**Software Requirements Specification**

**for**

**Voting System**

**Version 1.00 approved**

**Composed by**

**Sunny Qin (qing0002), Yingjin Zhang (zhan4973)**

**Rex Zhu (zhu00100), Xiaohui Chao (chao0070)**

**University of Minnesota**

**9/26/2019**

**Table of Contents**

**Table of Contents 2**

[**Revision History**](#_heading=h.30j0zll) **3**

[**1. Introduction**](#_heading=h.1fob9te) **4**

[1.1 Purpose](#_heading=h.3znysh7) 4

[1.2 Document Conventions](#_heading=h.2et92p0) 4

[1.3 Intended Audience and Reading Suggestions](#_heading=h.tyjcwt) 4

[1.4 Product Scope](#_heading=h.3dy6vkm) 4

[1.5 References](#_heading=h.1t3h5sf) 4

[**2. Overall Description**](#_heading=h.4d34og8) **5**

[2.1 Product Perspective](#_heading=h.2s8eyo1) 5

[2.2 Product Functions](#_heading=h.17dp8vu) 5

[2.3 User Classes and Characteristics](#_heading=h.3rdcrjn) 6

[2.4 Operating Environment](#_heading=h.26in1rg) 7

[2.5 Design and Implementation Constraints](#_heading=h.lnxbz9) 7

[2.6 User Documentation](#_heading=h.35nkun2) 7

[2.7 Assumptions and Dependencies](#_heading=h.1ksv4uv) 8

[**3. External Interface Requirements**](#_heading=h.44sinio) **8**

[3.1 User Interfaces](#_heading=h.2jxsxqh) 8

[3.2 Hardware Interfaces](#_heading=h.z337ya) 9

[3.3 Software Interfaces](#_heading=h.3j2qqm3) 9

[3.4 Communications Interfaces](#_heading=h.1y810tw) 10

[**4. System Features**](#_heading=h.4i7ojhp) **10**

[4.1](#_heading=h.2xcytpi) Sign up and Log in 10

[4.2 Class Info Set up 12](#_heading=h.3bjmm1uarlyw)

[4.3 Ranking Course 1](#_heading=h.vy0c4nmj4wf7)2

[4.4 Lock Ranking System 1](#_heading=h.sshdo0vjou4y)8

[4.5 Automatically Enroll Class 1](#_heading=h.n55xcitcaqyu)9

[4.6 Generate Camper’s Schedule 2](#_heading=h.1ci93xb)1

[4.7 Generate Report of a Current Class List 22](#_heading=h.bcc3nygyjsx5)

[4.8 Manually Enroll and Drop 2](#_heading=h.oxe9eyn2x6yn)4

[**5. Other Nonfunctional Requirements**](#_heading=h.3whwml4) **27**

[5.1 Performance Requirements](#_heading=h.2bn6wsx) 27

[5.2 Safety Requirements](#_heading=h.qsh70q) 27

[5.3 Security Requirements](#_heading=h.3as4poj) 27

[5.4 Software Quality Attributes](#_heading=h.1pxezwc) 28

[5.5 Business Rules](#_heading=h.49x2ik5) 28

[**6. Other Requirements**](#_heading=h.hm6arzt88cmj) **28**

[**Appendix A: Glossary**](#_heading=h.147n2zr) **28**

[**Appendix B: Analysis Models**](#_heading=h.3o7alnk) **29**

[**Appendix C: To Be Determined List**](#_heading=h.23ckvvd) **29**

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Rex | 09/26/2019 | Initialize the SRS document | 1.00 |
| Rex | 09/20/2019 | Add the approving functions for camp director clients in 1.4. | 1.01 |
| Yingjin | 09/20/2019 | Modify section 4 to set up. | 1.02 |
| Sunny | 09/21/2019 | Fill in 2.6 and 3.1 in detail. | 1.03 |
| Yingjin | 09/21/2019 | Change the structure of section 4. | 1.04 |
| Rex | 09/21/2019 | Update 2.7. | 1.05 |
| Xiaohui | 09/21/2019 | Revise 1.1, 1.5 and 2.1. | 1.05 |
| Sunny | 09/22/2019 | Modify 2.2, 3.1, 4.1, TBD list, review. | 1.06 |
| Yingjin | 09/22/2019 | Insert contents of Appendix B instead of being empty. | 1.07 |
| Yingjin | 09/22/2019 | Put forward some questions in Appendix C about the Hardware Interface section to explain the undeclared information instead of providing wide information. | 1.08 |
| Sunny | 09/22/2019 | Elaborate on 2.3 to detail the characteristic of camper director level by inputting some use cases. | 1.09 |
| Rex | 09/22/2019 | Specify the Hardware Interface to concrete implementation instead of a confusing definition. | 1.10 |
| Sunny | 09/22/2019 | Retrieve the source of reference and put the links up into *Reference* section. | 1.11 |
| Rex | 09/22/2019 | Generate a Use Case diagram and plug it into doc. | 1.12 |
| Sunny | 09/22/2019 | Reclaim the functions in section 2.2 in detail by Use Case diagram. | 1.13 |
| Yingjin | 09/22/2019 | Improve the functional and non-functional cases by reformatting the content and inserting some use cases analysis. | 1.14 |
| Xiaohui | 09/23/2019 | Adjust the style of SRS and check grammar faults. | 1.14 |
| Sunny, Yingjin | 09/23/2019 | Have a final review of the SRS. | 1.14 |
| Rex | 09/23/2019 | Rematch the index of SRS and push the SRS to GitHub. | 1.14 |

# Introduction

## Purpose

The Software Requirement Specification comprises of a concrete information about Voting System which includes its intended use, architecture design, deployment method and other relevant resources. This document faces users and developers interacted with Voting System.

## Document Conventions

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

Documents.

## Intended Audience and Reading Suggestions

The target readers for this Software Requirements Specification (SRS) document are users, developers, managers, and contractors. This SRS document comprises the logic and the structure of this system. Also, this SRS document introduces the concrete procedures for deploying the CRSS. Specifically, in the following document, it will include general overview(chapter 2), function features(chapter 4), non-functional requirements(chapter 5) and external interfaces (chapter 3).

