Capstone Management System (CapMS)

John Curley
Danny Glover
Daniel Rodenberg
Victor Tran

Table of Contents

1 Introduction

1.1 Executive Summary

With the rise in demand for higher education, the numbers of students enrolling in colleges and universities have been at a steady climb. Students not only come into higher educations systems to both learn about their chosen field of study and potential field of profession, but they also come to use the education system as a platform to showcase what they know and how they can apply this to their future workplace. This can be vastly different depending on the different coursework and degree program that a student pursues.

Despite the differences in work throughout the undergraduate level, the thing that is constant in every college and every degree program is a requirement for a capstone project of some sort to complete the process of obtaining a degree. This can manifest itself in many different ways. Some programs require a paper, some require experimentation and data collection, and others require a project. This capstone project is the opportunity for students to show future employers how all of their education has come together. Often times it is also a way for students to show how they can work in teams similar to a professional environment.

Our project has set out to make this showcasing easier. Students put their hard work and time into their capstone projects and the last thing they should have to worry about is showcasing that project themselves. CapMS looks to be an open source tool for students and teachers to showcase their work in a uniform way so that their hard work will not be wasted.

1.2 Acronyms and Terminology

Agile software development - Iterative software development method with frequent testing and team interaction to promote flexibility and communication.

Back-End System - Supports front-end systems by managing access to data, information, and assets.

Bootstrap UI- A user interface framework created by Twitter for the streamlining of web application development.

CapMS – Capstone Management System. Name of project/application.

CMS – Content Management System. Provides a maintenance interface for future users with limited coding experience.

CSS – Cascading Style Sheet. Language used to define how web pages written in HTML will be presented and displayed in web browsers.

DPX - Digital Picture exchange

Dyno - Heroku uses small containers called 'Dynos' to run user specified commands. See section 2.5 for further information.

EXR - Is a high dynamic range image format.

FERPA – Family Educational Rights and Privacy Act. Federal law that protects the privacy of student education records

Flat UI- The specific bootstrap theme being selected for CapMS.

Front-End System - System that users interact with directly.

Gem - An extension library for the Ruby language. See section 2.6 for more detailed description.

Gemfile - A format for describing gem dependencies for Ruby programs.

GIF - Graphics interchange format. This is a video clip that has a time limit and repeats when it hits the end of the clip.

Git - Free and open source version control system

Github - Web based hosting and interface for projects using Git.

Heroku - Cloud based hosting service. Used for hosting web applications. Supportive of multiple languages including ruby.

HTML – Hyper Text Markup Language. Tag-based markup language used to create web pages.

Hypertext Transfer Protocol (HTTP) - HTTP is the foundation of data communication for the World Wide Web.

JPEG - It is compression for digital pictures.

MYSQL - Is the most popular open source relational database.

PDF - Portable document format

Platform as a service (PaaS) - is a category of cloud computing services that provides a computing platform and a solution stack as a service.

PNG - Portable network graphics

Postscript - is a computer language for creating vector graphics

RoR – Ruby on Rails. Web design framework.

Ruby - A dynamic, reflective, object-oriented, general-purpose programming language.

Scrum - Agile software development framework that includes specific timeframes for meetings and project goals.

Secure Shell (SSH) - Cryptographic network protocol for secure data communication, remote command-line login, remote command execution, and other secure network services between two networked computers.

Secure Sockets Layer (SSL) - Cryptographic protocols which are designed to provide communication security over the Internet.

SVG - Scalable vector graphics

TDD - Test-Driven Development. Method of developing and improving software via frequent tests created to address specific issues or bugs.

TIFF - Tagged image file format

UI – User Interface. What the user of the application sees.

Web Server - Either the hardware (the computer) or the software (the computer application) that helps to deliver web content that can be accessed through the Internet.

1.3 Problem Statement

Currently the engineering department at the University of Missouri-Columbia and many other higher education systems do not have a system that allows students to showcase capstone projects after completion of the course. This leads to loss of work every semester. Various causes include database rollover, non-centralized storage of data, and human error by professors and students. Additionally, the lack of a well-defined system leads to questions of ownership and publication. Due to legal implications of FERPA, many capstone projects go completely unused despite their potential for future educational or practical purposes.

In conjunction with these issues, Capstone Management System (CapMS) will address several other issues within the capstone course process. As of now there is no standard system to form and manage capstone teams. This duty generally falls on the professor, who turns to inefficient methods and tools to assist students in forming teams. Furthermore, there is no centralized space for creating and sharing project ideas along with the inability of industry professionals to actively contribute.

CapMS will be an open source web application that facilitates the overall process of a capstone course. A message board system will allow students, staff, and industry professionals to share project ideas prior to forming teams. From there a team management system will allow students to connect to each other to move forward with project ideas. Finally the project management system provides students a way to publish their team's progress. The students will be given the option to publish their progress with varying degrees of privacy.

2 Problem Analysis and Research

2.1 User Interface

With the rise in the demand for web sites and web base applications, there has also been a huge rise in the options and tools for creating them. A major component of any application is the user interface. This is what the user sees of any given application. This allows for the user to interact with the business end of the application and accomplish the tasks that they set out to do. How straightforward and obvious the user interface is can determine the success or failure of a web application, and how widely applicable it can become. This is crucial for consideration when designing and developing a user interface. Another recent aspect and concern on user interface development has also been the widening of devices and device sizes being used to access these applications. Devices that have access to these web applications can range from the size of a smart phone all the way to a full fledged desktop. This can create a lot of aesthetic discrepancies and what the user sees can be distorted causing confusion to the user and leading them to avoid using the application.

2.1.1 Straightforward UI design

User interfaces can present itself in many different ways. But underneath the bells and whistles is a design standard that is used in order for users to be able to easily understand and use.

The first aspect of straightforward design is a uniform way to traverse and move throughout the website. This is crucial for creating an overall theme and feel for the web application. The most widely used and accepted way of doing this is a navigation bar. This navigation bar is constant and the same throughout the page. The navigation bar consist of links to all the major parts of the web application itself. This usually consists of links to the home page and a profile page if one is necessary. This allows for users to get to the standard pages if they are lost within the application. Also a common and almost standard practice is to have the home link on the far left hand side of the navigation bar. This common practice will also help prevent confusion due to its use.

The next way to incorporate a straightforward UI is to compartmentalize different features on any given page. This is the incorporation of things such as dividers or a contrast of color so that users can see and be able to mentally group related items together. This allows for users to focus on information and options relevant to them at the time. This will eliminate confusion about what links and information are relevant to what and eliminate mistakes that stem out of confusion.

Another way of creating a straightforward and easy to use UI is to make obvious what each element on the page does. This can happen in a number of ways. The first would be a color scheme. This means changing the color of links on the page or using icons that are descriptive of what the link does or leads the user to. Using icons has become more of a standard due to the fact that it can also declutter an interface and make everything more aesthetically pleasing to the user.

2.1.2 Mobile first UI

Mobile first development is a recent development that has occurred in the realm of website design, but this also seems to be the wave of the future for website design. This is the concept that websites should be designed with mobile platforms in mind. This means highly scalable websites that take into the consideration that the application will be accessed from all sort of devices with varying capabilities and dimensions. This means designers must focus on the core content of any given application. This is due to the loss of 80 percent of screen space when transitioning from a standard computer to a phone.

The way to implement mobile first web design is to start small and build up. By that it means that a web application should start with building functionality for mobile devices first. It is easier to scale up to full fledged computers than down to a mobile device. This is beneficial due to the simple fact that if something works on a mobile device, it will work on a desktop, but not vice versa. This allows the application to reach it's maximum audience without restrictions as opposed to working the other way. By adopting this mobile first methodology, it also will force a straightforward UI. This forces developers to cut out all of the bloat and things that are not necessities and focus on user experience and simplicity.

