Executive Summary

At the University of Victoria (UVic), course registration is a difficult and stressful process for students. Although registration is somewhat straightforward for those individuals that manage to follow their curriculum throughout their degree, it can become an enormous headache for students who get behind in their courses due to illness, poor grades, or other reasons. The web applications provided by the University and third parties for registration have a number of problems that Puzzle would like to address by providing a new website that serves as a complete solution for student's degree planning needs.
There is a website developed by a past student at UVic called ScheduleCourses.com which serves as a significant improvement to the University provided site. It allows students to select a list of courses, and then shift the course times selected around in order to make an ideal schedule. It provides a dynamic visualisation of your timetable which is very helpful for mitigating conflicting course times.

The most significant issue with ScheduleCourses.com is that there is no simple way to view how your selection of courses in a particular semester will affect your final goal of completing your degree program. Often, students will fail to register in a prerequisite during a particular term. This will cause them to be unable to take required courses in the future, putting them further behind in their degree. Similarly, courses are often offered in specific terms, and an unaware student may need to wait a year to take a particular course. Our system will attempt to take into account a user's program requirements and provide a means of visualizing how their selected registration will affect their final graduation date, and future schedule requirements.
This is a fairly large project which will require a lot of data entry regarding various program requirements at UVic. As such, Puzzle will develop a proof of concept that shows how our course scheduling system will work with the Software Engineering program at UVic. If this concept is satisfactory, it will then be possible to move forward with the other programs offered at the school.

S1 Functional Specification

Graduate!

University Degree Planner

Created by: Puzzle

Authors:

Brendan Heal
Jonah Rankin
Ali Nobari
Spencer Mandrusiak

Table of Contents

1.0 Project Summary

The Graduate! web application will help students at the University of Victoria plan their degree by providing students with tools to create custom tailored schedules, which accommodates each individual's specific needs. Many University programs have highly specific degree requirements and the University of Victoria is no different. Just Right Showers has expressed interest in commissioning a system that will assist UVic students by planning their entire degree. Puzzle's Graduate! system will help users by allowing them to customize certain aspects of the schedule, such as having no evening classes or only four classes in a semester, to accommodate for the busy lifestyle of a university student. This will give students more freedom when planning out their degree and also help them keep track of what they have or have not completed. Another way Graduate! benefits students is it will automatically regenerate a new schedule if a student marks that they have failed a course; therefore the student will not have to deal with the burden of rearranging their schedule if such an unfortunate event occurs.

1.1 Important Features

Graduate! has many important features such as:

- Generate a schedule for a specific degree/program
- Allow users to customize aspects of the schedule such as number of courses per term, time preferences, and specify any work terms to avoid classes being scheduled during that time
- Make automatic adjustments to the schedule if a user makes alterations, changes their preferences, or fails a course
- Allow a user to manually enter desired courses such as when students have not declared their program
- Adheres to University restrictions on courses such as term offerings, pre and co-requisites, maximum credits per term, etc.
- Allow user to view, update and save the schedule to their profile, as well as make modifications at a later date

1.2 Hardware

Our software will run on all major operating systems including: Mac OS X, Linux, and Windows. Graduate! will be designed to run on any modern HTML5 compliant browser including: Chrome, Firefox, Safari, and Internet Explorer.

1.3 Performance

Graduate! is expected to be available at least 95% of the time in a year (approximately 18 days of down time) and leave users without access to their schedules for no more than 48 hours at a time. While Graduate! is not safety critical software, it is important that scheduling services be available for users and that course registration will not be compromised because of a failure of the Graduate! system.

The course scheduling algorithm is expected to complete in less than 5 seconds, and during processing it must provide the user with feedback.

At Puzzle we value our users privacy and the final product will utilize industry standard encryption and security practices to protect user information.

2.0 User Interaction

2.1 Application Views

2.1.1 Front Page

The Front Page is the landing page for the application with options to register, login, and get more information about Graduate!.