Reading the document, readers can follow the sequence of Chapter 1->Chapter 2->Chapter 4->Chapter 5-> Chapter 3->Chapter 6 and Appendix.

## Product Scope

Universities and colleges offer a variety of camp courses in a certain period. The product helps courses to be well organized for both campers, instructors, and administrators is complex. Under this circumstance, the Class Rank and Sort System (CRSS) is designed to help universities and colleges to organize and manage courses for campers, instructors and help administrators manage the courses. Campers can use this system to get enrolled in their preferred courses while instructors may be able to check the details of his classes.

## References

*Textbook:*

*Software Engineering, Ian Sommerville, Tenth Edition, ISBN 10: 0-13-394303-8, ISBN*

*13: 978-0-13-394303-0*

*System Requirement Specification:*

[*https://github.umn.edu/umn-csci-5801-f19/repo-Team12/blob/master/Homework1/docs/CSci5801HW1\_Fall2019\_v1.pdf*](https://github.umn.edu/umn-csci-5801-f19/repo-Team12/blob/master/Homework1/docs/CSci5801HW1_Fall2019_v1.pdf)

*Use Case Link:*

[*https://github.umn.edu/umn-csci-5801-f19/repo-Team12/blob/master/Homework1/docs/Use%20case%20sorted%20version\_draft1.pdf*](https://github.umn.edu/umn-csci-5801-f19/repo-Team12/blob/master/Homework1/docs/Use%20case%20sorted%20version_draft1.pdf)

*Software Requirements Specification for Gephi:*

[*https://canvas.umn.edu/courses/134519/pages/software-requirements-specification-supporting-documents?module\_item\_id=2883887*](https://canvas.umn.edu/courses/134519/pages/software-requirements-specification-supporting-documents?module_item_id=2883887)

*Software Requirements Specification for Web App:*

[*https://canvas.umn.edu/courses/134519/pages/software-requirements-specification-supporting-documents?module\_item\_id=2883887*](https://canvas.umn.edu/courses/134519/pages/software-requirements-specification-supporting-documents?module_item_id=2883887)

*Priority Levels:*

[*http://www.jumpmind.com/services/support/priority-levels*](http://www.jumpmind.com/services/support/priority-levels)

# Overall Description

## Product Perspective

The Class Rank & Sort System is a useful, self-contained product. It is designed for 3 user groups, i.e. campers, instructors, and camp directors. For campers, this system is supposed to be used after the camper decides to join the camp. It allows campers to submit the course ranking and check the course schedule. For instructors, before teaching their course, they need to check their class list and each camper’s course schedule. For camp directors, they need this system to set up course info before the campers submit their ranking forms. Also, this system helps the camp director to manage the course enrollment process both automatically and manually. It is very important for all three user groups. All the functions mentioned above can be accomplished by this system itself.

## Product Functions

Major functions’ summary:

* Create an account: create a new account
* Login: log in the system with username and password
* Set up the class info: Write the class information to the system
* Rank class: rank courses, save and submit
* Run automatic enrollment: Automatically enroll campers in classes
* Edit class roster: manually add/remove campers to/from classes
* Generate -> class list: generate a current list of course and display
* Generate -> camper’s schedule: generate a schedule for a particular student and display
* Print -> print current page out

The diagram below introduces the basic functions of the CRSS system.

