Software Requirements Specification

for

Camper Class Rank & Sort System

Version 1.0 approved

Prepared by Adam Wall, Christopher Hanson, Gabriel Jardio, Patrick Perez

CSCI 5801

02-08-2020

**Table of Contents**

Table of Contents.................................................................................... i

Revision History...................................................................................... 3

1. Introduction....................................................................................... 3

1.1 Purpose........................................................................................... 3

1.2 Document Conventions..................................................................... 3

1.3 Intended Audience and Reading Suggestions..................................... 3

1.4 Product Scope.................................................................................. 4

1.5 References....................................................................................... 4

2. Overall Description.......................................................................... 4

2.1 Product Perspective.......................................................................... 4

2.2 Product Functions............................................................................ 4

2.3 User Classes and Characteristics....................................................... 5

2.4 Operating Environment..................................................................... 5

2.5 Design and Implementation Constraints............................................. 5

2.6 User Documentation......................................................................... 5

2.7 Assumptions and Dependencies........................................................ 6

3. External Interface Requirements..................................................... 6

3.1 User Interfaces................................................................................. 6

3.2 Hardware Interfaces......................................................................... 10

3.3 Software Interfaces........................................................................... 11

3.4 Communications Interfaces............................................................... 11

4. System Features................................................................................. 11

4.1 Management of Classes and Students by Administration…………… 11

4.2 Use of sorting system to create classes for students…………..….. 12

4.3 Schedule System that involves no class overlap…………………… 13

4.4 Camper class-ranking based on preference………………………… 14

5. Other Nonfunctional Requirements................................................ 15

5.1 Performance Requirements............................................................... 15

5.2 Safety Requirements......................................................................... 15

5.3 Security Requirements...................................................................... 16

5.4 Software Quality Attributes............................................................... 16

5.5 Business Rules................................................................................. 16

6. Other Requirements.......................................................................... 16

Appendix A: Glossary............................................................................. 16

Appendix B: Analysis Models................................................................ 17

Appendix C: To Be Determined List..................................................... 17

Appendix D: Questions ……………………………………………………17

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| **Class Rank & Sort System (CRSS)** | 02/02/2020 | Initial Version | 0.1. |
|  | 02/08/2020 | Added illustrations | 0.2. |
|  | 02/09/2020 | Finalized initial draft for submission | 1.0 |

# **1.** **Introduction**

## **1.1** **Purpose**

The purpose of this document is to present a detailed description of the software Class Rank & Sort System (referred to as CRSS going forward). It will explain the purpose and features of the software, the interfaces of the software, what the software will do and the constraints under which it must operate. This document is intended for users of the software and also potential developers.

## **1.2** **Document Conventions**

This Document was created based on the IEEE template for System Requirement Specification Documents

## **1.3** **Intended Audience and Reading Suggestions**

* Camp Administrators who want to run the system to assign schedules to campers
* Instructors who want a better understanding of how the system runs and how to create new classes.
* Software Developers working on improving the project further

## **1.4** **Product Scope**

CRSS is a tool that allows the organization and assignment of campers to classes offered by the camp. Campers are allowed to submit a ranking of classes that they would prefer to enroll in that are provided by instructors. The system employs a ranked voting algorithm to best accommodate all campers. The system allows administrators the tools to analyze enrollment and make manual changes to enrollment

## **1.5** **References**

* HW 1 SRS Reference sheet
* Software Requirements Specification Document Template
* Use Case Structured Specification Template Example

# **2.** **Overall Description**

## **2.1** **Product Perspective**

This product is a self-contained system for the running and administration of camp classes. It provides an interface for campers to rank classes, instructors to create classes and schedules, and for administrators to manage enrollment.

## **2.2** **Product Functions**

* Let campers rank the classes they want to take
* Let the administrator or their assistant input class information into system
* Make a schedule for campers based on their rankings and registration time
* Let the administrator change a camper’s schedule manually after it has been created
* Let campers view and print their personal schedules
* Give administrators the ability to analyze schedules
* Allow administrators to address schedule block issues, settle class duplicates, etc.
* Allow administrators to add new classes
* Allow instructors to view and print class rosters
* Allow administrators the ability to lock down the system to stop additional campers from submitting rankings.

