



Software Requirements Specification

for

Majorizor

Version 2.1 Approved

Prepared by Emily Campbell, Jackson DeMeyers, Joseph Mortefolio,
and Lingling Yao

May 2, 2017

Table of Contents

1 - Revision History	5
2 - Introduction	6
2.1 - Purpose	6
2.2 - Intended Audience and Reading Suggestions	6
2.3 - Project Scope	6
3 - Overall Description	8
3.1 - Product Perspective	8
3.2 - Product Features	8
3.3 - User Groups and Characteristics	8
3.4 - Operating Environment	9
3.5 - Design and Implementation Constraints	9
3.6 - Assumptions and Dependencies	9
4 - System Features	10
4.1 - Account Creation	10
4.1.1 - Description and Priority	10
4.1.2 - Stimulus/Response Sequences	10
4.1.3 - Functional Requirements	11
4.2 - Login	11
4.2.1 - Description and Priority	11
4.2.2 - Stimulus/Response Sequences	11
4.2.3 - Functional Requirements	12
4.3 - Change User Group	12
4.3.1 - Description and Priority	12
4.3.2 - Stimulus/Response Sequences	12
4.3.3 - Functional Requirements	12
4.4 - Delete User	13
4.4.1 - Description and Priority	13
4.4.2 - Stimulus/Response Sequences	13
4.4.3 - Functional Requirements	13

4.5 - Master Schedule Import	13
4.5.1 - Description and Priority	13
4.5.2 - Stimulus/Response Sequences	14
4.5.3 - Functional Requirements	14
4.6 - Major and Minor Selection	14
4.6.1 - Description and Priority	15
4.6.2 - Stimulus/Response Sequences	15
4.6.3 - Functional Requirements	16
4.7 - Progress Tracking	16
4.7.1 - Description and Priority	16
4.7.2 - Stimulus/Response Sequences	16
4.7.3 - Functional Requirements	17
4.8 - Schedule Creation	17
4.8.1 - Description and Priority	17
4.8.2 - Stimulus/Response Sequences	17
4.8.3 - Functional Requirements	18
4.9 - Advisor Overview	19
4.9.1 - Description and Priority	19
4.9.2 - Stimulus/Response Sequences	19
4.9.3 - Functional Requirements	19
4.10 - Advisee Selection	20
4.10.1 - Description and Priority	20
4.10.2 - Stimulus/Response Sequences	20
4.10.3 - Functional Requirements	20
5 - External Interface Requirements	22
5.1 - User Interfaces	22
5.1.1 - Navigation Bar	24
5.1.2 - Student Screens	24
Student Portal	24
Major and Minor Selection	25
Schedule Builder	26
5.1.3 - Advisors Screens	26
Advisor Portal	26

Select Advisees	27
5.1.4 - Admins	28
Admin Portal	28
5.2 - Hardware Interfaces	29
5.3 - Software Interfaces	29
5.4 Communication Interfaces	30
6 - Other Nonfunctional Requirements	31
6.1 - Performance Requirements	31
6.2 - Safety Requirements	31
6.3 - Security Requirements	31
6.4 - Software Quality Attributes	31
7 - Key Milestones	32
8 - Other Requirements	33
9 - Requirement Change Management	34
Appendix	35
Appendix A: Glossary	35

1 - Revision History

Name	Date	Reason For Changes	Version
All	3/2/2017	Initial Working Draft	1.0
Jackson DeMeyers	4/30/2017	Recommended Improvements	1.1
All	5/1/2017	Updated Section 4 - System Features to reflect what is currently implemented	2.0
Jackson DeMeyers	5/2/2017	Updated section 5.1 with Images of current UI	2.1

2 - Introduction

2.1 - Purpose

This document details the software requirements for the Majorizer project for Clarkson University.

At Clarkson University, many students decide to take on multiple majors and minors during their undergraduate career. However, it is difficult for both students and advisors to determine if the student has enough room in their schedule to allow for the extra major and minors. It is also difficult for both students and advisors to track how far they are through their course work.

The purpose of the Majorizer project is to provide an easy way for both students and their advisors to manage additional minors and double majors in one centralized place. As per discussion with our customer, the project scope should be limited to the ECE, Math and Computer Science departments.

2.2 - Intended Audience and Reading Suggestions

This Software Requirements document is intended for:

- Developers - To review the project's scope and capabilities, and to better understand which areas of the project they should focus their efforts on.
- Testers - To understand the expected behavior of each feature, and determine whether or not the features perform as intended. This document should help keep testing organized and methodical.
- End Users - To understand the services made available by the system to each user group.

2.3 - Project Scope

Majorizer is an application built as an ASP.NET web application using the Bootstrap framework, with a MySQL database backend. It provides tools that allow students and advisors to easily plan, schedule, and track students' academic progress through multiple majors and minors in the ECE, Math, and Computer Science departments at Clarkson University.

Majorizer offers ten general services:

- Account Creation
- Login
- Change User Group
- Delete User
- Master Schedule Import
- Major and Minor Selection

- Progress Tracking
- Schedule Creation
- Advisor Overview
- Advisee Selection

ASP.NET Session State Overview. - [Web].

Available: <https://msdn.microsoft.com/en-us/library/ms178581.aspx>

Bootstrap Core Team. N.d. Bootstrap - Getting Started (v4.0.0-alpha.6) [Web].

