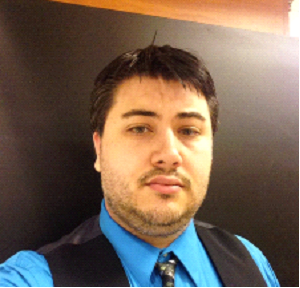
1/1/2014

Johnathan Armke, Kavy Rattana, Kyle Willcox

Hawk Overflow

Hawk Overflow

UNCW’s Code Forum



Johnathan Armke

Lead Developer



KavyRattana

Database Architect

QA Specialist



Kyle Willcox

Project Manager

Table of Contents

Contents

[**1.** **Introduction** 3](#_Toc405262587)

[**2.** **Background and Related Work** 7](#_Toc405262588)

[**3.** **Resources** 10](#_Toc405262589)

[**4.** **Delegation of Responsibilities** 11](#_Toc405262590)

[**5.** **Unexpected Problems** 13](#_Toc405262591)

[**6.** **Budget** 14](#_Toc405262592)

[**7.** **Risk Plan - RMMM** 18](#_Toc405262593)

[**8.** **Constraints and Restrictions** 19](#_Toc405262594)

[**9.** **Major Software Functions** 20](#_Toc405262595)

[**10.** **Requirements Use Case Diagram** 21](#_Toc405262596)

[**11.** **Design Use Case Diagram** 22](#_Toc405262597)

[**12.** **Requirements Activity Diagrams** 23](#_Toc405262598)

[**13.** **Design Activity Diagrams** 26](#_Toc405262599)

[**14.** **Requirements Activity Diagrams** 27](#_Toc405262600)

[**15.** **Design Models** 28](#_Toc405262601)

[**16.** **Sequence Diagram** 31](#_Toc405262602)

[**17.** **State Diagram** 34](#_Toc405262603)

[**18.** **Requirements/Data Dictionary** 35](#_Toc405262604)

[**19.** **PROJECT TESTING** 37](#_Toc405262605)

[**20.** **Non-functional Requirements** 39](#_Toc405262606)

[**21.** **Bibliography** 40](#_Toc405262607)

# **Introduction**

**1.1 Importance of Project:**

Hawk Overflow, both as a concept and as a reality, is extremely beneficial to the students and staff of this university. As a resource for people to share helpful information about the course they are taking, Hawk Overflow is an unprecedented resource targeted specifically at UNCW’s needs. This application could make the process of teaching computer science principles more effective and give students the ability to learn from students before them. While the computer science program is designed to be challenging while remaining effective at teaching programming, real-world software engineers use question based forums and other forms of colleague collaboration every day to aid their work.

Our mission at Hawk Overflow will be revolutionary in design to inspire understanding of core concepts, rather than copy and pasting source code from other developers. Under Creative Commons’ CC BY-SA ShareAlike license, users can even submit what they feel would be improvements to the interface and running of the site, without concern of copyright or other creative infringement [4].

**1.2 Similar Projects:**

Hawk Overflow was inspired by a well-known software engineering question based message board called Stack Overflow. StackOverflow is the flagship site of the stack exchange network and features questions and answers on a wide range of topics in computer programming. [1]. The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers in a fashion similar to the Wikimedia Foundation [10] or Digg [9]. As StackOverflow is also a proponent of the CC BY-SA, we feel this is an excellent modeling choice for our project.

**1.3 Issues to be addressed**

One major concern the Hawk Overflow team had when designing the user experience of our site was preventing the sharing of assignments or answers on our site. This is an ethically immoral issue because not only would the site prevent the student from thinking how to code on their own but supplying assigned answers to homework would by pass in the initial purpose of the assignment: to learn through individual thought [11]. To combat this we are giving professors the ability to delete questions they feel are too directly related to an assignment or test. This and preventing students from uploading source code to the site should suffice in suppressing sensitive information from being shared.

Due to the fact that Hawk Overflow has the potential to share sensitive information about school assignments, security is absolutely imperative. Each user must have an encrypted password that cannot simply be obtained by unauthorized users [12]. Professor and administrators must have special delete/edit permissions that restricted to identified users and their rights will stored in the database. While there are not any legal issues [4] associated with our application, the team attempting to create a change for the better in the way students look for help online about their studies. We are developing this application with the intent that its societal impact will provide a way for students and professors to help each other better understand major concepts in computer science.

**1.4 Problem/Purpose:**

As our team has progressed through the CSC class structure over the past few years, we noticed a trend in the younger generation of students seeking advice and knowledge from more experienced students. The spreading of information about programming concepts and best practices is vital to enlightening and empowering the software engineering community, but this information can be hard to come by when first starting out. Often times searching on the Internet for help reveals results that are way beyond the comprehension of a new computer science student and these students learn best when concepts are simply explained in terms they can understand.

The whole computer science department could benefit as whole by having some way to aggregate and search through questions related to computer science and get answers from people who have been through the same problems. This source needs to be a place accessed by professors and students that contains beneficial information but not to simply give away answers to assignments, instead this source should be a place where some could better understand computer science rather than get source code. Comprehension, not completion, should be the byword of every user of Hawk Overflow.

**1.5 Scope & Objectives:**

Our solution to this problem is going to be Hawk Overflow, the brainchild of a conversation between two of the group members several semesters ago after discussing a Java I project. This web application would be an interactive forum inspired by Stack Overflow that would be purposed for sharing ideas and concepts about computer science within the realm of assignments here at UNCW, rather than the far broader questions about syntax and source code like Stack Overflow. Furthermore, instead of giving students line-by-line instructions to the coding assignment, the site would host answers and examples that seek to describe pseudo-code in words, or explain the meaning or function of certain procedures.

The site should have a dynamic answering system where the best answers are voted upon and appear in order of how well the answer the question. In addition, user profiles will be created to distinguish people who have answered many questions successfully and separate professors from students’ administration rights. Last a searchable hash should be created in order for students to query the question database if the question they are wondering has already been asked or simply learn about future classes for enrollment.

**1.6 Success Criteria:**

Hawk Overflow depends on a large and knowledgeable user base in order to function optimally so the most important success criteria is to accurately manage every question asked and store the meta data within that question consistently. This information needs to found easily using the search feature included on every page. User base is another critical goal we need to meet because without a well functioning user profile system, managing who has rights to what question will be very difficult. Security is will not be overlooked for this site and having encrypted passwords for every user will be a must. Lastly the graphical user interface should be simply put together yet still allow quick and intuitive navigation between pages.

# **Background and Related Work**

**2.1 Abstract:**

To better understand how Hawk Overflow should be used its critical to know how it was designed and why the developers created the user experience the way they did. First understanding a basic problem of why they application could useful to a student is necessary for seeing its main functionality. Next comes the design of the meta-information about the questions posted and how their responses are organized. In addition, understanding what users have certain rights and why some users should be able to edit or remove content is a key concept in keeping the site behavior according to school conduct.[11] Lastly having the question base employ an effective search model to allow for information to easily be found by those looking for it is essential for this site to reach its full potential.

**2.2 Sample Usage**

In order to understand why an application like Hawk Overflow could be useful one must first know the issues faced by a student in the computer science program at UNCW. Computer science assignment are typically assignments that can be solved in a multitude of ways each presenting a different solution or interpretation of a solution that meets the given requirements.[15] For example when Dr. Taglarini asked his students to create a genetic algorithm to solve the Traveling Salesman problem, he got a uniquely written algorithm from each student, but the basis of each algorithm was modeled off an idea that incorporated genetic fundamentals to find optimal traveling routes. So a good question to be posted to the Hawk Overflow site would be “What are the fundamentals of a genetic algorithm?” instead of viewing the source code of a previously created genetic algorithm. [4]

**2.3 Question Meta Data:**

One of the most defining characteristics about Hawk Overflow is the way it will incorporate a dynamic question response system that will allow for users democratically elect the best answer to a question. Each question will open the door for a new set of responses from users to display their knowledge on a particular subject and the best answer will be displayed on the top of the page (right under the actual question) based upon how many up or down votes it received from other users. If a user (possibly the one who asked the question) finds a particular answer to satisfactorily explain the question posed they would choose to vote that answer to the top of the page.[10] However in the instance of a response being incorrect, the user could also elect to vote the answer down to the bottom of the page in an effort to prevent the spread to misinformation.[9]

**2.4 Users and Rights**

Hawk Overflow would be a useless application without a group of dedicated, responsible, and ethical users. So in order to create and manage a secure and functional user-base we are requiring every person to sign in using their UNCW email before the contain to access the information stored on the site. While it is important to address censorship concerns users will have to be made aware of the the code of conduct for the site and know what kind of answers or questions cross the bounds between explaining a concept vs. supplying an answer.[13] It should be the job of the moderators and professors to look at questions and responses pertaining to their classes and be the judge of what content should stay and what should be edited or deleted.

**2.5 Searching Functionality**

Hawk Overflow is being built with the intention that that one day it may contain over a thousand questions spanning a wide range of subjects. So for in order for all this data to be easily found and accessed by those who need it a powerful search tool must be created to query the database. This search tool should work similar to other search engines found online such as Google, Stack-overflow, or Amazon by including fuzzy search functionality and proper distinction of key words. In order to accommodate users who don’t ever know how to express exactly what they are looking the engine must bring back a list of the closest matching questions or responses. Lastly in an effort to better protect the database user input forms will utilize regex pattern matching functions in order to protect against injection attacks or other unwanted query errors.

