Software Requirements Specification

for

Dynamic Course Planner

Version 1.0

Prepared by

Group Name: Dynamic Course Planner

|  |  |  |
| --- | --- | --- |
| Daniel Castro |  | DanielCastro@letu.edu |
| Baughn Welch |  | BaughnWelch@letu.edu |
| Nathan Kleoppel |  | NathanKleoppel@letu.edu |

|  |  |
| --- | --- |
| Instructor: | Dr. Baas |
| Course: | Software Engineering I |
| Date: | February 6, 2017 |
|  |  |

Contents

Revisions iii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 2

2 Overall Description 3

2.1 Product Perspective 3

2.2 Product Functionality 3

2.3 Users and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 3

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3 Specific Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 6

3.3 Behaviour Requirements 7

4 Other Non-functional Requirements 9

4.1 Performance Requirements 9

4.2 Safety and Security Requirements 9

4.3 Software Quality Attributes 9

5 Other Requirements 11

Appendix A – Data Dictionary 12

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| 1.0 | Castro  Welch  Kleoppel | First version of the requirements written during team meeting. | 02/03/2017 |

# 

# Introduction

*The dynamic course planner is an academic career planner that allows a student to dynamically plan all four or more years of their classes in one place. It allows a student to select their major and then drag and drop related classes into each semester. Students will be able to manually add extra courses to the planner as electives and they will be able to account for classes taken over the summer or transferred from other institutions.*

## Document Purpose

This document outlines the requirements specifications for the Dynamic Course Planner software engineering project. Its purpose is to give the design and implementation teams a guide towards a completed product. With the help of these requirements, some gray areas have been addressed and will provide a clearer understanding of the end product.

## Product Scope

The purpose of the Dynamic Course Planner software product is to supplement and guide students in their semesterly course planning. The product will help to encourage interactive planning between the student and the course options. With that, the goal of the software is to diagram the semesters of classes each student will take in their program from the congested, unorganized, hard to follow course catalog in place now.

## Intended Audience and Document Overview

This particular requirements document shall be informative to the designers, developers, project creator, and manager (Dr. Baas) as to further details and guidance of the Dynamic Course Planner from the baby stages during prototype development. The “Overall Description” section will be very beneficial to the design team in their work to cover the basis of the software end goals from the start. The “Specific Requirements” will be dedicated to the developers and testers to foster a complete well rounded software meeting all of the necessary requirements contained within that section of this requirements document.

## Definitions, Acronyms and Abbreviations

**DCP:** Dynamic Course Planner

**GPA:** Grade point average

**IEEE:** Institute of Electrical and Electronics Engineers

**Project Creator:** Corey Naas

**Project Manager:** Dr. Baas

**Student:** Primary end user/client

## Document Conventions

All formatting for this SRS is based around the baseline IEEE format standard. No extra format or typographical conventions have been altered or added from the standard. Numeric subsections separate individual pieces from the large primary sections (1-5). Please refer to the IEEE standard documentation online for further details if needed.

## References and Acknowledgments

Thus far, this requirements document does not reference any alternative documentation. All references have been discussed with either the requirements group or project creator. Although some clarification on structure has been provided by:

I. Sommerville, Software Engineering (9th edition), 9th ed. Boston: Addison-Wesley Educational Publishers, 2010.

# Overall Description

## Product Perspective

This particular product, the DCP, is an updated and more user guided tool for collegiate students to plan their semesters to complete a sought degree. It will be founded around the course catalog of the specific institution it is applied to. For instance, here at LeTourneau, the most current course catalog from the school is simply a list of required courses needed to attain eligibility to graduate. However, there are no user interface friendly services available for students to layout a semester by semester sequence. This particular gap is what this product seeks to fill.

**2.1.1 – Environment Interaction Diagram**

Dynamic Course Planner

Degree List

Course List

University Course Catalog

## Product Functionality

* Allow users to drag and drop courses into their schedule
* Allow users to add courses manually
* Allow users to add more semesters
* Allow users to save degree profiles
* Allow users to create degree profiles

## Users and Characteristics

* Students – considered to know basic operation for normal applications. (Prominent User)
* Professors – considered to know basic operation for normal applications.
* Administration – expected to be familiar with the interface for any tech support questions.

## Operating Environment

This application is based on C# and Microsoft Visual Studio and hence will require the operating environment that can run applications that are developed from them. The operation system will be Windows 7 and up.