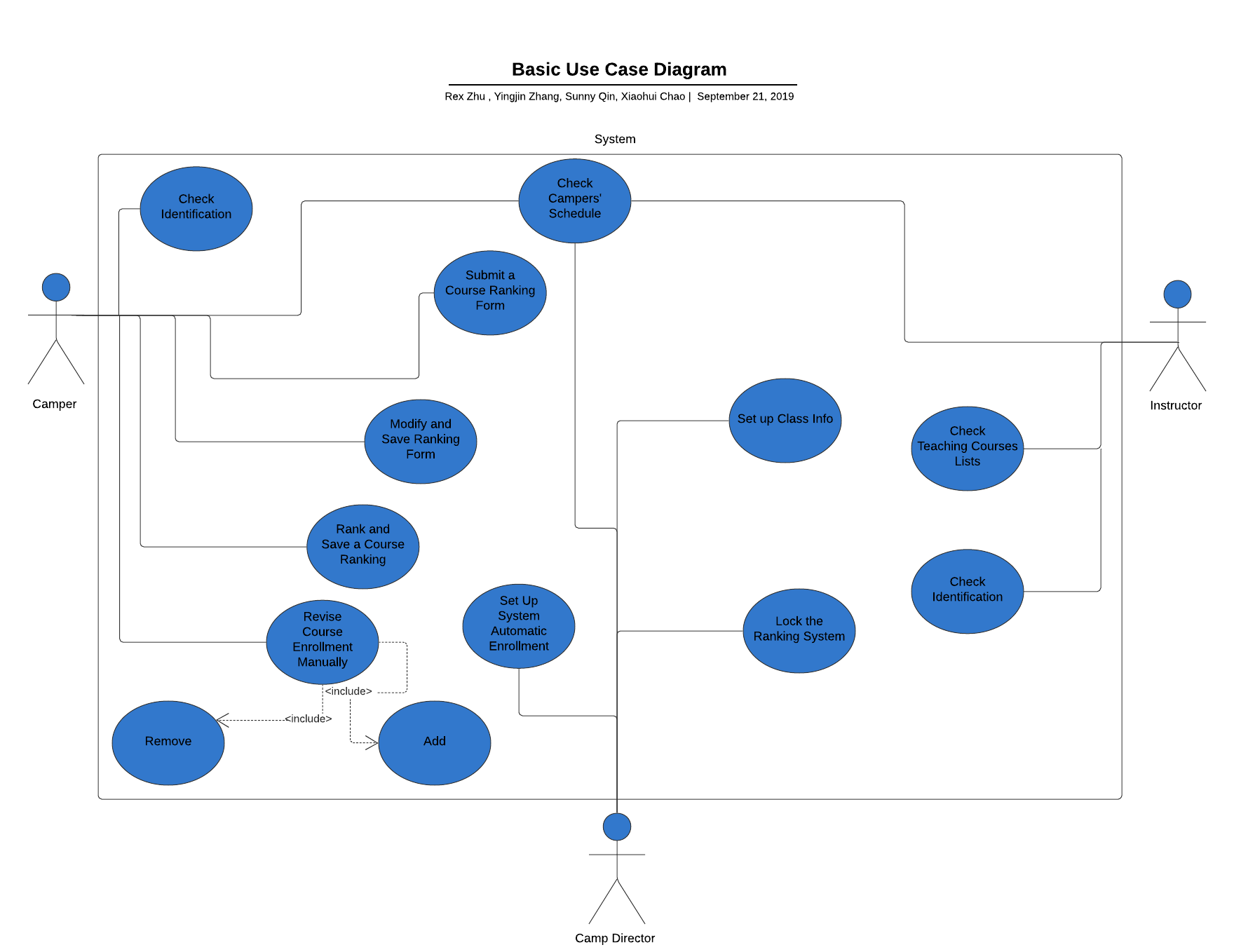


Figure 1.1

## User Classes and Characteristics

The system users can be divided into three classes, each having different sets of rights (authorization).

1. Camp director level

The camp director has the highest privilege. He is able to analyze classes for enrollment (low/high), add or remove class offerings, analyze student needs (empty blocks, duplicates, etc.) The camp director will run the software once to place the campers in the classes based on their rankings when the camp registration closes. After that, the camp director will need to add or delete a camper to the class manually.

1. Instructor level

Instructors are able to manually enroll or remove a camper from his offered class. Instructors are able to check an online report of a current class list and are able to print it. They are also able to check a camper’s schedule online.

1. Camper (i.e. attendee) level

Campers have the lowest privilege level. They are able to rank the preferred classes and check their course schedules after being assigned their classes by the administrators.

## Operating Environment

* Windows 7/8/10
* Mac OS X
* Linux/ Ubuntu/ CentOS

## Design and Implementation Constraints

***<Question section> More details are included in Appendix C***

*CRSS is developed in Java.*

Constraints for Design: It is hard to estimate how many educational institutes adopt CRSS. Therefore, the design for the computation center or what size of Amazon Web Service is hard to decide.

Constraints for Implementation: It is very expensive to get the middleware authorized.

## User Documentation

***<Question section> More details are included in Appendix C***

There will be three user manuals along with the system for different levels of users. The camper’s manual will include how to use the system to rank, modify and submit the ranking form, and also where they can find the class schedule after automatic enrollment. The instructor’s manual will include everything related to accessing his offered courses and the camper’s schedule. The administrators’ manual will include the basic operation they can do with the system such as locking the ranking system, running automatic enrollment and doing manual enrollment.

There will also be an online help to connect with the software development team so that no matter what kind of problems related to the system, they can provide help, especially on the system features’ problems.

## Assumptions and Dependencies

***<Question section> More details are included in Appendix C***

The development of the CRSS system requires several third-party software.

For the designing phase, Axure is adopted to design interfaces as well the Lucidchart is used for designing the logic of the CRSS.

For the implementation of the CRSS, several middleware and frameworks such as Kafka, thrift, Dubbo, and Zookeeper are possibly put into use. This system is designed with a relational database. The specific type of relational database is MySQL. An alternative plan for realizing the CRSS architecture can be ELK framework which is a famous agile development framework. E stands for Elasticsearch, which is a database and a potential index construction engine for editing these courses. L stands for Logstash, which is for creating logs for updating the CRSS. K stands for Kibana, which is for visualizing all the results.

# External Interface Requirements

## User Interfaces

***<Question section> More details are included in Appendix C***

Anticipation of user interfaces based on the description of this project:

*Based on the use case diagram (figure 1.1), we can have a general image of the logical interface. It is shown below, but the details related to the interface standards are not included due to lack of information. We still need to talk with our users to make sure of the specific requirements for this section.*

Main page: Every time the user connects to the webpage, it will show them the main page of the camp’s basic information. On this page, it may include a navigation menu containing every function in our system that users can select. Also, there should be a login button on the main page to allow users at different levels to get access to their movements.

Camper’s page: Every camper should have the option to see the class list and ranking form during the camp after they log in. The ranking form has two main columns corresponding to the ranking number and course name. Besides, the ranking form should have the option to let campers modify and submit. After automatic enrollment, the camper’s page should contain an option to let them see their class schedule and allow them to print it. The class schedule page will have two sessions corresponding to the morning course and evening course. The course showing in the schedule will show their name, instructor and course identifier.

Instructor’s page: Every instructor’s page should have the option to get access to their class roster and also an option to see the report of the current class list. Inside the class roster page, it will show every camper’s information and also an option to see a specific camper’s schedule.

Administrator’s page: Administrators’ page should contain all of the important options that they can perform, including “lock the ranking system”, “Automatic enrollment” and also “Manual enrollment”. In the manual enrollment page, it will show all the classes with their identifiers. There should be a drop-down menu to let them choose which course to modify manually. When they get into the class roster page, they can select “remove” or “add” to modify the campers in the class and select “submit the changes” at the end.

Error messages should be printed clearly for their reasons so that users can know the processing of the system. Also, the feedback of the system should be comprehensible and correct.

## Hardware Interfaces

***<Question section> More details are included in Appendix C***

Anticipation of hardware interfaces based on the description of this project:

The CRSS system runs on two types of hardware facilities - personal computers as clients and AWS as servers to maintain the CRSS.

Also, several high-elasticity clusters are required. One of the clusters is for deploying clusters of MySQL which is required for storing data and enhancing data security. One cluster involves middleware such as Kafka or rabbitMQ to manipulate message queues. Some servers each with the strongest Xeon CPU array and 128G memories install CentOS and Apache Dubbo framework to distribute computers in the whole system. Several high-performance computers with Zookeeper are selected to be supervisors for the whole system.

