6-7-2016

Engineering Submission Portal

August Beyer
Santa Clara University

Jonathan Sofer
Santa Clara University

Joseph Villanueva
Santa Clara University

SANTA CLARA UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Date: June 3, 2016

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY

August Beyer
Jonathan Sofer
Joseph Villanueva

ENTITLED

Engineering Submission Portal

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Thesis Advisor

Department Chair

Engineering Submission Portal

by

August Beyer
Jonathan Sofer
Joseph Villanueva

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science in Computer Science and Engineering
School of Engineering
Santa Clara University

Santa Clara, California
June 7, 2016

Engineering Submission Portal

August Beyer
Jonathan Sofer
Joseph Villanueva

Department of Computer Engineering
Santa Clara University
June 7, 2016

ABSTRACT

One of the requirements for all engineering students at Santa Clara University is to complete a senior design project. The senior design project is a year-long group project (with team sizes of typically 2-4 people) in which students work with faculty advisors to develop a product, idea, or applied research in the field of their study. This work is structured through the senior design class in which students receive information about the due dates and formats of their deliverables. Currently, senior design projects are submitted on printed paper. This method is unreliable as papers may become lost or unorganized resulting in a delayed review process. Printing multiple copies of reports is also not environmentally friendly and it can be overly complicated for interdisciplinary groups with multiple advisors. As an alternative, we propose using an online web portal to handle the senior design submission process. By using a service that is tailor-made for senior design, the website will ensure that the senior design submission process is smooth, easy, and reliable for everyone.

0.1 Acknowledgements

We would like to thank the following people for their assistance and guidance in the completion of this thesis and the overall senior design project and process:

- Darren Atkinson, our Senior Design advisor
- Allia Griffin, our English advisor
- Chris Tracy, for assistance with our website's account creation

Table of Contents

List of Figures

Chapter 1

Introduction

One of the requirements for all engineering students at Santa Clara University is to complete a senior design project. The senior design project is a year-long group project (with team sizes of typically 2-4 people) in which students work with faculty advisors to develop a product, idea, or applied research in the field of their study. This work is structured through the senior design class in which students receive information about the due dates and formats of their deliverables. However, unlike most classes in which students work on the same assignments and turn in their work to the same professor, the senior design class has different expectations for different groups. Some departments choose to have the advisors collect the students deliverables while other departments have the work collected in their senior design class. Deadlines for specific deliverables may be imposed by either the advisor, the teacher, or even the department. Additionally, groups may be interdepartmental or even interdisciplinary depending on the scope and scale of the project.

The current solution to submitting senior design projects is to simply hand submit your printed deliverables. This method is unreliable as papers may become lost or unorganized resulting in a delayed review process. Printing multiple copies of reports is also not environmentally friendly and it can be overly complicated for interdisciplinary groups with multiple advisors. Another alternative is Google Drive[2] or Dropbox[5], however, neither of these solutions results in an appropriately organized project allowing for easy submission. A third existing solution, Camino[1], is also insufficient in many ways. Camino is a website where students can submit work to their professors. However, for the purposes of submission for senior design it does not have the functionality required. Camino, while an excellent all-purpose tool, cannot restrict file formats, is bloated with unnecessary features, and the group mechanism does not allow for group submission, a critical requirement for senior design projects. Most importantly, though, neither existing online solutions nor traditional paper formats ensure the secure and reliable retrieval of senior design projects several years after their submission.

Our solution is to use an online web portal to handle the senior design submission process. By using a

service that is tailor-made for senior design, the website will ensure that the senior design submission process is smooth, easy, and reliable for everyone. Students will be able to upload their submissions so that their advisor(s) can easily access the submissions digitally. Advisors will have the ability to restrict file formats, ensuring that students turn in appropriate deliverables. The system also allows advisors easy access to projects from previous years without having to search through an endless assortment of papers. By implementing a cascading set of requirements, our system will adapt to the policies of any specific department or advisor with ease. We are positive that our submission portal will provide a more encompassing and focused experience than other potential solutions.

Chapter 2

Requirements and Uses

2.1 Requirements

There are requirements that our system must fulfill and constraints that it must abide by. These are outlined in the following sections. Functional requirements are requirements which involve what the system will be able to perform. Nonfunctional Requirements are requirements which dictate how the system will perform. Design Constraints are restrictions placed upon our system that must be held to be usable. If a requirement is critical, it will be prioritized first and will be a core functionality. If a requirement is recommended, while it may not be a core functionality, it will be completed before the first release given enough resources. Finally, if a requirement is suggested, it will be completed given there are enough resources for its completion. These are non-essential and therefore may not be a part of the system on the core release.

2.1.1 Functional Requirements

These are the functional requirements our system will adhere to. categorized into critical, recommended and suggested priorities.

Critical Functional Requirements

These are critical functional requirements detailing first priority abilities our system will have.

- The system will provide a method for users to log-in.
- The system will allow users to upload / download specified deliverables.
- The system will control user access based on whether the user is a student, faculty advisor, or system administrator.
- The system will give advisors the ability to create and associate students with groups.

- The system will provide an easy method of archiving files (e.g., the ability to zip up all deliverables from the 2016-2017 year).

Recommended Functional Requirements

These are recommended functional requirements detailing second priority abilities our system will have.

- The system will enforce file formats as specified by departments and advisors
- The system will provide superusers with a web-interface to add, modify, or delete projects / requirements.
- The system will cache user's log-in information in the browser with cookies so users do not have to constantly authenticate.

Suggested Functional Requirements

These are recommended functional requirements detailing third priority abilities our system will have.

- The system will have a notifications panel to quickly update users of submissions / feedback.
- The system will allow advisors to assign grades to groups (and potentially individuals).

