**Software Requirements Specification**

**for**

**Class Rank & Sort System (CRSS)**

**Version \_\_\_\_\_\_\_**

**Prepared by:**

**Sami Frank**

**Jake Waro**

**Allison Miller**

**Declan Buhrsmith**

**University of Minnesota – CSci 5801**

**February 3, 2020**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction**

1.1 Purpose 1

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

1.4 Product Scope

1.5 References

**2. Overall Description**

2.1 Product Perspective

2.2 Product Functions

2.3 User Classes and Characteristics

2.4 Operating Environment

2.5 Design and Implementation Constraints

2.6 User Documentation

2.7 Assumptions and Dependencies

**3. External Interface Requirements**

3.1 User Interfaces

3.2 Hardware Interfaces

3.3 Software Interfaces

3.4 Communications Interfaces

**4. System Features**

4.1 Classes to search

4.2 Assign section numbers

4.3 Lock camper rankings

4.4 Login to system

4.5 Create camper schedule

4.6 Rank camper selections

4.7 Manually add, cancel, update schedules

4.8 Add class details

4.9 **TBD**

4.10 **TBD**

**5. Other Nonfunctional Requirements**

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

5.5 Business Rules

**6. Other Requirements**

**Appendix A: Glossary**

**Appendix B: To Be Determined List**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Jake | 2.3.20 | Initialize document | 0.001 |
| Jake | 2.4.20 | Product Perspective | 0.002 |
| Sami | 2.4.20 | External Interfaces (sec 3) | 0.003 |
| Jake | 2.5.20 | Complete Section 2 | 0.004 |
| Jake, Sami, Declan | 2.6.20 | Merge Document | 0.005 |
| Sami | 2.8.20 | Questions for sec 3. add to appendix | 0.006 |
| Jake | 2.8.20 | Create consistent format, review document for logical and grammatical errors | 0.007 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

*<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>*

## Document Conventions

*<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>*

## Intended Audience and Reading Suggestions

*<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>*

## Product Scope

*<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>*

## References

*<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>*

# Overall Description

## 2.1 Product Perspective

This product (CRSS) is a new, self-contained product. It is being used as a stand-alone system for a summer camp to build campers’ schedules. This product is not part of a larger system.

Outstanding questions:

*Q 2.1.1* - Will the system need to integrate with any existing software / hardware systems the camp already employs?

*<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>*

## 2.2 Product Functions

**All Actors**:

* Login: user logs into the system using a username and password given to them that was generated by the system.

**Campers**:

* Select class: campers can select from the list of available courses that they want to rank.
* Rank: for the selected class, camper assigns a rank to the course.
* View camper schedule: displays the campers class schedule.
* Print camper schedule: prints a copy of the campers schedule.

**Instructors**:

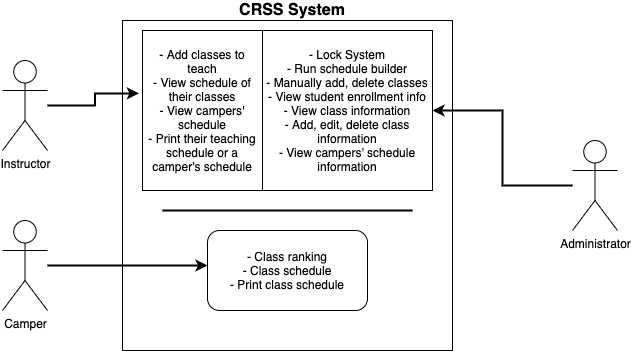
* Input classes: instructors are able to input the classes they are going to teach for each summer session.
* View instructor schedule: display a schedule of the class that the instructor teaches.
* Print instructor schedule: prints the specific instructor’s schedule.
* View camper schedule: displays the campers class schedule.
* Print camper schedule: prints a copy of the campers schedule.

**Administrators**:

* Input class information: Administrators can input the camp session class information into the system. This information consists of:
  + Class titles and section numbers
  + Class times
  + Number of seats per class
  + Class information
* Save class information: system saves the information that was input for the class information.
* Assign section numbers: assigns a unique section number to each class that has been added to the system by the instructors.
* Build schedules: takes the campers’ rankings and creates schedules that incorporates the ranking preferences.
* Lock System: Locks the system for the purposes of assigning section numbers and building class schedules. The system is locked once per camp session and will not be unlocked afterwards.

**System**

* Determine user: the system verifies the credentials from the login screen. If invalid, returns a message describing the credentials as invalid. If valid, allows the user to pass to the respective next screen.



