2014

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

A WEB APPLICATION FOR CAPSTONE MANAGEMENT FOR THE MSCSIS PROGRAM

James Grooms

A Capstone Project Submitted to the

University of North Carolina Wilmington in Partial Fulfillment

of the Requirements for the Degree of

Master of Science in Computer Science & Information Systems

Department of Computer Science

Department of Information Systems and Operations Management

University of North Carolina Wilmington

2014

Approved by

Advisory Committee

_____ _____

Dr. Devon Simmonds Dr. Jeff Cummings

_____

Dr. Douglas Kline, Chair

Abstract

The purpose of this project is to provide the MSCSIS program a Capstone Management System (CMS) for the students, faculty, administrators, and program director to automate the capstone thesis or project process. The CMS is a web application that captures data and automates the workflow of a student's capstone thesis or project. The CMS is designed to automate the process as much as possible and capture useful data for reporting to the program director. The CMS uses many of the latest web technologies available today including C# with the ASP.net 4.5 framework, the latest Bootstrap user interface framework, and a single page application (SPA) design for each role in the system. The CMS is compatible with all modern web browsers and is designed for use on mobile devices such as tablets and phones. The CMS enables students, faculty, and the program director to complete the capstone process using an integrated web application. The CMS simplifies the current manual approval process saving time for all stakeholders. The CMS also provides high level reporting capabilities for high level decision making.

Table of Contents

1. Introduction

In order to complete the UNCW Masters of Science in Computer Science and Information Systems (MSCSIS) program each student must complete either a capstone thesis research paper or a capstone project. The program is designed so that each capstone is managed by a thesis advisory committee comprised of a chairperson and members. The process for formally declaring a capstone, forming a committee, scheduling and presenting a proposal, and scheduling and defending a capstone is manual and cumbersome to the students, faculty, and program director. The CMS is a web application that automates the existing paper-based process as much as can be done and provides a database for reporting and monitoring the progress of students in the program.

Since the inception of the MSCSIS program the director has been required to keep a record of what students are in the program, what capstones the students are completing, and what status of each student is in relation to program completion. Maintenance of this data was scattered across documents, spreadsheets, and email and was impossible to report on or really control. The CMS provides the central repository of program data regarding students, capstones, capstone committees, proposals, and defenses. The system also replaces the need for the manual routing of paper forms necessary to keep the students and committee members accountable to their capstone parameters.

In this paper, I specify the case for developing the CMS and the benefits provided that were previously unavailable using the existing system. The decision to use the rapid application development approach is discussed. Deciding which web development technologies to use and

why is discussed in section 4. Sections 5 through 9 consist of detailed system analysis and system design including the use cases, user interface designs, and test plan. Sections 10 and 11 detail the results of the project execution, future works, and findings. The remaining sections explain the lessons learned, skills acquired, and the appendices.

The goal of the CMS is to semi-automate the current capstone process, provide a repository of data for the program director to better manage the MSCSIS program, and clarify to the users how to complete the capstone and ultimately earn the Masters of Science in Computer Science and Information Systems from UNCW.

2. Motivation

2.1. Problem Statement

Each student must complete either a formal thesis research paper or a thesis project under the guidance of a capstone committee. Currently, the processes and activities involved in completing a thesis paper or project include forming a capstone committee, presenting a capstone proposal, and defending the thesis. There is not currently a system for the program director or graduate studies office that provides information around this critical process. Since each student only goes through this process once, they are unfamiliar with the process and must be guided through it, placing a burden on advisors. Each step in the process requires the student and faculty to fill out, sign, and track a paper form.

The problem is that students, faculty, and the graduate programs office do not have a means for managing the process. The program director has to take time each semester to organize a

capstone orientation event that explains the process and schedule. Students are not aware of how to form capstone committees, schedule and propose a capstone, or schedule the final defense.

The lack of knowledge of the process outside the program director and graduate office leads to time spent answering the same questions from students and faculty over and over again. The orientation session is helpful, but there is no way to ensure that all students in the program are educated about the process and are provided with a readily available source of information regarding their capstone progress.

The CMS addresses the primary problem by serving as the single source for students and faculty to reference how to complete their capstone project or thesis and have a timeframe available to assist in planning each step.

2.2. Key Benefits

- Reduce the time required to complete the paperwork required to complete the capstone or thesis
- Improve faculty and student understanding of the capstone workflow
- Automate the manual processes of gaining approvals for a capstone committee, scheduling proposals, and, scheduling defenses using a web application and email notification

- Save time and increase efficiency by eliminating 4 of the 6 paper forms required to complete a capstone
- Increase faculty/student communication with email notifications to facilitate the processes
- Provide reports to monitor capstone progress
- Record the announcements for capstone proposal and defense

3. Platform

3.1. Review of Technologies Used.

Hybrid Single Page Application design (SPA)

The CMS uses the principles of single page application, meaning the user experience is streamlined so that all the functionality is contained within a single web page. Because there are three types of users in the CMS there three single page applications divided by role. The division of application functionality by role makes the system a hybrid single page application as opposed to a true single page application where all functionality for all the users is managed and executed within a single web page. The SPA strategy is used to promote organized, easily maintainable code, to provide a streamlined user interface, use asynchronous transactions to increase system performance, and leverage the advantages of AJAX.

Developers will be able to easily maintain or extend the functionality of the CMS because of the use of a single web page per user role. If a software bug is found or new features identified for users in the student role, then all the changes need only be made in the

student.aspx page and related code file. Using a SPA methodology promotes the good development practices of modularizing software as well as creating clear separation of concerns within the functionality. The intent is to provide clearly written software that is not spread out over the project so the code is grouped together and easily traversed and readable.

The SPA design of the CMS greatly assists the users in training and learning the system since there is only essentially one interface to learn. The student page is the most complex page with an input for the capstone title, a form for requesting a chairperson, a form for requesting a committee, and a form for entering the information regarding proposal and defense. The director and system administrator have only to learn a single grid to manage their roles in the system. Committee members do not have an interface; rather, they interact with the system only through email notification links.

ASP.Net 4.5 with C#

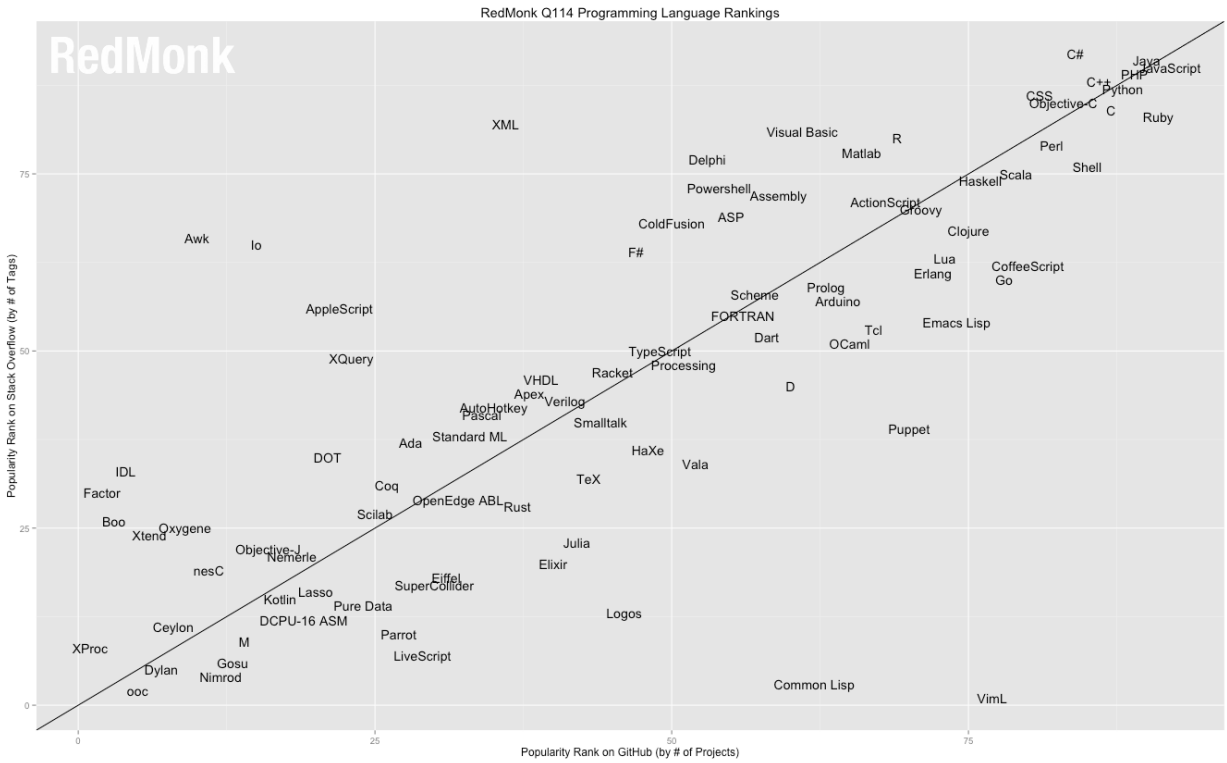
C# is one of the dominant web application languages in use today and consistently ranks in the top 5 for languages to know and use for web application development across industry, government, and academia. Two surveys from February of 2014 confirm C#'s popularity and widespread use. A survey by the online course provider Lynda.com for Mashable produced the following list of languages in order of popularity:

1. Java
2. C
3. C++
4. C#

5. Objective-C
6. PHP
7. Python
8. Ruby
9. JavaScript
10. SQL

Another industry analyst, RedMonk has taken a different approach and determined language popularity by examining the frequency of projects on GitHub and questions on StackOverflow:

1. JavaScript
2. Java
3. PHP
4. C#
5. Python
6. C++
7. Ruby
8. C
9. Objective-C
10. CSS



AngularJS

AngularJS is an open source JavaScript framework developed at Google and released in 2009. It uses the Model View Controller (MVC) architecture and leverages a user's internet browser to create a functioning web application. The AngularJS library interprets HTML markup along with specialized tag directives to interact with a web form. RESTful Web API services along with JSON can be used to provide interaction with a database server such as SQL Server. The way it works is that once the DOM is rendered AngularJS reads the markup looking for the AngularJS directives and injects the defined text or data into the markup

using an injector and two-way data binding between the data (view) and the resulting web page (scope).

DotNetOpenAuth with Google

One challenge of any web application is managing the users that will require access to the system. Table structure and security architecture must be built on top of any other functionality to ensure the user that logs in is an authentic user and has access to the application. The user login and password maintenance requirement has become a burden for web application developers and headache for system and network administrators to enforce password strength and reset durations. Google, and others, provides an open source solution that can be integrated with nearly any web application for free. The Google sign-in protections and procedures are more than adequate for the CMS and users are likely to already have an account. The familiar Google sign-in panel serves as the login page and relieves the developer of storing usernames and passwords. It also handles the many support requests around logging into the system since all that is handled through the Google interface.

OpenID is an open standard that provides user authentication by a relying party and identity provider [1], eliminating the need for web developers to implement their own user profile management systems. OAuth is an open protocol that provides for secure API authentication. The CMS acts as the relying party and Google is the identity provider.

HTML5/CSS3

HTML5 provides the web application developer with the easiest method of laying out web pages to date. Using HTML5 and CSS 3 will reduce the amount of markup required to render the web pages significantly. HTML5 with CSS3 represents the latest user interface design techniques available to date. The student interface uses a modal dialog window interface. The modal effect is accomplished using components of CSS3 and HTML5. As of this writing, Microsoft Internet Explorer does not support the World Wide Web consortium standards for inter browsers and is unable to render the modal windows. Students must use a compliant web browser such as the Mozilla Firefox browser or Google Chrome. The program director and administrator's interface is unaffected.

Bootstrap

Bootstrap is a user interface (UI) framework developed by Mark Otto and Jacob Thornton at Twitter during a hack week in 2010. They named it bootstrap and released it as an open source project in August 2011. By February 2012, Bootstrap was, and remains, the most popular development project of GitHub.

Bootstrap is a collection of free tools that provide developers with an easy to use design framework. It includes pre-built style sheets, methods, and JavaScript libraries. Web sites built with bootstrap are built to be mobile-first and have a responsive design. The responsive design element of the framework enables the web application to sense the user's browser and automatically structure the elements on the page for optimal viewing. Using bootstrap eliminates the need to develop a web site for traditional desktop or laptop user and again for

mobile device users. Bootstrap is compatible with all modern web browsers, but may not work correctly with older versions of Microsoft's Internet Explorer.

Zurb is an interaction design firm that offers their own frontend framework for user interface development similar to Bootstrap called Foundation. Both Foundation and Bootstrap offer integrated style sheets with pre-defined styles and a grid system, however, Bootstrap also includes a substantial icon set to use as well. The biggest advantage Bootstrap has over Foundation is the "mobile first" design of the framework meaning the Bootstrap framework is designed to function at a superior level on a phone or tablet in addition to a traditional desktop environment. Foundation merely offers some capabilities to handle a smaller screen environment.

AJAX

Asynchronous JavaScript and XML, or AJAX is a method for exchanging data between the server and browser without re-loading the entire web page. To "Ajaxify" a web page, the developer uses the XHTMLRequest object to send specified data to the server for processing. The server receives and processes the request. A response is then sent back to the page in XML format for parsing. The web browser interprets the server response and updates the user interface.

The primary benefit of using AJAX is an improved user experience. This is because data captured from the web page can be sent, processed, and the interface updated without the page refreshing. The Google search bar is one of the most common examples of AJAX in

action on the web. Each key press event sends information to Google's servers and search algorithms and enables the "search sensing" of the Google search engine.

AJAX is implemented on each user interface in the CMS so that there are no page refreshes for any user in the system (the traditional post-back model). AJAX is the enabling technology for the SPA design, and avoids the traditional post-back.

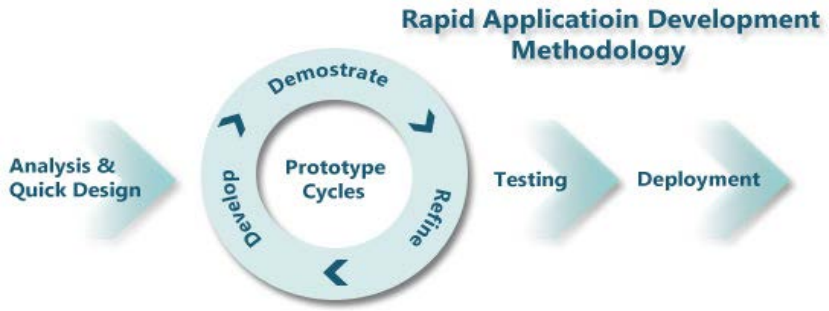
Jquery

Jquery is a collection of JavaScript libraries that simplifies JavaScript programming. Each web page exists in a general construct known as the document object model or DOM. One of the primary advantages of using the JavaScript programming language is the ability to interact with and manipulate the DOM to affect the user interface. Essentially, the DOM is designed as a large hierarchical tree data structure with parent, child, and sibling elements. Traversing the DOM in programming is not trivial. Jquery offers a framework of JavaScript to more easily select the elements of the DOM and change the user interface at runtime.

3.2. Development Methodology

The CMS will be developed using an iterative rapid application development (RAD) methodology. The RAD development methodology consists of four key phases; Requirements Planning, User Design, Construction, and Cutover. By using RAD methodology, the developer can quickly generate a working prototype or proof of concept to be reviewed by the project sponsor. Each review session with the sponsor leads to additional refinement and requirements to bring the system progressively closer to the vision of the

project sponsor. There are three scheduled major release iterations each followed by a review session with the project sponsor where a new RAD iteration will begin. In addition to the major releases several ad hoc minor releases will be deployed and tested.



The chosen IDE is Visual Studio 2013. Visual Studio 2013 is the preferred IDE for developing ASP.NET web applications with C#. Visual studio is the ASP.Net IDE for web application development offered by Microsoft. In addition to offering development tools, features, and code hinting, Visual Studio also supports the other web technologies of Jquery, AJAX, HTML5, Bootstrap, and database stored procedure development. The current backup strategy as implemented on the hosting environments (MISCAPSTONE & MISCAPSTONE [SQL]) is adequate for the needs of the CMS for supporting business continuity.