2.1.2 Nonfunctional Requirements

These are the nonfunctional requirements our system will adhere to, categorized into critical, recommended and suggested priorities

Critical Nonfunctional Requirements

These are critical nonfunctional requirements detailing first priority abilities our system will have.

- The system will be modular, allowing advisors and department heads to tailor the specification of their deliverables to their students.
- The system will be secure, preventing unauthorized users from viewing, modifying, or deleting files on purpose or by accident.
- The system will be robust, only making use of libraries and frameworks that have guaranteed long-term support.
- The system will be aesthetically pleasing, user-friendly, and intuitive.

Recommended Nonfunctional Requirements

These are recommended nonfunctional requirements detailing second priority abilities our system will have.

- The website will be accessible from mobile devices (phone, tablet, etc.)

Suggested Nonfunctional Requirements

These are suggested nonfunctional requirements detailing third priority abilities our system will have.

- The system will include support for older browsers (e.g., IE 7).

Design Constraints

These are design constraints that our system must abide by to be usable.

- The system must be web based.
- The system must run on the SCU Engineering Computing Center server.

2.2 Use Cases

Figure 2.1 shows the use case diagram that describes our system. In this diagram, it is shown that the advisors have the ability to add students to groups, grade deliverables, download deliverables, add assignments and view assignments. This is a much larger variety of activities compared to students who are limited to viewing assignments, including their grades, and uploading deliverables.

2.2.1 Actions Explained

Add Students to Groups

1. Actor: Advisors
2. Goal: Organize students into their Senior Design Groups to allow them to assign deliverables
3. Preconditions: User has logged in as an Advisor and has been asked to oversee a senior design group by its members
4. Postconditions: User has created a group and is able to give them assignments, grade them and in the future download their deliverables
5. Steps:
   (a) Navigate to a page allowing adding students to groups

(b) Create a blank new group

(c) Enter each student's information individually into the group

(d) View the group you have created

(e) Confirm that it is correct

6. Exceptions: The group is incorrectly entered

Grade Deliverables

1. Actor: Advisors
2. Goal: Grade the deliverables that students have uploaded for assignments
3. Preconditions: User has logged in as an Advisor, they have already given their students an assignment through the system and their students have uploaded their response
4. Postconditions: User has graded the assignment that is viewable to both the advisors and the students
5. Steps:
   (a) Navigate to a page allowing grading
   (b) Select the group you want to grade
   (c) Select the assignment to grade
   (d) Assign the assignment a grade
6. Exceptions: The students have not yet uploaded an assignemnt

Download deliverables

1. Actor: Advisors
2. Goal: Download students' previously uploaded assignments
3. Preconditions: User has logged in as an Advisor and the deliverable has already been uploaded by the students
4. Postconditions: User has downloaded all of the work that a group of students have done for offline use
5. Steps:
   (a) Navigate to a page allowing downloading of previous assignments

(b) Select what year you want to download from

(c) Select what discipline you want to download from

(d) Select a specific group you want to download from

(e) Confirm all selections are correct

(f) Download deliverables

6. Exceptions: The specifications are incorrectly entered

Add Assignments

1. Actor: Advisors
2. Goal: Give the students the information they require to create a deliverable
3. Preconditions: User has logged in as an Advisor, has created a group and has created the specifications for a deliverable
4. Postconditions: User has created an assignment for the group that both students and advisors can view
5. Steps:

   (a) Navigate to a page allowing adding assignments to groups

   (b) Select group to give an assignment to

   (c) List or upload specifications

6. Exceptions: The group selected is incorrect

View Assignments

1. Actor: Advisors or Students
2. Goal: View assignments and grades given to those assignments
3. Preconditions: User has logged in to the system, has selected the specific group if advisor and the student has been given assignments
4. Postconditions: User has viewed their assignments
5. Steps:

   (a) Navigate to a page allowing viewing of the assignments

(b) Only for advisors: Select groups

(c) View assignments

6. Exceptions: The students have not been assigned any work yet

Upload Deliverables

1. Actor: Students
2. Goal: Upload their deliverables in response to the given assignments
3. Preconditions: User has logged in as a student and has completed their deliverable in the correct format.
4. Postconditions: User has uploaded their assignment and it is now ready to be either graded or downloaded
5. Steps:

   (a) Navigate to a page allowing uploading of deliverables

   (b) Select assignment to upload a deliverable for

   (c) Upload the assignment

6. Exceptions: The uploaded assignment is not of the correct file format

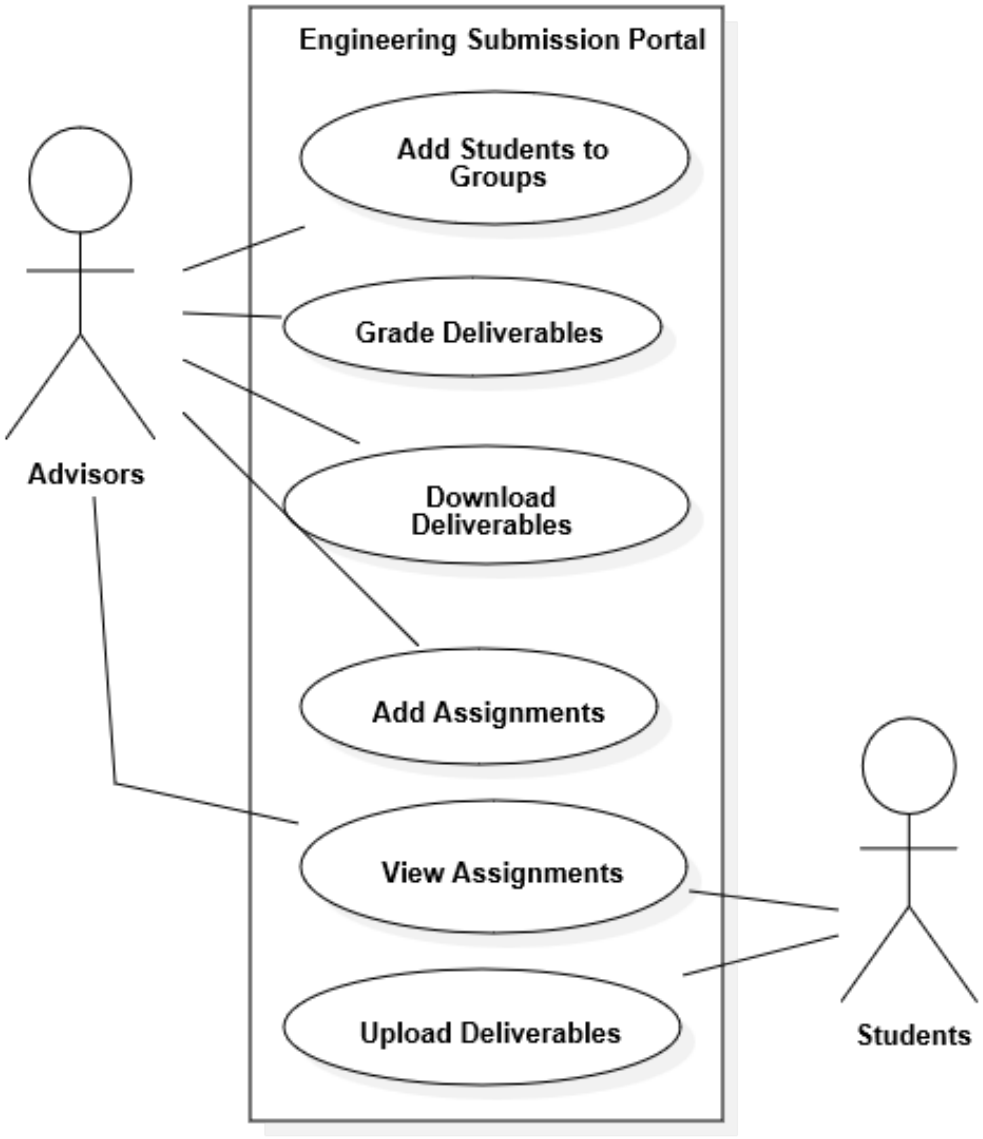

Figure 2.1: Use Case

2.3 Activity Diagrams

On the following three pages are three activity diagrams for the major actions that our system will allow students or advisors to make. Figure 2.2 refers to the process of adding students to a group. The way that this is completed is by first creating a new group, then entering the students' information individually; this is an action that can only be used by advisors. Figure 2.3 refers to the process of downloading previous assignments. This is accomplished by choosing the specific group to download from through filtering. The filter first asks for which year the advisor wants to download from, then the specific discipline and finally the specific group. All of the options will also allow for downloading all deliverables immediately by specifing "All years", "All disciplines" or "All groups" to download from. This is an action only to be completed by advisors. Finally, figure 2.4 refers to the process by which students will upload their assignments to the system. This is done having the students first select an assignment to upload for, then choosing their deliverable to upload and finally confirming their submission.

