“Quick Feedback”

Group 5

Report 2 due: 11/8/15

Table Of Contents

[1. Introduction](#h.6jgpq1ywmsbr)

[1.1 Description of the Problem](#h.mf1cw0q10cxv)

[1.2 Goals and Objectives](#h.a6vdi8xjq0lw)

[1.3 Purpose](#h.rwagdp64nn8v)

[1.4 Scope](#h.cffa6myaceln)

[1.5 Definitions, Acronyms, and Abbreviations](#h.fw14cbp9vidq)

[1.6 Development Environment](#h.s6dlg4z1ckyl)

[1.6.1 Software](#h.fuirki1xdul2)

[1.6.2 Hardware](#h.jqamdoyd2hej)

[1.7 Operational Environment](#h.i68mqlilfgh8)

[1.7.1 Software](#h.nlhwndzhyrnw)

[1.7.2 Hardware](#h.z3rdxb86qnaa)

[2. Requirement Description](#h.vridnq3cybnw)

[2.1 Functional Requirements](#h.tx9pshycxux)

[2.1.1.a Functional Requirements for Users – Student](#h.s4foyohatond)

[2.1.1.b Functional Requirements for Users – Professor](#h.pnioo3h6epcf)

[2.1.2 Functional Requirements for Administrators](#h.n1znc641x99n)

[2.2 Non-Functional Requirement (Quality Attributes)](#h.habyyka9tf1o)

[2.2.1 Security Requirements](#h.6b7ohoew3kn7)

[2.2.2 Performance Requirements](#h.1unusqoda00p)

[2.2.3 Reliability Requirements](#h.flegpeuay77)

[2.2.4 Availability Requirements](#h.x62yq7mlxk3g)

[2.2.5 Efficiency Requirements](#h.93yv8qiaxlfz)

[2.2.6 Usability Requirements](#h.qhcfs0feid34)

[2.2.7 Maintainability Requirements](#h.55gv7u514gx0)

[2.2.8 Portability Requirements](#h.wcn5z9se7o0c)

[2.2.9 Testability Requirements](#h.yvslgahnt0rp)

[3. Management Process](#h.7ou1g9rmxtzd)

[3.1 Project Schedule](#h.gcl20w9zfmwq)

[3.1.1 Working Days Overview](#h.mwjjv2wogrkp)

[3.1.2 Tasks](#h.ixiq9xn1t0pd)

[3.2 Project Plan](#h.kmbklumpj9p)

[3.2.1 Phase Plan](#h.6zn4oxrd8zhm)

[3.2.2 Iteration Plan](#h.p1rtimy4m7lt)

[3.2.3 Releases](#h.et044k78jvr6)

[3.2.5 Project Resourcing](#h.v88x5kfee438)

[3.3 Project Monitoring and Control](#h.mfixk0z4zin3)

[3.3.1 Budget Control Plan](#h.3zkr8puyfkkv)

[3.3.2 Quality Control Plan](#h.ok9z8jh7nnai)

[3.3.3 Reporting Plan](#h.qd7j4svt6zf9)

[3.3.4 Measurement Plan](#h.g5cw1c76e91y)

[3.4 Risk Management Plan](#h.novqcpfhu0wf)

[4. Design Description](#h.4np3d5r6oo6d)

[4.1 Product Perspective](#h.yl3o9xggfa84)

[4.1.1 Product Vision](#h.5bz24lexih7a)

[4.2 Product Features](#h.1k7307aiv1ax)

[4.3 Use Cases](#h.8ci6hby7wduh)

[4.3.1 Use Case 1](#h.4qh5s1c6p1ku)

[4.3.2 Use Case 2](#h.uh3a7rjtudkw)

[4.3.3 Use Case 3](#h.oclqodmvjhwa)

[4.3.4 Use Case 4](#h.xelwjub1kfub)

[4.3.5 Use Case 5](#h.4tihrbrtsygi)

[4.3.6 Use Case 6](#h.iuqhkj3ah8y2)

[4.3.7 Use Case 7](#h.fc6n0eulnnjz)

[4.3.8 Use Case 8](#h.qv09r1yjs6lh)

[4.3.9 Use Case 9](#h.zca1r3xr1wbm)

[4.3.10 Use Case 10](#h.sxyvfzowg5nr)

[4.3.11 Use Case 11](#h.8hqf9dxh5lty)

[4.3.12 Use Case 12](#h.akt8qj9njoyw)

[4.3.13 Use Case 13](#h.3ywfzplwkb43)

[4.3.14 Use Case 14](#h.czj2ycvoha5)

[4.4 Sequence Diagrams](#h.qucibwfknu1i)

[4.5 Domain Model Diagrams](#h.u3qih1u5ao3j)

[4.6 System Class Diagrams](#h.w6p09gqqpkwk)

[4.7 Database Information (If Applicable)](#h.h4c343tcscc1)

[4.7.1 Database Tables (If Applicable)](#h.yjdyjea03mg3)

[4.8 Initial Sketch](#h.bdwlvgcjwtph)

[4.9 Interface Sketch](#h.f5e6umgcbzag)

[5. Test and Integration Plans and Results](#h.6m62ndshcvvy)

[5.1 Test Plans](#h.g3lq96w3c8ai)

[5.2 Test Cases](#h.g3lq96w3c8ai)

[5.3 Test Results](#h.g3lq96w3c8ai)

[6. Installation Instructions and User Documentation](#h.fia274is1ok3)

[6.1 Prerequisites](#h.kakacsvos8ef)

[6.2 Database Installation (If Applicable)](#h.a6sy38th6rlr)

[6.3 System Administration User (If Applicable)](#h.9ni3ff64ush5)

[6.4 User Manual, Operational Manual and Instruction](#h.p3kvvnj9l0p3)

[6.4.1 User Manual - Professor](#h.3tq9whti7a0)

[6.4.1 User Manual - Student](#h.mu5a75ggvz2h)

[6.4.3 Instruction to run](#h.mx762c8rnmue)

[6.5 Server Manual](#h.cxth4yfevxfh)

[7. Recommendations for the Enhancement](#h.vy7ay1c0l4qi)

[8. References and Bibliography](#h.lc0q4379s3vo)

Team Number: **5**

Project Manager: **Vinh Ha** vha3@csu.fullerton.edu

Team Members: **Francisco Rivas** frankie7413@gmail.com

**Sami Saleh** srafi1992@gmail.com

**Yuri Van Steenburg** yuuuri@csu.fullerton.edu

**Julia Nguyen** liddojulia@csu.fullerton.edu

**Michael Hatcher** lightball20@gmail.com

|  |  |
| --- | --- |
| **Member’s Background/Technical Strength** | |
| Francisco | Python/JavaScript/HTML/CSS/C#/C++/SQL |
| Sami | C++/C#/Python/HTML/SQL |
| Yuri | C++/Java/HTML/CSS/SQL |
| Vinh | C++/Java/Bash/Python |
| Julia | C++/Python |
| Michael | C++/Python/C#/HTML/CSS |

# 1. Introduction

## 1.1 Description of the Problem

A student’s ultimate goal in college is to be able to comprehend, analyze, and think critically of the subjects that are taught in class; furthermore, a professor’s ultimate goal is to impart knowledge, understanding of the material, and invoke the student’s engagement with the material. However, some students feel intimidated or timid when contacting a professor and would rather not ask questions or voice their comments and concerns, which could be beneficial to both parties. Consequently, the professor would not be able to notice if a student is struggling to understand the material, not getting enough information in lectures, gauge the class understanding, or etc.