## Software Interfaces

***<Question section> More details are included in Appendix C***

Anticipation of the software interfaces based on the use case diagram (Figure 1.1):

HTTPS is involved in Spring Boot, MyBatis or ELK(ElasticSearch/ Logstash/ Kibana).

TCP is embedded in internet programming.

IPC is in CentOS communication cross-platform.

Also, other security and transmitting protocols are within Kafka, Dubbo, and Zookeeper.

## Communications Interfaces

***<Question section> More details are included in Appendix C***

*Anticipation of the communications interfaces based on the use case diagram (Figure 1.1):*

HTTPS is adopted to be the connection protocol when campers are logging, verifying and submitting sheets and receiving courses’ schedules as well as instructors try to log on, verify identification and obtain courses’ schedules by HTTPS protocol too. The transfer data will be encrypted by SSL under HTTPS protocol.

# System Features

## Sign up and log in

4.1.1 Description and Priority

The username and password will be generated by the system and given to the users. Rights to different activities will be restricted based on user type. A user without a username and password hardly does any activities in the system, so it is of high priority.

Priority component ratings:

benefit(9), penalty(9), cost(9), risk(9)

***<Question> More details are included in Appendix C***

4.1.2 Stimulus/Response Sequences

System: generate username and password for each user and restrict the rights for each user type.

4.1.3 Functional Requirements

|  |  |
| --- | --- |
| **Name** | Assign user a username and a password (Sign up) |
| **ID** | UC\_012 |
| **Description** | For first-time users, the username and password are assigned by the system. System will randomly assign them a username and password. |
| **Actors** | Instructors, campers, camp directors |
| **Organizational Benefits** | Based on the levels of the users, they can allow different levels of users to have different privileges. It will save time to operate on the system and protect the system’s security. |
| **Frequency of Use** | Once when they first get into the system |
| **Triggers** | User selects an option to create an account |
| **Preconditions** | Users are verified as to their levels.  Users are first-time logged on. |
| **Postconditions** | Users get an username and password and automatically log in. |
| **Main Course** | 1. System generates a username and password based on user’s level 2. System checks whether the username and password has already been generated before (see AC1) 3. System notifies user the username and password 4. System prompts users to confirm these. 5. User confirms to accept the username and password. (see AC2, EX1) 6. System logs in users |
| **Alternative Course** | AC1. The username and password have already been generated before.   1. Return to main course step 1   AC2. User wants another username and password.   1. Return to main course step 1 |
| **Exceptions** | EX1 Users do not want to get a username and password   1. Return to Main page |

|  |  |
| --- | --- |
| **Name** | Log in |
| **ID** | UC\_011 |
| **Description** | For different levels of users, they need to log in first to get access to the movements |
| **Actors** | Instructors, campers, camp directors |
| **Organizational Benefits** | It can ensure that no one will misuse their authority to do other’s work. The limitation of their authority will limit their choices and will save their time for doing improper things. |
| **Frequency of Use** | Every time the users want to use the system to achieve their goals |
| **Triggers** | User selects an option to log in |
| **Preconditions** | Users want to achieve some goals that may need the authority of its level. |
| **Postconditions** | Users log in and can achieve their following goals |
| **Main Course** | 1. System prompts user to enter their username and password 2. Users enter their username and password (see AC1) 3. User confirms to log in (see EX1) 4. System logs user in and notifies user successfully log in |
| **Alternative Course** | AC1 User does not have a username and password   1. Remind “Assign username and password” Use case. 2. Return to main course step 2. |
| **Exceptions** | EX1 Users do not want to log in right now.  1. Return to Main page |

## Class info set up

4.2.1 Description and Priority

The camp director can set up the course information after discussing it with instructors. The system shall generate a form with the black of course name, section number, course cap, course time, instructor name and classroom address. And the director can fill out the form and save it. The form can be modified several times and needs to be loaded correctly next time.

This feature is of a low priority since if this feature does not work, there are other ways to let campers know about the class information such as making an announcement.

benefit(5), penalty(3), cost(3), risk(0)

***<Question> More details are included in Appendix C***

4.2.2 Stimulus/Response Sequences

Camp director: choose an option to set up the course

System: generate a new blank form if there is no saved form; or load previous saved form to let the director modify it.

4.2.3 Functional Requirements

|  |  |
| --- | --- |
| **Name** | Set up the class info (enrollment caps and time) |
| **ID** | UC\_001 |
| **Description** | The director sets up the class info (time blocks and enrollment caps) after consulting the instructor. |
| **Actors** | Camp director |
| **Organizational Benefits** | The time blocks and enrollment caps are set up to avoid class conflict and maintain good class size and student-teacher ratio |
| **Frequency of Use** | Multiple times, instructor and camp directors may change the class info before the enrollment process. |
| **Triggers** | The camp director chooses an option to set up the class info. |
| **Preconditions** | After discussion, the camp director and instructors confirm the time block of each class and the class cap. |
| **Postconditions** | The class time and enrollment cap are visible to all users |
| **Main Course** | 1. System determines the user is not logged in as directors level and redirects user to log on (see AC1) 2. User logs on as the camp director level (see AC2,EX1). 3. System prompts a form for users to put in the class name, class identifier, time block, location, instructor name, any special instructions for the class, and enrollment cap (see EX2). 4. The user fills in the information. 5. User chooses an option to submit the form. 6. System prompts user to confirm 7. User confirms the changes in the schedule (see AC3, EX3) 8. System stores the saved class info and redirects the camp to view class information. |
| **Alternative Course** | AC1 System determines users is already logged on   1. Return to Main Course step2.   AC2 User logs off again.   1. Return user to main course step 1.   AC3 System determines an instructor teaches more than one class in a single block.   1. System notifies camp director that an instructor cannot teach more than one class in a single block 2. Return user to Main Course step 3. |
| **Exceptions** | EX1 The user is not identified as the camp director level.   1. Notify users that only the camp director has the authority to do so. 2. Return user to Main Course step 1.   EX2. User does not want to modify the class info.   1. Return user to Main page   EX3. User does not confirm the submission   1. Return user to Main Course step 3. |

## Ranking course

4.3.1 Description and Priority

System shall generate a form with a course name column and a rank number column for a camper. And if the form for that camper existed, then load the information in that form. After being edited by a camper, the form shall be saved by the system.

