Streamlining the Capstone Process: A Time-Saving Approval System For Graduate Theses/Projects

James Grooms
jag03rd@gmail.com
Engineering Software Solutions
Wilmington NC 28401

Douglas Kline
klined@uncw.edu

Jeffrey Cummings
cummingsj@uncw.edu

Information Systems / Operations Management
University of North Carolina Wilmington
Wilmington NC 28403

Abstract

Capstones have become an integral part of many information systems programs, both at the undergraduate and graduate level. One of the challenges can be tracking the process from the start of the capstone to completion. This paper describes the analysis, design and implementation of a web application for the approval workflow of a master's program in information systems. The system replaces a paper form based process that was confusing, time-consuming, and error-prone. The system uses asynchronous JavaScript, responsive design, and clickable email links to provide a native-like look-and-feel on mobile devices, and reduce approval time. Student statuses are stored in a relational database, and program-level reports are provided for administrative decision making.

Keywords: capstone management, software development, web application, mobile development, responsive design, asynchronous JavaScript, single page application.

1. INTRODUCTION

Program capstones are common occurrences in various levels of Information Systems degrees whether that be creating a capstone course for undergraduates (Schwieger & Surendran, 2011) or completion of a thesis/capstone for graduate programs (Kline, et al, 2012). They provide students the opportunity to expand on what they have learned in the program and apply this knowledge to a real world project (Bruhn, 2004) while often times interacting with external clients (Reinicke & Janicki, 2011). As rewarding and important as this requirement is, administration and completion of these projects can be problematic, often without a clear process (Goold, 2003, Novitzki, 2001). Furthermore, faculty supervising the capstone project often take various approaches to supervision which can cause challenges in tracking where the capstone is in the university process (Perez, et al. 2012). While there have

been some tools suggested for managing the actual project being undertaken (Olarte, et al. 2014), there still remains issues surrounding the administration of the capstone process itself. Issues include various signatures on multiple forms from faculty and directors as well as the ability to track students throughout the process.

This paper presents a system for managing the workflow of thesis/capstone project requirements for a graduate program in information systems. The system was completed as a capstone project degree requirement for the program in the spring of 2014. The project took approximately one year and incorporated technologies such as AJAX and Bootstrap, resulting in a web application that has a native look-and-feel on a wide range of devices. The main goals were to help guide students through the process, and ease the burden of obtaining approvals for the steps in the process.

For the current master's program, students have to seek approvals throughout the various stages of the thesis/capstone project. These include:

- A faculty member's agreement to chair the committee
- Multiple faculty members' agreement to be on the committee
- Committee members' agreement to a proposal date
- Committee members' agreement to the defense date

These approvals are typically achieved through hand-written signature on a physical form. Since students only perform this once, they are unfamiliar with the process, complicating matters. The physical forms are recorded in a spreadsheet and filed in physical file folders, one for each student. This made it difficult to answer questions such as:

- Which students have completed their proposal, but not their defense? (for the director)
- How many committees am I on? (for a faculty member)
- Where is my approval form? (Who am I waiting on?) (for a student)

The remainder of the paper describes the requirements analysis, design, and implementation of a system to streamline the approval process, and provide decision making information to students, faculty and staff. Section 2 describes the current manual process, and lays out the requirements of the new system. Section 3 presents the architecture of the completed system, and the technical design decisions made along the way. Section 4 reviews the completed system and its implementation process, as well as the resulting benefits.

2. ANALYSIS

Current Process

The legacy process involved the following forms, in number sequence, with required signatures:

1. Capstone/Thesis registration form
    - Student
    - Committee Chair
    - Director
2. Establish Committee form
    - Committee Chair
    - Committee Members (2 +)
    - Director
3. Proposal Scheduling form
    - Student
    - Chair & Committee Members
    - Director
4. Proposal Approval
    - Chair & Committee Members
    - Director
5. Defense Scheduling form
    - Student
    - Chair & Committee Members
    - Director
6. Final Defense Approval
    - Chair & Committee Members
    - Director

Four of the six forms (1, 2, 3 & 5 above) are for informational use only, creating some formality and ensuring awareness of events by all stakeholders. Without the forms, miscommunication amongst the stakeholders on dates and committee membership was common. Signatures for these four forms were the responsibility of the student, entailing finding each person individually. In recent years, the forms had been created as digitally-signable Portable Document Format (pdf) files, with the intention that the file be email-routed. However, not everyone could digitally sign the document, and it would get lost in email inboxes. As a result, the pdf files were not consistently used.

The most important forms in the process are the Proposal Approval (#4 from list) and Final Defense Approval (#6 from list). Signatures on

these forms indicate completion of a degree requirement. These two forms were typically signed immediately following a proposal or defense, when all committee members were together in a single room.

After all signatures were obtained, the program coordinator entered individual form information into a common spreadsheet. The paper forms ended their trip in physical file folders, one for each student. The spreadsheet was used for quick one-student status lookups such as "Has Jane Smith proposed? Who is her chair?"

