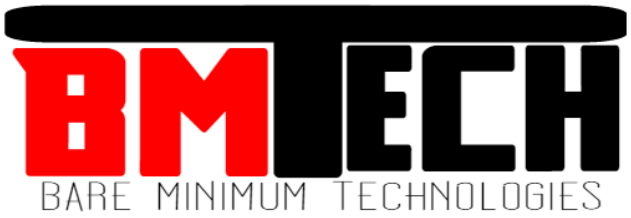


CPP Schedule Builder

1st Quarter Final Report

Cal Poly Pomona

Bement, Johnson,
Le, Llamas, Rutherford

ECE 480

Table of Contents

Introduction

As students of Cal Poly Pomona we have personal experience with the stress and frustrations that go hand-in-hand with the registration process each quarter. While CPP has implemented a few utilities to aid with this process, there is still much confusion surrounding the overall process of planning out and selecting courses over however many years a student is enrolled. We at BMTech are solving this problem using our schedule building software and database systems. This document provides an overview of our results from the first quarter of development and our vision for the future.

Abstract

Development of a schedule-building application for ECE students to utilize in creating their quarterly schedules or to plan an approximate schedule for their time as a student at Cal Poly. This application consists of two primary processes, a "slow" and "fast" brain approach. The "slow" process is responsible for determining class combination options and accounts for all of the associated course properties. The "fast" process receives potential schedules and implements them to determine the most efficient schedule or plan that allows all required courses or prerequisites to be met for current and successive quarters. Additionally, our system utilizes habituation to improve schedule generation time and to generate schedule templates as more students use the utility and tests are completed successfully to train the system. Three specific examples of this are:

1. Incoming Freshmen and Transfer Students generally have similar / the same ECE courses to start with, continued generation of this starting schedule would allow for a generalized starting schedule to test first.
2. Because of the term limitations on certain courses, schedules that start in Fall with courses X, Y, Z will likely have the same successive courses for the Winter and Spring quarters due to availability, which provides a frame of reference for successive schedule generation.
3. Graduating Seniors will likely have similar / the same remaining courses to complete, allowing a generalized graduating senior schedule template.

Glossary

Student

An individual who is enrolled in courses at Cal Poly Pomona.

Advisor

A faculty member that is assigned to students as a mentor.

Administrator

A user that will have the option to modify the current databases and user privileges.

Course

A class articulated in the Cal Poly Pomona course catalog.

ECE

The College of Electrical & Computer Engineering, primarily used as a student's major listing.

BroncoDirect

The system students and faculty use to enroll in classes, check financial aid, view progress on their respective degrees, or view their transcript.

Co-requisites

A course that is required to be taken at the same time as a course, oftentimes a lab.

Prerequisites

A course that is required to be taken before another course.

DPR (Degree Progress Report)

A report that shows what courses a student has taken and what courses still need to be taken in order to obtain the degree the student has declared as their major.

Bottleneck

A high priority course that is a prerequisite for multiple other courses that often prohibits a student from advancing until completion of that course.

GUI

Graphical User Interface

Ontology

A collection of data about data; a database or collection with all corresponding attributes, relationships, or axioms pertaining to all objects in the collection.

Project Overview

The Goal

The goal of our project was to create a program that would benefit the students of Cal Poly Pomona and to have the foundation for expanding to other schools. Looking at CPP's current approach and the future challenges of switching from the Quarter system to the Semester system, we agreed that one of the challenges students encounter at Cal Poly is understanding what classes they need to prioritize in order to complete their degree in an efficient manner. With this topic in mind, our group decided on creating an Intelligent Schedule Builder for our project.

Unlike the current schedule builder which only allows you to create your desired class schedule, our program was designed to take the available class listing on BroncoDirect and run the list through an algorithm to create a customized list of classes that the student should take for that quarter. Our program considers not only what classes the student has taken and is eligible to take, but it also considers how many successive courses rely on that prerequisite, how often a course is offered, and whether the course has any co-requisites. Based on these rules and any specified user-preferences, the program will generate a schedule that the student can save and refer back to. As a proof of concept, we downloaded the list of available Electrical and Computer Engineering (ECE) courses offered at CPP and are using the ECE curriculum sheet as an example degree. We are not including general-ed or other majors at this time, but the final product shall be compatible with any major and general education requirements.

