Software Requirements Specification

For

Automated Intelligent Advisement Interface

Version 1.0

10/12/2016

Michael Harootoonyan

Megan Meyers

Sina Eradat

Jonathan Carrasco

Izzy Hasson

Brandon Garcia

Andrea Lee

Prepared for

COMP 380 Section 15900—Introduction to Software Engineering

Instructor: R. Lingard

Fall 2016

**Table of Contents**

[**1. Introduction 1**](#_gjdgxs)

[1.1 Purpose of the Document 1](#_30j0zll)

[1.2 Purpose of the Product 1](#_3znysh7)

[1.3 Scope 1](#_2et92p0)

[1.4 Definitions, Acronyms, and Abbreviations 1](#_tyjcwt)

[1.5 References 2](#_3dy6vkm)

[1.6 Overview 2](#_1t3h5sf)

[**2. General Description 2**](#_4d34og8)

[2.1 Product Perspective 2](#_2s8eyo1)

[2.2 Product Functions 3](#_17dp8vu)

[2.3 User Characteristics 4](#_3rdcrjn)

[2.4 General Constraints 4](#_26in1rg)

[2.5 Assumptions and Dependencies 4](#_lnxbz9)

[**3. Specific Requirements 5**](#_35nkun2)

[3.1 Functional Requirements 5](#_1ksv4uv)

[*3.1.1 Provide Difficulty Rating 6*](#_2jxsxqh)

[*3.1.2 Provide Comprehensive Requirements Chain 7*](#_z337ya)

[*3.1.3 Create Short-Term Schedule 8*](#_3j2qqm3)

[*3.1.4 Modify Short-Term Schedule 10*](#_1y810tw)

[*3.1.5 Create Long-Term Planner 11*](#_4i7ojhp)

[*3.1.6 Issue Impacted-Graduation Warnings for Planner 12*](#_2xcytpi)

[*3.1.7 Optimize Changes in Major 13*](#_1ci93xb)

[3.3 Non-Functional Requirements 14](#_2bn6wsx)

[*3.3.1 Performance 14*](#_qsh70q)

[*3.3.2 Reliability 14*](#_3as4poj)

[*3.3.3 Availability 14*](#_1pxezwc)

[*3.3.4 Security 14*](#_49x2ik5)

[*3.3.5 Maintainability 14*](#_2p2csry)

[*3.3.6 Portability 14*](#_147n2zr)

[**4. Deliverables 14**](#_3o7alnk)

[**5. Open Issues 14**](#_23ckvvd)

# 1. Introduction

This document specifies the software requirements for the Automated Intelligent Advisement Interface, or AI2. The subsequent sections of the software requirements specification will explain the purpose of the document and the purpose of the product. In addition, the sections will address the scope; definitions, acronyms, and abbreviations; references; and overview.

## 1.1 Purpose of the Document

The purpose of this document is to provide any software engineer with the information necessary to design and implement the required software product, AI2. This document is intended for stakeholders as well, so that they may obtain a better understanding of the software that will be provided to them and how it may be used. This document will allow all parties to come to agreement on what is expected to be delivered. Finally, this document will be presented to COMP 380 for its approval.

## 1.2 Purpose of the Product

AI2 is a dynamic, responsive, and comprehensive solution for California State University, Northridge (CSUN) students that will assist against “bad advisement.” CSUN currently possesses myNorthridge Portal, which serves as a centralized hub for users to access campus resources and services. Among them is the SOLAR system, which deals with student administration, human resources, and financials. In addition, it offers registration and other degree-related planning tools. However, as AI2’s name implies, the existing services are riddled with errors and inefficiencies. AI2 is a program that will complement CSUN’s existing SOLAR system; by working in tangent, AI2 will either improve or expand upon its capabilities to provide students with accurate and up-to-date information.

## 1.3 Scope

This project will produce software known as AI2, which will be accessible 24/7 via the web. It will help students graduate on time by offering long-term planning guides and short-term confirmation to ensure that the student is able to proceed with their next step as planned. In addition, the software will give students the tools needed to make better informed decisions, including providing balanced class suggestions based on a difficulty rating that is more precise than unit counts, and automatically checking for an inefficient ordering of classes.

The software is intended to aid students, not dictate them; the only time the software will stop a student is if it is impossible for a student to proceed. While the software will warn against decisions that will impact a student’s ability to graduate on time, the student will be the final arbiter.

## 1.4 Definitions, Acronyms, and Abbreviations

Database: Class Catalog. A database that contains information for all classes offered for a particular semester, including course, dates, times, and professor, amongst others.

