Software Requirements Specification

(SRS) Document

Hawthorne Groveland University Intramural Sports System

11/7/2022

Document Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Edited by | Reason |
| 1.0 | 11/7/2022 | Henry Hsieh | Initialize first version of SRS |
| 1.1 | 11/16 | Adam Ranieri | Reformatted, added diagrams |

Contents

[Introduction 3](#_Toc119501538)

[1.1. Purpose 3](#_Toc119501539)

[1.2. Document Conventions 3](#_Toc119501540)

[1.3. Intended Audience 4](#_Toc119501541)

[1.4. Scope 4](#_Toc119501542)

[1.5. References 4](#_Toc119501543)

[General Description 4](#_Toc119501544)

[2.1. Product Perspective 4](#_Toc119501545)

[2.2. Product Features 5](#_Toc119501546)

[2.3. User class and characteristics 5](#_Toc119501547)

[2.4. Operating Environment 5](#_Toc119501548)

[2.5. Constraints 5](#_Toc119501549)

[2.6. Assumptions and dependencies 5](#_Toc119501550)

[System Requirements 6](#_Toc119501551)

[3.1. Functional Requirements 6](#_Toc119501552)

[3.1.1. Unregistered player functionalities: 6](#_Toc119501553)

[3.1.2. Player functionalities: 6](#_Toc119501554)

[3.1.3. Referee functionalities: 7](#_Toc119501555)

[3.1.4. Admin functionalities: 7](#_Toc119501556)

[3.1.5. Captain functionalities: 8](#_Toc119501557)

[3.1.6. The RESTful API should have the following functionalities: 8](#_Toc119501558)

[3.2. Design Requirements 8](#_Toc119501559)

[3.3. Testing Requirements 8](#_Toc119501560)

[External Interface Requirements 9](#_Toc119501561)

[4.1. User Interfaces 9](#_Toc119501562)

[4.2. Hardware Interfaces 9](#_Toc119501563)

[4.3. Communication Interfaces 9](#_Toc119501564)

[4.4. Software Interfaces 9](#_Toc119501565)

[Non-Functional Requirements 9](#_Toc119501566)

[5.1. Performance Requirements 9](#_Toc119501567)

[5.2. Safety Requirements 9](#_Toc119501568)

[5.3. Security Requirements 9](#_Toc119501569)

[5.4. Software Quality Attributes 9](#_Toc119501570)

[5.5. Other Requirements 10](#_Toc119501571)

## Introduction

# 1.1. Purpose

This Software Requirements Specification (SRS) document describes the functions and details of the Intra-Mural program. This document should serve as the basis for a software team to be able to modify the program and test it to a certain extent.

# 1.2. Document Conventions

|  |  |
| --- | --- |
| JSON | JavaScript Object Notation |
| REST | Representational State Transfer |
| IMSS | Intra-Mural Sports System |
| CRUD | Create/Read/Update/Delete |

# 1.3. Intended Audience

This document is to be sent to all members of the Scrum Development and testing teams. All members should read this document to gain an overview of what must be achieved in this application. The product owner of the application is Adam Ranieri.

# 1.4. Scope

This application is marketed to Hawthorne Groveland University to track and obtain information with regards to player stats, games, and scheduling. IMSS will allow players, referees, admins, and captains to easily search up information with regards to player stats, game histories, venues, and scheduling.

# 1.5. References

[**Figure A IMSS ERD**](https://drive.google.com/file/d/1mnBbo5tHnfnw2lriBqL0Ro4sotf8SBgn/view?usp=share_link)

[**Figure B User Stories**](https://docs.google.com/spreadsheets/d/1wd57JNdhyG_sCtvMHq1XUdIWOJkZcoDFWGuPwHXooLg/edit?usp=share_link)

**Figure C Use Case Diagrams**

[**Front-End Git Repository**](https://github.com/adamranieri/intra-mural-frontend)

[**Back-End Git Repository**](https://github.com/adamranieri/IntraMuralsApp)

## General Description

# 2.1. Product Perspective

Hawthorne Groveland University is currently using a pen and paper system to manage their intramural sports teams. This includes venues where the games are held, who is going to be the referee and what teams are playing. Player stats in the game are currently untracked. Hawthorne Groveland University wants to manage all Intramural related activities with a dedicated system. They have found other systems like shared excel sheets to be too cumbersome and have failed to scale with their 100’s of players.

# 2.2. Product Features

This application will consist of two parts.

First, a user facing web application that will allow players, referees, admins, and captains to interface with the system. This application will be the primary way for all to interact with the system.

Second, there should be a RESTful web service for the querying of the database. This RESTful web service will eventually be integrated into their existing systems. Security of these systems will be handled later by a separate team.

# 2.3. User class and characteristics

2.3.1 Player

- A user who is a student or faculty member who is participating in intramural sports

2.3.2 Referee

- A user who is a student or faculty member who is officiating intramural sports

2.3.3 Admin

- Faculty member with high level of permission in the system

- Create, submit, and update all games/venues/seasons/player stats.

- Create new admins

2.3.4 Captain

- A player who is in charge of a team

# 2.4. Operating Environment

This backend web server is designed to run on a Linux VM hosted in the cloud. The cloud service is up to the discretion of developers/operations team. The only stipulation being that any hosting service have an SLA uptime of at least 99.5%. The frontend of the application should be statically hosted. Anticipated workload of the backend is expected to be a few dozen concurrent users at any one time. As small of VM is preferred to keep costs to a minimum.