**2.6 Related Work:**

The primary related work to our project is, of course, the site we are homaging: stackoverflow.com. The primary functions (upvote, downvote, tagging) are already present and well-functioning on the source site. The reason why we feel UNCW needs its own version is the complexity that inherent in a site populated by people from every grade of experience. By pre-conditioning the majority of questions with class tags, we can try to maintain a consistent level of relative experience in the answers. The inability of novices to understand the complexity of some of the responses on Stack Overflow diminishes the site’s ability to impart knowledge at multiple levels, a shortcoming we intend to amend with our work here.

Digg.com and reddit.com also include a dynamic rating system, which allow topics to rise and fall according to their popularity and usefulness. This is done to a very acceptable degree on these sites, and we merely use them as a point of reference to ensure that our standards are sufficiently high.

# **Resources**

**3.1 Dreamweaver:**

In order to design the document object model and the scripting of the site we need a powerful integrated development environment that does not over complicate the development process. Due to the fact that all three our team members have already taken CSC 475 (Web Application Development with Server side scripting) we each already owned and are familiar with Adobe Dreamweaver. This integrated development environment has everything we need to push our code to the site and manage/debug code effectively.

**3.2 Twitter Bootstrap:**

One of the favorite tools our team has used in previous web projects is Twitter Bootstrap for the CSS styling of Hawk Overflow. This beautifully basic pre-prepared CSS styling allows us to apply classes to the document object model giving them a modern look that a quality web application must have in order to be taken seriously in the current era. Twitter Bootstrap will give us to the tools to design our site but ultimately it relies on the developer to use them correctly.

**3.3 PHP/MySQL:**

In order to build a sturdy back end to the site our team is going to construct a database using PHPmyAdmin because it is another tool the whole team is familiar and proficient with. The queries and data management structures are all concepts we have employed before in previous web classes so we are confident in our ability to build a reliable backbone for the information hosted on Hawk Overflow.

**3.4 JIRA:**

To handle the project management side of the project, Kavy introduced us to JIRA, an application to communicate requirements/goals and the issues regarding the site. It will be used to outline user stories, define tasks and write bugs the team needs to be aware of. In addition JIRA has a chat client called HipChat we will use for quick communication between the team and the sharing of documentation.

# **Delegation of Responsibilities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tasks** | **Kyle** | **Jon** | **Kavy** |
| Design database to store questions and user data |  |  | X |
| Write queries to access the correct information |  |  | X |
| Implement a hash cloud within the database |  |  | X |
| Create PHP functions to manipulate model/view |  | X |  |
| Implement a dynamic voting system |  | X |  |
| Create a user ranking hierarchy |  | X |  |
| Design a visually pleasing UI theme | X |  |  |
| Style and place DOM elements accordingly | X |  |  |
| Manage User stories and maintain team velocity | X |  |  |
| Test main functionality as well as edge cases |  |  | X |
| Design system architecture before implementation | X | X | X |
| Create test cases and sprint planning | X |  |  |
| Assess and research potential risks |  | X |  |

*Table 1: Hawk Overflow Responsibility Matrix*

# **Unexpected Problems**

**Meeting planning:**

Due to the fact all three of us having demanding schedules and jobs it becomes difficult finding a time for all three of us to meet in person. Therefore sometimes someone might have to use Google+ or HipChat to take part of the conversation. Facebook and Google Drive was later implemented to share file and communications to expedite updates to the project.

**Satisfying Requirements:**

It is important to be explicit as possible when defining user stories because no matter how detailed a product manager can be it is always possible for the developer to misinterpret them. This leads to wasted development time and unnecessary bugs. In order to mitigate this problem, Kyle, the team manager, will have frequent communication with the developers and make sure Jon and Kavy are writing fluent and cohesive code.

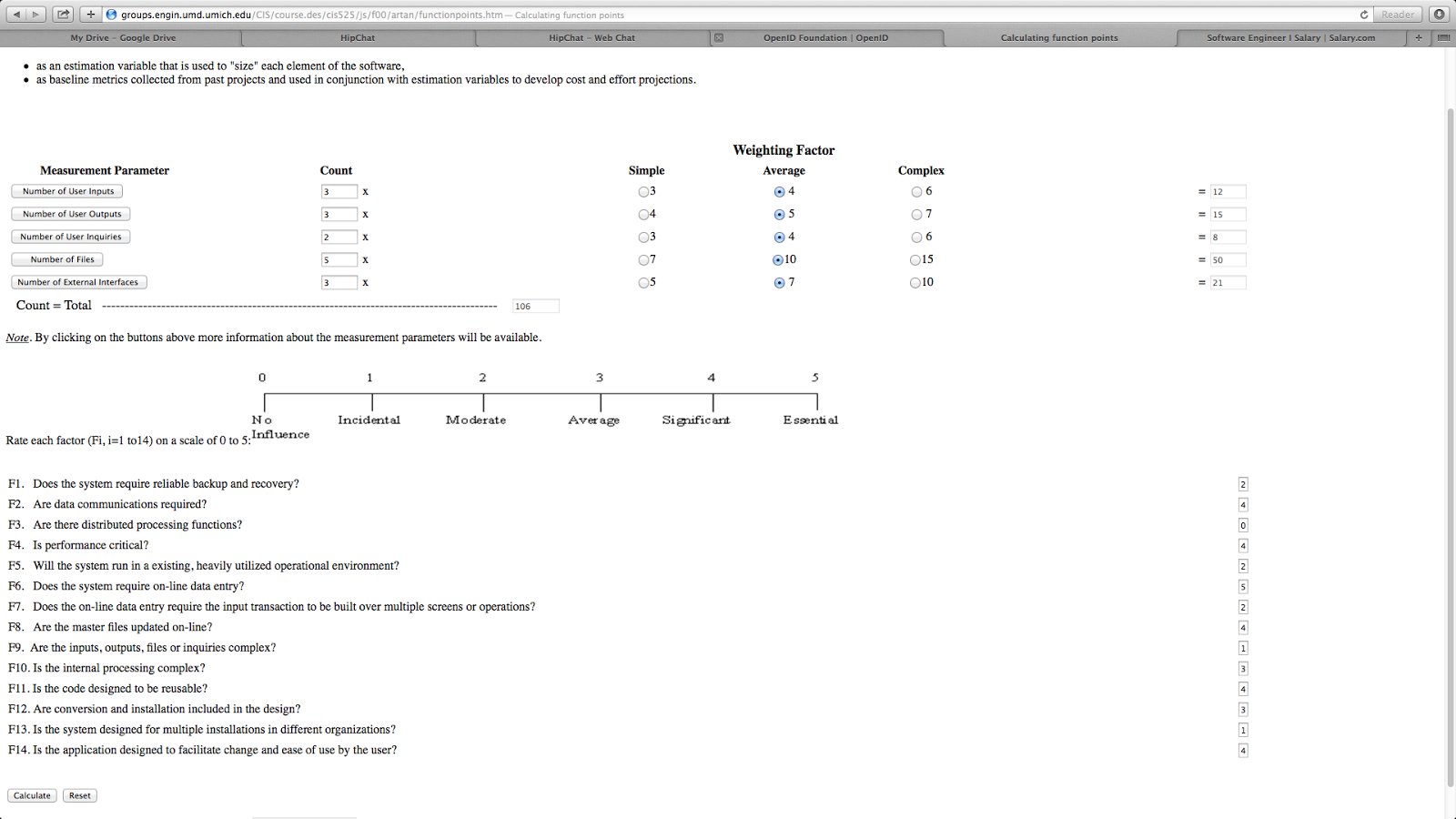
**Software issues:**

Learning how to fluently use GIT and SourceTree to push and pull new items to the repository was a skill that wasn’t universally possessed by the group prior to beginning the project, and definitely slowed down certain updates [18].