4. System Analysis/Design

4.1. Project Charter

The project charter for the CMS is found in Appendix B.1 below. It identifies Dr. Kline as the project sponsor, James Grooms as the developer, Drs. Cummings and Simmonds as additional committee members. Dr. Cumming is asked to review the user interface design

and provide feedback for the design as well as review for the thesis paper. Dr. Simmonds has been asked to serve as the committee member available for literature review and to suggest edits and structure for the thesis paper. Dr. Kline is the committee chair person and project sponsor and is responsible for providing the system requirements, reviewing the features as they are released, and providing guidance and direction required to complete the project. The charter also defines the expected development timeline and milestones for completion by May 2014.

4.2. Actors Diagram

The Actor Diagram for the CMS is found in Appendix A.1 below. Each component of the diagram represents a user interface in the system. The single page application development aspect for the project means there is a single web page for each interface.

4.3. Use Cases

The use cases for the CMS are found in Appendix D.1 below.

4.4. Workflow Diagram

The Capstone Management workflow diagram is found in Appendix C.1 below. The workflow for a capstone is the same regardless of the choice of project thesis or a thesis research paper. The student first must identify his capstone and form a capstone committee. A committee is formed by first requesting a faculty member from the MSCSIS department to serve as the chairperson of the committee. The chairperson then assists the student in selecting additional members and then provides his approval of the committee. The program director then approves the committee as a MSCSIS capstone committee. Once the additional members give their acceptance to serve on the capstone committee the capstone committee is complete. With the committee established, the student works on the capstone until he and his

committee chairperson agree that the student may propose his capstone thesis. The student must receive approval from all members of the committee to schedule the date, time, and location for the proposal. A public announcement must be posted on the LinkedIn group for the MSCSIS program. One the proposal is given the program director will designate the proposal complete. The student must then complete the same approval for scheduling the defense once he and his chairperson are ready for the capstone to be defended. Again, a public announcement on the LinkedIn group is required. After the defense, the program director is responsible for designating the defense completed. Once a capstone defense is completed the capstone workflow is completed.

4.5. Data Model

The complete data model for the CMS is found in Appendix H.1 below. The primary tables for the system consist of Capstone, CapstoneStatus, CommitteeMember, and CapstoneActivity. The Capstone table includes attributes for the capstone title, chairperson, proposal date, and defense date. The CommitteeMemeber table includes the attributes for capturing who the members are for each capstone. CapstoneStatus is the table that stores the data indicating the workflow steps completed. The CapstoneActivity table captures the capstone actions that are recorded by the system such as notification activity. There is a table named CapstoneAdminConfiguration that holds the control data for identifying the program director and administrator's authentication email addresses as well as the globally unique identifiers (GUIDs) that are embedded in the email notifications.

4.6. System Design

The CMS is designed an OLTP web application that manages the lifecycle and workflow of a capstone. ASP.NET 4.0 and C# is the platform and language used to write the software with a Microsoft SQL Server (v.2008) as the online database system. There are a several components of the design that are integrated using other libraries or APIs and are described below. Logging into the system, controlling the workflow, and handling inactive workflow steps are some of the design considerations taken into account. Appendix M below is the system architecture diagram for the CMS.

4.6.1. Login & Authentication

Logging into web applications means the system must store and manage user accounts with login names and passwords. There are considerations for security that must be taken into account such as strength of password and encryption methods. The CMS avoids this hassle all together by using the Google OAuth API. Using this federated login only requires the user to register an account with Google if they do not already have one. The federated login works by calling the Google API and requesting an authentication token that validates the user email address and password given are registered users with google. The .Net Forms Authentication is model is similar in that it also issues authentication tokens.

The code required to use Google federated login process is documented below. The code for AuthenticationEnpoint.aspx is documented in Appendix G below.

Web.Config file (<system.web> node):

```
<authorization>
   <deny users="?"/><!-- Deny all anonymous users -->
 </authorization>
 <authentication mode="Forms">
   <!-- Enable FormsAuthentication-->
    <forms loginUrl="default.aspx" protection="All" path="/" timeout="30"
    requireSSL="false"/>
 </authentication>

...

<!-- Grant access to this page to everyone -->
<location path="AuthenticationEndpoint.aspx">
  <system.web>
    <authorization>
        <allow users="*"/>
    </authorization>
  </system.web>
</location>
```

AuthenticationEndPoint.aspx.cs

1. AuthenticationEndpoint.aspx actually does the logging in by detecting if the user is logged in based on the presence or absence of an authentication token. The user clicks the login with Google Image on Default.aspx to start the process.

2. AuthenticationEndpoint.aspx verifies when the callback parameter is absent and calls the PerformGoogleAuthentication method.

3. The PerformGoogleAuthentication method sets up a callback address and also sets a callback parameter with the value "true" and defines which attributes it would like to have from Google. The CMS requests first name, last name and email address. The Google email address is the foreign key that links the Google account to the CMS.

4. PerformGoogleAuthentication redirects the user to google, including with the information of the requested attributes

5. The user logs into Google if not already logged in.

6. The user might need to grant access at google to allow the requested information to be sent to your page (if he did already, this step will be skipped).

7. Google redirects back to the AuthenticationEndpoint.aspx page with the attributes and the callback parameter in the query string, which was defined in a step earlier.

8. AuthenticationEndpoint.aspx requests callback parameter which is now set to "true", so the page calls the HandleAuthenticationCallback method

9. The HandleAuthenticationCallback method uses again the DotNet-Auth library to simply verify the posted data, if they were compromised, an exception is thrown.

10. With the parameters that were posted back from Google and the forms authentication authorization cookie as well as an AuthorizedUserCookie is set and the user redirected back to Default.aspx.

11. The user is now logged in. There are 3 roles in the system and the user will have a new menu option available depending on their role. Student can now select the My Capstone

link. The program director can now select Manage Capstones and the system administrator can now select the Administer Users.

4.6.2. Capstone Workflow

Each capstone must complete the capstone workflow. In general, the workflow for each capstone consists of completing several steps with approvals along the way via email from the committee members. A graphical representation of the workflow can be found in appendix C below. The program director may change the status of any capstone in the event a capstone needs to be moved back or forward in the workflow due to extenuating circumstances, however, the director may only move the capstone to one of the milestone statuses.

The workflow steps and emails that are generated for all capstones are as follows:

Status	Action	Email is sent to	Design Note
Not Started	Student registers with CMS	Program Director	
Begin Capstone	Student enters capstone title	N/A	Milestone
Request Committee Chair	Student requests chair	Chairperson (MSCSIS Faculty)	
Add Committee Members	Student Completes Committee Membership Form	Chairperson	Affiliate Members are not subject to approval process
Committee Approved by Chair	Chair reviews selected committee	Student, Director	Chair must approve the committee. If declined, then the capstone rolls back to Add Committee Members
Committee Approved by program director	Director reviews selected committee	Student	Director must approve the committee. If declined, then the capstone rolls back to Add Committee Members
Committee Complete	system action	Student	Milestone
Request Approval for Proposal Date	Student enters proposal information	Committee	All committee members must approve
Proposal Date Approved	Committee accepts proposal date	Student, committee, director, admin	If declined the capstone rolls back to Committee Complete. If approved a calendar request is sent with an email notification
Post Proposal on LinkedIn	Student enters URL for LinkedIn announcement	Committee	
Proposal Complete	Program Director marks capstone as proposed	Student	Milestone
Request Approval for Defense Date	Student enters defense information	Committee	All committee members must approve
Defense Date Approved	Committee accepts defense date	Student, committee, director, admin	If declined the capstone rolls back to Proposal Complete. If approved a calendar request is sent with an email notification
Post Defense on LinkedIn	Student enters URL for LinkedIn announcement	Committee	
Defense Complete	Program Director marks capstone as defended	Student	Milestone

4.7. Status Duration

There are certain status steps where duration plays an important role. In order to prevent a capstone workflow from stalling out because of inactivity the system will change the status of a capstone if a duration threshold is met. This duration threshold is evaluates each time a student accesses the system. If the status duration has expired, then the system changes the status to a previous status to enable the student to repeat the expired step.

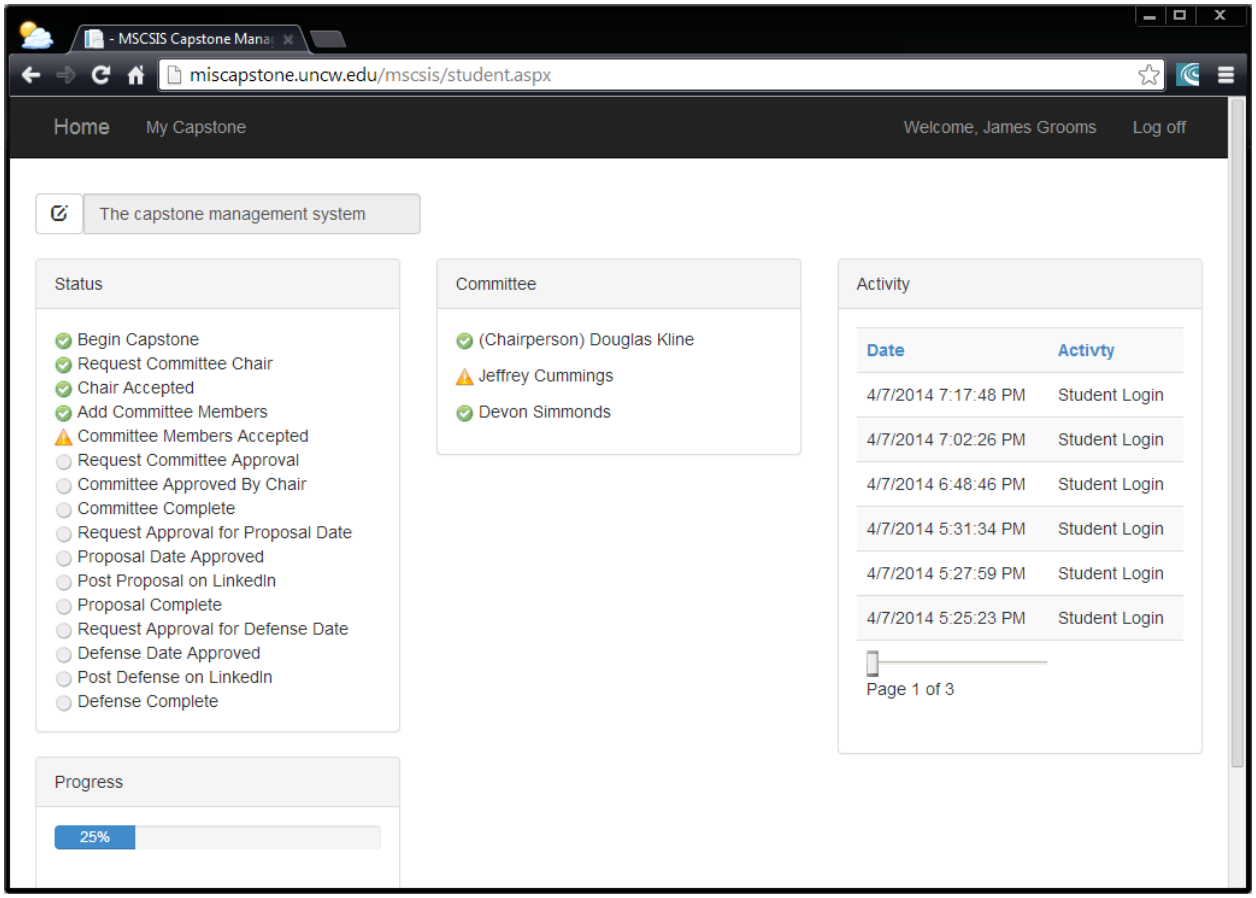
Status	Duration	Status Change on Expiration
Request Committee Chair	7 days	Begin Capstone
Committee Members Accepted	7 days	Chair Accepted
Request Committee Approval	7 days	Add Committee Members
Proposal Date Approved	7 days	Committee Complete
Defense Date Approved	7 days	Proposal Complete

4.8. System Roles and Interfaces

There are 5 roles in the CMS: the student, the program director, the system administrator, the committee chairperson, and the committee member. There are 3 user interfaces in the system as well as the email approval process for facilitating the actions for each of the roles.

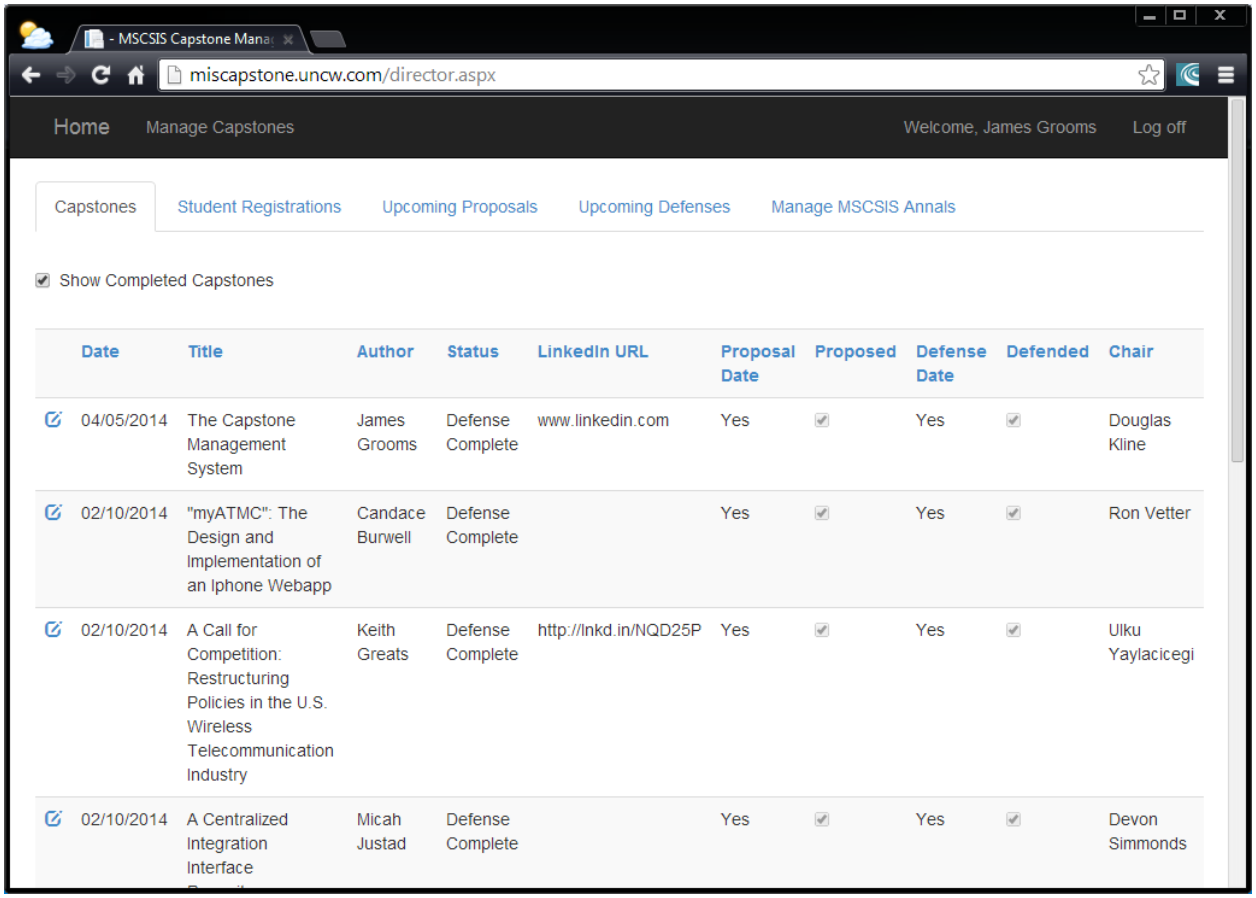
The student is the primary role for the system. He is the driver of the workflow for his capstone. The interface allows the student to set the capstone title, Request a committee chairperson, request committee members, add affiliate members, record the LinkedIn announcement URL, as well as record the date, time, and location for the capstone proposal

and defense. The screenshot below is an example of the student's user interface.