2.1.2 Login Page

The *Login Page* contains a form for users to enter their Email and Password.
The **Cancel** button returns the user to their previous page — most commonly the Front Page. The Login button proceeds with login validation. If the submitted credentials are valid

the user is taken to the*Program Management Page*. The *Login Page* may alternately appear as a modal window over-top the previous view.

2.1.3 Program Management Page

The *Program Management Page* is the home view for the application. From this view a logged-in user will be able to:

- Review their schedule including completed classes, the current semester and future semesters.
- Add, modify, and remove courses
- Define program preferences
- Auto-generate schedules based on their preferences.

Figure 1: Mock-up of Program Management View

2.2 Use Cases

2.2.1 Registration

When a user first accesses the web application they arrive at the *Front Page*; users have the option to *Login* or *Register*. When the registration button is pressed the registration process begins:

1. The user enters their account information including: Email, Password, and University. After entering their information, the user presses the **Next** button to proceed to the second registration step.
2. The user specifies their course and program information. First, the user selects their program. Choosing a program populates the required courses list. Optionally, the program can be unspecified and all classes must be added manually. Second, the user adds other courses, as well as specifying electives.[1] Third, the user marks all the courses in the list that they have completed. After all program information has been entered, the user finalizes account registration by pressing the **Finish Registration**[2,3]
3. After finishing the registration process, the user is redirected to the *Program Management Page*.[4]

[1] It is not required for users to specify all electives to register. Initially electives will be specified as generic classes such as *Technical Elective* or *Complementary Study*. In order to schedule an elective, it must be a specific course. This can be done after the registration process.

[2] Users will have the option to modify program details and add or remove courses after registration.

[3] The course and program specification process may be simplified through integration with a user's UVic account. This would allow for automatic retrieval program information and completed courses.
[4] Account confirmation emails are not a high priority for initial prototypes. They will be implemented if time permits. If account confirmation is implemented, users will be sent to the login page and notified that they must check their email and confirm their account.

2.2.2 Login

When arriving at the applications *Front Page*, pressing the **Login** button brings the user to the*Login Page*. On the *Login Page,* the user enters their *email* and *password*. They then press the**Login** button, which redirects them to the *Program Management Page*. Alternatively, on the *Login Page* the **Cancel** button returns the user to the *Front Page*.

Sample Interaction: Jane has just arrived at the *Front Page* of Graduate!. Since she already has an account, Jane selects the **Login** button, bringing her to the *Login Page*. Jane enters her email and password, followed by pressing the **Login** button. Once authorized, Jane is brought to the *Program Management Page*.

2.2.3 Mark Courses Complete

Once registration has been successfully completed, Graduate! will automatically mark courses as completed based on the user's transcript. Graduate! will update completed courses each time the user logs in to the application.

Sample Interaction: Jay has already started using Graduate! and has a generated schedule. Because Jay has just finished his term and passed his courses, one of which is "SENG 321", he navigates to the *Program Management Page*. When Jay arrives at the page, he selects **View Schedule** and notices that Graduate! has already marked "SENG 321" as complete.

2.2.4 Alternate – Mark Courses Complete

Upon registration, the user will be prompted to select all courses which they have already completed. Alternatively, the user may go to the *Program Management Page* and select **Add**

Completed Course, where they will search for the appropriate course. Once they have found the desired course, the user marks a checkbox indicating the course has been completed.

Sample Interaction: Jay has already started using Graduate! and has a generated schedule. Because Jay has just finished his term and passed his courses, one of which is “SENG 321”, he navigates to the *Program Management Page*. When Jay arrives at the page, he selects the **Add Completed Course** button, which displays a list of all courses relevant to Jay’s degree. Jay searches for “SENG 321” and selects the checkbox indicating he has completed the course.

2.2.5 Add Courses to Schedule

A registered user will be able to add courses to their schedule using the *Program Management Page*. On the *Program Management Page*, the user will select a particular term and a list of suggested courses for this term will appear.

Sample Interaction: Jay has recently finished registering for Graduate!. He navigates to the*Program Management Page* and selects **View Schedule**, followed by **Summer 2017** to view his recommended schedule for that semester.