# **Budget**

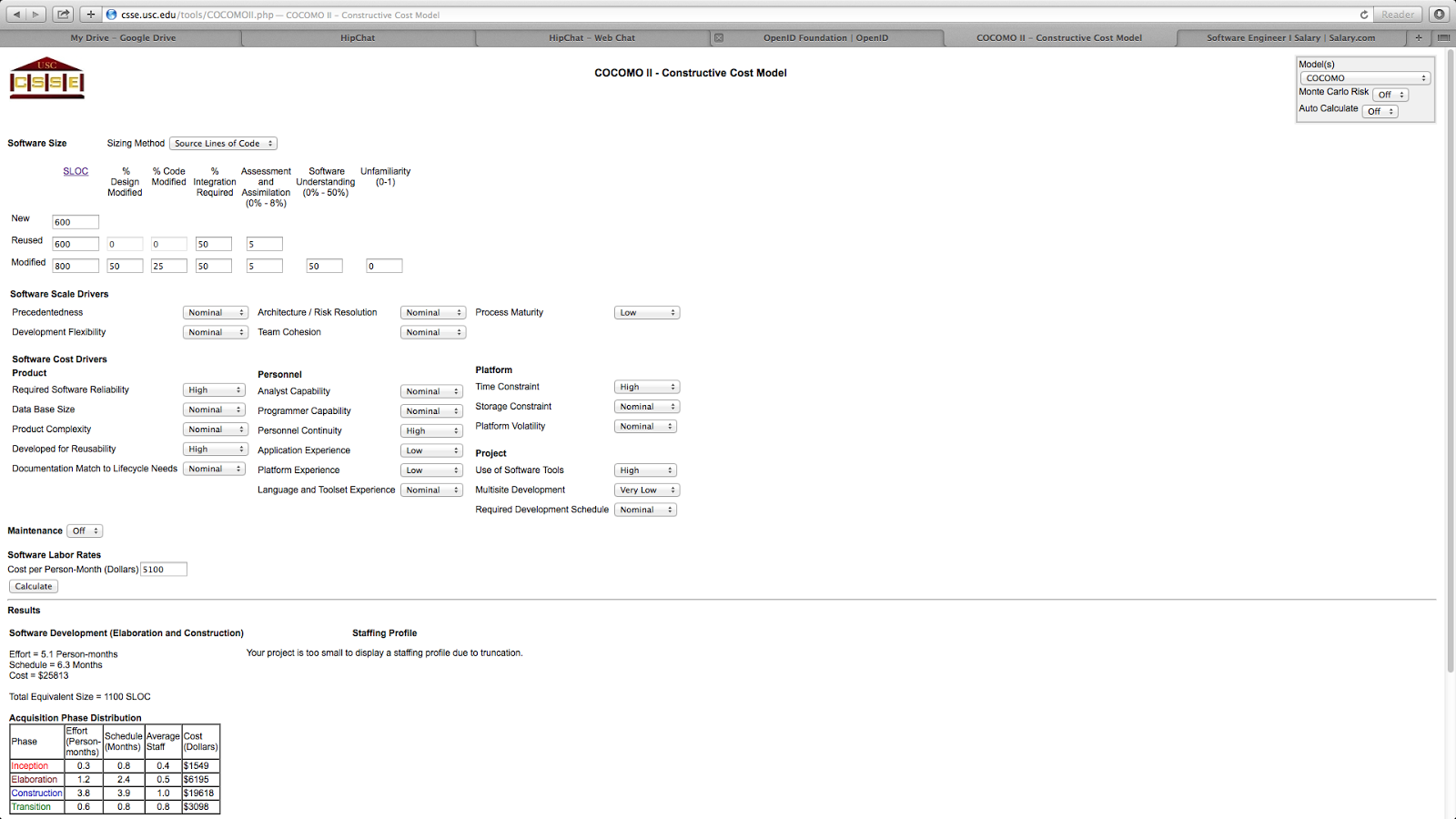
**6.1 Derivation:**

Given code reusability and established trends from both the textbook [6] and USC’s CS department website [13], the lines of code we assume the project will require total roughly 2000 [600 new, 600 reused, 800 modified]. The Function Points, similarly determined, total up to 131, from the original conservative estimates of 35. This was due to improper assessment from design documents of the amount of input/output areas for users, as well as reweighting of the F-factors [14].



*Figure 1. Function Point calculations.*

By the COCOMO II method, we initially calculated the approximate required effort to be 4.6 person months, and the projected schedule to be 2.9 months from date of team formation. Given that the median salary for a software engineer is $61,178, meaning ~$5100 a month, and a projected schedule of 6.3 physical months and 5.1 person months, the projected cost has reduced from the initial projection of $46,354 to $25,813. While the cost has come down, somehow the projected time of completion has increased, meaning that we may have to re-assess certain features of the site. The new figures were written estimating a worst case scenario, so this most likely accommodates the changes in time-constraints. For a full expression of the variables from which the projections stated here are derived, please note the following figure: a screenshot from USC’s COCOMO II Calculator [13].



*Figure 2. COCOMO II Model calculations.*

**6.2 Scheduling**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Mid Sept** | **End Sept** | **Mid Oct** | **End Oct** | **Mid Nov** | **End Nov** | **Dec [Final]** |
| Prototype [Rough] | X | X |  |  |  |  |  |
| Initial coding – Back end |  | X | X |  |  |  |  |
| Initial coding – front end |  |  | X | X | X |  |  |
| Apply visual standards |  |  |  | X | X | X |  |
| Front end Testing |  |  |  |  | X | X | X |
| Back end Testing |  |  |  |  |  | X | X |

*Table 2: Scheduling Sequence (Tentative)*

**6.3 Model used:**

Due to the time constraints previously discussed, the schedule model we used is the waterfall, with some overlaps at critical points and/or areas where backtracking might be needed.

# **Risk Plan - RMMM**

**7.1 Mitigation:**

We have determined our schedule patterns early, and flexibly. This is to avoid impacts from other commitments. We have also pre-defined our standards for peer review, so as not to miss critical details later, in any rush to completion.

**7.2 Monitoring:**

We, as near-graduates, are all painfully aware of the need to maintain strong positive morale regarding the completion of the project. Similarly, we know the value of good communication, and have established multiple channels [HipChat, JIRA, Bitbucket, Facebook, etc.] to uphold regular high-quality communication.

**7.3 Management:**

Due to the plan’s basic composition, as broken out in this dossier, rapid intervention is possible, though not ideal. If meetings/deadlines are neglected, adjustment of responsibilities can be made, but will be communicated during submission of peer reviews.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Low Impact | Moderate Impact | High Impact |
| High Probability | Time delays due to scheduling conflicts. | Error in prototype, all descendant items reworked. | No current items. |
| Moderate Probability | Search functionality not behaving correctly. | Difficulty learning new tools for deployment and monitoring. | Missed deadline for presentation/submission at some midpoint. |
| Low Probability | Commitment of improper code, forcing a reversion to older branch. | Overwriting an entire code branch, requiring immediate rewrites. | Team member drops out. |

*Table 3: Risk Assessment*

# **Constraints and Restrictions**

Technological restrictions include the limitations of accessibility of the servers in the event of natural disasters or the like, but the importance of paper-prototyping is not unknown to us.

Time constraints have been met as best as possible given the nature of being working students with some measure of social responsibility. This is reflected in the schedule described previously, as well as in our myriad of communication and submission tools.

# **Major Software Functions**

**9.1 Student functions**

* Log-in / Log-out
* Manage own Questions
* Ask questions
* Modify own questions
* Reply to questions
* Favorite questions
* Up & Down Vote