The legacy manual process had many problems. First and foremost was trying to achieve compliance from the students and faculty. Even though the processes were well documented and in a Student Handbook, many students and faculty were recurrently unaware of the forms and processes to complete the capstone. Faculty would forget whose committees they were on. Students had difficulty obtaining signatures promptly. Forms would remain on a faculty member's desk for weeks, or get lost entirely. Forms were commonly filled out after-the-fact, which is contradictory toward the purpose of the forms -- to notify of future events. Notification of future events also affected the degree requirement that proposals and defenses be public. We had to ask: If there is no future notification of a defense, can we still consider it to be public?

Another problem of the legacy process was a lack of reporting capability. There was no easy way to aggregate across students for reporting and decision making. For example, "how many students have proposed but not defended?" would require a manual count in the spreadsheet (if it was up-to-date) or pulling all students files individually.

In summary, the existing process was chaotic and frustrating for all stakeholders. Thus, the decision was made to eliminate the manual process and design/build a new system for tracking and approving capstones/theses.

Design Goals

The new system was meant to benefit not only faculty and students but those responsible for tracking and coordinating the capstone/thesis process (e.g. program administrators). Under the current approach, students coordinate with the faculty and program administrators to complete their capstone project or thesis. Signatures on the paper forms document that faculty were in agreement on committee membership as well as event dates and times. However, students bore the brunt of reaching consensus among parties and gathering signatures. Because of graduation deadlines, this could potentially have serious consequences.

Because of the chaotic and inconsistent nature of the legacy process, it was not possible to establish metrics or quantify the process. Thus, we were unable to establish quantifiable goals such as "reduce the in-process time for form B by 20%". We anticipated that, based on the stakeholders' feedback, that any improvement would be welcome.

The first design goal for the system was to reduce in-process time for an approval. Students still had to reach consensus among party members, but there shouldn't be delays due to the paperwork and getting signatures. Reducing in-process time would save time for all stakeholders.

The second goal was to remove the need for physical signatures. For most of the approvals, a "legally-binding" physical signature was unnecessary; evidence of notification and approval would be sufficient. Forms 1-3 and 5 above are all examples of unnecessary physical signatures. If we could document notification and approval, then physical signatures would be unnecessary.

The third design goal was to provide some status reporting for all the stakeholders. Reports for all stakeholders needed to be based on the same information. Students needed to know their current status, and their position in the process. Faculty needed to know their committee obligations and upcoming proposals and defenses. Program administrators needed program-level aggregate reports to make decisions.

Architecture Decisions

These design goals led to several key decisions regarding the system. First, the system would be "in the cloud", in the form of a database-driven web site. A common database would keep all stakeholders up-to-date with the same information. Second, emails would "push" approval requests. The emails would have "Approve"/ "Decline" links. This would speed the process, and not require users to log in to a system to approve/decline. Third, all interfaces

would be mobile-friendly. A request and approval could happen entirely on any device including small-screen mobile phones. Again, this would speed the in-process time.

Several other requirements became apparent through the analysis. Because many committee members were not affiliated with the university, the system could not use university accounts for authentication. To overcome these challenges, a decision was made to use Google Third-party authentication using OpenAuth for the initial system, with the intention of adding others in the future, e.g., Facebook or Twitter.

For ease-of-use and more natural interactivity, it was clear that we would need to use asynchronous server calls (AJAX) (for more information on AJAX, see Garret, 2005). This would eliminate the page refreshes that can be disconcerting on a wireless device with spotty service.

Two external systems were identified as possibly needing interfaces in the new system: LinkedIn and the program mailing list. The program maintains an active LinkedIn group for alumni, current students, employers, and others. All proposal and defense announcements are posted to LinkedIn in a fairly consistent format. This is normally done by the student, but the design team choose to include this to increase automation. In addition, a mailing list is used primarily for internal program announcements to faculty and current students. Again, this is normally manually done by a program administrator, but we thought that it might be automated to some extent.

Finally, the approval system would need to integrate with an existing system that published capstone/thesis documents on the web. This publishing system lists all completed thesis in list form, provides an indexer-friendly "landing page" with abstract and citation information for each document, enables social media "likes" and sharing for each document, and cross links to faculty, students, and related publications. The new system would reside on the same web server, and share the same database.

3. DESIGN

Appendix 1 shows the workflow of the system, developed through interviews with key stakeholders. It is more formal than the paper-form process, and represents some subtleties not captured before. For example, the composition of the committee requires committee chair approval before the requests go to the faculty. Of course, it was expected that students would discuss their committee selection with their chair, but there was no enforcement of this, which led to some misunderstandings under the legacy process. Note that "decline"s or time-outs at each stage in the process merely revert the student to the previous stage.

Appendix 2 is the Actor Diagram describing the users and how they interact with the system. Note that some of the use cases are initiated from within an email, and do not require a full login by the user. The four main Actors are:

- Student
- Chairperson
- CommitteeMember
- Director

In addition to the main Actors above, we needed a user interface for a System Administrator. This would be used to monitor and manage the technical aspects of the system. The general public is the final Actor, consuming notifications and items published in LinkedIn, the mailing list, and the annals web site.

The "Student Requests Chairperson" use case is shown in more detail as part of Appendix 3. This is typical of the use cases for this system – it is short and relatively simple. There are no complex interactions. Appendix 4 includes emails that are created via the student request for chairperson. The student initiates an approval, an email is sent to the potential chairperson, who clicks on "Accept" or "Decline" links, and the action is recorded. Status update emails are sent, and the student's status changes to "Chair Accepted".