## Outstanding questions:

*Q 2.2.1* - should the system save campers’ rankings after every ranking? Or, will the camper only be able to submit their rankings once, and upon submission, the rankings will be saved?

*Q 2.2.2* - should the class information input menu contain functionality to input class titles and section numbers, class times, number of seats, and other information all on one screen? Or, should this functionality be split into different menus?

## *Q 2.2.3* - how should the assignment of section numbers be triggered? Is this something that gets explicitly triggered by the director? Or, will this process be encapsulated within the build schedules functionality?

## 2.3 User Classes and Characteristics

The three main actors of this product will be **campers, instructors,** and **administrators**. The users are expected to be competent in basic single-page application navigation, that is, they are able to login to a system, and navigate at that first level of the application by using different interactable tools.

* **Camper** – a camper will be a kid in their youth, likely between the ages of 6 – 17. They will be enrolled in the camp. This may or may not be their first experience at Camp Voyager. Prior to the camp session, campers will make use of the class ranking functionality. During the camp session, campers will make use of the schedule viewing functionality.
* **Instructor** – instructors are expected to be adults 18 years of age or older. They will be teaching summer classes such as swimming or climbing. The nature of their work will likely not involve much interaction with technology. Prior to the camp session, instructors will make use of the functionality to input classes they will teach. During the camp session, they will make use of the functionality to view their class schedules as well as campers’ schedules.
* **Administrator / Director** – administrators will be camp employees either in higher positions such as the camp director or system administrator. Their use of the system will be more of a management focus, and they will likely use the system more than instructors. Administrator users will have privileges to all of the functionality listed in section 2.2. Prior to camp, administrators will make use of the functionality to input class information, build schedules, and manually add, edit, or delete class schedule information. During the camp session, administrators will make use of the functionality to view instructors’ and campers’ schedules as well as the functionality to manually add, edit, or delete class schedule information.

## 2.4 Operating Environment

Using the newest JDK and SDK released by Oracle, Java SE 12, the following platforms and operating systems will have the capabilities to utilize the product:

* Windows:
  + Windows Server 2019, (64-bit)
  + Windows Server 2016, (64-bit)
  + Windows Server 2012 R2, (64-bit)
  + Windows Server 2012, (64-bit)
  + Windows 10, (64-bit)
  + Windows 8.x, (64-bit)
  + Windows 7 version SP1, (64-bit)
* Linux:
  + Oracle Linux 7.x, (64-bit)
  + Oracle Linux 6.x, (64-bit)
  + Red Hat Enterprise Linux 7.x, (64-bit)
  + Red Hat Enterprise Linux 6.x, (64-bit)
  + Unbuntu Linux 18.10, (64-bit)
  + Unbuntu Linux 18.04 LTS, (64-bit)
* macOS
  + 10.12+ excluding version 10.15, x64
* Java virtual machines (JVMs):
  + Must be 64-bit

Outstanding questions:

*Q 2.4.1* - Are there any specific requirements for a programming language to use for this product?

## 2.5 Design and Implementation Constraints

There are no foreseeable corporate, regulatory, or hardware limitations of notice.

Outstanding questions:

*Q 2.5.1* - Will CRSS need to interface with any other applications?

*Q 2.5.2* -How will the data be stored? Should the system use cloud technologies to store data, or have a local database?

*Q 2.5.3* - Are there any language requirements? Are there any other forms of communication accessibility to be considered?

*Q 2.5.4* - How should the product handle user security to sign into the system? If using an API to handle security, will the functionality be placed on the API or within the product’s code to verify a user?

*Q 2.5.5* - Who will be responsible for maintaining the software after it is delivered?

## 2.6 User Documentation

Before building a plan on the schedule of delivering CRSS, we will need to understand all deliverables in order to build a timeline.

Outstanding questions:

*Q 2.6.1* - Are there any manual / reference materials requested? These materials would require time apart from building the program to prepare. Knowing if any tutorials or additional documentation is needed will help determine product timelines.

## 2.7 Assumptions and Dependencies

Assumptions

1. This document has been prepared based on an initial statement from the client about the product requested. We have not had a chance to communicate with the client, so we have listed questions in locations throughout the document that require additional information as to not make certain assertions. Appendix B can be referenced to locate sections with outstanding questions.
2. We are building an entirely new system; however, we have made the assumption that we will make use of APIs to aid in our efforts. Types of APIs we may use would consist of tools such as graphics libraries, or a specific database technology.
3. CRSS will use Java SE 12. This is the most recent Java SE release that includes both a published JDK and SDK. However, there are other versions of Java as well as other languages that could be used to build CRSS.
4. CRSS is built on the assumption that the camp and all actors have internet access.

