Degree Audit Web Form for Computer Engineers

Julian Callin Pranav Bheda Carlos Rivera

COEN 174 - Software Engineering

Dr. Atkinson and Nathan Matsunaga

11 October 2016

Table Of Contents

Abstract

The current computer engineering degree tracking system used at Santa Clara University is outdated in layout and inefficient in communicating degree requirements in a clear and concise manner. The proposed solution is to use a website where both students and their advisors can access a one page form that updates dynamically to student class input. In the proposed solution, a summary of the completed requirements and possible additional classes to fulfill remaining requirements will be downloadable and easy to distribute.

Introduction

At Santa Clara University, computer engineering students and advisors meet to discuss graduation requirements, which include core classes, technical electives, and educational enrichment electives. Right now, students and advisors have no easy way to view this information in an organized manner.

Currently students and advisors have access to a web-based tool called the “Degree Progress Report,” through which graduation requirements are tracked. This system for obtaining a degree audit is through the school’s online portal, eCampus, which fails to give information in the most efficient way.

Reports are rendered as a webpage in plain text which is often hundreds of lines in length. In addition, a single font size, ascii art formatting, and the use of a simple two-color coding scheme limits the student’s ability to discern necessary information from the report. Students and advisors have difficulty seeing which classes have fulfilled a particular requirement (electives, educational enrichment, major, etc.). Students are unable to store or transport this information since the report is not available in a downloadable format. Another issue with the system is that viewers are unable to see which classes can count towards educational-enrichment. Through eCampus, advisors have to laboriously identify and count remaining classes that can be used towards educational-enrichment.

Our alternative to the degree progress report is a website where students and advisors can track their requirement progress using a one-page form. Students can enter all courses they have completed and plan to take in future quarters. This information will be stored persistently between sessions. In an organized, color-coded, and downloadable manner, our system will dynamically display satisfied and remaining requirements. Our system will provide students the ability to tinker with their future schedule. In addition, unused credits will be displayed so students can decide to use them for educational enrichment.

Our system will allow students and advisors to see which core, elective, educational-enrichment, and major requirements have already been fulfilled and which ones remain, in a clean visually appealing, and responsive manner.

Requirements

This section outlines the requirements for our system to meet its functionality goals.

Functional Requirements

- The system will take a student’s input and give information regarding completion of degree requirements.
- The system will tell students whether a major requirement is complete or incomplete.
- The system will tell students whether a core requirement is complete or incomplete.
- The system will tell students whether they possess extra class credits which can be used toward education enrichment requirements.
- The system will persist a student’s degree audit between sessions.
- The system will allow a student or adviser to download or print the degree report.

Non-Functional Requirements

- The system will be user friendly.
- The system will be easily testable.
- The system will be easily maintainable.