Database: Course. A database that contains all courses offered by the college, including a course’s prerequisites.

Database: General Education Requirement. A database that contains general education requirements, including what courses can fulfill a particular requirement.

Database: Student Grade. A database that contains grades given to students in each class.

Database: Major. A database that contains all majors the college offers, as well as their required courses.

Database: Miscellaneous Requirement. A database that contains miscellaneous requirements, such as the Upper Division Writing Proficiency Exam.

Database: Student. A database that contains student information, such as name, date enrolled, major, and classes taken.

Database: Difficulty Survey. A database that contains raw data from surveys given to students near the end of the semester, where they provide an estimate of the number of hours per week spent studying for a particular class, among others (see: Assumptions and Dependencies).

myNorthridge Portal. A centralized hub that gives users access to campus resources and services. This is used to access SOLAR.

SOLAR. CSUN’s current system which houses information and services for student administration, human resources, and financials.

Student. A user who attends CSUN, presumably for the purpose of obtaining a degree.

Stakeholder. Anyone interested in the outcome of the software, including students, staff, and software developers.

## 1.5 References

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, dated 25 June 1998, by Software Engineering Standards Committee of the IEEE Computer Society.

## 1.6 Overview

The rest of the software requirements specification contains four sections. Section 2 provides a general description, which goes over the product perspective, product functions, user characteristics, constraints, and assumptions. Section 3 contains specific requirements, including functional and non-functional requirements. Section 4 reviews deliverables, and section 5 encompasses any open issues.

# 2. General Description

This section offers insight on what is needed by the product and what is expected of its users, as well as the constraints placed on the product.

## 2.1 Product Perspective

Users will be able to access the product via CSUN’s myNorthridge Portal on the web by using the school-provided login. The program will store the user’s submitted details so that users may leave and re-access their information at any time.

Like SOLAR, this product will have access to CSUN’s databases, including student information, course prerequisites, and class availability.

## 2.2 Product Functions

The product must provide a number of functions, including the ability to suggest difficulty ratings, provide a comprehensive requirements chain, create a short-term schedule, modify the schedule, create a long-term planner, issue warnings for planner choices that impact graduation date, and optimize changes in major.

The difficulty rating function provides a difficulty rating that is tailored to each course and provides a closer estimate than unit count alone. It may be used manually by students while they create their schedule; however, it will mainly be used automatically by the program when it determines which class suggestions to make. Rating is determined by survey data gathered from students and professors specifying the estimated hours per week spend on the coursework outside of class, as well as expected grades. The actual grades are used to estimate difficulty as well. Higher ratings denote higher time commitment to obtain a passing score.

The comprehensive requirements chain function will provide a list of prerequisites, co-requisites, and any supplementary post-requisites in a chain. It may be used manually by students while they create their own schedule; however, it will mainly be used by the program itself so that it may make more informed decisions when determining how soon a certain class must be taken to fulfill requirements within a given timeframe. Take, for example, a simplified chain that goes from COMP 110 to 182 to 282 to 380 to 490 to 491. If a student transferred and already completed 110 and 182, those links will be cut from the chain. If a student has a major that only requires COMP classes up to 380, the 490 and 491 links will be severed. The final chain for the student would be 282 to 380. (Note that this is a simplified example. The program would take into consideration all prerequisites, co-requisites, and post-requisites, such as the additional PHIL 230 prerequisite for COMP 380, the many lab co-requisites, and all classes that require COMP 110, 122, etc. as prerequisites. The term “chaining” is used to give a concise visual.)

The short-term schedule will allow students to create an error-free plan for the coming semester(s) based on when class schedule data is available. Unlike the standard enrollment process, this schedule creator will provide real-time feedback to the student and make optimized suggestions that will aid the student in graduating on time. The schedule creator will access the student’s completed classes; the classes currently offered in the semester of choice; the major, general education, and other requirements pending; the chain function for pre- or co-requisites; and the load balance function to build an optimal schedule for the student within the parameters set forth in the search filters. The filters include student’s status (e.g. full time), personal scheduling requirements (e.g. no class after 4pm), and class status (open, closed, waitlist), among others. Based on this information, the system will give a priority rating to every required course. Take, for example, the earlier COMP 110 to COMP 491 chain, as well as a theoretical MATH 103 to ACCT 220 to IS 312 chain. Because the COMP chain is longer (six classes, as opposed to three), the schedule creator will give a higher priority to COMP 110 to a freshman student than to MATH 103. Priority is null for classes whose prerequisites have not been met. Classes with similar priorities may be reordered based on their difficulty rating so as to ensure no semester is give too difficult or too easy a load. A schedule is considered optimal when the difficulty rating falls within the predetermined acceptable range, the number of units is commensurate with the pace required (e.g. 15 units for full time), no extraneous or erroneous classes are selected, and the requirements chain is maximally shortened.