## Design and Implementation Constraints

The application is to be built in Visual Studio. The catalog for the different class will have many variables and thus must be considered when selecting the way to store the information, i.e. a database. An ArrayList of the course library has been considered. The application must respond in a reasonable time frame. Must be useable. End user will maintain and simply add extra courses and functionality as they desire. A future possibility is a file downloaded from the university incorporating the most updated catalog of courses and degree offerings.

## User Documentation

Ideally, a simple user guide with screenshots would suffice to supplement the product. A detailed description of the primary interactions between the student and product are critical. Primarily how each text field should be filled to maintain consistency with the display of the product overall. Additionally, it would be beneficial to include an extensive introduction about the basis of the product being predominantly user entry based. Making it clear the product is mostly to provide the tools to guide the semesterly planning.

## Assumptions and Dependencies

* Student knows what meets the requirements of a specified elective credit.
* The student is taking university approved courses and enrolled in a degree that is university sanctioned.
* Student knows what a reasonable course load is for a single semester.
* Student is knowledgeable about basic user interfaces in general without trouble dragging classes to semesters.

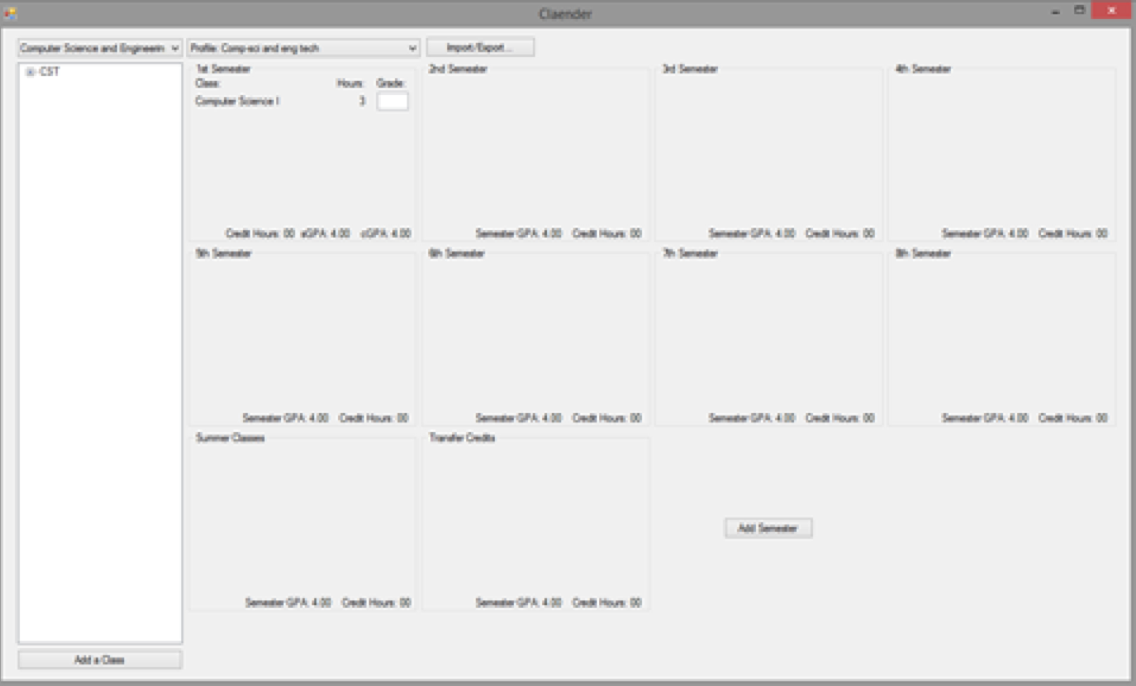
# Specific Requirements

## External Interface Requirements

### User Interfaces

At first glance the user should see a window with two combo boxes side by side starting from the top-left of the main window and with an import/export button to the right of the combo boxes. Directly below the combo boxes should be a tree viewer where the user can view his/her classes. There will be a table to the right of the tree viewer displaying semesters. The first cell that is empty should contain a button, labeled “add semester”. Now at the bottom of the tree viewer there should be a button, labeled “add classes”, see Figure 1.

The first drop down box should have all available majors the users university provides, and the next drop down box should list profiles the user has created. The Import/Export should allow the user to import or export profiles. The add semester button add a semester cell to the table. The tree viewer should display classes available.

There should be a second window when the user clicks the “add class” button. In this window, there should be three text boxes for class name, credits hours, and prerequisites. Then two checks boxes for fall and spring only classes and finally an add class button at the bottom, see Figure 2.