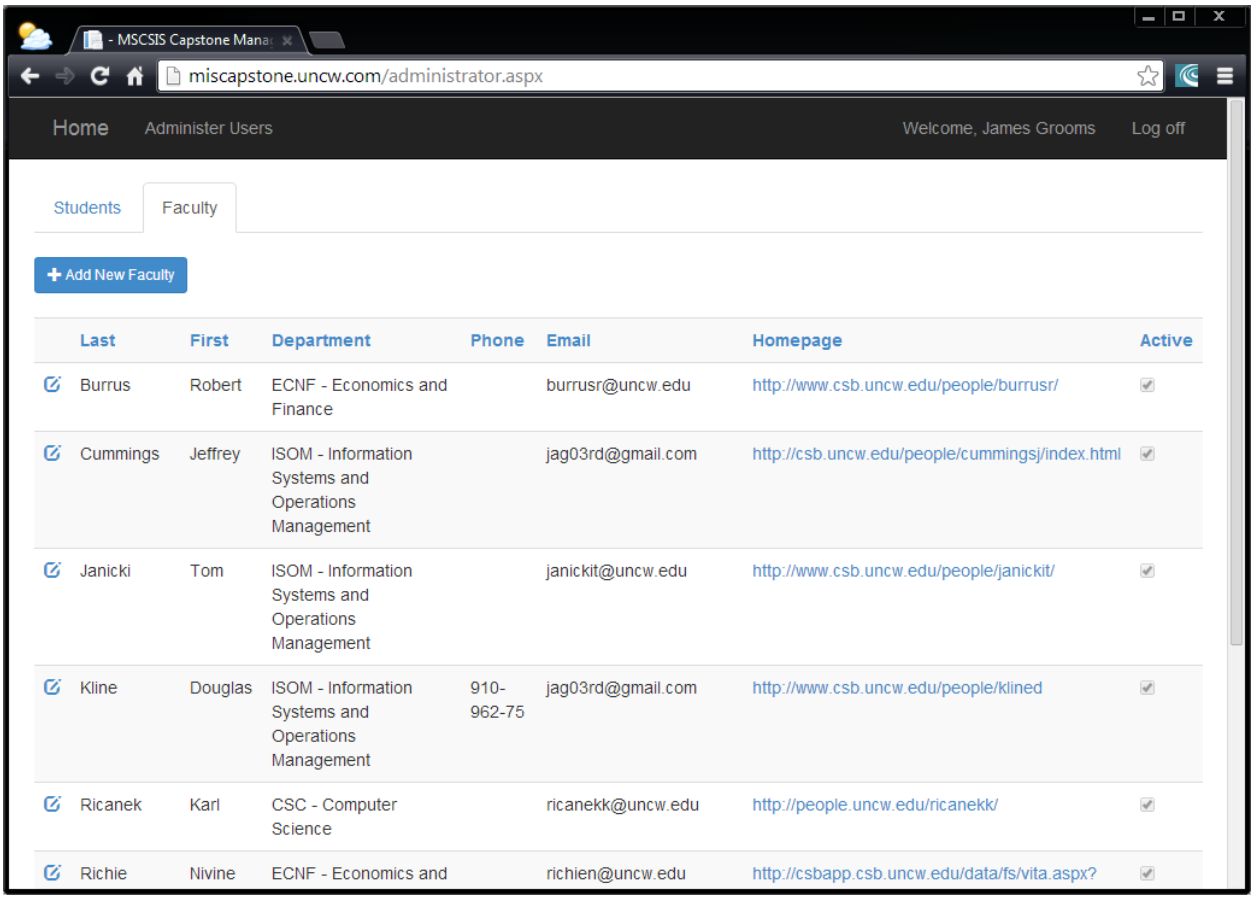


The program director is responsible for acknowledging student registration requests, indicating a capstone has completed either a proposal or defense. The program director may also override the workflow for a capstone. The screenshot below is an example of the

program director's user interface.



The system administrator is responsible for the data entry of students and faculty into the system. An example of the user interface is below.



When the committee chairperson or other committee members are required to accept or decline a capstone email request they are not required to login. The committee members can simply accept or decline based on a link embedded in the email body. The template for each email type is in appendix G below.

The email approval functionality works by clicking a link embedded in the email body. The contents of the link provide the system with the information required to complete the data update. In addition to the ids the system also uses a globally unique identifier (GUID) in the link to prevent URL spoofing. The value of the GUID is controlled in a configuration control

table that identifies the acceptance GUID and the decline GUID.

4.9. System Diagram

The CMS diagram can be found in Appendix C.1 below. The system is designed using an n-tiered architecture meaning there is a separation of concerns to modularize the system for ease of development and maintenance. The application layer consists of the web pages, code files, and cascading style sheets that are used to render each web page. The business logic/process layer consists of the code files and configuration files that provide the validation and processing logic of the system. The data access layer is made up of the many stored procedures to add, update, read, and delete the system data.

4.10. Criteria for Success

The CMS can only be determined to be a successful project if the students and faculty for the MSCSIS program adopt the application to drive capstone management and the use of paper forms and manual processes for completing the workflow are eliminated. The CMS is designed as a powerful tool to improve the capstone experience for all the stakeholders in the program. The application is expected to become available for use by existing students in June of 2014 with all MSCSIS students using the system in the fall.

5. Project Plan

5.1. Scope

In Scope

The scope of the Capstone management is to provide a web application to help students complete the capstone workflow process. Students must be able to form capstone committees as well as schedule capstone proposals and final defense. Committee members must be able to accept a committee request and view the status of their capstone committees. The program director must be able to manage the completion of proposals and defenses for each capstone and view online reports. There must also be a designated administrator to maintain the users of the system

Out of Scope

Functionality that explicitly ruled out of scope is the uploading of any files within the system. Additionally, no user interface is being developed to manage the workflow. The capstone workflow is designed in the database architecture and executed with programming process logic. A capstone committee may have non-UNCW faculty, however, they will not have the ability to interact with the system. The student will provide the first name, last name, email and affiliation for each affiliate committee member.

5.2. Timeline

The expected timeline is 10 months. August 2013 – May 2014

Task Name	Duration	Start
System Analysis & Design	**70 days**	**Mon 8/19/13**
Project Plan	1 day	Fri 9/27/13
Project Charter	21 days	Fri 8/30/13
Use Cases	25 days	Mon 9/2/13
Workflow Diagram	6 days	Sat 9/28/13
Data Model	46 days	Sun 9/1/13
Analysis & Design Complete	0 days	Sat 11/16/13
Capstone Proposal	0 days	Tue 11/19/13
System Development	**91 days**	**Sun 12/1/13**
System user login	10 days	Mon 12/9/13

Student Module [rev 1]	13 days	Fri 12/13/13
Committee Member Module[rev 1]	5 days	Mon 1/6/14
Program Director Module [rev 1]	6 days	Sat 1/11/14
System Administrator Module [rev 1]	3 days	Sat 1/18/14
Workflow Engine [rev 1]	6 days	Wed 1/22/14
System Review [rev 1]	3 days	Wed 1/29/14
Student Module[rev 2]	19 days	Sat 2/1/14
Revision 1 Complete	0 days	Sat 2/1/14
Committee Member Module [rev 2]	19 days	Sat 2/1/14
Program Director Module [rev 2]	19 days	Sat 2/1/14
System Administrator Module [rev 2]	19 days	Sat 2/1/14
System Review [rev 2]	3 days	Wed 2/26/14
Workflow Engine [rev 2]	19 days	Sat 2/1/14
Revision 2 Complete	0 days	Sat 3/1/14
Student Module [final]	22 days	Sat 3/1/14
Committee Member Module [final]	22 days	Sat 3/1/14
Program Director Module [final]	22 days	Sat 3/1/14
System Administrator Module [final]	22 days	Sat 3/1/14
Workflow Engine [final]	22 days	Sat 3/1/14
System Review [final]	5 days	Mon 3/31/14
Development Complete	0 days	Fri 4/4/14
Implementation	**11 days**	**Fri 4/4/14**
Database Setup	11 days	Sat 4/5/14
Website setup	11 days	Sat 4/5/14
Closing	**10 days**	**Mon 4/21/14**
Capstone Proposal	10 days	Mon 4/21/14

5.3. Resources

The expected resources required to complete the system are as follows:

- Dr. Kline – Project Sponsor/Subject Matter Expert/Chairperson
- Dr. Cummings – Committee member and system design consultant
- Dr. Simmonds – Committee member and literature review
- James Grooms – Design, Development, Testing & Implementation of the system
- Microsoft Visual Studio 2013 – Integrated Development Environment
- Microsoft SQL Server 2012 – Relational Database Management System

- QA MSCSIS Resources: Michael Abate, Jazzman Capezza, Andrew Keener, Rich Mautz

5.4. Limitations

The limitations of the CMS are time and resources. There is a single developer working 10 – 15 hours per week for 12 weeks in an effort to complete the requirements of the system.

6. Project Risks and Mitigation Measures

The risks to project are documented in the Project Charter as listed in Appendix B.1 below. The primary risk to the project is a lack of user buy-in which can be mitigated with a communication strategy aimed at ensuring students and faculty are aware of the system and the many benefits provided. The system is designed with minimal complexity to mitigate the risk associated with software development and to create a user friendly application.

Another risk to the project is the learning curve associated with AngularJS. If the learning curve is deemed too steep then the developer may need to develop the application in a difference and better known language.

7. Development Milestones by Month

7.1. Fall 2013

- Project charter completed

- System design/analysis is completed
- System development technologies identified

7.2. December 2013 - January 2014

- Google Login with OpenAuth/OpenId
- Begin student Module
- 2-3 development iterations and review with the project sponsor

7.3. February 2014

- Complete program director user interface
- Complete administrator user interface
- Complete student user interface
- 2-3 development iterations and review with the project sponsor

7.4. March 2014

- Identify QA testers and user acceptance testing
- Ongoing QA/testing
- 2-3 development iterations and review with the project sponsor

7.5. April 2014

- Ongoing QA/testing
- Implement system on UNCW web and database servers.

7.6. May 2014

- Complete development of final changes identified
- Final development iterations and meetings with project sponsor
- Implement into production environment
- Complete System Turnover

8. Confirmation of Predictions

The proposed scope and timeline turned out to be reasonably accurate in terms of completing the requirements on time. Taking the time to do the analysis in the fall of 2013 did a good job of setting the scope because there were no initial features removed from the deliverables list. The risk identification exercise proved to be extremely valuable. A key risk factor was the learning curve associated with developing the software using AngularJS which was unknown to the developer. The developer used 5 weeks of time in December of 2013 and into January 2014 before determining trying the learn AngularJS was not progressing fast enough and the project would be at risk. The decision was made to scrap the AngularJS development language for the more traditional ASP.NET using C#. This decision made it possible to implement the Bootstrap framework and the calendar request attachments that would have been ruled out of scope otherwise. Development using visual studio, which is the integrated development environment for C# application, began in earnest in February. The challenge was to develop a web application using the rapid application development methodology and integrating many web technologies such as Bootstrap, C# web forms, Jquery, and HTML5 for the application layer, C# classes for the business logic/process layer, and AJAX, ASP.NET datasets, and stored procedures with a

SQL Server database for the data layer. Each technology is required to integrate seamlessly to complete the application. The developer and project sponsor communicated weekly regarding progress and to demonstrate the latest working prototype to assess the requirements and ensure each deliverable was met. The web application consists of the web pages and code files hosted on a UNCW web server (MISCAPSTONE) and a database environment hosted on a UNCW (MISCAPSTONESQL) database server. The deployment process involved copying the web pages and code file to the location on the server where the web site is running and then deploying the SQL scripts to create the table structure and stored procedures. The testing process consisted of the developer unit testing each feature set by validating the use cases defined in the analysis phase. Once the first iteration of the whole application was completed, a user acceptance test was conducted with volunteers from MSCSIS students and faculty.

9. Conclusions

9.1. Lessons Learned

I saw this capstone and more generally, the MSCSIS program, as an opportunity to refresh my skillset and really "plug in" to the emerging technologies in web application development. I learned that 2013 was the year that AngularJS really took off and so I intended to develop this capstone project using it. I invested months in online tutorials at sites like plurlsight.com. I scoured the web for YouTube videos hoping to quickly learn how to develop a full OLTP web application with data base connectivity, CRUD, AJAX, MVC, and SPA architecture. Time quickly ran out on me and by the end of 10 weeks I had made real progress, but not enough to be able to complete the requirement of the capstone that lay before me. I had to decide whether I was to go back to my capstone committee chair and ask

to pare down the number of interfaces and deliver a project that did not meet the original specifications.

I am thankful that I recognized the risk posed to the project by learning an unknown development language and made the decision not to continue down the AngularJS path. Instead, I decided to bring other emerging technologies into the project that would help me work on my weaknesses as a developer. My primary area of weakness concerning web application development is user interface design. After some research I learned about the Bootstrap UI framework that was released by Twitter in early 2011 and saw it as my opportunity to implement the latest technology for interface design and improve myself as a designer. The Bootstrap framework has exceeded my highest expectations in terms of providing a way to deliver an attractive user interface design that also works on the mobile and standard platforms.

9.2. Key Accomplishments

I believe the key accomplishment of the CMS is the SPA design of the web application. The system is able to manage all the activities of the roles within a single page per role. With the use of Bootstrap and AJAX, the interfaces for students, the director, and the admin all operate without a single page refresh and look great on any device. In addition to the SPA design, the integration of multiple web technologies stands out as a key accomplishment. I have successfully implemented a user interface framework within the ASP.Net 4.5 development framework and also leveraged Jquery and OpenAuth with a federated login via Google. Each technology presents its own challenges and I believe I was able to successfully

weave them all together effectively.

9.3. Future Work

The following features have been discussed as areas of development that may be pursued as enhancements to the system.

1. Sending Text Notifications
2. OpenAuth/OpenID with LinkedIn
3. Upload file attachments
4. Extensibility into additional UNCW graduate programs
5. A reporting module that provides graphs and charts and key metrics for the program director

10. Skills Acquired During the Project

Over the course of this capstone I have been able to complete my goal of participating in the MSCSIS program by acquiring an updated skillset for web application development. Some of the technology disciplines that I advanced in were:

- AngularJS
- Visual Studio 2013 IDE with NuGet Docs
- Bootstrap 3.0
- HTML5

- CSS3
- Jquery
- AJAX
- ASP.NET 4.0 Framework
- Project Management

11. Acknowledgements

I would like to acknowledge Dr. Doug Kline who served as my capstone committee chair person and the project sponsor for the capstone. I truly appreciate his always being available to me for questions or guidance.

I would also like to acknowledge Dr. Tom Janicki for his assistance in this project. Dr. Janicki is neither a committee member nor stakeholder in my capstone, yet assisted greatly in the course of implementation.

Most significantly, I would like to offer my heartfelt thanks to my wife for her love, patience, and tireless support during this capstone project and throughout my time in the MSCSIS program. I would not have been able to accomplish this without her.

12. Works Cited

[1] Andrew, Paul. "What to Use Windows Workflow Foundation For?"*Http://blogs.msdn.com/b/pandrew/archive/2007/02/01/what-to-use-windows-workflow-foundation-for.aspx*. N.p., 1 Feb. 2007. Web. 3 Mar. 2012. <http://blogs.msdn.com/b/pandrew/archive/2007/02/01/what-to-use-windows-workflow-foundation-for.aspx>.

[2] David Recordon and Drummond Reed. 2006. OpenID 2.0: a platform for user-centric identity management. In *Proceedings of the second ACM workshop on Digital identity management* (DIM '06). ACM, New York, NY, USA, 11-16. DOI=10.1145/1179529.1179532 http://0-doi.acm.org.uncclc.coast.uncwil.edu/10.1145/1179529.1179532

[3] ASP.NET Forms Authentication Overview. (2013, September 23). Microsoft, Copyright 2013, All rights reserved. Retrieved 9 2013, September from Microsoft: http://msdn.microsoft.com/en-us/library/7t6b43z4.aspx

[4] "Bootstrap." *Bootstrap*. N.p., n.d. Web. 31 Mar. 2014. <http://getbootstrap.com/>.