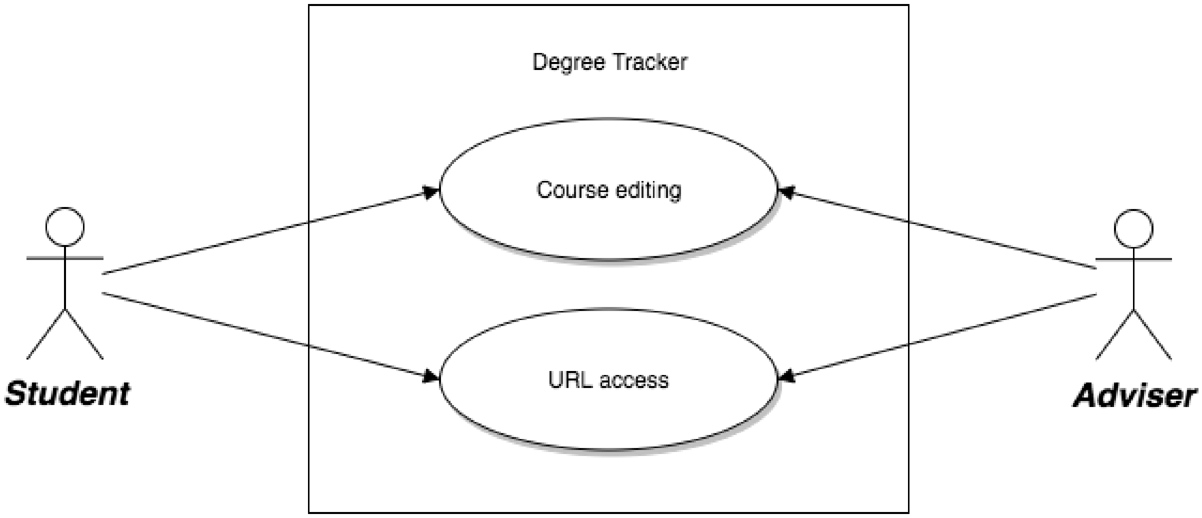
Design Constraints

- The system must run on the SCU Engineering Compute Center Linux and Windows machines.
- The system must retain functionality on both web browsers, Firefox and Chrome.
- The system must be completed by the week 10 presentation date.

Use Cases

This section describes the situations in which the users of our system will perform specific functions. The actors for our system will be the student and their respective adviser; both will have equal access and editing privileges.

Use Case Diagram



Use Case 1: Course Editing

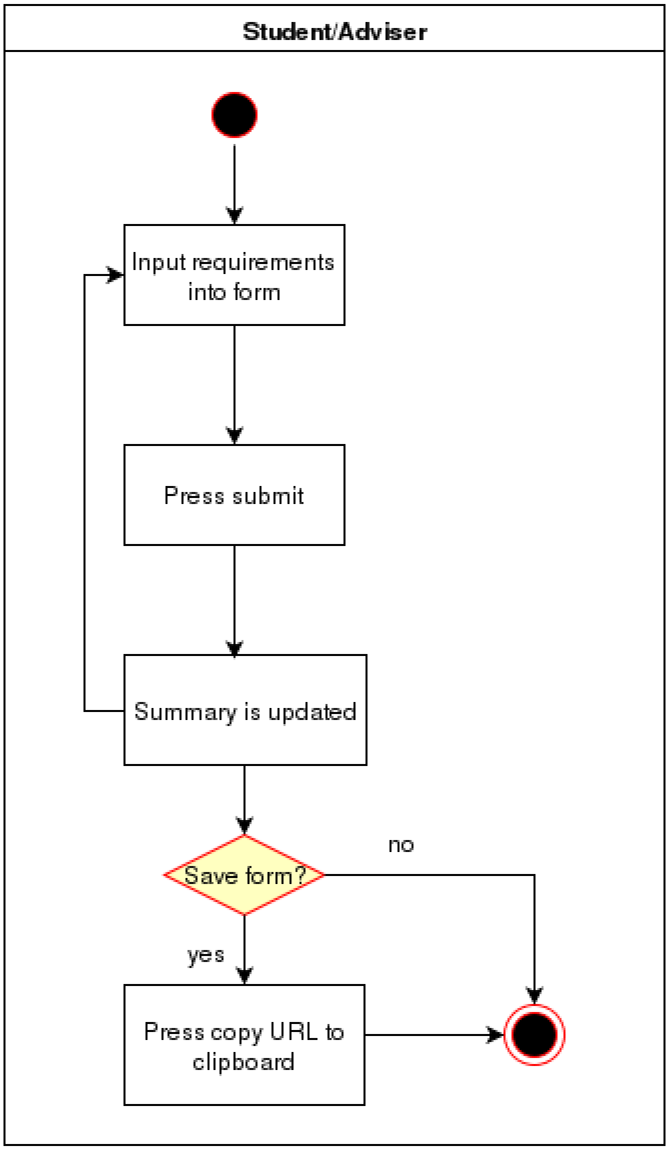
Actor	Students, Advisers
Goal	Edit current and potential courses
Preconditions	Student must access website
Postconditions	Fulfilled requirements will reflect updated taken courses list
Steps	1. Search by URL 2. Fill out class schedule 3. Refresh
Exceptions	Report does not exist

Use Case 2: URL Access

Actor	Students, Advisers
Goal	Access student-specific tracker report using URL
Preconditions	Student report must exist
Postconditions	View student progress using degree tracker
Steps	1. Search using generated URL
Exceptions	Invalid URL, report does not exist

Activity Diagram

This activity diagram describes the flow of action that a user will experience when using our system.