The modifying schedule function allows students to take suggestions the schedule creator offers and tailor it to their needs. For instance, a student may customize their schedule to have difficult load in one semester, so they may afford an easy load the next and be able to squeeze in a part-time job. Another example is students who are considered full time, but only want to take 12 units in a semester as opposed to 15. In addition, modifying suggested schedules allows a student to take classes out of personal interest, rather than to fulfill any requirements.

The long-term planner allows students to create an outline for graduation. The function offers requirements for easy review, so a student can ensure they have planned for all relevant classes, and allows a student to input their desired graduation date so the planner can offer tips that are more tailored to the student’s needs.

The planner will automatically check for requirements and prerequisites. The warning function alerts students of any decisions that will push back their possible graduation date (as compared to their desired graduation date) and the changes that can be made to resolve it.

Finally, the optimize change in major function offers aid to those who want to change their major. It allows students to match classes they’ve already taken to majors that can use it as part of their requirements, allowing them to make a smooth transition from one major to another. For instance, a third year Computer Science student looking to change majors will be suggested the Math major over a History major, as the student would have taken more classes that will count for the former.

## 2.3 User Characteristics

It is expected that all users are able to read and understand English, are familiar with the basic Graphical User Interface (GUI) components of Windows or Mac operating systems, and are comfortable using web browsers. In addition, they are presumed to possess a thorough understanding of the campus policies regarding class enrollment, course requirements, and graduation requirements.

## 2.4 General Constraints

The product will be integrated within myNorthridge Portal and SOLAR’s services; as such, the product must be compatible, draw from the same data sources, and be written in the same language, with the same format for documentation. This ensures that changes in one may be reflected in the other. In addition, access to data is governed by national and state legislation, and users must retain their privacy with special care given to security.

## 2.5 Assumptions and Dependencies

It is assumed that AI2 will run on the same web-infrastructure as myNorthridge Portal, which will be used to handle user login, security, and user access (that is, one student will not be able to obtain the information of another student from within their account). AI2 will also have access to the same databases as other services offered by myNorthridge Portal, including student information, major requirements, and the class catalog.

As CSUN’s web services already perform poorly under a number of circumstances, it is further assumed that the servers will be upgraded to handle the additional load as a result of this product. In addition, because the product will provide a service online, it is assumed any hardware used to access AI2 will have the necessary processing power.

Finally, some form of data is necessary for determining the relative difficulty of courses. It will be assumed that data can be gathered from course surveys, which include fields for specifying the estimated hours per week outside of class a student spends studying, as well as the student’s expected grade. In addition, it will be assumed that the system will have access to anonymized scores from each class, summarized by grade.

# 3. Specific Requirements

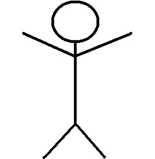
This section lists the function and non-function requirements of the software.

## 3.1 Functional Requirements

This section offers more insight in the functional requirements of the program. To reiterate, they are the ability to provide a difficulty rating, provide a comprehensive requirements chain, create a short-term schedule, modify the schedule, create a long-term planner, issue warnings for planner choices that impact graduation date, and optimize changes in major. Use-cases are provided for each function, with a priority of 5 being high and a priority of 1 being low.







