

SCUCourses.fish: Course Schedules for Freshman

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

December 1, 2014

Contents

1	Introduction	1
2	Requirements	1
3	Conceptual Model	2
4	Use Cases	3
5	Architectural Diagram	5
6	Technologies Used	5
7	Design Rationale	5
8	Risks Table	6
9	Test Plan	6
10	Development Timeline	7

List of Figures

1	Mock up for major and test selection	2
2	Mock up for entering test scores	3
3	Mock up for potential course schedule	3
4	Use Case diagram for our system	4
5	Architectural Diagram for our product	5
6	GANTT chart for the development of our team project	8

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

The following are the requirements to fit with the needs expressed to our team by the customer. 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
- Select courses believed to be covered by transfer credits
- Generate a first-year course schedule

- 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 Conceptual 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.



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.

Did you take any of these APs per chance?

Bio	Calc AB	Calc BC	Chem	Chinese	Comparative Government & Politics	Comp Sci A	English Lang					
English Lit	Enviromental Science	European History	French Lang	German Language	Human Geography	Italian Language	Japanese Language					
Macroecon	Microecon	Music Theory	Physics B	Physics C Mechanics	Physics C Electricity & Magnetism	Psychology	Spanish Language					
<table border="1"> <tr> <td>Spanish Literature</td> <td>Statistics</td> <td>US Govt & Poli</td> <td>US History</td> <td>World History</td> </tr> </table>								Spanish Literature	Statistics	US Govt & Poli	US History	World History
Spanish Literature	Statistics	US Govt & Poli	US History	World History								

And how did you do on these?

Calculus AB 1 2 3 **4** 5

Environmental Science 1 2 3 4 **5**


Normally we'd ask for your Calculus Readiness Exam score, but your Calculus score is just too good for that.

Figure 2: *Mock up for entering test scores*

As seen in Figure 3, the user will then be shown his or her potential schedule. The user can select courses that he or she believes to be covered by previously earned credit. Potentially covered courses will appear to the side and be replaced by a new course. This course will have an X button on it in case the user wishes to undo the selection. The user will be able to print the course schedule by clicking the print icon in the corner.

Wunderbar! Here's your schedule...

If you believe you already have credit for a course, simply click it.

Fall	Winter	Spring
MATH 12	MATH 13	MATH 14
COMM 2	Culture & Ideas 1	Culture & Ideas 2
Critical Thinking & Writing 1	Critical Thinking & Writing 2	Religion, Theory & Culture 1
[filler]  COEN 10	COEN 11	COEN 12
ENGR 1		




Figure 3: *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 4 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.

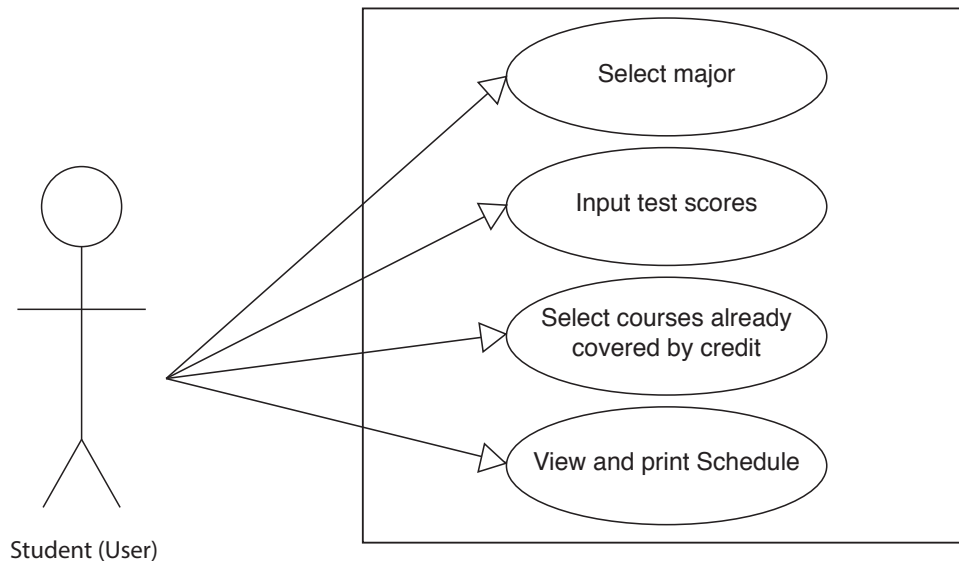


Figure 4: *Use Case diagram for our system*

Below is an in-depth description of the use case scenarios shown above.

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

5 Architectural Diagram

SCUCourses.fish will be an online application and users will be able to access our application through any of the major web browsers. The architectural diagram for the application is shown in Figure 5. The user will log onto the internet using a web browser and navigate to SCUCourses.fish, and it will then pull our information from the servers in order to provide the user with our service.

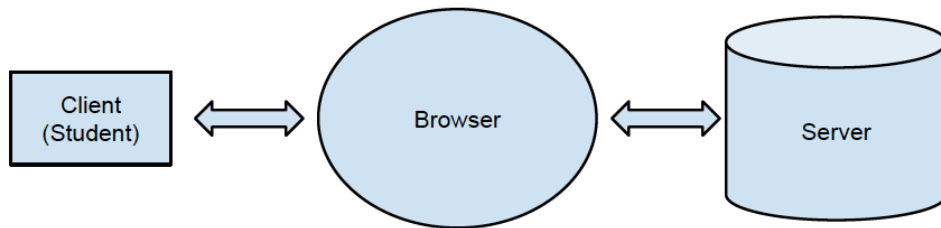


Figure 5: *Architectural Diagram for our product*

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:

- 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 reading progression. The limited use of bolding and color draws the users attention to the most important sections.

8 Risks Table

This project does come with risks. To mitigate the damage, we have 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. Table 1 is what we consider the risks for the completion of our project.

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

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 6. 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.

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	0.9	2	1.8	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*

	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 6: GANTT chart for the development of our team project