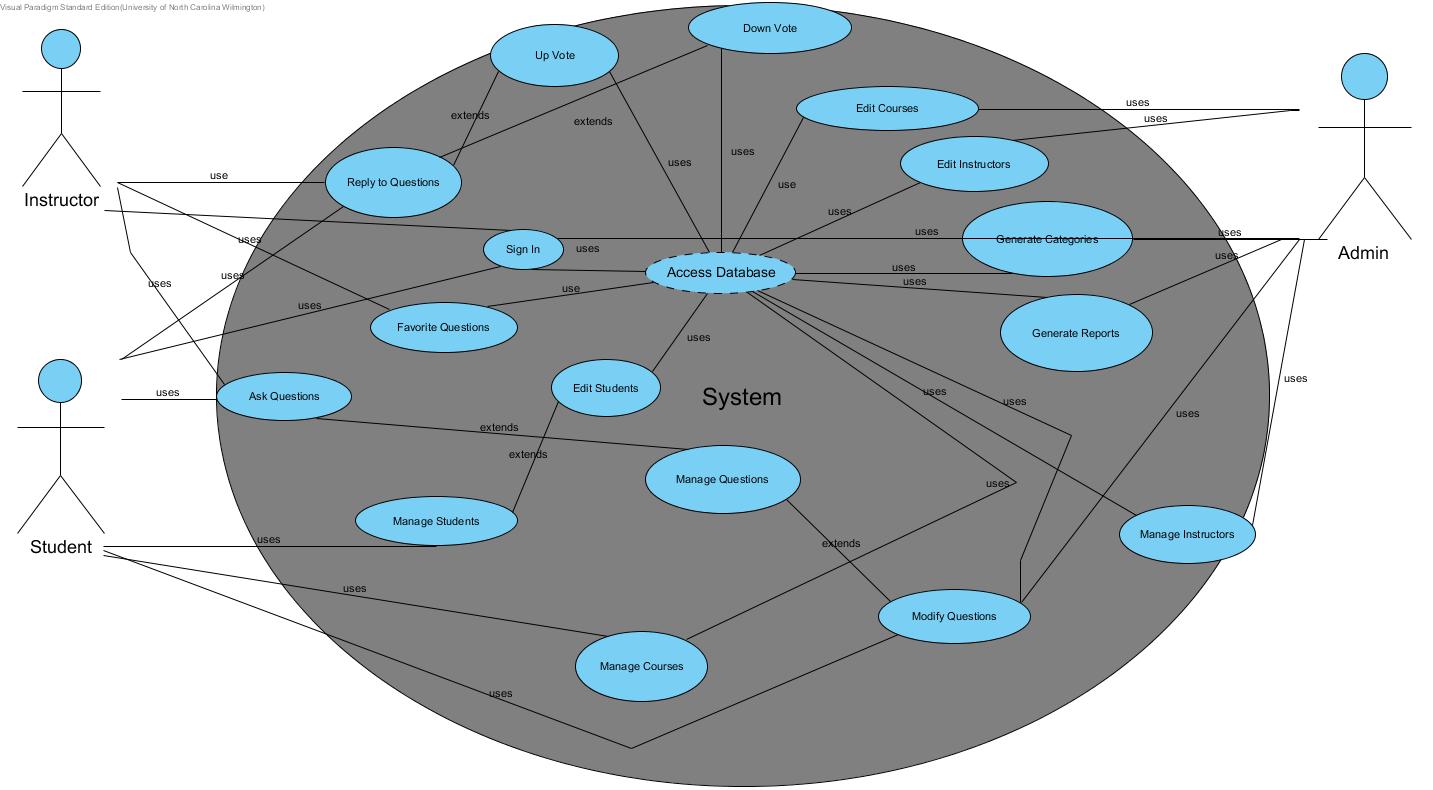
**9.2 Instructor functions**

* Manage students
* Manage courses
* Generate Reports
* Student functions

**9.3 Admin functions**

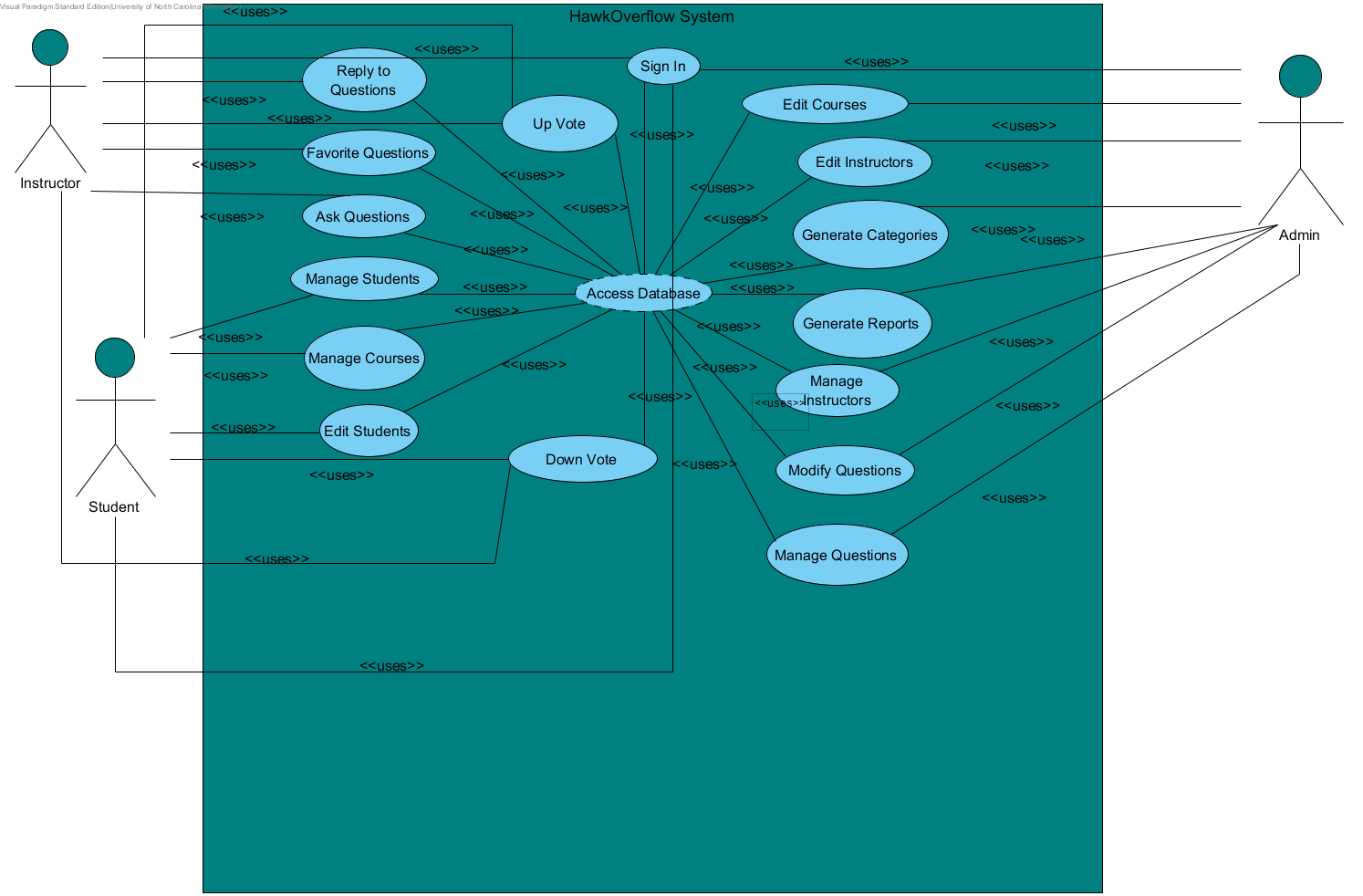
* Generate Reports
* Create categories
* Manages users
* Manage instructors
* Manage questions
* Student and Instructor Functions