[5] "JQuery." *JQuery*. N.p., n.d. Web. 12 Apr. 2014. <http://www.jquery.com/>.

[6] "AJAX Introduction." *AJAX Introduction*. N.p., n.d. Web. 12 Apr. 2014. <http://www.w3schools.com/ajax/ajax_intro.asp>.

[7] "HTML5." *Mozilla Developer Network*. N.p., n.d. Web. 12 Apr. 2014. <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5>..

[8] "What Do You Want to Learn Today?" *Online Video Tutorials & Training*. N.p., n.d. Web. 31 Mar. 2014. <http://www.lynda.com/>.

[9] "RedMonk - Analysis for the People, by the People." *RedMonk Home Comments*. N.p., n.d. Web. 31 Mar. 2014. <http://redmonk.com/>.

[10] "AJAX Tutorial." *AJAX Tutorial*. N.p., n.d. Web. 09 Apr. 2014. <http://www.w3schools.com/ajax/default.ASP>.

[11] McConnell, Steve. *Code Complete*. Redmond, WA: Microsoft, 2004. Print.

[12] Sommerville, Ian. *Software Engineering*. Boston: Pearson, 2011. Print.

[13] Fowler, Martin. *UML Distilled: A Brief Guide to the Standard Object Modeling Language.* Boston: Addison-Wesley, 2004. Print.

Appendices

Appendix A. Terminology

MSCSIS – The UNCW Masters of Science in Computer Science and Information Systems program.

API – Application Programming Interface

AJAX – Asynchronous JavaScript and XML

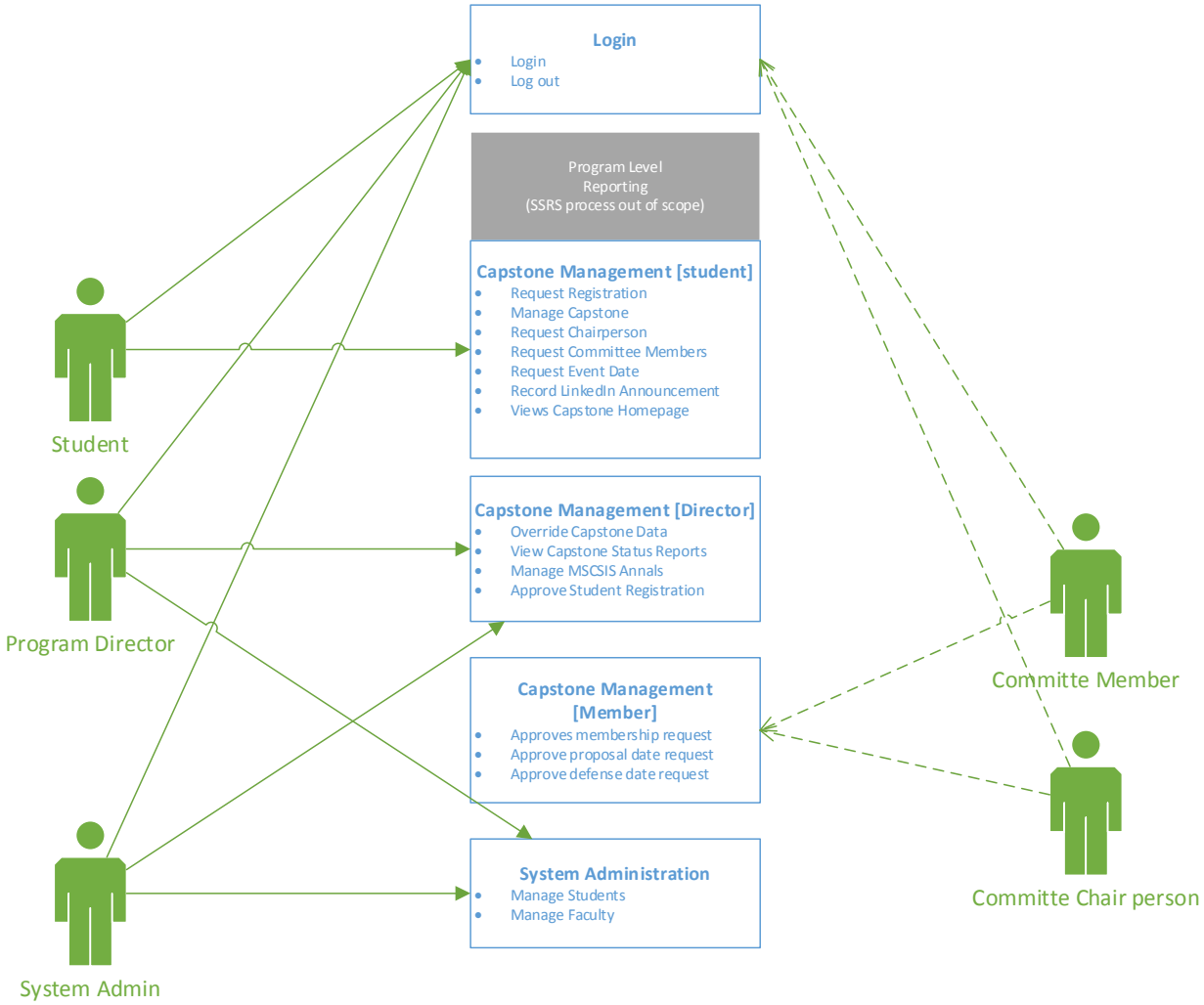
RAD – Rapid Application Development

SPA – Single Page Application

IDE – Integrated Development Environment

CMS – Capstone Management System

Appendix B. Actors Diagram



Appendix C. Project Charter

Project Charter - Capstone Management System

Business Purpose

The purpose of the CMS is to provide an online web application that assists both the students and faculty of the UNCW MSCSIS program in successfully navigating the capstone process including selecting a chairperson, forming a committee, scheduling and announcing the capstone proposal, and scheduling the capstone oral defense. Additionally, the CMS establishes an automated process, replacing the current manual process that provides program-level reporting on the capstone experience in the MSCSIS program.

Business Objectives

- Reduce the time required to form a capstone committee
- Improve the student understanding of the capstone workflow
- Automate the manual system using a web application and email notification
- Save time and increase efficiency by eliminating 4 of the 6 forms required to complete a capstone
- Increase faculty/student communication with email notifications to facilitate the workflow
- Increase the percentage of students that graduate on time with reports to monitor capstone progress, workload, and auditing

- Record the public announcements for capstone proposal and defense

Desired Features

- Identify a capstone chairperson
- Form a capstone committee
- Post a public announcement of proposal and defense on linkedin.com 10 business days preceding the event and record the link URL in the system
- Schedule capstone proposal and defense date, time, and location
- Process a committee request online
- Process a committee request through email
- Maintain a record that all final forms are completed
- Manage users
- Manage UNCW MSCSIS capstone annals
- Manage the UNCW MSCSIS capstone workflow
- Capstone data reporting
- Provide a capstone activity dashboard for each user type

Critical Success Factors

- Simplicity of use
- Faculty/staff training
- Adoption of the system

Constraints

- Time: Must be completed prior to May 2014
- Internet connectivity

Risks

- User buy-in
- Learning curve of the development technology
- Meeting the project deadlines
- Scope creep
- Incomplete requirements

Roles and Responsibilities

- James Grooms: Developer
- Dr. Douglas Kline: Committee chairmen[Project sponsor]
- Dr. Jeff Cummings: Committee member [System design]
- Dr. Devon Simmonds: Committee member [Literature review]

Approach

- Complete system analysis and design in the fall semester
- Propose CMS
- Develop web application using rapid application development (RAD) with multiple iterations and review with the stake holders during the spring semester

Locations of Interest

- University of North Carolina Wilmington, Computer Information Systems (CIS) building
- Application hosting environment: CIS 2035a

Actors

- Student
    - Request registration
    - Initialize a capstone project or thesis
    - Request committee chairperson
    - Request additional committee members (faculty or other)
    - Schedule Proposal
    - Schedule Defense
    - View progress

- Committee Chairperson
    - Process committee request
    - Approve committee
    - Approve proposal request
    - Approve defense request

- MSCSIS Program Director
    - Approve registration request
    - Complete proposal
    - Complete defense
    - View reports

- System Administrator
    - Manage users

Preliminary Execution Architecture

- Development Environment: Visual Studio 2013, C#, AngularJS, AJAX, Twitter Bootstrap 3, HTML5, CSS3, Jquery
- Client/server architecture with hybrid single page application (SPA) interface

Project Infrastructure

- Web application will reside on the MISCAPSTONE server
- Database will reside on the MISCAPSTONE SQL server

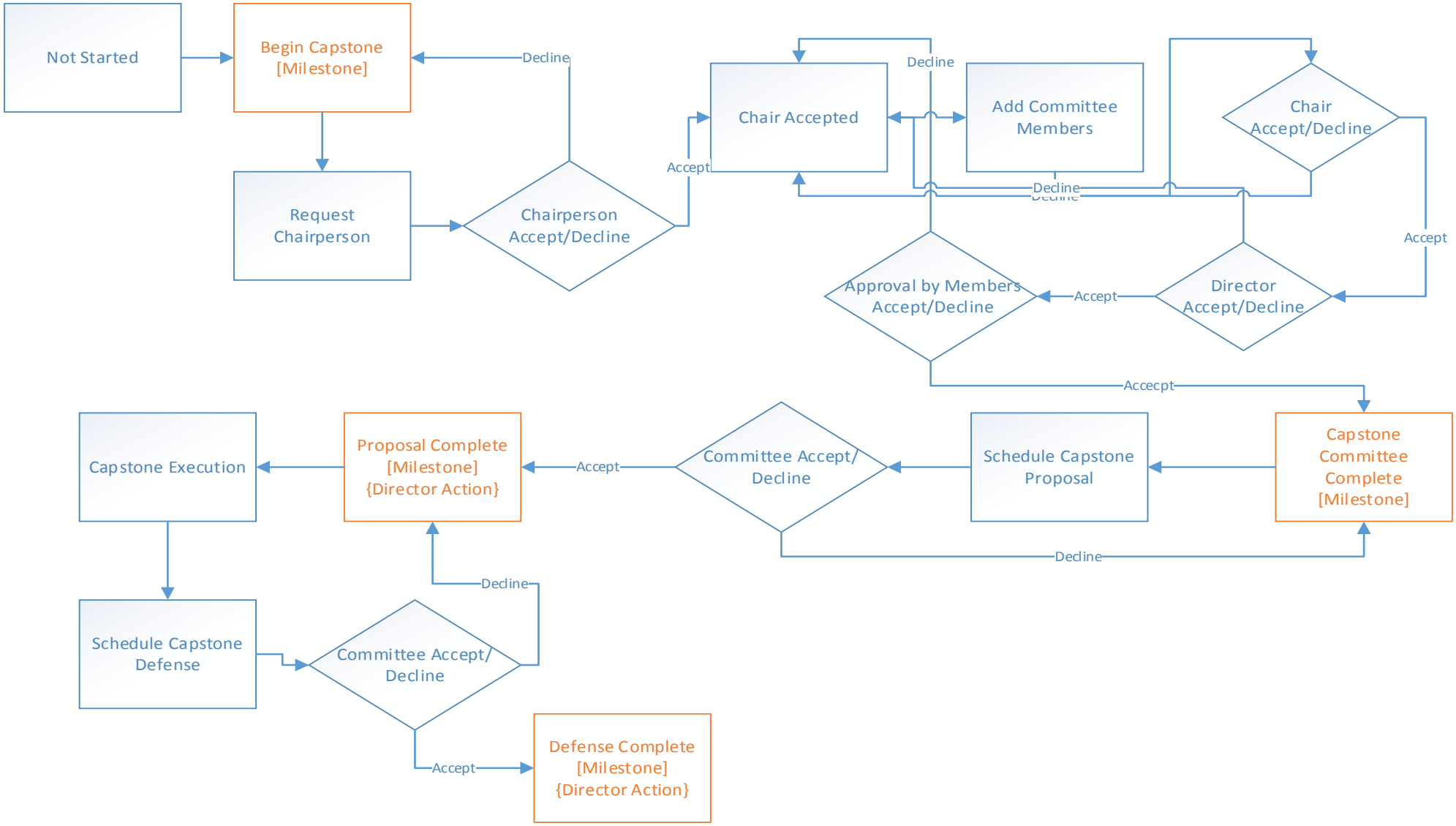
Project Release Strategy

- Rollout of the software will occur during the final weeks of the spring 2014 semester with expected use the following fall

Application Contingency

After completion, the application will be maintained by the faculty and students in the MSCSIS, Information Systems, and Computer Science departments.

Appendix D. Workflow Diagram



Appendix E. Use Cases

Use Case	**Login**	
Description	User signs into the system using their Google account	
Frequency	Frequent: 10-30/day	
Actors	Student, Committee Member, Committee Chairperson, Program Director, System Administrator	
Related Use Cases	N/A	
Stakeholders	Student, Committee Member, Committee Chairperson, Program Director, System Administrator	
Happy Pathway	User signs into the system with their Google account	
Preconditions	User has created a Google account	
Post-Conditions	User is authenticated and redirected to the homepage	
Flow of Events	**Actor**	**System**
	1. User clicks the "Sign in with Google" button	2. Google open authentication API is used to authenticate the user and returns an authentication token 3. User is redirected to the homepage 4. User clicks the Home navigation link 5. Action is recorded 6. User's homepage is displayed
Exception Conditions	1. If a user enters invalid credentials an error message is displayed	

Use Case	**Log out**	
Description	User logs out of the system	
Frequency	Frequent: 10-30/day	
Actors	Student, Committee Member, Committee Chairperson, Program Director, System Administrator	
Related Use Cases	N/A	
Stakeholders	Student, Committee Member, Committee Chairperson, Program Director, System Administrator	
Happy Pathway	User signs out of the system	
Preconditions	User is currently logged into the system	
Post-Conditions	User is logged out of the system and redirected to the login page	
Flow of Events	**Actor**	**System**
	1. User clicks the logout button	2. User's authentication token is destroyed 3. Action is recorded 4. User is redirected to login page
Exception Conditions	User closes the browser window without signing out—login is not required as long as the authentication token is still valid	

Use Case	**Student Requests Registration**	
Description	The student registers with the system	
Frequency	Episodic: 10-30/semester	
Actors	Student	
Related Use Cases	Login, Log Out, Director Approves Registration Request	
Stakeholders	Student, Program Director	
Happy Pathway	Student enters UNCW 850* number and an email request is sent to the Program director	
Preconditions	User has not previously registered with the system	
Post-Conditions	Email is sent to the Program Director for approval	
Flow of Events	**Actor**	**System**
	1. Student Enters UNCW 850* number 2. Student Clicks the Save button	3. Send email to the Program Director requesting approval
Alternate Paths	Student Logs out of the system	
Exception Conditions	System Timeout. Student is logged out	

Use Case	**Student Manages Capstone**	
Description	The student updates the capstone title	
Frequency	Episodic: 10-30/semester	
Actors	Student	
Related Use Cases	Login, Log Out	
Stakeholders	Student	
Happy Pathway	Student enters title of capstone thesis or project	
Preconditions	User has not created a capstone title	
Post-Conditions	Capstone is entered into the system	
Flow of Events	**Actor**	**System**
	1. Student enters name of capstone thesis topic or capstone project 2. Student Clicks the save icon	3. Capstone is added to the system if does not exist 4. Action is recorded
Alternate Paths	User updates the title of the capstone thesis or project	
Exception Conditions	System Timeout. No data is stored	

Use Case	**Student Requests Chairperson**	
Description	The student requests a UNCW faculty member as the committee chairperson	
Frequency	Episodic: 10-30/semester	
Actors	Student	
Related Use Cases	Login, Log Out	
Stakeholders	Student, Committee Member	
Happy Pathway	Student selects UNCW faculty member as chairperson and notification goes out	
Preconditions	User does not have a committee chairperson	
Post-Conditions	Committee chairperson is emailed a request	
Flow of Events	**Actor**	**System**
	1. Student clicks Request Committee Chair from status panel 2. Student selects chair person 3. Student clicks save button	4. Email notification is sent to selected user 5. Action recorded
Alternate Paths	N/A	
Exception Conditions	System Timeout. No data is stored	

