**RecDawgs**

Recreational Sports Management System

Version 1.0

21 March 2016

Team9

Justin Tumale  
Logan Jahnke

Jay Springfield

Bowen Yang

**Table of Contents**

|  |  |
| --- | --- |
| [RecDawgs](#h.9rerm8q0r3h3) | 0 |
| [Recreational Sports Management System](#h.be4dw9zaxsbc) | 0 |
| [Table of Contents](#h.nngih44j9nfm) | 1 |
| [1 - Introduction](#h.an0gkew36epe) | 3 |
| [1.1 - Purpose](#h.laa1xhnftfwx) | 3 |
| [1.2 - Design Goals](#h.ejuuo9ob3g53) | 3 |
| [1.2.1 - Definition:](#h.lgiqoox0ig7n) | 3 |
| [1.2.2 - Trade-Offs:](#h.rx9o9qja4uyf) | 3 |
| [1.3 - Definitions, Acronyms, and Abbreviations](#h.p38z188l1wo6) | 4 |
| [1.4 - References](#h.661s04wm3l7a) | 4 |
| [1.5 - Overview](#h.kafj46cij9qs) | 4 |
| [2 - Proposed System Architecture](#h.qvs3qh3dfuh4) | 5 |
| [2.1 - System Overview](#h.rdz1znyj31vi) | 5 |
| [2.2 - Subsystem Decomposition](#h.ici5i9vtjhw6) | 6 |
| [2.2.1 Layers](#h.2vtfn3m4nt4s) | 7 |
| [2.2.2 Coherence](#h.931d4hyv1pf) | 7 |
| [2.2.3 Coupling](#h.olxsoxjb9osv) | 7 |
| [2.3 Concurrency](#h.bs3cwrvtgxrr) | 7 |
| [2.3.1 Identification of Threads](#h.5hcpz0vhlxw1) | 7 |
| [2.4 Hardware/Software Mapping-may need some minor revisions](#h.2fgyev2l0dj2) | 8 |
| [2.5 Data Management](#h.i18dkov5nt23) | 8 |
| [2.5.1 Persistent Objects](#h.v9eyvwzdhn90) | 8 |
| [2.5.2 Database](#h.l8uey5rgnbvo) | 8 |
| [2.6 Global Resource Handling](#h.ngf6qwdokbz2) | 9 |
| [2.6.1 Access Control Security List vs. Capabilities Security](#h.4z5lypl2glna) | 9 |
| [2.7 Software Control](#h.b3testcgj6x8) | 9 |
| [2.7.1. Centralized design](#h.wc7szkbbxxgf) | 9 |
| [2.7.2 Event-Driven](#h.n8gk8efjibvd) | 9 |
| [2.7.3 Concurrent Processes](#h.xflqyxmxb1no) | 9 |
| [2.8 Boundary Conditions](#h.lirz9am1efi2) | 10 |
| [2.8.1 Initialization](#h.7kpiyds4wynw) | 10 |
| [2.8.2 Termination](#h.e90sq0nq9jf1) | 10 |
| [2.8.3 Failure](#h.kmajd7k36qzk) | 10 |

# **1 - Introduction**

## **1.1 - Purpose**

RecDawgs is a Recreational Sports Management System designed to support the management of several recreational sports leagues for a small college with minimal involvement of college staff.

## **1.2 - Design Goals**

The product to be produced is titled ‘RecDawgs’. The design goals of RecDawgs includes its definition and trade-offs.

### 1.2.1 - Definition:

*User-friendly*- We want users to be able to navigate the system and perform desired functions with ease. In order to do this, the user-interface must be simple, sleek, and organized in a logical fashion.

*Availability-* This system should be able to handle a significant load of users, concurrently. The more users that register for RecDawgs, the more urgent the need for availability.

*Good performance-* Users will be able to access RecDawgs services with a short

response time in order to add to the quality of their experience.

*Reliability-* The cost of application failure is high, as it may result in a loss of

registered students along with the use of time and effort to maintain code.

Therefore, this system will be well tested to perform quality operations.

*Flexibility-* RecDawgs must allow for adaptation of changing needs and therefore will be designed will be designed with modularity, simplicity, and readable code in order to achieve this goal.

### 1.2.2 - Trade-Offs:

* The ease of use prevents it from doing functions that are too complex.
* RecDawgs will be efficient for the web-interfaces, such as Internet Explorer, but will not be portable to other platforms.