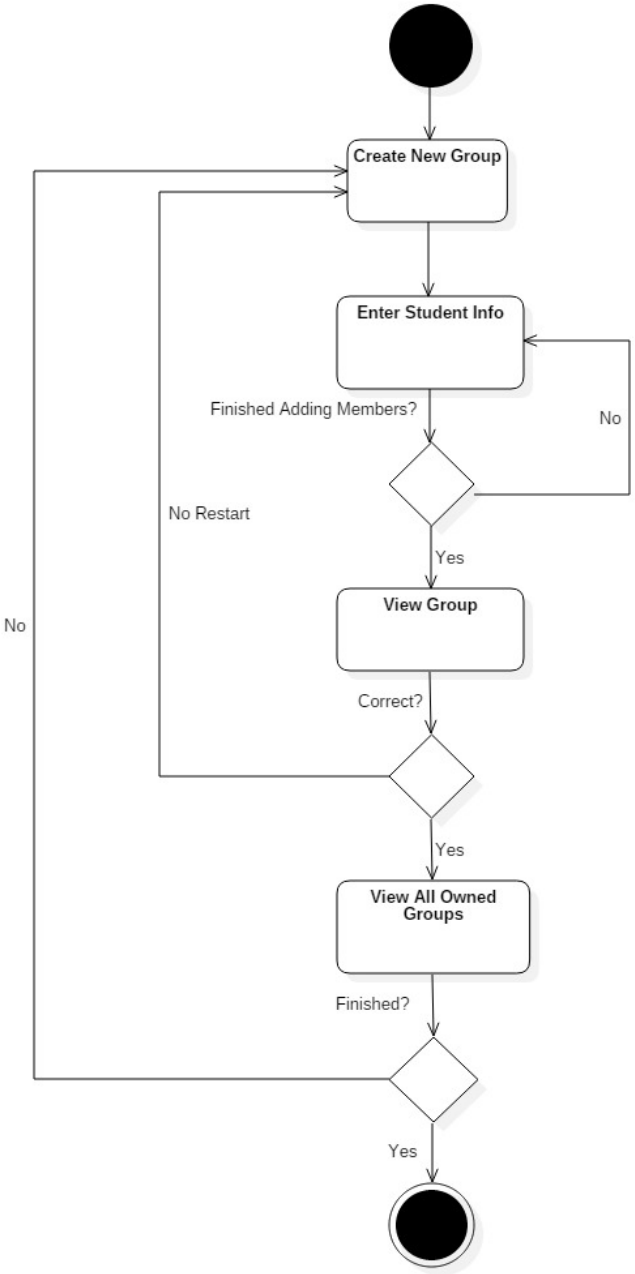

Figure 2.2: Add Students to Group

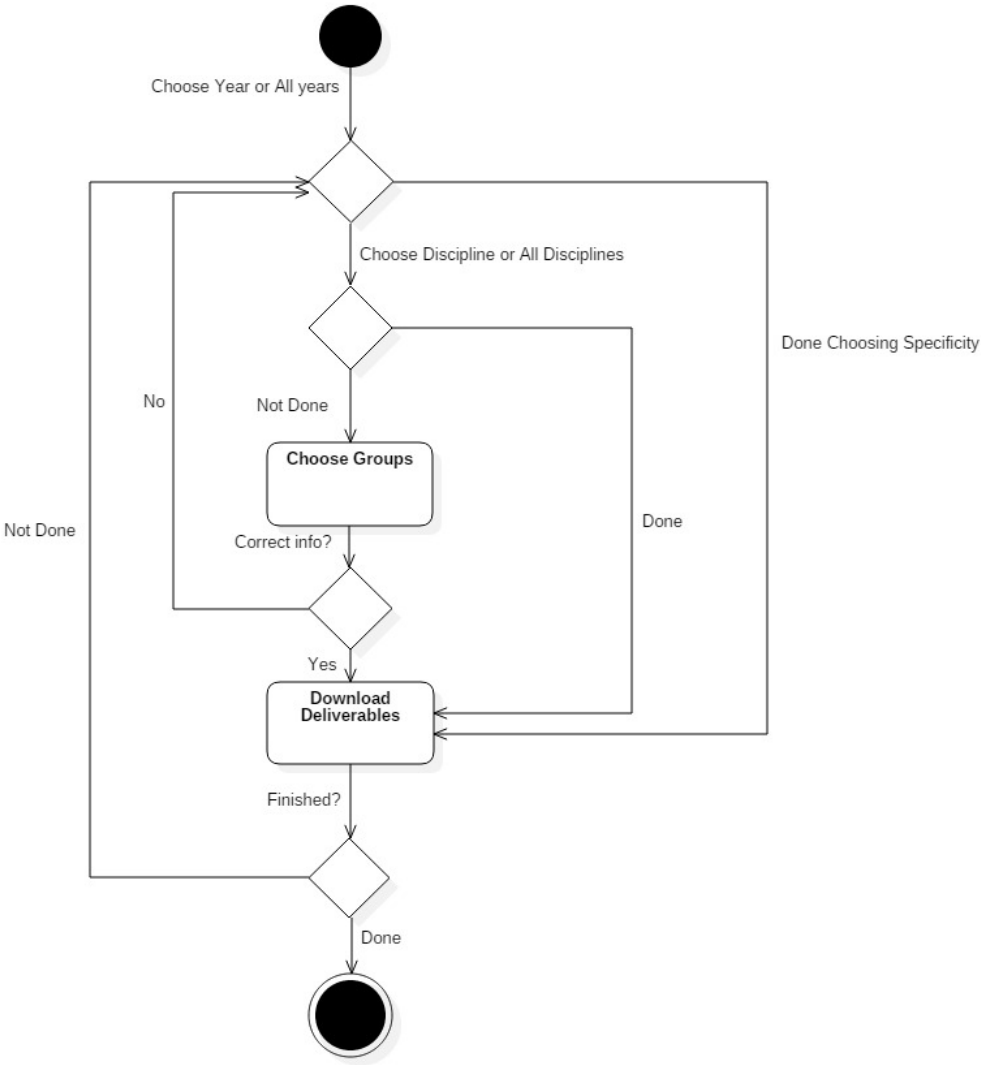

Figure 2.3: Download Previous Assignments

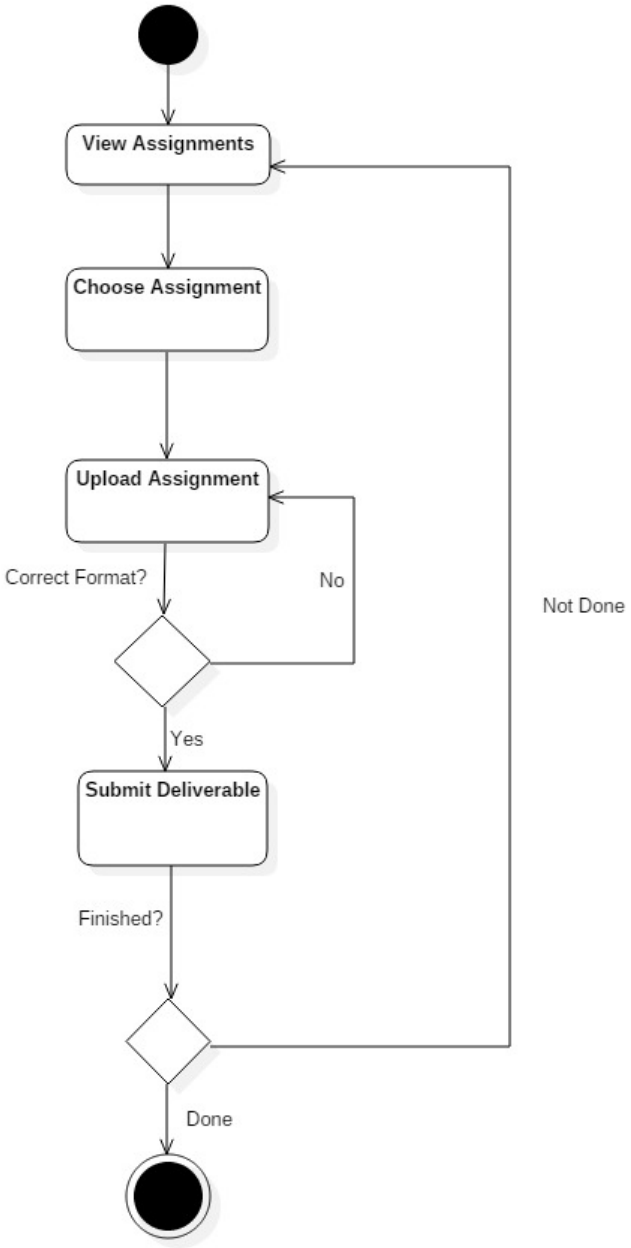

Figure 2.4: Submit Assignments to System

Chapter 3

Product Design

Included below are images of the Engineering Submission Portal's three main pages: the login page, the student's landing page, and the advisor's landing page. Figure 3.1 shows how a student or advisor will login with their Novell ID and password. Figure 3.2 shows what the general layout of the student's landing page will look like. This is the page that the student is greeted with after successfully logging in to the system. From this page, the student will be able to see recently submitted deliverables, turn in a completed deliverable, or navigate elsewhere on the system. Figure 3.3 shows what the advisor's landing page will look like. This is the page that the advisor is greeted with after successfully logging in to the system. From here, the advisor can see recently submitted deliverables, or navigate elsewhere on the system.

Also included in this section is our entity relationship diagram, which is figure 3.4. This diagram shows how each of the entities within our system are related.