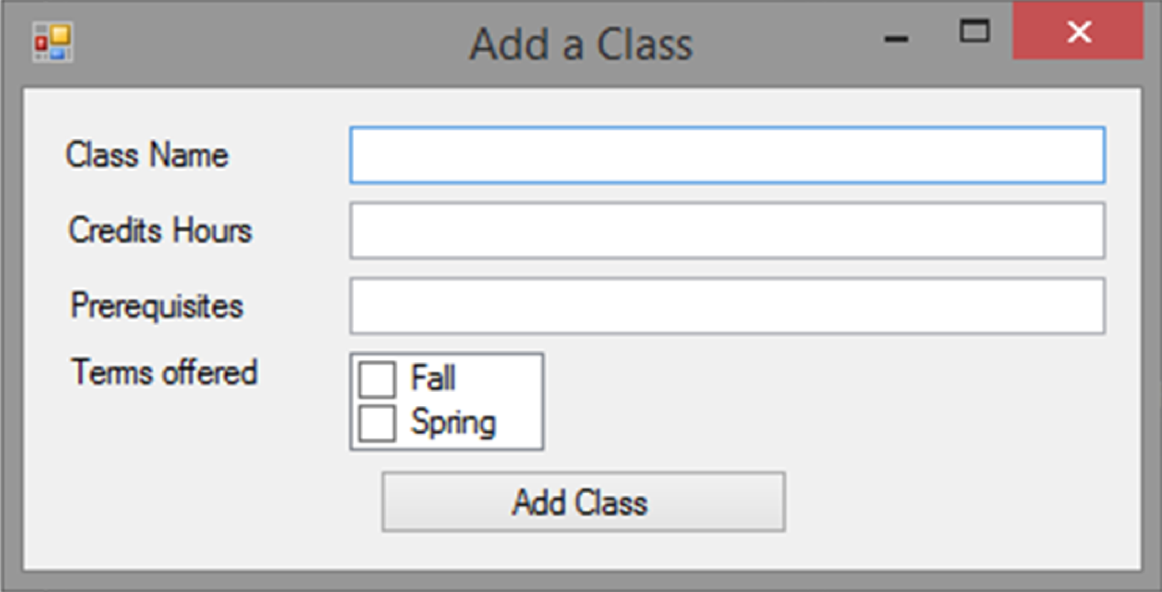
Figure 1 - Main window

Figure 2 – Add Class window

### Hardware Interfaces

The only hardware interfaces would be the operating system, in this case Windows 10. Since, the stand-alone application does not have any designated hardware, it does not have any other direct hardware interfaces.

### Software Interfaces

The product shall make use of the operating system calls to the file management system to store and retrieve data from files containing profiles that the user has created.

### Communications Interfaces

The product will not have any communication interfaces or deal with any network servers. However for future versions, there could be a downloadable file for available classes that a university provides. The file would probably be available from the university’s information technology website.

## Functional Requirements

***3.2.1 The User - Student***

***3.2.1.1 Functional requirement 1.1 - Degree Selection***

*A user should be able to select an available major, that his/her university offers, from the drop down menu. When a selection has been made, the tree viewer should display class types associated with the chosen major. For example, if the user selects the major, Computer Science and Engineering, then the class types would be COSC, EETC, BIBL, etc.*

***3.2.1.2 Functional requirement 1.2 - Importing Profile***

*The user shall be able to load a profile by clicking the Import/Export button. When the user loads a profile, the profiles name should be displayed in the profile drop down menu. Any other profile names previously in the drop down menu should shift down. The product should read the file and display the correct major in the drop down menu, also display all the classes in the tree viewer associated with the major or mentioned in the profile file. Finally, the product should display semesters and the classes and the grade associated with each semester.*

***3.2.1.3 Functional requirement 1.3 - Exporting Profile***

*The user shall be able to export a profile by clicking the Import/Export button. When the user exports a profile, the profile file will be created in the same directory as the product. The file should be saved as a text file.*

***3.2.1.4 Functional requirement 1.4 - Profile Selection***

*The user shall be able to select a profile by going to the profile drop down menu and selecting the desired profile name. When a profile has been selected the product should display the following: the correct major, associated with the profile, in the drop down menu, the correct classes in the tree viewer, the correct number of semester in the table, and the classes and class grade in correct semester cell.*