Google acts as the third-party authenticator, using the OpenAuth protocol. In cases where a full login is required, users are presented with a Google Login dialog box. Upon successful authentication, an SSL connection is established with the server, and future interaction is encrypted. Google handles changes to passwords, password reminders, etc. Users have one less username/password to remember.

Bootstrap (Otto & Thornton, 2013) was used to style the web interfaces and provide responsive screens for devices of all sizes. Screens were

primarily designed for small screens such as smart phones. For larger devices, Bootstrap elegantly expands to fill the screen and "unstack" UI components. We found that for the short use case interactions that make up most of this system, the user-preferred device was a smartphone.

AngularJS (https://angularjs.org/) was seriously evaluated and considered for use in this system. The two very attractive features of this framework were two-way binding and declarative form validation. However, the main platform of ASP.Net with C# and JavaScript met our actual needs. The user interfaces were not complex enough to require two-way binding, and the ASP.Net controls provided declarative form validation without round-trips to the server. In the end, we felt that the potential benefits of AngularJS did not outweigh the complexity of an additional framework.

The general paradigm for this system was a Single-Page Application (SPA). Technically, there are multiple pages – one for each main Actor, and for interactions requiring an initial page load (below):

- Default.aspx
- Director.aspx
- Student.aspx
- AuthenticationEndpoint.aspx
- EmailResponse.aspx

This was mainly done to organize the code base. However, the Default, Director, and Student pages could easily have been combined. All interactions within these pages were exclusively done by asynchronous JavaScript calls to the server (AJAX).

The database schema is shown in Appendix 5. Most of the tables already existed to support the document publishing system for the annals. The five tables in the lower right of the diagram were added to support the approval system. The CapstoneActivity table acts as a log, recording every action associated with every capstone. The CapstoneStatus table keeps the major status changes represented in the workflow. The CapstoneAdminConfiguration table holds system configuration information, such as the email templates, which can be modified without recompiling the system. CapstoneAction and Status hold the valid values for the related tables. In large part, the workflow and operation of the system could be modified entirely by changing table entries without changes to the code base.

An example of various screens the student interacts with is included in Appendix 6. These screens exemplify the relatively simple use cases that make up the bulk of the interactions with this system. They fit well on a small screen, and have the look-and-feel of a native application. The Request Committee Members screen represents a more complex interaction, allowing students to add committee members not already in the database, e.g., experts from the professional community. This interaction still works very well on a small screen with touch-screen data entry.

Appendix 7 shows the main status page for a student. On the left are the steps that must be completed (from top to bottom), with a green checkmark indicating completion. This screenshot represents a student who has completed the entire process. The progress bar shows percentage complete for the entire process. The center panels give summary information about the capstone. The right panel shows the log of all activity associated with this capstone.

This screen solved a significant problem for the program. Students only go through this process once in their life, and were therefore not familiar with the sequence of activities. This system lays out the process clearly from top to bottom, and forces a "lock-step" navigation through the process. Students found this extremely helpful.

The new system also includes a program director's view into the data showing the status of all capstones (in-process and completed). In the past, program-level reports were manual, and difficult to create and update. The new view shows all completed capstone and thesis documents for the entire program. This includes various tabs along the top of the screen providing program-level reports for "Upcoming Proposals", etc. These reports are always available and represent real-time status of students' progress (Note: a screenshot of the system has been removed to maintain anonymity but will be included in the final submission, if accepted).

The administrator user interface (not shown) includes screens for entering users (students

and faculty), and manually recording or modifying information.

The overall architecture of the system is shown in Appendix 8. The entire system is broken into three main layers: User Interface, Business Logic, and Data Access. The User Interface has surprisingly few pages compared with traditional web-form development. The Business Logic layer consists of utility classes that are loaded on application startup, configuration, and pages to support Google OpenAuth Federated login.

The Data Access layer consists of datasets XML Entity Framework datasets, and sixty eight (68) stored procedures. The stored procedures reside on a MS SQL Server database server and are written in Transact-SQL. Each one is very specific, and performs minimal database operations. Transactional consistency is enforced with BEGIN TRAN, COMMIT TRAN, and ROLLBACK TRAN directives, and appropriate transaction isolation levels.

4. CONCLUSION/DISCUSSION

The system was well received by both students and faculty. While presenting the system, the entire workflow was completed by a student and committee in about 5 minutes, with all participants on smart phones. The participants were not trained ahead of time, and had no prior experience with the system. This is compared to the average time currently to complete just one step in the process which usually takes 1 to 2 weeks. The goal of the system is to streamline the process and centralize all paperwork/process steps to eliminate loss of signed forms.

Most of the design goals were met. Physical signatures were eliminated for all forms except the Proposal and Defense approvals. These two forms represent degree program requirements, which merited more formal physical requirements. These two forms are the most easily completed, since they are generally signed at the end of the proposal and defense, when the committee is all in one room.

Third-party authentication was accomplished through Google to support participants from outside the university. This was surprisingly simple, and required minimal code. Authentication is generally a difficult part of a system and would have taken significant time and effort if it was incorporated into the scope of the project.