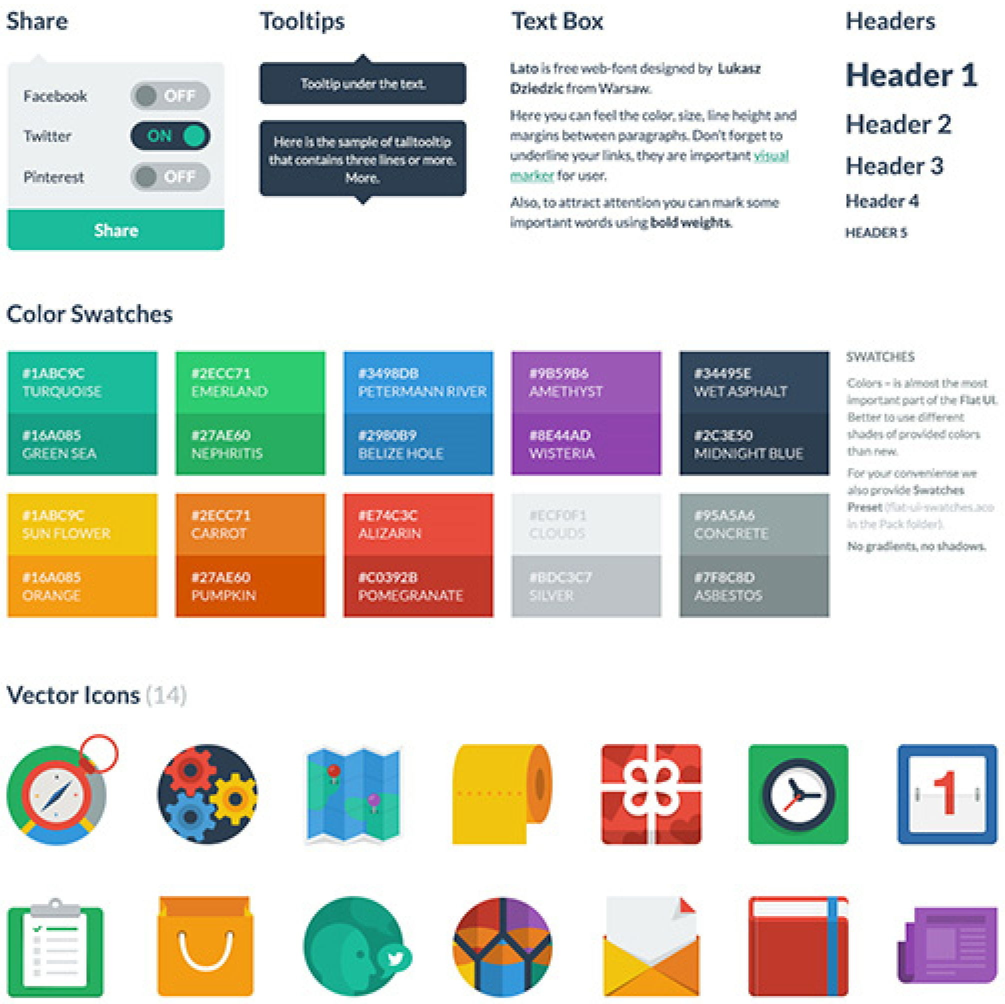
2.1.3 Bootstrap UIs

With the gaining popularity in website design and website development. There have been creations of frameworks to make the process easier. Rather than building from the ground up. These frameworks give developers a shell to work from and takes the complexities of hand building and designing out of the equation. With that being said, one of the most popular design frameworks being used is bootstrap. This is a design framework built by Twitter. They allow users to download shell sites fully loaded with CSS and a basic HTML files. This allows for uniformity and a similar look and feel throughout the website. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional extensions.

One of the major upsides to bootstrap is also the functional capabilities of responsive design. This means that it incorporates the standards for dynamic adjustments relative to device and device sizes. By using bootstrap, we are able to implement all of the UI standards that we are striving to achieve. The UI standard and uniformity are easily accomplished with the tools and layout offered by bootstrap. This will allow for more focus on functionality and performance and a little less on the aesthetic of the site. This still requires appropriate utilization of the offered tools. But the burden of developing the tools are now a non-issue.

2.1.4 Flat UI

Flat UI is a bootstrap based framework that allows users to use a shell design for the look and feel of their website. This allows for developers to focus on the core functionality of the project as opposed to the design. This will allow developers to cut the time time of design astronomically. With the flat UI package, developers can also used the prebuilt icon library. This is also an element of graphic design that can be eliminated from the process. Below in Fig. 1 is a collage of FlatUI icons and features for a feel of CapMS's appearance.



Tooltip under the text.

Here is the sample of talltooltip that contains three lines or more. More.

Lato is free web-font designed by Lukasz Dziedzic from Warsaw.

Here you can feel the color, size, line height and margins between paragraphs. Don't forget to underline your links, they are important visual marker for user.

Also, to attract attention you can mark some important words using bold weights.

Colors – is almost the most important part of the Flat UI. Better to use different shades of provided colors than new.

For your conveniense we also provide Swatches Preset (flat-ui-swatches.aco in the Pack folder).

No gradients, no shadows.

Figure 1. Sample image of FlatUI icons and general appearance.

2.2 Family Educational Rights and Privacy Act

On August 21, 1974, President Gerald Ford signed the Family Educational Rights and Privacy Act in order to protect student's rights at all levels of education. These rights belong to the child's parents or guardians until the age of 18 when they transfer to the student [1]. The law applies to all schools and universities funded, partially or fully, by a program under the U.S. Department of Education with the general purpose of protecting both student privacy and intellectual property. Since the intended audience for CapMS will be universities and the majority of universities in the U.S. are funded at least partially by the U.S. Department of Education, FERPA regulations must be obeyed in it construction and execution.

By default, CapMS will set all submitted documents and files to private, meaning that those documents and files will follow FERPA under the assumption that the student has not given permissions to any other person. Obviously private documents must be kept secure and hidden from everyone without permission to view them. This responsibility will fall under the responsibilities of the university or admin utilizing CapMS, as the application itself does not have built in storage for documents and files. The university is obviously prohibited from sharing these records without written consent; however, there are special circumstances in which student records may be released without written consent of the student to which they pertain. The U.S. Department of Education's websites specifies those circumstances as follows:

· School officials with legitimate educational interest;
· Other schools to which a student is transferring;
· Specified officials for audit or evaluation purposes;
· Appropriate parties in connection with financial aid to a student;
· Organizations conducting certain studies for or on behalf of the school;
· Accrediting organizations;
· To comply with a judicial order or lawfully issued subpoena;
· Appropriate officials in cases of health and safety emergencies; and
· State and local authorities, within a juvenile justice system, pursuant to specific State law [2].

An integral feature of CapMS allows students to choose to make their efforts public so that other people, including potential employers, may see all the work they have put into their projects. In order to legally share any student's work, however, permission must be granted from every student involved. Section 99.30 of FERPA, as seen below, outlines exactly what a written consent form must entail in order to be considered fair and valid [3].

§99.30 Under what conditions is prior consent required to disclose information?

(a) The parent or eligible student shall provide a signed and dated written consent before an educational agency or institution discloses personally identifiable information from the student's education records, except as provided in §99.31.
(b) The written consent must:
(1) Specify the records that may be disclosed;
(2) State the purpose of the disclosure; and
(3) Identify the party or class of parties to whom the disclosure may be made.
(c) When a disclosure is made under paragraph (a) of this section:
(1) If a parent or eligible student so requests, the educational agency or institution shall provide him or her with a copy of the records disclosed; and
(2) If the parent of a student who is not an eligible student so requests, the agency or institution shall provide the student with a copy of the records disclosed.
(d) "Signed and dated written consent" under this part may include a record and signature in electronic form that—
(1) Identifies and authenticates a particular person as the source of the electronic consent;
and
(2) Indicates such person's approval of the information contained in the electronic consent.

99.30(a) refers to section 99.31 of FERPA which outlines situations in which written permission is not required to share academic records, none of which apply to the intended use of CapMS. Section 99.30(b) outlines the three general rules that a written consent form should be developed around. In the case of CapMS, a written consent needs to detail exactly which documents, files, images, videos, and any other personal information will be disclosed. The purpose of disclosure also must be stated, which will differ based on institution, but should be mostly limited to showcasing students' efforts and educational purposes. Additionally, the exact audience that will have access to those items must be disclosed in full. CapMS by default will set all items to private so that only members of the team have access, but when items are published it will be made clear that anyone in the general public will have viewing access. Section 99.30(d) also pertains to CapMS as the agreement form will be digital. Therefore a student must be logged into an authentication system that confirms their identity and it must be explicitly stated that agreeing electronically is the equivalent of a physical signature and date. Furthermore, students need to be informed that sharing of their work is not required and in no way will affect their grade or standing. The consent form must make it obvious that sharing is strictly optional and voluntary.

An additional feature of CapMS allows students to include links to various social media applications such as Facebook or LinkedIn. However, it needs to be made clear that these links are not required and are completely optional, as it is a violation of FERPA to require students to post publicly without prior consent to make their work public [4].

2.3 Content Management System Research

One of the problems with creating a website is the ability for someone who does not have any experiencing in programming to maintain the site. Employees today will have software engineers create a website for them and when it is finished they will pay the developers for their work and then have no way to update the website on their own. Developers use HTML and CSS to develop the front end design of a website. HTML and CSS are not difficult to learn, but it does take time to learn the syntax and how to implement the tags. Most companies that have smaller websites will have a tech savvy employee learn the basics of web design and have that employee create a simple website. What happens when that employee leaves? Who will take the time to update and maintain the site? Developers have created a new way for employees with little programming knowledge to be able to maintain their site and that is through a Content Management System (CMS). “A CMS is a computer program that allows publishing, editing and modifying content as well as maintenance from a central interface. Such systems of content management provide procedures to manage workflow in a collaborative environment. These procedures can be manual steps or an automated cascade. CMS's have been available since the late 1990s [5].”

A CMS is a way for someone with little to no programming experience to be able to maintain and update a website. There are hundreds of CMS's out there and they all work with different frameworks. For almost any language you are able to find a CMS that will work with it. One of the most popular CMS's is WordPress, it is an open source blogging tool that is based on PHP and MYSQL. All CMS's have their own interface and they all have their pros and cons. Plugins are a very popular functionality of a CMS. “In computing, a plug-in (or plugin, extension, or add-on / addon) is a software component that adds a specific feature to an existing software application [5].” This allows the user to add something that the CMS does not already have that can make something even more powerful than it already is.

