possible. 3. The system overall design should be simple and easy on the eye . 4. The system will use clean and easy to understand language. 5. The system will warn the user about errors by displaying an appropriate message to each specific error. 6. The system will provide details on how to do such actions when the user presses the help button. |  |
| Reliability | * The system shall be completely operational at least 90% of the time. * Down time after a failure shall not exceed 2 hours. |  |
| Performance | * The system should be able to support 50 simultaneous users. * The system shouldn’t take more than one second to respond . |  |
| Supportability | * The system should be designed around ease of migration and upgradability . * The system would be easy to install and run (one click away) * The system will use White-Box extensibility , only core team members would be able to add/change the system |  |

**Technologies**

**Database : MongoDB and mongodb atlas for hosting .**

**Rest-api :Spring boot and AWS for hosting .**

**Client : web app built using reactjs, redux and hosted on AWS.**

**Tests: Junit and postman .**

**General : Chrome Dev Tools , vscode , vim , SmartGit , bitBucket , springToolSuit4.**

**Communication: Zoom, gdrive&google docs, whatsapp.**