***3.2.1.5 Functional requirement 1.5 - Drag and Drop Classes***

*The user should be able to drag and drop classes from the tree viewer to a semester cell in the table. When a class has been dragged on top a semester cell, The application should be able to display the class name, the hours, and a text box for the grade. If this is the first class dragged to the semester cell then a label called “Class:” should be displayed above the class name. Also, display the total credit hours at the bottom of the semester cell. If the user drags a class to a cell, named nth semester, and if the user has not drag that class prerequisites to a cell, named (n-1)th semester, then the class label shall be colored in red. If the user drags a spring only class to a fall semester, then that class label shall be colored in purple.*

***3.2.1.6 Functional requirement 1.6 - GPA Calculation***

*When a user inserts a letter grade in the grade text box, then the product should automatically calculate the career grade point average and the semester grade point average. The calculation should be displayed at the bottom of the semester cell.*

***3.2.1.7 Functional requirement 1.7 - Add Classes***

*When a user clicks the “add a class” button a window will pop up. In this window, the user can specify the class’s name, hours, prerequisites, and terms offered. When the user clicks the add class button, the window will disappear and the name of the class should be displayed in the tree viewer. If the class type does not already have a branch, then a branch will be add where the class name will go under.*

***3.2.1.8 Functional requirement 1.8 - Add Semester***

*When the user clicks add semester button then a new cell will be added to the table. The semester should be called nth semester, where n is the previous cell number plus one. There should also be semester GPA and credit hours displayed at the bottom of the cell.*

## Behaviour Requirements

### Use Case View

The Operator, or student, shall be able to perform two main system functions: add classes, import and export profiles. Add Classes use case is the operation that allows the user to define a class and add the class to a semester. Add Semester use case is the operation that allows the user to create a new semester cell in the table. Import and Export Profiles use case is an operation that allows the user to load or open profiles.

### Class Diagram View

Figure 3 – Class diagram showing variables, methods, and associations between them.

# Other Non-functional Requirements

## Performance Requirements

All performance requirements are in addition to all user interface actions being performed in a timely manner by the product.

**4.1.1 – Performance Requirements and Considerations**

* GPAs shall be displayed in real time based on user entries.
* Dialog boxes shall open immediately upon button click.
* User entry should be entered immediately upon confirmation button click.
* No parallel processes known thus far.
* Any edits to the ArrayList or database of course library and course selections should be realtime updated as well.

## Safety and Security Requirements

Perhaps an explanation dialog box would be useful to give notice to the end client the minimum security implemented into the software. Leaving the liability to the user to protect personal information as desired. Thus, a very minimum security level will be applied to the product.

**4.2.1 - Safety/Security Requirements and Considerations**

* The product shall protect the privacy of student grades with necessary consideration.
* Student information and entries shall not be transferred if program is shared between user systems.
* A confirmation box should be utilized in the case of grade alterations once a grade has already been entered.

## Software Quality Attributes

**4.3.1 – Reliability**

The product shall be free of bugs. Thus, extensive testing by current students both associated and not associated with the product development teams shall be conducted to hash out and bugs and deeply cover all major parameters that will be attempted.

**4.3.2 – Consistency**

The DCP shall maintain consistency throughout all areas of the product. This includes but not limited to title formatting, text field labeling, sizing in general, course lables, and many others. As mentioned in section 2.6, the user shall be sufficiently notified of the parameters being set to labeling and different areas to ensure consistency with the product even after user entry.

**4.3.3 – Adaptability**

Adaptability to both updated course/degree information and other institutions would be a highly sought after attribute. This could be achieved through using basic data structures such as our ArrayList idea for example that could easily be adaptable to different applications at other universities in need of a course planner for studnets.

**4.4.4 – Usability**

Finally, the Dynamic Course Planner needs to be usable. Although many students in the modern day know computer interfaces past a novice level for the most part, it should still be straightforward in labeling and design to permit beginner computer users to use the product. On top of that, it should reduce complexity as much as possible by keeping course information simple and basic and maintaining a spaced out, uncluttered main screen to keep things organized and usable.

# Other Requirements

One of the reuse objectives for the DCP is for future developers to have the ability to reuse the software for application at new institutions or for updated program and course information at current institutions.

Appendix A – Data Dictionary

|  |  |
| --- | --- |
| *Variables* | *ArrayList (CourseLibrary)*  *ArrayList (CurrentSemesters)*  *interfaceButtons*  *treeViewer*  *profileName*  *Refer to section 3.3.2 (Class Diagram) for all others besides primary.* |
| *States* | *MainScreen*  *AddClass*  *AddSemester* |
| *Functional Requirements* | *Degree Selection*  *Import Profile*  *Export Profile*  *Profile Selection*  *Drag-and-Drop Classes*  *Calculate GPA*  *AddClasses*  *AddSemester* |