The Capstone Management System (CapMS) will be open source and developed in Ruby on Rails (RoR). RoR is a very powerful programming language that is growing rapidly. It is an open source web framework. The CMS that will be used for the CapMS is RefineryCMS (Refinery). Refinery is a CMS that supports RoR. It was founded in 2004 and the lead developer is Philip Arndt. Refinery has a few key principles. The first one being, “"*The Rails Way"*- where possible Refinery embraces conventions used in Rails, allowing any Rails programmer to leverage existing knowledge to get up and running quickly [6].” It has an “End user focused interface - Refinery's user interface is simple, bright and attractive so end users feel invited and not overwhelmed [6].” It also has “Awesome developer tools - Refinery makes it easy for developers to add functionality and change the front end look and feel [6].” Their last key principle is to “Encourage and Help Others - Refinery has an active community on Google Groups and IRC. If you ever have a question there is usually someone able and willing to assist you [6].”

RefineryCMS was selected as the CMS because of it's wide use across the internet. According to a number of resources Refinery is one of the top and most used CMS for developing with RoR. Another reason is that it is completely free.

One of the problems with RefineryCMS is the installation process of ImageMagick, which is a gem. There are a number of things that a user must have installed and working on their computer before they are able to use Refinery. You must first have a working installation of Ruby. This is something that is very straight forward, if you want to use refinery it uses Ruby so the user must have a working version of Ruby. Next the user must have ImageMagick installed. This has caused problems when attempting to install ImageMagick. “ImageMagick® is a software suite to create, edit, compose, or convert bitmap images. It can read and write images in a variety of formats (over 100) including DPX, EXR, GIF, JPEG, JPEG-2000, PDF, PNG, Postscript, SVG, and TIFF. Use ImageMagick to resize, flip, mirror, rotate, distort, shear and transform images, adjust image colors, apply various special effects, or draw text, lines, polygons, ellipses and Bézier curves [7].” In order to be able to deal with images in Refinery ImageMagick is what needs to be installed. The only problem is is that it is a known error for people that use Refinery. One of those ways to fix this bug is making sure that all of your versions are updated and that they all work with each other. A lot of times Ruby might be updated to much and ImageMagick won't work with it.

2.4 Deploying a Ruby on Rails App to Heroku

2.4.1 Background

Our web application will need to be hosted on a web server for testing and initial deployment. Instead of using the University of Missouri's on site server system Babbage, we will be using a cloud based platform as a service (**PaaS**) called Heroku. Heroku will allow our team to deploy, run, and manage our web application built in Ruby. Heroku was founded in 2007 by Orion Henry, James Lindenbaum, and Adam Wiggins. The service was originally built to host Ruby applications but now is a multi language platform. Heroku is designed with the intention of making the deployment of applications easier, with the ultimate goal of leaving more time for development teams to work on their applications instead of hosting. The following pages will describe the steps required to deploy a Ruby on Rails application using the Heroku service [8].

Before getting started with Heroku the following things need to be installed on the local machine: [9]

- Locally installed version of Ruby, Rubygems, Bundler, and Rails 4+
- Git

2.4.2 Step 1: Sign Up

A login must be created through https://signup.heroku.com/signup/dc Accounts are free to create [10] [11].

2.4.3 Step 2: Install Heroku Toolbelt

The Heroku tool belt includes the 'Heroku Client', a command line interface for creating new apps or deploying apps that you have created. This command line tool acts as an interface to Heroku's API and also allows the user to modify apps names, configure add ons, and take back ups.

Installing the Heroku tool belt is as easy as downloading the .pkg file from the Quickstart page and running the installer.

The tool belt includes version control software Git as well as Foreman, software to test apps locally.

Once the Heroku Client is installed command line instructions can be used by using the preceding keyword heroku i.e. 'heroku login.'

2.4.4 Step 3: Login

Your machine must be authenticated with your account by logging in via the command line with the credentials created in step 1. To do so type 'heroku login' on the command line and enter login information. If this is the first deployment a public ssh key will need to be generated, click yes when prompted to create a new key. The Heroku app will tell you where it has placed your new key [10].

New public ssh keys can also be generated after login with the command line instruction 'heroku keys:add'

A detailed explanation of how Heroku uses SSH keys can be found here: https://devcenter.heroku.com/articles/keys

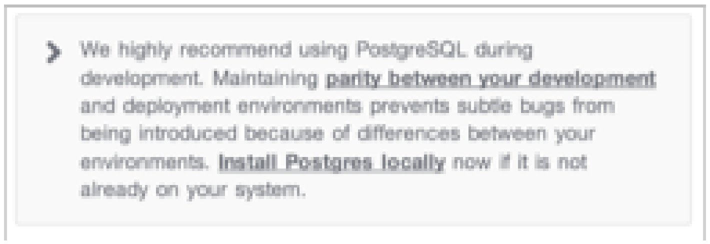
This concludes the basic setup needed before deployment [10].

2.4.5 Step 4: Deployment

To run on Heroku the app must satisfy the following requirements:

1. Configured to use PostgreSQL database
2. Contain a Gemfile that lists all dependencies
3. Contain rails_12factor in production group of Gemfile
4. App must be running on Rails 4.x [8].

Note on PostgreSQL: The following instruction is given on Heroku's website concerning the use of Postgres database [8].



We highly recommend using PostgreSQL during development. Maintaining parity between your development and deployment environments prevents subtle bugs from being introduced because of differences between your environments. Install Postgres locally now if it is not already on your system.

Once these requirements are satisfied navigate to the root directory of the Rails app. The entire directory needs to be added to Git and can be done by running the following commands: [8]

1. git init

2. git add .
3. git commit -m “init”

Now to deploy the app run the following commands again from the app’s root directory: [8]

1. heroku create
2. git config -e (this is to check to make sure git remote added correctly)
3. git push heroku master

The app is now deployed, last step is to migrate the app’s database manually be running the command: [8]

- heroku run rake db:migrate

The app is now deployed to Heroku’s servers. It can be viewed by first making sure the web process is running and then using the heroku client with the following commands:

- heroku ps:scale web=1 (makes sure one dyno is running web process)
- heroku open (opens application in web browser)

Heroku will assign a random url during the development/testing phase and once the app is fully polished a custom domain can be created [8].

2.5 Dynos and the Dyno Manager

2.5.1 Background

Heroku uses small containers called ‘Dynos’ to run user specified commands. On their website Heroku describes Dynos in the following manner:

> “You can think of it as a virtualized Unix container—it can run any command available in its default environment.The commands that run within dynos include web processes, worker processes (such as timed jobs and queuing systems), and any process types declared in the app’s Procfile. Admin and maintenance commands can also be run in one-off dynos [12].”

2.5.2 Web Dynos

Web Dynos host your application and responds to HTTP requests. Having a single web Dyno running will result in the Dyno being put to sleep after one hour of inactivity. This means that the first request after that has happened will take slightly longer since the Dyno needs to wake up. Heroku’s free tier includes 750 free dyno-hours per application per month. This means that running multiple web Dynos to avoid this sleeping after inactivity behavior would surpass the monthly allowance of the free tier [12].

2.5.3 Routing

‘Routers’ receive all HTTP and SSL requests and pass them to web dynos to handle the requests. To keep things running smoothly and efficiently these routers use a randomizing algorithm to spread traffic across web dynos [13].

2.5.4 Worker Dynos

These execute background work. This includes processing queues, running code, or any computation heavy work that you don’t want the web Dyno to get bogged down with [12].

2.5.5 Scalability

Dynos are a one feature that makes Heroku great for quickly scaling an application. Developers can add or remove dynos allocated to their app at any time. This means if your app needs an increase in power or sees a large increase in traffic all you have to do is log in and add more Dynos. This is much easier than buying more ram for a server you are hosting independently to run the app. Costs for scaling up can be calculated at https://www.heroku.com/pricing. Dyno's are prorated by the second they are used so you can safely try out a new set up and revert it back without being billed for an entire month's use [14].

2.5.6 Reliability

Dynos are distributed across an 'elastic execution environment'. This is a fancy way for saying the hardware is set up like a cloud and your app may using hardware that is located in separate locations. This increases reliability because a hardware failure in one location won't result in every part going offline. Furthermore, each Dyno's status is constantly monitored and any failure results in a new Dyno being allocated to replace the failed one immediately. This aspect of leaving the physical hardware to Heroku relieves the team of things like creating contingency plans for what to do in the event of a server crashing in the middle of the night [12].

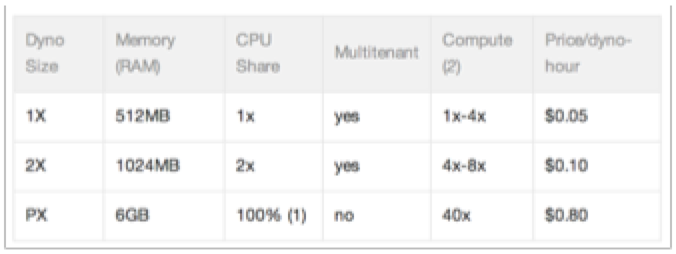
2.5.7 Logs

Heroku keeps detailed logs to let you know when and where your app may need more power. This helps you make informed and timely decisions about increasing your dynos. According to Heroku's article on logging this includes all of your app's running processes, system components, and backing services. The following types of logs are available to track errors and judge if Dynos need to be increased (from heroku's article on logging) [15]:

> *App logs* - Output from your application. This will include logs generated from within your application, application server and libraries[15]. (Filter: --source app)
> *System logs* - Messages about actions taken by the Heroku platform infrastructure on behalf of your app, such as: restarting a crashed process, sleeping or waking a web dyno, or serving an error page due to a problem in your app [15]. (Filter: --source heroku)
> *API logs* - Messages about administrative actions taken by you and other developers working on your app, such as: deploying new code, scaling the process formation, or toggling maintenance mode. (Filter: --source heroku --ps api) [15].