Student



### 3.1.1 Provide Difficulty Rating

|  |  |  |
| --- | --- | --- |
| Number |  | 001 |
| Name |  | Provide Difficulty Rating |
| Summary |  | Each course is given a difficulty rating (a more refined estimate than from unit count alone), which is calculated from expected hours per week of studying and the pass/fail ratio. The higher the rating, the more difficult the class. This rating can be used by the student to better balance their choice of classes; in addition, the program will automatically use this when making semester suggestions. |
| Priority |  | 5 |
| Preconditions |  | Student is logged into their myNorthridge Portal Accessible Databases: Course, Student Grade, Difficulty Survey |
| Postconditions |  | Student presented with a course difficulty score for each class and their overall schedule. |
| Primary Actor(s) |  | Student |
| Secondary Actor(s) |  | CSUN Databases |
| Trigger |  | Student selects [SEARCH CLASSES] from the enrollment menu |
| Main Scenario | Step | Action |
|  | 1 | System displays list of courses, categorized by department, with a difficulty rating by each course |
|  | 2 | Student selects courses for the semester |
|  | 3 | System displays selected courses for student with individualized and summarized difficulty rating |
|  | 4 | System displays [Approve], [Modify], and [Cancel] options |
|  | 5 | Student clicks [Approve] |
|  | 6 | System stores this information and returns to pre-trigger screen |
| Extensions | Step | Branching Action |
|  | 3a | If difficulty rating is too high, the system will issue a warning (acceptable range of difficulty in this case is calculated based on the average of their combined courses; if it exceeds a predetermined threshold, it will be considered “too high”) |
|  | 3b | Student acknowledges warning |
|  | 3c | Return to step 4 |
|  | 5Aa | Student clicks [Modify] |
|  | 5Ab | Return to step 1 |
|  | 5Ba | Student clicks [Cancel] |
|  | 5b | System returns to pre-trigger screen |
| Open Issues |  |  |

### 3.1.2 Provide Comprehensive Requirements Chain

|  |  |  |
| --- | --- | --- |
| Number |  | 002 |
| Name |  | Provide Comprehensive Requirements Chain |
| Summary |  | Rather than providing prerequisites for one class at a time, this function provides the entire chain from start to finish. Although it may manually be used by students, its main purpose is to aid the program when making course suggestions (e.g. longer chains would have a higher chance of bottlenecking, and the program would take that into account when determining priority). |
| Priority |  | 5 |
| Preconditions |  | Student is logged into their myNorthridge Portal Accessible Databases: Course |
| Postconditions |  | The student receives an entire chain of requirements for the specified course, from the first prerequisite to the last class in the chain, as well as any co-requisite information. |
| Primary Actor(s) |  | Student |
| Secondary Actor(s) |  | CSUN Databases |
| Trigger |  | Student selects [CHECK PREREQUISITES AND POSTREQUISITES] |
| Main Scenario | Step | Action |
|  | 1 | System displays list of courses |
|  | 2 | Student selects course |
|  | 3 | System displays a flowchart of courses, with any prerequisites completed by the student appropriately marked. |
| Extensions | Step | Branching Action |
|  | 2a | Student cancels |
|  | 2b | System returns to pre-trigger screen |
| Open Issues |  |  |

### 3.1.3 Create Short-Term Schedule

|  |  |  |
| --- | --- | --- |
| Number |  | 003 |
| Name |  | Create Short-Term Schedule |
| Summary |  | Accesses necessary student information and user input to create an optimal schedule of classes to register for in the upcoming semester in accordance with the student’s graduation goals. |
| Priority |  | 5 |
| Preconditions |  | Student is logged into their myNorthridge Portal Accessible Databases: Class Catalog, General Education Requirement, Major, Miscellaneous Requirement, Student |
| Postconditions |  | User has accepted semester schedule or has returned to main menu |
| Primary Actor(s) |  | Student |
| Secondary Actor(s) |  | CSUN Databases |
| Trigger |  | Student selects [CREATE MY SCHEDULE] |
| Main Scenario | Step | Action |
|  | 1 | System displays available terms (a term is “available” when it has sufficient class availability information) |
|  | 2 | User selects desired term |
|  | 3 | System displays menu of available filters for class search, such as open/closed/waitlist or schedule preferences (e.g. no classes after 4pm). |
|  | 4 | User selects the desired filters |
|  | 5 | System creates and displays proposed schedules with the option to generate additional choices for a class, if available |
|  | 6 | User selects desired option |
|  | 7 | System updates schedule and returns to pre-trigger screen |
| Extensions | Step | Branching Action |
|  | 1a | System states no terms are available |
|  | 1b | User clicks to acknowledge |
|  | 1c | System exits to main menu |
|  | 4a | User cancels creation of schedule |
|  | 4b | System exits to main menu |
|  | 5Aa | System alerts user that no classes match search parameters |
|  | 5Ab | User clicks to acknowledge |
|  | 5Ac | Return to step 3 |
|  | 5Ba | User requests to see alternate system-generated choices for the requirement category of any class displayed |
|  | 5Bb | System displays next class choices |
|  | 5Bc | User selects one or goes back to original choices |
|  | 6Aa | User opts to Modify Short-Term Schedule |
|  | 6Ba | User cancels schedule creator |
|  | 6Bb | System asks if user is sure they wish to exit now |
|  | 6Bc | User clicks to confirm |
|  | 6Bd | System exits to main menu |
| Open Issues |  |  |