This is a core system feature in our system because campers should select classes they want and this ensures that the automatic enrollment can be processed. Thus this is of high priority.

benefit(9), penalty(9), cost(9), risk(2)

***<Question> More details are included in Appendix C***

4.3.2 Stimulus/Response Sequences

Camper : Choose an option to start ranking or modify a previous ranking, then save the

ranking from.

System: Generate a ranking form that can be edited by campers.

4.3.3 Functional Requirements

|  |  |
| --- | --- |
| **Name** | Rank and save a course ranking |
| **ID** | UC\_003 |
| **Description** | Each camper can rank the courses. Based on that result, they can enroll in the classes they like. |
| **Actors** | Camper |
| **Organizational Benefits** | It can help campers get enrolled in their favored courses. |
| **Frequency of Use** | Each camper may rank multiple classes, so they may use multiple times. |
| **Triggers** | User selects an option to rank the class in the form |
| **Preconditions** | The course list the camp offered is confirmed.  User is going through the course list and finds class he/she wants to take |
| **Postconditions** | The course ranking form is updated with the course name and the corresponding ranking number. |
| **Main Course** | 1. System determines user is not logged in and redirects user to log on (see  AC2)  2. Users log on (see AC3, AC4)  3. System prompts user to rank for the course  4. User fills in the number and confirms to rank it right now. (see AC1, AC2, EX1, AC6)  5. System saves the course name in the form  6. System redirects the user to their saved ranking form to view the form. |
| **Alternate courses** | AC1. The number has already been filled in before.   1. System notifies users that the overlapping of the number happened in the ranking form. 2. See “Modify and save the form” use case   AC2: System determines users is already logged on   1. Return to Main Course step3   AC3: User logs off again   1. Return user to main course step 1   AC4: User does not have an account already.   1. User get assigned user name and password 2. Return user to main course step 1.   AC5. The course name has already been filled in the form before   1. System notifies the camper that the overlapping of the course happened in the form. 2. See “Modify and save the form” Use case |
| **Exceptions** | EX1 Camper decides not to rank it.   1. Return to the main page. |

|  |  |
| --- | --- |
| **Name** | Modify and save the ranking form |
| **ID** | UC\_004 |
| **Description** | Campers can modify the form before they decide to submit it. They are allowed to replace the courses or change the number of courses. |
| **Actors** | Camper |
| **Organizational Benefits** | This may give the users an opportunity to change their form so that their submitted form is what they most like. |
| **Frequency of Use** | Unknown |
| **Triggers** | Camper selects the option to modify their saved ranking form |
| **Preconditions** | User has a ranking form and it has already been saved before. |
| **Postconditions** | The form is updated. No same number or the same course would appear in the form. The new form should be viewable. |
| **Main Course** | 1. System determines users is not logged in and redirects user to log on (see AC1) 2. Users log on (see AC2) 3. System redirects user to the ranking form 4. User edit the form 5. User chooses an option to save the form 6. System prompts user to confirm the saving movement 7. User chooses confirm saving(see EX1, AC3, AC4) 8. System saves the form (see EX2) 9. System redirects users to view the updated form. |
| **Alternative Course** | AC1: System determines user is already logged on   1. Return to Main Course step5   AC2: User logs off again   1. Return user to main course step 1   AC3. The number has already been filled in before.   1. System notifies users that the overlapping of the number happened in the ranking form. 2. Return to Main course step 3   AC4. The course name has already been filled in the form before   1. System notifies the camper that the overlapping of the course happened in the form. 2. Return to Main course step3 |
| **Exceptions Exceptions** | EX1: User does not want to save it right now   1. Return user to main course step 3   EX2: System fails to save the form   1. System notifies users that an error has occurred when saving the form 2. Return user to main course step 1 |

|  |  |
| --- | --- |
| **Name** | Submit the ranking form |
| **ID** | UC\_005 |
| **Description** | Campers can submit the ranking form after they complete the ranking and decide to submit it. |
| **Actors** | Campers |
| **Organizational Benefits** | The automatic enrollment system can enroll the campers in courses based on their course ranking |
| **Frequency of Use** | Once before deadline |
| **Triggers** | A user selects an option to submit the form |
| **Preconditions** | All courses should be in the course system.  Students add legitimate courses into the form. |
| **Postconditions** | The ranking is visible to the campers after they submit the ranking. |
| **Main Course** | 1. System determines users is not logged in and redirects user to log on (AC1) 2. User logs on(see AC2) 3. System prompts user to confirm submitting the form 4. Users confirm to submit now (see EX1) 5. System saved the form (see EX2, EX3, EX4) 6. System redirects user to review their form |
| **Alternative Course** | AC1:System determines the user has already logged on as camper   1. Return to Main Course step 3   AC2: User logs off again   1. Return user to main course step 1 |
| **Exceptions** | EX1: User does not want to submit it right now.   1. Return to Main Course step1   EX2: User does not have five more rankings.   1. System notifies the user that his rankings are not enough 2. See “ Rank and save courses” Use case   EX3: User has two same ranking or two same course name in the form   1. System notifies the user that their rankings contain the same number or same course 2. See “ Modify and save the form” Use case   EX4: The numbers in the form are not consecutive.   1. System notifies the user that their rankings are not well sorted. 2. See “ Modify and save the form” Use case |

## Lock ranking system

4.4.1 Description and Priority

After the deadline of ranking courses, the camp director can lock the ranking system so that no one can modify it anymore. This feature also makes sure that the automatic enrollment can be processed successfully. So it is probably of high priority.

benefit(6), penalty(6), cost(5), risk(3)

***<Question> More details are included in Appendix C***

4.4.2 Stimulus/Response Sequences

Camp director: choose an option to lock the system

System: make the ranking forms non-changeable and save all information on ranking forms.