Overall, the look-and-feel of the final system was good and user-friendly. The out-of-the-box Bootstrap styles met most of the needs of the system with very few changes being made to these styles. On a smartphone, the pages feel like native applications and are very responsive. Receiving an email request, clicking on an approval link, and seeing the confirmation screen is smooth and easy.

The design goals of integration with LinkedIn and the mailing list was limited, mainly due to time and resource limitations. Students still had to enter their announcement on LinkedIn. They could then copy the LinkedIn announcement URL and enter it in the system. That URL could then be placed on web pages and used by program administrators to send the mailing list. Even with this limited support for the external systems, all parties saved time, and the process was more consistent.

Our main concerns for this project were time and resources. The system used cutting edge features which were new to the developer. Web-based systems are notorious for having many technologies that must work together, and many languages (JavaScript, JQuery, Bootstrap, C#, ADO.Net, SQL, T-SQL). However, the final architecture was relatively simple and elegant. Detailed, comprehensive systems analysis and design phases were extremely helpful in reducing the overall time. The implementation phase ran smoothly without significant design changes.

Possible future enhancement for the system include:

- Text notifications
- More third-party authentication support: Facebook, LinkedIn, Twitter, etc.
- File uploads
- Extension to other graduate programs at the university
- Graphical reports

Text notifications would require a Short Message Server (SMS) capability, typically through a third-party provider such as Google. Third-party authentication extension is relatively easy, but requires developer accounts, libraries, and configuration for each provider.

Providing the ability for students to upload the final document, and request approvals for it would truly complete the process. This would eliminate large email attachments and "lost in the email inbox" issues. Furthermore, documents could be instantly published to the annals website without manual processes.

From the program director's viewpoint, more aggregate reporting would be immensely helpful. For example, a pie-chart representing the current student body's status would show the "pipeline", and help with enrollment management. Time-series style graphs could show graduations over time. Analytics could show, for example, the average time-in-process by committee chairperson.

Overall, the benefits of the system are significant. Students have a clear view of the process, and are relieved of the need to track down faculty for signatures. Faculty and administrators can see the progress of students. Program-level reporting is available for better decision-making.

4. REFERENCES

Bruhn, R.E. & Camp, J. (2004). Capstone Course Creates Useful Business Products and Corporate-Ready Students. ACM SIGCSE Bulletin, 36(2), 87-92.

Garrett, Jesse James. (2005) Ajax: A new approach to web applications. Retrieved June 19, 2015 from https://courses.cs.washington.edu/courses/cse490h/07sp/readings/ajax_adaptive_path.pdf

Goold, A. (2003). Providing Process for Projects in Capstone Courses. Proceedings of the 8th Annual Conference on Innovation and Technology in CS Education, ACM SIGCSE Bulletin, 35(3), 26-29.

Kline, D.M, Vetter, R., Barnhill, K. (2012) Building an Effective Interdisciplinary Professional Master's Degree. Information Systems Education Journal, 11(6) pp 40-49.

Novitzki, J.E. (2001). Critical Issues in the Administration of an Integrating Capstone Course. Proceedings Informing Science and Information Technology Education 2001, 372-378.

Olarte, J. J., Dominguez, C., Jaime, A., & Garcia-Izquierdo, F. J. (2014). A tool for capstone project management in computer science engineering. In *Computers in Education (SIIE), 2014 International Symposium on,* 65-68.