# **Requirements Use Case Diagram**



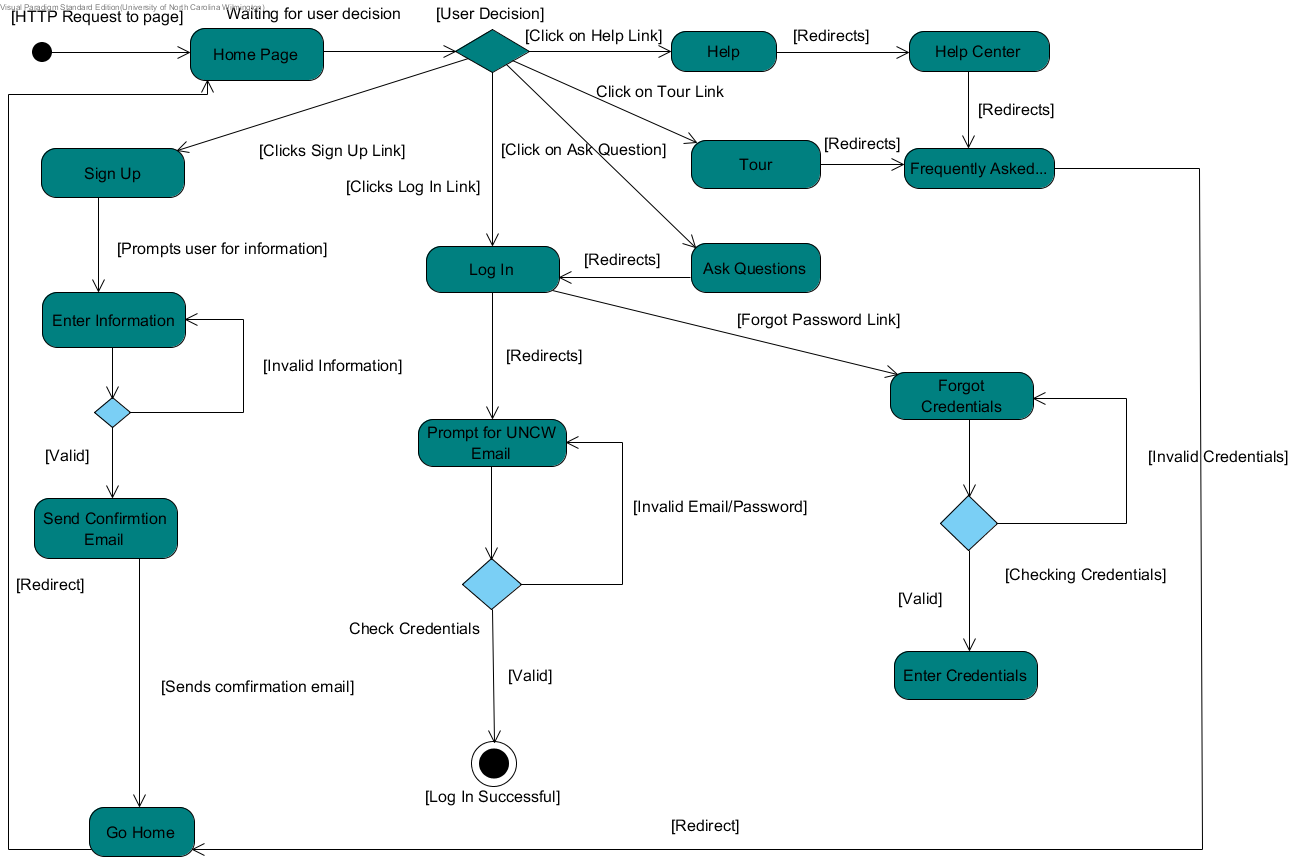
Figure

# **Design Use Case Diagram**

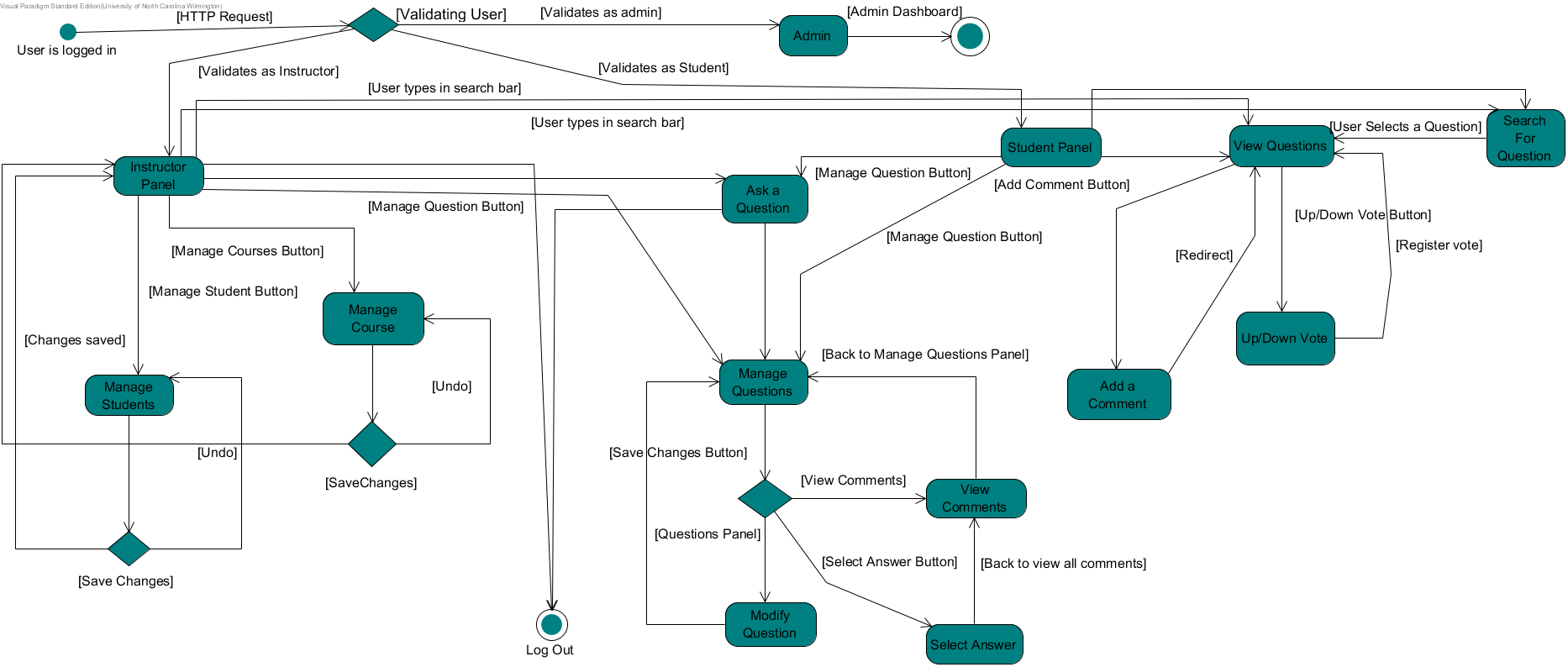


Figure

# **Requirements Activity Diagrams**

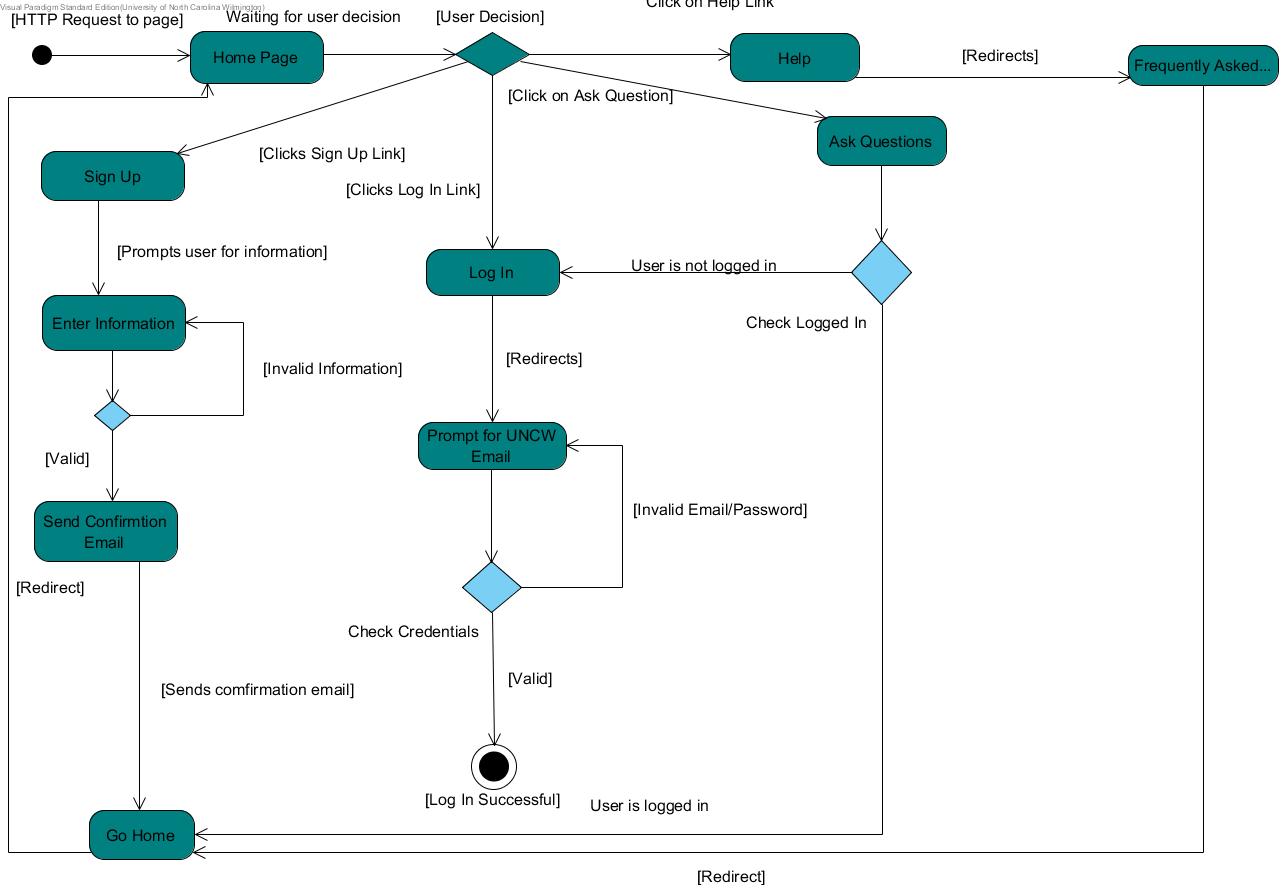


Figure

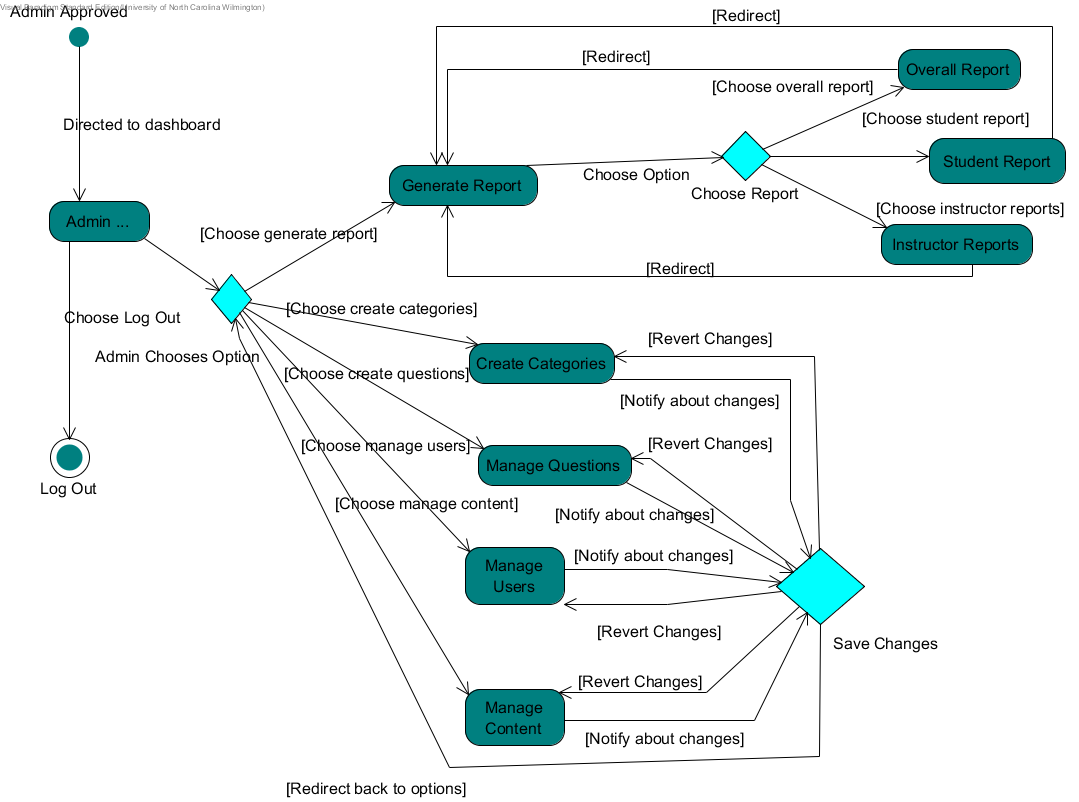


Figure

# **Design Activity Diagrams**

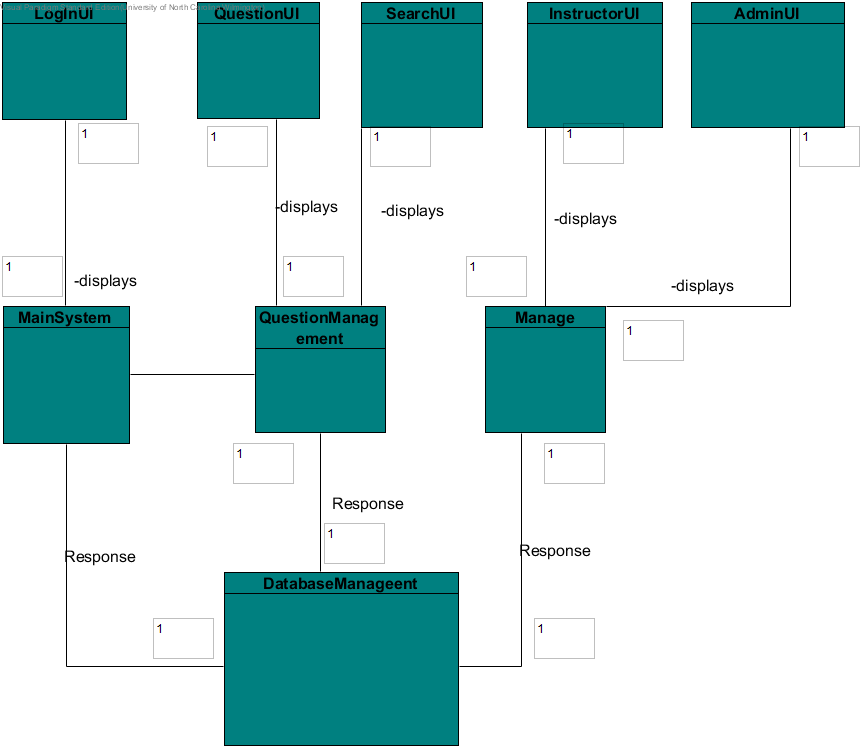


Figure



Figure

# **Requirements Activity Diagrams**



Figure

# **Design Models**

**16.1 Design Constraints:**

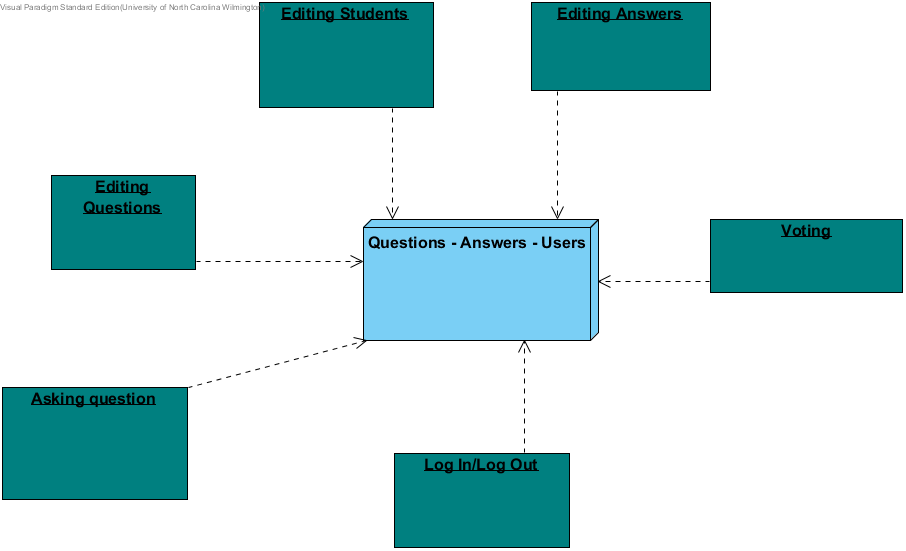
The software will the designed to run completely over the Web. A internet connection is required at all times to be able to use any of the functionality. Therefore, the design of the software will be as minimalistic as possible for it to run on any device. The design should be well simplistic enough to run on general browsers and display on mobile devices. The website will work around a login so that permissions are given out accordingly.

**16.2 Architectural Design:**

The architectural design that is used to represent the project are the three-tiered style and the shared-repository style. The web application is primarily centered around a database which if feeding information to other systems. Information about the questions, answers, students, teachers and even admins is continually being requested from the database. To handle all the types of information, subsystems are created to handle and arrange them to display to the UI at the top level.

In figure 14.1, a three-tier design architecture displays the different levels. The UI is at the top which will request information from one of the subsystems Then, that request is taken to a section for the type of data. Finally, data is retrieved from a central database and fed back to the UI.