Use Case	**Student Request Committee Members**	
Description	The student requests additional committee members	
Frequency	Episodic: 10-30/semester	
Actors	Student	
Related Use Cases	N/A	
Stakeholders	Student, Committee Member	
Happy Pathway	Student forms a committee of UNCW faculty including 1 member of either CIS or MIS and at least 1 additional member that may be faculty or an affiliate member	
Preconditions	User has a committee chairperson	
Post-Conditions	Email requests for approval are sent to chairperson for approval	
Flow of Events	**Actor**	**System**
	1. Student clicks Add Committee Members link in the status panel 2. Student completes the form for selecting members 3. Student clicks Send Requests button	4. Email notification is sent chairperson for approval 5. Affiliate members are added to the committee as accepted 6. Action Recorded
Alternate Paths	Student Adds an External committee member and notification goes out	
Exception Conditions	System Timeout. No data is stored	

Use Case	**Student Views Capstone Homepage**	
Description	The capstone homepage provides the student with at-a-glance knowledge of the progress being made toward completing the capstone workflow. The homepage also provides a history of the actions that have taken place during the lifecycle of his capstone.	
Frequency	Episodic: 10-30/week	
Actors	Student	
Related Use Cases	Login, Log Out	
Stakeholders	Student	
Preconditions	Student has completed registration	
Post-Conditions	Capstone homepage is displayed for the student	
Flow of Events	**Actor**	**System**
	1. Student logs into the CMS 2. Student clicks the My Capstone menu option	3. Capstone homepage is displayed to the student
Alternate Paths	This is the first login and a registration request form must be completed	
Exception Conditions	System Timeout. No data is stored	

Use Case	**Student Requests Event Date**	
Description	The student submits a proposal and defense date, time, and location to the committee chairperson for approval	
Frequency	Episodic: 10-30/Semester	
Actors	Student, Committee Chairperson	
Related Use Cases	User Accepts email request, User Declines email request, User repeats email request	
Stakeholders	Student	
Preconditions	N/A	
Post-Conditions	Capstone is in the status of Committee Complete	
Flow of Events	**Actor**	**System**
	1. Student completes form for event request 2. Student clicks Save button	3. Email notification is sent to capstone chairperson 4. Action recorded
Alternate Paths	N/A	
Exception Conditions	System Timeout. No data is stored	

Use Case	**Student Records LinkedIn Announcement**	
Description	The student is responsible for making a public announcement of their proposal and defense for their capstone	
Frequency	Episodic: 10-30/semester	
Actors	Student	
Related Use Cases	Login, Log Out, Student Views Capstone Homepage	
Stakeholders	Student Committee Member, Committee Chairperson	
Happy Pathway	User enters the URL to the LinkedIn announcement	
Preconditions	Capstone is in a status of Proposal Data Approved or Defense Date Approved	
Post-Conditions	URL is recorded and displayed on the student homepage	
Flow of Events	**Actor**	**System**
	1. Student logs into the system 2. Student clicks the Post on LinkedIn link in the status panel 3. Student enters URL 4. Student clicks Save button	5. URL is recorded and displayed 6. Student clicks Log Out
Alternate Paths	User updates faculty record data	
Exception Conditions	System Timeout. No data is stored	

Use Case	**User Accepts Email Request**	
Description	Each email request has a link for accept and decline to click indicating the response to request. Email requests are sent out for committee member requests, committee approval, and proposal/defense date approval	
Frequency	Episodic: 10-30/semester	
Actors	Faculty Committee, Committee Chair	
Related Use Cases	User Declines email request, User repeats email response	
Stakeholders	Student, Committee Member, Committee Chair	
Happy Pathway	User clicks the Accept link in the body of the email	
Preconditions	Email request has not been responded to	
Post-Conditions	Email response has been processed	
Flow of Events	**Actor**	**System**
	1. User receives notification with links to click indicating response. 2. User clicks the accept link	3. System processes the response 4. A message is displayed to the user indicating the response is complete. 5. Email notification sent to the student 6. Action recorded
Alternate Paths	User clicks the decline link, Acceptance required a status change by the system	
Exception Conditions	N/A	

Use Case	**User declines email request**	
Description	Each email request has a link for accept and decline to click indicating the response to request	
Frequency	Episodic: 10-30/semester	
Actors	Faculty Committee, Committee Chair	
Related Use Cases	User accepts email request, User repeats email response	
Stakeholders	Student, Committee Member	
Happy Pathway	User clicks the decline link in the body of the email	
Preconditions	Email request has not been responded to	
Post-Conditions	Email response has been processed	
Flow of Events	**Actor**	**System**
	1. User receives notification with links to click indicating response. 2. User clicks the decline link	3. System processes the response 4. A message is displayed to the user indicating the response is complete. 5. The status of the capstone is rolled back so that the workflow step may be repeated if needed. 6. Email notification sent to the student 7. Action recorded
Alternate Paths	User clicks the accept link	
Exception Conditions	N/A	

Use Case	**User Repeats Email Request**	
Description	Each email request has a link for accept and decline to click indicating the response to request	
Frequency	Episodic: 10-30/semester	
Actors	Faculty Committee, Committee Chair	
Related Use Cases	User accepts email request, User declines email response	
Stakeholders	Student, Committee Member	
Happy Pathway	User clicks the decline or accept link in the body of the email	
Preconditions	Email request has already been responded to	
Post-Conditions	No action taken towards the workflow	
Flow of Events	**Actor**	**System**
	1. User receives notification with links to click indicating response. 2. User clicks the a link in the body of the email	3. System determines the response has already been completed 4. A message is displayed to the user indicating the response has already been completed.
Alternate Paths	N/A	
Exception Conditions	N/A	

Use Case	**Registration Request Approved**	
Description	The program director approves a MSCSIS registration request.	
Frequency	Episodic: 10-30/semester	
Actors	Director	
Related Use Cases	Login, Log Out	
Stakeholders	Student, Director, Administrator	
Happy Pathway	Director or Administrator marks student request as approved	
Preconditions	User has completed the registration request	
Post-Conditions	Student is approved as a user	
Flow of Events	**Actor**	**System**
	1. Email notifies director of student registration 2. Director logs into the system 3. Director checks the approved check box an clicks the save icon	4. Email is sent to student notifying the director has approved their student registration request
Alternate Paths	N/A	
Exception Conditions	System Timeout. No data is stored, No email is sent	

Use Case	**Director Manages Capstones**	
Description	The program director is responsible for marking capstones as proposal/defense complete and may update a capstone's title or move it to a milestone status	
Frequency	Episodic: 10-30/Semester	
Actors	Director, Administrator	
Related Use Cases	Login, Log Out	
Stakeholders	Student, Director	
Preconditions	N/A	
Post-Conditions	Capstone has been created	
Flow of Events	**Actor**	**System**
	1. Director logs into the system 2. Director selects a capstone record and enters data 3. User clicks the Save button	4. Email notification is sent when proposal/defense completed 5. Capstone status updated 6. Action recorded
Alternate Paths	Only title is updated, status is changed, proposed/defended is checked off	
Exception Conditions	System Timeout. No data is stored	

Use Case	**Director Manages Capstone Annals**	
Description	The program director maintains the annals of MSCSIS program that provides public access to the repository of capstone work.	
Frequency	Episodic: 10-30/Semester	
Actors	Director, Administrator	
Related Use Cases	Login, Log Out	
Stakeholders	Director	
Happy Pathway	Data for the MSCSIS annals is recorded	
Preconditions	N/A	
Post-Conditions	Capstone is completed	
Flow of Events	**Actor**	**System**
	1. Director logs into the system 2. Program director opens the capstone on the Manage MSCSIS Annals tab 3. Director enters the data 4. Director clicks the Save button	5. Data is saved in the database 6. Director clicks Log Out
Alternate Paths	N/A	
Exception Conditions	System Timeout. No data is stored	

Use Case	**System Administrator Manages Students**	
Description	Student records are maintained by the administrator that record data for each student such as GMAT/GRE scores, undergraduate university/degree, etc.	
Frequency	Episodic: 10-30/semester	
Actors	System Administrator, Director	
Related Use Cases	Login, Log Out	
Stakeholders	System Administrator	
Happy Pathway	User adds new student records to the system	
Preconditions	N/A	
Post-Conditions	Student record is created	
Flow of Events	**Actor**	**System**
	1. Administrator logs into the system 2. Administrator selects a student record and enters data 3. System Administrator clicks Save button	4. Student data is recorded 5. Administrator clicks Log Out
Alternate Paths	N/A	
Exception Conditions	System Timeout. No data is stored	

Use Case	**System Administrator Manages Faculty**	
Description	New faculty records must be added into the system and existing faculty records maintained	
Frequency	Episodic: 10-30/semester	
Actors	System Administrator, Director	
Related Use Cases	Login, Log Out	
Stakeholders	System Administrator, Committee Member, Committee Chairperson	
Happy Pathway	User creates new faculty record or updates existing faculty record	
Preconditions	N/A	
Post-Conditions	Faculty record is created or updated	
Flow of Events	**Actor**	**System**
	1. Administrator logs into the system 2. Administrator selects a faculty record for edit or clicks the Add Faculty button 3. Administrator enters data 4. System Administrator clicks Save button	5. Faculty record is created or updated 6. Administrator clicks Log Out
Alternate Paths	User updates faculty record data	
Exception Conditions	System Timeout. No data is stored	

Appendix F. Event List

Subject	Verb	Object	Frequency	Arrival Pattern	Response	Use Case
Student	Manages	Capstone project or thesis	10-30 per Semester	Episodic	Data recorded	Student Manages Capstone
Student	Requests	Committee chairperson	10-30 per Semester	Episodic	Request email is sent to chairperson	Student Requests Chairperson
Student	Requests	Committee members	10-30 per Semester	Episodic	Request email is sent to members	Student Request Committee Members
Student	Schedules	Capstone Event	10-30 per Semester	Episodic	Data is recorded	Student Requests Event Date
Student	Views	Capstone progress	10-30 per week	Episodic	Data is displayed	Student Views Capstone Homepage
Student	Manages	LinkedIn Announcement	10-30 per week	Episodic	URL is displayed, Email is sent to committee	Student Records LinkedIn Announcement
Committee Member	Manages	Membership request	10-30 per week	Episodic	Notification email is sent to student	User Accepts email request, User declines email request, User repeats email request
Committee Chairperson	Manages	Membership request	10-30 per Semester	Episodic	Data recorded, email is sent to student	User Accepts email request, User declines email request, User repeats email request
Program Director	Manages	Entry in the MSCSIS annals	10-30 per Semester	Episodic	Data recorded	Director Manages Capstone Annals
Program Director	Views	Capstone Progress	10-30 per Semester	Episodic	Data is displayed, email is sent to student	Director Manages Capstones
Program Director	Views	Capstone Report	10-30 per Semester	Episodic	Data is displayed	Director Manages Capstones
Program Director	Manages	Capstones	10-30 per Semester	Episodic	Data recorded	Director Manages Capstones, Director Approves Registration Request
System Administrator	Maintains	Student	10-30 per Semester	Episodic	Data recorded	System Administrator Manages Students
System Administrator	Maintains	UNCW Faculty/Staff	1-5 per Semester	Episodic	Data recorded	System Administrator Manages Faculty

Appendix G. Federated Login using Google OAuth

AuthenticationEndPoint.aspx.cs

```
using DotNetOpenAuth.OpenId.Extensions.AttributeExchange;
using DotNetOpenAuth.OpenId.RelyingParty;
using System;
using System.Security;
using System.Web;
using System.Web.Security;
using System.Web.UI;

namespace CapstoneMgmt
{
  public partial class AuthenticationEndpoint : System.Web.UI.Page
  {
    // The query string variable that comes back from Google
    private const string CALLBACK_PARAMETER = "callback";
    // The page to return after authentication is complete
    private const string RETURNURL_PARAMETER = "ReturnUrl";
    // the source for the request
    private const string AUTHENTICATION_ENDPOINT = "~/AuthenticationEndpoint.aspx";
    // the URL for the Google Authentication API
    private const string GOOGLE_OAUTH_ENDPOINT = "https://www.google.com/accounts/o8/id";

    FetchRequest fetch = new FetchRequest();

    protected void Page_Load(object sender, EventArgs e)
    {
      //Check if User is already authenticated and has a ReturnUrl
      if (User != null && User.Identity != null && User.Identity.IsAuthenticated
        && !String.IsNullOrWhiteSpace(Request.Params[RETURNURL_PARAMETER]))
      {
        Response.Redirect(Request.Params[RETURNURL_PARAMETER]);
      }
      else
      {
        //Check if either to handle a call back or start an authentication
        if (Request.Params[CALLBACK_PARAMETER] == "true")
        {
          //Google has performed a callback, let's analyze it
          HandleAuthenticationCallback();
        }
        else
        {
          //There is no callback parameter, so it looks like we want to sign in
          PerformGoogleAuthentication();
        }
      }
    }

    protected void PerformGoogleAuthentication()
    {
      using (OpenIdRelyingParty openid = new OpenIdRelyingParty())
      {
        //Set up the callback URL
        Uri callbackUrl = new Uri(GetCallBackURL());

        //Set up request object for Google Authentication
```

```
        IAuthenticationRequest request = openid.CreateRequest(GOOGLE_OAUTH_ENDPOINT,
DotNetOpenAuth.OpenId.Realm.AutoDetect, callbackUrl);

        //Let's tell Google what we want to have from the user: First Name, Last Name,
and Email
        SetFetchParameters();

        //Attach the fetch parameters to the request
        request.AddExtension(fetch);

        //send request to Google Authentication
        request.RedirectToProvider();
      }
    }

    public void HandleAuthenticationCallback()
    {
      try
      {
        OpenIdRelyingParty openid = new OpenIdRelyingParty();
        var response = openid.GetResponse();
        if (response == null) { throw new Exception("Authentication Response is Null"); }

        if (response.Status == AuthenticationStatus.Authenticated)
        {
          FetchResponse fetch = response.GetExtension<FetchResponse>();

          if (fetch != null)
          {
            //set the user cookie
            SetUserAuthCookie(fetch.GetAttributeValue(WellKnownAttributes.Name.First)
              , fetch.GetAttributeValue(WellKnownAttributes.Name.Last)
              , fetch.GetAttributeValue(WellKnownAttributes.Contact.Email));

            //set the forms authorization ticket/token/cookie
            FormsAuthentication.SetAuthCookie(response.ClaimedIdentifier, false);

            //Redirect the user back to the login page
            FormsAuthentication.RedirectFromLoginPage(response.ClaimedIdentifier, false);
          }
          else //we didn't fetch any info. Too bad.
          {
            throw new Exception("Authenticated, but no data returned");
          }
        }
        else
        {
          throw new Exception("Did not authenticate");
        }
      }
      catch (Exception ex)
      {
        throw new Exception(string.Format("Authentication Fail: {0}", ex.Message));
      }
    }

    private string GetCallBackURL()
    {
      //set variables using c# ternary operators
```

```
      //http://msdn.microsoft.com/en-us/library/ty67wk28.aspx

      string _protocol = Request.IsSecureConnection ? "https://" : "http://";
      string _port = Request.Url.IsDefaultPort ? String.Empty : String.Concat(":",
Request.Url.Port);
      string _host = Request.Url.Host;

      return String.Format("{0}{1}{2}{3}?{4}=true",
        _protocol, _host, _port, Page.ResolveUrl(AUTHENTICATION_ENDPOINT),
CALLBACK_PARAMETER);
    }
    private void SetFetchParameters()
    {
      //use the DotNetOpenAuth library to setup what will be fetched from the provider
      fetch.Attributes.AddRequired(WellKnownAttributes.Contact.Email);
      fetch.Attributes.AddRequired(WellKnownAttributes.Name.First);
      fetch.Attributes.AddRequired(WellKnownAttributes.Name.Last);
    }
    private void SetUserAuthCookie(string FirstName, string LastName, string Email)
    {
      //set the authorized user cookie and the values
      HttpCookie AuthorizedUserCookie = new HttpCookie("AuthorizedUserSettings");
      AuthorizedUserCookie["FirstName"] = FirstName;
      AuthorizedUserCookie["LastName"] = LastName;
      AuthorizedUserCookie["Email"] = Email;
      AuthorizedUserCookie.Expires = DateTime.Now.AddHours(1);
      Response.Cookies.Add(AuthorizedUserCookie);

    }

  }
}
```

Appendix H. System Email Templates

Student Registration Request

To: **Director**

Subject: **MSCSIS Capstone Registration**

{Student Name} has completed the registration process for the CMS. Please login and confirm {Student Name} as a MSCSIS student.