***<Question section> More details are included in Appendix C***

4.4.3 Functional Requirements

|  |  |
| --- | --- |
| **Name** | Lock ranking system |
| **ID** | UC\_006 |
| **Description** | After the deadline of the ranking course, the director shall lock the ranking system so that no camper can submit ranking anymore. |
| **Actors** | Camp director (Administrator) |
| **Organizational Benefits** | This process can make the enrollment process more efficient. After locking the system and automatic enrolling, there are only small changes made to the class. |
| **Frequency of Use** | Once after ranking deadline |
| **Triggers** | Director selects the option to lock the ranking system |
| **Preconditions** | After the deadline of ranking courses. |
| **Postconditions** | No change can be made by camper and can only be made by the camp director. Campers will know this after the system is locked. |
| **Main Course** | 1. System prompts user to confirm locking the system 2. Director confirms to lock (see EX1) 3. System determines user is not logged in and redirects director to log on (see AC1) 4. Director logs on (see AC2, AC3) 5. System locks all the ranking forms. (see EX2) 6. System notifies the director successfully. (EX3) |
| **Alternative Course** | AC1: System determines director is already logged on   1. Return to Main Course step5   AC2: Director logs off again   1. Return user to main course step 3   AC3: Director does not have an account already.   1. System assigns the user a username and password 2. Return user to main course step 4. |
| **Exceptions** | EX1: Director decides not to lock right now   1. Return user to Main page   EX2: System fails to lock the ranking   1. System notifies the director that the locking process is failed 2. Return director to Main Course step1   EX3: System fails to lock   1. System notifies the director that there is an error occurred when notifying the camper 2. Return director to main course step 6 |

## Automatically enroll class

4.5.1 Description and Priority

The system will be run by an administrator to place the campers in the classes based on their rankings one time when the camp registration closes. This feature ensures that the campers can automatically enroll campers in the classes based on their preferences. If this functional requirement does not work well, then campers cannot enroll in the camps and the administrator should manually enroll all campers in, which costs a lot and is not very accessible, then the camps probably cannot be held on time. It will be an urgent problem that blocks the use of the system until the issue is resolved. Thus, it is of high priority.

Priority component ratings:

benefit(9), penalty(9), cost(9), risk(9)

***<Question> More details are included in Appendix C***

4.5.2 Stimulus/Response Sequences

Camp director: choose to run the software

System: Enroll campers into classed based on their rankings

4.5.3 Functional Requirements

|  |  |
| --- | --- |
| **Name** | Run automatically enrollment system |
| **ID** | UC\_007 |
| **Description** | Administrators will run the software to place campers in the classes based on their rankings. |
| **Actors** | Camp Director |
| **Organizational Benefits** | Control the deadline of submitting ranking system |
| **Frequency of Use** | Once after the ranking system gets locked |
| **Triggers** | A camp director clicks into generating courses’ schedules. |
| **Preconditions** | The ranking system is locked by an administrator. |
| **Postconditions** | All submitted ranking forms are handled.  A schedule of each camper is generated. |
| **Main Course** | 1. System determines user logging status and redirects director to log on (see AC1) 2. User logs on (see AC2, AC3) 3. System prompts user to confirm running automatic enrollment 4. User confirms to run it(see EX1) 5. System looks up the earliest ranking form in the system submitted by campers 6. System tries to enroll the camper in the class from the top ranking in the ranking form. (see EX2,EX3) 7. System saves the enrollment result (see EX4) 8. System redo steps 5-7 until there is no form saved in the system. 9. System notifies the user that automatic enrollment is completed. |
| **Alternative Course** | AC1: System determines director is already logged on   1. Return to Main Course step 3   AC2: Director logs off again   1. Return user to main course step 1   AC3: Director does not have an account.   1. System assigns the user a username and password 2. Return user to main course step 3. |
| **Exception** | EX1 User does not confirm to run   1. Return user to Main Course step 1   EX2 The current preferred class is full.   1. Return user to Main Course step 6   EX3. There is a time conflict that occurred in the enrolled course and preferred course.   1. Return user to Main Course step 6   EX4. Saving the enrollment result failed.  1. Return to Main Course step 5 |

## Generate the camper’s schedule

4.6.1 Description and Priority

Camper schedules will show the time block (i.e. morning or afternoon), the class enrolled in that block, the location of the class, any special instructions for the class, and the instructor(s) names. These schedules will be printable. It is useful for campers to view their classes and know the class information. If this feature is not satisfied, it should be fixed if time permits but can be postponed since there are other ways to let campers know their schedule.

Priority component ratings:

benefit(6), penalty(7), cost(7), risk(3)

***<Question> More details are included in Appendix C***

4.6.2 Stimulus/Response Sequences

Camper/Instructor: choose an option to view the class schedule.

System: System generates the camper’s schedule and displays it with a button to print it.

4.6.3 Functional Requirements

|  |  |
| --- | --- |
| **Name** | Display camper’s class schedule |
| **ID** | UC\_010 |
| **Description** | Campers and instructors check the camper’s class schedule after automatic enrolling. Campers can print the schedule. |
| **Actors** | Instructors, campers |
| **Organizational Benefits** | Campers can view their class schedule so that they can know the classes they enrolled in and know the information about those classes(time, section, instructor, location).  Instructor can view the camper's schedule. |
| **Frequency of Use** | Quite often. Campers may want to see the schedule multiple times and instructors may also check their students’ schedule. |
| **Triggers** | Users select the option to display a camper’s class schedule |
| **Preconditions** | The auto-enrollment system has already been run. |
| **Postconditions** | The camper’s schedule is displayed and can be printed. |
| **Main Course** | 1. System determines the user is not logged in and redirects the user to log on(see AC1).  2. User logs on(see AC2)  3. System generates the camper’s schedule.  4. System displays the class schedule  5. Users choose an option to print the schedule (see AC3)  6. Users confirm to print(see EX2)  7. System connects to printer(see EX1)  8. Print the class’s schedule |
| **Alternative Course** | AC1 System determines the user is already logged on as administrators level.  1. Return to Main Course step 3.  AC2 User logs off again.  1. Return user to Main Course step 1.  AC3 Users choose not to print the schedule.  1. Return to Main course step 3 |
| **Exceptions** | EX1 Connection fails  1. System notifies user to check the printer setting  2. Return user to Main Course step 4  EX2 User does not confirm to print  1. Return user to Main Course step 3 |

## Generate a report of a current class list

4.7.1 Description and Priority

Instructors need to be able to see an online report of a current class list and should be able to print it. Since the information of the class is set up by the camp director, so if the system cannot generate it, the core functional feature is not influenced much, and the director may be able to mutually generate a current class list. So it is of low priority.