## 1.2 Goals and Objectives

The goals are as follows:

1. Allows students to anonymously send a professor a question, comment, or concern.
2. Allow professors to create an account and manage it.
3. Allow professors to respond and send an email to the entire class.
4. Allow professors to respond to an individual message with student approval.
5. Allow professor to select classes that he/she is teaching.
6. Archive questions to refer at a later date.
7. Prevent students from misusing quick feedback.

## 1.3 Purpose

The primary purpose of this website is to provide students the ability to anonymously send messages, questions, comments, or concerns to their professors. The website shall provide instructors with an account interface that will allow them to reply to feedback via broadcasting email to a class or reply to the specific student that sent the feedback while keeping his/her privacy. Furthermore, the website shall provide an archive for the professor to review past feedback.

## 1.4 Scope

To build a system where students can submit anonymous feedback messages to a particular professor according to the course the student is currently taking. The system will be usable on mobile and desktop internet browsers. The system is meant to be professional where a student can anonymously message his concerns, praises or questions to the professor. Furthermore, the professor would be able to reply to individual who send him the original message or select to message original message and his reply to the current students enrolled in the course. Additionally, the professor would have the options on every new semester to signed up to new courses he/she is teaching, view messages from past semesters, and be able to erase messages. The goal of the system is to give timid students an opportunity to conceal their identity to have the courage to ask questions, give praise, or state concerns that they would normally not be willing to communicate in person. Therefore, professors would be granted the opportunity to read feedback that students would refrain from writing until student evaluations in the end of the year. Since a professor may teach several classes per semester, the database shall be able to maintain all incoming messages and store messages as the school year continues. In addition, an administrative user would monitor all incoming messages to remove any unwanted spam, add courses to database, manage professor accounts and take any support tickets from students and professors.

## 1.5 Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Word** | **Definition** |
| App | a self-contained program or piece of software designed to fulfill a particular purpose |

## 1.6 Development Environment

### 1.6.1 Software

|  |  |
| --- | --- |
| **Classification** | **Software Requirements** |
| Operating System | Windows 7, 8.1, 10 |
| Operating System | Ubuntu 14.04 LTS, Ubuntu 15.04 |
| Operating System | OS X 10.10 |
| Back End Framework | Node.js Framework, Express.js Web Framework |
| Front End Framework | Bootstrap |
| Database Management System | MongoDB |
| Text Editor | Sublime Text 3, Atom |
| Version Control | Git |
| Remote Repository | GitHub |

### 1.6.2 Hardware

|  |  |
| --- | --- |
| **Classification** | **Hardware Requirements** |
| Processor | Intel Pentium |
| Memory | 4GB RAM |
| Storage | 20GB SSD |

## 1.7 Operational Environment

### 1.7.1 Software

|  |  |
| --- | --- |
| **Software** | **Classification** |
| Windows 8.1 64-bit | Operating System |
| OS X | Operating System |
| Linux - Debian, Ubuntu, etc | Operating System |
| iOS | Operating System |
| Android | Operating System |
| HTML5 | Markup Language |
| Firefox, Chrome | Web Browser |

### 1.7.2 Hardware

|  |
| --- |
| **Hardware Requirements** |
| 1 gigahertz (GHz) or faster processor 32 bit or 64 bit processor |
| 4GB RAM |
| 8GB available hard disk space |
| DirectX 9 graphics device with WDDM 1.0 or higher driver |

# 

# 

# 

# 2. Requirement Description

## 2.1 Functional Requirements

Two types of users are defined: Student and Professor

#### **2.1.1.a Functional Requirements for Users – Student**

|  |  |
| --- | --- |
| **User Requirements: Interface Requirements** | |
| IRS01 | User shall be able to select a course |
| IRS02 | User shall be able to submit a feedback |

#### 

#### **2.1.1.b Functional Requirements for Users – Professor**

|  |  |
| --- | --- |
| **User Requirements: Interface Requirements** | |
| IRP01 | User shall be able to create an account |
| IRP02 | User shall be able to log in |
| IRP03 | User shall be able to select current course(s) |
| IRP04 | User shall be able to view a message |
| IRP05 | User shall be able to view his/her archive |
| IRP06 | User shall be able to reply to message |
| IRP07 | User shall be able to reply to the class |
| IRP08 | User shall be able to delete a message |
| IRP09 | User shall be able to forward to his/her personal email account |
| IRP10 | User shall be able to upload student email list |

### 2.1.2 Functional Requirements for Administrators

|  |  |
| --- | --- |
| **Administrator Requirements** | |
| AR01 | User shall be able to log in |
| AR02 | User shall be able to verify professor email |
| AR03 | User shall be able to update an account |
| AR04 | User shall be able to add a course for the professor |
| AR05 | User shall be able to delete student message |

## 2.2 Non-Functional Requirement (Quality Attributes)

### 2.2.1 Security Requirements

|  |  |
| --- | --- |
| **Security Requirements** | |
| SR01 | The web application shall prevent students from misusing the anonymous feedback by preventing students from spamming a professor |
| SR02 | The system must have means to avoid, detect, and recover from attacks and unauthorized access |

### 2.2.2 Performance Requirements

|  |  |
| --- | --- |
| **Performance Requirements** | |
| PER01 | The page call to the server shall not take longer than 30 seconds to resolve |
| PER02 | The page shall respond within 2 seconds after each user action |

### 2.2.3 Reliability Requirements

|  |  |
| --- | --- |
| **Reliability Requirements** | |
| RR01 | The web app shall ensure that if a question is submitted that an email shall be sent to the correct professor’s email address |
| RR02 | The web app shall ensure that all questions are archived in the right folders  · The professors shall be able to move them as they see fit |

### 2.2.4 Availability Requirements

|  |  |
| --- | --- |
| **Availability Requirements** | |
| AVR01 | The app shall be available to the students and professors 24/7 when school is in session |

### 

### 2.2.5 Efficiency Requirements

|  |  |
| --- | --- |
| **Efficiency Requirements** | |
| ER01 | The RAM usage shall take up no more than 25%. |

### 2.2.6 Usability Requirements

|  |  |
| --- | --- |
| **Usability Requirements** | |
| UR01 | The web app shall be user friendly |
| UR02 | Professor’s names and courses shall be listed in a drop down menu |
| UR03 | A textbox and submission button shall be available for students to type and submit their questions |
| UR04 | Professors shall be able to navigate through an archive of questions separated into different folders on a separate page |
| UR05 | With ease of use, it will allow students and professor to efficiently use the web application, which will improve students’ needs and professor’s teaching performance. Also, students ask the professors any questions easily, which will encourage them to come back to use this service |

### 2.2.7 Maintainability Requirements

|  |  |
| --- | --- |
| **Maintainability Requirements** | |
| MR01 | The web app shall be compatible with new versions of a browser and it’s older versions |

### 2.2.8 Portability Requirements

|  |  |
| --- | --- |
| **Portability Requirements** | |
| PR01 | The web app shall be formatted for various types of devices (mobile phones, tablets, laptops, and desktop computers |
| PR02 | The web app shall be available on: Internet Explorer, Safari, Google Chrome, and Firefox |