Click Here to login

This is a system generated email. Please do not reply.

Student Registration Response

To: Student

Subject: MSCSIS Capstone Management System

Congratulations! The program director for the UNCW MSCSIS program has approved your registration for the capstone management system. Please login to continue.

Click Here to login

This is a system generated email. Please do not reply.

Committee Chairperson Request

To: MSCSIS Faculty Member

Subject: {Student Name} Capstone Chairperson Request

Hello,

Please consider becoming my chairperson for my capstone committee.

Title: {Capstone Title}

Click this link to accept: I accept

Click this link to decline: I decline

Click Here to login

This is a system generated email. Please do not reply.

Committee Chairperson Response (Declined)

To: Student

Subject: MSCSIS Capstone Committee Request Response

Your capstone committee chairperson request has been declined by {Chairperson Name}. Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Committee Chairperson Response (Accepted)

To: Student

Subject: MSCSIS Capstone Committee Request Response

Congratulations! Your capstone committee chairperson request has been accepted by {Chairperson Name}. Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Committee Member Request

To: Selected UNCW Faculty Member(s)

Subject: {Student Name} Capstone Committee Member Request

Hello,

Please consider becoming a member of my committee.

Title: {Capstone Title}

Click this link to accept: I accept

Click this link to decline: I decline

Click Here to login

This is a system generated email. Please do not reply.

Committee Member Response (Decline)

To: Student

Subject: MSCSIS Capstone Committee Request Response

Your capstone committee faculty member request has been declined by {Faculty Name}. Please login to the capstone management system to continue

Click Here to login

This is a system generated email. Please do not reply.

Committee Member Response (Accepted)

To: Student

Subject: MSCSIS Capstone Committee Request Response

Congratulations! Your capstone committee faculty member request has been accepted by {Faculty Name}.. Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Committee Issued For Approval

To: {Chairperson}

Subject: {Student Name} Capstone Committee Approval Request

Hello,

Please review the capstone committee below and click the appropriate link to accept or decline

Click this link to accept: I accept

Click this link to decline: I decline

Committee

(Chairperson) {Chair Person Name}

{Committee Member Name 1}

{Committee Member Name 2}

{Committee Member Name N}

This is a system generated email. Please do not reply.

Committee Approval Request (Declined)

To: Student

Subject: MSCSIS Capstone Committee Approval Request Response

Your capstone committee chairperson has declined your committee. Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Committee Approval Request (Accepted)

To: Student

Subject: MSCSIS Capstone Committee Approval Request Response

Congratulations! Your capstone committee chairperson has approved your committee. Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Proposal Date Request

To: Chairperson

Subject: MSCSIS Capstone Committee Approval Request Response

Hello,

Please review the capstone request for proposal below and click the appropriate link to accept or decline

Click this link to accept: I accept

Click this link to decline: I decline

Capstone: {Capstone Title}

Event: {Proposal Date/Time & Location}

This is a system generated email. Please do not reply.

Proposal Approval Request (Declined)

To: Student

Subject: MSCSIS Capstone Proposal Approval Request Response

Your request for proposal date has been declined by {Faculty Name}. Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Proposal Approval Request (Accepted)

To: Student

Subject: MSCSIS Capstone Proposal Approval Request Response

Congratulations! Your request for proposal date has been approved by {Faculty Name}. Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Proposal Event Email (.ics file attached for calendar)*

To: Capstone Committee

Subject: Proposal for {Student Name} Capstone

Title: {Capstone Title}

To add this appointment to your calendar, please open and save the attached file.

Click Here to login

This is a system generated email. Please do not reply.

LinkedIn Announcement Email

To: Capstone Committee

Subject: LinkedIn Announcement for {Student Name} Capstone

Hello,

You may view the LinkedIn Announcement by clicking the link: Click Here

This is a system generated email. Please do not reply.

Proposal Complete

To: Student

Subject: MSCSIS Capstone Management System

Congratulations! The program director for the UNCW MSCSIS program has approved your capstone proposal.

Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Defense Date Request

To: Chairperson

Subject: MSCSIS Capstone Committee Approval Request Response

Hello,

Please review the capstone request for Defense below and click the appropriate link to accept or decline

Click this link to accept: I accept

Click this link to decline: I decline

Capstone: {Capstone Title}

Event: {Defense Date/Time & Location}

This is a system generated email. Please do not reply.

Defense Approval Request (Declined)

To: Student

Subject: MSCSIS Capstone Defense Approval Request Response

Your request for defense date has been declined by {Faculty Name}. Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Defense Approval Request (Accepted)

To: Student

Subject: MSCSIS Capstone Defense Approval Request Response

Congratulations! Your request for defense date has been approved by {Faculty Name}. Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Defense Event Email (.ics file attached for calendar)*

To: Capstone Committee

Subject: Defense for {Student Name} Capstone

Title: {Capstone Title}

To add this appointment to your calendar, please open and save the attached file.

Click Here to login

This is a system generated email. Please do not reply.

Defense Complete

To: Student

Subject: MSCSIS Capstone Management System

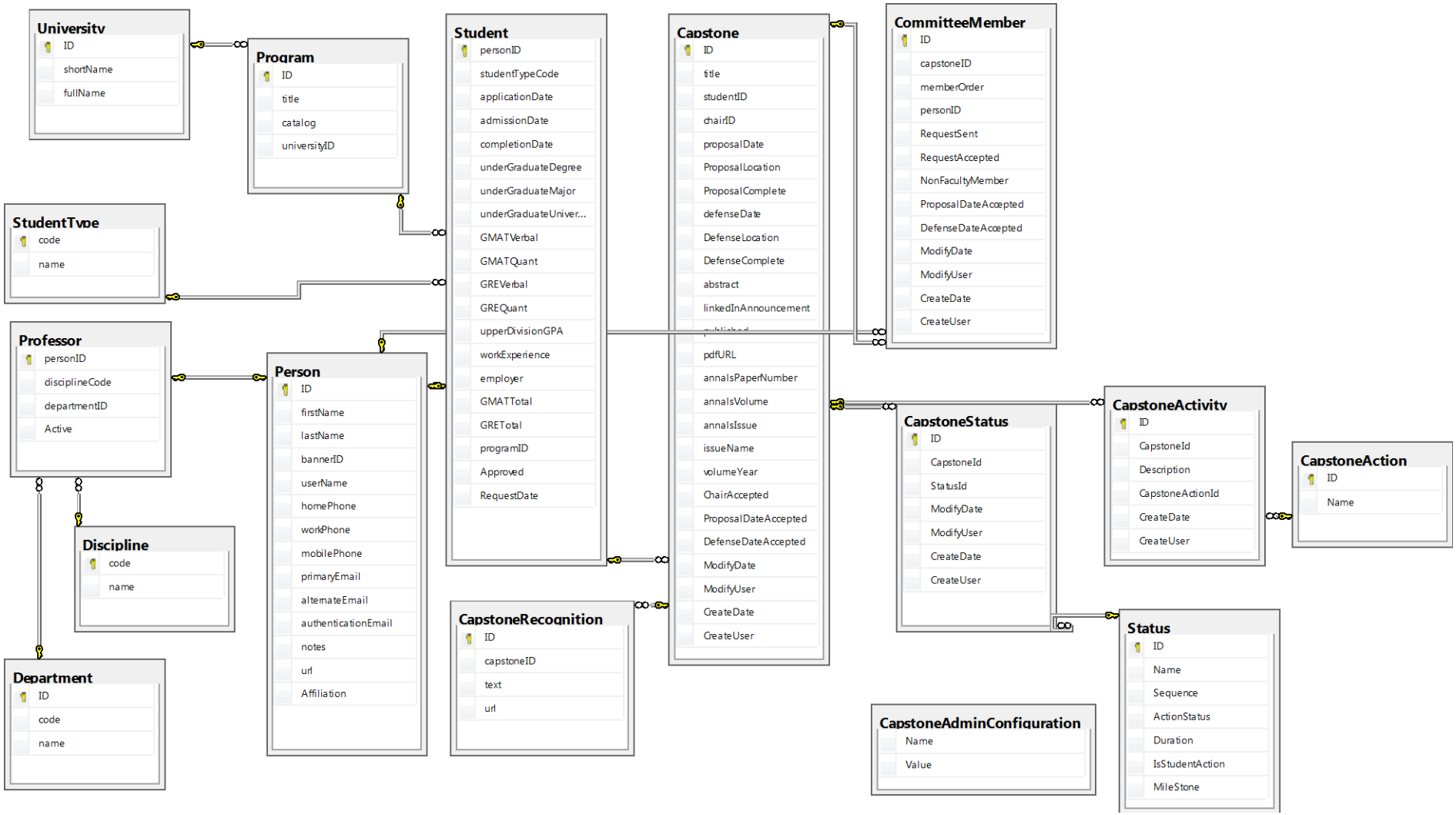
Congratulations! The program director for the UNCW MSCSIS program has approved your capstone final defense.

Please login to the capstone management system to continue.

Click Here to login

This is a system generated email. Please do not reply.

Appendix I. Database Diagram



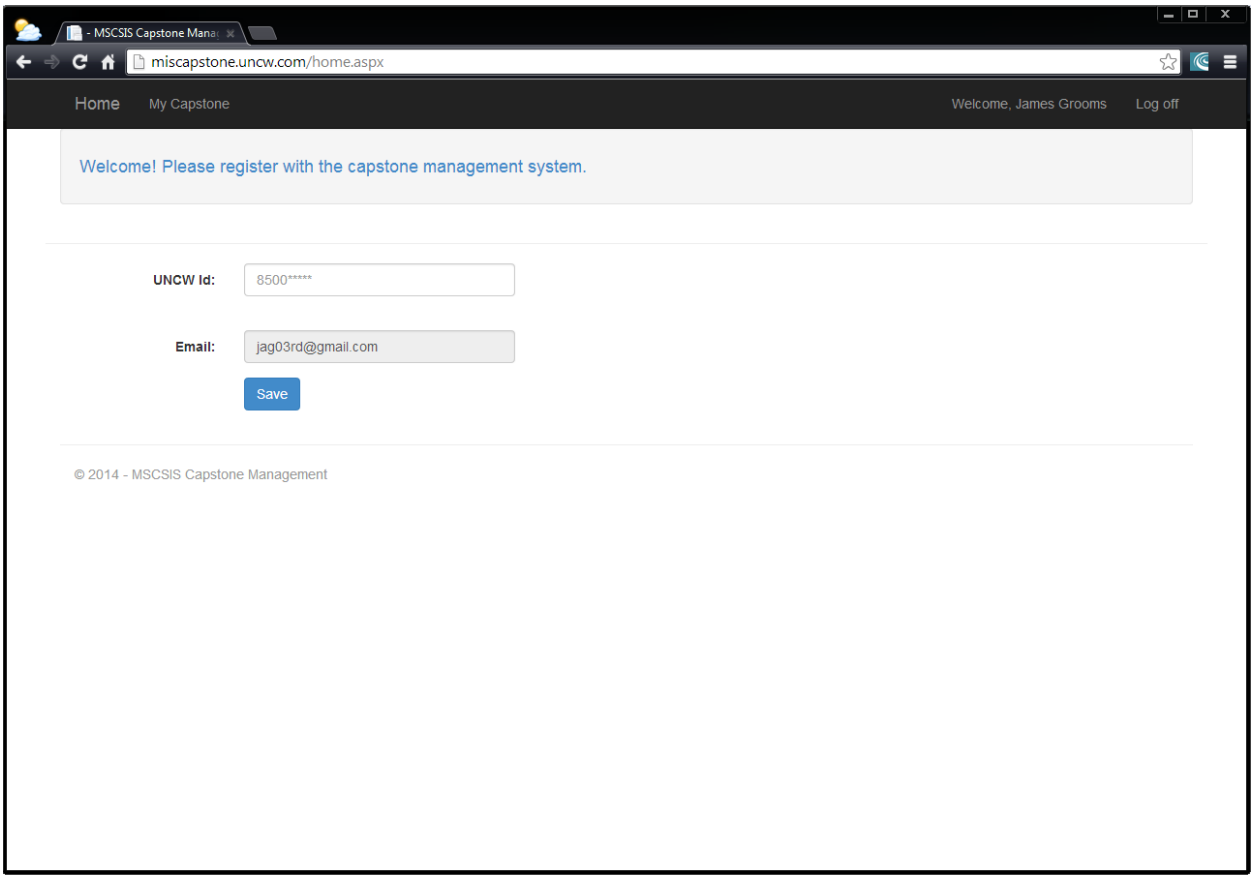
Appendix J. A User's Guide: Student

Login to the system with your Google login by opening a web browser and navigating to: http://miscapstone.uncw.edu/mscsis/default.aspx. Click the login in with Google image at the top of the page. 

The navigation bar at the top of the screen now has a menu option for My Capstone

First Time Login:

A registration form must be completed by the student on their first login. Students request registration and are approved by the program director:



Success! You have completed the registration process. Look for an email soon indicating the program director has accepted your registration.

Capstone Title

Begin your capstone by entering the title.

Edit your capstone title using the input provided. Click the edit icon  to enable the title box. Enter your title and click the save icon  to save the changes.

Status

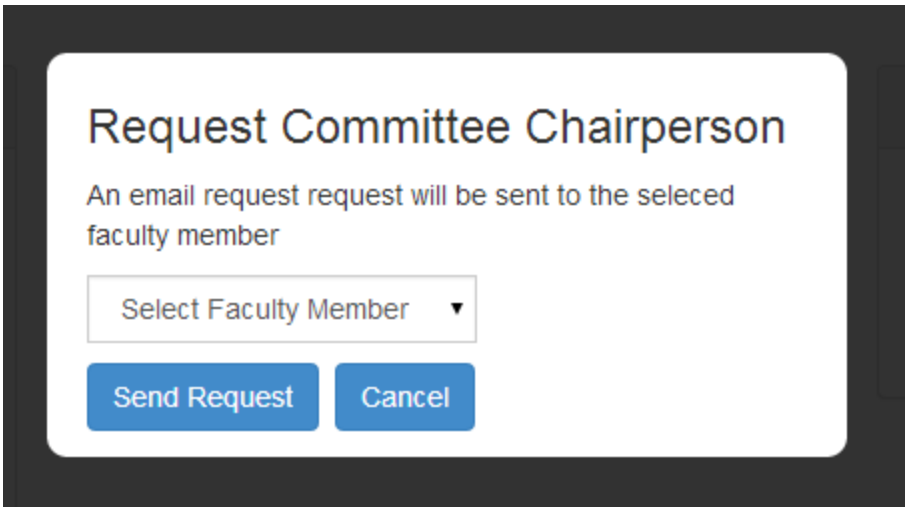
The status panel displays the workflow steps the capstone must complete. Some status steps require student actions and are displayed as hyperlinks in the sytem. Each link opens a modal window with a form to complete the workflow step.

Some workflow steps are compelted by the other sytem users are are indicated to the student with a waiting icon  beside the status name.