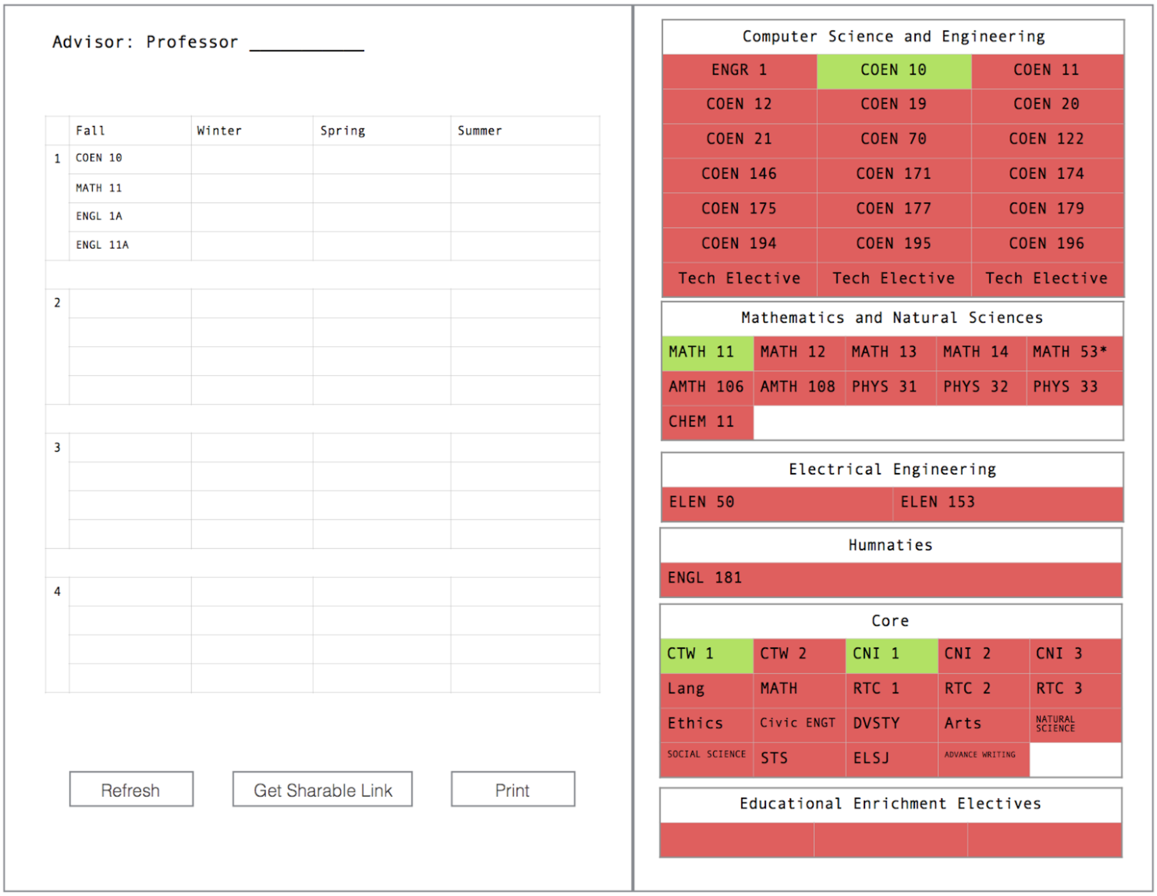


Conceptual Model

The conceptual model describes how users will interact with our solution. It describes how they will use it, how they perform functions that the website offers, and what the users will see.

This model shows what the users will view when they are using our website. Users will be able to type in each of the classes they have taken each quarter, over all four years. The bottom shows three buttons, *Refresh*, *Get a Shareable Link*, and *Print*. After the user types in which classes they took, they can click the *Refresh* button. Then, the cells on the right side will turn from red to green, depending on the classes and what requirements they satisfy. Users can then proceed to generate a link where they can send their degree audit form to an advisor or print out the form in a clear and organized manner.