### 2.2.9 Testability Requirements

|  |  |
| --- | --- |
| **Testability Requirements** | |
| TR01 | TBD |

# 

# 

# 

# 3. Management Process

## 3.1 Project Schedule

### 3.1.1 Working Days Overview

|  |
| --- |
| **Date: September 3, 2015** |
| Group members: Francisco, Julia, Sami, Vinh, Yuri  Topics:   1. Description of problem 2. Objective and goals 3. Functional Requirements 4. Nonfunctional Requirements   Outcome: The team came up with the topic for our project as well as several requirements needed to start our project. We also discussed the basic structure of what our system will provide for users. |
| **Date: September 6, 2015** |
| Group members: Francisco, Julia, Yuri  Topics:   1. Web framework   Outcome: Decided to go with Django version 1.8. Django tutorial update in the works. |
| **Date: September 7, 2015** |
| Group members: Francisco, Julia, Michael, Sami, Vinh, Yuri  Topics:   1. Clarification of roles 2. Project name ideas 3. Begin creating use cases 4. Clarification of project goals and features   Outcome: Some ideas for the project name were, Anon asks, Prof. Com., Profback, Prof Connect, Noname. A name still has not been decided. The testers are Yuri and Michael, programmers are Sami and Michael, program lead is Francisco, design lead is Julia, and project manager is Vinh. We created a basic list for use cases to get started. Discussed briefly anti-spam feature that will be needed. Anonymity and simplicity. Planned to create sketches of the student’s perspective of the interface to share next meeting. |
| **Date: September 9, 2015** |
| Group members: Francisco, Julia, Michael, Sami, Vinh, and Yuri  Topics:   1. Initial web design 2. Use cases   Outcome: Updated the list of use cases required for our web application and use case format chosen. Divided the use case responsibilities accordingly to each member. First draft of each use case was planned to be completed by the scheduled meeting date of September 12, 2015 at 8:00pm. Vinh’s sketch of the message/feedback form design was chosen and will be implemented with some minor modifications. |
| **Date: September 12, 2015** |
| Group members: Francisco, Julia, Sami, Vinh, Yuri  Topics:   1. Review use cases 2. Web layout   Outcome: Message/feedback form is placed on the homepage. An early preview of the website is shared by Francisco. All use cases are reviewed, modified, and formatted. |
| **Date: September 14, 2015** |
| Group members: Francisco, Julia, Michael, Sami, Vinh, Yuri  Topics:   1. Ask professor questions   Outcome: An early rough draft of the use case diagram is drawn out by Sami. For now the title was chosen to be “Quick Feedback”. Plan to continue editing use cases after clarifying a few things with the professor. Plan to add “Admin login” use case. Plan to fix alternative flows and exceptions for the use cases. For the use cases, instead of using “administrator”, “student”, or “professor”, we will replace them with “user”. Will edit descriptions of use cases to fit the use case goals. Plan to scan sketch of the feedback form. The professor provided us with a way to look up other professor's email addresses for our whitelist. She also would prefer a 2-step authentication for creating an account. Meeting scheduled for 9/17/15 at 9:00pm or 9/20/15 at 6:00pm. |
| **Date: September 20, 2015** |
| Group members: Francisco, Michael, Sami, Vinh, Yuri  Topics:   1. Part 3 Management Process   Outcome: Assigned sections of part 3 Management Process accordingly to each member. The assigned parts are to be completed by 9/23/15 class meeting. |
| **Date: September 21, 2015** |
| Group members: Francisco, Julia, Michael, Sami, Vinh, Yuri  Topics:   1. Framework switch 2. Review use cases 3. Part 3 Management Process   Outcome: We will now be using Node.js framework for the backend instead of Django. A few use cases were clarified and updated. An updated use case document titled “Use Case v2” is created. We reviewed each section of part 3 and will complete the assigned sections by next class meeting. |
| **Date: September 23, 2015** |
| Group members: Francisco, Julia, Michael, Sami, Vinh, Yuri  Topics:   1. Database 2. Risk management   Outcome: We decided to use MongoDB for our database and started designing our database schema. Our risks management section is edited after clarifying a few issues with the professor. An updated demonstration of the website was previewed in class. Key architectural risks analyzed, documented, and resolved. Final Use Case Diagram scheduled to be completed. |
| **Date: September 30, 2015** |
| Group members: Francisco, Julia, Michael, Sami, Vinh, Yuri  Topics:   1. Use case justifications 2. Functional requirements 3. Update documentation   Outcome: Sami and Julia will fill in the justifications for the use cases that we have selected for being the most important requirements for the system. Our functional requirements require some modifying and will worked on. Meeting scheduled for Friday at 8:00 pm, to discuss final thoughts and plans before submitting Report 1. Will be reviewing report one and making final edits/changes before the due date of 10/4/2015 Sunday night. |
| **Date: October 02, 2015** |
| Group members: Francisco, Julia, Sami, Vinh  Topics:   1. Use case preconditions and postconditions 2. Update documentation   Outcome: Fixed and updated the use case preconditions and postconditions to show the states of the system. The goals of the use cases are updated as well. The overall documentation is reviewed and edited after group conversations. |
| **Date: October 07, 2015** |
| Group members: Francisco, Julia, Sami, Yuri (with Prof. Gofman)  Topics:   1. Emailing whole class functionality 2. Getting feedbacks from Prof. Gofman   Outcome: We showed Prof. Gofman test version of our web app running on Frank’s local drive. Prof. Gofman tested few dropdown menu for Prof. page, worked fine, and liked the UI. Functionality of emailing a whole class is decided to be in scope of our app now. |
| **Date: October 12, 2015** |
| Group members: Francisco, Julia, Michael, Sami, Vinh, Yuri  Topics:   1. Domain Model 2. System Sequence Diagram 3. New use case   Outcome: The domain model diagram is updated, checked by the professor and was approved. We made a quick sketch of a System Sequence Diagram for an interaction between student and the system. Added a new use case for the professor/user which allows the professor to send a reply email to all the students. Minor functional requirements update. System Sequence Diagrams are assigned accordingly to each member and is to be done by next meeting. |
| **Date: October 14, 2015** |
| Group members: Francisco, Julia, Sami, Vinh, Yuri  Topics:   1. System Sequence Diagram 2. Report 1   Outcome: Most of the System Sequence Diagrams are completed and all are merged into one file. For Report 2, we plan to fix the numbering of the flows in our use cases and fix our use case diagram. We also plan to fix our project scope. The professor recommends that the project scope should be much longer in length compared to our current project scope. |
| **Date: October 21, 2015** |
| Group members: Francisco, Julia, Michael, Sami, Vinh, Yuri  Topics:   1. Domain Model   Outcome: Mainly revised assignment 2 using feedback received from professor. Redesign of the Domain model version 2 after meeting. |
| **Date: November 2, 2015** |
| Group members: Francisco, Julia, Sami, Vinh, Yuri  Topics:   1. Interaction Sequence Diagram 2. Todo list   Outcome: Began Interaction Sequence Diagram for student, professor, and administrator. Product perspective and product features assigned. A short todo list was created for items needed for Report 2 before submission. |
| **Date: November 4, 2015** |
| Group members: Francisco, Julia, Sami, Vinh, Yuri  Topics:   1. Use Case 14 2. Revise and update Report 2   Outcome: Report 2 is updated and corrected using the professor’s feedback from Report 1. Decided to remove Use Case 14 “Clear student emails”. Added new diagram models and sketch into Report 2. Next meeting scheduled for 11/6/15 at 7:00 pm. |
| **Date: November 6, 2015** |
| Group members: Francisco, Julia, Sami, Vinh, Yuri  Topics:   1. Use Cases   Outcome: Reordered, revised, and updated use cases. This also required us to update our SSDs, requirements, and Use Case Diagram. Product perspective and features is completed by Sami and added to the report. Next meeting scheduled for 11/07/15 at 7:00 pm. |
| **Date: November 7, 2015** |
| Group members: Francisco, Julia, Michael, Sami, Vinh, Yuri  Topics:   1. Use Case 13 2. Finalizing Report 2   Outcome: Finished up last Use Case, updated Use Case Diagram, and updated SSD. Updated product perspective and added future outlook for our web application. Updated scope and Use Case justifications. |