2.5.8 Sizes

Dynos come in 3 different sizes and the requirements of the application should be thoroughly analyzed before deciding which size is appropriate. For our apps purposes increasing to a larger Dyno will not likely be necessary. Below is a graph displaying Dyno size options (Fig. 2) [16].



Dyno Size	Memory (RAM)	CPU Share	Multitenant	Compute (2)	Price/dyno-hour
1X	512MB	1x	yes	1x-4x	$0.05
2X	1024MB	2x	yes	4x-8x	$0.10
PX	6GB	100% (1)	no	40x	$0.80

Figure 2. Dyno Size options.

2.6 Ruby Gems

2.6.1 Background

Our application is built in the Ruby language which provides us a host of useful add-ons and libraries called ‘Gems’. Gems are open source pieces of software and free to use. The functions gems can execute range from authentication and security to entire applications such as message boards and content management. Our development team will be taking advantage of Gems to avoid recreating software that is already well polished and freely available through the open source community.

According to rubygems.org there are **75,093** gems available for free via the RubyGems library [17].

2.6.2 RubyGems

RubyGems is a library that allows the developer to install Gems directly from the command line. Typing ‘gem’ will give a basic help message with various commands and how to use them. The most common are search and install [18].

2.6.2.1 Search

The ‘gem search’ command can be used to search for available gems on RubyGems. This is used by typing ‘gem search’ followed by the name of gem you are looking for. Available gems will be displayed with their version numbers [18].

2.6.2.2 Install

Downloading and installing a gem is as easy as typing ‘gem install’ followed by the desired gem’s name. RubyGems will display the installation progress and give you a status of how many gems were installed once the process is complete [18].

2.6.3 Gemfile

To easily maintain all the Gems the application is dependent on we will use what is called a Gemfile. A gem file is a file in the root directory of the application that contains at least one gem source (such as **www.rubygems.org**) as well as a list of Gem names and version numbers. Gems can be placed into groups to maintain organization or if the group is only used in certain parts of the application. If no group is specified the Gem will be placed in the 'default' group [19].

2.6.4 Require

Gems can also be included directly into code instead of using a Gemfile. This is done by typing 'require gemName' at the top of the code and replacing gemName with the appropriate gem's name.

3 Requirements

CapMS will allow for the use of single sign-on servers for user authentication or the creation of new user accounts. Universities will be able to use single sign-on servers in conjunction with created user accounts to facilitate students, faculty, and outside mentors.

3.1 User Interface Requirements

3.1.1 Mobile first UI standard

This web app will adapt the standards of developing mobile first, giving it compatability throughout all devices regardless of size and resolution.

3.1.2 Straightforward and easily understood accessibility for all users

The user interface will allow for users to be able to easily sign in and access their accounts. This will be a straightforward login and logout process that would be common knowledge to the average user. This web app is designed to be straightforward and obvious but having a primary navigation bar that will allow users to access the main components of the CapMS. When not logged in, the navigation bar will include a log in button, teams view button, and an about button for information about the app and its creators.The things accessible from the navigation bar when logged in are the home page, the message board, the teams view page, a notifications modal button, the profile page, edit profile page, email professor, the current user’s team page, and logout button. The home page will be different depending on whether or not a user is signed in. If the user is not signed in, this page will be a simple login page, if the user is logged in the page will contain the class’ name, and image for the class, and links to each team’s page. Users will be able to click on said links to view team members, descriptions, and whatever the given team has decided to publish.

3.1.3 Team formations

Upon signing in the user will be able to access the rest of the web app along with the signed in version of the home page. The view teams page will also change to allow users to see classmates that are not currently on a team. This will allow for easy team formations. Also every profile icon will be a link that will take you to that user’s profile page which will include all of their interests and biography information. If the current logged in user is on a team, they will be able to click a button below an available user’s profile picture to invite them to their team. Also, if a user is not currently on a team, there will be a “Create Team” option that will allow the user to create and automatically join a team.

3.1.4 Easy way of finding and advertising your interests

From the navigation bar the user can click on the user-named link on the right and then follow the ‘edit profile’ link. This will allow users to edit and add to their profile. Anything from the user’s biography to the user’s resume and the team that the user belongs to is viewable and editable from this page. This will dictate what other users will see when they click on your name throughout the web app. This will help other students and users to know a little more about any given user.

3.1.5 Project privacy and publication

A user can click the user-named link on the right side of the navigation bar and then follow the link to their team’s profile. This profile includes the team’s members, the team’s description, and files that have been uploaded whether they have been published or not. This also lets users upload files that are relevant to their project. This can include papers, videos, or any other sorts of media. These items will only be viewable by members of the team until the team engages on and completes the publication process.

3.1.6 Open communication and collaboration

By clicking on the ‘Message Board’ link on the navigation bar, a valid user can post ideas and suggestions for project ideas. These entities will be able to comment on these post to try and flush out ideas and gauge interests from students and peers. This will help the formations of teams and allows for students to be involved in projects that challenge them and engage their interests.

3.1.7 Open Source

Developing a project with the capability of being open source allows for users to have free license and the actual code so that they are able to improve or change what they want. Essentially it’s a blueprint for the project. This project will have the ability for any user to customize their pages however they see fit. Open source makes our code open to the general public and that is essential to creating a site that multiple colleges and universities are able to use.

3.2 FERPA Compliance

As the focus of CapMS is to allow students the opportunity to publish their capstone work for public view, it is vital that all FERPA regulations are strictly followed and enforced. Every time a new document or file is submitted, each student must be asked for permission to publish and make the file public. If each student does not agree, then that file must be kept secure and private as FERPA requires. Permission to publish is granted when each student in the team is presented a written agreement and each student digitally signs and dates that written agreement. The requirements of this agreement can be found in section 2.2. At any time, however, a student should be allowed, through the CapMS website, to change the status of their files from private to public. FERPA also requires that files be kept in a secure location to ensure privacy for sensitive student information. While CapMS will establish a file storing management system, the security of the server will be up to the institution utilizing CapMS. Since institutions that will be using CapMS will almost certainly already be responsible for keeping FERPA regulations with student information, having a secure location to store files should not be an issue.

3.3 Constraints

There are a few constraints that must be considered when developing this project. The first and the most important is to comply with FERPA standards. This extends from logging in, profile creation, and all the way to the publication of work. The next constraint would be the learning and development of proficiency in new technologies such as Ruby on Rails and integration of a CMS. Finally the biggest constraint will be time. It is very easy to take on too much with a project of this nature and due to the limited time, the possibility of not completing the project in it's entirety is a real possibility.

Due to the nature and growing community of web application developers, the constraints of this project are fairly minimal.

4 Design

4.1 Interface Design

4.1.1 Navigation

A navigation bar is constant and the same throughout the page. The navigation bar consist of links to all the major parts of the web application itself. For CapMS, we have chosen to include a message board link, an all teams view page, a notification pop-up modal, and a user-named drop down. This drop down includes links to the current user's profile page, edit profile page, team page, a link to email the professor, and a logout button. If no user is logged in, the user-named drop down will instead be a login button and only a teams view page and an about pop-up modal button with some information about the website and its creators.

4.1.2 Home Page

The home page for a user who has not yet logged in will appear only as a log-in page. From there, however, a non-logged in user can still use the navigation bar to access the all teams view page as well as the about pop-up modal as discussed earlier. For a logged in user, the home page will display the name of their class and links to each of the teams currently formed for the class.

4.1.3 Sign In Page

This will be a simple sign in page that takes in a username and password. This page will only appear to users that have not yet logged in and will be accessible through the home button or the log in button at the far left side or far right side of the navigation bar, respectively.

4.1.4 Single Team Page

The single team page can be accessed at several points in the website by clicking on the team's name or on the team's picture. This page will have the team's name displayed along with each user's name and picture on the team. It will also have a team description and any files made public by the members of the team. If the current user is on the team or is an admin, they will see uploaded files regardless of their privacy setting. When the current user is on the team, there will exist an 'Upload' button that will allow anyone on the team to upload a file to their team's page. This file will default to private. Additionally, if the current user is on the team, there will be options next to each file depending on that file's status. If the file has not yet been requested to make public by anyone, the options will include 'Publish' and 'Edit' to start the publication process or to edit the file itself, respectively. If someone else on your team has started the publication process, then there will be the options to agree to publish, halt pending publication, and edit. If you have already agreed to publish, but not everyone on your team has agreed yet, then the options will be to halt the pending publication or edit the file. Finally, if the file is public then the option to make it private will appear as a button for the file. When viewing their own team's page, users will also have the option to leave the team and edit the team page in the form of buttons below the team's name at the top of the page. An admin will not be able to initiate or approve publication, but at any time they will have access to the 'Make Private' option for public files of any team. This page is accessible by any user, logged in or not. Only the features of the page will vary depending on permission levels.

