

SCUCourses.fish: Course Schedules for Freshman

Team Aces: James Terry, Nick Peacock, Tracey Acosta, Benjamin Giglione

December 2, 2014

Contents

1	Introduction	1
2	Requirements	1
3	System Model	2
4	Use Cases	3
5	Architectural Diagram	4
6	Technologies Used	4
7	Design Rationale	5
8	Risks Table	6
9	Test Plan	6
10	Development Timeline	8
11	Lessons Learned	8
A	User Manual	9

List of Figures

1	Mock up for major and test selection	2
2	Mock up for entering test scores	2
3	Mock up for entering transfer credits	3
4	Mock up for potential course schedule	3
5	Use Case diagram for our system	4
6	Architectural Diagram for our product	5
7	GANTT chart for the development of our team project	10

1 Introduction

All freshmen at SCU goes through an advising period during their summer orientation. The advising period is brief, stressful, and not as informative as many would like it to be. Disparities between the number of faculty advisors and students arise frequently. The first-year schedules themselves are frequently complicated by AP credits, transfer credits, and other exemptions. Sorting through these exemptions is not a straightforward process, as the number of possible course exemptions is rather large. Based on AP scores and the math readiness exam a new student could start in four different math courses. New students, who are unfamiliar with the scheduling process, are told to attempt to determine which courses are covered by their credits and generate schedules before meeting with an advisor. Their proposed schedules are often poorly constructed, which the advisor must then salvage. Given that the advising process is relatively brief, advisors cannot always create optimal schedules for students. The current process leaves many new students without adequate attention, the faculty feeling unhelpful, and the schedules unoptimized.

Team Aces proposes a better solution to SCU's current advising process: SCUCourses.fish, a web application. On the surface we will have a clean and intuitive interface that will allow new students to input their previously earned credit along with their proposed major. In turn they will receive optimized first-year schedules outlining which courses they should take. Our goal is to have the application optimized for the Computer Science and Engineering major and the Web Design and Engineering major. This web application will include core, C&I, CTW, and major specific courses. Students will be asked for their AP scores and possible transfer credits to determine which SCU courses are covered, and will be given possible alternative schedules. For instance, if a student received an AP score of 5 in Calculus AB his or her schedule will be adjusted so that he or she starts in MATH 13 instead of MATH 11. The outputted first-year schedules will be exportable in a form that can be easily saved, emailed, and printed.

Using Team Aces' web application will alleviate stress and error caused by this initial advising process. Students will have a clearer grasp of which courses need to be taken and when they should be taken. They will also know what information advisors want before their advisement period since the web application will have already asked for it. Advisors will be able to focus their time on the complex cases that our web application simply cannot handle. Our application will by no means be perfect, but should be able to handle a majority of cases. That being said, we believe that our web application will provide a noticeable improvement to the advising process.

2 Requirements

Based on the needs expressed to our team by the customer we determined the following are the requirements. They are split up between functional, non-functional, and constraints.

2.1 Functional

The functional requirements are requirements that must be fulfilled in order to make the web application work. The web application will allow students to:

- Work for Computer Engineering and one other engineering major, Web design was chosen
- Select any AP and IB tests taken and receive scores, and transfer credits and generate first year schedule
- If two quarters in a row have a CORE slot, and C&I sequence in those slots
- Print a proposed schedule

2.2 Non-functional

The nonfunctional requirements are requirements describe how the system will behave. The web application will be:

- Portable (will work on major browsers like Internet Explore, Chrome, Firefox, and Safari)
- User friendly (straight forward, simplistic, easily understood)

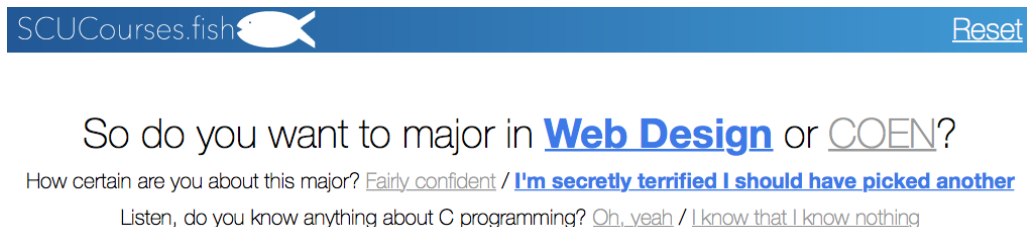
2.3 Constraints


The web application must be able to:

- run on design center computers
- be a web based application

3 System Model

Our web application was designed as follows so as to meet our requirements set by the customer. Initially, as seen in Figure 1 the user will be prompted to selected his or her major and types of tests (AP and IB) he or she has taken.



