RecDawgs

Recreational Sports Management System

Version 1.0

21 March 2016

Team9

Justin Tumale

Logan Jahnke

Jay Springfield

Bowen Yang

Table of Contents

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

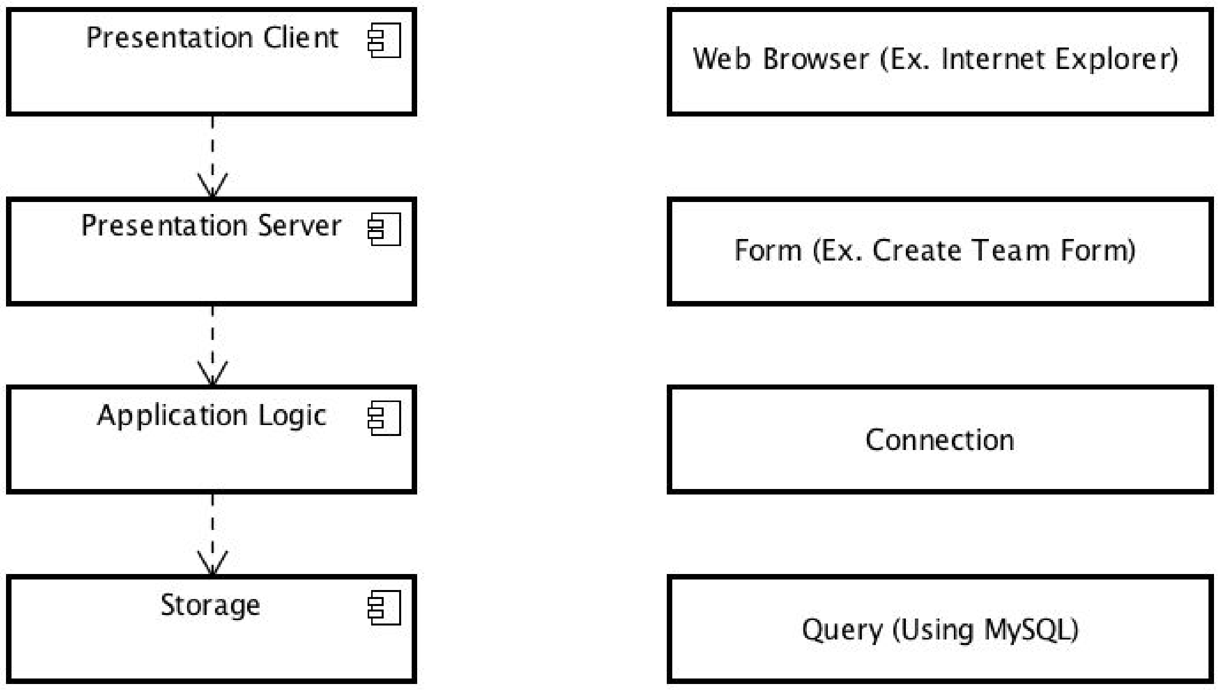
1.5 - Overview

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.