4.1.5 Profile Page

This profile page is what appears when a user clicks on a profile picture, or navigates to their own profile via the navigation bar's user-named drop down for logged in users. This page consists of a name at the top of the page, a profile picture, the user's 'about me', and the user's resume. Also, if the current logged in user is on a team and is viewing an available user's profile page, there will be an invite button to invite them to the logged in user's current team. The 'about me' section will include an email, biography, and optional links to linkedin and github accounts.

4.1.6 Message Board

The message board page will have all the current messages listed with the student that posted it, a subject, a summary of the message, and the option to click to comment. Upon clicking the comment option, a user will be shown the full title and text of the original message and any comments that have been made on the message, with the user that posted the comment. Additionally, there will be a new message option on the main message board page that can be clicked to create a new thread entirely. This page is only accessible to those that are logged in.

4.1.7 All Teams Page

The all teams page will consist of every team divided into sections that will include the team's name and all the profile pictures and names of that team's members. The user, logged in or not, will be able to click the team's name to view the respective team page or any of the user's profile pictures to view their profile page. Additionally, when not logged in, the user sees each team subdivision, and then a list of all the students in the class at the bottom, whether or not they are in a team. However, when logged in, the user will instead see a list of available users at the top with options to invite them to a team, if the current user is on one, or the option to create a team, if the current user is not yet on a team. After clicking the 'Create Team' button, a pop-up modal will appear asking the user to enter the team name and a team description. After creating the team, the user will automatically be put into that team.

4.1.8 Notifications

The navigation bar has a button named 'Navigation' that brings up a modal of two different types of notifications (named 'bulletins' in the database). The modal will display any invites you currently have to a team, including the team's name that you have been invited to and an option to accept or decline the invitation. Additionally, if someone on your team has requested to publish a team document, there will be a section in the notification modal displaying that a request to publish has been made, how many files have been requested for publication, and a link to your team page to view those pending publications. If a user does not have any notifications, this modal will display that message to the user. This button is only available on the navigation bar when logged in. To show the user that they have notifications, a red box with a white number inside will be displayed next to the notifications button on the navbar, the number representing how many notifications the user has. A similar red box will appear next to the message board button if there are any new messages or comments, the white number representing how many messages or comments have been added.

4.1.9 Admin Tools

Additional features will be present on the user profile page whenever a admin-level user is logged in. One such feature is the ability to view a list of students currently recognized as in the class and the ability to add a user to that list. This list is essentially the 'approved_users' table in the database which ensures that a user attempting to log in is not only a registered university student, but also actually in the class. Furthermore, the admin has the ability to view and add users to the industry professionals list that keeps track of valid email addresses and passwords for users that are not students. Furthermore, scrolling down reveals a list of available users and the users on each team. Each team will have a ‘Disband Team’ button that the admin can use to delete the team, adding each of its users to the available users category. The admin can also click the drop down below each user and select to view their profile, delete the user completely, or remove the user from the team if the user is currently on a team.

4.2 Backend Design

4.2.1 Web Stack

Our application will be used soley on the internet, and thus its infrastructure will replicate a fairly traditional webstack. A web framework will connect the application's core functionality and data to the frontend. This framework will live on a web server that is constantly connected to the internet.

4.2.2 Web Server

This consists of a hosting server to host the application's code as well as handle HTTP requests and provide responses. For development and testing the team will be using Heroku - a cloud based PaaS hosting service. This will allow us to focus on developing code and not be concerned with maintaining a physical server. Furthermore, Heroku's cloud based infrastructure will allow for testing different arrangements and amounts of server hardware with little or no fee.

4.2.3 Application Framework

The applications code will be written in ruby and thus employ the web framework Ruby on Rails (RoR). RoR will further simplify our development by providing a sound architectural framework as well as common web necessities such as url routing and authentication. Using RoR will help us avoid reinventing systems that are already created and well established by the open source community. RoR's avid open source community will also help our project grow once we release it to the public.

4.2.4 Database

The data held and used by our application will be in an SQLite3 database. This database will be created locally in the team's development environment and then migrated to Heroku for testing and deployment. The ERD for this database can be seen below in Fig. 3.

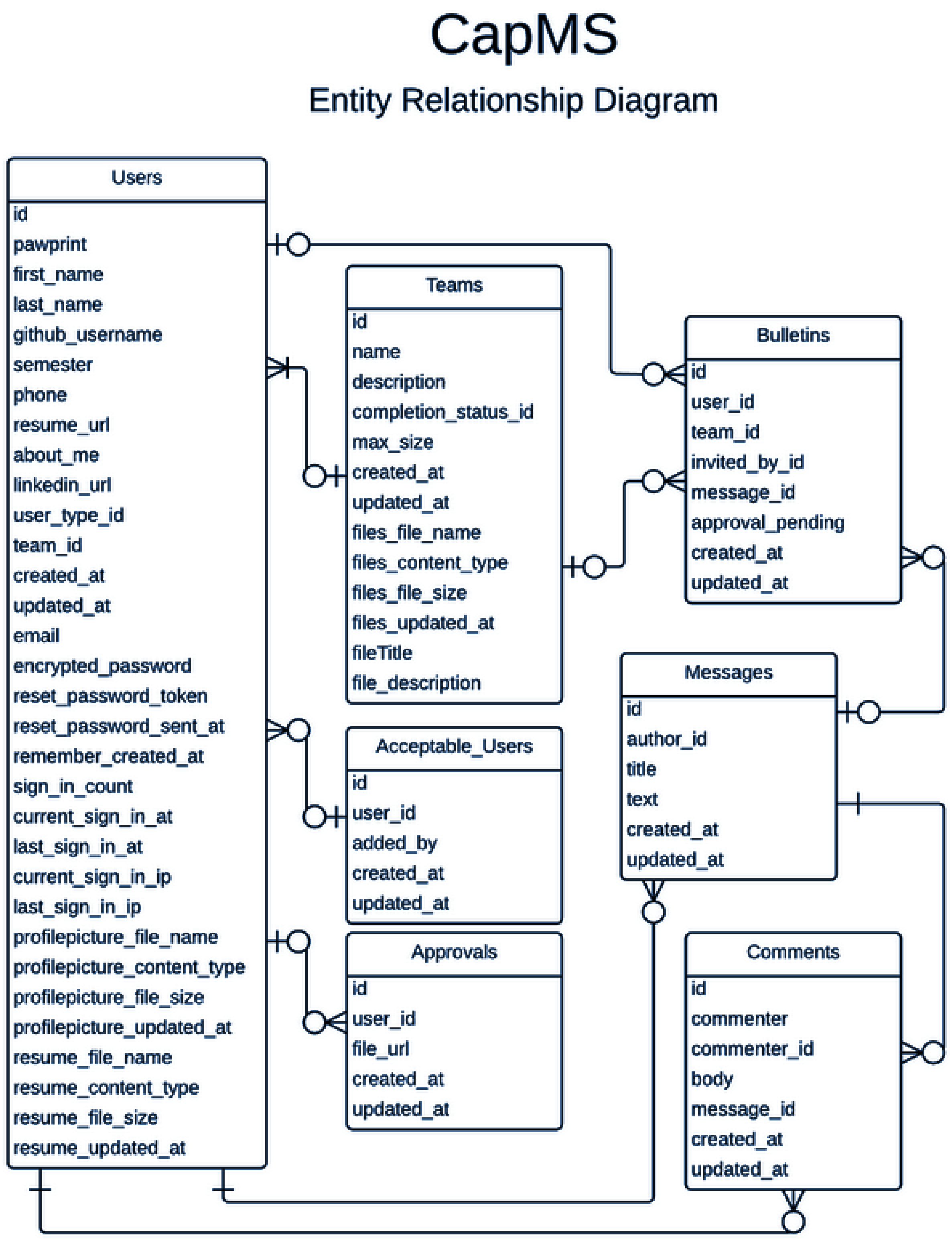


Figure 3. CapMS ERD.

4.2.4.1 Users Table

The users table stores a variety of self-explanatory user information such as pawprint, first name, last name, URLs to external websites, resume file information, and some contact information. The table also includes team_id that serves as the foreign key to the team table, linking users to a team. User_type_id is also of note as it defines which level of permission a user has: student, industry, or admin.

4.2.4.2 Teams Table

The teams table has one to many users and contains the basic information about teams such as their names, descriptions, and a maximum size. The teams table also contains file information for files so that they can be easily linked to AWS. In Fig. 3 these columns, all starting with 'files,' are condensed to help fit into a reasonably sized figure. In our actual database, each of these columns had to be duplicated for each possible file a team can upload (up to 10). This is due to an issue with paperclip and AWS interactions, which will be further discussed later.

4.2.4.3 Acceptable Users Table

This table is simply to keep track of each user added by an admin as part of the class. Because CapMS utilizes single sign on, we required a table to keep track of which user pawprints are actually a part of the class as the single sign on server simply returns true or false on a valid or invalid pawprint/password combination for any student. Without referencing this table upon an attempted sign-in, any student of the university with a valid pawprint and password would be able to log into our system.