## 

## **2.3** **User Classes and Characteristics**

* Campers - only access to one function, submit class ranking. (Question: should campers be able to change their answers after submitting a class ranking?) Can view & print own schedules.
* Instructors - least important, inputs very little into the system directly. Only tells the Administrator or assistant what information they want to put into the system. Can view and print a class list, and can view any camper’s schedule. (Question: can they only view their own class list, or all classes available?)
* Administrator - most important, can access the system before and after registration time and has access to most functions in software. This includes manually removing or adding students to classes and kickstarting the sorting of classes.
* Administrative assistant - only access to one function, inputting class information into the system

## **2.4** **Operating Environment**

* The software must be able to operate on any home computer or device (Question: is the software is going to be accessed through a program/app or a website?)
* What operating system will this software run on? Windows? Linux?
* If the software is a program, it should be able to operate as a stand alone program/app without disrupting any other programs/apps

## **2.5** **Design and Implementation Constraints**

* Database storing class info, student rankings, and the schedule made should have enough space to store said data
* Should be compatible with any web browser (if software is web-accessible)
* Needs basic security so no people outside the camp system can’t access the software
* The system should be able to handle various input loads in a short amount of time without corrupting data in the system
* Question: Should the user be able to access the software with any language?

## **2.6** **User Documentation**

* Implement tutorials in the software that campers have to use before they have full access to the system
* Any user documentation should be implemented in the software itself

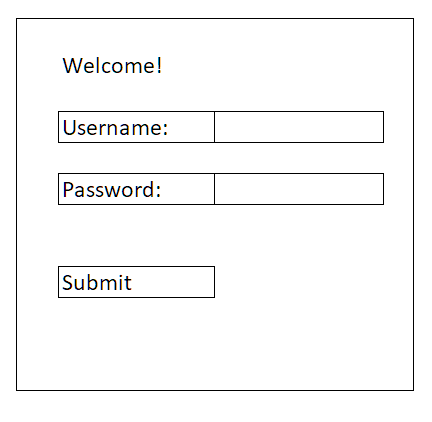
## **2.7** **Assumptions and Dependencies**

* Assume that a menu is available that the rest of this software can be built on
* Question: What assumptions can be made about the platform this project will be developed in?

# **3.** **External Interface Requirements**

## **3.1** **User Interfaces**

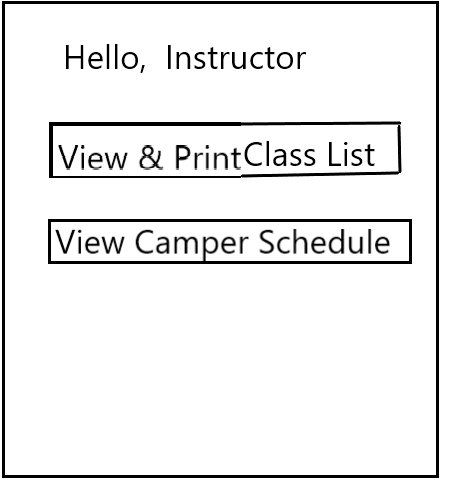
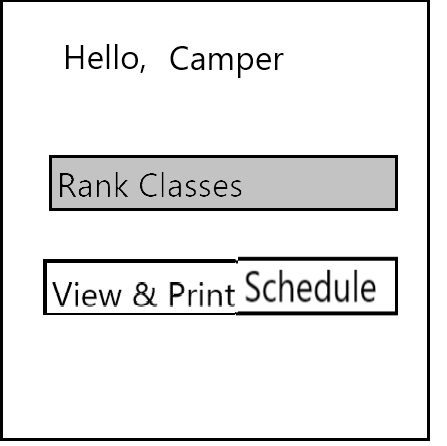
Login:



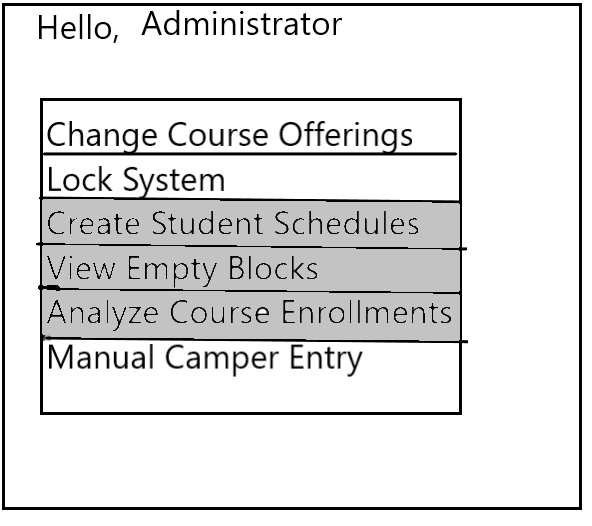
Home Pages:

Functions which can’t be accessed have their buttons greyed out

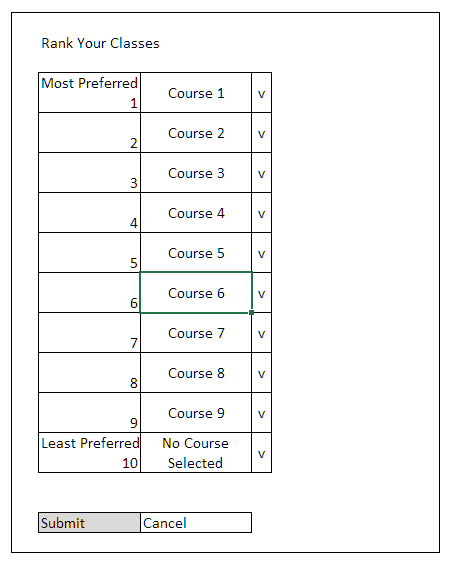
Campers: Instructors:



Administrator (before system has been locked, and student schedules created):

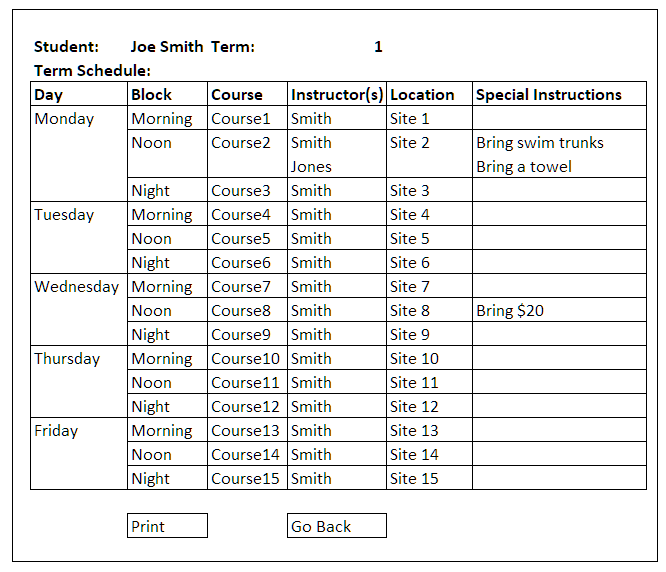


Ranking Mechanism:

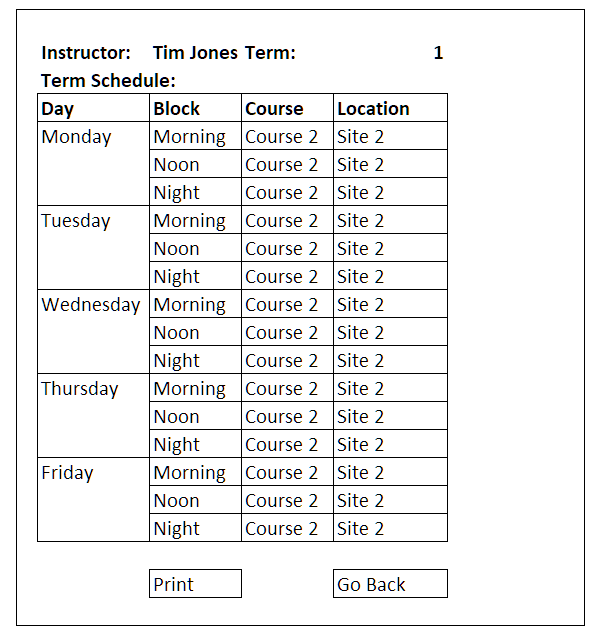


View/Print mechanism:

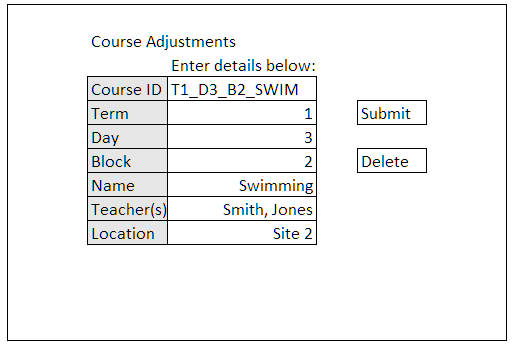
Student (is also instructor-accessible, with the print option greyed out):