Otto, M., & Thornton, J. (2013) "Bootstrap." Bootstrap. http://getbootstrap.com/

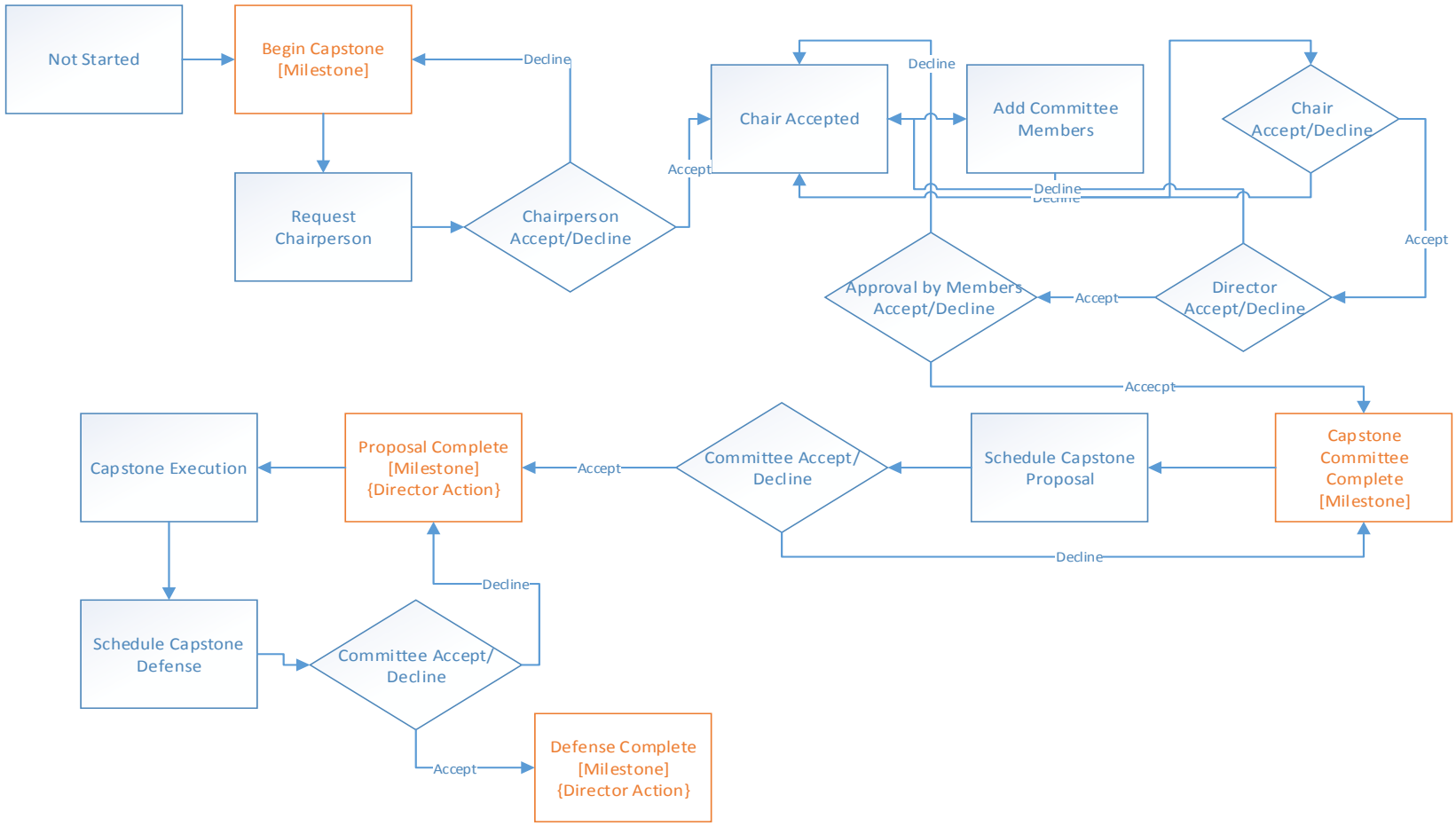
Pérez, C. D., Elizondo, A. J., Garcia-Izquierdo, F. J., & Larrea, J. J. O. (2012). Supervision typology in computer science engineering capstone projects. *Journal of Engineering Education*, *101*(4), 679-697.

Reinicke, B., Janicki, T. (2011). Real World Projects, Real World Problems: Capstones for External Clients *Information Systems Education Journal,* 9(3), 23-27

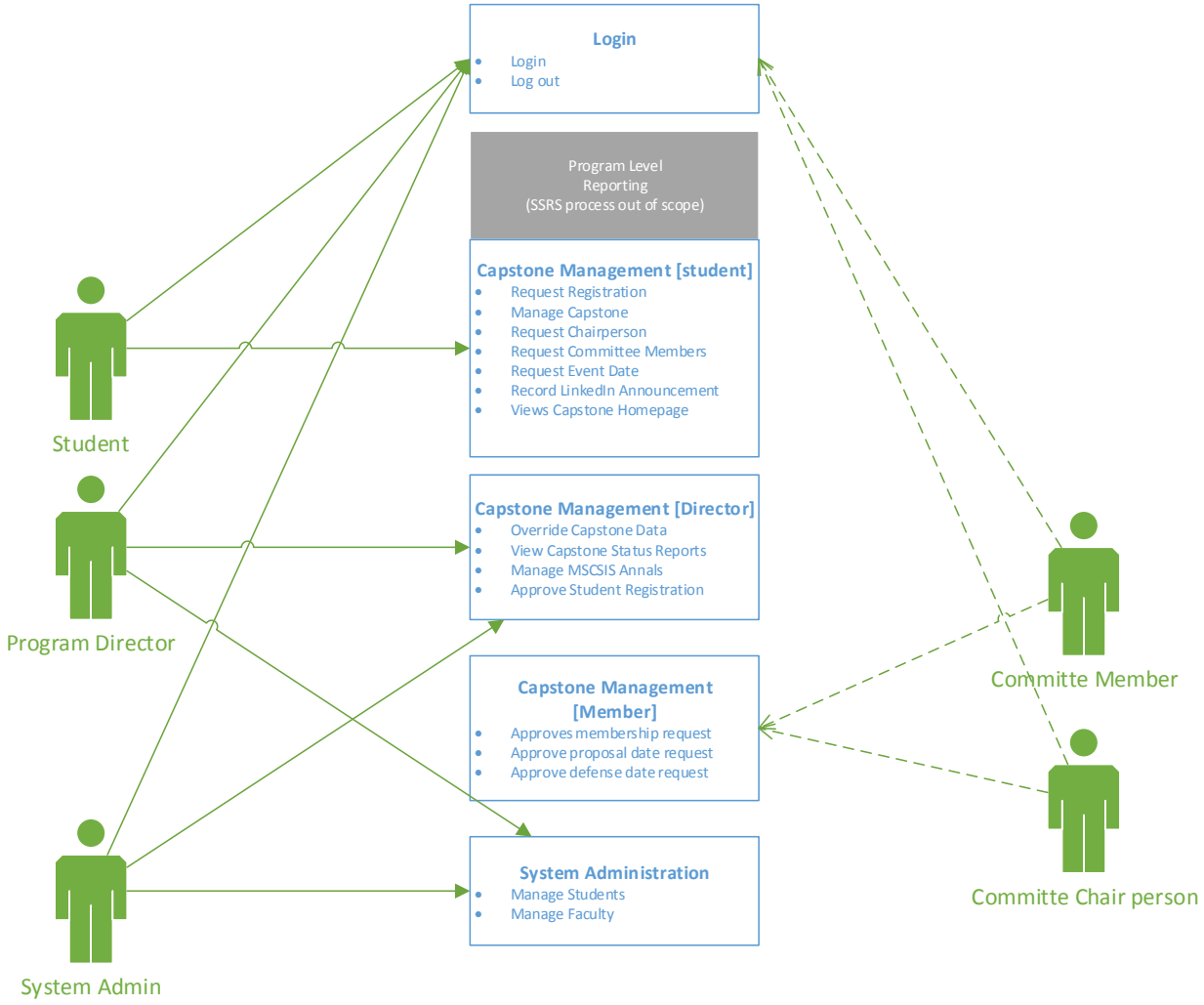
Schwieger, D., Surendran, K. (2011). Incorporating Capstone Courses in Programs Based on the IS2010 Model Curriculum. *Information Systems Education Journal,* 9(2), 65-74.