SCUCourses.fish  [Reset](#)

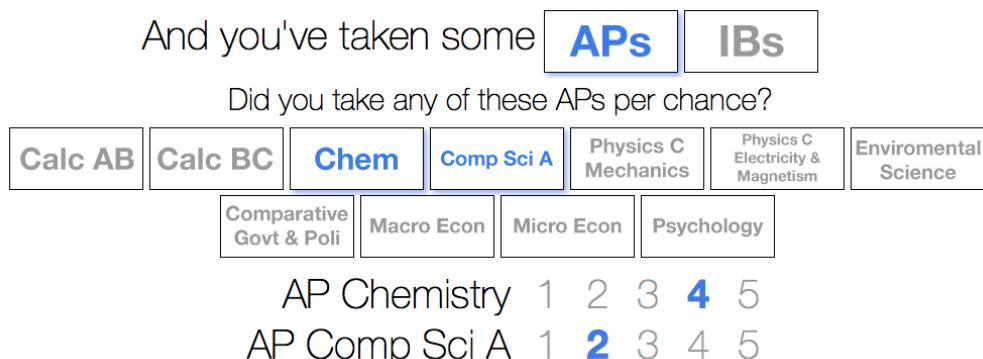
So do you want to major in [Web Design](#) or [COEN](#)?

How certain are you about this major? [Fairly confident](#) / [I'm secretly terrified I should have picked another](#)

Listen, do you know anything about C programming? [Oh, yeah](#) / [I know that I know nothing](#)

Figure 1: Mock up for major and test selection

The user will then be prompted to select the tests he or she has taken in the past. A list of the user's selected tests will populate below, where the user will be able to select the the scores of these selected tests. The user will also be prompted to input his or her Calculus Readiness Exam score if applicable. This can be seen in Figure 2.



And you've taken some [APs](#) [IBs](#)

Did you take any of these APs per chance?

Calc AB	Calc BC	Chem	Comp Sci A	Physics C Mechanics	Physics C Electricity & Magnetism	Enviromental Science
Comparative Govt & Poli	Macro Econ	Micro Econ	Psychology			

AP Chemistry 1 2 3 **4** 5

AP Comp Sci A 1 **2** 3 4 5

Figure 2: Mock up for entering test scores

As Figure 3 shows, the student will then be prompted to select which courses they have transfer credits for. In the case where they have credit for multiple courses in a series the system will check and cross out the pervious courses in the sequence.

Did you pass the calculus readiness exam? [Yes, with flying colors](#) / [Just put me in pre-calc](#)

Do you have any transfer credit? [But of course](#) / [Sadly, I don't](#)

Credit in: [Science](#) / [COEN](#) / [Math](#)

☒ COEN 10 ☒ COEN 11 ☒ COEN 12 ☐ COEN 19

Figure 3: *Mock up for entering transfer credits*

As seen in Figure 4, the user will then be shown his or her potential schedule as well as suggesting which CORE credits should be taken in free spots in the schedule. The printer version has a printer friendly color scheme as well as providing an audit of the information the student inputted into the system.

Fall	Winter	Spring
MATH 11	MATH 12	MATH 13
RTC 1 [CORE]	Diversity [CORE]	ELSJ [CORE]
Social Sci [CORE]	C&I 1 [CORE]	C&I 2 [CORE]
CTW 1	CTW 2	RTC 2 [CORE]
ENGR 1		




Figure 4: *Mock up for potential course schedule*

4 Use Cases

The following use case scenario provides a brief but in-depth explanation of the web application functions and how a user would interact with them. Figure 5 is the use case diagram, which illustrates how the web application work. The student (user) has the ability to choose his or her major, select any previously taken tests and add the respective scores, select courses that are covered by any previous credit, and view and print the proposed schedule. Below is an in-depth description of the use case scenarios shown in Figure 5.