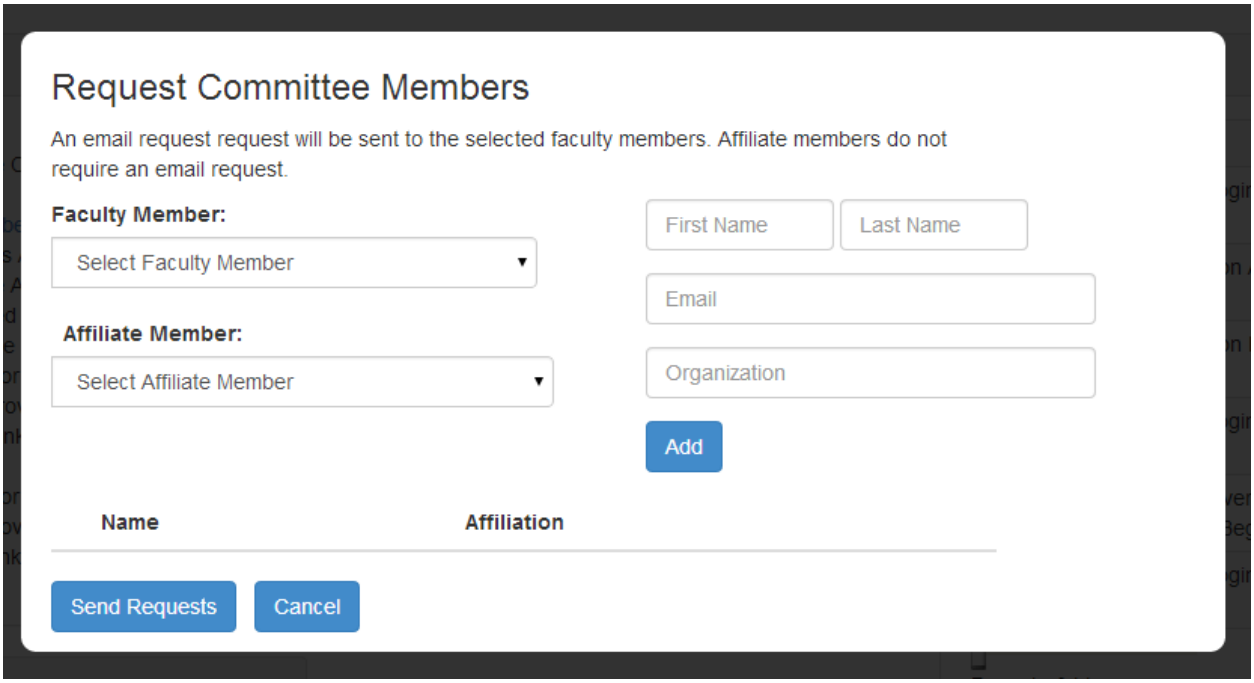
The Proposal Data approved and Defense Data Approved stauses send out an emil notificaiton to the committee along with an attached *.ics file that adds the event to their electronic calendar application (Outlook, Google, etc.).

When each status is completed a green check icon  is displayed beside that status.

- Begin Capstone
- Request Chair

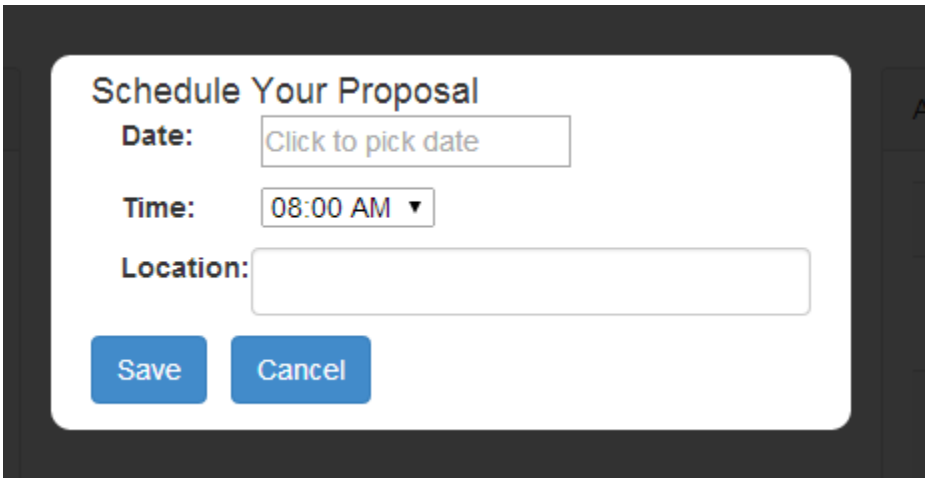


- Add Committee Members

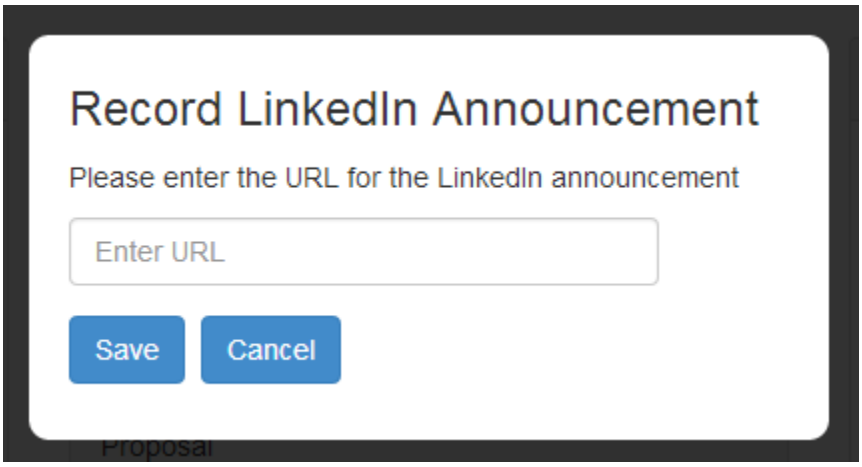


- Committee Members Accepted
- Request Committee Approval
- Committee Approved By Chair

- Committee Complete
- Request Approval for Proposal Date

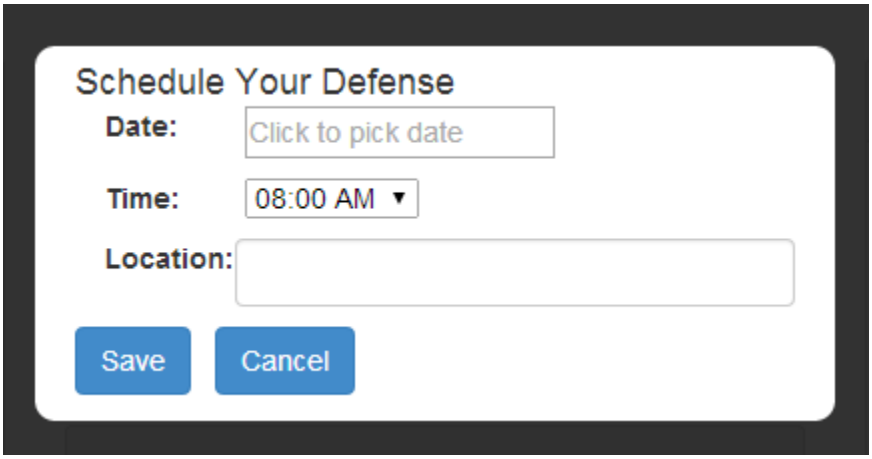


- Proposal Date Approved
- Post Proposal on LinkedIn

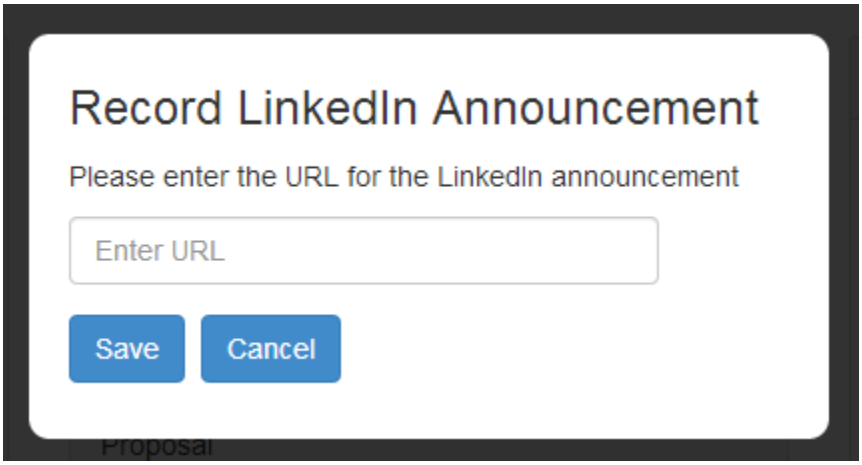


- Proposal Complete

- Request Approval for Defense Date

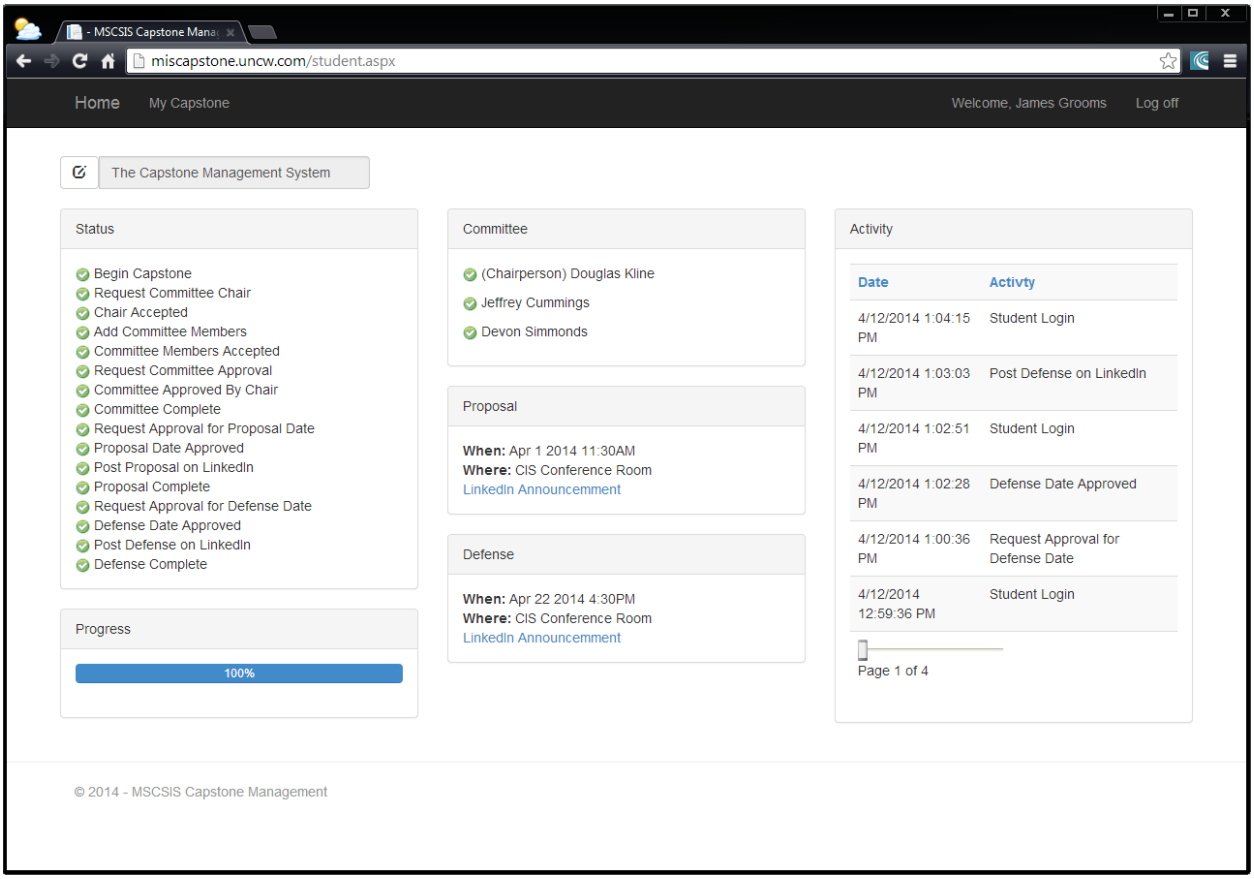


- Defense Date Approved
- Post Defense on LinkedIn



- Defense Complete

A completed capstone example:



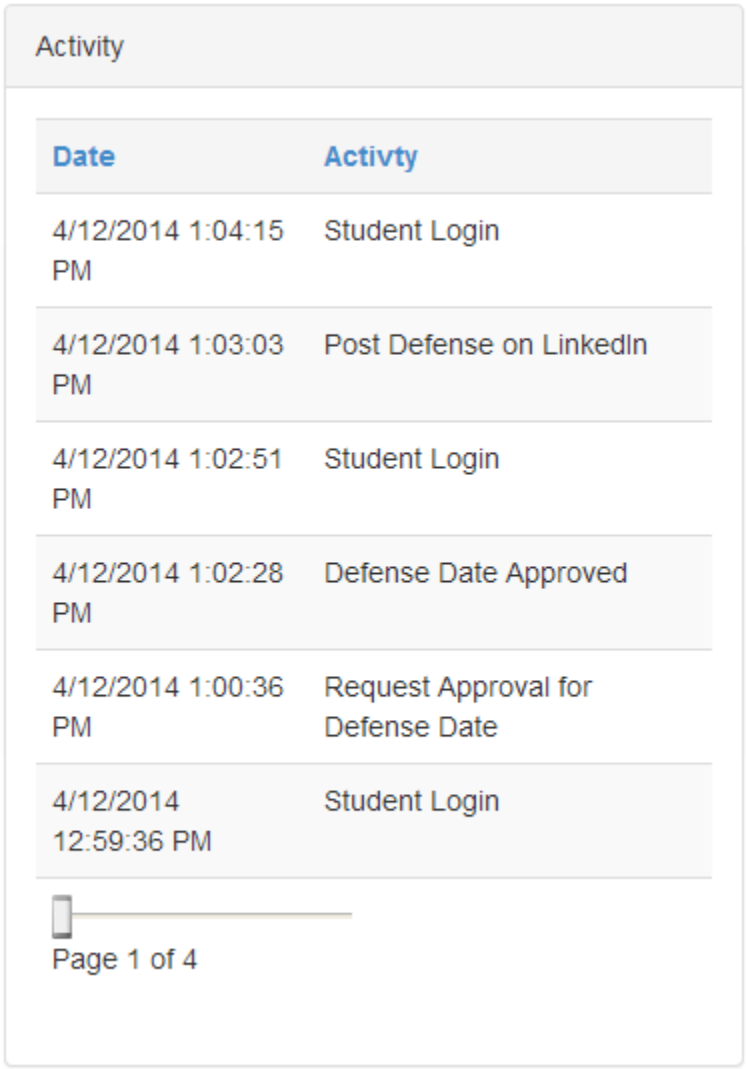
Committee

The committee panel displays the committee members of the capstone.

A green check icon beside a committee member's name indicates the committee member has accepted the request to be on the capstone committee. A waiting icon beside a committee member's name indicates a request has not been accepted.

Activity

The activity panel displays all the actions captured during the lifecycle of the capstone.