The program takes in data through .csv files. User and Course information is stored in excel sheets and imported in as a comma-delimited file. Once the data has been imported, the program populates the corresponding object properties with the values stored in the load file. Once the course objects are created and a specific user is selected, the schedule generator is able to access all relevant information required to assemble a schedule.

For the future of BMTech's Schedule Builder, the program will be usable by students of all majors at CPP. The long-term goal is to create a base program that will be available to any and every college across the US and potentially the world as well. The program will have a number of customizable options that are enabled and tailored by each individual school's faculty. More features and customization will be available as we continue to monitor and update the base program from user feedback.

Design and Implementation

The Language

Because of our team's prior experience with the language and the ease of interaction between the functional code with the graphical user interface, we decided to develop our project in C#. As students at Cal Poly Pomona, we have access to the full version of Microsoft Visual Studios and were able to take advantage of the intuitive Web Form design tools as well as the C# Application Development tools.

The GUI

Our user-driven design for the schedule builder necessitated an intuitive user interface. The primary users are the Students and Faculty, each with their own purpose for accessing the system. While the students' primary intent of use will be the generation of schedules, the faculty have a wider range of options depending on system maintenance, database changes, and scenario testing. We decided to focus on the student-driven interface in our first quarter development phase as this will be the primary "customer" for the generated schedules. Functionally, we needed user authentication and program access, the basis for selecting courses, a method of generating resulting schedules, and the ability to display and save the resulting schedules.

Ultimately, our system will be integrated with the existing CPP Bronco Direct web services that will be responsible for user authentication. For our proof of concept, we included a login screen *(App. A, Fig. 1)* to verify user credentials with our directory before loading the main form. From there, the current course listing is accessible via a dropdown menu (*App. A, Fig. 2)* for adding any course preferences the student may have. After selecting any preferred courses, the schedule generator will compile a list of which courses the student is eligible for, and sort them by priority. From this list, any time conflicts will be addressed, and a schedule will be generated. The resulting schedule and time slots are displayed on the form *(App. A, Fig. 3)*, and are available to print and save.

The Objects

Our objects fall into two categories: user or course. Course objects are functionally identical; while each course has different properties, they are static objects that are only used as references. Users on the other hand each have different needs from the system and different levels of clearance. We designed our user objects with set clearance levels that determine what methods are available after login. For instance, while a faculty member may have access to student information for the purpose of advising or determining course offerings for a new quarter, a student does not have access to other students' personal information. Additionally, we can use the clearance of the user as a selector between which form is accessed. A student will go directly to the schedule generator at login, while the system administrator will be able to select from a wider variety of features.

We constructed our initial load files in Microsoft Office Excel to be stored as a comma-delimited csv file *(App. B)*. As Cal Poly currently only releases future classes for one quarter at a time, we used Winter 2017 as our official planned quarter when testing the generator. For our proof of concept design, we limited each course offering to a single section, reducing our possible course options to 60. Each course object retained the Cal Poly listings for meeting days, time of day, any co or prerequisites, and their unit value. For the students, we include their full name, BroncoID, current major, current status, and any completed coursework to simulate integration with BroncoDirect and pulling from the Degree Progress Report.

After importing all relevant data, our algorithm begins by running the master list of available courses through a series of filters specific to the currently selected student. The first stage is a comparison against all courses a student has currently completed and eliminating any that would be a repeat (assuming a student has successfully passed that class). The second stage is pre-requisite validation and flagging any course that the student has meet the requirements for. The remaining course listing is then sorted by priority, taking into account any previously selected preferred courses when the student initiated the schedule generator. The algorithm then compares multiple schedule options to find the "best-fit" schedule that allows the most critical courses to be completed while avoiding any time conflicts.

End Results

Product Status

During the design process of this this project, our end-goal was to implement a fully functioning schedule builder with an integrated GUI to handle all interactions with the system. As the project came to its final weeks, we came to the realization that our goal was too ambitious given the amount of time that we had. The complexity of the system prevented us from accomplishing some of our goals within the ten week period. However, we successfully created a system that would prioritize courses based on the student's major, prerequisites, days of the week, time of day, and units and be able to generate a schedule.