Available: <http://v4-alpha.getbootstrap.com/getting-started/browsers-devices/>

3 - Overall Description

3.1 - Product Perspective

Majorizor is a new, independent system that Clarkson University students and advisors can use to schedule undergraduate courses and monitor academic progress. Currently, Clarkson University uses the PeopleSoft system for class enrollment and changes to majors and minors. The disadvantage of PeopleSoft is that it does not provide students with a complete list of the course requirements they need to graduate; the task of selecting courses and fulfilling academic requirements is left to students and advisors. Majorizor offers a means for students and advisors to automate the process of course selection and progress tracking.

3.2 - Product Features

The Majorizor system provides the following features:

- Account Creation
- Login
- Change User Group
- Delete User
- Master Schedule Import
- Major and Minor Selection
- Progress Tracking
- Schedule Creation
- Advisor Overview
- Advisee Selection

3.3 - User Groups and Characteristics

The Majorizor application is intended for use by three groups of users:

1. USER (student)
 - Student users can manage and track their own accounts, but not able to see other user accounts of any type, or have access to advisor and admin user features.
 - New user accounts will be registered as this level.
2. ADVISOR
 - Advisors will be able to manage and track all students which they select as their advisees.
 - Advisors consist solely of approved Academic Advisors at Clarkson University.
 - Advisors must register a normal account, and then request to be elevated to an Advisor account by emailing a current Majorizor Admin.
 - Advisors can access and use the system as both an ADVISOR and USER account.
3. ADMIN

- Administrators can manage the permissions of Majorizer users and handle master schedule imports. They also have backend access to the application.
- These users should be developers or specifically trained advisors. They will need to be granted admin access through another administrator.
- Admins can access and use the system as both an ADMIN and ADVISOR account.

3.4 - Operating Environment

Majorizer is platform independent, but requires browsers which support Bootstrap, as detailed in Bootstrap's documentation: <http://v4-alpha.getbootstrap.com/getting-started/browsers-devices/>

The Bootstrap framework is built to be compatible with most devices, including mobile and touch devices. This will allow our application to also work on mobile devices, although this is not a constraint or priority for this application.

Due to time constraints, the Majorizer application will be tested on only Windows 10 using the Google Chrome browser.

3.5 - Design and Implementation Constraints

Majorizer is platform independent and is written in ASP.NET C# and uses the Bootstrap framework. It also has a MySQL Database backend. Anyone who wishes to work on the Majorizer project should have knowledge with ASP.NET, C#, Bootstrap, jQuery, and MySQL.

3.6 - Assumptions and Dependencies

Majorizer is dependent on having a device running a browser compatible with Bootstrap, as discussed in Bootstrap's documentation:

<http://v4-alpha.getbootstrap.com/getting-started/browsers-devices/>

4 - System Features

Majorizor consists of nine main features. Below are descriptions of each system feature and the functional requirements the features necessitate. The purpose of this section is to clearly outline how each feature can be accessed, and to what ends it may be used.

4.1 - Account Creation

Majorizor requires users to login to the system in order to access its features. Before a user can login to the system, they must create an account.

4.1.1 - Description and Priority

The Majorizor system allows users to register an account. Users will use these accounts to login to the system each time they visit. The accounts are used to store information about each individual user in the MySQL database, including email, first and last name, securely hashed passwords, user group, and other application specific data. Since accounts are required to use the system, this is a high priority feature.

4.1.2 - Stimulus/Response Sequences

User's can access the account creation feature by visiting the Registration tab on the Login/Registration page of the website, easily accessible from the navbar when not currently logged in.

Once on the Register tab, users will be required to enter their first and last name, email address, and password. Passwords must be entered twice to verify there are no typos. After all the fields are filled out, the user will click the "Register" button, which will send the provided information to the backend to attempt to register a new user.

First, on the application side, all fields are checked to make sure they are not empty and that the provided passwords match. If any of these cases are false, then the application throws an exception, which is then caught and displays a detailed warning message to the user. If all cases are true, then the password is salted with a randomly generated 10 character salt, and then hashed using a SHA256 hashing method provided by the `System.Security.Cryptography` namespace.

Next, the user's first and last name, email, password, and randomly generated salt are passed to a stored procedure on the database which does an insert on both the user and user_password tables to fill the corresponding rows. The stored procedure checks that the email provided is unique to all other emails in the database to ensure no two users have the same email. If the stored procedure fails to register a user for any reason, then an exception is caught and the error

is displayed in a error message on the application's UI. If the registration is successful, the application automatically logs the new user into the system and redirects them appropriately.

Upon registration, each user is given a unique userID. This userID will be used as a foreign key constraint anytime the user is references in another database table to store application data.

4.1.3 - Functional Requirements

- **REQ-1:** Accounts require a first and last name, email, and password.
- **REQ-2:** Passwords must be randomly salted and then hashed. The salt must be stored in the database to allow for re-hashing upon login.
- **REQ-3:** Passwords must be entered twice to verify no typos exist.
- **REQ-4:** No to users can share an email address.
- **REQ-5:** Any fail case of the register feature must display an error/warning to the user.

4.2 - Login

Majorizor requires users to login to the system in order to access its features.

4.2.1 - Description and Priority

The Majorizor system requires users create accounts, and use the accounts to login and access the system. Once a user has registered a new account, each time they visit the application, they will go to the Login/Registration page, accessed easily via the navbar when not currently logged in. Since users are required to login, this feature is high priority.

4.2.2 - Stimulus/Response Sequences

Once on the login screen, a user must enter their email address and password they used to register their account and press the "Login" button. If either the email or password fields are empty, a warning is displayed to the user and Login is unsuccessful.