4.2.4.4 Approvals Table

The approvals table uses a user id as a foreign key to connect a single approval to both a user and a file. Approvals are used to determine what state of publication a file is in. If no approvals exist for a file from users on the team the file belongs to, then the option to start the publication process will be presented. If an approval exists for every member in a team for a single file on that team, then the file has been made public. If some approvals exist but not for all the members of the team, then the publication process has started, but not everyone has agreed to make the file public. When one member of the team declines the request to publish something, then all the approvals are deleted for users on that team and for the file that a user has decline to make public.

4.2.4.5 Bulletins Table

The bulletins table contains the information for notifications in the CapMS system. These bulletins can be one of three types: an invite to a team, an alert that there is a new message or comment on a message, or a request to approve the publication of a file. If the bulletin is an invite to a team, than the team_id field will contain a valid team_id. If the bulletin is to notify the user of a new message or comment, the message_id field will be a valid message_id. And if the bulletin is a request for publication, the approval_pending field will be set to true.

4.2.4.6 Messages and Comments Tables

These tables' purposes are fairly obvious. The messages table keeps track of titles, body text, and who created the message. The comments table does the same with a comment's title, body text, and who created the comment. A message can have zero to many comments.

4.2.5 File Hosting

Files uploaded to the application will handled via an external gem as well as Amazon's SW3 Service. Specifically, the gem Paperclip will be used to efficiently and effectively process user uploads and store them on a SW3 server using the SW3-sdk Gem. If testing reveals that the amount of file uploading greatly exceeds Amazon's free tier of services this may be moved to a locally hosted datastore. Regardless future users of the application will be able to decide where their specific deployment stores uploaded files.
CapMS will support the uploading of Microsoft files, pdf, videos, and images. This will allow groups to appropriately display their work without AWS having to account for too many file types. CapMS strongly encourages the uploading of pdf so that viewers can see the files the way they were meant to be seen and leaves less to the user having the appropriate viewing tools.

As for organization and security, AWS will automatically create directories and store the directory information in the database. This will allow files to be made secure within the database.

4.2.6 FERPA

CapMS will include features to ensure FERPA regulations are complied with at all times. When a user submits a new file, it will be set to private and stored in a secure location as determined by the institution utilizing CapMS. Because CapMS will simply allow the institution to point to the location of uploaded files, the security of this location is largely in the hands of the institution. At any point in time, a user on a team can request to make a file public. This action will send each member of the team associated with the file a notification asking them if they would like to make the file publicly viewable. Included in the message will be a digital agreement form that follows the FERPA standard for gaining permission to publish student information publicly, as found in section 2.2. This publication will not be activated until permission is granted from every member of the team. This will be accomplished using a count of approvals from users on a team of a specific file.

5 Project Management

5.1 Software Development Methodology

The team has decided to use an agile development methodology in order to encourage a flexible and efficient development environment. Agile development allows for pieces of the project to be implemented in a logical order and allows for design issues to be addressed as the project progresses rather than all at once at the end. It was decided this system best fits a capstone project due to the strict end-of-semester deadline and desire to have a fully functional product at the conclusion of the class. Using a sequential design process like the waterfall method might result in a final product with more features, but also more bugs and glitches as well. The frequent testing of a product using the agile methodology also facilitates changes of the design as needed throughout the semester. Because some changes are inevitable, it will be much more efficient to address these changes during the construction of CapMS instead of after its completion. For example, if we were to decide to change anything on the back end, it is probable that some front end features will need to be altered as well. The earlier this is addressed in the project, the less features will be implemented and therefore less changes will have to be made.

We will also implement a version of the scrum methodology to establish a consistent and effective way of communicating as a team. This includes establishing specific meeting dates and times that fit everyone's schedule to help ensure progress is being made every week. We have developed a testing and development schedule that closely resembles the sprints in a typical scrum-style methodology that we will use to keep development of CapMS on track. Additionally, we will be using a form of test-driven-development (TDD) to fix bugs and other issues with CapMS. Whenever an issue is found, it will be logged and someone from the team will create a test scenario that can only be successful if the issue has been resolved. This creates a clear distinction between features that are considered "working" and those that are considered "broken."

5.2 Testing and Development Schedule

5.2.1 Semester 1 (Spring 2014)

Milestone 1 (02/12/14)

- Team creation
- Idea generation

Milestone 2 (2/28/14)

- Idea meeting with relevant parties.
- Idea finalization

Milestone 3 (03/05/14)

- Meeting with capstone professors
- Flush out requirements analysis

Milestone 4 (03/12/14)

- Team leader established
- Project name chosen
- ERD development

Milestone 5 (03/19/14)

- Discussed design options and requirements.
- Established design
- Started discussion on functionality requirements and desires.

Milestone 6 (04/09/14)

- Identified technological need such as frameworks.
  - Decided on using Ruby on Rails and a bootstrap front-end
  - CMS and back-end to be decided.

Milestone 7 (4/23/14)

- Finalization of project technologies
  - Flat UI- Web Interface
  - Ruby on Rails
  - Refinery CMS- Content Management System
  - Amazon S3- File Storage
  - Heroku- Back-end Deployment
- Began Finalization of semester report

Milestone 8 (05/05/14)

- Semester presentation

Milestone 9 (05/09/14)

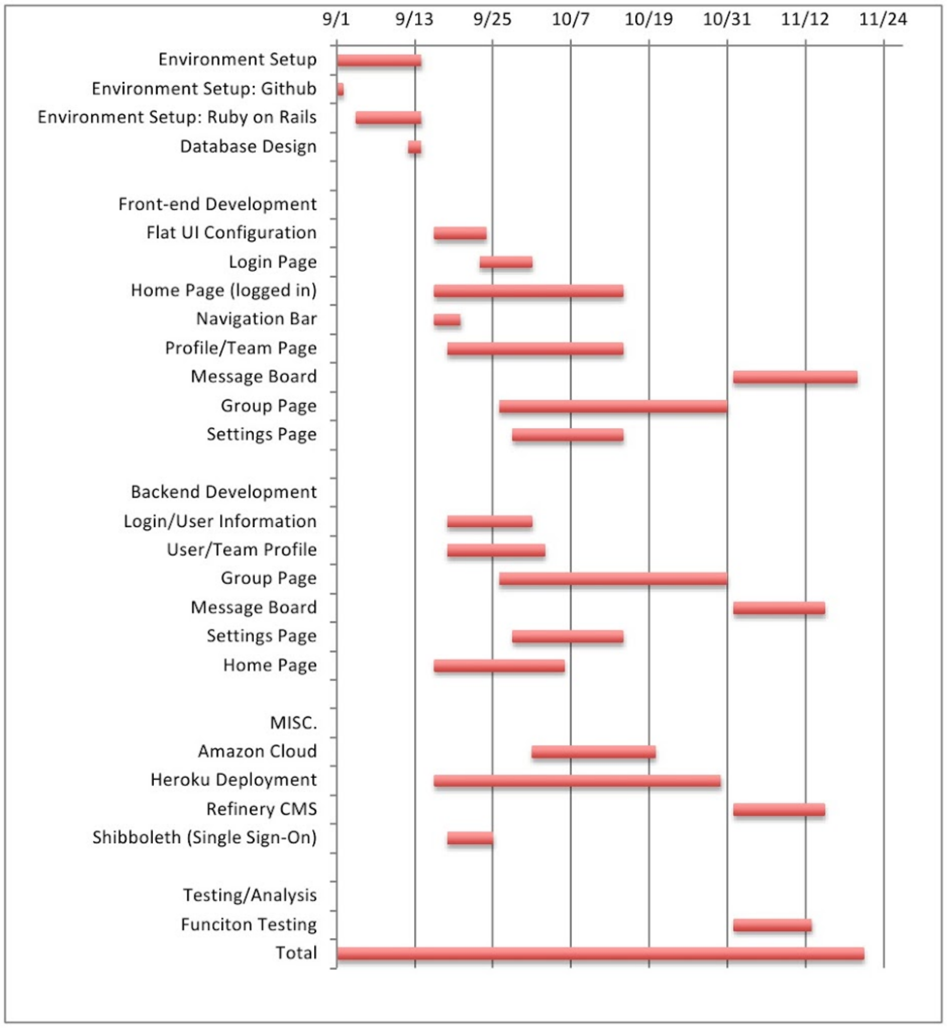
- Final Draft of semester report

5.2.2 Semester 2 (Fall 2014)

CapMS - Capstone II
Fall 2014

Tasks	Start Date	End Date	Duration	Assignee
Backend Design				
Environment Setup	9/1/2014	9/14/2014	13	Team
Environment Setup: Github	9/1/2014	9/2/2014	1	Team
Environment Setup: Ruby on Rails	9/4/2014	9/14/2014	10	Team
Database Design	9/12/2014	9/14/2014	2	Team
Front-end Development				
Flat UI Configuration	9/16/2014	9/24/2014	8	Victor
Login Page	9/23/2014	10/1/2014	8	Danny/John
Home Page (logged in)	9/16/2014	10/15/2014	29	Team
Navigation Bar	9/16/2014	9/20/2014	4	Team
Profile/Team Page	9/18/2014	10/15/2014	27	Daniel/Victor
Message Board	11/1/2014	11/20/2014	19	Danny
Group Page	9/26/2014	10/31/2014	35	John/Daniel
Settings Page	9/28/2014	10/15/2014	17	Danny/Vicor
Backend Development				
Login/User Information	9/18/2014	10/1/2014	13	Danny/John
User/Team Profile	9/18/2014	10/3/2014	15	Daniel/Victor
Group Page	9/26/2014	10/31/2014	35	John/Daniel
Message Board	11/1/2014	11/15/2014	14	Danny/John
Settings Page	9/28/2014	10/15/2014	17	Victor/Danny
Home Page	9/16/2014	10/6/2014	20	Team
MISC.				
Amazon Cloud	10/1/2014	10/20/2014	19	John/Daniel
Heroku Deployment	9/16/2014	10/30/2014	44	Team
Refinery CMS	11/1/2014	11/15/2014	14	Daniel/Victor
Shibboleth (Single Sign-On)	9/18/2014	9/25/2014	7	Victor
Testing/Analysis				
Funciton Testing	11/1/2014	11/13/2014	12	Team
Total	9/1/2014	11/21/2014	81	Team

CapMS - Capstone II
Fall 2014



5.3 Testing Plan

Test 1

- Test initial deployment to Heroku
- Test live URL
- Traverse navigation bar and test links
- Check formatting of placeholder text
- test GitHub installation
- test Rails installation

Test 2

- Querying and posting data.
- Check table dependencies

Test 3

- Loading dummy data from database to web app.
- Posting data (message boards, user profile creation and editing, team profiles creation and editing).