Dependencies

1. With the assumption stated that we will be using APIs as part of our product, there may be specific dependencies for using those tools, such as needing to use a specific version, or simply noting that the module needs to be installed. We do not have explicit dependencies to list for APIs, but rather, the assumption that we will have dependencies according to APIs the product would use.

# External Interface Requirements

## 3.1 User Interfaces

While our team has not conducted elicitation interviews with Camp Voyager to discuss the user interfaces of the Class Rank & Sort System (CRSS), there are some basic functionalities that can be discussed.

There will be security features such as a login screen to ensure each user has access to the correct user interface (I.e. a camper doesn’t see action items that only an administrator can execute).

Some standard buttons and functions for all users will include logging out, saving and canceling certain actions.

There will be detailed error messages to constrain a user from not following guidelines set in Camp Voyagers requirement specification.

* Error messages while ranking:
  + Message that constrains a camper from only ranking 10 classes.
  + Message notifying a camper that they can’t rank the same class title.
* Error message while logging in:
  + Message notifying the user that credentials failed.
* General responsive messages:
  + Message confirming that action has been saved and/or updated.
  + Confirming logout of system.

\* This is not a comprehensive list as our team has not yet completed elicitation interviews.

## 3.2 Hardware Interfaces

The main hardware interface that will be taken into consideration is the connection between the CRSS software and a printer.

* The CRSS software will print a schedule when a user executes a print action.

Outstanding questions:

*Q 3.2.1* - Are there any requirement specifications for the printer?

## 3.3 Software Interfaces

The CRSS software will need a database of all users and their authorization levels. This information will allow the CRSS software to handle which user interface to show a user upon login.

It has not been clarified how this information will be transferred to the CRSS, whether it be an integration between an existing Camp Voyager software, a manual import of needed data, or something else. It has not been clarified how user information will be transferred to the CRSS.

Outstanding questions:

*Q 3.3.1* - How will the CRSS obtain pertinent user information in order to create a basic profile for each user, with access to everything in their authorization level?

*Q 3.3.2* - Will the CRSS have to integrate with an existing Camp Voyager system that obtains rostering for who’s enrolled/staffed at the camp?

*Q 3.3.3* - If the CRSS is not integrating with an existing system, how will / should the CRSS obtain this pertinent user information?

## 3.4 Communications Interfaces

There will need to be clarification on how a user’s username and password is relayed to the user. It begs the question if the CRSS will integrate with an existing Camp Voyager software to handle communication or if the CRSS will need more user information like an email address or something else to handle communication of login.

If the CRSS is integrating with existing Camp Voyager software:

Our team will need more information on what functionalities their existing software has for 3rd parties, and where in the existing software the CRSS should route this information.

If the CRSS is handling the communication of login information to the user:

There will be sensitive information communicated, our team will need more information from Camp Voyager on the level of security the CRSS should ensure.

Our team will also need to know how to format the message, and if there should be more information besides the username and password.

Outstanding questions:

* It is unclear whether or not the CRSS is communicating to a user their log in information, or if this is handled by Camp Voyager some other way.

*Q 3.4.1* - If the CRSS is handling login information, do we need security features with those credentials?

*Q 3.4.2* - Should it be required that the user changes their password upon first login to the system?

*Q 3.4.3*  - Should there be a time limit from when communication of credentials has been made to the first use of those credentials?

# 4. System Features

## 4.1 Add Classes

|  |  |
| --- | --- |
| **Name** | Add Classes |
| **ID** | inst\_001 |
| **Description** | Instructors can add the classes they will be teaching, to the system |
| **Actors** | Instructors |
| **Organizational Benefits** | Allows for instructors to easily communicate the classes they intend to teach for a given camp session. |
| **Frequency of Use** | Once per session |
| **Triggers** | Instructor logs into the system. |
| **Preconditions** | System has to exist. The classes they will teach have to be valid options for the summer camp. Instructor is logged into the system. System is unlocked. |
| **Postconditions** | The system has the class options that will be available for campers to rank. |
| **Main Course** | 1. Instructor is on landing page where they have options to add new classes 2. Instructors selects to add a new class 3. Display shows fields to populate:    1. Class name    2. Number of students 4. System saves the new class 5. System redirects instructor to be able to add additional classes |
| **Alternate Courses** | AC1 Instructor logs out   1. Instructor returns to login portal (See use case “Login to System”)     AC2 Instructor inputs a class they have already added   1. System displays an error stating the class already exists 2. Returns user to Main Course step 1. |
| **Exceptions** | EX1 System fails to save newly added class   1. System notifies instructor error has occurred in saving the newly added class. 2. Return to Main Course step 1. |