When all fields are set and the Login request is made, a stored procedure is called to retrieve to retrieve the salt stored in the database for the provided email. Once the salt is retrieved, then the provided password is hashed using the same hashing algorithm from the account creation feature. Then, the newly hashed password and the user's email are sent to another stored procedure which returns the userID of the user with the provided email if the newly hashed password matches the stored password. Next, a User object is created using the returned userID, and the user's email, userID, and user group are stored in the application's Session variable. Since each instance of the application has its own Session variable, multiple users can be logged into the system at the same time.

If at any point the database determines the user does not exist, or that the provided password is incorrect, then an exception is generated, the login is unsuccessful, and a detailed error is displayed to the user informing them that either their email or password are incorrect.

4.2.3 - Functional Requirements

- **REQ-1:** User must provide both email and password.
- **REQ-2:** Passwords must be re-salted and hashed before being passed to the MySQL database to attempt login.
- **REQ-3:** Any fail case of the login feature must display an error/warning to the user.

4.3 - Change User Group

Admin users should be able to change the user group of any user registered to Majorizor. This is required to elevate new accounts to both ADVISOR and ADMIN user groups when required.

4.3.1 - Description and Priority

When a new administrator or admin registers to the system, they will be registered, by default, as a normal student USER. Thus, this feature is required to elevate new advisor and administrator to these user groups. Administrators currently registered to the system will be able to login to the Admin Portal and use the User Management panel to control the user group of every user registered in the system. Since this can easily be accomplished by an Administrator with backend access to the MySQL database, this feature is low priority.

4.3.2 - Stimulus/Response Sequences

After a new Majorizor advisor or administrator register to the system, their accounts must be escalated from USER to either ADVISOR or ADMIN. Administrators already registered in the system can access this feature from the Admin Portal page, using the User Management panel.

Within the User Management panel exists a searchable table with the following rows: Name, Email, User Group, Change, Delete User. For each row in the table, the "User Group" column shows the current user group for the corresponding user. In the "Change" column, there is a dropdown list with three options: USER, ADVISOR, and ADMIN.

The logged-in administrator can search for any user in the table and use the drop down list to change the user's current user group to the user group selected by the drop down list. When the drop down list's selected item changes, the selected user group and userID are passed to a stored procedure which will update the user table in the MySQL database for the given UserID. If the change is successful, then the table should be re-loaded to display the result. If the change is unsuccessful, then the application should throw an exception and display a detailed error to the user.

4.3.3 - Functional Requirements

- **REQ-1:** Administrators can update the user group of any registered user to any user group.
- **REQ-2:** Any fail case of the change user group feature must display an error/warning to the user.

4.4 - Delete User

Admin users can delete any user from the Majorizor system. This will allow any old or unnecessary accounts to be removed from the system when required.

4.4.1 - Description and Priority

Accounts may be accidentally registered, setup incorrectly, or become unneeded. When this occurs, administrators will be able to remove any account from the system via the Admin Portal's User Management panel.

4.4.2 - Stimulus/Response Sequences

Within the User Management panel exists a searchable table with the following rows: Name, Email, User Group, Change, Delete User. For each row in the "Delete User" column, there is a delete button. When pressed, the delete button will remove the corresponding user from the Majorizor system. A stored procedure is executed to remove the account from the user table in the MySQL database, which then causes a series of triggers and cascading delete statements to remove all references of the user from all tables in the system. Once deleted, a user's account cannot be recovered.

4.4.3 - Functional Requirements

- **REQ1:** Admins can remove any user from the system.
- **REQ2:** Removal should remove all references to the provided user from all MySQL database tables, permanently deleting the user.

4.5 - Master Schedule Import

Majorizor requires Clarkson's master schedule to be stored in the MySQL database in order to allow for schedule creation. Since Majorizor is an independent system from all other Clarkson services, the application must provide a feature to import the master schedule, exported from Peoplesoft, into the system.

4.5.1 - Description and Priority

Majorizor is an independent system at Clarkson University, but requires schedule information from Clarkson's Peoplesoft system to enable the schedule creation feature. As such, a method for parsing and importing the master schedule export from Peoplesoft into Majorizor must exist.

This feature will only add courses from the Peoplesoft export if subject is equal to EE, MA, CS. It will also add FY100, ES100, ES110, ES220, ES222, ES250, ES260, ES340, ES405, ES499, CM131, CM132, PH131, PH132, UNIV190, STAT383 and FY100. These courses currently include all required courses for all majors and minors Majorizor provides, excluding knowledge area and electives. This list is subject to change as the majors and minor requirements at Clarkson University changes.

Although this feature can easily be accomplished by an administrator with backend database access, this feature will be used at least once per semester. Thus, this feature has a medium priority.

4.5.2 - Stimulus/Response Sequences

This feature can be accessed by administrator accounts in the Master Schedule Import panel on the Admin Portal page.

Once an administrator has the master schedule export from Peoplesoft, the file should be exported into a “|” delimited .csv file. To change this go to Control Panel > Clock, Language, and Region > Change date, time, or number formats > Additional Settings > change the List separator to a “|” instead of “,” (Win10). Then, save the Peoplesoft exported schedule to a .csv. It will be pipe delimited.

After the file is saved as a “|” delimited .csv file, an administrator can use the Browse button in the panel to select the .csv file from their local disk, and then click the Upload button to begin processing. The file is read, line by line, and if the course matches any courses listed above, the data from the line, including courseID, subject, catalog, section, name, startTime, endTime, and day, is inserted into a Course object, and then added into a list of Course objects.