### 3.1.2 Tasks

|  |
| --- |
| **Project Manager – Vinh** |
| The project manager will organize, manage, and guide the project to meet the project requirements. The project manager will be in charge of scheduling meetings and recording important information and topics that are brought up during these meetings. The phase plans and iteration planning is the job that the project manager has to undertake and it is also important for the project manager to make sure all the participating team members understand the goals of each iteration. The project manager is also in charge of organizing the team and managing expectations in order to achieve all the objectives of the project. It is also the responsibility of the project manager to communicate the project status in order to stay on schedule. |
| **Design Lead – Julia** |
| The design lead is responsible for the written description of the software. The written document will then be given to the software development team to guide the team to the overall architecture of the software product. The documentation will help with the integration of the software in object-oriented design. The design lead is responsible for creating a data design/structure, architecture design, interface design, and procedural design. |
| **Programming Lead – Frank** |
| The programming lead shall be in charge of implementing the overall design of the project during the construction phase. Furthermore, the programing lead shall overlook the structure of the code through making sure the code is manageable and properly implemented. The programming lead shall be responsible of implementing backend technologies such as database and network connectivity. The programming lead shall overlook that the front-end technologies are implemented to communicate with the backend technologies. In the construction and transition phase, the programing lead shall communicate the code progress to the lead tester and tester in order to prepare for future test case runs. |
| **Programming – Sami** |
| The programmer shall assist the lead programmer in implementation throughout the progress of the project. It is the duty of the programmer to ensure the front end meets customer requirements using best-fit technology. In addition to working alongside the program lead, the programmer shall discuss with the design lead in ensuring the project is within scope of the project. This programmer shall ensure all requirements of the front end is met while coordinating with the program lead in communicating with the backend. |
| **Lead Tester – Yuri** |
| The Lead Tester (LT) shall perform the manual function testing on the designated devices at the earliest opportunity for complete functionality for precise, rigorous system checks prior to the system being seen as having the ability to go live with the client. Responsibilities of the LT will be communicating all test issues and concerns effectively and then executing the correct procedures in order to ensure the appropriate fix. The LT will coordinate the daily workload of each Tester by giving direction on projects to facilitate the results being completed in a timely fashion. The test results will be analyzed, discussed, and concerns resolved with the project manager and the design lead to improve correctness and performance for project web apps in order to meet customer's performance metrics. |
| **Tester – Michael** |
| The Tester shall design and executive test plans with the Test Lead in order to allow the pursuit of high levels of quality in the functionality of the product. The Tester shall be in charge of performing and recording various test cases and results. The tester shall report any unexpected results that came to pass during tests to the Test Lead. The tester shall carry out all possible tests to find any potential bug(s). The Tester shall be responsible for documentation of all test results and then to compare with the expected results. Any and all anomalies and issues will be discussed with the Test Lead. |

## 3.2 Project Plan

The project shall follow the Unified Process and each iteration shall be 11 days long. There are 8 iterations planned until the release date.

### 3.2.1 Phase Plan

|  |  |  |
| --- | --- | --- |
| **Phase** | **Starts on** | **Ends on** |
| Inception Phase | September 1 | September 12 |
| Elaboration Phase | September 13 | October 4 |
| Construction Phase | October 5 | November 7 |
| Transition Phase | November 8 | November 28 |

### 3.2.2 Iteration Plan

|  |  |  |
| --- | --- | --- |
| **Iteration # and Objective** | **Starts on** | **Ends on** |
| Inception Phase - Iteration 1 | September 1 | September 12 |
| · Initiate project  · Project approval  · Identify team organization  · Develop product vision  · Define the scope of the project  · Define project requirements  · Prepare project environment  · Begin documentation  · Set up communication links between team members  · Architecture decision | | |
| Elaboration Phase – Iteration 2 | September 13 | September 24 |
| · Initial design of web application  · High-level risk features implemented  · Sketch system  · Define system actors  · Use cases  · Identify and resolve high-level risks  · Address development issues  · Database design and database schema  · First UML Use Case Diagram | | |
| Elaboration/Construction Phase – Iteration 3 | September 25 | October 6 |
| · User feedback and adaptation  · Identify and resolve low-level risks  · UML Use Case Diagram  · Architecture baseline defined  · Demonstration  · Finalize Report 1 | | |
| Construction Phase – Iteration 4 | October 7 | October 18 |
| · Code and check application  · User feedback and adaptation  · Refine Use Cases and Use Case Diagram  · Update requirements  · Update scope  · Domain Model Diagram  · System Sequence Diagrams | | |
| Construction Phase – Iteration 5 | October 19 | October 30 |
| · Code and check application  · Update requirements  · Sequence Interaction Diagram  · Domain Model Diagram revision | | |
| Construction/Transition Phase – Iteration 6 | October 31 | November 11 |
| · Code and check application  · Testing Frontend and Backend  · Use Case revision  · Product perspective  · Product features  · Finalize Report 2 | | |
| Transition Phase – Iteration 7 | November 12 | November 23 |
| · | | |
| Transition Phase – Iteration 8 | November 24 | November 28 |
| · Release | | |

### 3.2.3 Releases

November 29, 2015

### 3.2.5 Project Resourcing

Human resources are noted in 3.1.2 Task section.

The tools used are as follow:

* Google Drive: Documents, Spreadsheet
* Laptop/Desktop running Window 8/Linux Ubuntu/OSX
* Node.JS
* Sublime Text 3
* Git/GitHub
* MongoDB

Time Plan:

|  |  |
| --- | --- |
| **Role** | **Hours worked** |
| Project Manager | 12 weeks, average 8 hours per week |
| Design Lead | 12 weeks, average 8 hours per week |
| Program Lead | 12 weeks, average 10 hours per week |
| Program Assist | 12 weeks, average 10 hours per week |
| Test Lead | 12 weeks, average 8 hours per week |
| Tester | 12 weeks, average 8 hours per week |

## 

## 

## 3.3 Project Monitoring and Control

### 3.3.1 Budget Control Plan

|  |  |  |
| --- | --- | --- |
| **Cost Type** | **Cost** | **Total** |
| **Phase I** |  |  |
| **Labor** | **$1,080** | **1,080** |
| **Material** | **$600** | **1,680** |
| **Phase 2** |  |  |
| **Labor** | **$1,080** | **1,080** |
| **Other** | **$100** | **$1,180** |
| **Sub Total** |  | **$2,860** |

### 3.3.2 Quality Control Plan

* The project manager shall update the scheduling the project and ensure that the project schedule is followed and updated on a weekly basis.
* The product manager shall manage the project budget to ensure that the project does not go over budget.
* The product manager shall manage the risk daily and update them when necessary.
* Quality control will occur between each iteration where the requirements will be checked and compared with the part of the project the group is currently working on to ensure that all the requirements listed were included in the project.

### 3.3.3 Reporting Plan

The project manager shall collaborate with the development team and coordinate the meetings with the client at each iteration basis. The development team members shall attend the meetings with the client to gauge their satisfaction of the software in the current developmental stage and gather the client’s feedback for possible implementation in future iterations. The details of each meeting shall be recorded by the project manager or a specific team member who will be designated by the project manager. At the beginning of each iteration, the development team meetings will be held to report each member’s status and make sure that every team member understands the objective of the current iteration. The project manager will assess the team’s progress and adjust workloads accordingly.