W3 Schools (2014) *AJAX Introduction*. http://www.w3schools.com/ajax/ajax_intro.asp

APPENDIX 1. CAPSTONE MANAGEMENT SYSTEM WORKFLOW



APPENDIX 2. CAPSTONE MANAGEMENT ACTOR DIAGRAM



APPENDIX 3. CHAIRPERSON REQUEST USE CASE

Use Case	**Student Requests Chairperson**	
Description	The student requests a 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 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	

APPENDIX 4. CHAIRPERSON REQUEST SYSTEM EMAILS

Committee Chairperson Request

To: 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: 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

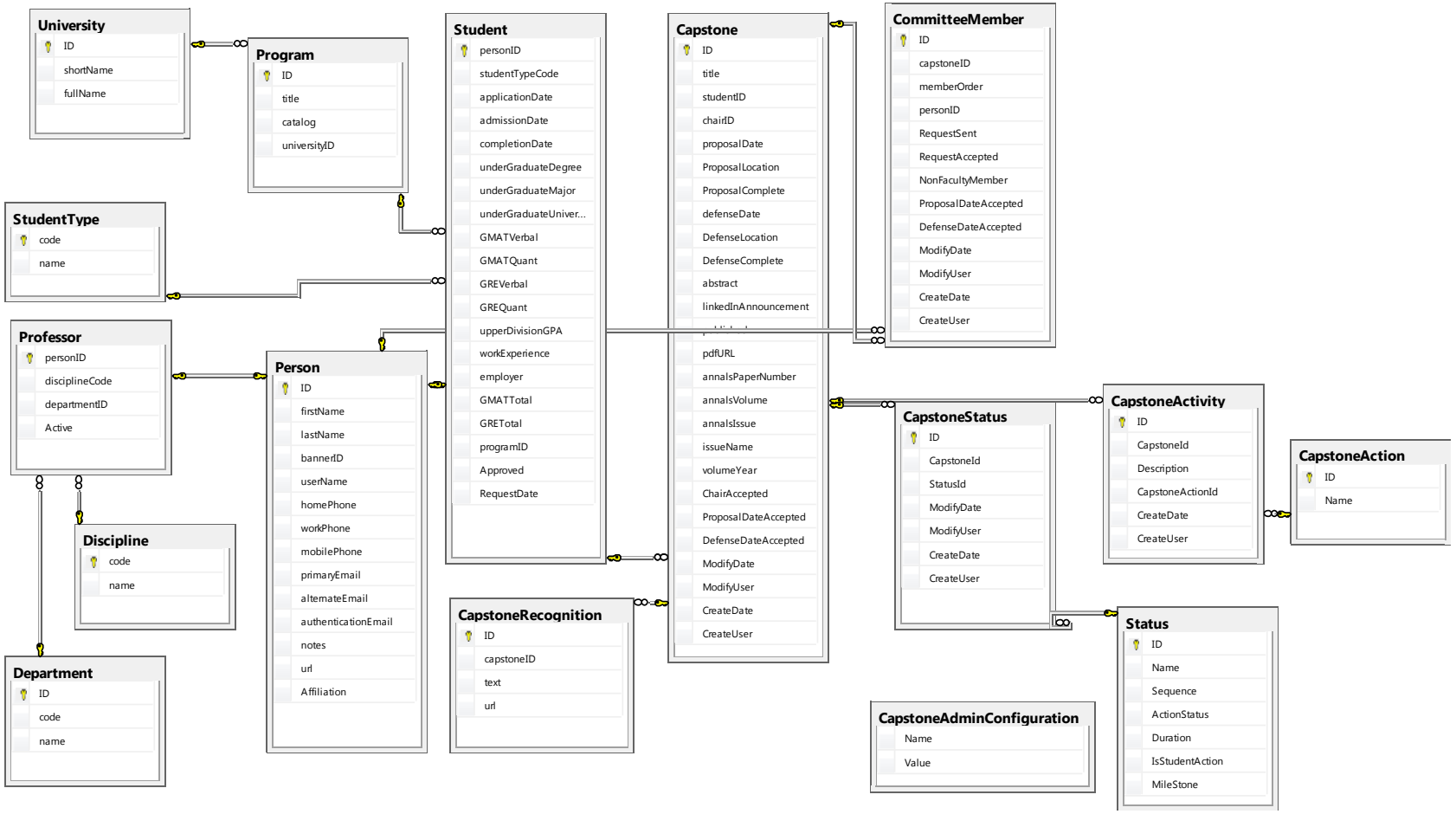
Subject: Capstone Committee Request Response