The share repository is the second architecture used to represent the web application. Since this is specifically a web application that pulls large amounts of data every second, everything is set up to communicate with the database. Without it, none of the features of the web application will not function at all.



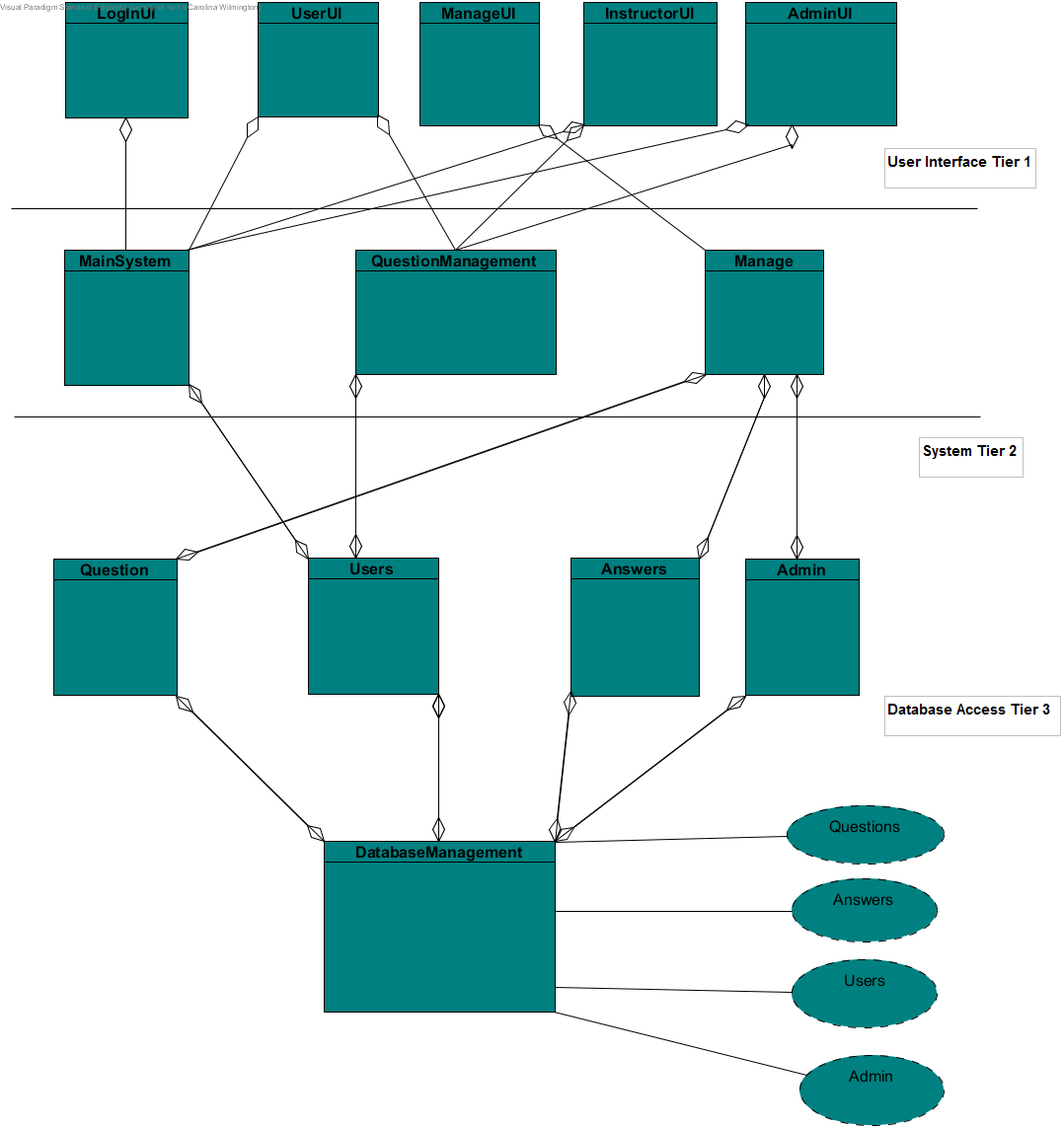
Figure

**Subsystem Design:**

Each subsystem design allows functionality for a particular part of the web application. The login logout subsystem is responsible for signing in and signing out users as well as giving permissions to the correct user whether it be a student, teacher, or admin. The questions subsystem is responsible for handling all answers and questions functions including creating, editing, and deleting. If any changes are made to any of the subsystem, it will automatically be queried to the database. The users will always be navigating within user interfaces as long as they are logged into the system.

**Class Diagram of Subsystems**

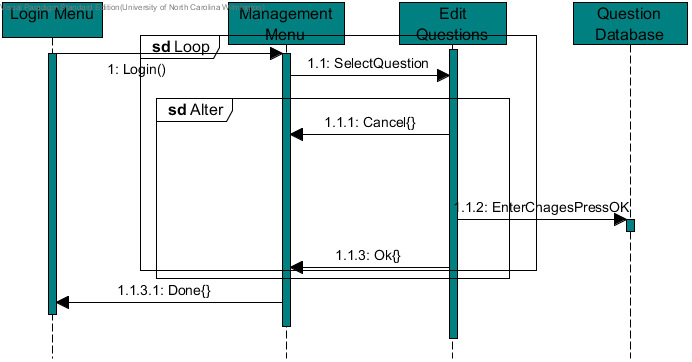
To edit questions, the user first has to login to the web application and further verification will commence. Once the type of user has been verified, the user interface for the management menu will change based on the permissions that have been set. The user will choose the question that they would like to edit. Once the user has made all changes, they can press “OK” which will then query the database. Pressing the cancel button will reset the fields for the user to start over. The user will continue to follow the same loop when editing questions.



