Student Registration System

Phase II: Object-Oriented Software Requirements Specification

By Team 9

Project Manager Carson Schutter

Team A Nicholas Malonee

Eric O’Sullivan

Marcus Fields

Team B Robert Jarmin

Mi Gao

CPS 410

Spring 2017

Central Michigan University

Contents

**Chapter 1**  System Overview 3

**Chapter 2** User Interface 4

**Chapter 3** Functional and Non-Functional Descriptions

3.1 Functional 6

3.2 Non Functional 6

**Chapter 4** Performance Description 7

**Chapter 5** Exception Handling 8

**Chapter 6** Acceptance Criteria 9

**References**

**Appendices**

A: Resources

* User Interview
* Relevant Articles

B: Functional Models

C: Object Models

D: Dynamic Models

E: Gantt Charts for Phase II

**Chapter One: System Overview**

Central Michigan University offers the ability to register for classes online with registration software. Though the program works, there are some aspects that can and should be improved upon. Upon logging into the online system, students are required to choose the academic year for which they want to register. This process is a tedious since given when a student logs in they will have only two real options, that is to drop current classes or sign up for upcoming classes. The action of selecting the semester to make changes is an unnecessary step for students to make.

At certain points in the semester students can register for the next semester of classes. There are also specific points of time in the semester when students can still drop from a class while being eligible for a refund. After a time in the semester if students drop their class they may still be charged for the course and depending on the specific time in the semester, they may or may not be eligible for and Incomplete as their grade versus a failing grade that could affect aide the student is receiving.

Students often may not be sure when registration starts for them or they may have confusion when a class can safely be dropped. Creating an intuitive function to the existing registration software that would assist students in automatically placing them in the most likely semester to make changes would help with some confusion and make registration simpler. With intuitive software assistance and a simper User Interface (UI) students will have an easier time with course registration.

**Chapter Two: User Interface**

Having a simple easy to navigate user interface (UI) would allow students to make changes to their schedule with ease. While only making small user interface changes so allow for easier navigation through course registration and withdrawal from a course. Rather than a drop-down box that would allow students to view previous semesters that cannot be changed buttons would be more useful. Changing the ability to use buttons based on what students can do at any point in the semester would prevent time wasted trying to decide what can and cannot be done. Creating more buttons to the system, or even the removal of drop-down boxes would eliminate undo complexity in the system and for users. Keeping a simple straight forward UI design is the plan with this project.

For the UI in our project we plan to keep most things the same as what is currently in the system with a few exceptions. Rather than having a drop-down box for selecting the desired semester, there will instead be buttons that direct the user to the current semester or the next semester. For instance, if a student is within the first four weeks of class in the spring of 2017, the options for buttons available would be: Spring 2017; Summer 1 2017. With Spring 2017 being the default semester. Progressing through the semester to week 12 options would change adding new buttons: Spring 2017; Summer 1 2017; Summer 2 2017; Fall 2017. With Fall 2017 being the default semester. The program would remove Spring 2017 from registration options once that semester ends, to create efficiency and ease of use.

**Chapter Three: Functional and Nonfunctional Descriptions**

3.1 Functional Requirements

The Registration System allows students to register and drop classes from their current or future schedule. When students log into the system they will be able to:

1. Select semester for registration
2. Register or Drop from classes from selected semester

3.2 Non-Functional Requirements

The Non-Functional Requirements are as follows:

1. Availability. Our project when completed should be available to all Central Michigan University students through Central Michigan’s website.
2. Security. The registration system will allow for a secure login and protect student information.
3. Functionality. This program will function properly without crashing. When students register for a class, it will be reflected on their Central Michigan student account and they will be billed appropriately.

**Chapter Four: Performance Description**

The programs performance will be measured in three aspects. The first being usability. Usability being measured by how easy a student can access the program, look for classes and subsequently register for classes. The second aspect of performance will be clarity. Clarity will be measured by how clear the program’s functions are to users. Making sure that students understand how to drop, add, and search for classes is important. The final metric to determine performance is efficiency. Having a program that loads promptly and guides users through its functions without crashing or unacceptable wait times.

To measure our programs performance, we will look at these three aspects. Usability will be determined by having access to students and seeing if they can log in, and if they can register for classes. Clarity will be measured by how many questions a user has during the registration process within our program. Finally efficiency will be measured by the time it takes to load the program and register for classes.

**Chapter Five: Exception Handling**

During runtime of a program there can be certain conditions require special processing. This is referred to as exception handling. There may be cases when students try to register for a class without the proper requirements being met, making the program susceptible to crashing. Some common concerns within our program are as follows:

1. Students my not have completed prerequisites for a class
2. Students my try to register before or after the registration period.
3. Students have holds on their account or may not be a current student.

To circumvent these exceptions we will create functions to check for specific criteria. If a student does not have the required prerequisite courses, there will be an error message, “The required prerequisite courses must be taken before you can register for this class, if you feel you have received this error by mistake try to register again. If the problem persists contact your class adviser.” Similar error messages will be shown outlining why students cannot register and how to fix the issue on their side.

**Chapter Six: Acceptable Criteria**

Our Acceptance Criteria for this project is the minimum viable product as defined by our Functional and Non-Functional description. Essential having a program the compiles and runs allowing the user to register and drop classes as they desire. The program should not crash and should get stuck in any infinite loops. The program should also terminate properly and handle all exceptions as they come.