### 3.3.4 Measurement Plan

Each team member shall seek feedback data in based on their designated area of their current task in the developmental process, the team’s progress will be assessed during the team meeting once per iteration. Thus all team members will be able to determine if the development process is going along as planned or if the team is behind schedule. Any concerns and discussions about improvements that can be made to the software will be addressed during the meeting in order to help each individual team member work more efficiently during the next iteration. If the team’s progress is determined to be behind schedule, the team member’s task and responsibility will need to be re-evaluated. At that point-in-time the project manager will re-assign tasks and responsibilities to individual team members with the team’s approval.

## 3.4 Risk Management Plan

|  |  |  |
| --- | --- | --- |
| **What is the risk?** | **Solution** | **Priority** |
| **Time Constraint** | Lay out deadlines for each iteration | Medium |
| **Scope of the project** | Pre-defining what extent of the project | High |
| **Choosing the right development tools** | Experimenting each tool to see what tools work best with | High |
| **Skills on web development** | Interview potential members for skills in web development | Medium |

**Time Constraint**

Because of the amount given, timing is crucial within this project to deliver the final product without delay. To prevent any possibility of losing track from the deadline, a plan that lays out all the objectives on each iteration would be essential to handling any unknown complications in the future. By doing so, while additionally building upon the system, can reduce the risk on timing constraint.

**Scope of the Project**

Every project has its own domain with its extent at how much a project can function. This “scope of the project”, is a high of risk because simply because of one question: does it fit our objective? In order to keep the scope under control, necessary steps in knowing what the goal is or whether this function is within scope can allow the project to be deliver in a timely basis, keep the flow of the project into the right direction.

**Choosing the right development tools**

Originally, Flask and Django was what the team has chosen as the framework to build our project upon. However, because of our lack of knowledge within the framework and the amount of time necessary to learn the frameworks, they were scrapped. This risk comes right off from the start of the project due to knowing which frameworks each team member has knowledge upon.

**Skills of web development**

The project needed the web skills from the start just to begin with. The risk is moderate that could potentially become a high risk. The reason for this is because of simply know which members has the necessary coding skills to build the web application itself. To counter this, interviewing potential members and describing their strong skills can quickly resolve any issues in coding.

# 4. Design Description

## 4.1 Product Perspective

The future outlook of the Quick Feedback web app is building features that will expand the usage of the app to accommodate our client as well as allowing its application to potential clients. The features planned for future updates include:

* Linking the web app to social media (such as Facebook, Twitter, and LinkedIn). The app will use social media as another means in which the professor can reply to questions asked by a professor. For example, a professor would be able to tweet out a reply to a question.
* Creating a class calendar - Professors can post the due date of their assignments this helps students ask relevant questions to the assignment before the due date. In addition, professors may also set up an email reminder that will send students an email the day before the assignment is due.
* Send emails to all courses, sections and students from previous semesters - Professors can send emails to each student they are currently teaching or have taught to inform them about the school’s current events or beneficial opportunities. This feature allows a professor to stay in touch with their students and provide students with opportunities that may help them in the future.
* Email attachments - Professors can attach files to their emails. Email attachments can be referring to various scholarly articles related to a topic to provide students with more information if they are interested in reading about it.
* Profanity checker - Messages sent by the students will be checked and filtered for profanity to create a good academic environment for both the students and professors.
* Creating a mobile app - Students can download a mobile app. Students can ask anonymous questions from their smartphones and tablets.
* Creating a forum of the previous asked questions for each professor - Students can look through a forum of previous questions asked for the same course. Therefore the students can find answers faster.

### 4.1.1 Product Vision

“Quick Feedback” is a web-based service that provides honest communication that improves the relationship between professors and their students in the university type of setting. Unlike regular email communication or other types of student evaluation channels provided by schools, our web-application presents a communication link for students freely and anonymously.

## 4.2 Product Features

The following highlights the entirety of features Quick Feedback entails:

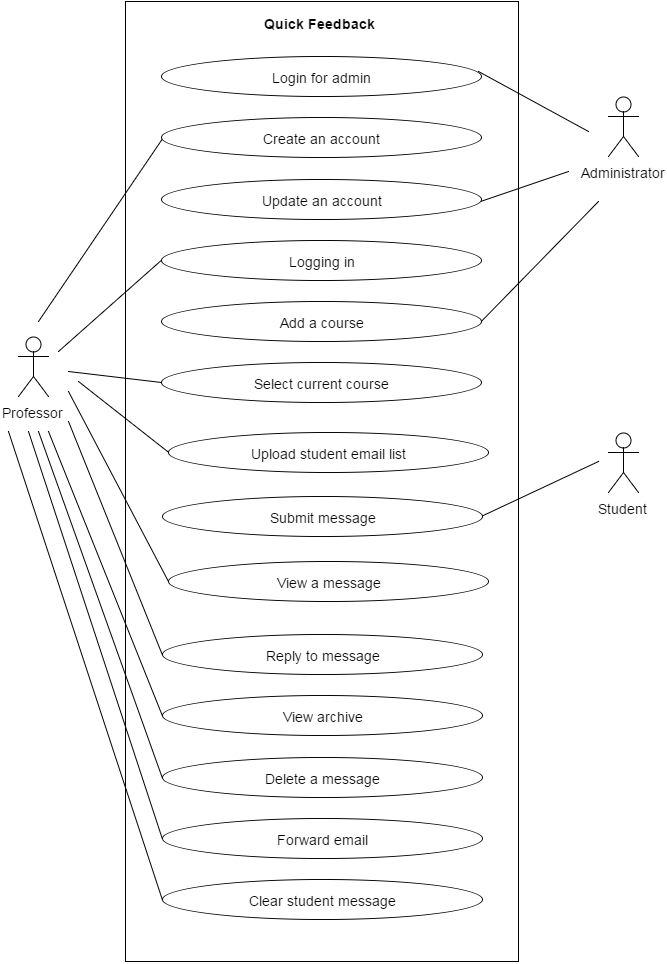
* User Account - Through secure login, Professors able to view their account information, view new messages of students feedback, view past messages, archive past courses (along with their respective message). These accounts are tied solely through their university account, meaning only professor’s with emails are able to sign up.
* Feedback Submission - Students are given a form within the web application to navigate to the course they wish to submit their feedback upon. In addition to anonymous submission, students have the options for individual response back from the professor by entering their email. These email are strictly confidential and are kept hidden.
* Individual reply - When Professors reads a student’s message, they decide on whether to respond to the individual message. However, this only occurs when a flag is provided signify a student email was given, which in turn can give the option for an individual response.
* Mass classroom email response - Professor are able to send a mass response to their courses. By uploading a .csv formatted file listing all of their student’s email account for each course, the professor simply selects the mass email response and submit the required fields.