Priority component ratings:

benefit(4), penalty(2), cost(6), risk(2)

***<Question> More details are included in Appendix C***

4.7.2 Stimulus/Response Sequences

Instructor: choose to view the current class list and print it if necessary.

System: generate the current class list and display it.

4.7.3 Functional Requirements

|  |  |
| --- | --- |
| **Name** | Display a current class list |
| **ID** | UC\_002 |
| **Description** | Instructors can see an online report of a current class list |
| **Actors** | Instructors |
| **Organizational Benefits** | Instructors can have a clear view of their teaching duties.  Students can be offered well-organized course schedules via the system based on the display of teaching resources. |
| **Frequency of Use** | Very often |
| **Triggers** | Instructor chooses an option to view the course list |
| **Preconditions** | The class information has been added to the system by a camp director. |
| **Postconditions** | A list of all course information at this time showed on the screen and can be printed. |
| **Main Course** | 1. System determines the user is not logged in as directors level and redirects user to log on(see AC1)  2. User logs on(see AC2)  3. System generates a list of current classes and displays it.  4. User chooses an option to print the course (see AC3)  5. Users confirm to print  6. System connects to printer(EX1)  7. Print the course list out |
| **Alternative Course** | AC1 System determines the user is already logged on as directors level.  1. Return to Main Course step 3.  AC2 User logs off again.  1. Return user to Main Course step 1.  AC3 User does not choose to print  1. Return user to Main Course step 3 |
| **Exceptions** | EX1 Connect fails  1. Notify user to check the printer setting  2. Return user to Main Course step 4 |

## Manually enroll and drop

4.8.1 Description and Priority

Campers probably add courses or drop courses after the automatic enrollment system has been run, then at this time, the administrator should manually add or remove their information to the course roster. This feature is quite important since late attendees need to enroll in classes and students would like to change their classes. So it is of high priority.

benefit(9), penalty(6), cost(7), risk(3)

***<Question> More details are included in Appendix C***

4.8.2 Stimulus/Response Sequences

Administrator: choose an option to make a change in course enrollment

System: load the course roster for the administrator.

4.8.3 Functional Requirements

|  |  |
| --- | --- |
| **Name** | Add a camper to a class manually |
| **ID** | UC\_008 |
| **Description** | Campers may want to add a class after the enrollment system has been run, and late campers need to enroll in classes. |
| **Actors** | Administrators(instructors/ camp directors) |
| **Organizational Benefits** | Perhaps Increasing profit by charging tuition for classes added.  Offer a way for campers to take more classes. |
| **Frequency of Use** | Quite often. Many campers may not be satisfied with the classes schedule generated by the auto-enrollment system and choose to drop camps and add others. Some attendants decide to join the camp after the deadline, so administrators need to add camps for them. |
| **Triggers** | Camper wants to add a class to his/her schedule.  Administrator selects the option to add a camper into a class. |
| **Preconditions** | The auto-enrollment system has already been run.  A camper asks for adding specific camp spontaneously. |
| **Postconditions** | The class is added to the camper’s schedule.  The camper enrolls in the class.  The class is on his/her schedule when the camper views the schedule.  The information of the camper is added to the class roster. |
| **Main Course** | 1. System determines the user is not logged in as administrators level and redirects the user to log on(see AC1).  2. User logs on(see AC2)  3. System asks the user to select a class’ roster to access.  4. User selects a class’ roster.  5. System prompts users to confirm the choice.  6. User confirms the choice(see AC4, EX3).  7. System redirects user to the class roster page  8. User enters camper’s information to the class roster.  9. System prompts users to confirm adding movement.  10. User confirms the information(see EX1,EX2, AC5).  11. System adds the camper’s information to the class roster(see EX4).  12. System updates the camper’s schedule  13. System redirects the user to the class roster to view the roster. |
| **Alternative Course** | AC1 System determines the user is already logged on as administrator level.  1. Return to Main Course step 3.  AC2 User logs off again.  1. Return user to Main Course step 1.  AC3 User decides to remove the camper from conflict class.  1. See “Remove camper from class” User Case.  AC4 User decides to cancel the choice  1. Return user to Main Course step 3.  AC5 User decides to cancel adding movement.  1. Return user to Main Course step 3. |
| **Exceptions** | EX1 System detects that the camper has already been in the class roster.  1.System notifies users that a camper cannot enroll in a camp twice.  2.Return user to Main Course step3.  EX2 System detects that another enrolled camp is in that same block with the class the camper wants to add.  1.System notifies users that a camper cannot be enrolled in more than one camp in a given block of time.  2.System asks the user whether he/she wants to remove the camper from the conflict camp first and then back to add one.  3. User decides to not do so. (see AC3)  4. Return user to Main Course step 3.  EX3 System detects that the class is full.  1. System notifies users that there is no space in the class.  2. Return user to Main Course step 3.  EX4 System fails on adding information to the class roster.  1.System notifies users that an error has occurred.  2.Return user to Main Course step3. |