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

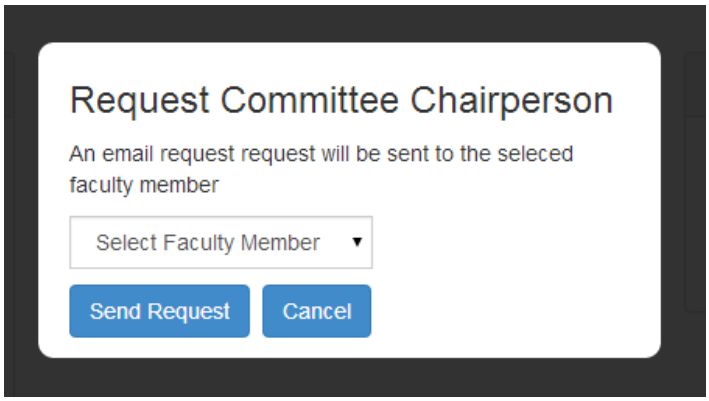
Click Here to login

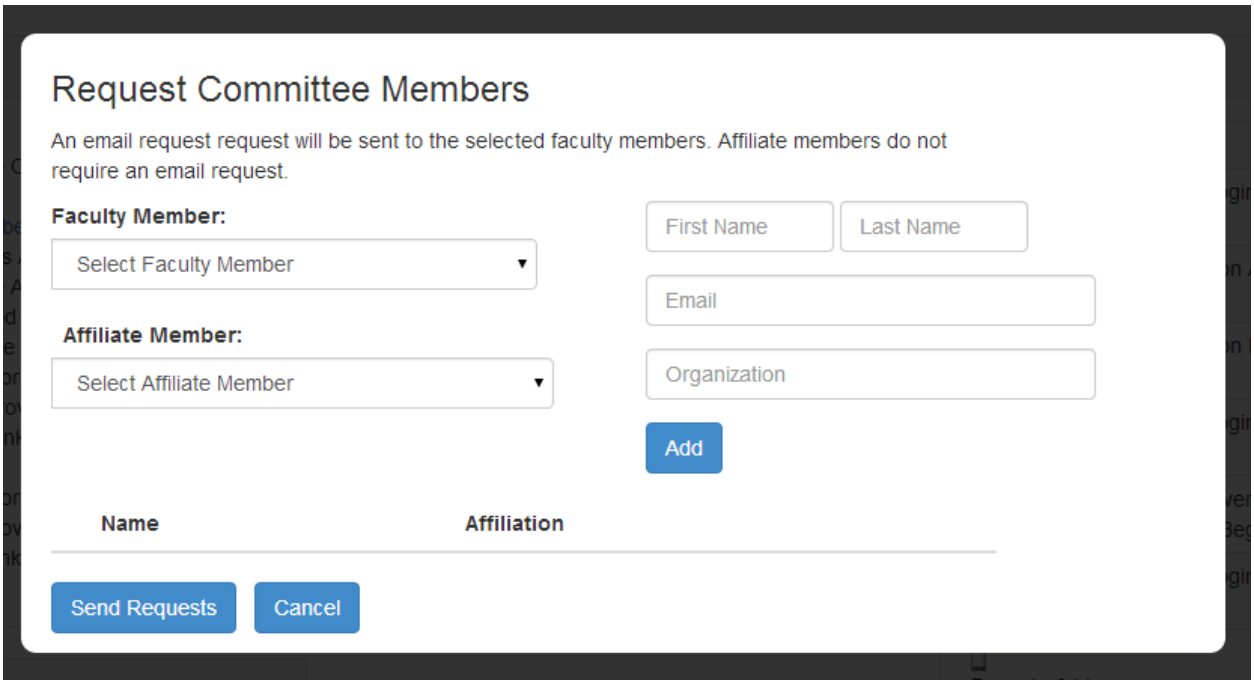
This is a system generated email. Please do not reply.