## **1.3 - Definitions, Acronyms, and Abbreviations**

* JavaScript (JS)
* Hypertext Markup Language (HTML)
* Cascading Style Sheets (CSS)
* Java Database Connectivity (JDBC)
* Java ServerPages (JSP)

## **1.4 - References**

Almstrum, V., Dr. (n.d.). Software Requirements Document Model. Retrieved February 1, 2016, from http://web.stonehill.edu/compsci/CS400/SoftwareRequirementsDocument.doc

Kochut, K. J., Dr. (2016). Term Project. Retrieved February 01, 2016, from http://cobweb.cs.uga.edu/~kochut/teaching/x050/TermProject.html

McKinnon, A. D. (2005, February 09). Software Requirements Specification Template. Retrieved February 1, 2016, from http://www.tricity.wsu.edu/~mckinnon/cpts322/cpts322-srs-v1.doc

http://www.slideshare.net/parag/goals-of-software-design-the-main-goals

https://msdn.microsoft.com/en-us/library/aa292150(v=vs.71).aspx

## **1.5 - Overview**

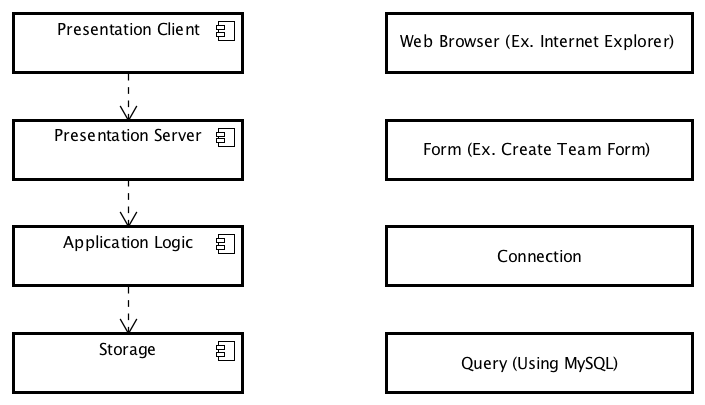
This system design document for RecDawgs contains the introduction, a specification for the proposed system including the subsystems, concurrency, hardware/software mapping, data management, global resource handling, software control, boundary conditions, and a glossary.

# **2 - Proposed System Architecture**

## **2.1 - System Overview**

RecDawgs consists of a four-tier architecture. This includes the Presentation Client, the Presentation Server, the Application Logic, and the Storage. The Presentation Client is found on the Client’s device using a web browser such as Internet Explorer. The Presentation Server is found on a cluster of servers provided by the University of Georgia. The Application Logic contains all application code and entities. The Storage stores persistent data.

WE NEED TO CHANGE THIS DIAGRAM TO MATCH THE FOLLOWING ONE BELOW



## **2.2 - Subsystem Decomposition**

### 2.2.1 Presentation Layer - UnregisteredUserUI Subsystem

|  |  |  |  |
| --- | --- | --- | --- |
| *Operation* | *Return* | *Arguments* | *Description* |
| logIn |  | username  password | Verifies the user identity and logs them in. |
| register |  | Username  firstName  lastName  password  studentID  major  email  address | Registers a new authenticated user with the given credentials. |

### 

### 2.2.2 Presentation Layer - RegisteredUserUI Subsystem

|  |  |  |  |
| --- | --- | --- | --- |
| *Operation* | *Return* | *Arguments* | *Description* |
| logout |  |  | Logs the Authenticated User out. |
| viewAccount | User |  | Allows the Authenticated User to view their account details. |
| modifyAccount | User | firstName  lastName  password  major  email  address | Allows the Authenticated User to change their account details. |
| viewLeagues | List<League> |  | Displays the leagues. |
| viewTeams | List<Team> | League | Displays the teams in the given League. |
| viewMyTeams | List<Team> |  | Displays the teams that the Authenticated User is a member/captain of. |

### 

### 2.2.3 Presentation Layer - AdministratorUI Subsystem

### 

|  |  |  |  |
| --- | --- | --- | --- |
| *Operation* | *Return* | *Arguments* | *Description* |
| logout |  |  | Logs the Authenticated User out. |
| viewAccount | User |  | Allows the Authenticated User to view their account details. |
| modifyAccount | User | firstName  lastName  password  major  email  address | Allows the Authenticated User to change their account details. |
| viewLeagues | List<League> |  | Displays the leagues. |
| viewTeams | List<Team> | League | Displays the teams in the given League. |
| viewMyTeams | List<Team> |  | Displays the teams that the Authenticated User is a member/captain of. |
|  |  |  |  |

### 

### 2.2.4 Application Logic- UnregisteredUserControl Subsystem

|  |  |  |  |
| --- | --- | --- | --- |
| *Operation* | *Return* | *Arguments* | *Description* |
| logIn |  | Username password | Verifies the user identity and logs them in. |
| register |  | Username  firstName  lastName  password  studentID  major  email  address | Registers a new authenticated user with the given credentials. |
| viewLeagueStatistics | List<League Stats> | League Name | Allows the user to view stats on a specific league |

### 

### 2.2.5 Application Logic- RegisteredUserControl Subsystem

|  |  |  |  |
| --- | --- | --- | --- |
| *Operation* | *Return* | *Arguments* | *Description* |
| logOut |  |  | Logs the Authenticated User out. |
| viewAccount | User |  | Allows the Authenticated User to view their account details. |
| modifyAccount | User | firstName  lastName  password  major  email  address | Allows the Authenticated User to change their account details. |
| viewLeagues | List<League> |  | Displays the leagues. |
| viewTeams | List<Team> | League | Displays the teams in the given League. |
| viewMyTeams | List<Team> |  | Displays the teams that the Authenticated User is a member/captain of. |
| reportMatch (\*requires team captain status) |  | team1Score team2Score | Records the score of the two teams that played in a match. |
| leaveTeam |  | User TeamName | Allows the user to leave the team he chooses. |
| viewLeagueStatistics | List<League Stats> | League Name | Allows the user to view stats on a specific league |