Actor: Student

Goal: Select major

Pre-condition: Actor is a new student at Santa Clara University

Post-condition: Schedule adjusts

Scenario:

- Click on major drop down list
- Click on major

Actor: Student

Goal: Input test scores

Pre-condition: Actor is a new student at Santa Clara University and chosen a major

Post-condition: Actor has successfully inputted test scores and viewed an optimized schedule

Scenario:

- Click on types of test
- Click on tests taken
- Click on scores for tests taken

Actor: Student

Goal: Select courses already covered by credit and view optimized schedule

Pre-condition: Actor is a new student at Santa Clara University and chosen a major

Post-condition: Schedule adjusts

Scenario:

- Click on courses already covered

Actor: Student

Goal: View and print schedule

Pre-condition: Student inputs Credits

Post-condition: Schedule Printed

Scenario:

- Click print

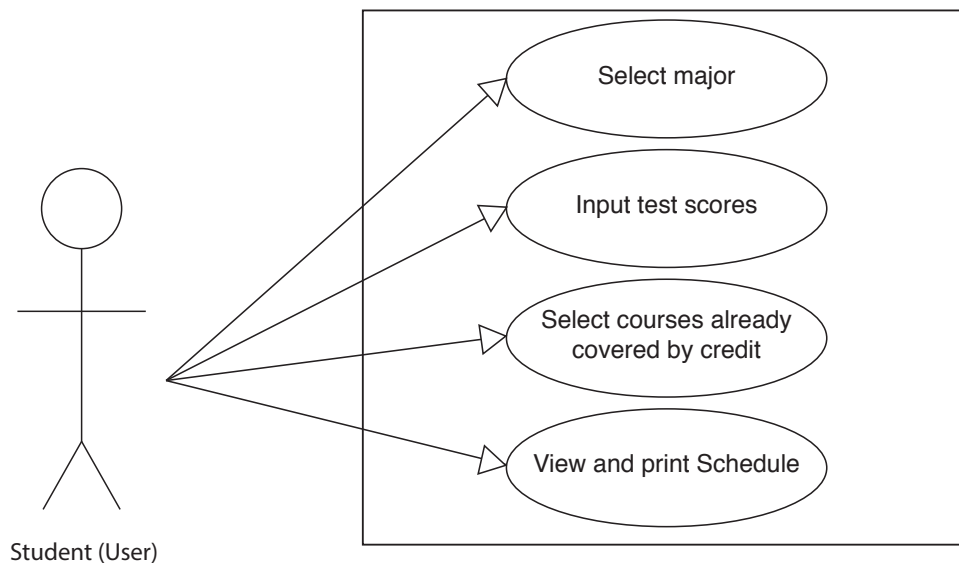


Figure 5: *Use Case diagram for our system*

5 Architectural Diagram

SCUCourses.fish will be an online application and users will be able to access our application through any of the supported web browsers. The architectural diagram for the application is shown in Figure 6. The user will log onto the internet using a web browser and navigate to SCUCourses.fish, and it will then pull all our information from the servers once. The user will then be able to use our application by interacting with the browser.

6 Technologies Used

To best fulfill the requirements detailed in Section 2 and meet the design shown in Section 3, we will use the following technologies:

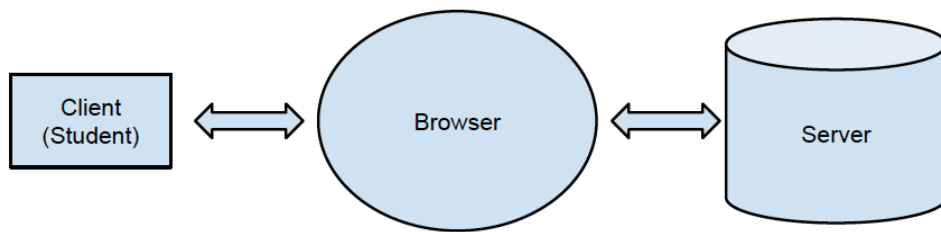


Figure 6: *Architectural Diagram for our product*

- We will be using HTML5 to create the framework for the web application. HTML stands for HyperText Markup Language and is the most common language for web content. It provides the structure upon which other code is built on.
- We will use CSS3 to format and style the content on the page. CSS stands for Cascading Style Sheets, and will allow for the stylization of the content so that the web application will look usable.
- We will use JavaScript to create the logic upon which the web application runs. JavaScript is the backbone of the project as the functions created using JavaScript will define the schedules.
- This web application will be hosted on the Santa Clara University Design Center servers.

7 Design Rationale

For this project we will be working with HTML5, CSS3, and JavaScript. We chose HTML5 because it is the standard for the web. It is easily scalable from desktop to mobile devices when paired with CSS3.

CSS3 goes hand in hand with HTML5. It allows us to create an attractive facade for the structure laid by HTML5. It is vital that this web application looks appealing as users are put off by old fashioned and ugly design.