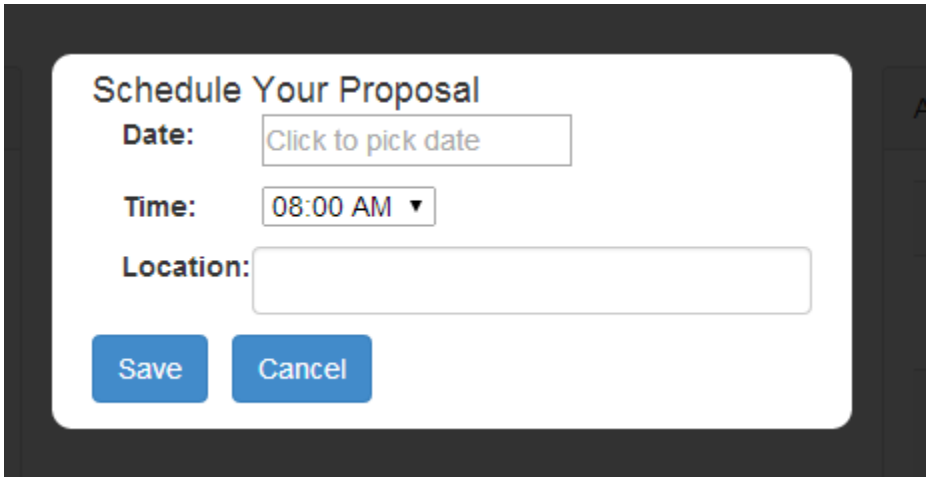
APPENDIX 5. CAPSTONE MANAGEMENT DATABASE SCHEMA



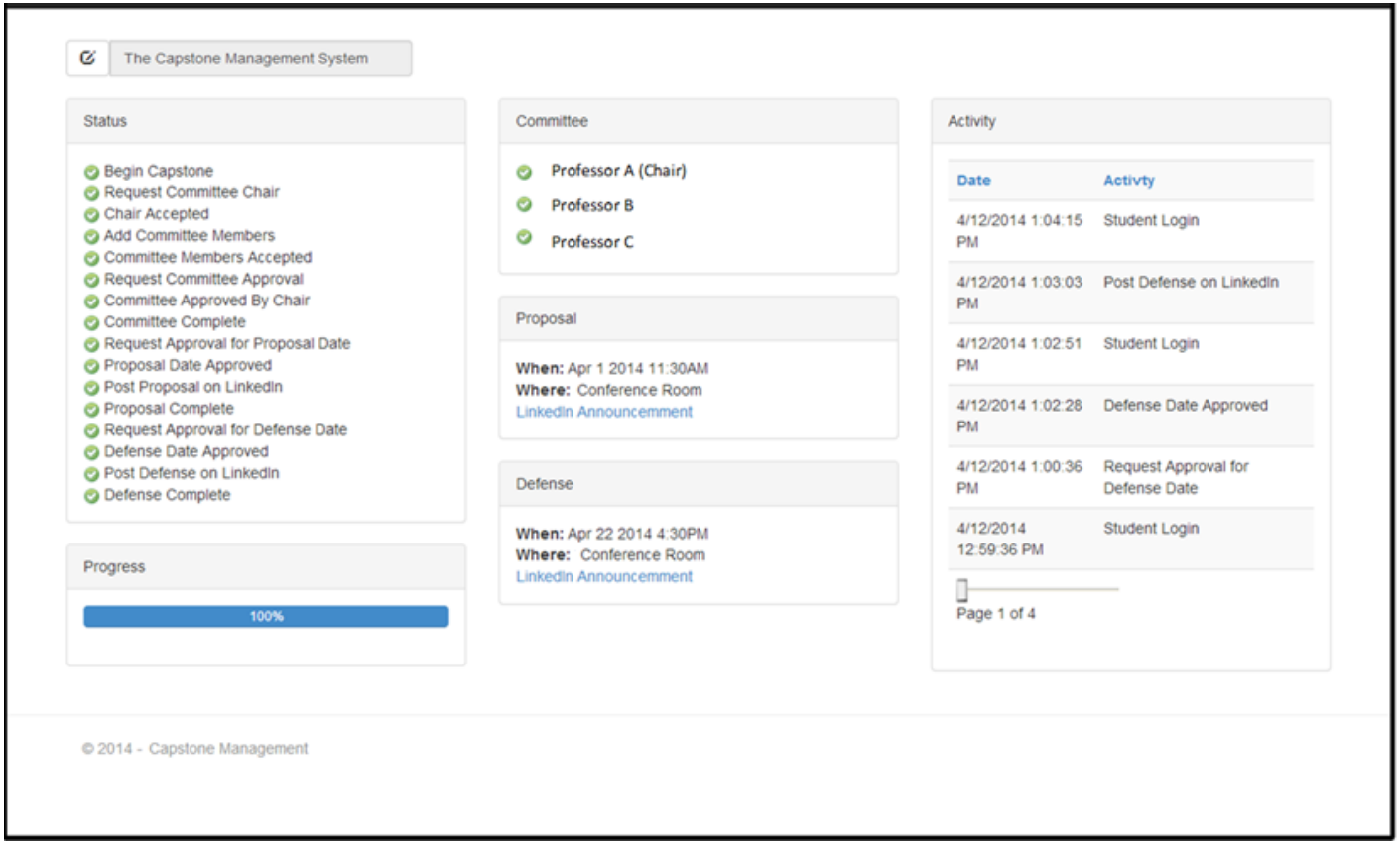
APPENDIX 6. CAPSTONE MANAGEMENT SYSTEM SCREENSHOTS







APPENDIX 7. STUDENT STATUS HOMEPAGE



APPENDIX 8. CAPSTONE MANAGEMENT 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