## 

## 

## 

## 4.3 Use Cases



### 

### 4.3.1 Use Case 1

|  |  |
| --- | --- |
| Use Case ID | 01 |
| Use Case Name | Login for admin |
| Primary Actor(s) | Administrator (admin) |
| Goal | User successfully logins to their account. |
| Precondition | None |
| Post Condition | session instance *amsg* was created  *amsg* was assigned to unique ID |
| Main Flow | 1. User selects “Administrator Login” 2. System forwards user to the administrator login web page 3. User types in their email in the “email” textbox 4. User types in their password in the “password” textbox 5. User clicks the “Login” button 6. System successfully validates email/password combination 7. System forwards the user to their account page |
| Alternatives Flows | 5.a. System alerts user that email/password combination is incorrect  5.a.1. System forwards user back to Main Flow #2    5.b. System alerts user that the account does not exist in the database  5.b.1. System forwards user back to Main Flow #2 |
| Exceptions |  |
| Notes | Even though the project is based mainly around professor and student, it is important to allow administration access to the system because of it satisfaction of AR01 functional requirement. A specific user is necessary to maintain the web application itself. |

### 4.3.2 Use Case 2

|  |  |
| --- | --- |
| Use Case ID | 02 |
| Use Case Name | Create an account |
| Primary Actor(s) | Professor |
| Goal | User successfully creates an account. |
| Precondition | None |
| Post Condition | 1. Professor account instance *profAcct* was created 2. *profAcc*t was assigned to unique ID |
| Main Flow | 1. User selects “Sign up” button to sign up for an account on “Home” web page 2. System displays a blank sign up form 3. User selects their affiliated “College” 4. User inputs their first name in the “First Name” text box 5. User inputs their last name in the “Last Name” text box 6. User inputs a valid professor “.edu” email address 7. User provides a password that must be 8 characters long, that includes 1 digit and 1 letter 8. User re-type password 9. User clicks the “Submit” button 10. System verifies input information by sending a verification email containing a secret code to the user 11. User receives the email in their email inbox 12. User will use the secret code within the email and input into the web page 13. System verifies the code 14. System creates an account saving the user’s information along with a unique ID into the database |
| Alternatives Flows | 9.a. System displays an error message: “Account already exists.”  9.a.1. System navigates the user to login page (UC:02)  9.b. System displays an error message: “Email is not in .edu format.”  9.b.1. System navigates user back to Main Flow #5  9.c. System prompts “Password must be 8 characters long contains at least 1 letter and 1 digit”  9.c.1. Return to Main Flow #7 |
| Exceptions |  |
| Notes | The importance of being able to create an account satisfies the functional requirement IRP01 “Professors shall be able to sign up for an account”. This requirement is important in allowing only the professor to see the student’s responses while preventing other students from viewing them (Use case ID: 06). The professor can also archive the messages (Req. ID: IRP04), forward it to an email account (Req. ID: IRP07), delete messages (Req. ID: IRP06), and customize the web page to his or her liking such as adding a class and deleting a class (Req. ID: IRP08) |

### 4.3.3 Use Case 3

|  |  |
| --- | --- |
| Use Case ID | 03 |
| Use Case Name | Update an account |
| Primary Actor(s) | Administrator |
| Goal | User successfully updates the professor’s account. |
| Precondition | UC:01  UC:02 |
| Post Condition | Professor instance *profAcct* *was* updated |
| Main Flow | 1. User clicks on “Users” on their administrator page 2. System renders a list of professor accounts 3. User selects the desired account 4. User modifies the professor’s account information 5. User selects “Update” 6. System updates the account information in the database |
| Alternatives Flows | 5.a. User selects “Delete”  5.a.1. System prompts message: “Are you sure? This account will permanently be deleted?”  5.a.2. User selects “Yes”  5.a.3. System updates the user database  5.b User selects “No”  5.b.1 System cancels message prompt  5.b.2 Return to Main Flow #3. |
| Exceptions | 2.a. User fails to locate the desired account |
| Notes | Updating an account falls into two main importance: Allowing the administrator to monitor if there is any misuse of the system, and giving professors the option to update information within their account. This serves a purpose of giving professor flexibility, which satisfies the requirement of AR03. |

### 4.3.4 Use Case 4

|  |  |
| --- | --- |
| Use Case ID | 04 |
| Use Case Name | Logging in |
| Primary Actor(s) | Professor |
| Goal | User successfully logs into their account. |
| Precondition | UC:02 |
| Post Condition | Session instance *si* was created  *si* was associated with each user’s account |
| Main Flow | 1. User selects “Login” button on the “Home” web page 2. System launches the login screen 3. User inputs their email into the “Email” textbox 4. User inputs their password into the “Password” textbox 5. User clicks a “Login” button 6. System successfully validates email/password combination 7. System forwards the user to their account page |
| Alternatives Flows | 5.a System alerts user that the input was an incorrect email/password combination  5.a.1. System navigates the user back to Main Flow #3  3.a. User selects “Recover Password”  3.a.1 System forwards the user to the password recovery page  3.a.2 User inputs their email into the “Email” textbox  3.a.3 User clicks a “submit” button  3.a.4 System emails the user with a temporary password  3.a.5 User uses the password to login  3.a.6 System redirects user to a Reset Password page  3.a.7 User types in their new password  3.a.8 User clicks a “submit” button  3.a.8 System updates the user’s account |
| Exceptions |  |
| Notes | The importance of being able to logging into an account is that it satisfies the functional requirement IRP02. This requirement is important in allowing the user to view a message sent to his or her account (Req. ID. IRP03), view messages that were previously asked (Req. ID. IRP04), reply to certain messages (Req. ID. IRP05), delete certain messages (Req. ID. IRP06), forward messages to an email account (IRP07), and select the current course he or she is teaching (Req. ID IRP08) |

### 4.3.5 Use Case 5

|  |  |
| --- | --- |
| Use Case ID | 05 |
| Use Case Name | Add a course |
| Primary Actor(s) | Administrator |
| Goal | User successfully adds a course. |
| Precondition | UC:01  UC:02 |
| Post Condition | course instance *gcours was created*  *gcours* was assigned to unique ID |
| Main Flow | 1. User selects the “Add Course” tab 2. User selects the desired college 3. User selects the desired major 4. User inputs the course name 5. User selects the “Add” button 6. System adds the course into the database |
| Alternatives Flows |  |
| Exceptions | 5.a “College” not properly selected  5.a.1 System prompts an error message  5.a.2 Administrator selects “Okay” within prompt  5.a.3 Return to Main Flow #3  5.b “Major” not properly selected  5.b.1 System prompts an error message  5.b.2 Administrator selects “Okay” within the prompt  5.b.3 Return to Main Flow #4 |
| Notes | The importance of this use case is to keep the project within scope. Because the entirety of the project is based on being anonymous, it is necessary to clear the student email in case of unauthorized access to the system that causes emails to become compromised. This use case satisfies the functional requirement for administration AR05. |

### 4.3.6 Use Case 6

|  |  |
| --- | --- |
| Use Case ID | 06 |
| Use Case Name | Select current course |
| Primary Actor(s) | Professor |
| Goal | User successfully selects the courses that they are currently teaching. |
| Precondition | UC:04  User is on their account’s home page |
| Post Condition | course instance *cours* was created  *cours* was assigned to unique ID |
| Main Flow | 1. User selects “Account Settings” 2. User selects “Add Course” 3. System provides list of universities 4. User selects the desired university 5. System provides list of majors 6. User selects the desired major 7. System provides list of courses 8. User selects the desired course(s) 9. User clicks the “Submit” button 10. System saves courses within the database |
| Alternatives Flows | 7.a. System does not display desired course  7.a.1. User selects “Not seeing your course?” hyperlink  7.a.2. System renders: “Submit a course ticket.” form  7.a.3. User inputs desired course  7.a.4. User clicks “Submit”  7.a.5. System receives request for a course ticket |
| Exceptions |  |
| Notes | The importance of allowing a user to “select a course” is that it satisfies the functional requirement IRP08. Since the user may be teaching different courses each semester, giving the user the ability to adjust to their current teaching semester. This requirement is important in allowing the students to select a course in which they want to ask the question in (Req. ID IRS01), and allows the user to archive the message to view at a later date (Req. ID. IRP04). |