We will also use JavaScript for the logic in our web application. This includes assigning the proper courses to the corresponding variables in HTML5. It will contain the algorithms the web application uses to generate the student's optimized schedule. We chose JavaScript because it is fairly easy to use and integrate into HTML5 and CSS3.

We chose to ask students for their AP and IB scores as opposed to what class they are coming from because we do not know which courses will cause exemption. We give the students the option to select courses from the generated courses that they believe they have credit for.

We chose to use a minimalistic user interface so that users would not be distracted by any unnecessary design. The "top to bottom" flow of the interface follows the natural progression of eye movement. The web application loads with only one question, with others appearing below as the top ones are answered, requiring the user to fill out everything so they don't accept their schedule prematurely. The limited use of bolding and color draws the users attention to the most important sections. The goal was to create something that is not only extremely functional, but to create something that was extraordinarily cool while being functional.

We chose the Web Design major as our second major because we were familiar with it. One of our group members is in the web design program, and the major is remarkably similar to the Computer Engineering major.

8 Risks Table

This project does come with risks. To mitigate the damage, we created a risk table detailing the severity, impact, probability and mitigation strategies. Probability is measured from $0 < p < 1$ with 1 being the most probable. Severity is measured from 1 to 10, with 10 being the most severe. The impact these risks have on the project is calculated by multiplying probability with risk. The risk table is shown in Table 1. Throughout the project we had to make adjustments to the risk table, including adding a new risk. One of the minor changes we made was we raised the probability of having bugs and errors from .7 to 1 because we realized it is impossible to write perfect code the first time and we had a few bugs and errors when we wrote our code. The second minor change is we raised the probability of our knowledge of web languages being a risk from .1 to .9 because we only have one Web Design major in our group and the rest of us had to do more research on how to do certain things in JavaScript. The major change we made was adding the risk of misunderstanding the requirements, because during our conversations with the customer and our demo we realized we had made assumptions.

9 Test Plan

In order to determine the success of the project, we have devised a series of tests so that the implementation of the web application goes according to plan. Below are the details of the tests.

- Major Selection
 - Test: User should be able to select Computer Science and Engineering or Web Design and Engineering
 - Result: Should build major appropriate freshman schedule
- No AP or Transfer credits
 - Test: Enter in no AP scores or scores that are too low and no transfer credits
 - Result: Should produce default freshman schedule
- Exempting math
 - Test: Enter AP scores that will exempt each level of Math
 - Result: Should move up math Curriculum by quarters equal to math courses exempt
- Exempting COEN 10 or science Sequence
 - Test: Enter AP Scores or Transfer Credits for each of the COEN 10 and Science Sequence
 - Result: Should create a space in schedule filled with CORE
- A space in two quarters in row
 - Test: Enter AP scores or Transfer credits that causes a CORE space two quarters in a row
 - Result: Should fill with C&I sequence if the sequence is not already in the schedule otherwise fill with CORE
- Print
 - Test: Try to print created schedule
 - Result: Should print created schedule

Furthermore we did extensive alpha testing on every function that we created to ensure we found bugs as we were writing code. Then after the system was built the site was given to roommates, friends and family for beta testing, This ensured that all of the system's functions worked well with each other and the system as a whole functioned properly.

The results of our latest tests are that all the functionalities work and we get the desired output for all of our test cases. During our early stages of testing we discovered minor bugs in our code, once those were fixed we began testing the functionalities on our prototype that

Risk	Consequences	Prob.	Severity	Impact	Mitigations	Mitigations 2
Misunderstanding Requirements	Have to redo previously completed work to fit requirements	0.9	10	9	Frequently ask the customer and manager questions clarifying requirements	Demo system to customer and manager before deadline to verify the requirements are met
Personnel	Work for that persons assigned section of the project falls behind	0.5	8	4	Get a significant amount of the work done ahead of time to compensate for possible losses in personnel	Keep a list of understudies so we can re-assign sections to currently active members
Time runs out	Not all desired functions of the project completed by the turn-in date	0.3	9	2.7	Prioritize features in the system and do the most essential first	Perform extensive time management throughout the course of production
Bugs and Errors	We'll have to go back into the code, find out what's causing the problems, and bugfix them	1	2	2	Do ample testing every step of the way to find bugs sooner	Run the program through debugging software
Knowledge of web languages	Progress on coding is greatly slowed	0.6	2	1.2	Model our code after some appropriate pre-existing examples of the language	Redistribute coding duties that may be difficult for one or more group members
Version control	Entire pages of code must be rewritten	0.05	5	0.25	Use GitHub which has built in Version Control	Divide the code up into multiple documents, to minimize individual risk