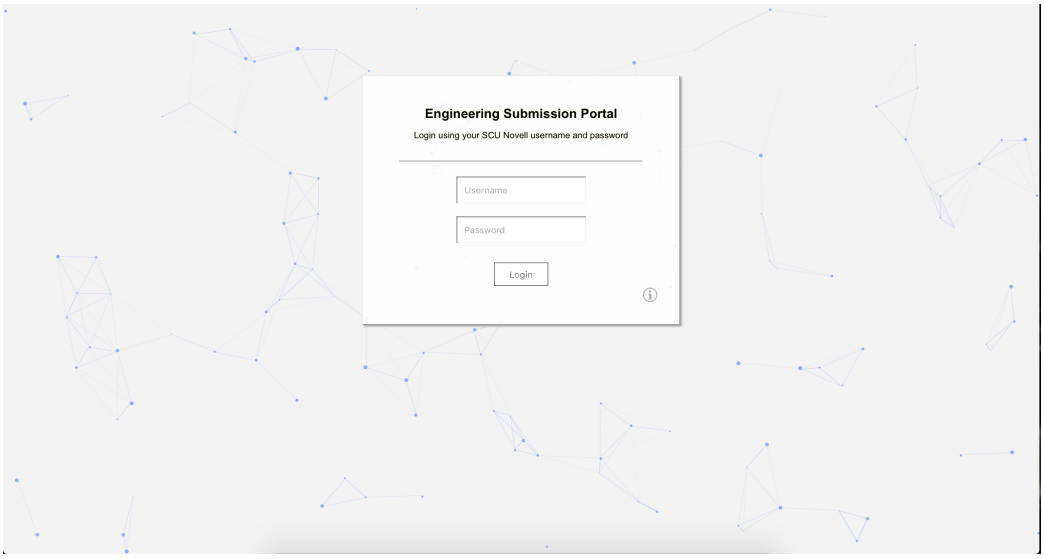

Figure 3.1: The Login Page

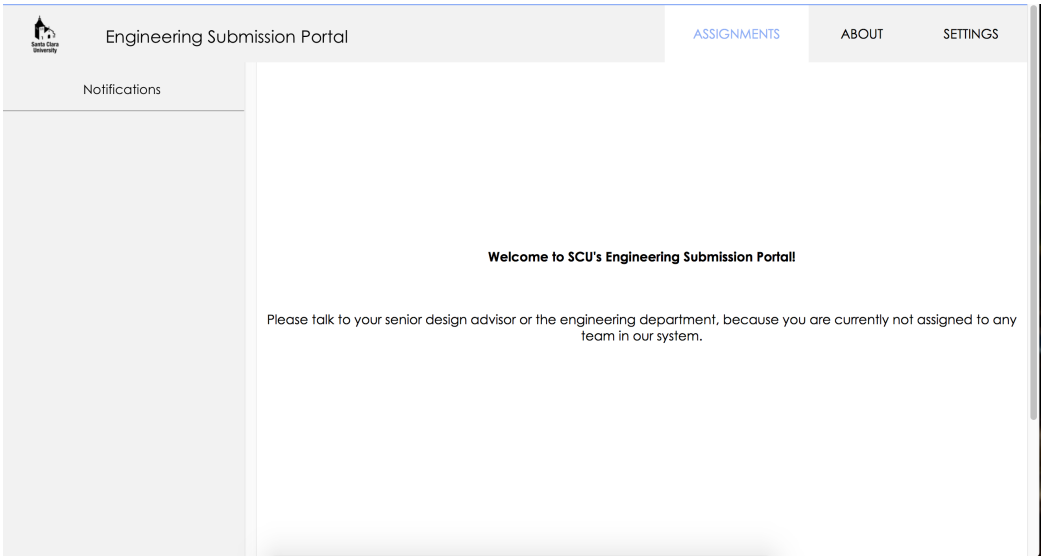

Figure 3.2: The Student's Landing Page

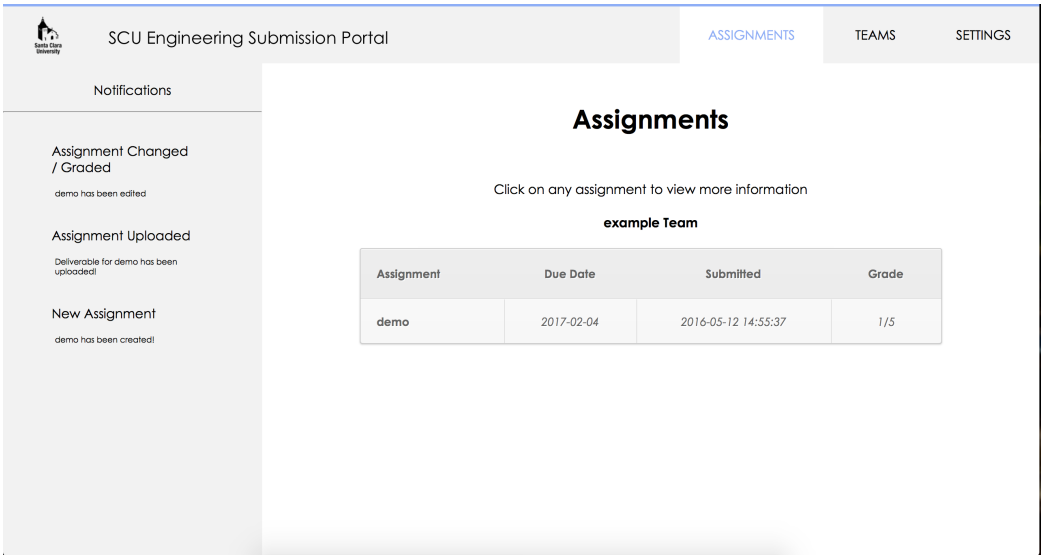

Figure 3.3: The Advisor's Landing Page

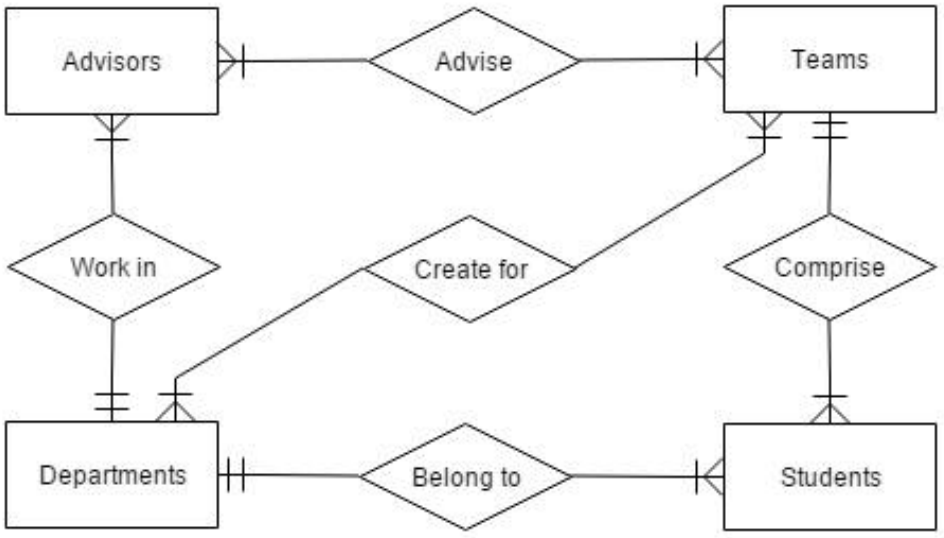

Figure 3.4: Entity Relationship Diagram

Chapter 4

Technologies Used

The following are the different languages, frameworks and systems that our system will use. As this system will be an online portal, it will require a website with a graphical frontend and a data structure to store data.

- HTML - HyperText Markup Language is used to create a layout for a website
- CSS - Cascading Style Sheets are used to more easily style HTML
- Javascript - An interpreted language used to allow a webpage to be interactive
- jQuery [3] - A language which allows simple smooth animations on the frontend application
- PHP [4] - Scripting language allowing communication between the HTML, CSS, Javascript and jQuery ”frontend” and the database ”backend”
- MySQL - A relational database system that will serve as the ”backend” database

Chapter 5

Architectural Design

The engineering submission portal will be based on a client-server architecture. Users will use the web browser of their choice to connect to our server over HTTP or HTTPS (for logging in). To connect to