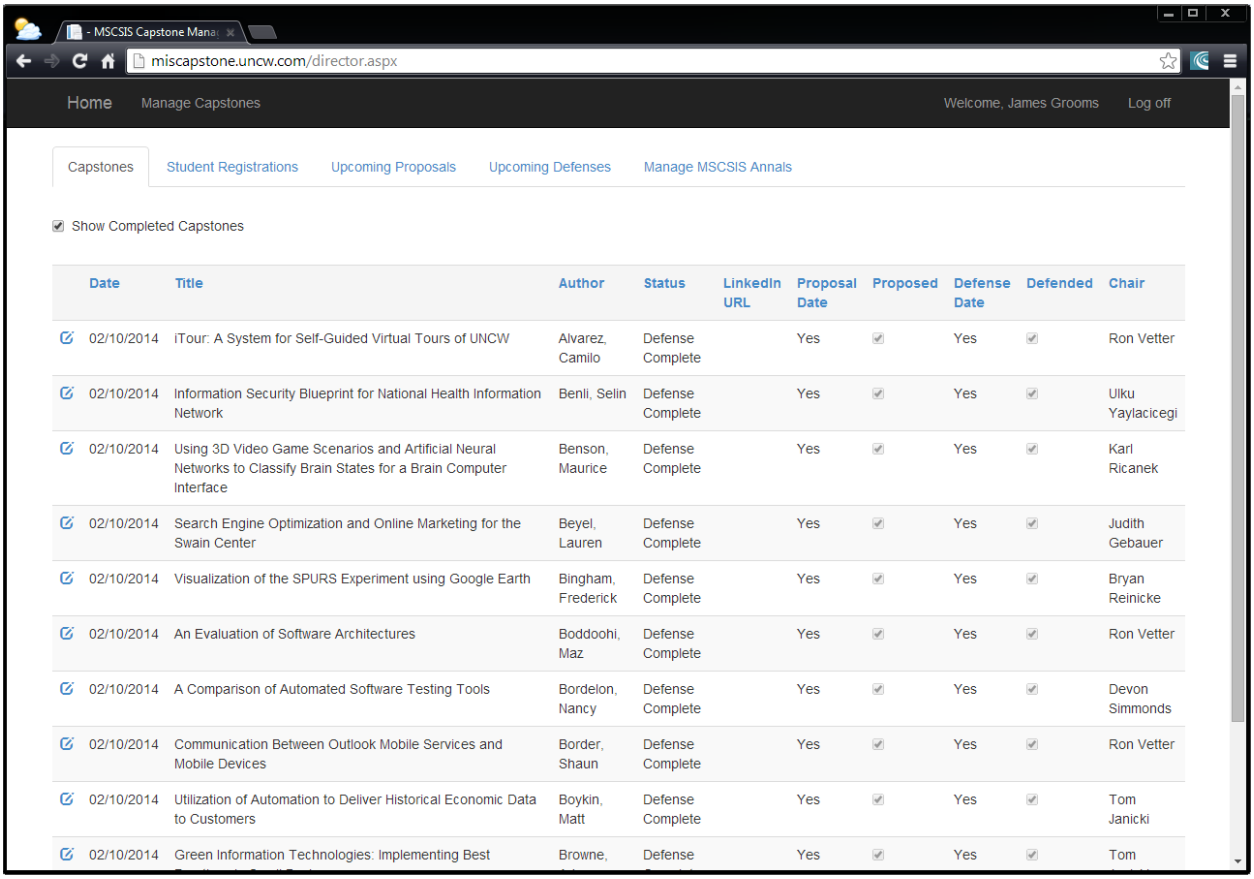


Activities are added for each student action as well as the status changes throughout the workflow.

Appendix K. A User's Guide: Program Director

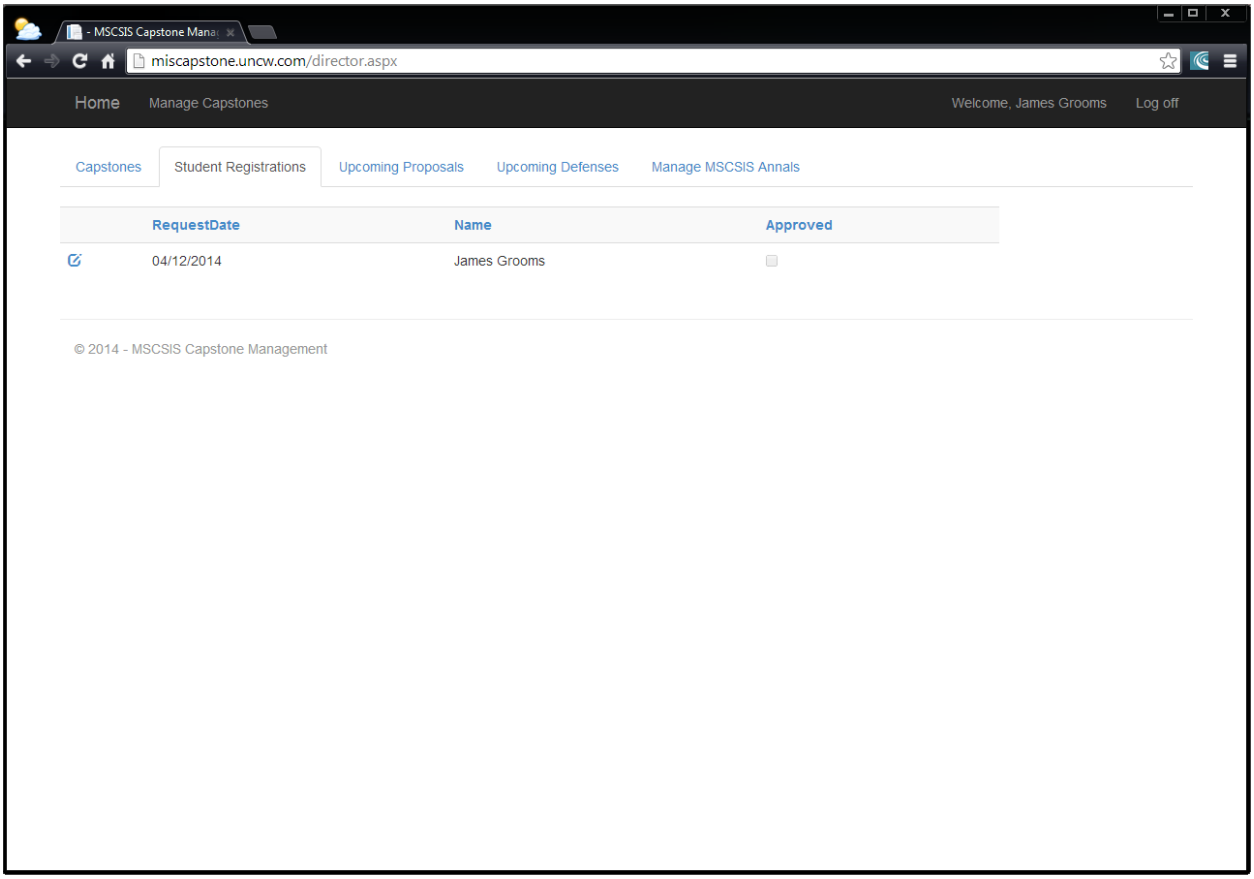
Login to the system with your Google login by opening a web browser and navigating to: http://miscapstone.uncw.edu/mscsis/default.aspx. Click the login in with Google image at the top of the page. 

The navigation bar at the top of the screen now has a menu option for Manage Capstones



Student Registrations:

An email notification is generated for each student registration prompting the program director to login to the CMS in order to accept the registration. Once logged in to the system the program director need only select the Student Registrations tab to view the pending registration requests.



Click the edit icon on each student row to enable editing. Check the box in the Approved column and click the save icon to complete the registration approval. Click the cancel icon

 to cancel the editing of the student record. Only students with pending registrations are displayed on the Student Registrations tab.

Capstone Management:

The program director is responsible for marking each capstone as proposed and defended. He may view the upcoming proposals and defenses on the corresponding tabs in the interface. The Capstones tab displays all the capstones in the system in a tabular format also called a grid. Capstones that have been completed are not displayed by default. The director may filter the grid of open capstones using the select box provided. Clicking the Show Completed Capstones checkbox will change the data in the grid to only those capstones with a Defense Complete status.

*Proposal /Defense Completed***:** To mark a capstone as proposed or defended simply navigate to the student's record on the Capstones tab and click the edit icon  to put the record into edit mode. Check the check box under the Proposed or Defended heading as appropriate. Click the save icon  to save the changes. Click the cancel icon  to cancel the editing of the student record. An email is sent to the student notifying them of the proposal/defense complete.

Capstone Data: The program director may also change the title of any student's capstone as well as move a capstone to one of the milestone statuses to wither move a capstone forward or back in the workflow.

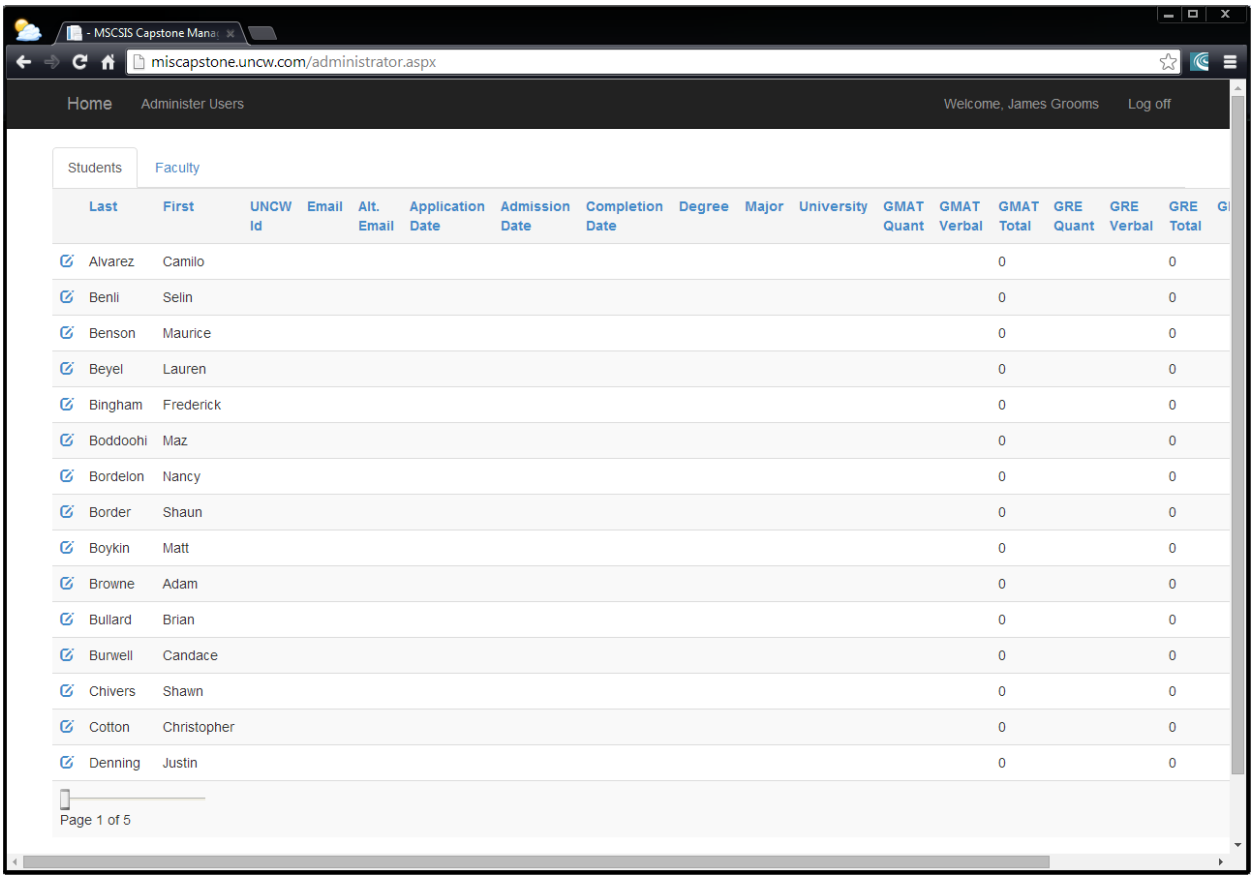
Manage MSCSIS Annals

The program director captures the data attributes for completed capstones using a grid. Click the edit icon to put the record into edit mode. Enter the data into the text fields and click the save icon to save the changes. Click the cancel icon to cancel the editing of the record.

Appendix L:. A User's Guide: System Administrator

Login to the system with your Google login by opening a web browser and navigating to: http://miscapstone.uncw.edu/mscsis/default.aspx. Click the login in with Google image at the top of the page. 

The navigation bar at the top of the screen now has a menu option for Administer Users



Student Administration

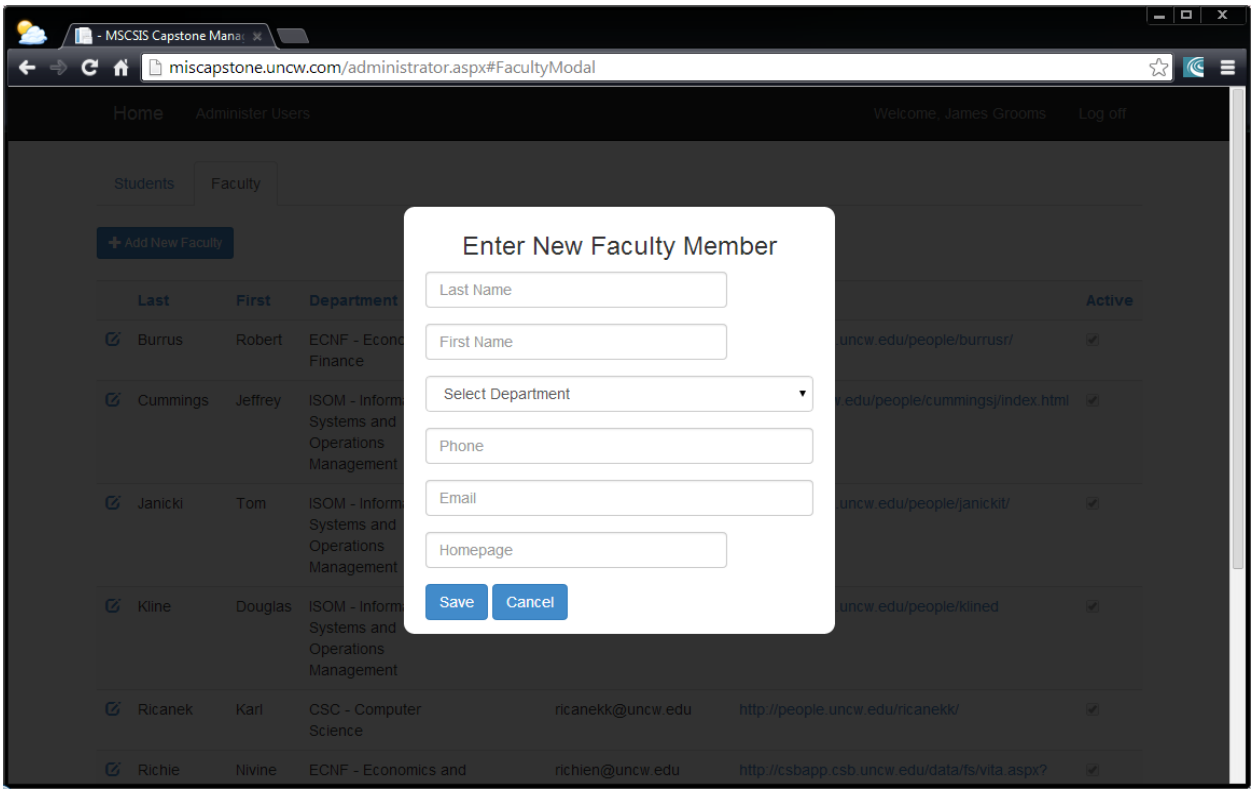
The system administrator is responsible for maintaining the data associated with each student in the CMS. Click the edit icon to put the record into edit mode. Enter the data into the text fields and click the save icon to save the changes. Click the cancel icon to cancel the editing of the record.

Faculty Administration

The system admin is also responsible for adding and maintaining the faculty members in the CMS. Use the add faculty 

button and enter the faculty information presented in the modal window. To edit a record click the edit icon to put the record into edit mode. Enter the data into the text fields and click the save icon to save the changes. Click the cancel icon to cancel the editing of the record.

Add New Faculty: Modal Window



Appendix M: System Architecture

User Interface / Application

ASPX Pages:
Default.aspx/Default.aspx.cs
Student.aspx/Student.aspx.cs
Director.aspx/Director.aspx.cs
Adminisrtrator.aspx/Administrator.aspx.cs
EmailResponse.aspx/EmailResponse.aspx.cs
Home.aspx/Home.aspx.cs
Site.Master/Site.Master.cs

Bootstrap UI files:
Content/bootstrap.min.css
Content/bootstrap-theme.min.css
Content/modal.css
Content/Site.css
Script/bootstrp.min.js
Script/jquery-1.10.2.min.js
Script/modernizr-2.6.2.js
Script/respond.min.js

Business Logic Layer

CSB_APP

Utility Classes:
App_Start/CapstoneEmail.cs
App_Start/CapstoneCommitee.cs
App_Start/CapstoneStatus.cs
App_Start/Activity.cs

Global Enumerations / Settings:
App_Start/AppConfig.cs
Web.config

Federated Login:
AuthenticatonEndpoint.aspx
AuthenticatonEndpoint.aspx.cs

Data Access Layer

Datasets:
Activity.xsd
Adminitrator.xsd
Capstone.xsd
Committee.xsd
Config.xsd

Stored Procedures