## 4.2 Assign section numbers

|  |  |
| --- | --- |
| **Name** | Assign Section Numbers |
| **ID** | admin\_001 |
| **Description** | The system assigns section numbers to classes that have been added to the system by instructors. |
| **Actors** | Administrators |
| **Organizational Benefits** | Allows for clarity in the schedule by uniquely identifying classes, because multiple instructors may be teaching the same class, just at different times. |
| **Frequency of Use** | Once per session. |
| **Triggers** | Administrator selects to assign section numbers. |
| **Preconditions** | System has to exist. Administrator is logged into the system. The system must be locked. |
| **Postconditions** | Each class offered is now uniquely identified. |
| **Main Course** | 1. Administrator is on landing page where they have options to lock the system. 2. Instructors selects to lock the systems. 3. The instructor then selects to assign section numbers 4. System determines section numbers and saves these to the system. 5. The system displays a list of classes and their respective class information. |
| **Alternate Courses** | AC1 Administrator logs out   1. Administrator returns to login portal (See use case “Login to System”).     AC2 System is already locked   1. Return to Main Course step 3.     AC3 Section numbers have already been assigned and need to be reassigned   1. Return to Administrator’s class editing page (See use case “Manually add, cancel, update schedules”). |
| **Exceptions** | EX2 System fails to save section numbers   1. System notifies administrator error has occurred in saving the section numbers. 2. Return to Main Course step 3.     EX3 System fails to lock   1. System notifies administrator error has occurred in locking the system. 2. Return to Main Course step 2. |

## 4.3 Lock camper rankings

|  |  |
| --- | --- |
| **Name** | Lock Camper Rankings |
| **ID** | admin\_002 |
| **Description** | This action will disable the ability for campers to rank classes. |
| **Actors** | Administrators |
| **Organizational Benefits** | Will allow Camp Voyager to have a static set of rankings to be able to execute scheduling software. |
| **Frequency of Use** | Once per session, before session begins. |
| **Triggers** | Administrator presses a lock button to execute action. |
| **Preconditions** | System has to exist. Administrator must have unique login credentials. Administrator must allow enough time for campers to rank so there’s data to use. |
| **Postconditions** | Campers can no longer rank classes. |
| **Main Course** | 1. System prompts administrator to login.  2. Administrator is on landing page where they have option to lock the system.  3. Instructors selects to lock the systems.  4. A confirmation pops up to confirm system is locked. |
| **Alternate Courses** | AC1 System is already locked  1. Lock option will be unclickable. |
| **Exceptions** | EX1 System fails to lock rankings  1. System notifies administrator error has occurred in locking the rankings.  2. Return to Main Course step 2. |

**4.4 Login to system**

|  |  |
| --- | --- |
| **Name** | Login to System |
| **ID** | user\_001 |
| **Description** | This will be the first action by any user using the CRSS. |
| **Actors** | Administrators, Instructors, Campers. |
| **Organizational Benefits** | Ensures access to only those who have been granted it. Allows for actors to be routed to the correct user interface given their level of authorization. |
| **Frequency of Use** | Campers: multiple times before session starts.  Instructors: multiple times before and during the session.  Administrators: multiple times before and during the session. |
| **Triggers** | Anytime a user wants to access the CRSS. |
| **Preconditions** | System has to exist. User must have valid credentials. |
| **Postconditions** | User will have access to actions within their authorization level. |
| **Main Course** | 1. User navigates to system. 2. User is prompted to login. 3. User has access to actions within their authorization level. |
| **Alternate Courses** | AC1 User was already logged in.   1. Return to Main Course step 3. |
| **Exceptions** | EX1 System fails to interpret users login credentials.   1. Return to Main Course step 2.   *Q 4.4.1* - Would Camp Voyager want a security implementation to limit the amount of times the EX1 flow can happen? If so, how would they like to handle being locked out of the system? |

