Software Maintenance Document

For

Majorizor

Version 1.0 Approved

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

May 3, 2017

Table of Contents

1 - Introduction

This document exists to help future developers of Majorizor understand where they should begin improving the existing features of the application. The current version of Majoriozor, “v0.1”, did not achieve all of our teams initial goals, however, does provide an organized class structure, clean and minimal UI, and all features currently implemented, with the exception of progress tracking, are fully functional.

The first features of Majorizor which should be improved are listed in the Required Features section. These features include Schedule Creation, Progress Tracking, and the Student and Advisor Portals. Majorizor will be an extremely powerful application if these features were able to guide students in creating a semester to fill their major/minor requirements, and if the requirements were presented in an easy way for students to track their way through all of their degrees. Thus, we believe these features are the highest priority and should be accomplished first.

Along with the Required Features, and any other improvements the next team may have, we also believe the features listed in the Recommended Improvements should be implemented to help Majorizor become more user friendly, add usability, and become more secure.

2 - Required Features

2.1 - Schedule Creation

A basic schedule creation function has been implemented along with a user interface to support it. However, this implementation was done in a version of the application that did not support the most recent changes to the Login and Major/Minor Selection functions. Therefore, the code for schedule creation must be manually added to the current version of the project.

There are several improvements to be made to the Build Schedule function once it is integrated. For one, the function does not currently handle co-requisites. It also cannot currently accommodate the academic requirement for knowledge areas. Once the schedule creation function is integrated, this complexity will have to be built in.

2.2 - Progress Tracking

Although there is a working version of progress tracking implemented in the current version of Majorizor (v0.1), it is very basic. In the current version, the only aspect of progress tracking visible to the users through the UI is a percentage of completion calculated as (total # of courses taken) / (total # of courses required for a major) * 100. The current version of the progress tracker, as implemented in the `ProgressTracker` and `ProgressInformation` classes also tracks a list of `Course` objects for both a student’s taken and required courses for all their majors and minors combined. This feature is not currently used in the UI.

Ideally, the progress tracker should have more functionality, as a strong progress tracking system for advisors is a key requirement for the system. The progress tracker should both track more information in the backend and display more information to the user. In the backend, the progress tracker should keep a list of courses for required classes and taken classes per student's majors and minors.

For example, if a student has a Software Major (SE) with a Math Minor (MA), then the progress tracker should keep 4 lists, two required list; one for the SE major and one for the MA minor, and two taken lists; one for the SE major and one for MA minor.

Then, we could track percentage of each major taken, and display a list of courses the student still needs to take, sortable by major. The percentages could be displayed with Bootstrap progress bars, and the required courses could be displayed in a datatable, using the dataTables.Bootstrap.min.js plugin (used on Admin Portal and Advisee Selection).

2.2.1- Potential Implementation Strategy

The ProgressTracker class should be modified to something similar to the following.

```
class ProgressTracker
{
    private Student { get ; set; }

    // Student.major#.reqCourses EXCEPT Student.takenCourses)
    public List<Course> reqMajor1 { get; private set; }
    public List<Course> reqMajor2 { get; private set; }

    // Student.minor#.reqCourses EXCEPT Student.takenCourses)
    public List<Course> reqMinor { get; private set; }
    public List<Course> reqMinor2 { get; private set; }

    // Student.major#.reqCourses "inner join" with Student.takenCourses
    public List<Course> takenMajor1 { get; private set; }
    public List<Course> takenMajor2 { get; private set; }

    // Student.minor#.reqCourses "inner join" with Student.takenCourses
    public List<Course> takenMinor1 { get; private set; }
    public List<Course> takenMinor2 { get; private set; }

    // Percentages
    // Calculate using credits not # of courses.
    //
    // Credits will need to be added into the database and Course class.
    // The column exists in the `course` table, but is all null.
    public int major1Percent { get; private set; }
    public int major2Percent { get; private set; }
    public int minor1Percent { get; private set; }
    public int minor2Percent { get; private set; }
    public int overallCompletion { get; private set; }


    // Constructor
    public ProgressTracker(int userID)
    {
        student=new Student(userID);
    }

    public CalculateProgress()
```

```
    {
        // fill in all List<Course> members with appropriate data

        // calculate percentages
    }
}
```

This gives the ability to retrieve required and taken courses by major and minor, get the percent completed per major and minor, and get the overall completion. Both the Student Portal and Admin Portal should be updated to allow users to take advantage of all of this information. The Student Portal should show all progress information in the provided panel. The Advisor Panel should display this information in a pop-up modal when activated by the View Progress button in a student's panel.

2.3 - Student Portal

The Student Portal is currently only partially implemented, due to the progress tracking and schedule building features not being completely finished. Once these features are finished, the student portal should be updated to allow the student to access all features of Progress Tracking, Schedule Creation, and Major and Minor Selection.

2.4 - Advisor Portal

The advisor overview section of Majorizor currently is only partially implemented. This section should allow advisors to use the Major and Minor Selection, Schedule Creation, and Progress Tracking features for each of their students on the Advisor Portal. Each feature should be accessible through pressing the corresponding button in a student's panel, and should bring up a pop-up modal with a complete UI for each feature. Advisors should be able to fully manage each student, just as the student can manage themselves.

3 - Recommended Improvements

These improvements are improvements that were not in the requirements of the application, but would enhance the usability and quality of life of Majorizor.

3.1 - Account Creation

Possible account creation improvements includes:

- Add password strength requirements
  Require passwords to be specific length and have a certain number of special characters. This will improve account security and minimize the risk of accounts being compromised.
- Require users to use Clarkson University Emails

Make sure all registered users are members of Clarkson University. Majorizor is specific to this University, so no other users should be allowed access.

Both of these potential improvements could be handled by using either Clarkson's Active Directory or Google's sign-in API to limit users to already registered Clarkson University accounts. This could eventually lead to tighter integration to other Clarkson Services.

3.2 - Master Schedule Import

Possible master schedule import improvements include:

- Provide feedback to the administrator if import is successful
  If the master schedule is imported successfully, show a green alert to the user that the import was successful. The `ExceptionHandler` class would need to be modified to allow for green alerts to be created.
- Improve exception error messages for Popup errors
  Add specific error messages for when schedule importing fails. For example, if it fails during file parsing, create an error that states something along the lines of:
  "The file could not be processed. Please make sure you are uploading a "|" (pipe) delimited .csv file created from Peoplesoft's master schedule export."

3.3 - Delete User

Possible delete user improvements:

- Ask for confirmation
  The application should ask the Admin if they really want to remove the user, instead of allowing the Admin to delete an account permanently in one click. This will help avoid accidentally losing user data of a current user.

3.4 - Select Advisees

Possible improvements for advisee selection:

- Hide other advisors' advisees
  Do not show students in the Add Advisees table if they currently are assigned an advisor. This will prevent advisors from adding another advisors student, and make sure students can only have one advisor. This currently only displays an error to the user.

3.5 - Major and Minor Selection

Possible improvements for major/minor selection:

- Require at least 1 major
  When a student is initially selecting majors and minors, they are not allowed to proceed until they have selected at least one major. However, after initial setup,

students can drop all majors. This should be modified so students cannot drop below one major.
If the student only has one major, hide the remove major button.

- Prevent the user from selecting incompatible majors and minors
When a student selects a major or minor, the system currently does not check to see if they are compatible with any of the users other selected majors and minors. This allows the user to select two of the same major, a major and minor in the same field, and other combinations that are not permitted such as a Computer Engineering major with a Software Engineering minor. In the future, the system should prevent the user from making such combinations.