Table 1: *Possible risks to completing the project*

had end to end functionality. We found a couple of bugs with our Cultures & Ideas placement, it would fill any two consecutive core sections with C&I making it possible for it to appear multiple times in a schedule. The second bug is that after the C&I is placed, if the course where the first C&I was placed was added back in, it would remove the C&I 1 but it would leave C&I 2 in the schedule by itself. Our solution to these bugs was to delete the C&I placement

after every change and add it back in if there were still two consecutive core courses. What we noticed was that our solution not only fixed our bugs but also optimized the placement of the C&I courses by making sure it is placed in the earliest possible slot. Because of the extensive testing we did as we wrote the code this was the only major bug we found during the testing.

10 Development Timeline

In order to make sure we stay on track and finish the application on time we have a Gantt chart, shown in Figure 7. The Gantt chart shows the assignments that need to be completed, the person who is in charge of completing it and the deadline for that assignment.

11 Lessons Learned

While working on the project and presenting it, we learned a couple of lessons. The three most significant lessons we learned were presentation etiquette, not to use prezis, and to talk and demo to our customer frequently. When we presented our design review, we learned about presentation etiquette through mistakes that we made. We now know that we should wear appropriate dress shoes, we should not stand in front of the screen nor turn or back to the audience, and we should not make any motions with our phone in our hand. During that presentation, we presented using prezis. We now know that we should not use prezis for professional presentations because they can be distracting and there are very few opportunities to use it correctly. Lastly we learned to talk and demo to our customer frequently because when we had our first demo, our customer pointed out to us some of the assumptions we made and we were able to clear up any misunderstandings of the requirements. The lesson is that if you meet with the customer earlier and more frequently, you will make sure you understood the requirements and you have a greater chance of delivering what the customer wants.

A User Manual

The first step to running our web application is to open a web browser. Any of the latest popular browsers should work. You can either open the file locally if you have it or go to “www.scucourses.fish” to start the web application. Browser windows larger than 1700px will shift the web application into a horizontal landscape of viewing ease.

Now you are ready to create your schedule. You need to select between the Computer Engineering (COEN) major or the Web Design and Engineering major to begin to generate a schedule. There is a standard schedule that has been created at the bottom of the page. From this point on you have the ability to view the schedule, which will update automatically as you continue to answer the questions that follow.

Questions will appear after you answer the initial questions. If you took any AP and/or IB exams select the “AP” and/or “IB” buttons, respectively. Click all the applicable test buttons, and then enter your score. If you accidentally select a score for a test you did not take, simply click the test button above to remove the scores effect from the schedule. You can even remove the entire AP or IB sections by clicking on the AP or IB button.

Below this a question will ask you if you passed the calculus readiness exam, and below that you will be asked if you have transfer credit, answer the questions. If you have credit, checkboxes with possible transfer credit courses will populate below. Some of these checkboxes may override others as credit in a course further along in a series of courses implies credit in the previous ones. Clicking a section again will remove the selected transfer credits from that section and hide the checkboxes.

Once finished with the above questions and buttons, your schedule is complete. You can reset the entire schedule by clicking the “Reset” button in the top right of the the page to start over. You may switch between either majors at any time without any of your other choices being affected. You can also click the printer button in the bottom right of the page to generate a printer friendly page.

	1	2	3	4	5	6	7	8	9	Thanksgiving	10
	9/21-9/28	9/29-10/5	10/6-10/12	10/13-10/19	10/20-10/26	10/27-11/2	11/3-11/9	11/10-11/16	11/17-11/23	11/24-11/30	12/1-12/7
Requirements Gathering	9/23/2014										
Problem Statement			10/7/2014								
Design Document				10/14/2014						Legend	
Analysis and Design:							11/4/2014			Team	
Create default schedules							11/4/2014			James	
Create questionnaire							11/4/2014			Tracey	
Formulas for AP credit							11/4/2014			Nick	
Transfer credit							11/4/2014			Ben	
Build:								11/16/2014			
Interface								11/16/2014			
JavaScript default schedule								11/11/2014			
Javascript logic								11/11/2014			
JavaScript for Web major								11/11/2014			
Testing										11/30/2014	
Make corrections										11/30/2014	
Create presentation									11/15/2014		
Presentation									11/18/2014		
Demo											12/2/2014

Figure 7: GANTT chart for the development of our team project