### 

### 2.2.6 Application Logic- AdministratorControl Subsystem

### 2.2.7 Object Layer (general)

### 2.2.8 Persistence Layer

### 2.2.9 Layers

RecDawgs consists of a four-tier, closed architecture. This means there are four layers used in our system design, and each layer depends only on the one layer directly below it. The layers include the Presentation Client, the Presentation Server, the Application Logic, and the Storage. The Presentation Client can access anything on the Presentation Server, depending on the user’s authentication level (UnregisteredUserUI, RegisteredUserUI, AdministratorUI). The Presentation Server can access anything on the Application Logic, depending on the user’s authentication level (UnregisteredUserControl, RegisteredUserControl, AdministratorControl). The Presentation Server accesses the Application Logic to engineer new objects, delete objects, and edit objects. And finally, the Logic is executed and stored onto the Storage.

### 2.2.10 Coherence

RecDawgs has a high coherence because the classes in the subsystem(s) perform similar tasks and are related to each other via associations.

### **2.2.**11 **Coupling**

RecDawgs has a low coupling because a change in one subsystem only affects a few other subsystems.

## 2.3 Concurrency

RecDawgs will be a concurrent-enabled application which allows multiple users to be logged in at the same time across the globe. The users will be able to create/edit/delete several objects by accessing the database simultaneously with other users.

### 2.3.1 Identification of Threads

* Multiple people can be logged on at the same time
  + Admin(s)
  + User(s)
* Database access
  + Viewing statistics
  + Viewing team/league/account information

## 2.4 Hardware/Software Mapping-may need some minor revisions

The four-tier architectural style (Presentation Server, Application Logic, Object Logic, and Storage) is realized in hardware as two systems: Server and Client. The Server contains the Presentation Server, Application Logic, and Storage. The Client contains the Presentation Client and the User Interface.

The Presentation Client is a web browser (ex. Internet Explorer) which will retrieve and send data to the Server while presenting data to the Client. The web browser is capable of running JS which can alter the data displayed on the Client.

The Presentation Server and Application Logic layers reside inside the Server. The interface is realized through HTML and CSS. The HTML is wrapped in JDBC which allow for a dynamic application to be distributed through the web. The JDBC use the Java programming language.

The Storage layer resides inside the Server. The persistent data is stored using MySQL server relational database system (RDBMS). This layer ensures security and data persistence.

## 2.5 Data Management

### 2.5.1 Persistent Objects

Persistent objects will be stored and retrieved from the RecDawgs Database. The data require access at fine levels of details by multiple users. Multiple application programs can access the data. The data management requires infrastructure and is located in the Storage layer.

### 2.5.2 Database

RecDawgs contains mechanisms for describing data, managing persistent storage and for providing a backup mechanism, it provides access to the stored data, and contains information about the data (“meta-data”), also called data schema.

## 2.6 Global Resource Handling

### 2.6.1 Access Control Security List vs. Capabilities Security

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | League | Team | Match | Score Report | Venue |
| User/Not Registered | view() | view() |  | view() |  |
| Admin | <<create>> create()  view()  edit()  delete() | <<create>>  create()  view()  edit()  delete() | end() | report() | <<create>>  create()  edit()  delete() |
| Student | join()  view() | <<create>>  create()  view()  edit() | play()  forfeit() | report() |  |

## 2.7 Software Control

### 2.7.1. Centralized design

RecDawgs has a centralized design. There is one manager object (Admins), which control all other objects RecDawgs contains. This is evident in many cases such as creating venues, creating leagues, and even assigning the Team captain when the previous team captain leaves. If this were a decentralized system then the players would be able to decide between themselves who the new captain would be.

### 2.7.2 Event-Driven

RecDawgs is Event-driven. After RecDawgs is started up, it will stay online and wait for external events such as requesting for score or logging in. When these external events occur, RecDawgs will dispatch the appropriate object depending on the given information from the event. The objects will then accomplish the given event. (Taken & reworded from textbook page 276)

### 2.7.3 Concurrent Processes

Concurrent processes will be limited to users viewing data on the database. What will not be concurrent will be data access. (basically same as threads section)

## 2.8 Boundary Conditions

### 2.8.1 Initialization

* All account information needs to be accessed at startup time
  + Account type
  + Name
  + Address
  + Major
  + Student ID
  + Email
* The user interface displays the GUI at startup time

### 2.8.2 Termination

* Some single subsystems cannot terminate
  + RecDawgs Server
  + User Management
* Subsystems are notified if a single subsystem is terminated
* Subsystems are notified of updates

### 2.8.3 Failure

* If the RecDawgs server fails, the system crashes
* If the User Management subsystem fails, users will not be able to log in
* The system can recover via restart