2.2.6 Move Courses

On the *Program Management Page*, courses can be shifted between semesters by drag-and-drop. The user will be notified graphically if they are moving a course to an incorrect semester. This notification will appear for:

- Missing prerequisites
- Course not offered during the semester[1]

The schedule should take into account usual course offerings by term as many students want to completely plan out their degree not just 8 months of it.
Term restrictions are often consistent but there should be a note stating that this is subject to change.

[1] This criteria depends on the release of course schedules — usually 4-8 months ahead of their offering. This criteria will not be applied to courses beyond this period.

*Sample Interaction:*Jane decides she no longer wants to take “CSC 225” in the summer semester. Thinking ahead, Jane determines the only other available semester for this course is in the fall. Once Jane arrives at the *Program Management Page,* she drags “CSC 225” from her summer semester schedule to her fall semester schedule. Graduate! verifies the course is available during this semester, and that Jane has the proper prerequisites. Upon success, Graduate! automatically saves the new schedule.

2.2.7 View recommended schedule

For a user to view the recommended schedule generated by Graduate!, they must navigate to the*Program Management Page*, where they will select **View Schedule**. This will display the recommended schedule based on the criteria specified by the user. From this screen, the user has the option to specify which semester they are interested in viewing.

Will the user be able to create multiple schedules or will they only be able to manipulate and save one? If multiple, then there should be a way to name the different schedules.

Sample Interaction: Jane has recently completed all of the information required to create a schedule. To view her schedule, Jane navigates to the *Program Management Page* where she selects the **View Schedule** button. Jane's generated schedule is loaded and she can continue to make further adjustments as needed.

2.2.8 Change User Preferences

During registration, the user will be prompted to select all of their preferences and restrictions to put in place for their schedule. Alternatively, the user may navigate to the *Program Management Page*. The user will then select **Preferences** and mark a checkbox for the preferences they wish to have in place for their schedule. These preferences include *Set Start/End Time* and *Set Maximum Course Load*[1]. They allow a user to control the time that their day will begin and end, as well as assign a maximum number of courses to their schedule per term. Once the user has selected all necessary preferences, they click **Finish** to generate the updated schedule.

[1] More options may be added in the future or at the request of the client.

Restriction on having no courses in given term (co-op or travel), minimum course load and target graduate date are high priority user preference areas.

Sample Interaction: Jay has been using Graduate! for some time now. Jay decides for his next semester he does not want more than four classes, so he navigates to the *Project Management Page*. Jay selects **Preferences** to initiate the process of updating his current preferences. He marks the checkbox next to "4" under the "Maximum Courses per Term" heading, and clicks **Finish**. Graduate! updates Jay's schedule with the newly specified criteria, and displays it on the screen.

2.3 Glossary

Framework: A collection of software libraries that provide generic functionality that can be customized or extended. A framework provides a fast way to implement a commonly needed function.

Bootstrap: A HTML, CSS, JavaScript framework that facilitates fast and clean user interface design

Operating System: System software that manages computer hardware and software resources and provides common services for computer programs (reference:https://en.wikipedia.org/wiki/Operating_system)

Use Case: A list of steps defining the interaction between a role and a system, to achieve a goal (reference: https://en.wikipedia.org/wiki/Use_case)

Web Application: A client-server software application in which the client (or user interface) runs in a web browser (reference: https://en.wikipedia.org/wiki/Web_application)

3.0 Management Plan

3.1 Feature Breakdown

The core features of the Graduate! application is its course scheduling algorithm, intuitive user interface, user authentication, course and user data storage, course information collection, and Netlink integration.

3.1.1 Course Scheduling Algorithm

Using a table of all the courses and prerequisites that a student needs to take, a vertex graph can be created. From this graph, a list of courses can be created in such an order that no course is taken before its prerequisite. This is done by moving through the graph in a topologically sorted ordering. Next, the list can be moved around based on two constraints; no course can be taken before their prerequisite, and the course must be placed in the semester in which it is offered (every group of blocks in the list represent a term). Variations of this list are created until one grouping creates an optimal degree plan that the user can follow.