Figure

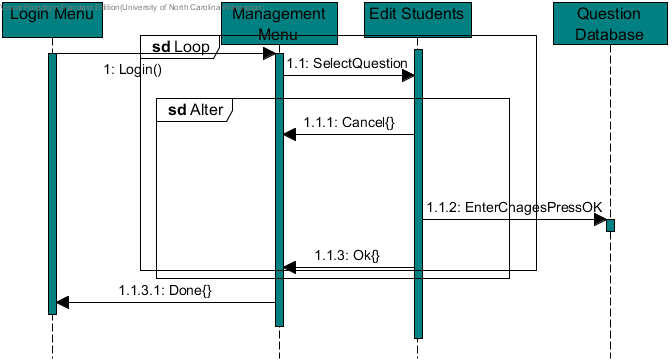
# **Sequence Diagram**

**Edit Sequence Diagram**



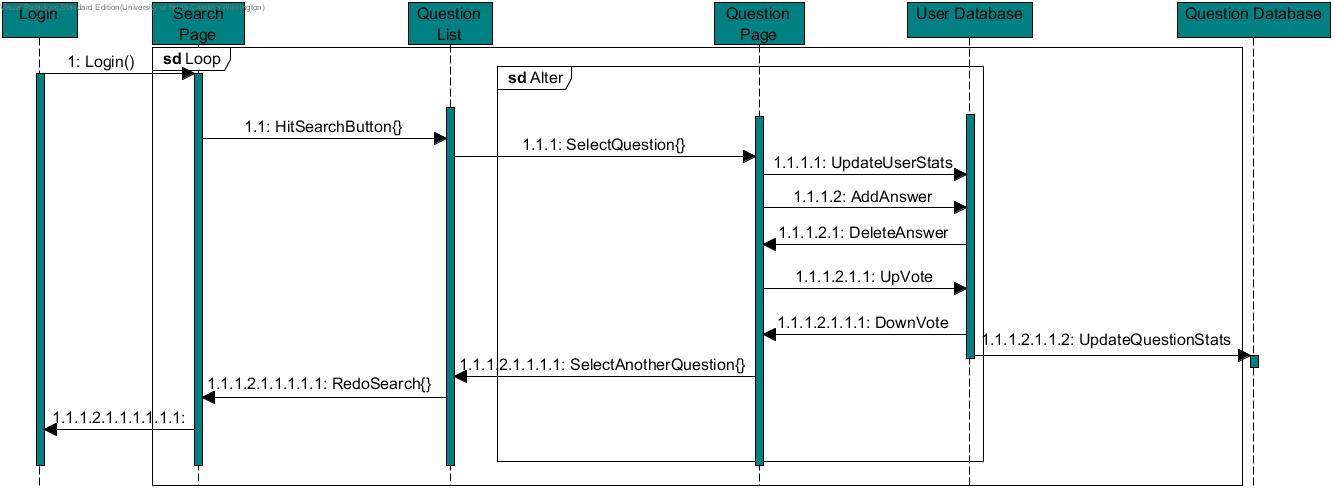
Figure

**Edit Students Process Diagram**



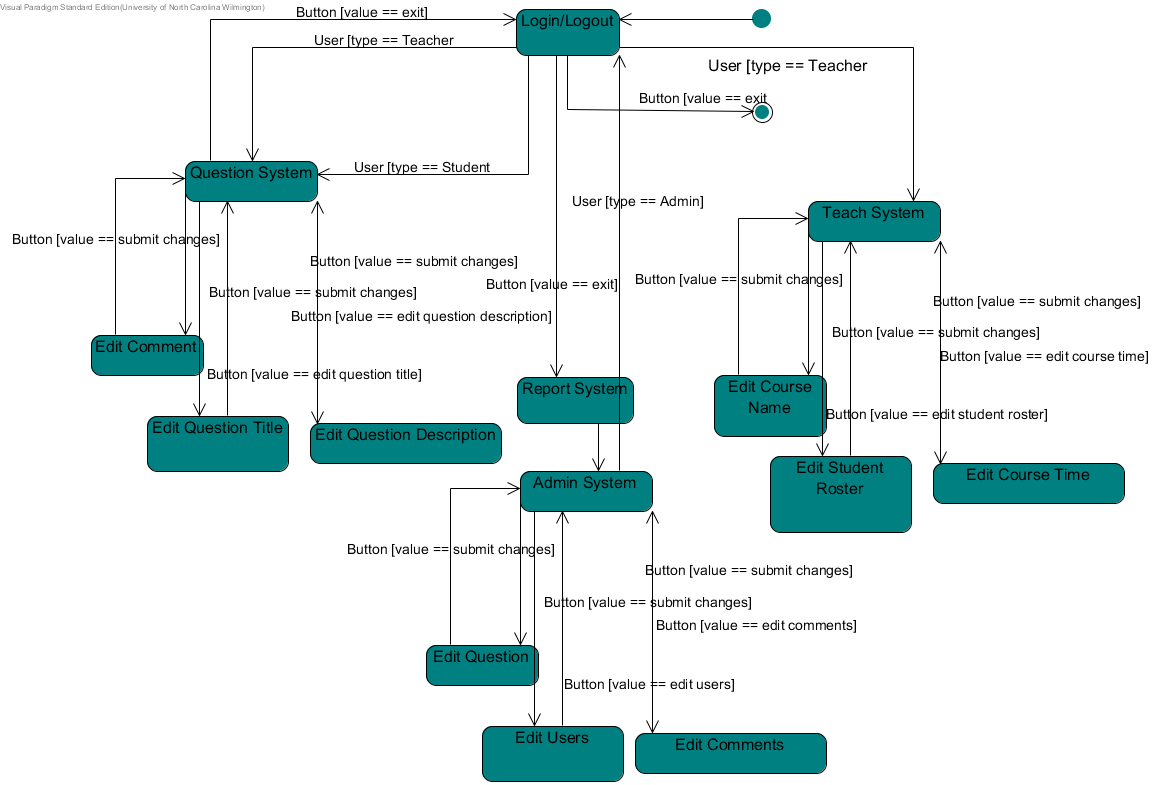
Figure

**Question Search Process Sequence Diagram**



Figure

# **State Diagram**



Figure

# **Requirements/Data Dictionary**

UP / DOWN VOTE

* + - Users may give a question an up vote or a down vote to indicate their approval/disapproval of its usefulness.
    - May need to include a per-user limitation to ensure fair and non-coerced voting.
    - Integer value stored as a property of the question.

TAG

* + - Makes it easier to find the right questions by attaching specific keywords that are relevant to the question. Will initially be generated by the parser on question submission.
    - String array stored as a property of the question.

USER

* Normal person using Hawk Overflow. Users include students, instructors, and admins.

STUDENT

* Can post, comment, and vote. A student is a registered student at the University of North Carolina at Wilmington. The student is either taking a computer science course with any of the professors in the computer science department.

INSTRUCTOR

* + - An instructor is a registered professor at the University of North Carolina at Wilmington.
    - Can post, comment, and vote.
    - Can also delete questions.

ADMIN

* Oversees all user activities. Created to be able to make changes to the website.
* Has all abilities of Instructor and Student, as well as some more site-specific controls.
* Can suspend/block non-admins.

REPUTATION

* + Giving active users more credibility when responding to questions.
  + Based on consistency of posting (new questions / comments) and overall vote status of said posts. 25% posting/75% vote-power split to ensure fairness, by limiting the amount of credibility generated by just posting.

# **Implementation**

**19.1 Languages Used:**

1) PHP: Main scripting languages for interacting with Databases

2) JavaScript: Alternate scripting languages for backend coding

3) HTML/CSS: Styling and organizing the elements on the website

4)SQL: For all database queries

**19.2 Software Functions Implemented:**

1) redirect\_user : A php method to move the a user to new link provided as a parameter

2) check\_login : Queries the database to see if the user credentials provided are valid

3) echo : Used to write elements to document object model in PHP

4) $\_POST : A method to post information from one page to another in PHP

5) Comment\_function: Handles events when a user enters a comment in Javascript

6) $\_SERVER : A function used to get meta info about the server in PHP

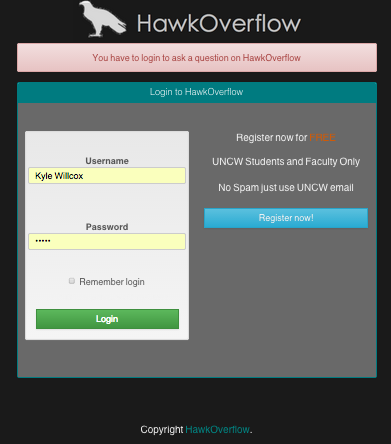
**19.3 Software Functions Not Implemented**

1) show\_image : Could have been implemented in PHP but time did not permit.

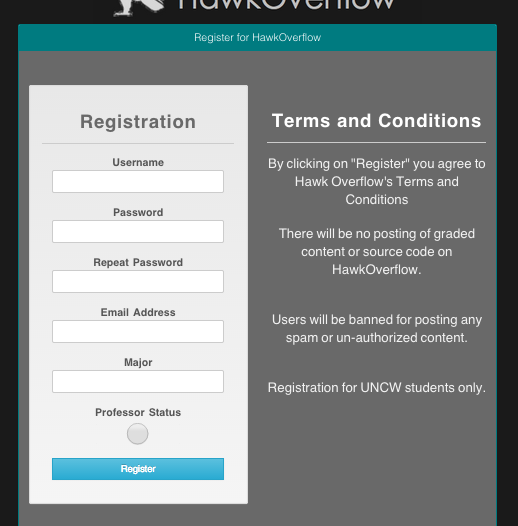
2) get\_user: Code to query the database for user information was used frequently. However no time was left to make a function for it.