### 3.1.4 Modify Short-Term Schedule

|  |  |  |
| --- | --- | --- |
| Number |  | 004 |
| Name |  | Modify Short-Term schedule |
| Summary |  | The user takes a system-suggested schedule and tailors it to suit their needs, including being able to add, remove, replace, or otherwise modify classes. |
| Priority |  | 4 |
| Preconditions |  | Student is logged into their myNorthridge Portal Student has an existing short-term schedule (see: previous function) Accessible Databases: Class Catalog, General Education Requirement, Major, Miscellaneous Requirement, Student |
| Postconditions |  | Student has customized an error-free class schedule that meets their needs |
| Primary Actor(s) |  | Student |
| Secondary Actor(s) |  | CSUN Databases |
| Trigger |  | Student is within a suggested schedule and selects [MODIFY] |
| Main Scenario | Step | Action |
|  | 1 | System displays all selected classes with offered customization options |
|  | 2 | Student selects a class to modify |
|  | 3 | System displays choices (e.g. remove, swap) |
|  | 4 | Student selects an option |
|  | 5 | System checks final schedule and notifies user of any warnings (e.g. if user is below the necessary amount of units per semester to graduate) |
|  | 6 | Student acknowledges and confirms |
|  | 7 | System stores this information and returns to the pre-trigger screen |
| Extensions | Step | Branching Action |
|  | 2a  2b  2c  2d | Student keeps modifies no class but chooses to add a new one to the schedule  System offers a list of suggested classes as well as a list of all other classes  Student selects class(es) to add  Return to step 2 |
|  | 4a | Student selects any option that requires adding a new class (e.g. swap) |
|  | 4b | System provides list of available options |
|  | 4c | Student makes a selection |
|  | 4d | Return to step 7 |
|  | 6a | Student cancels any changes |
|  | 6b | System returns to pre-trigger screen |
| Open Issues |  |  |

### 3.1.5 Create Long-Term Planner

|  |  |  |
| --- | --- | --- |
| Number |  | 005 |
| Name |  | Create Long-Term Planner |
| Summary |  | The user makes a plan from their present point up until graduation |
| Priority |  | 3 |
| Preconditions |  | Student is logged into their myNorthridge Portal Accessible Databases: Courses, General Education Requirement, Major, Miscellaneous Requirement |
| Postconditions |  | Student has created an error-free planner that can be used as a guideline to obtain the necessary requirements to graduate |
| Primary Actor(s) |  | Student |
| Secondary Actor(s) |  | CSUN Databases |
| Trigger |  | Student selects [CREATE PLANNER] |
| Main Scenario | Step | Action |
|  | 1 | System displays the student’s requirements to serve as guideline |
|  | 2 | Student inputs desired graduation month and year. In addition, student selects courses they intend to take and place them in their planner, in the relevant section based on semester and year |
|  | 3 | System checks that all requirements are fulfilled, that prerequisites are properly accounted for, and that the plan matches with the desired graduation time |
|  | 4 | Student saves information |
|  | 5 | System stores this information and returns to pre-trigger screen |
| Extensions | Step | Branching Action |
|  | 2Aa  2Ab | Student arranges courses in such a way that their possible graduation date exceeds their desired graduation date  System Issues Impacted-Graduation Warnings for Planner |
|  | 2Ba | Student saves planner information but does not complete it |
|  | 2Bb | System returns to pre-trigger screen |
|  |  |  |
| Open Issues |  |  |

### 3.1.6 Issue Impacted-Graduation Warnings for Planner

|  |  |  |
| --- | --- | --- |
| Number |  | 006 |
| Name |  | Issue Impacted-Graduation Warnings for Planner |
| Summary |  | Warn the student if their intended plan will impact their desired graduation date |
| Priority |  | 3 |
| Preconditions |  | Student is logged into their myNorthridge Portal Accessible Databases: Courses, General Education Requirement, Major, Miscellaneous Requirement |
| Postconditions |  | Student has been warned of any risks their graduation plan possesses and has either accepted the risks or adjusted their plan |
| Primary Actor(s) |  | Student |
| Secondary Actor(s) |  | CSUN Databases |
| Trigger |  | Student submitted a plan for a semester where courses chosen will push back their desired graduation date |
| Main Scenario | Step | Action |
|  | 1 | System freezes planner and displays a warning |
|  | 2 | Student clicks warning |
|  | 3 | System displays details about how the course will impact their graduation date and provides a list of suggested courses to prevent the date from being pushed back |
|  | 4 | Student chooses from the list of options |
|  | 5 | System displays updated plan and unfreezes the planner |
| Extensions | Step | Branching Action |
|  | 4a | Student chooses to ignore the warning |
|  | 4b | System unfreezes the planner |
| Open Issues | 1 | The system gives the student final control on whether or not to ignore a warning. Is it safe to assume that the student will properly handle it, or should we require they explain their reasoning first and have an advisor manually approve or deny it? |