### 4.3.7 Use Case 7

|  |  |
| --- | --- |
| Use Case ID | 07 |
| Use Case Name | Upload student email list |
| Primary Actor(s) | Professor |
| Goal | User successfully upload a list of student emails |
| Precondition | Must use csv format  UC: 02  UC: 04 |
| Post Condition | Email instance *Eml*was created  *Eml* was assigned to unique ID  *Eml* was associated to *profAcct* |
| Main Flow | 1. User selects ‘view course’ page  2. System forwards user to course page  3. User filters classes  4. System returns classes information  5. User selects course  6. System returns course page  7. User clicks “select file” button  8. User selects a csv file  9. User clicks “upload” button  10. System reads in the file  11. System displays ‘File Upload successfully’’  12. System forwards the user to their account page |
| Alternatives Flows | 5.a.System alerts user “file not recognized”  5.a.1. System forwards user back to Main Flow 2 |
| Exceptions |  |
| Notes | The User may have multiple courses. To send an email to the entire course the professor can upload a csv of all their student’s email address. |

### 4.3.8 Use Case 8

|  |  |
| --- | --- |
| Use Case ID | 08 |
| Use Case Name | Submit message |
| Primary Actor(s) | Student |
| Goal | The student successfully submits message to the Professor |
| Precondition | UC:05  UC:06  User is on the student page |
| Post Condition | message instance *msg was created*  *msg* was assigned to unique ID |
| Main Flow | 1. User selects their desired “College” 2. User selects their desired “Major” 3. User selects their desired “Course” 4. User inputs a subject in the subject field 5. User inputs a message in the message field 6. User presses the “Submit” button 7. System sends a message to the professor’s account |
| Alternatives Flows | 4.a User inputs their personal email into “Email” textbox  4.a.1. User returns back to Main Flow #6 |
| Exceptions | 1.a.. System does not provide the desired “College”  2.a. System does not provide the desired “Major”  3.a. System does not provide the desired “Course” |
| Notes | The use case fills in the requirement of IRS02. Because this is use case encapsulate the vision behind the project, it is necessary within the scope. |

### 4.3.9 Use Case 9

|  |  |
| --- | --- |
| Use Case ID | 09 |
| Use Case Name | View a message |
| Primary Actor(s) | Professor |
| Goal | User successfully views a message |
| Precondition | UC:04  UC:08 |
| Post Condition | message instance *msg* was displayed |
| Main Flow | 1. User selects “View Messages” option 2. System lists the anonymous messages the student have sent to that particular professor 3. User selects course tab 4. User selects message he/she wishes to view 5. System displays the message |
| Alternatives Flows |  |
| Exceptions |  |
| Notes | The importance of allowing a user to “view a message” is that it satisfies the functional requirement IRP03. This requirement is important in allowing the user to see the messages the students have sent him or her (Req. ID. IRS02), as well as allowing the user to forward the message, if it is possible reply to that message, and archive the message (Req. ID. IRP07, IRP05, IRP04) |

### 4.3.10 Use Case 10

|  |  |
| --- | --- |
| Use Case ID | 10 |
| Use Case Name | Reply to message |
| Primary Actor(s) | Professor |
| Goal | User successfully replies to message. |
| Precondition | UC:07  UC:09 |
| Post Condition | message instance *rmsg* was created  *rmsg* was assigned to unique ID |
| Main Flow | 1. The user clicks “reply to all” button in message 2. The system pops out reply pop up box 3. The user types in subject in subject line 4. The user inputs message in body 5. The user clicks the “submit” button 6. The system sends an email to all the students on the course |
| Alternatives Flows | 1.a User clicks on “Reply” button  1.a.1 The system pops out reply pop up box  1.a.2 The user input subject in subject line  1.a.3 The user inputs message in body  1.a.4 The user clicks the “submit” button  1.a.5 User selects “Send”  1.a.6 System sends the message to the student |
| Exceptions | 6.a. System fails to send reply to the student  6.a.1. System displays: “Failed to send message, the email may be invalid.”  6.a.2. System forwards user back to Main Flow #1 |
| Notes | The importance of allowing a user to “reply to a message” is that it satisfies the functional requirement IRP05. This use case is important when the students request for an individual feedback when they submit a question (Req. ID. IRS02). |

### 4.3.11 Use Case 11

|  |  |
| --- | --- |
| Use Case ID | 11 |
| Use Case Name | View archive |
| Primary Actor(s) | Professor |
| Goal | User successfully views archive message(s) |
| Precondition | UC:04  UC:08 |
| Post Condition | Archive message instance, *am*, was created and displayed |
| Main Flow | 1. User selects “View archive” 2. System displays a list of courses that the user has taught 3. User selects desired course to view 4. System displays a list of semester the courses had been taught 5. User selects desired semester to view 6. System displays the list of messages 7. User selects desired message to view 8. System displays the message |
| Alternatives Flows |  |
| Exceptions | 2.a. System does not display all the courses taught by the user  5.b. System does not display all the semesters within the course  6.c. The desired message is not listed. |
| Notes | The importance of allowing a user to “view archive” is that it satisfies the functional requirement IRP04. This requirement allows the user to reread old messages they have previously archived in (Use case ID: 06) |

### 

### 

### 4.3.12 Use Case 12

|  |  |
| --- | --- |
| Use Case ID | 12 |
| Use Case Name | Delete a message |
| Primary Actor(s) | Professor |
| Goal | User successfully updates a message |
| Precondition | UC: 09 |
| Post Condition | message instance *msg* was updated |
| Main Flow | 1. User selects “Delete” 2. System prompts a message: “Are you sure you want to delete?” 3. User selects “Delete” on the prompt message 4. System deletes message from database 5. System prompts another message: “Message successfully deleted.” 6. System removes a message from “View Messages” screen |
| Alternatives Flows | 3.a. Professor selects “Cancel”  3.a.1. System prompts message: “Message was not deleted.”  3.a.2. System forwards User back to Main Flow #2 |
| Exceptions |  |
| Notes | The importance of this use case satisfies the requirement of IRP07. This is necessary to free space in the account’s database |

### 

### 

### 4.3.13 Use Case 13

|  |  |
| --- | --- |
| Use Case ID | 13 |
| Use Case Name | Forward email |
| Primary Actor(s) | Professor |
| Goal | User successfully forwards a message. |
| Precondition | UC:09 |
| Post Condition | message instance *fmsg* was created  *fmsg* was assigned to unique ID |
| Main Flow | 1. User selects “Forward Email” 2. System prompts: “Are you sure? This message shall be forwarded to your .edu email.” 3. User selects “Forward” 4. System forwards the message 5. System prompts a message: “Message has been forwarded.” |
| Alternatives Flows | 2.a. Professor selects “Cancel”  2.a.1. System prompts: “Forward message canceled.”  2.a.2 System navigates user to UC: 09    4.a. System fails to forward the message  4.a.1. System prompts: “Message failed to send. Would you like to try again “Yes”, “No”.”  4.a.2. User selects “Yes”  4.a.3. System navigates user to Main Flow #5 and forwards the message again  4.b. System fails to forward the message  4.b.1. System prompts: “Message failed to send. Would you like to try again “Yes”, “No”.”  4.b.2. User selects “No”  4.b.3. System navigates user to UC:09 |
| Exceptions |  |
| Notes | The importance of this use case satisfies the requirement of IRP08. This allows professor to view the message outside of web application |