Our initial goal was to be able to access and read our databases housing all the information necessary for the schedule builder to perform prioritization. Using comma-separated values files, we managed to read the data and store the various information in lists for further manipulation. In order to test our schedule builder, the .csv file contained user information that listed a variety of courses representing the student's completed courses to simulate the user's academic progress.

Based on the student's major, whether they enrolled as an electrical or computer engineer, the system would filter the courses displaying only those mandatory to the student's particular major. With the student course history available to the schedule builder the system would use that information to also determine if prerequisites were met. When prioritizing courses, the system would also check any time preferences selected by the user. The system prioritizes courses based on the lowest course number that student needs to complete.

We have a fully functioning interface that allows us to select the course, start time and end time from separate drop-down menus. The drop-down menus are populated using the .csv database containing the course catalog and all available times. After the user inputs their desired courses, the GUI displays the selected courses in a schedule format using the built in data grid view. We were not able to fully integrate the prioritization algorithm within our GUI, however the GUI itself as a standalone interface functions as intended, and the framework for fully linking it with our prioritization algorithm is in place.

Using several test cases, the prioritization algorithm works as intended when implemented using the console window. We have confirmed this for several users, each with their own separate degree progress. Based on which courses have been satisfied by the user, the schedule builder returns a recommended list of courses that the student is required to take. Our main objective of designing a system that effectively generates a prioritized schedule based on the student's degree progress report was ultimately met.

Discussions

As with every project, there were several obstacles that presented themselves throughout the course of a project. However, we came together as a team to overcome them. It has been a learning experience for each of us and we are proud of the product we have created.

Among the problems we encountered was the issue of scheduling. As students, we each have different schedules, so collaborating and meeting as a team wasn't always possible. To accommodate for this, our solution was to set up sessions where we could meet up remotely by using a messaging service such as "Slack". This made it so that we could meet with each other without always having to physically travel to the meeting place. We also wanted to make sure we were all working on the most up to date version. To accomplish this our team utilized the online service called GitHub. Other obstacles we encountered were reading the data from the database. To overcome this, we set the data file as a ".CSV" file. This made it easy to pull the information into lists in C#. The last obstacle we ran into was integrating the different aspects of our project. The solution for this was communication and adequate testing.

Throughout the course of this project, we were exposed to many new project management utilities, communication tools, and resources for online collaboration. Additionally, we had many opportunities to practice clear, precise, and effective communication as well as time management for individual projects, and managing the time frame of the project as a whole. Each member was stretched in their software skills, interface design, and the many stages of development throughout the quarter. This project has been remarkably enjoyable, and it is with much excitement that we are able to say that we were able to successfully meet many of the requirements laid out in our Software Requirements Document.

Conclusion

The primary purpose of this project was to illustrate the various procedures that must be followed throughout the life-cycle of a software project. While there are features that we were not able to include due to the time restriction of fall quarter, we successfully designed the required algorithm to generate schedules and laid the foundation for fully integrating our system with Cal Poly Pomona's web services. We believe that this project can improve the entire process of planning out courses for a single quarter and in constructing an overall plan for each year of enrollment. Though we at BMTech will no longer be responsible for the continued design of this project, we hope that it still inspires change and demonstrates that a new method of approaching the problem of assigning courses is possible.

We would like to extend a heartfelt thank you to the fellow teams of ECE 480 for their feedback throughout the quarter, and the questions they brought to us that helped shape our design and goal for the system. Additionally, we would like to thank Dr. Yin for her instruction and feedback on all portions of our project, as well as for the introduction to Software Engineering, and the processes involved.