Once the entire file has been parsed, the Course list is passed to a method which, for each Course in the list, inserts into the master_schedule table in the MySQL database. If the semester's schedule had already been inserted into the system, any changes are applied to the database. If all inserts are successful, the schedule import feature was successful. If the parser is unable to read the file, or the courses are unable to be inserted into the database, then an exception is thrown and a detailed error is displayed to the user.

4.5.3 - Functional Requirements

- **REQ-1:** Admins can upload a master schedule .csv into the system.
- **REQ-2:** Perform an insert or update upon inserting into the database. This will allow for any changes made to section, start and end time, and days of the week to be updated in the database.
- **REQ-3:** .csv file must be “|” (pipe) delimited.
- **REQ-4:** Only required courses by Majorizer's provided majors and minors, excluding knowledge areas and electives, should be processed.
- **REQ-5:** Any fail case of the master schedule import feature must display a detailed error/warning.

4.6 - Major and Minor Selection

The Majorizer system is built to assist students and advisors with planning undergraduate careers. Thus, one of the most important features is the ability to select both Majors and Minors.

4.6.1 - Description and Priority

Student users can select up to two majors and two minors. On first time login, students will be required to select at least 1 Major before continuing. A student may add or remove majors at any point throughout their undergraduate career. Since selecting majors and minors is a core feature of the Majorizer system, there is a high priority requirement.

4.6.2 - Stimulus/Response Sequences

For Student users, the option to add and remove majors is available in two locations; at the bottom of the Student Portal, and via the Major/Minor Selection page, accessible through the navbar. For advisor users, this option is available on the Advisor Portal via a pop-up modal for each user activated by clicking the Select Majors/Minors button in the student's panel.

There are four key functionalities that this feature encompasses; the addition and removal of academic majors and minors. These functionalities are made available when the below conditions are met. The scope of the Majorizer system is limited to majors within the ECE, Computer Science, and Math departments. These departments encompass five majors; Electrical, Computer, and Software Engineering, Math, and Computer Science. They also encompass minors for each available major except for Computer Engineering.

➤ **Add Major**

This functionality is only available on empty major slots, otherwise the Remove Major functionality is enabled for that slot.

Upon pressing the "Add" button next to an available major slot, a popup with a dropdown list of available majors the student is not yet committed is displayed to the user. When the user selects a major from the list, it is updated in the MySQL database, and the Student object within the application is updated to reflect the new major. The major slot is filled with the new major, and the "Add" button is changed to a "Remove" button.

➤ **Add Minor**

This functionality is only available on empty minor slots, otherwise the Remove Minor functionality is enabled for that slot.

Upon pressing the "Add" button next to an available minor slot, a popup with a dropdown list of available minors the student is not yet committed is displayed to the user. When the user selects a minor from the list, it is updated in the MySQL database, and the Student object within the application is updated to reflect the new minor. The minor slot is filled with the new minor, and the "Add" button is changed to a "Remove" button.

➤ **Remove Major**

This functionality is only available on filled major slots, otherwise the Add Major functionality is enabled for that slot. Upon pressing the "Remove" button next to a filled

major slot, the major will be removed from the database, the Student object, and the major slot. The major slot then enables the Add Major functionality.

Currently, all majors can be removed from a student's account. In the future, this should be modified to always require at least one major.

➤ **Remove Minor**

This functionality is only available on filled minor slots, otherwise the Add Minor functionality is enabled for that slot. Upon pressing the "Remove" button next to a filled minor slot, the minor will be removed from the database, the Student object, and the minor slot. The minor slot then enables the Add Minor functionality.

4.6.3 - Functional Requirements

- **REQ-1:** The system will allow users to select up to two majors and two minors for any student.
- **REQ-2:** Any changes made during Major/Minor selection will cause all features which use the object to be reloaded to reflect the changes.
- **REQ-3:** Only non-selected majors/minors should be displayed to the user to select.

4.7 - Progress Tracking

Students should be able to view their progress report on their students home page and advisors should be able to view their advisees' progress report on their home page.

4.7.1 - Description and Priority

Students can view their progress report on their homepage which include the courses they had completed, the courses they are taking right now and their selection for future courses. Advisor should be able to view their advisees and only their advisees' progress report by clicking the "View Progress" button on their homepage.

4.7.2 - Stimulus/Response Sequences

➤ **View Progress in advisor home page**

Advisor should be able to view their advisees' progress reports. So far the percentage of progress bar is calculated by the required courses that advisee had taken and divided by the total required courses, then times a hundred. When advisor click the "View Progress" button, a new window should pop up and display advisee's progress report as well as the required courses an advisee had completed.

➤ **View Curriculum Outline in student home page**

Students can keep track of their progress on their home page. There is a button called "View Curriculum Outline" in the student home page and once students click

the button, it will bring students to a new page and display their current status in the Clarkson University. The Majorizer will display a progress bar for students which they provided a easy view. There are three different color for the progress bar to help them keep track their academic. The red color means students did not complete more than 30% of the required courses. The yellow color means students did complete more than 30% but less than 70% of the required courses. The green color means student are completed more than 70% of their required courses.

4.7.3 - Functional Requirements

- **REQ-1:** The system will allow students to view their progress reports.
- **REQ-2:** The system will allow advisors to view their advisees' progress reports.
- **REQ-3:** The system will show three different colors in the progress bar.

4.8 - Schedule Creation

Once student users have specified their majors and minors, the Majorizer system maps out users' undergraduate schedule.