Test 4

- Upload and view files.
- Communication from database to Amazon S3

Test 5

- Single sign on
    - incorporating University single sign on to access web app
- General create user
- Privacy testing
- Published vs. Unpublished documents
- Member vs. Non Member privileges

Test 6

- Publish documents
- Email notifications
- Going from private to public

Test 7

- Using CMS edit the shell of web page
- ex: changing University of Missouri Capstone to University of Illinois Capstone.

Test 8

- Simulate user case
    - Test hypothetical semester

5.4 Required Resources and Costs

5.4.1 Hosting Server

A web server to host the application as the development team creates it. This will also be used for testing the application.

- Type: This will be a cloud based server provided by Heroku.
- Capacity: Traffic will be minimal for most of the testing, Heroku will allow for scaling up to see how the app handles larger traffic flows.
- Cost: This will be a free service.

5.4.2 Developer Machines

Each member of the development team will require a laptop to host their development environment.

- Type: Macbook Pro 13” 2.5 GHz will have sufficient processing power for our development.
- Cost: $1199 x 4 = $4800

5.4.3 Space

Developer work will be split between working together in a space that we will require and working remotely. We believe the space required can easily be found for free either on the University of Missouri’s campus or coffee shops nearby.

5.443 Internet Connection

Most of our development and testing will rely on a solid internet connection. Any internet downtime will cost the team time and money.

- Company: Mediacom Cable
- Package: Prime Plus Internet. Download speeds up to 30 Mbps, upload speeds up to 2 Mbps.
- Cost: $44.95 per month x 3 month development period = $135

5.4.5 Employee Costs

Our development team will be working for an estimated 9 week time table.

- Rate: $20 per hour
- Time: 8 Hours of work per developer each week
- Single Employee Cost: $20 * 8 hours per week * 9 weeks = $1440
- Total Cost: $1440 * 4 = $5760

5.4.6 Additional Costs

Flat UI Pro - $30 - Provides access to additional front end designs and logos

5.4.7 Summary of Costs

Developer Machines………...	$4800
Internet Connection………….	$135
Employee Costs……………..	$5760
Additional Costs……………..	$30
	$10,725

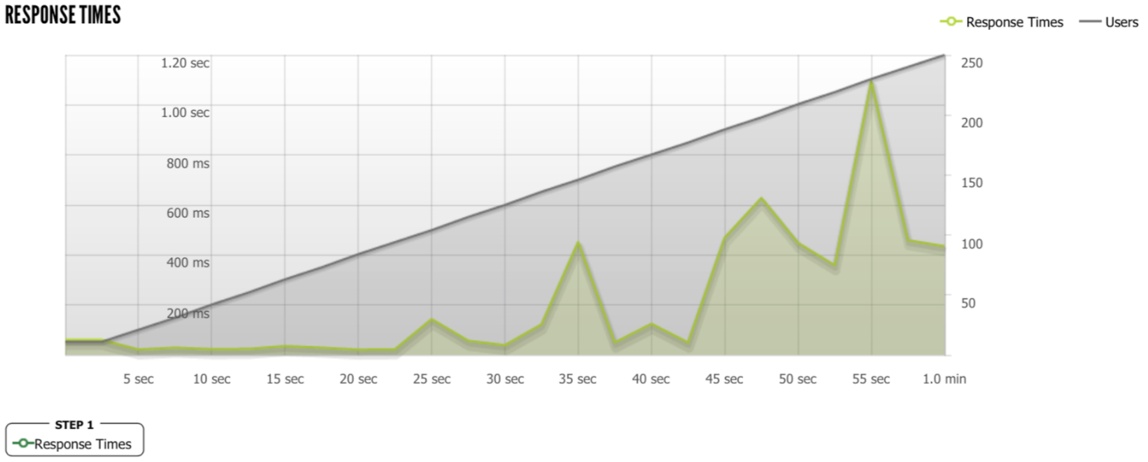
Our total cost to create this web application in 9 weeks will be $10,725.

6 Results

6.1 Blitz Load Testing

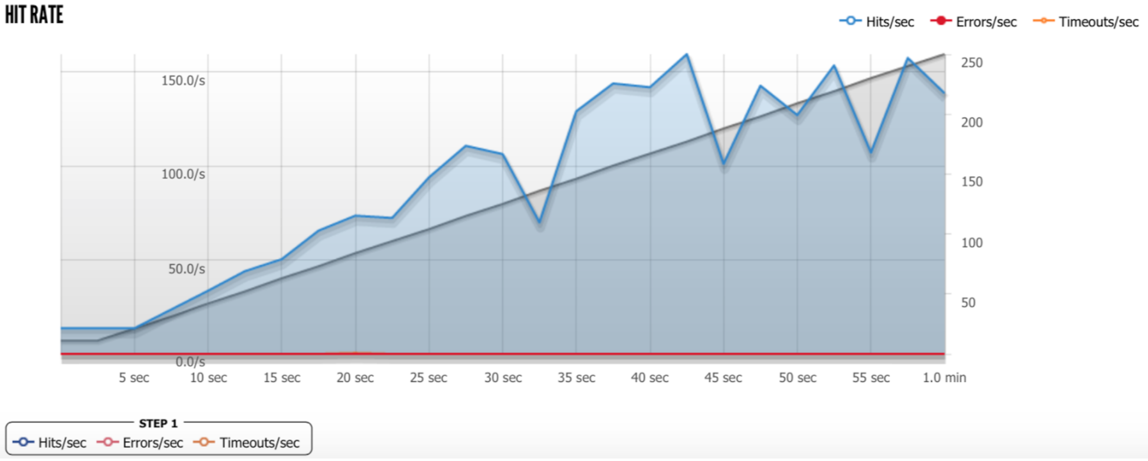
Quantitative load testing was carried out by using the third party application Blitz. Attached to this document are the results of two different blitz tests, one with 100 concurrent users and another with 250. Each test lasts one minute and increases the load from one to the max amount of users set, increasing the hits per second at the same time. CapMS uses two different forms of Platform as a Service (PaaS) to handle traffic to the application as well as asset storage. Heroku is used as the web server of the application and Amazon Web Services handles the storage of user assets. The design decision to use cloud based PaaS allows our application to be truly scalable, and essentially have no hard limits on traffic or storage. Heroku, the PaaS that acts as our applications web server handles traffic through lightweight unix style containers called Dynos. The number of dynos allocated for our app can be increased or decreased at any time, giving the app the ability to automatically scale up if traffic requires it. Heroku's free tier includes one dedicated web dyno and the attached testing results show that one web dyno will be more than sufficient. Load testing through Blitz showed that using a single dyno our app was able to handle 250 concurrent users with an average response time of .217 seconds. Additionally the same test shows a 99.98% successful response rate after 5661 hits to the application in one minute. If for some reason the application did need more than a single web dyno, the pricing is prorated down to the second so the owner would only be charged for the exact time of increased traffic. CapMS uses Amazon Web Services (AWS) for storing students and team assets. These include profile pictures, student resumes, and any team related files such as videos, pdfs, etc. Like Heroku, AWS is automatically scalable resulting in zero expected down time if the limits of the current tier are reached. The free tier of AWS includes up to 5 gigabytes of data. Our development team does not expect a classroom to exceed this free tier, but as previously stated the owner of the classroom could pay for additional storage if and when needed.

RESPONSE TIMES



The max response time was: **1094 ms @ 230 users**

HIT RATE



DATE: 11/17/2014

TEST FROM : VIRGINIA

Query URL: http://secure-lowlands-5959.herokuapp.com:80

Started at: Mon Nov 17 2014, 05:32:03 -06:00

Finished at: Mon Nov 17 2014, 05:32:03 -06:00

ANALYSIS

This rush generated **5,661** successful hits in **60 seconds** and we transferred **4.03 MB** of data in and out of your app. The average hit rate of **94/second** translates to about **8,151,840** hits/day.

The average response time was **217 ms**.