## 4.5 Create camper schedule

|  |  |
| --- | --- |
| **Name** | Create Camper Schedule (run system) |
| **ID** | admin\_003 |
| **Description** | Once ranking is locked, software is ran to create campers’ schedules. |
| **Actors** | Administrator |
| **Organizational Benefits** | Automates the schedule building process. |
| **Frequency of Use** | Once per session. |
| **Triggers** | Build process action is initiated |
| **Preconditions** | System has to exist. System must be locked. Must have camper data to run. Administrator is logged in. |
| **Postconditions** | Schedules have been created and are available for viewing / printing. |
| **Main Course** | 1. Admin chooses to run the software. 2. System confirms they want to build the schedule. 3. Schedules are created. |
| **Alternate Courses** | AC1 Admin clicks cancel on confirmation   1. System returns to Main Course step 1. |
| **Exceptions** | EX1 System fails to retrieve rankings   1. System returns an error message detailing the error. 2. System returns to Main Course step 1.   EX2 System fails to generate schedules   1. System returns an error message detailing the error. 2. Return to Main Course step 1 |

## 4.6 Rank camper selections

|  |  |
| --- | --- |
| **Name** | Rank Classes |
| **ID** | camper\_001 |
| **Description** | Ability to rank classes according to camper preferences. |
| **Actors** | Camper |
| **Organizational Benefits** | Assists campers in getting schedules they enjoy. |
| **Frequency of Use** | *Q 4.6.1* - once per session?  *Q 4.6.2* - multiple times per session? |
| **Triggers** | Camper signs up for camp. |
| **Preconditions** | System has to exist. Camper is logged into the system. |
| **Postconditions** | Camper has ranked class data on file. |
| **Main Course** | *Q 4.6.3* - If camper hasn’t ranked courses, would the first menu guide them to rank classes?  *Q 4.6.4* - Would camper have to always navigate to certain menu page to rank classes?  *Q 4.6.5* - Does the system save temporary rankings? Or should it only save on submission?   1. Camper selects course and selects a respective rank 2. System saves course/ranking combination 3. System redirects camper to additional courses to rank 4. Camper submits class rankings |
| **Alternate Courses** | AC1 Camper ranks classes but doesn’t save  1. Camper returns to Main Course \_\_\_.  AC2 Camper is not enrolled for summer camp  1. Camper is directed to sign up for summer camp which entails receiving credentials to login to the ranking system.  2. Camper is logged in through sign-up process  3. Return Camper to Main Course Q2/Q3.    AC3 Camper is indifferent to courses    *Q 4.6.6* - Would the camper have to select rankings? If they have no preferences, should they still fill out the rankings?  *Q 4.6.7* - Can the system save a ‘null’ ranking list given no data? |
| **Exceptions** | EX1 System fails to save course ranking  1. System notifies Camper error has occurred in saving the course ranking.  2. Return to Main Course step 1. |

## 4.7 Manually add, cancel, update schedules

|  |  |
| --- | --- |
| **Name** | Manually Add, cancel, update schedules |
| **ID** | admin\_004 |
| **Description** | Administrator is given the ability to make manual changes to schedules of any student. |
| **Actors** | Administrator, Student, Instructor |
| **Organizational Benefits** | Serves as a precaution in case any conflicts arise, in this case they can be direct or indirect to the schedule. |
| **Frequency of Use** | As often as required |
| **Triggers** | A student, for whatever reason, is no longer allowed to take a certain class. |
| **Preconditions** | System must exist. A student must also exist and this student must be taking at least one class. The system must be secure. The Administrator must have their own account credentials. |
| **Postconditions** | The Student will be assigned a different class to the one they were previously enrolled in. |
| **Main Course** | 1. System prompts Admin to login 2. Admin is on landing page where they have options to make a class change 3. Admin selects student whose schedule needs change 4. Display shows:    1. Old class information (Time, Class ID, Schedule)    2. New class information (Time, Class ID, Schedule) 5. System saves the new schedule 6. System notifies all affected parties, the student, the instructor which had its student removed, and the instructor which had the student added |
| **Alternate Courses** | AC1 System determines Admin is logged into site   1. Admin returns to Main Course step 2.     AC2 Admin logs out   1. Admin returns to Main Course step 1.     AC3 Admin enters the same class, one with the same Class ID, the Student is being removed from   1. System displays an error stating these are the same classes 2. Returns user to Main Course step 2. |
| **Exceptions** | EX1 System fails to interpret Admin’s login credentials   1. System notifies Admin an error occurred in validating credentials. 2. Return to Main Course step 1.     EX2 System fails to save revised schedule   1. System notifies Admin error has occurred in saving the revised schedule. 2. Return to Main Course step 2.     EX3 Administrator losing connection to portal while editing schedule  1. System displays internet connection not stable warning upon completion of a form and requires the Admin to submit the form upon stable connection |