4.8.1 - Description and Priority

The Majorizer generates an undergraduate schedule for student users based on the required courses for each major and minor to which the student has committed. The system also accounts for pre- and co-requisites, as well as limitations on semester offerings when creating an undergraduate schedule. This feature is one of the core functions of the Majorizer system, so it is classified as high priority.

4.8.2 - Stimulus/Response Sequences

The generation of a student user's schedule is an automated process that runs each time the Add Major, Remove Major, Change Major, Add Minor, Remove Minor, or Change Minor service is used. The user may edit their generated schedule in one of three ways: by adding a class, moving a class, or removing a class. These services are available as buttons beneath the student user's generated schedule on the 'View Schedule' page.

➤ **Add Class**

Majors at Clarkson University require the student to complete Knowledge Area (KA) and Elective courses. Students can choose from an array of courses which meet the KA and Elective requirements. Because students have a variety of options, the Majorizer system will not automatically add KA and Elective courses when generating a student user's schedule. When a student user is adding a class, the system will automatic look back on student's class history and check to see if the student meet all the requirement and prerequisite for this class.

When a student uses the 'Add Class' service, they will be taken to the master list of courses. From here, they can select a class they would like to add to their schedule. Once they have selected a class, the Majorizer system will prompt the user to select a semester to add the class to from a drop down menu. The options will be presented as '<semester> <year>'; for example, a student user might select 'Fall 2018' as the semester to which to add their selected class. The system will then prompt the user to 'Confirm' or 'Cancel' their changes with two respectively labeled buttons.

In the case that a new schedule is generated for the user after they have added a class, the user-added class will keep its place in the schedule. The only exception to this is if the generated schedule changes the location of an added class's prerequisites or co-requisites. In this case, the system will remove the added class from the schedule and inform the user via system message that a class has been removed from their schedule.

➤ **Move Class**

Students are able to move their class if there is an open position for another time slot without having conflict with other class period. When click the button "move class", the new window will display and asked for student users confirmation by clicking the confirm/cancel button. Once the student users click the "confirm" button, a message will display in the screen to notify student users that the class had been successfully move from schedules.

➤ **Remove Class**

Students are able to remove their class anytime through the semester before the final exam week. In the beginning of the semester, student user can remove their class directly by clicking the "remove" button in the page, after student user confirm the action, class will be remove from student schedule. However, removing class in the middle or at the end of the semester will require student user to file a add/drop form. Student user need to communicate with their course professor and advisor in order to remove class. In the add/drop form, signatures from course professor and advisor are both required. After student user complete the add/drop form, the administrator will remove class for student and update the new class schedule in both student and advisor landing page.

4.8.3 - Functional Requirements

- **REQ-1:** The system will provide service for users to add major, change major, remove major service only applied to student who has more than one majors.
- **REQ-2:** The system will provide service for user to add minor, remove minor, or change minor.
- **REQ-3:** The system will prohibit the user from adding or moving a class to a semester at the start of which the prerequisites and co-requisites for the class have not yet been completed.
- **REQ-4:** The system will not permit a user to add a class already in the user's schedule.
- **REQ-5:** The system will not permit a user to add a class equivalent to a class already in the user's schedule.

4.9 - Advisor Overview

In the advisor overview page, the advisors should be able to view their current advisees' records which including their name, majors, minors if they have one, expected graduation date and progress reports. Since having a minor is option for Clarkson student,, if an advisee does not have a minor, it will display as N/A for minor.

4.9.1 - Description and Priority

The Majorizer generate advisees' records and display in the advisors' home page. Advisors can only view their current advisees' records. This feature is one of the core functions of the Majorizer system, so it is classified as high priority.

4.9.2 - Stimulus/Response Sequences

The generation of an advisor overview an advisee is an automated process that runs each time the when an advisor press view schedule, view progress and note. Each button will bring advisor to a new pop up window and display the information for the corresponding advisee.

➤ **View Schedule**

Advisors should be able to view their advisees' current schedule, preview schedule and their selection courses for the following semester in order to give an advice. When advisor click the "View Schedule" button, a new window will pop up and display advisee's current schedule, pass schedule and their selection courses for next semester if there is one.

➤ **View Progress**

Advisor should be able to view their advisees' progress reports. So far the percentage of progress bar is calculated by the required courses that advisee had taken and divided by the total required courses, then times a hundred. When advisor click the "View Progress" button, a new window should pop up and display advisee's progress report as well as the required courses an advisee had completed.

➤ **Note**

Advisor can leave a note for their advisees for future references. When advisor click the "Note" button, a new window should pop up and allow advisor to write on, once advisor done with writing, he/she can click the submit button and the note will delivered to advisees as well as to their email that they use to register for the Majorizer.

4.9.3 - Functional Requirements

- **REQ-1:** The system will permit advisors to view only their current advisees' records.

- **REQ-2:** The system will provide service for advisor to view their current advisees' class schedule.
- **REQ-3:** The system will provide service for advisor to view their current advisees' progress reports.
- **REQ-4:** The system will provide service for advisor to leave a note for their current advisees.

4.10 - Advisee Selection

Advisors should be able to select a new advisee and remove advisee on the Advisee Selection Page. Advisors can search for an advisee by their name.

4.10.1 - Description and Priority

The Majorizer generate a list of user who has registered an account in Majorizer and appear under the Advisee Selection. Only those users registered as students will show up in the Advisee selection page. This feature is one of the core functions of the Majorizer system, so it is classified as high priority.