Mockup



Advisor: Professor ___________

	Fall	Winter	Spring	Summer
1	COEN 10			
	MATH 11			
	ENGL 1A			
	ENGL 11A			
2				
3				
4				

Computer Science and Engineering		
ENGR 1	COEN 10	COEN 11
COEN 12	COEN 19	COEN 20
COEN 21	COEN 70	COEN 122
COEN 146	COEN 171	COEN 174
COEN 175	COEN 177	COEN 179
COEN 194	COEN 195	COEN 196
Tech Elective	Tech Elective	Tech Elective

Mathematics and Natural Sciences				
MATH 11	MATH 12	MATH 13	MATH 14	MATH 53*
AMTH 106	AMTH 108	PHYS 31	PHYS 32	PHYS 33
CHEM 11				

Electrical Engineering	
ELEN 50	ELEN 153

Humnaties
ENGL 181

Core				
CTW 1	CTW 2	CNI 1	CNI 2	CNI 3
Lang	MATH	RTC 1	RTC 2	RTC 3
Ethics	Civic ENGT	DVSTY	Arts	NATURAL SCIENCE
SOCIAL SCIENCE	STS	ELSJ	ADVANCE WRITING	

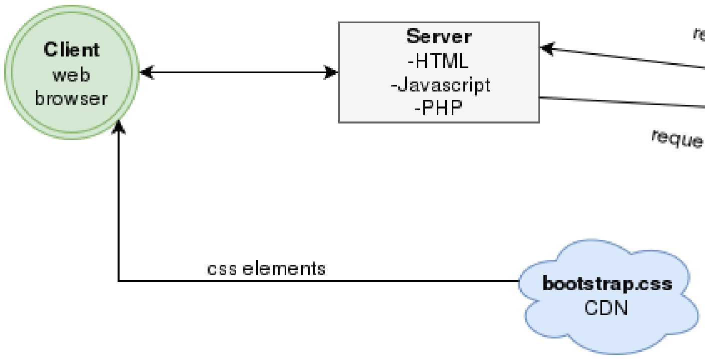
Educational Enrichment Electives		

Technologies Used

- HTML: A simple, universal web markup language
- Bootstrap CSS: A framework for web design and formatting
- Javascript: A popular web language
- JSON: A text format for storing hierarchical data
- Git: A method of source code control
- Github: A host for source code and a collaborative environment

Architectural Diagram

The architectural diagram give a high-level representation of how all the components in our system interact to provide functionality to the user.



Index.html is designed using bootstrap.css elements. If a user does not specify a URL besides '/index.html', Index.html will take user input and process it using script.js. Script.js assesses which requirements are completed and sends the information to the server where a PHP script writes it to a JSON file. Script.js will re-populate index.html with data which the user can observe. If a user submits a url such as '/index.html?user="JAMES"', the PHP on the server will attempt to read information from JAMES.json and send requirements data from a previous session back to the client for population of the web form. This architecture fits our solution because it is simple and does not require the implementation of a complex web server or a relational database.

Design Rationale

UI Design overview

The one-page UI layout was chosen because it eliminates the need for users to navigate between pages and scroll extensively to find information. The vertical page division was implemented in order to present the user with a side-by-side view of the data they entered and the information that was retrieved for them. This model does not require the user to remember which requirements were entered by splitting the website into an submission page and a display page.

Technical Design Overview

Basic web technologies such as HTML, CSS, Javascript and JSON are used because they are lightweight, stable, portable, and very simple. Bootstrap CSS provides the webform with a sleek and modern look, HTML is interpreted easily by any browser, JSON simplifies the storing and reading of data, and Javascript integrates with the other technologies smoothly as a front-end data processor and server requester. We chose to use the URL-JSON access method over a relational database because a relational database is hard to set up on the design center computers. Also, the functionality of our system does not require advanced querying provided by a relational database management system.