RESPONSE TIMES	TEST CONFIGURATION	OTHER STATS
FASTEST: **22** MS	REGION: **VIRGINIA**	AVG. HITS: **94** /SEC
SLOWEST: **1,095** MS	DURATION: **60** SECONDS	DATA TRANSFERED: **4.03** MB
AVERAGE: **217** MS	LOAD: **1-250** USERS	

HITS **99.98%** (5661)

ERRORS **0.00%** (0)

TIMEOUTS **0.02%** (1)

HITS

This rush generated **5,661** successful hits. The number of hits includes all the responses listed below. For example, if you only want **HTTP 200 OK** responses to count as Hits, then you can specify **--status 200** in your rush.

CODE	TYPE	DESCRIPTION	AMOUNT
302	HTTP	Moved Temporarily	5661

HTTP 302 MOVED TEMP... **100%** (5661)

TIMEOUTS

The first timeout happened at **20 seconds** into the test when the number of concurrent users was at **84**. Looks like you've been rushing with a timeout of **2000 ms**. Timeouts tend to increase with concurrency if you have lock contention of sorts. You might want to think about in-memory caching using redis, memcached or varnish to return stale data for a period of time and asynchronously refresh this data.

6.2 Future Work

6.2.1 Content Management (RefineryCMS) Requirements

There are a number of prerequisites for using RefineryCMS. This is the specific reason why we were not able to implement Refinery. With our implementation of Rails the version that we were using was to up to date and Refinery required us to lower our dependencies of gems. To change the dependencies of the gems would have required us to essentially change our whole website. Once the version of Refinery is up to date with the Ruby on Rails version then Refinery would be ready to be implemented. Below are the steps to implement Refinery.

6.2.1.1 Install Ruby on Rails.

Refinery is a Content Management system that runs with Ruby on Rails therefore you need to make sure that you have a working version of Ruby in order to work with Refinery.

6.2.1.2 Have the RubyGems that are recommended, installed

The only thing that Refinery's website tells you to do is to make sure that the latest version of Gems installed

6.2.1.3 Install and configure a database

Refinery's website has SQLite, MySQL and PostgreSQL as there recommended databases to install.

6.2.1.4 Install ImageMagick

ImageMagick gives a lot of people trouble when they are working with refinery. In the research paper on Content Management Systems it talks more about this. The main problem is that the versions don't always line up and that causes the system to crash.

6.2.2 Mobile Testing

Although we used a mobile first development methodology, we were not able to accomplish quite as much mobile testing as desired due to extensive testing for the desktop. While the application should work well on mobile from using both Twitter Bootstrap and relative sizing in CSS and Sass files, more extensive mobile testing needs to be performed in the future.

6.2.3 FERPA Compliance

The CapMS website currently abides by FERPA regulations in all areas except potentially one. Currently the message when agreeing to make a file public might not be robust enough to meet FERPA standards. While it is clear that the file will be made viewable to the public, there is no digital signature and date set up. In the future, some sort of digital signature needs to be acquired.

6.2.4 Paperclip

Due to the lack of experience with paperclip there are improvements that could be made to clean up and make our code for efficiency. We were unable to utilize the code efficiency because we were unaware of the limitations of paperclip and how it behaved with our database. With paperclip, each file that gets uploaded has fields automatically generated for it. This means that each file that we tried to create on a team would then in turn just be replaced on the team. Therefore we had to create a specific number of allowed files per team. What we would want to do is to put these files in a separate table and manage them by file with a team association rather than a team will file associations. This would allow for more uploads and much cleaner code.

7 References

[1] http://www2.ed.gov/policy/gen/guid/fpco/ferpa/leg-history.html
[2] http://www.ed.gov/policy/gen/guid/fpco/ferpa/index.html
[3]http://www.ecfr.gov/cgi-bin/retrieveECFR?gp=&SID=dd7f593706cea3fab4436fdd36a920b5&r=PART&n=34y1.1.1.1.33
[4]
http://www.facultyfocus.com/articles/teaching-with-technology-articles/ferpa-and-social-media/
[5] refinerycms.com
[6] http://en.wikipedia.org/wiki/Content_management_system
[7] http://www.imagemagick.org/
[8] https://devcenter.heroku.com/articles/getting-started-with-rails4
[9] https://www.heroku.com/about
[10] https://devcenter.heroku.com/articles/quickstart
[11] https://signup.heroku.com/signup/dc
[12] https://devcenter.heroku.com/articles/dynos
[13] https://devcenter.heroku.com/articles/http-routing
[14] https://www.heroku.com/pricing
[15] https://devcenter.heroku.com/articles/logging
[16] https://devcenter.heroku.com/articles/dyno-size
[17] www.rubygems.org
[18] http://guides.rubygems.org/rubygems-basics/
[19] http://bundler.io/v1.6/man/gemfile.5.html

8 Appendices

8.1 Documentation/Tutorials/APIs

Refinery CMS guide: http://refinerycms.com/guides

Heroku documentation/guide: https://help.heroku.com/

GitHub main website: https://github.com/

FlatUI showcase: http://designmodo.github.io/Flat-UI/

Ruby on Rails tutorial: http://www.railstutorial.org/book/frontmatter

Amazon Web Service main website: https://aws.amazon.com/

8.2 Biographies

8.2.1 Danny Glover Bio

Hi my name is Danny Glover. I call Seattle home but have been here at Mizzou for the last 4 and 1/2 years. I'm open to almost any type of project but have the most experience in developing web applications. I believe my education and experience will allow me to contribute results to a team from day one.

Education
• I am currently working towards a B.S. in Computer Science and completed a minor in
mathematics and business. I plan on completing my degree in December of 2014.!
• I originally pursued a degree in mechanical engineering and completed 6 semesters of
coursework. Originally studying mechanical engineering has given me a strong background in physics, thermodynamics, and other related physical sciences.

Experience
• I have been employed for 6 months as a software engineer for Brookside Properties, building an API and a custom CRM client that focuses on software longevity and efficiency. This position has given me the experience to analyze client needs and develop and deploy code in a timely fashion.
• My research project at Shanghai University in the summer of 2013 gave me valuable
experience programming in a fast paced team environment. I developed a mobile game for Android and learned some of Java's more complex networking techniques such as sockets and multithreading. It also provided me with insight into Chinese culture and software development. I look forward to building something awesome with you this semester!

8.2.2 Daniel Rodenberg Bio

My name is Daniel Rodenberg and my degree is a bachelor's of science in Information Technology. It took me a while to find my major and I didn't switch to IT until my second semester sophomore year. After that semester I got two internship offers and accepted an offer from Cerner Corporations. I have been working for them since the summer of 2012. They have a nice program where they allow you to work through the school year and that's what I have done for almost two years now. I'm a software intern there and mainly work with a database team organizing all of the databases and making sure that the databases have the programs on them that make them run properly.

I have taken a number of computer science classes. I took the intro to the web class, object oriented, database, and software engineering. My primary focus is Web Applications. I have been working with a friend on a website where we have been using Ruby on Rails. This has been something that I have really enjoyed.

One of my favorite things to do is watch and play sports so if there is a team that is planning on working on something sports related then I would gladly join. Another thing that sounded interesting to me is working on a robot. I know that Dale talked about Matthew Dickinson having robots available for students to work with.

8.2.3 John Curley Bio

I am a currently a senior with a planned graduation date of December 2014 with a BS in computer science. I also graduated in December of 2012 with a BS in mechanical engineering. I decided to pursue a computer science degree sometime in the last year of obtaining my original degree after realizing that I have a much greater passion for computer science and programming than I do for the mechanical engineering field. Ideally I wish to pursue a career in which I can utilize both degrees, although I realize that these types of jobs are not at all common. Thus, more realistically, I see myself pursuing a more typical career in the computer science field

My time in the mechanical engineering program has helped me with many aspects of computer science courses and I believe that they will especially help with capstone. Not only was technical writing a major component of the mechanical engineering coursework, but my experience with a capstone class in the past will help with project planning and execution. I also have experience with two different internships where I worked on large scale projects. These internships were with BIS Frucon Engineering in Ballwin, MO and Conagra Foods in Marshall, MO. I have experience with C, Java, PHP, HTML, and SQL as well as MATLAB software. I am interested in mobile development and web application development but I am open to any idea that is creative and/or unique.

I am currently on a team with Victor Tran and Danny Glover and while we have discussed many options for our project, we have not yet decided specifically what we will be working on.

8.2.4 Victor Tran Bio

My name is Victor Tran. I am a CS/IT major and already have a degree in philosophy. I have taken all the normal course work that are prerequisites to this class such as Database and web development. Last semester the majority of my classes involved web based applications. The first one was in database. My group created a web app that took information from the city of Chicago website and using google maps, pin crimes that have occurred throughout the city. On top of that we mapped out the L train stops and rated them in safety. The site also allowed the user to enter start and stop addresses and maps out crimes that occurred along the way. The second web page that I made was for my web development class. I made a joke webpage to find my friend a lover. I made galleries and a working blog that can be updated using log information. On top of these classes I've also taken a lot of IT classes such as video editing and digital effects. This semester I am enrolled in the mobile app development class which I am very excited to take.

As for my group in this project I talked to my group from my database class and the three of us, John Curley, Danny Glover and I were wanting to work together. I don't think that we are opposed to another person or two joining us. We sort of discussed doing a project using mobile platforms or something of that nature but nothing really jumped out to us quite yet.