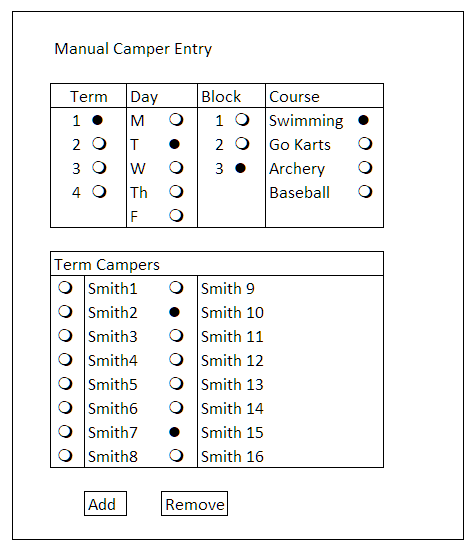
Instructor:



Course Entry/Adjustment Mechanism:



Manual Camper Entry Mechanism:



## **3.2** **Hardware Interfaces**

**Questions:**

* Will this software be used as an application for mobile devices or will it be used exclusively on a desktop?
* Will the analysis and student sorting portions of the program run on a local device, or will some form of cloud computing be involved?

## 

## **3.3** **Software Interfaces**

**Questions:**

* What sort of database will this program’s information be stored on? A pre-existing one, or does one need to be constructed?
* What sort of server will the online portion of this program be running on?

## **3.4** **Communications Interfaces**

**Questions:**

* How will students and staff be notified when the schedule has been made? Should this be done outside of the application?
* What level of network and network security is needed for this program?

# **4.** **System Features**

## **4.1** **Management of classes and students by administration**

4.1.1 Description and Priority

This feature includes class enrollment analysis, which will better help and allow the camp administrators to make better decisions involving class offerings. This feature also includes introducing or modifying class offerings for camp administrators, which campers will be able look and browse through as they make their class rankings. This feature is of high priority, as without the ability to input/modify class offerings, there will be classes for campers to look through as they make their rankings, making the system non-functional. Also the ability to browse through class enrollment statistics will enable administrators to cut out less popular classes and introduce new classes, improving the appeal of the camp.

4.1.2 Stimulus/Response Sequences

For analyzing course enrollment, the administrator will select the analyze course enrollment analysis button on the menu system and the system will respond by displaying the statistical results on the screen. For introducing or modifying class offerings, the administrator will select the change course offerings button on the menu system and the system will respond by opening the course entry/adjustment mechanism.

REQ-1: MENU SYSTEM

A menu system must be implemented allowing users to select the desired feature.

REQ-2: SCHEDULE GENERATION

The software must be able to generate schedules so that course enrollment statistics can be made and analyzed.

REQ-3: CLASS ASSIGNMENT DETERMINATION

The software must be able to determine if initial class assignments have been made to see if course enrollment can even be analyzed.

REQ-4: COURSE ENTRY/ADJUSTMENT MECHANISM

Details - TBD

**Questions:**

* Should we mark off classes from dropdown when a class has been selected in another ranking?
* Should the save function be able to be used when all classes have been selected or can a user save part of a form?
* Should the saving of ranking and submitting of ranking to a database be the same button or a different button?

## **4.2** **Use of sorting system to create classes for students**

4.2.1 Description and Priority

This feature takes the database created by students and classes and sorts them into schedules. The whole system has a high priority as well as it is the main function of the software. This feature will also allow the admin to see how sorting was completed. Allowing an admin to see how the sorting was completed would be useful so that it is known how the sorting was completed if there were any issues. Students and staff will also be emailed after the process has been completed if the email of each student or staff was provided.

4.2.2 Stimulus/Response Sequences

The first step to run this

1. All students that want to be sorted must save a ranking of classes to the software (UC\_8)
2. Admin selects an option to lock class ranking and students out of the software (UC\_9)
3. Admin kickstarts process of sorting (UC\_7)
4. Admin is sent a report and can see analysis of software
5. Students and Teachers are notified of the sorting process has been completed

4.2.3 Functional Requirements

REQ-1: Note when all students have locked in rankings for classes (Seq. 1)

If all students have not submitted rankings, notify administrators of this but still allow for the kickstarting (or locking) of the program to take place.

REQ-2: Administration has an option (button) to lockdown all ranking from students, this lockdown process also stops all students from using the ranking classes option (Seq 2 and 3)

REQ-3: Admin is given analysis and reasoning for the sorting that was completed

Details: TBD

**Questions**

* Is there a list of every student that needs to fill out the form to ensure that everyone has filled out the form?