BMTech

David Bement

Lukas Johnson

David Rutherford

Tony Le

Josue Llamas

Appendix A

GUI – Graphical User Interface

Figure 1 – Login Prompt

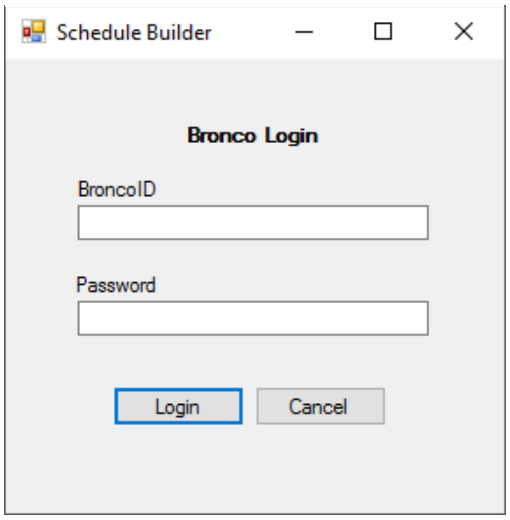


Figure 2 – Course Dropdown

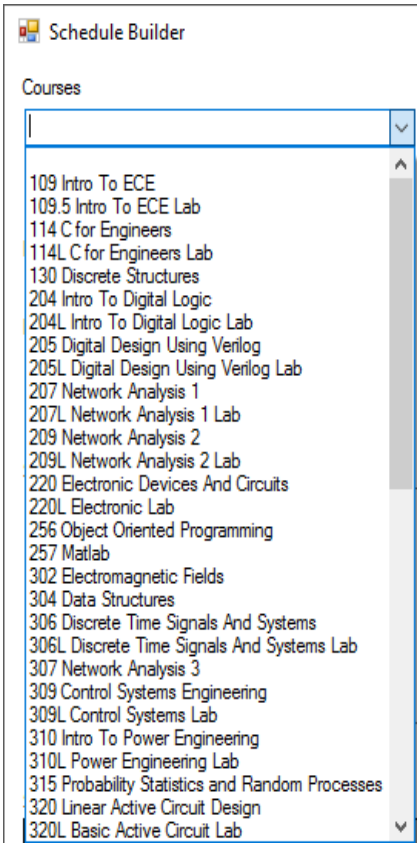
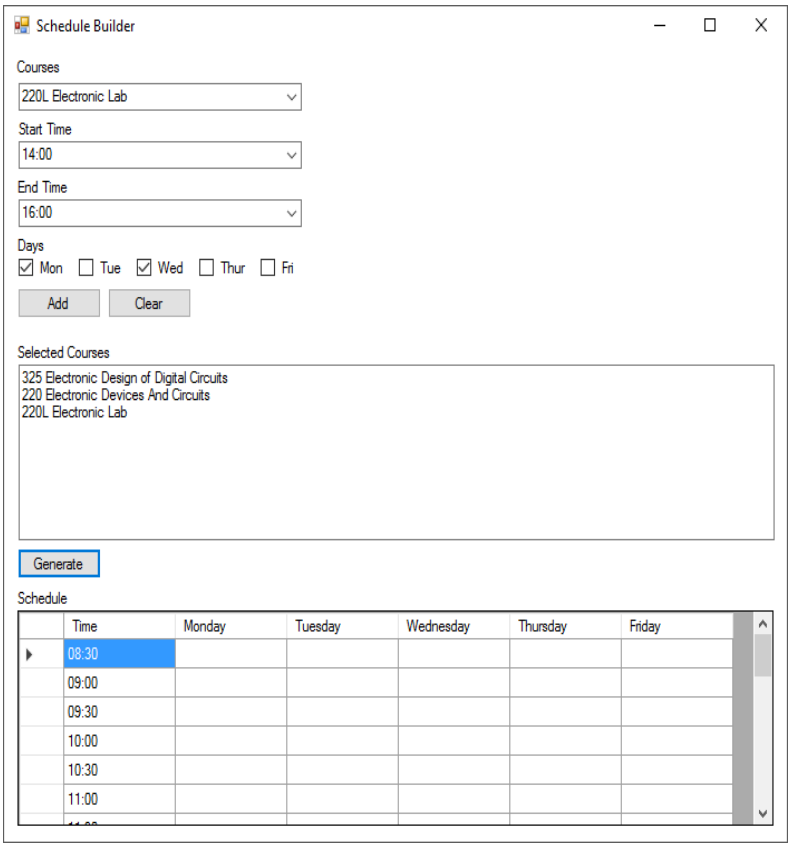