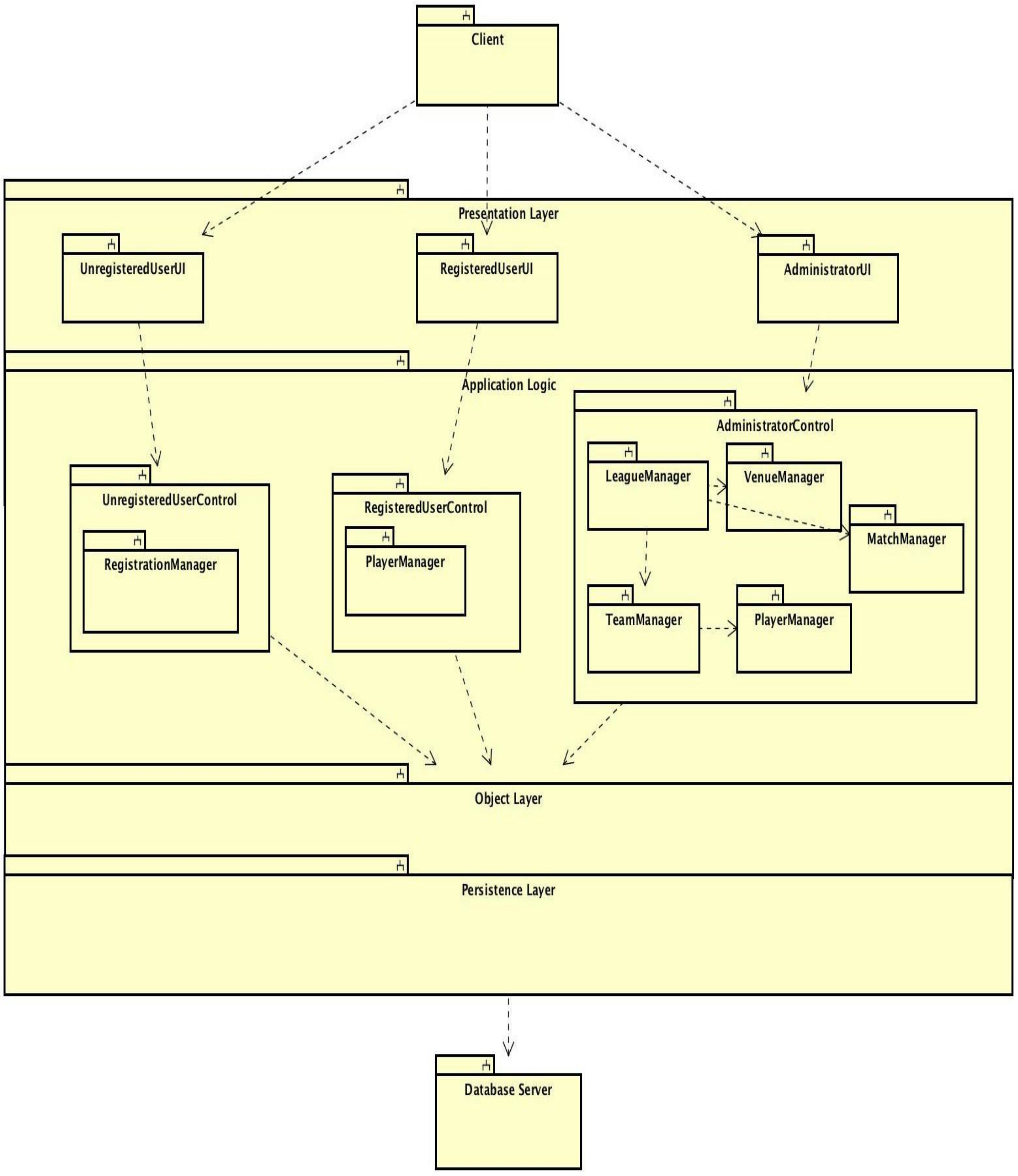
2 - Proposed System Architecture

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.



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.
viewLeagues	List<League>		Displays the leagues.
viewTeams	List<Team>	League	Displays the teams in the given League.

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 Administrator out.
viewAccount	Administrator		Allows the Administrator to view their account details.
modifyAccount	Administrator	firstName lastName password email address	Allows the Administrator to change their account details.
viewLeagues	List<League>		Displays the leagues.
viewTeams	List<Team>	League	Displays the teams in the given League.
editLeague	League	leagueName List<Team> sport	Allows the Administrator to change a specified league's details.
createLeague	League	leagueName List<Team> sport	Allows the Administrator to create a league with the specified details.
appointCaptain	Captain	captainName teamName league	Allows the Administrator to appoing a player from a team to be that team's captain.
createTeam	Team	teamName	Allows the

		List<Player> league teamCaptain	Administrator to create a team by specifying its name, the players in the team, the league it's in, and its team captain.
editTeam	Team	teamName List<Player> league teamCaptain.	Allows the Administrator to modify a team.
createLeagueSchedule	Schedule	league List<sportsVenue>	Allows the Administrator to create a schedule to be played by a league.
createSportsVenue		address	Allows the Administrator to create a sports venue for matches to be held at.
reportMatch	gameStats	score1 score2 winner	Allows the Administrator to report the results of a match if there's a dispute among team captains.