Clarify how the algorithm determines what the "optimal" schedule is.
How will you ensure the generated schedule conforms to the users preferences.
What action will the system take when not all of the constraints can be satisfied?

3.1.2 Intuitive Interface Design

The creation of an easy to learn and use interface is critical in the creation of this application. To achieve this goal the Graduate! will utilize modern web application features such as drag-and-drop modification of schedules. Beyond the use of modern tool providing the user with constant feedback will provide the information as to whether or not they are performing valid actions when modifying their schedule. If a user attempts to reorder a course such that it is in the term before its prerequisite there will be clear indication on screen that the action cannot be completed and the courses will revert to their previous valid position.

Rather than revert the schedule, the system should warn the user about the error and provide an option for a manual overide. (Such as situations where people get access to a course even when they don't have the prerequisite)

3.1.3 User Interface Layout

To maximize the screen area a side, or top, menu bar will provide the user with quick access to additional features such as the selection of new courses, the modification of schedule preferences, and changes to account information. Refinement of the user interface will occur through rapid prototyping utilizing styling framework tools provided by bootstrap.

3.1.4 Data Collection/Scraping

The data for both the courses and their pre-requisite/co-requisite can be collected from the UVic databases. The information collected also includes the term in which the courses are offered in and the lab sections that come with it. The data is then transferred onto the personal server, where the course scheduler will be retrieving the data from.

3.1.5 Database Creation

A database will be developed created that stores all the relevant information for the Graduate! application. The database will need to store the collected program requirements in order to associate the courses based on prerequisites, corequisites,labs, electives and alternates. Similarly, the course offerings will be stored based on the particular semesters they are offered. User information and their saved schedule will also be stored in the database.

3.1.6 Netlink Integration

Using your Netlink account, the service is able to view the courses that a user has already taken, or is enrolled in. This way, it can better tailor the remainder of the degrees scheduling to the user. After determining the courses the user has both taken and is enrolled in, those courses can be removed from the overall group of courses that are going to be used to create the schedule for the user. This removes redundancies in what courses the user has to take. As well, pre-requisites can be determined for what courses are going to be scheduled, in case the user took some courses out of the average ordering of the courses they were supposed to take.

For user authentication - Will there be anything in place for password management? IE: Lost passwords, changing passwords, etc.

3.1.7 User Authentication

Users will initially register on the site and provide some required information to the Graduate! application. They will then be able to login with their selected username and password. All sensitive user information will be stored in a safe, encrypted form.

3.2 Possible Implementations

There are a number of technologies suitable for prototyping the Graduate! web application.

3.2.1 Deployment

The application will be deployed using Heroku (www.heroku.com). The service provides an excellent support for multiple web development frameworks and facilitates rapid prototyping. Heroku provides free-of-charge small-scale databases and simple deployment using Git.

3.2.2 Persistent Storage

Heroku provides integrated add-ons for a variety of database management systems. Given the experience of the development team, PostgreSQL or MongoDB will be used.

3.2.3 Server-side Technologies

The requirements for the back-end of the Graduate! web application can be fulfilled by Ruby on Rails, Django, or Node.js. All of the candidate frameworks provide the routing, templating, database object relational mapping, and REST api integrations which are critical to the development of Graduate!.

3.2.4 Client-side Technologies

To create an engaging user interface, the application will rely on a JavaScript library such as Angular, JQuery, or D3. Additionally, to rapidly develop a clean interface, Bootstrap will be employed.

Angular is an extensive front-end web development framework supporting templates and routing. Angular may not be necessary given the choice of Server-side technology.

Implementing a drag-and-drop interface is highly desirable; it will make moving courses between semesters more intuitive and make the application friendly for mobile users. There are several technologies for displaying interactive graphics including SVG (scalable vector graphics), HTML5 Canvas, and JQueryUI.

The use of SVG is particularly promising because they are highly dynamic and would provide the ability to display custom schedules as an interactive graph. To interact with SVG graphics JQuery can be used, but alternative libraries such as D3 (https://d3js.org/) may be better suited as it is specifically designed for the creation of interactive graphics.