### 3.1.7 Optimize Changes in Major

|  |  |  |
| --- | --- | --- |
| Number |  | 007 |
| Name |  | Optimize Changes in Major |
| Summary |  | Provide a list of optimal major changes based on the student’s completed classes and a major’s prerequisites |
| Priority |  | 2 |
| Preconditions |  | Student is logged into their myNorthridge Portal Accessible Databases: Course, Major, Student |
| Postconditions |  | Student has either decided to change their major or stay with what they have. If the former, they have been notified of which classes they’ve taken are applicable to the new major and have been directed to a new page that will allow them to formally request a change in major. |
| Primary Actor(s) |  | Student |
| Secondary Actor(s) |  | CSUN Databases |
| Trigger |  | Student selects [MAJOR CHANGE SUGGESTIONS] |
| Main Scenario | Step | Action |
|  | 1 | System gathers data on courses the student has already taken and displays a list of majors based on the proportion of the student’s previous classes that can be reused to fulfill new major requirements |
|  | 2 | Student selects a major |
|  | 3 | System directs student to the relevant page to formally request a change in major |
| Extensions | Step | Branching Action |
|  | 2Aa | Student requests to see more majors |
|  | 2Ab | System loads an additional batch of majors for comparison (these choices are not as optimized as the previous batch) |
|  | 2Ac | Return to step 2 |
|  | 2Ba | Student specifies a major rather than taking a suggestion |
|  | 2Bb | System compares chosen major with current major |
|  | 2Bc | Return to step 2 |
| Open Issues |  |  |

## 3.3 Non-Functional Requirements

### 3.3.1 Performance

Response time. 99% of queries must return a response in < 2 seconds. 95% of all visible pages respond in 6 seconds or less, including infrastructure. Response times shall be measured using HP LoadRunner (or similar tool) located behind the firewall and in front of the web servers. Time measured will be from request to when last bit required to render page is returned. Performance tests shall not exceed 65% CPU utilization during the busy hour.

Workload. The software shall support 43,000 total users, which, on a busy day, will generate 5,000 interactions.

Platform. The software shall be web-based and platform-agnostic.

### 3.3.2 Reliability

The system shall have 99% probability of operating without failure for 1000 transactions.

System shall pass 98% of fault injection tests prior to date of delivery. Testing shall be done by Mu Service Analyzer or similar Network Level Fault Injector.

### 3.3.3 Availability

Uptime shall be commensurate with SOLAR and should be available 100% of the time that SOLAR is available, excluding scheduled maintenance. Should internet service be disrupted while sending information to the server, the information can be sent again for verification within 20 seconds of reconnection.

The user may restart after failure with loss of 1 transaction at most.

### 3.3.4 Security

Access to this system is via myNorthridge Portal username and password only.

User history logs shall be maintained for a minimum of 7 days for each user.

### 3.3.5 Maintainability

Mean Time to Change (MTTC) for known defects shall be less than 2 person-days by a software developer with 1-year experience.

### 3.3.6 Portability

The software must be written character set-neutral and endian-neutral.

# 4. Deliverables

On top of the software requirements specification, the final products to be delivered are the system design document, user interface document, step-by-step guides for students, and manuals for staff. In addition, the software’s source code and executable program will be provided via flash drive and cloud.

The system design document is to be delivered via Moodle on November 2, 2016; the rest is to be delivered to the college on December 12, 2017.

# 5. Open Issues

To reiterate from section 3, an open issue is whether or not a student is given too much freedom to ignore a warning. At the end of the day, the college will not force a student’s hand; if a student chooses not to take a prerequisite or a required course, there is nothing the college can do. However, the college assumes that the student will take responsibility; for instance, if they opt out of a prerequisite, the college assumes it would be because they intended to take the course at another institution and have the credits transferred. Is this wise, or should the college enforce more stringent steps to ignore a warning, other than simply clicking an acknowledgement?