our sever clients must first authenticate themselves using their SCU Novell login information. This information will be transferred over HTTPS to SCU's LDAP server. We will not be storing the client's password in our own database.

Once a user has been authenticated, our server will interface with our own database to fetch information unique to that user. For an advisor, this may include the complete list of his senior design groups, deliverables for each of those groups, and notifications that have accumulated since his or her last login. For a student, this may include other users in his or her group, a list of outstanding assignments, and a similar list of notifications. A PHP interpreter will be running on our server to retrieve this unique data, and, through a RESTful API will return this data to the client. The user's web browser will be running JavaScript that will use this data to dynamically populate a unique web page.

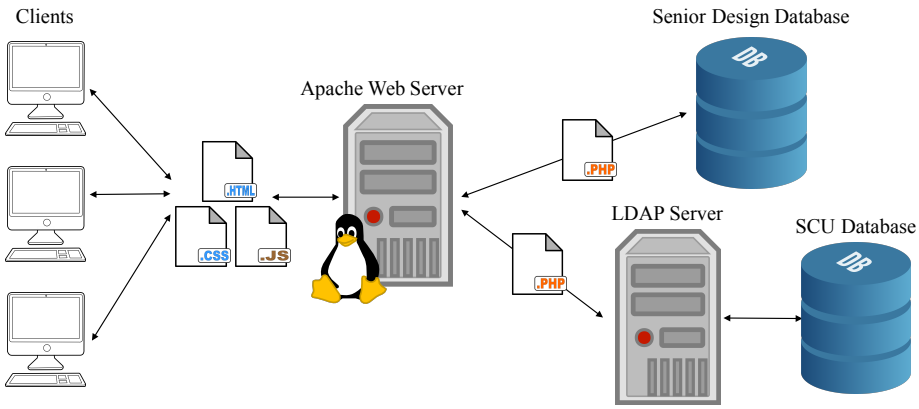

Figure 5.1: Architectural Model

Chapter 6

Design Rationale

6.1 Essential Design Languages

The following are the three languages which are essential for the creation of web content. This means that they are supported on all browsers without any additional set up and they all serve distinct functions which cannot be completed by any other language.