Figure 3 – Main Form with Schedule View



Appendix B

Load File Samples

Course Number	Course Name	Units	Major	Days	Begin	End	Core	Pre	Pre	Pre	Pre
325.5	ElectronicDesignofDigitalCircuitsLab	1	CP	200	900	1150	325	205			
330	IntroToSemiConductors	3	EE	130	1500	1615	X	220	302		
341	IntroToMicrocontroller	3	Both	240	1430	1545	341.5	205	205.5	207	
341.5	IntroMicrocontrollerLab	1	Both	400	1600	1850	341	205	205.5	207	
343	MicroProcessor1	4	Both	130	1400	1550	343.5	204	204.5		
343.5	MicroProcessor1Lab	1	Both	200	1600	1850	343	205	204.5		
405	CommunicationsSystem	4	EE	240	1000	1150	X	315	307		
405.5	CommunicationsSystemLab	1	EE	400	1600	1850	X	405			
407	CMOS Analog Circuits	4	Both	130	900	1050	X	320			
408	DigitalSignalProcessing1	3	Both	130	1200	1315	X	306			
408.5	DSPLab	1	Both	200	1300	1550	X	408			
418	IntegratedCircuitsDesignandFabrication	4	Both	130	1800	1950	X	320	330		
419	AdvancedControlSystems	3	Both	130	1330	1445	419.5	309			
419.5	AdvancedControlSystemsLab	1	Both	200	1600	1850	419	309			
422	PowerSystemAnalysis	3	Both	130	1030	1145	422.5	310			
422.5	PowerSystemsLab	1	Both	300	1200	1450	422	X			
424	DigitalSystemDesignUsingVHDL	3	Both	130	1630	1745	424.5	205	205.5		
424.5	DigitalSystemDesignUsingVHDLLab	1	Both	100	1800	2050	424	205	205.5		
425	ComputerArchitecture	4	CP	130	900	1050	X	341	341.5		
426	OSforEmbeddedAppications	3	CP	200	1300	1550	426.5	256	341	341.5	
426.5	OSforEmbeddedAppicationsLab	1	CP	400	1300	1550	426	257	341	341.5	
431	ComputerNetworks	3	CP	240	900	1015	431.5	341	341.5		

Admin_LVL	1	2	3	0	0	0	0	0
AdminType	Assistant	Staff	Administrator	Student	Student	Student	Student	Student
Last_name	Doe	Torrence	Kang	Johnson	Bement	Rutherford	Llamas	Le
First_name	John	Margel	James	Lukas	David	David	Josue	Tony
ID#	321456789	987654321	987654123	123456789	789654231	555123456	888654789	111456852
Major	X	X	X	EE	Cp	Cp	Cp	Cp
Student_LVL	X	X	X	Sophmore	Junior	Freshman	Senior	Senior
Completed1	X	X	X	109	109	109	109	109
Completed2				114	114	114	114	114
Completed3				114.5	114.5	114.5	114.5	114.5
Completed4				204	130	130	130	130
Completed5				204	204	X	204	204
Completed6				204.5	204.5		204.5	204.5
Completed7				205	205		205	205
Completed8				205.5	205.5		205.5	205.5
Completed9				256	256		256	256
Completed10				207	207		207	207
Completed11				207.5	207.5		207.5	207.5
Completed12				209	209		209	209
Completed13				209.5	209.5		209.5	209.5
Completed14				220	220		220	220
Completed15				220.5	220.5		220.5	220.5
Completed16				320	325		325	325
Completed17				320.5	325.5		325.5	325.5