## 4.8 Add class details

|  |  |
| --- | --- |
| **Name** | Add class details such as enrollment size, # seats, day, times |
| **ID** | admin\_005 |
| **Description** | The Administrator assigns the metadata that includes the enrollment size, number of seats per class, as well as when the classes are going to happen. |
| **Actors** | Administrators |
| **Organizational Benefits** | Organizes the class registration process and gives all parties involved important information about the classes offered at camp. |
| **Frequency of Use** | Once per class |
| **Triggers** | Administrator assigns metadata |
| **Preconditions** | System must exist. The system must be secure. The Administrator must have their own account credentials. Classes must exist in the system. |
| **Postconditions** | Each class offered now has enrollment size, number of seats, days class is offered, and the time of day it is offered. |
| **Main Course** | 1. System prompts administrator to login 2. Administrator is on landing page where they have options to add class metadata. 3. Administrator sets the metadata and saves their changes to the system.   system.   1. Student’s and Instructor’s will be able to see this information upon registering and viewing class schedules. |
| **Alternate Courses** | AC1 System determines Admin is logged into site   1. Admin returns to Main Course step 2.     AC2 Admin logs out   1. Admin returns to Main Course step 1.     AC3 Admin adjusts metadata for a class that does not exist   1. System displays an error stating this class does not exist in the system 2. Returns user to Main Course step 2. |
| **Exceptions** | EX1 System fails to interpret Admin’s login credentials   1. System notifies Admin an error occurred in validating credentials. 2. Return to Main Course step 1.     EX2 System fails to save revised metadata   1. System notifies Admin error has occurred in saving the metadata. 2. Return to Main Course step 2.     EX3 Administrator losing connection to portal while editing schedule  2. System displays internet connection not stable warning upon completion of a form and requires the Admin to submit the form upon stable connection |

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

*<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

An application is needed to make registering and looking at classes as easy as possible. This can be done via a web application, or a mobile application that runs on both IOS and Android.

One performance requirement is a database large enough and fast enough to store user account information (Usernames and Passwords), the user roles (Camper, Camp Director, or Instructor), and the schedules for both the instructors and the campers.

## 5.2 Safety Requirements

*<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>*

## 5.3 Security Requirements

*<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>*

Base level encryption is a requirement especially given the different account users using this application. Passwords should be considered strong, as defined by,<https://www.webopedia.com/TERM/S/strong_password.html>, and should be mutable given system administrator permission.

## 5.4 Software Quality Attributes

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

The application should be robust, be maintained if any bugs or exploits arise after public release. It should also be easy to use and work without any unexpected behavior, such as, and not limited to: freezing and crashing.

## 5.5 Business Rules

*<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>*

The application should be robust, be maintained if any bugs or exploits arise after public release. It should also be easy to use and work without any unexpected behavior, such as, and not limited to: freezing and crashing.

Campers are given the ability to rank classes, and to view and print their class schedules.

Instructors are given the ability to manually add a class to teach, and edit existing classes.

Camp director’s are given the ability to run the ranking process, manually enter information to class roster, set campers’ schedules, and set enrollment cap after consulting with the specific instructors.

System administrators are given the ability to reassign a Camper to a different class at any time.

# 6. Other Requirements

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

**Appendix A: Glossary**

*CRSS: Class Rank and Sorting System*

*Types of users:*

*1.* *Administrator: mainly the camp director, possibly someone the camp director grants access to this level of authorization.*

*2.* *Instructor: the camp counselors or the system users who run classes.*

*3.* *Camper: the user who will fill out class rankings, they will have the lowest level of authorization.*

*Types of use case ID’s:*

*1.* *User\_#: applies to all users of the CRSS.*

*2.* *Admin\_#: applies to admin level users of the CRSS.*

*3.* *Inst\_#: applies to instructor level users of the CRSS.*

*4.* *Camper\_# applies to camper level users of the CRSS.*

**Appendix B: To Be Determined List**

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*

Functionalities to be determined can be found in sections:

* 2.1
* 2.2
* 2.4
* 2.5
* 2.6
* 3.2
* 3.3
* 3.4