- HTML
  - It allows content to be displayed graphically for the system
  - This is one of three essential languages needed to create a web application as it is the only way to display information on the front end
- CSS
  - It is an excellent to organize HTML styles compared to inline style changes
  - It is the second essential language needed to create a web application as this is the only way to make theme changes to the whole website
- Javascript
  - This is the third essential language needed to create a web application as this is this serves to create an interactive system

6.2 Non-essential Frameworks, Languages and Systems

The following languages, frameworks and systems are used in the system. Although there may be alternatives that would be able to be used, the reasoning for using each are outlined below.

- JQuery

  - This will be used to allow for simple smooth animations on the frontend
  - This specific language will be used because it is easy to implement alongside Javascript

- PHP
  - This will be used to communicate between the database and frontend
  - This specific language will be used because it is most familiar among the group and it is widely documented
  - This language also is easily integrated with MySQL, the database that we will use to store the data of the system

- MySQL
  - This will be used as the database which will store the data that is entered through the system
  - This will be used over other systems because it will be the most maintainable for future generations to expand upon

6.3 Specific Design Choices

The following are the important design choices we have made that affect the system and the way the privileges or actions that the categories of the users of the system can make.

- Only Advisors are able to create and edit groups
  - This allows for easier organization between students and the advisor
  - This eliminates the need for an internal student structure and the need for a student to be a leader to create the group and manage it

- Advisors have centralized notifications of deliverables that their students have uploaded
  - This saves them time that they would otherwise need to put into checking each of their groups individually
  - This eliminates the need to continually email the advisors that students have uploaded their work

Chapter 7

Testing Plan

We will primarily be relying on two types of testing to test our system. The first type of testing will be unit testing. Unit tests are written concurrently with (or often before) the implementation of an object, function, or module to see if the desired outputs are being produced for given inputs. Unit tests will be written with the complete knowledge of implementation and will often reside alongside the source code.

The second type of testing will be black-box testing. Black-box testing will be done at a higher level of abstraction without any knowledge of the underlying source code. Black-box testing can occur at any level of software testing, but for the purposes of this project it will mostly refer to testing done on the entire system.

7.1 Unit Testing

Unit tests are typically written in the same language as the object, function, or module they are testing. The two main languages we will be testing are JavaScript on the front-end and PHP on the back-end. Several frameworks exist to write unit tests in JavaScript and PHP, but we chose the following two due to their widespread support, ease of implementation, and popularity:

- QUnit (JavaScript)
- PHPUnit (PHP)

Once all unit tests past we can integrate the tested object, function, or module into the rest of the code base and move on to black-box testing.

7.2 Black-Box Testing

Black-box testing will mostly consist of interacting with our system in the same way a typical user might. For example, we may try to submit an assignment as a student, or create an assignment as an advisor. However, in addition to interacting with our system in a ”smart” way, we will also be testing edge cases and atypical

scenarios to make sure our system does not break. Such edge cases may consist of a student trying to submit an assignment that wasn't yet uploaded, or having an advisor trying to add a student to two groups. Testing will also be cross-platform to account for different operating systems and web browsers. We plan to test on the two major desktop operating systems:

- Windows 7+
- Mac OS 10.6+

We also plan to test on the four major desktop web browsers:

- Chrome
- Firefox
- Safari
- Internet Explorer (Edge)

Chapter 8

Risk Analysis

Figure 8.1 contains a list of the risks that are relevant to this project. The chart also shows how impactful each of the risks are, as well as some mitigation strategies for these risks.

Name of risk	Consequence of Risk	Probability of Risk	Severity of Risk	Impact	Mitigation Strategy
Design does not match requirements, or does not satisfy customers	Design has to be changed in order to meet requirements or satisfy customers.	0.7	8	5.6	Talk to all departments of engineering frequently and show them prototypes to validate our design. Make our system more adaptable to changes.
Inexperience with certain technologies that lead to more development issues than expected	Wasting more time than necessary getting a certain aspect of the system to function properly causing us to reevaluate our development timeline.	0.6	6	3.6	Be aware of our proficiency with all technologies when designing the development timeline, and add extra time accordingly. Make use of online resources such as tutorials and technical forums, and build prototypes along the way. Be aware of which features of the system can be cut if necessary.
Issues related to team dynamics	If there is a class of ideas, we might not be able to settle on a single implementation or design. Arguments could lead to a lack of productivity.	0.5	6	3.0	Assign team leads, so that one person has final say in one specific aspect of the project (have one design lead, one customer relations lead, and one lead developer). Settle trivial matters with a simple vote.
Time constraints and or feature creep	Time is a limited resource for the project. If we fall too far behind, or if we keep adding features/functionality to the system, we might not be able to finish.	0.4	7	2.8	Because it can be hard to make up lost time, we should pad our cant chart with extra time. If we think a task might take 8 days, we can assign it 9 or 10 just to be safe. We should also be aware of which features are critical, which are recommended, and which features can be cut for the sake of time.
Compatibility issues with modern operating systems and web browsers	Our system might not function on all browsers properly, causing problems for our users.	0.5	4	2.0	Adequate testing on multiple platforms throughout product development should ensure a working system for all users. Use libraries that are functional across all browsers.

Figure 8.1: Risk Analysis

Chapter 9

Development Timeline

Figure 9.1 outlines the responsibilities for each group member. August Beyer (highlighted in red) is primarily responsible for creating our frontend user interface. Joseph Villanueva (highlighted in green) and Jonathan Sofer (highlighted in blue) are primarily responsible for managing the backend database, user authentication, and security. The development timeline starts on Monday, September 21st and continues through Spring 2016.

Figure 9.1: Development Timeline

Chapter 10

Societal Issues

The following will focus on the impact and issues surrounding the Engineering Submission Portal. It is important to recognize and consider the societal ramifications of our project.