# 2.5. Constraints

There are no constraints to this application.

# 2.6. Assumptions and dependencies

All users of this application will have high speed internet, modern web browsers, and modern software/hardware.

## System Requirements

# 3.1. Functional Requirements

The following serves as a high-level overview of functionalities to be performed by IMSS. Detailed user stories and acceptance criteria are linked in Section 1.5 of this document.

# 3.1.1. Unregistered player functionalities:

3.1.1.1. Register for an account

3.1.1.2. View the Captains of each team

3.1.1.3. View all the games

3.1.1.4. View all the seasons

3.1.1.5. View all the venues

3.1.1.6. View all the schedules

# 3.1.2. Player functionalities:

3.1.2.1. Login to the system

3.1.2.2. Update their own password

3.1.2.3. Update their own profile picture

3.1.2.4. Update their own height and weight

3.1.2.5. Choose if they want to display their biometric info or not

3.1.2.6. View the Captains of each team

3.1.2.3. View all the games

3.1.2.4. View all the series

3.1.2.5. View all the venues

3.1.2.6. View all the schedules

3.1.2.7. Apply to join a certain team

3.1.2.8. Apply to become an inactive player

3.1.2.9. Apply to become an active player

# 3.1.3. Referee functionalities:

3.1.3.1. Login to the system

3.1.3.2. Update their own password

3.1.3.3. Update their own profile picture

3.1.3.4. Update their own height and weight

3.1.3.5. Choose if they want to display their biometric info or not

3.1.3.6. View the Captains of each team

3.1.3.7. View all the games

3.1.3.8. View all the series

3.1.3.9. View all the venues

3.1.3.10. View all the schedules

3.1.3.11 Edit the scorecard for any game they officiate

# 3.1.4. Admin functionalities:

3.1.4.1. Login to the system

3.1.4.2. Update their own password

3.1.4.3. Update their own profile picture

3.1.4.4. Update their own height and weight

3.1.4.5. Choose if they want to display their biometric info or not

3.1.4.6. View the Captains of each team

3.1.4.7. View all the games

3.1.4.8. View all the seasons

3.1.4.9. View all the venues

3.1.4.10. View all the schedules

3.1.5.11. Create a season

3.1.5.12. Schedule all games in a season

3.1.5.13. Create new admins

3.1.5.14. Create new referees

3.1.5.15 Full CRUD abilities of games

# 3.1.5. Captain functionalities:

3.1.5.1. Login to the system

3.1.5.2. Update their own password

3.1.5.3. Update their own profile picture

3.1.5.4. Update their own height and weight

3.1.5.5. Choose if they want to display their biometric info or not

3.1.5.6. View the Captains of each team

3.1.5.7. View all the games

3.1.5.8. View all the series

3.1.5.9. View all the venues

3.1.5.10. View all the schedules

3.1.6.11. Approve or Deny requests from players to join their team

# 3.1.6. The RESTful API should have the following functionalities:

3.1.6.1. Register a team

3.1.6.2. Retrieve all teams

3.1.6.3. Retrieve all venues

3.1.6.4. Schedule a game

3.1.7.5. Retrieve all games

3.1.7.6. Retrieve all seasons

# 3.1.7 Additional Business Requirements

3.1.7.1 Games cannot be scheduled for past dates

# 3.2. Design Requirements

3.2.1. **Front-end**

3.2.1.1. The front end must be developed with HTML/CSS/JS

3.2.2. **Back-end**

3.2.2.1. The web server will be written in Java

3.2.2.2 JDBC will be used for communication with the database

3.2.2.3 PostgreSQL will be the RDBMS used

# 3.3. Testing Requirements

3.3.1. JUnit5 tests of all services within the application

3.3.2. Postman collection of API endpoints

3.3.3. Automated Selenium and Gherkin Acceptance tests with Cucumber

# External Interface Requirements

# 4.1. User Interfaces

The logic behind the interactions between the users and the software. This includes the sample screen layout, buttons and functions that would appear on every screen, messages to be displayed on each screen and the style guides to be used.

# 4.2. Hardware Interfaces

There are no hardware interfaces for this web application.

# 4.3. Communication Interfaces

Communication between the client and server should be entirely in JSON.

# 4.4. Software Interfaces

This application will be used in a browser by Players, Referees, Admins, and Captains. This application must be accessible and displayed correctly via Chrome and Firefox.

# Non-Functional Requirements

# 5.1. Performance Requirements

5.1.1. No one page of the web application should take longer than 3 seconds to load.

5.1.2. The user experience should be as straightforward as possible.

# 5.2. Safety Requirements

5.2.1. We do not anticipate any safety requirements for this application.

# 5.3. Security Requirements

5.3.2. There is no sensitive information stored in the database

5.3.3. There is no identifiable information in the database

5.3.4. Passwords may be stored in plain text

5.3.5. The site may be accessed over HTTP

# 5.4. Software Quality Attributes

5.4.1. Application source code should be readily available in a public repository on GitHub.

5.4.2. Features of the application should be added via branching and pull requests to create a detailed history of the development.

5.4.3. Static code analysis should be run at the end of every sprint to identify code smells

# 5.5. Other Requirements

API endpoints should be documented with OpenAPI 3.0. This will allow future applications to easily consume the web service aspect of this application.