3.3 Minimal System

This section describes the minimal system that will be provided by Puzzle by the end of the term. Puzzle is going to develop a proof of concept of the Graduate! application, that shows how it will work for a student taking the Software Engineering program at the University of Victoria.   Once the proof of concept is complete, adding other programs should be a simple matter of data entry. As there is only a month of development time available, it may not be possible to develop a fully functioning scheduling application by the project deadline. As such, Puzzle will focus on implementing solutions to the major weaknesses that are present in the current systems.

ScheduleCourses.com shows how a timetable builder system can be developed for a given semester, so our minimal system will not include such a feature. Rather, our program will leverage the program requirements provided in order to develop a plan for which courses will be taken in which semester, without the specific times being evaluated. This decision was made in part because it is difficult to develop and is already available, and because at this time, the University of Victoria only provides specific times for courses at most eight months in advance. This does mean that our suggested schedule will occasionally result in a course conflict, but at this time there is not a way to avoid this.

Our program will focus on providing students with a simple way to manage their program and view how their scheduling choices affect their course load and final graduation date. It will make it easy for students to fit electives in their schedule, and provide a list of possible electives available that can be taken in a given semester. This allows students to work out a schedule that gives them the electives they desire, as opposed to taking whichever elective happens to fit their schedule. The University provides information about which courses are offered in which terms. Our minimal system will also include an authentication system for users to register and login to the system. Each user will be able to select their program and maintain a saved version of their personalized schedule.

Netlink Integration will not be included in the minimal system. Rather than automatically updating based on a student’s completed courses, users will manually select their completed courses.

3.3.1 Summary

Graduate!’s main focus is to make planning a student’s degree easier and more customizable, while pertaining to the University’s restrictions such as prerequisites and term offerings. In the initial proof of concept, only the Software Engineering program at the University of Victoria will be included. The minimal system provided will include a web application that supports user authentication and dynamic schedule creation and visualization based on course offerings provided by the University.

3.4 Team Structure and Organization

The various components of the project have been broken down into tasks that will be taken on by individual team members of our group. The tasks include everything necessary in order to deliver the minimal system, but will be carried out with the final system in mind.

3.4.1 Web Application Development – Brendan Heal

This task involves the design and development of the web application including the selection of the database, front-end and back-end frameworks, configuration and deployment.

Phase 1: System Design

The web application will be designed with the selected front and back-end frameworks in mind.

Phase 2: Development and Deployment

A basic user authentication will be developed and deployed early so that the remaining development can occur.

3.4.2 Database Creation – Ali Nobari

This task involves the collection of data from the University of Victoria and creation of a database for the web application.

Phase 1: Data Collection

Two important pieces of information will be collected from the University of Victoria's website:

1. The program requirements for the Software Engineering program.
2. The course offerings by term.

Ideally, a scraper will be implemented that can access information from UVic's calendar page.

Phase 2: Database Creation

A database will be created which allows comprehensive access to the data collected as well as user information and user schedules.

3.4.3 User Interface Development – Jonah Rankin

A user interface will be developed according to the client’s requirements.

Phase 1: Prototyping

User interface prototypes will be developed and presented to the client. Upon satisfaction with the user interface proposed, phase 2 will begin.

Phase 2: Interface Implementation

The views will be implemented using the selected front-end framework.

3.4.4 Course Scheduling Algorithm – Spencer Mandrusiak

An algorithm will be implemented to solve the course schedule optimization problem.

Phase 1: Research

The course scheduling algorithm will be researched in order to ensure an optimal solution is selected.

Phase 2: Implementation

The course scheduling algorithm will be implemented and rigorously tested.

Can you please add a visual representation such as a possible GUI of the system to show the intended layout and navigation of the system.

If some elective has more prerequisites that aren't otherwise part of the degree, the system should take that into account. For example, if a student specifies that they want to take MECH 458 as an elective, the system should make note that the prerequisites for that class are MATH200 AND (PHYS216 OR ELEC216).