10.1 Ethical

The Engineering Submission Portal contains a lot of information about SCU students and faculty. Our primary social responsibility, then, is to protect the privacy of our users to the best of our technical ability. Furthermore, our website will be connecting to the SCU student database, so we must also keep in mind that any vulnerability in our code could compromise information not necessarily contained with our website. We therefore have an ethical responsibility to only use the most up-to-date security protocols for information transfer and authentication, inform users immediately in the case of a database compromise, and detect and blacklist users that attempt to gain unauthorized access.

10.2 Social and Political

The current senior design process makes it difficult for interdisciplinary teams to effectively communicate with their advisor(s) and submit their deliverables in a timely manner. We are confident that the Engineering Submission Portal will greatly reduce confusion and technical barriers for teams that have students and adviors from a wide array of different departments. The Engineering Submission Portal will also give advisors a platform to deliberate on assignments and grades.

10.3 Economic, Health and Safety, and Manufacturability

The economic impact of The Engineering Submission Portal is negligible; the cost of upkeep is only the server space and processing power required. Also, there are no direct health and safety concerns associated with this project since our code is meant for a relatively safe process - serving web pages. We don't necessarily

have to have the same low-level safeguards in place that would be necessary when writing code for airplane controllers, or other potentially high-risk projects. In addition, there is no manufacturing process associated with our project.

10.4 Sustainability and Environmental Impact

The traditional senior design submission process includes printing paper copies of multiple assignments. Because the Engineering Submission Portal allows student teams to submit their deliverables electronically, we can significantly reduce the amount of paper used by the engineering department.

Our system runs on the Santa Clara University Engineering Center servers. This means that we also have to consider the ethics surrounding the servers energy usage. While our system was designed to be light weight and manageable, it still needs electricity to work, and if the electricity is gathered in an environmentally unfriendly manner then our project also becomes environmentally unfriendly. We are relying on Santa Clara University to ensure that the electricity used on our campus is sourced with carbon neutrality in mind.

10.5 Usability

It is very important for websites to be user friendly. A user who is unable to easily navigate a web page will quickly become frustrated and leave the website. Even if an advisor requires his or her team to use the Engineering Submission Portal, we want everyone's experience to be that of wanting to use the website, not feeling forced to. To make our website user friendly we studied a lot of popular websites in terms of design and functionality and learned what made them intuitive. We then applied these principles to the design of the Engineering Submission Portal.

10.6 Lifelong Learning

Through this project, we were face with new problems and in that adversity new solutions needed to be created. For example, we needed to find the most efficient and clear way of creating our MySQL relational database and to do this we researched different styles of creating a backend relational database system and implemented and tailored a method to fit our needs. This process of research and finding new solutions to continue with the project are aspects of lifelong learning. We will be using this process to continue learning and becoming better engineers in our field.

10.7 Compassion

We are relying on Santa Clara University, a Jesuit learning institution with a history of educating the whole person, to educate our engineers to live their lives compassionately; it is important that engineers are not only technically skilled but are also applying their skills with a robust compassion. The Engineering Submission Portal allows students to focus on their work without letting the submission process hinder their development. Students should be focusing on building their technical skills and engineering with compassion, and not worrying about whether or not someone on their team dropped their paper documents in the correct office. Our solution abstracts away complexities surrounding the senior design process so that the student teams can develop compassionately.

Chapter 11

Conclusion

We hope that our project serves useful for Senior Design students in the years to come, as well as for the engineering department as a whole. We are confident that the online submission process will make it easier for both students to submit deliverables, and for advisors to manage assignments. Storing the documents on the SCU DC servers will also make it easier to collect archive information on all submitted senior design projects - currently only the final thesis is available in an online format. Moving the entire system online will also reduce SCU's overall carbon footprint. Lastly, we are making our code available publicly on GitHub, so any SCU students who wish to improve the website may feel empowered to do so.

11.1 Lessons Learned

11.1.1 Git Commits

When making a lot of changes it's easy to give vague descriptions for potentially meaningful git commits. This hinders development as it's difficult to go through the commit history and figure out where/why certain changes were made.

11.1.2 Write Modular Code

During development it is tempting to start writing code before high-level design decisions are made, but this generally leads to less then optimal code. Such code tends to be repeated in many locations, which can make future development more difficult.

11.2 Future Development

11.2.1 Mobile Friendly

A mobile app for iOS or Android device could be created (or ported through PhoneGap) to add support for mobile platforms. However, because students are less likely to submit and store engineering files on

their phones, we did not design our website with a mobile first philosophy. We believe that some students / advisors may like to check grades or receive notifications through their phone, so creating a mobile app with this limited functionality may enhance users' experience.

11.2.2 Expand Functional Converge Beyond Senior Design

While meeting with engineering staff it became apparent that some professors would like to use our system for their ordinary classes as well as senior design. There are some functional features we could add that would greatly improve usability for professors wishing to do so - for example, individual grades, teams that last only a quarter, and different sections for senior design in classes. Currently, professors could theoretically create a team with all students in their class should they wish to use the system outside of senior design. These features, however, would require a good amount of time and restructuring of our code base, which is why we did not focus on them during our development life cycle.

Bibliography

[1] ”About Us — Canvas.” Canvas - Instructure. Web.

[2] ”Google Drive.” Google Drive - Cloud Storage and File Backup for Photos, Docs and More. Web. 01 May 2016.

[3] ”JQuery API Documentation.” JQuery API Documentation. Web. 1 May 2016.

[4] ”PHP: Hypertext Preprocessor Manual.” PHP: Hypertext Preprocessor. Web. 01 May 2016.

[5] ”Secure File Sharing and Cloud Storage — Dropbox Business.” Dropbox. Web. 01 May 2016.