## **4.3** **Schedule System that involves no class overlap**

4.3.1 Description and Priority

This feature describes the ability for the software to create schedules for students that have no overlapping classes, as well as the ability for the software to determine if a class offering that an administrator or administrative assistant has inputted is invalid due to scheduling conflicts. This feature is of medium priority because it will allow for a well run and coordinated camp.

4.3.2 Stimulus/Response Sequences

For creating schedules that have no overlapping classes, the administrator will select the create student schedules button and the software will respond by creating non overlapping schedules for each camper and saving it on the system. For determining if a class offering is invalid due to scheduling conflicts, the administrator will submit a class offering and if it is invalid, the software will respond by displaying an error and not saving the class into the system.

4.3.3 Functional Requirements

REQ-1: MENU SYSTEM

A menu system must be implemented allowing users to select the desired feature.

REQ-2: SCHEDULE GENERATION

The software must be able to create schedules for students that have no overlapping classes.

REQ-3: COURSE ENTRY/ADJUSTMENT MECHANISM

After the user submits a course offering, the software must look through already offered classes and see if any are during the same time as the one being offered.

## **4.4** **Camper class-ranking based on preference**

4.4.1 Description and Priority

This feature allows students to rank classes by interest. A form will be given to students that has a menu with top choices that can be selected by students. This system as a whole has a very high priority as it is the main function of the system. Students must have a ranking of classes in order for other systems of the software to be performed.

4.4.2 Stimulus/Response Sequences

Prior to the feature of ranking classes, students must use the log in page to get onto their account. If a student hasn’t yet chosen their rankings, they will be presented with a menu to select if they want to start ranking classes. If they have completed this step then they will be sent to another function (UC\_2).

When students reach the ranking page to select classes. Ten dropdowns will be presented to students will all classes added to each dropdown. Students must select classes they want to take. When a student has finished selecting ten classes, they can then choose to save classes (UC\_8). The user will be flagged if the same class has been selected more than once or not all ten spots have been filled out. Once all ten classes (with no repeats) have been selected, the system saves the users choices and sends them back to a menu.

4.4.3 Functional Requirements

REQ-1: CLASS SELECTION START

On the menu page for students, a button must be created to send a user from the menu to the class selection system

REQ-2: LIST OF CLASSES AND DROPDOWNS

A list of all known classes must be provided to the system to create dropdown boxes of all classes. These dropdown boxes are saved when the project is saved.

REQ-3: SAVE/SUBMIT BUTTON

A save button must be implemented so that when pressed and all classes have been selected, the choices will be sent to a database will all students information and rankings

REQ-4: CORRECT CLASS SELECTION CHECK

The save button must also have correct testing so that a flag will be present if a student enters in classes incorrectly (more than one class in ranking.. etc)

**Questions:**

* Should save button and submit button be the same?

# **5.** **Other Nonfunctional Requirements**

## **5.1** **Performance Requirements**

In order to preserve fairness, it is vital that the time of each camper’s ranking submission be recorded accurately. Therefore the CRSS system must be able to receive this information rapidly after a camper has decided to submit their preferences.

**Questions:**

* Is there a timeline on how long the sorting process must take?
* Is there a timeline on how long the saving process of class must take?
* What sort of extraneous code can we cut with showing students in classes to help increase performance?

## **5.2** **Safety Requirements**

Only way safety could be at risk with this software is if personal information is requested from users. Then security issues could cause concern for safety. (See section 5.3)

## **5.3** **Security Requirements**

* All users will have a username and password assigned to them.
* Only verified users will have access to the system.
* The administrator has higher functionality than other actors.

## **5.4** **Software Quality Attributes**

* Usability is very important for the students and teachers so that this process can be done quickly and effectively
* Admin functions can be more complicated as it will provide additional features to help with correctness and adaptability

## **5.5** **Business Rules**

* The program can only run sorting software when all students have filled out their class rankings

Questions:

* As stated in previous sections - should we have the software prevent the sorting from being done until all students have filled out the ranking form?

# **6.** **Other Requirements**

**Appendix A: Glossary**

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

* Administrator
* Camper
* Instructor
* Blocks
* Session
* Class

**Appendix B: Analysis Models**

TBD if needed

**Appendix C: To Be Determined List**

* Most of section 5 was not explicitly stated but suggestions on what would be best and some elaboration would be needed to make sure it is the best option
* Section 4.2.4
* Section 4.1.3
* Appendix B

**Appendix D: Questions**

* Is there a specific scheme for IDs that the use cases should follow?
* Will this software be under any specific license?
* Can campers can change answers after submitting their first ranking?
* What is the minimum hardware interface needed for this program to run?
* How fast does the class ranking need to be.