Test Plan

The proposed solution has three major areas that need to be tested: saving, updating, and requirement fulfillment. The student's class input should also be tested to see that all classes and the requirements they fulfill are saved onto the student's profile. During the testing process, we will have various users use our website to see what classes they have left. We will consider all of their feedback when making changes to the user interface. Also, we will test the browser compatibility by checking to see if the website runs on Firefox and Chrome, on both the Linux and Windows machine in the Engineering Compute Center.

To make sure our logic works, we will test to see each class fulfills a requirement. For example, when we add COEN 10 as a class, we will make sure that the right hand column turns the COEN 10 cell from red to green. Then, we will test all the core classes, to make sure that the logic works. For example, when we add ECON 1, we will make sure that the Social Science core requirement cell turns green. There are also some classes that fulfill multiple requirements. POLY 2 fulfills the Social Science and CNI 3 requirements. When we add ECON 1, the Social Science should turn green. When we add POLY 2, the CNI 3 cell will turn green, and the Social Science will remain green. Then, if we remove ECON 1, the Social Science cell will remain green, since POLY 2 satisfies the requirement.

Alpha Testing

For the Alpha testing, each member of our team will each enter in all the classes that they have taken to make sure that the system performs as expected. As computer engineers with many completed requirements, we should be able to quickly discern obvious implementation flaws.

Beta testing

After each of the team members test schedules, we will ask COEN students to test our system. We will have each user type in their schedule to see which requirements they have left in order to graduate.

Risk Analysis

This section considers risks that our team faces during the development of our system and how we will work to mitigate those risks.

Risk table

Risk	Consequence	Probability	Severity	Impact	Mitigation Strategy
Run out of time	Cut features	.4	9	3.6	Ensure team members are on schedule
Unexpected difficulty of feature implementation	Increased time to production	0.6	6	3.6	Allocate more time than necessary for component completion
Test cases will not completely cover all possible requirement combinations	User may experience unexpected behavior	0.3	8	2.4	Write unit tests or smoke tests using data, do more integration and user acceptance testing
Developer Illness	Not enough time to complete the system	0.7	3	2.1	Eating and sleeping healthily, taking vitamins
Misunderstood Requirements	Customer is not satisfied	0.2	7	1.4	Meet with Professor Atkinson and Lab TAs often
Scope creep	Increased time to production	0.3	4	1.2	Plan thoroughly, stick to requirements

Development Timeline

This section contains a work schedule for all members of the team.

Gantt Chart

Tasks	**Weeks**										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**-**	**10**
Documentation											
Problem Statement	🟨	🟨									
Design Doc		🟨	🟨	🟨							
Design Review				🟨	🟨						
Initial Project Demo					🟨	🟨	🟨				
Final Report					🟨	🟨	🟨	🟨	🟨	🟨	🟨
Final Presentation					🟨	🟨	🟨	🟨	🟨		
Design											
Design Website Frontend				🟩	🟩	🟩					
Design Degree Audit Algorithm				🟥	🟥	🟥					
Design Website Backend				🟦	🟦	🟦					
Implementation											
Build Website Frontend					🟩	🟩	🟩	🟩	🟩	🟩	
Build Degree Audit Algorithm					🟥	🟥	🟥	🟥	🟥	🟥	
Build Website Backend					🟦	🟦	🟦	🟦	🟦	🟦	
Testing											
Test Browser Capability						🟨	🟨	🟨	🟨	🟨	🟨
Test Degree Audit Algorithm					🟨	🟨	🟨	🟨	🟨	🟨	🟨
Test ease and correctness					🟨	🟨	🟨	🟨	🟨	🟨	🟨
Test Usability				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨
Legend	🟨 Everyone			🟥 Julian			🟩 Pranav			🟦 Calos	

Conclusion

Using simple, portable web technologies, and a one-page user interface, our system will solve the many problems that plague the current degree audit on SCU eCampus. Students will have a visually pleasing, fast, portable, and user friendly way of assessing their degree progress. In addition to this, the student's adviser will be able to easily track the student’s progress.