|  |  |
| --- | --- |
| **Name** | Remove camper from a class manually |
| **ID** | UC\_009 |
| **Description** | Campers may want to drop a class so that they can add a new one. |
| **Actors** | Administrators(instructors/ camp directors) |
| **Organizational Benefits** | Afford a way to allow campers to change their mind. |
| **Frequency of Use** | Quite often. Many campers may not be satisfied with the classes schedule generated by the auto-enrollment system and choose to drop camps and add others. |
| **Triggers** | Camper wants to drop a class from his/her schedule.  Administrators select the option to remove a camper from a class |
| **Preconditions** | The auto-enrollment system has already been run.  A camper asks for dropping a specific camp spontaneously.  The camper already has a class schedule. |
| **Postconditions** | The class is removed from the camper’s schedule.  The information of the camper is removed from the class roster. |
| **Main Course** | 1. System determines the user is not logged in as administrators level and redirects the user to log on(see AC1).  2. User logs on(see AC2)  3. System prompts users to select a class’ roster to access further.  4. User selects a class.  5. System prompts users to confirm the choice.  6. User confirms the choice(see AC3).  7. System redirects user to the class roster page  8. User removes the camper's information from the class roster(EX1).  9. System prompts users to confirm delete action.  10. User confirms the removal(see AC4).  11. System removes the camper’s information to the class roster(see EX2).  12. System updates the camper’s schedule  13. System redirects the user to the class roster to view the roster. |
| **Alternative Course** | AC1 System determines the user is already logged on as administrators level.  1. Return to Main Course step 3.  AC2 User logs off again.  1. Return user to Main Course step 1.  AC3 User decides to cancel the choice  1. Return user to Main Course step 3.  AC4 User decides to cancel the movement.  1. Return user to Main Course step3. |
| **Exceptions** | EX1 User reports there is no information about the camper in the class’ roster.   1. Return user to Main Course step 3.   EX2 System fails on removing information to the class roster.   1. System notifies users that an error has occurred. 2. Return user to Main Course step3. |

# Other Nonfunctional Requirements

## Performance Requirements

***<Question section> More details are included in Appendix C***

The performance requirement and server that CRSS will use has not been decided by now. The physical machine to use will be determined by the camp. The speed of the camper’s and instructor’s connection will mostly depend on the hardware used rather than the characteristics of this system. However, this system is still expected to run as fast as possible in camper’s side because the enrollment process is based on the order of the submission of the ranking form. A small latency will influence the enrollment results much. Hence, the system requirements for camper’s connection including ranking, modifying and submitting process are more demanding.

## Safety Requirements

To ensure no CRSS’s user will lose any data, our system provides a mechanism to help users report their feedback(bugs or crack, etc.). And update the system regularly based on that feedback, which is very important because losing data for administrators can be a disaster, and so do campers and instructors.

There is no physical damage to the environment or any harm to users when using our system.

## Security Requirements

The server on which the campers may operate on has its security to prevent unauthorized *rank/modify/submit* access. And campers have no access to *enroll/change* the schedule after automatic enrollment.

The server on which the instructors reside has its security. They will have *read* access to the camper’s schedule and *enroll/drop* access after the automatic enrollment. Also, they have the *read/print* access to the current course list.

The PC on which the administrator resides will have its security. Only the camp director will have access to the machine and the program to *set up the class session*, *lock* the ranking system and *run* the automatic enrollment. There is no restriction to *enroll/drop* access during the manual enrollment for the camp director.

All the users will be assigned a username and password to protect their privacy and security. All the users of this system will also have *read* access to the class section information.

## Software Quality Attributes

Our system should be easy to learn for different levels of users. This system does not require users to have any knowledge before using it. A new user can reach proficiency for core functions after following the instruction given to them on their first day of camp. To achieve this goal, the instruction should comprehensible and the user interface should be clear and easy to follow. Specifically, the navigation should be simple. The error message should be comprehensible for a specific reason.

## Business Rules

Only A camp director can lock the ranking system and run the automatic enrollment system, as well as manually enrollment process.

A camper is authorized to edit their ranking form and submit it.

An instructor has access to the class information and can manually adjust some campers after automatic enrollment.

# 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.>***

* CRSS: Class Rank and Sort System, which is used by campers, instructors, and administrators during the camp.
* Course Identifier: Each course will have its identifier including its name and offered section.
* Class roster: the class page which contains class info and student info.
* Automatic enrollment: This is the process done when the ranking system for campers has been locked by administrators. After running this procedure, most of the campers will get enrolled in their favored classes. This process can be done only once.
* Manual enrollment: This is the process done after the automatic enrollment finished to modify the camper’s schedule manually. Only the instructor or the camp director has access to it. This process can be done multiple times.
* Cap: the capacity of a class in the camp.

# Appendix B: Analysis Models

*None*

# Appendix C: To Be Determined List

*2.5 Design and Implementation Constraints:*

*The description of this system does not describe the constraints in detail. We need to talk with the users to determine the specific, measurable requirement for this section.*

*2.6 User Documentation*

*We may need the specific requirements for documentation, such as “what kind of standard it can use?”, “Is there any documentation they may need along with the system?”*

*2.7* Assumptions and Dependencies

*More details about the language using and operating system may need to discuss with users. After deciding the databases, operating system and programming languages, we can fill in this section.*

*3.1. User Interfaces*

*The description does not include any information about the user interface design. Do we need to ask users about what kind of design standard we may want to use? Also, what kind of functions they may want to show on the main page?*

3.2 *Hardware Interface*

The hardware interface is hard to be determined now since we do not have a decision on what hardware should be used in this system. And we are not familiar with the hardware requirement, so we need further knowledge about this part.

*3.3 Software Interfaces*

The interfaces are hard to be determined right now. We are not digging into the details of implementation. So just anticipation was written in this section, which may provide a possible implementation method. Also, some questions may be given to users, like “Do they have any preferred system?”, “Do they get used to the database we proposed?” “What kind of software interfaces may need to be implemented first?”

3.4 *Communication Interfaces*

*We need to interview with the users about “what kind of security issues may bother them a lot while using communication tools?” Based on the responses, we may be able to design and develop our communication interfaces, because we do not want our design to raise security issues.*

*4.4 Generate the Course List*

*The course list can be viewed by instructors, but the content of the list is not clear. Is it the currently offered classes by the specific instructor or all classes offered by all instructors?*

*4.X.2. Priority*

*The priority for the different systems is difficult to tell. We may need to ask users that “which functions they think is the most important one?” Based on that ranking, we can set up the priority in section4.*

*5.1 Performance Requirement*

*There is no specific performance requirement in the description of this project. Need to talk with the users and make a decision on it in the future. For example, ask them how long the system should be run within? For different functions, do they have any run-time or performance requirement for that one?*