4.10.2 - Stimulus/Response Sequences

The generation of a advisee selection is an automated process that runs each time the when an advisor press Add or Remove button. Advisor can either add or remove advisee on the page.

➤ **Add Advisees**

Advisor are able to add a new advisee under add advisees section and the advisee does not have an advisor so far. Once the advisor click the "Add" button corresponding to the advisee, the advisee will remove from the add advisee section and add to current advisee section. Then the advisor should be able to view advisee information under advisor overview page.

➤ **Remove Advisees**

The students under the current advisee section are the current advisee for an advisor. Next to each advisee, there is a button call "Remove" which allows the advisor to remove them from the current advisee section. Once the advisor click the "Remove" button, the advisee is no longer assign to this advisee, then the advisee will appear to add advisee section and waiting for a new advisor.

4.10.3 - Functional Requirements

- **REQ-1:** The system will provide service for advisor to add or remove advisee on the Advisee Selection page.
- **REQ-2:** The system will provide service for advisor to search for an advisee.

- **REQ-3:** The system will permit an advisee show in both Current Advisee and Select Advisee sections. .

5 - External Interface Requirements

5.1 - User Interfaces

Majorizor uses a clean and minimal user interface which is consistent throughout all pages. The application uses Bootstrap, which provides many UI elements used throughout the application, including the following.

- Buttons



Figure 1 Primary Button

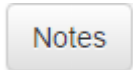


Figure 2 Secondary Button

- Navigation bar

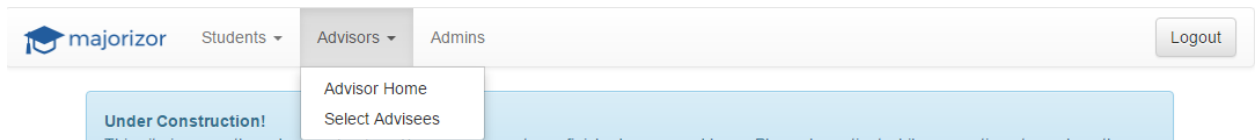


Figure 3 Main Navigation Bar

- Tabbed containers

Login

Register

Login

Email Address

demeyejg@clarkson.edu

Password

.....

Login

Not already a member? [Register](#)

Figure 4 Login/Registration Tab - Login

Login

Register

Register

First Name

Last Name

Email Address

Password

Verify Password

Register

Already a member? [Login](#)

Figure 5 Login/Registration Tab - Registration

- Panels

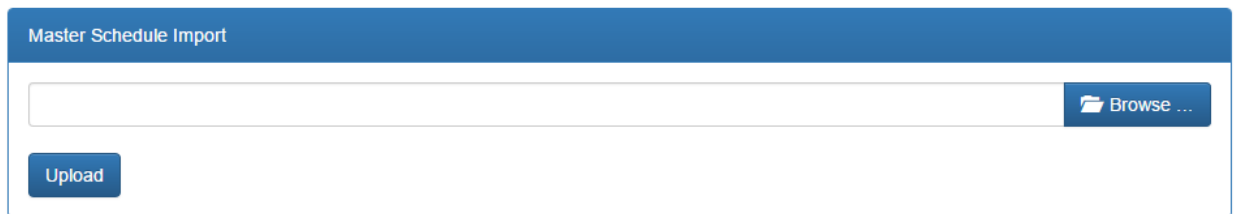


Figure 6 Bootstrap Panel

- Progress Bars



Figure 7 Bootstrap Progress Bar

5.1.1 - Navigation Bar

The navigation bar allows for easy navigation through the application. It is divided into three different sections, Students, Advisors, and Admins, each only visible if the logged-in user is at least the corresponding user group.

For example, the following images are the navigation bars for student and advisor users, respectively.

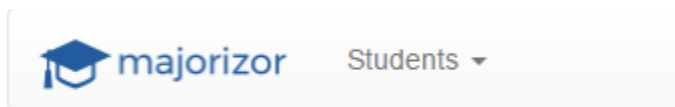


Figure 8 Student Navbar Options

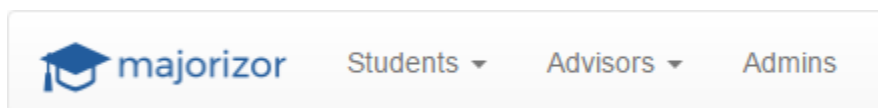


Figure 9 Admin Navbar Options

5.1.2 - Student Screens

Student Portal

The student portal provides an overview of the logged-in student's informations, and gives them quick access to view their schedules, track their progress, and change majors and minors.

Included features:

- Major and Minor Selection
- Progress Tracking
- Schedule Creation

Student Portal

Student Profile

Name: Tim Tester	Year: Sophomore	Majors: Electrical Engineering, Math
Advisor Name: Amy Advisor	Expected Graduation: Spring 2021	Minors: Electrical Engineering

Schedule Information

View Scheudle History:

View Semester Schedule

View Curriculum Outline:

View Curriculum Outline

Progress

Change Major/Minor

Select Majors

Major 1:

Electrical Engineering

×

Major 2:

Math

×

Select Minors

Minor 1:

Electrical Engineering

×

Minor 2:

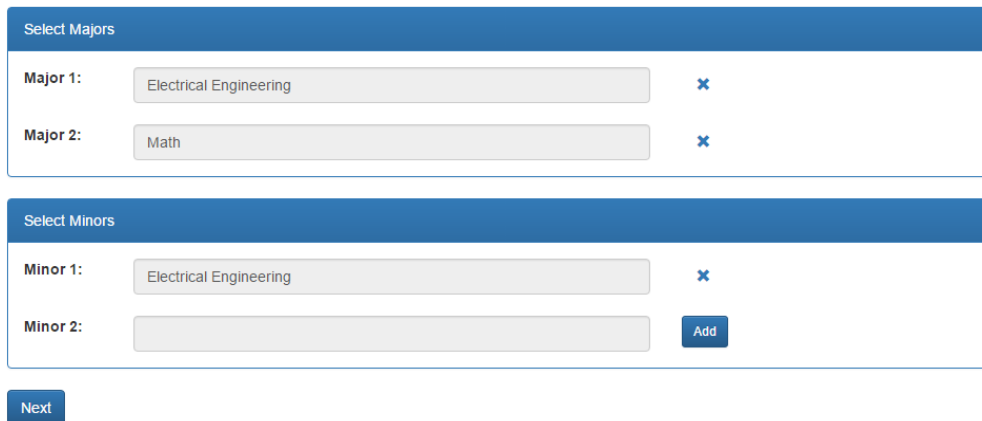
Add

Figure 10 Student Portal

Major and Minor Selection

This screen is dedicated to the Major and Minor Selection feature.

Major and Minor Selection



The image shows a user interface for selecting majors and minors. It consists of two main sections: 'Select Majors' and 'Select Minors'. The 'Select Majors' section has two rows: 'Major 1:' with a text input containing 'Electrical Engineering' and a blue 'x' button, and 'Major 2:' with a text input containing 'Math' and a blue 'x' button. The 'Select Minors' section has two rows: 'Minor 1:' with a text input containing 'Electrical Engineering' and a blue 'x' button, and 'Minor 2:' with an empty text input and a blue 'Add' button. Below these sections is a blue 'Next' button.

Figure 11 Major and Minor Selection UI

Schedule Builder

This screen is dedicated to the Schedule Creation feature.

5.1.3 - Advisors Screens

Advisor Portal

This screen allows advisors to view an overview of all of their selected advisees. It displays basic student information, and provides buttons to quickly change majors and minors, view progress, and view and modify schedules.

Included features:

- Major and Minor Selection
- Progress Tracking
- Schedule Creation

Advisor Portal

Tim Tester

Majors:

Electrical Engineering, Math

Year:

Sophomore

Minors:

Electrical Engineering

Expected Graduation:

Spring 2021

Select Majors/Minors

View Progress

View Schedule

Notes

53%

James Childs

Majors:

Computer Science

Year:

Sophomore

Minors:

N/A

Expected Graduation:

Spring 2021

Select Majors/Minors

View Progress

View Schedule

Notes

85%

Figure 12 Advisor Portal UI

Select Advisees

This screen allows advisors to select which students appear as their advisees and are displayed on the Admin Portal screen. This screen only features the Advisee Selection feature.

Select Advisees

Select Advisees

Use this panel to select which students you would like to see on your landing page.

Current Advisees

Show 10 ▾ entries

Search:

Name	Year	Graduation	Remove
James Childs	Sophomore	Spring 2021	<button>Remove</button>
Tim Tester	Sophomore	Spring 2021	<button>Remove</button>

Showing 1 to 2 of 2 entries

Previous

1

Next

Add Advisees

Show 10 ▾ entries

Search:

Name	Year	Graduation	Add
Bryan Wimbish	Sophomore	Spring 2021	<button>Add</button>
Frank Ocean	Freshman	Spring 2021	<button>Add</button>
Lorraine Alonso	Freshman	Spring 2021	<button>Add</button>
Michelle Smith	Freshman	Spring 2021	<button>Add</button>

Showing 1 to 4 of 4 entries

Previous

1

Next

Figure 13 Select Advisees UI

5.1.4 - Admins

Admin Portal

Admin users can find all of their user group specific features on the Admin Portal.

Included features:

- Master Schedule Import
- Change User Group
- Delete User

Admin Portal

Master Schedule Import

Browse ...

Upload

User Management

User Management

Show 10 entries

Search:

Name	Email	User Group	Change	Delete User
Alan Advisor	Alan@Advisor.com	ADVISOR	<-UserGroup->	✕
Amy Advisor	amy@advisor.com	ADVISOR	<-UserGroup->	✕
Bryan Wimbish	wimbisb@clarkson.edu	USER	<-UserGroup->	✕
Frank Ocean	frank@ocean.com	USER	<-UserGroup->	✕
Jackson DeMeyers	demeyejg@clarkson.edu	ADMIN	<-UserGroup->	✕
James Childs	childs@clarkson.edu	USER	<-UserGroup->	✕
Joe Morteolio	mortefjm@clarkson.edu	ADMIN	<-UserGroup->	✕
Ling Yao	ling@clarkson.edu	ADMIN	<-UserGroup->	✕
Lorraine Alonso	alonso@clarkson.edu	USER	<-UserGroup->	✕
Michelle Smith	smithm@clarkson.edu	USER	<-UserGroup->	✕

Showing 1 to 10 of 11 entries

Previous12Next

Figure 14 Admin Portal UI

5.2 - Hardware Interfaces

The Majorizor system neither relies on nor accommodates hardware interfaces of any kind.

5.3 - Software Interfaces

The Majorizor application is built on ASP.NET and Bootstrap, and thus will run on any system whose browser supports the Bootstrap framework. This allows Majorizor to run on any device, regardless of operating system.

Since Majorizor uses Bootstrap, it will also work on mobile and touch devices, although that is not an intended or high priority feature of the application.