**19.4 User Interface**

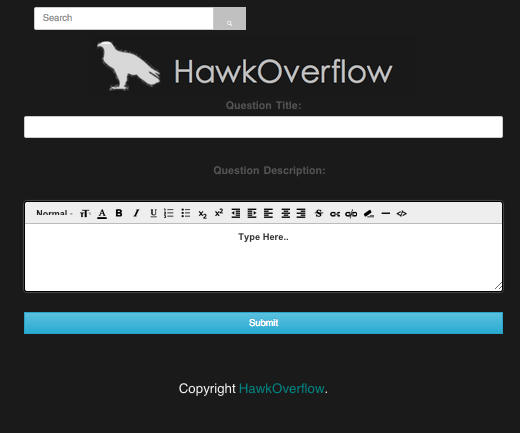
**Sign In Page**

****

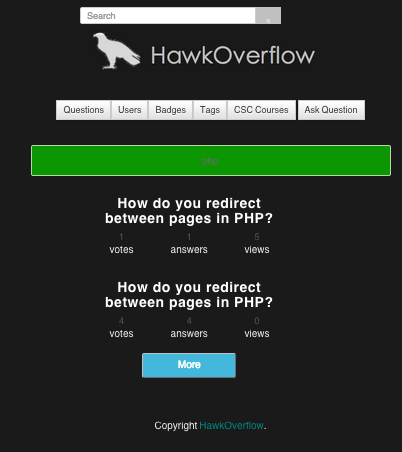
**Registration**

****

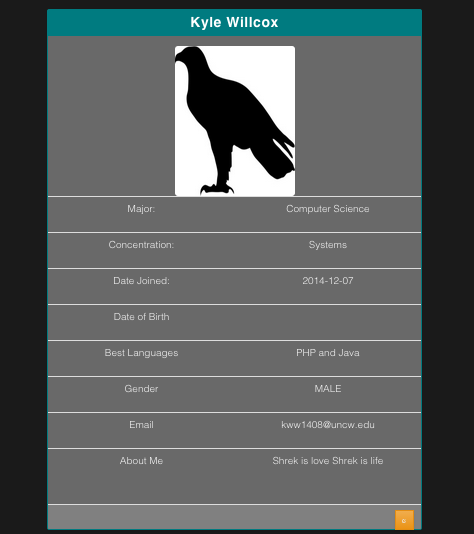
**Ask a Question**

****

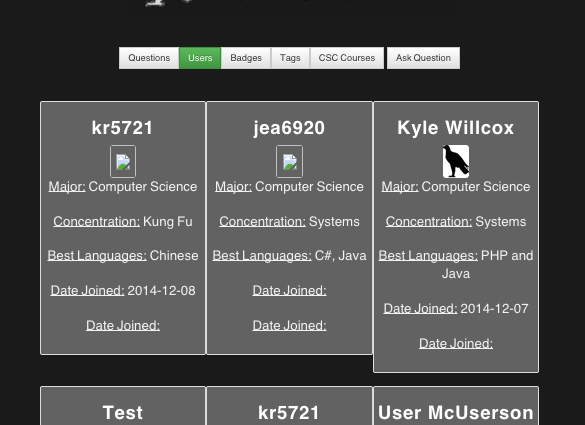
**Search for Questions**

****

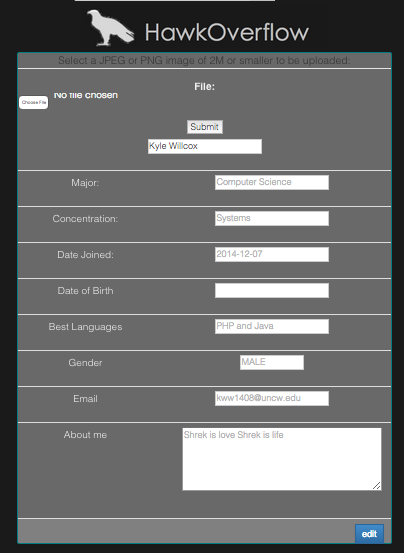
**Profile page**

****

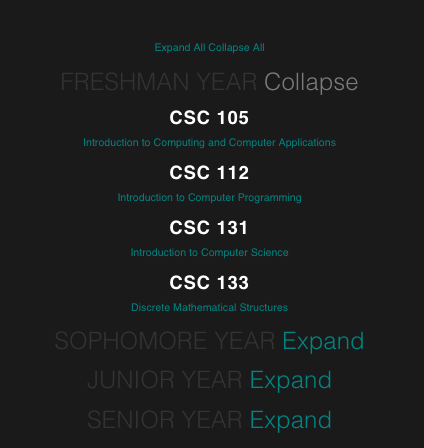
**All users**



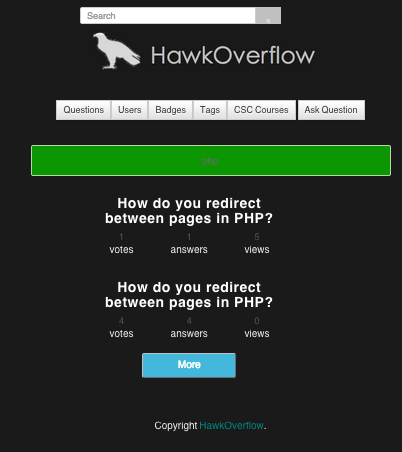
**Profile Editor**



**Courses page**

****

**Search Results**

****

# **PROJECT TESTING**

**General Testing Approach**

Our testing approach will begin as black-box, to ensure as smooth and reliable an experience as possible for our end-users, but will of course require a measure of post facto white-box editing should a sizeable bug be located [2]. The three primary test cases are usable under both jurisdictions, but were written with the black-box paradigm in mind [3].

**20.1 Current List of Test Cases:**

**Spam case:**

This is where we will stress test the search engine by sending in a query that is the same word or cluster of words repeated several times to ensure that the parser pulls out any repetitious terms before submitting the keyword query to the central database.

**Oven Mitt case:**

This will be where a query is submitted that is written by clumsily typing (as if one were wearing oven mitts), to ensure that the parser rejects non-valid search keywords/symbols. This is also a test case to ensure that inputs are sanitized against potential malicious code intrusions.

„ **Regex test:**

This will be similar to the Spam case, where we will submit a query using three equivalent terms (ex: CSC450, CSC 450, and CSC-450) to ensure that the parser can not only recognize all three terms as being valid, but acknowledge them as duplicates.

# **Non-functional Requirements**

**21.1 Ease of Use:**

We will stress test the website with a group of UNCW students first. We’ll get feedback from 10 students, then increase that size to 20 people. Once that is complete, we’ll open the website up to any student at UNCW.

**21.2 Security:**

We’ll devote time to find any holes in the website. By trying out ways to break the system, then make any changes to prevent future break-ins. Some of these methods will include URL redirection, SQL injection, DDoS attacks, and cross-site scripting [19].

**21.3 Performance/Efficiency:**

To ensure that the website runs efficiently and quick for users, the Bootstrap API will be used, shorter queries will be implemented (via the reduction parser), and PHP caching techniques will be embedded in the code [15] [16]. This will also go through some stress testing to see how well the website can hold up before becoming noticeably slow [17].

# **Bibliography**

* „ [1] "Tour." *Stack Overflow*. Stack Overflow, 1 Jan. 2014. n.p., n.d. Web. 20 Oct. 2014.
* „ [2] *Standard for Software Component Testing*. London: SIGIST, 2001. *Testing Standards - CO/UK*. BCS - SIGIST, 27 Apr. 2001. Web. 10 Oct. 2014. http://www.testingstandards.co.uk/Component%20Testing.pdf
* „ [3] Williams, Laurie. "White Box Testing." *White-Box Testing* (2006): n. pag. 01 Jan. 2006. Web. 30 Sept. 2014. http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf
* „ [4] "About The Licenses." *Creative Commons*. Creative Commons, 1 Jan. 2012. Web. 3 Oct. 2014. http://creativecommons.org/licenses/
* „ [5] Atwood, Jeff. *Stack Exchange*.

http://blog.stackoverflow.com/2010/11/dr-strangedupe-or-how-i-learned-to-stop-worrying-and-love-duplication/

* „ [6] Pressman, Roger S. *Software Engineering a Practitioner's Approach*. Boston, MA: R.S.man & Associates, 2005. Print.
* „ [7] Blaha, M., Rumbaugh, J. Blaha, Michael, and James Rumbaugh. *Object-oriented Modeling and Design with UML*. Upper Saddle River, NJ: Pearson Education, 2005. Print.
* „ [8] Pressman, Roger S. "R.S. Pressman & Associates, Inc." *R.S. Pressman & Associates, Inc.* R.S. Pressman & Associates, Inc., 1 Mar. 2011. Web. 5 Oct. 2014.
* [9] Digg.com – Frequently Asked Questions. Web.  
  http://digg.com/faq
* [10] Wikimedia foundation – Frequently Asked Questions. Web.  
  https://wikimediafoundation.org/wiki/FAQ/e[n](https://wikimediafoundation.org/wiki/FAQ/en)
* [11] UNCW Code of Student Life. Web.  
  http://uncw.edu/odos/documents/cosl-current.pdf
* [12] openID Foundation. Web.   
  http://openid.net/foundation/
* [13] USC – CSSE – COCOMO II Calculator. Web.  
  http://csse.usc.edu/tools/COCOMOII.php
* [14] Univ. of Michigan – Function Point Calculator. Web.  
  http://groups.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/artan/functionpoints.htm
* [15] Ancheta, Wern. wern-ancheta.com - Adventures in Web Development. Web.

http://wern-ancheta.com/blog/2014/01/26/getting-started-with-caching-in-php/

* [16] Bos, Wes. wesbos.com - Simple PHP Caching. Web.

http://wesbos.com/simple-php-page-caching-technique/

* [17] Stack Overflow. “How to stress test server memory…” Web.

http://stackoverflow.com/questions/10548400/how-to-stress-test-server-memory-and-processor

* [18] SourceTree - FAQ. Web.

http://www.sourcetreeapp.com/faq/

* [19] Cenzic/Trustwave. Vulnerability Report 2014. Web.

https://www2.trustwave.com/test-the-rest-vulnerability-report.html„