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	Registers a new authenticated user with the given credentials.

		major email address	
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

Operation	*Return*	*Arguments*	*Description*
logout			Logs the Administrator out.
viewAccount	Administrator		Allows the Administrator to view their account details.
modifyAccount	Administrator	firstName lastName password email address	Allows the Administrator to change their account details.
viewLeagues	List<League>		Displays the leagues.
viewTeams	List<Team>	League	Displays the teams in the given League.
editLeague	League	leagueName List<Team> sport	Allows the Administrator to change a specified league's details.
createLeague	League	leagueName List<Team> sport	Allows the Administrator to create a league with the specified details.
appointCaptain	Captain	captainName teamName league	Allows the Administrator to appoing a player from a team to be that team's captain.
createTeam	Team	teamName	Allows the

		List<Player> league teamCaptain	Administrator to create a team by specifying its name, the players in the team, the league it's in, and its team captain.
editTeam	Team	teamName List<Player> league teamCaptain.	Allows the Administrator to modify a team.
createLeagueSchedule	Schedule	league List<sportsVenue>	Allows the Administrator to create a schedule to be played by a league.
createSportsVenue		address	Allows the Administrator to create a sports venue for matches to be held at.
reportMatch	gameStats	score1 score2 winner	Allows the Administrator to report the results of a match if there's a dispute among team captains.

2.2.7 Object Layer (general)

Operation	*Return*	*Arguments*	*Description*
CreateLeague		-League name -Minimum # of teams -Maximum # of teams -Minimum # of team members -Maximum # of team members -Match Rules	Creates an object of type League with the specified parameters.

		-Indoor/Outdoor	
SaveLeague			Saves a newly created League object or modifications to an existing League object.
EditLeague	League	-Minimum # of teams -Maximum # of teams -Minimum # of team members -Maximum # of team members -Match Rules -Indoor/Outdoor	Modify an existing League object's attributes.
DeleteLeague		League name	Deletes a league object.
CreateTeam		Team name Team captain	Creates an object of type Team with the specified parameters.
SaveTeam			Saves a newly created Team object or modifications to an existing Team object.
EditTeam	Team		Modify an existing Team object's attributes.
DeleteTeam		Team name	Deletes a Tteam object.
CreatePlayer		username firstName lastName resident address Password Student number College major Email address	Creates an object of type Player with the specified parameters.

SavePlayer			Saves a newly created Player object or modifications to an existing Player object.
EditPlayer	Player		Modify an existing Player object's attributes.
DeletePlayer		Student number	Deletes a Player object.
CreateVenue		Venue name Address Indoor/Outdoor	Creates an object of type Venue with the specified parameters.
SaveVenue			Saves a newly created Venue object or modifications to an existing Venue object.
EditVenue	Venue		Modify an existing Venue object's attributes.
DeleteVenue		Venue name	Deletes a Venue object.
CreateAdministrator		firstName lastName Username password	Creates an object of type Administrator with the specified parameters.
SaveAdministrator			Saves a newly created Administrator object or modifications to an existing Administrator object.
EditAdministrator	Administrator		Modify an existing Administrator object's attributes.
DeleteAdministrator			Deletes an Administrator object.

2.2.8 Persistence Layer

Operation	*Return*	*Arguments*	*Description*
ReturnLeague	League	League ID	Returns a League object from the database.
SaveLeague		League ID	Saves a League object to the database.
DeleteLeague		League ID	Deletes a League object from the database.
ReturnTeam	Team	Team ID	Returns a Team object from the database.
SaveTeam		Team ID	Saves a Team object to the database.
DeleteTeam		Team ID	Deletes a Team object from the database.
ReturnPlayer	Player	Player ID	Returns a Player object from the database.
SavePlayer		Player ID	Saves a Player object to the database.
DeletePlayer		Player	Deletes a Player object from the database.
ReturnVenue	Venue	Venue ID	Returns a Venue object from the database.
SaveVenue		Venue ID	Saves a Venue object to the

			database.
DeleteVenue		Venue ID	Deletes a Venue object from the database.
ReturnAdministrator	Administrator	Administrator ID	Returns an Administrator object from the database.
SaveAdministrator		Administrator ID	Saves an Administrator object to the database.
DeleteAdministrator		Administrator ID	Deletes an Administrator object from the database.

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 subsystems 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

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 allows for a dynamic application to be distributed through the web. The JDBC uses 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 requires access at fine levels of detail 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 scores 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.

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.

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