Below are all frameworks, packages, and plugins used in Majorizor:

- .NET Framework 4.0
- Bootstrap v3.3.7

- jQuery v3.1.1
- MySql.Data v6.9.9 - Fully managed ADO.NET driver for MySQL
- DataTables v.1.10.13 - Searchable, sortable datatables plugin
- bootstrap-fileinput v4.3.9 - Bootstrap themed file input plugin

5.4 Communication Interfaces

Since Majorizor is a ASP.NET web application which uses a database backend, it depends on clients correctly working with the servers.

- Database Connection - The application must be connected to the MySQL database to allow users to login and use the application. Without a connection to the database, most features of the application will not work. Thus, our application relies on the TCP/IP protocols in order to maintain a connection to the database.
- HTTP Connection - The application is hosted on a server which must be able to interact with many clients at the same time. These clients will use the HTTP (which also uses TCP/IP) protocol to create connections to the host. Majorizor will require that the clients have an internet connection web browser in order to connect to the host and view the website.

This application does not handle any private or sensitive data, except for user passwords, which are hashed and salted, and never stored directly as plain text in the system. Therefore, Majorizor does not require encrypted connections to the database or website.

6 - Other Nonfunctional Requirements

6.1 - Performance Requirements

Majorizor is a web application and as such is not resource intensive. All modern devices with web browsers (laptops, desktops, tablets, phones, etc) should be able to easily run the application in the browser. The application's response time should be real-time with little delay. The only limiting element to the response time of the application is the connections between client and host, and between the host and the MySQL database. Given these connections are fast, response time should be nearly instant.

6.2 - Safety Requirements

There are no safety concerns associated with the use or effects of the Majorizor system. Additionally, Majorizor is a standalone system that draws data from PeopleSoft and the Clarkson University website. Because the system does not generate or host university-wide information, there is no danger of general data loss should the system go down.

The primary data loss concern is with the loss of student user data. To ensure

6.3 - Security Requirements

The Majorizor system stores users' passwords. As such, a high level of security is needed to protect such user credentials. All passwords are salted and hashed, and then the hashed password, along with the salt are stored in the user_password table in the MySQL database. Each salt is a randomly generated salt of length 10. When a user attempts to log into the Majorizor system, the system will search the database for the salt matching that user's email address. The Majorizor system will then attempt to hash the input password with the salt from the database. If the newly hashed password matches the password stored in the database, the user is logged in. This keeps user passwords confidential, and makes it difficult for anyone to access user accounts should the Majorizor system's database be compromised.

6.4 - Software Quality Attributes

This application aims to provide a consistent, visually pleasing UI throughout the application, along with simple and straightforward page design thanks to Bootstrap. The goal of the interface is to make services easily accessible and understandable to all users. Users should not find it difficult to understand the structure and flow of the application on their first visit.

7 - Key Milestones

#	Milestone	Target Completion Date	Comments
1.	User Group Setup	2/24/17	Complete
2.	Software Requirements Document	3/3/17	Complete
3.	Master Schedule Import	3/10/17	Complete
2.	Undergraduate Schedule Generation	3/24/17	
3.	Academic Major Selection	3/24/17	Complete
4.	Academic Minor Selection	3/24/17	Complete
5.	Software Design Document	3/31/17	Complete
6.	Curriculum Progress Tracking	3/31/17	
7.	Advisor Overview of Students' Schedules and Progress	4/7/17	
8.	Required Course List Export	4/7/17	
9.	Generated Schedule Export	4/7/17	
10.	Final Product Deadline	4/28/17	
11.	Software Testing Document	5/3/17	
12.	Software Maintenance Document	5/3/17	

8 - Other Requirements

The application has potential to be used in a production setting at Clarkson University. Since our team will not be able to support the application in the long term, Majorizer's backend code should be well documented and commented to allow for other teams to support the application in the future. In addition to this document and the Design Document, the comments within the code should help with the transition of handing this project off to other teams.

9 - Requirement Change Management

If a member of the Majorizor development team wishes to change a requirement, they will create a new version of the requirements document and change the relevant document sections. The team member will then log the changes made to the requirements in the Revision History section of the new document. Once these steps have been completed, the team member must notify the rest of the team via email or during a team meeting.

If the customer wishes to make a change to the requirements, they must inform the entire Majorizor development team with a formal request sent via email. The development team will discuss the request at the next team meeting, and generate a list of clarifying questions if necessary. A member of the development team will then respond to the client, acknowledging the request and including the team-generated questions. Once the client has answered the additional questions, the team will create a new version of the requirements document and change the relevant sections as per the demands of the client. The changes will be logged in the Revision History section of the new requirements document. After this process is complete, the changes can be implemented in the Majorizor system.

Appendix

Appendix A: Glossary

- **ASP.NET** - An open source web framework for building modern web apps and services with .NET
- **ECE** - Electrical & Computer Engineering Department at Clarkson University. Includes Computer, Electrical, and Software Engineering.
- **Bootstrap** - A popular open-source HTML, CSS, and JS framework for developing responsive web applications.
- **HTTP** - (Hypertext Transfer Protocol) An application protocol for distributed, collaborative, and hypermedia information systems. HTTP servers as the foundation for the World Wide Web.
- **Peoplesoft** - The main system used by Clarkson University to manage students, majors, minors, schedules, etc.
- **TCP/IP** - (Transmission Control Protocol/Internet Protocol) A protocol which serves as the basic communication protocol for all communication over the internet.