Appendix C

Demonstration Results

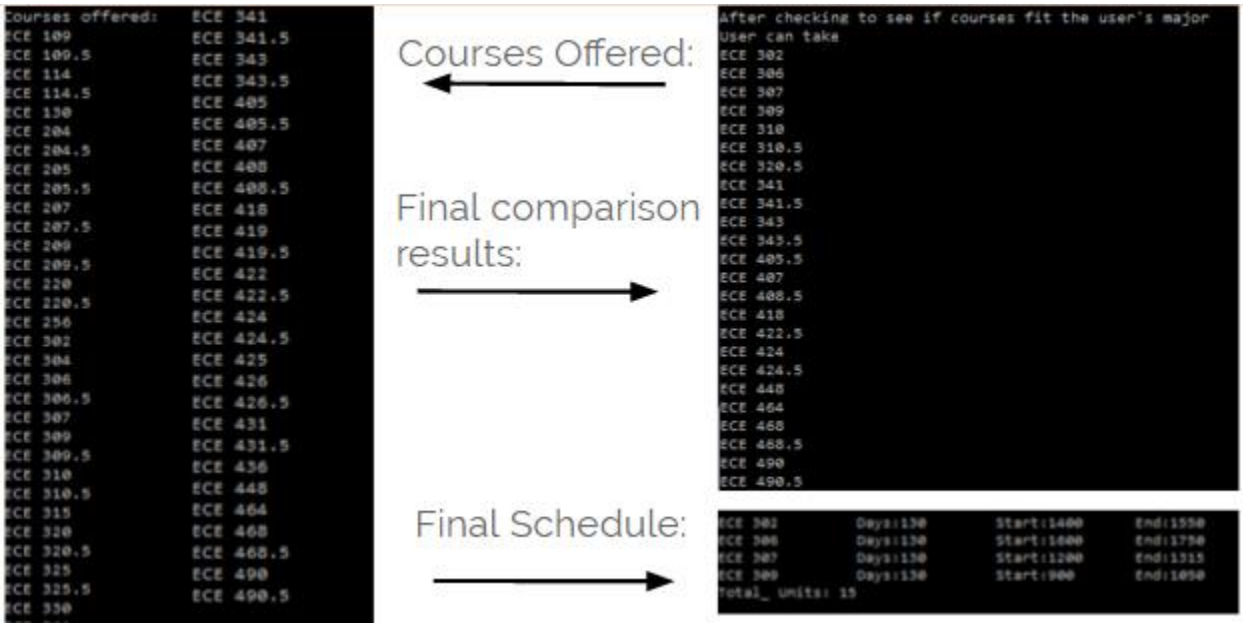
Figure 1 – Sample Student Object

```
User's name: Llamas, Josue
User's ID#: 888654789
User's type: Student
User's department/major: Cp
Student level: Senior
Courses taken:
109     114     114.5   130     204     204.5   205     205.5   256
207     207.5   209     209.5   220     220.5   325     325.5   306
306.5   309     309.5   304     341     341.5   425     425.5   315
302     431     431.5   426     426.5   414     414.5   343     343.5
464     480     408     408.5   481     482     467
```

Figure 2 – Sample Course Object

```
Course: 426.5
Course name: ComputerArchitecture
Units: 1        Major(s): CP
Day/Times: 400 1300 1550
Coreq: 426
Prereq: 257     342     341.5
Press any key to continue . . .
```

Figure 3 – Sample Schedule Generation Filtering Progression



References

Slack: Be Less Busy

A message application that integrates project management utilities, google drive, GitHub, and more in one handy messaging app, available on mobile, web, and desktop platforms.

https://slack.com/

Zoho: Online Project Management Software

A project management utility that integrates with many messaging platforms, online repositories, and many more services. An easy to use method of creating and tracking tasks, project progress, and timeline progress.

https://www.zoho.com/projects/

Google Drive - Cloud Storage & File Backup for Photos, Docs & More

Get access to files anywhere through secure cloud storage and file backup for your photos, videos, files and more with Google Drive

https://www.google.com/drive/

How people build software: GitHub

Online project hosting using Git. Includes source-code browser, in-line editing, wikis, and ticketing. Free for public open-source code.

https://github.com/

Protégé

Protégé is a free, open-source ontology editor and framework for building intelligent systems.

http://protege.stanford.edu/

Stack Overflow

Stack Overflow is the largest online community for programmers to learn, share their knowledge, and advance their careers.

http://stackoverflow.com/

Real-Time Maintenance Prioritization with Learning Capability

A radical approach for real-time maintenance prioritization where the main idea is drawn from neuroscience studies.

Meng-Lai Yin, PhD, Electrical & Computer Engineering, California State Polytechnic University, Pomona

Andrew J. Chan, Electrical & Computer Engineering, California State Polytechnic University, Pomona