## 

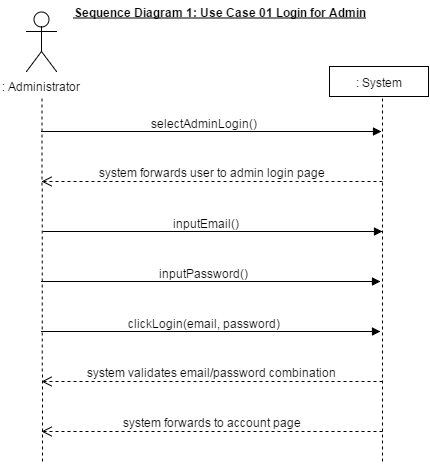
## 4.3.14 Use Case 14

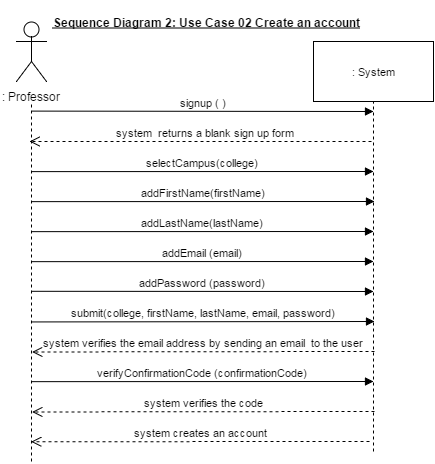
|  |  |
| --- | --- |
| Use Case ID | 14 |
| Use Case Name | Clear student message |
| Primary Actor(s) | Administrator |
| Goal | The user successfully clears a student message from the database |
| Precondition | UC:01  UC:02  UC:08 |
| Post Condition | Student email instance, *msg*, was updated |
| Main Flow | 1. User navigates to the “Manage Student Email” web page 2. User selects professor account 3. User selects major 4. User selects course 5. User selects message 6. User deletes message from database 7. System updates the message 8. System prompts message “Message is deleted” |
| Alternatives Flows |  |
| Exceptions |  |
| Notes | Removes message entirely from database. |

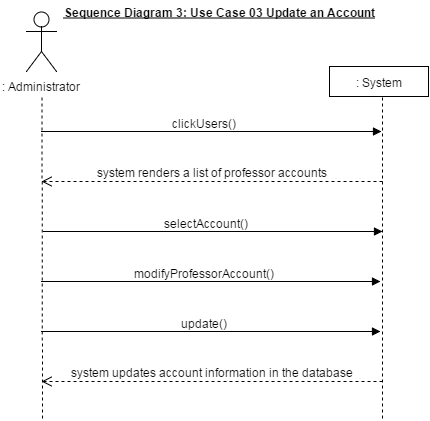
## 

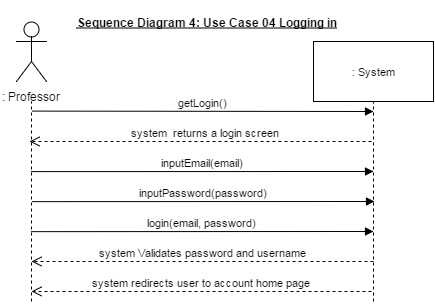
## 

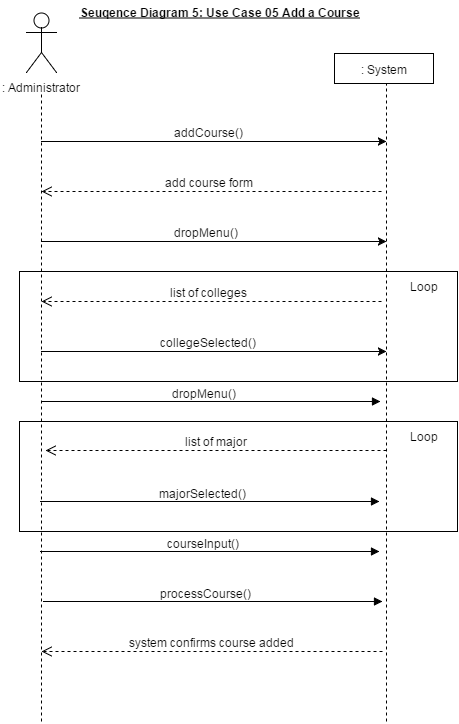
## 4.4 Sequence Diagrams

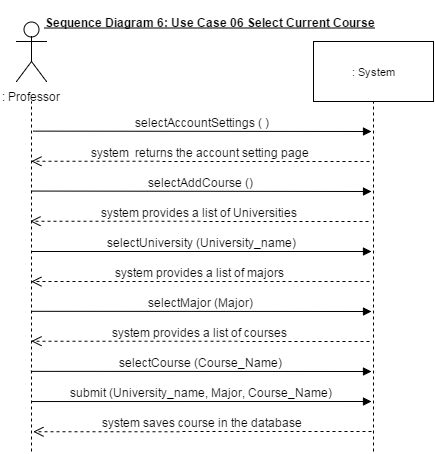


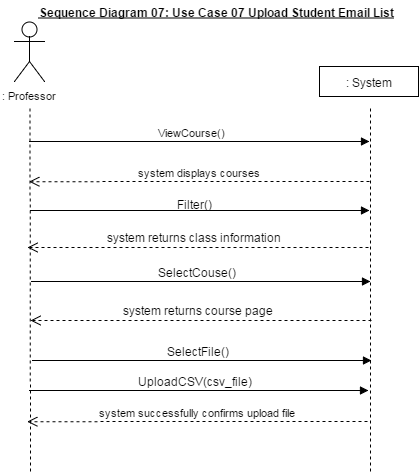


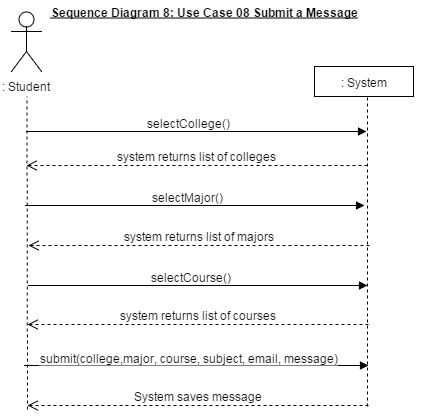


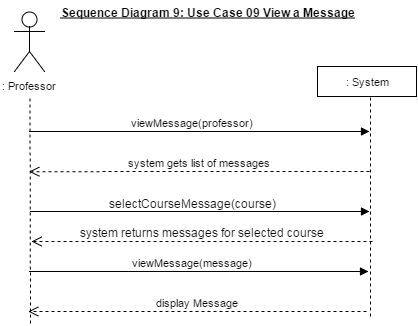


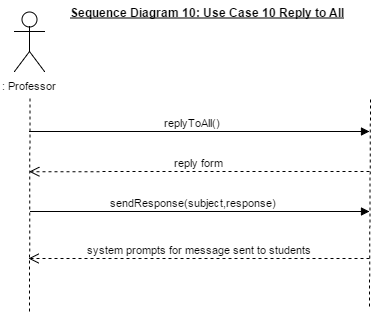


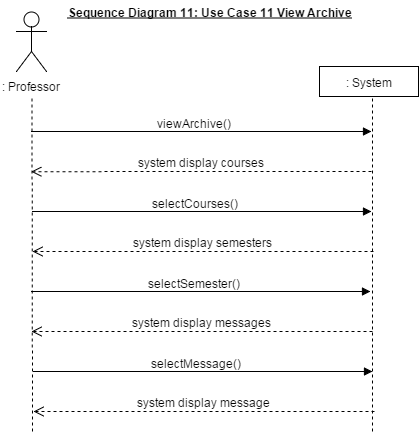


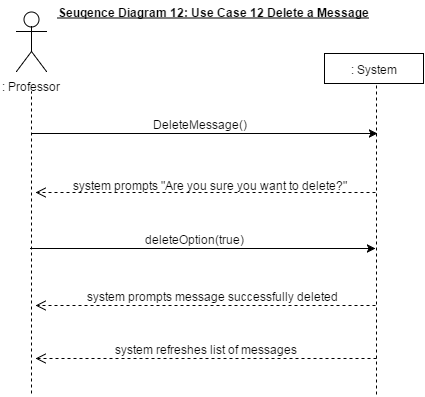


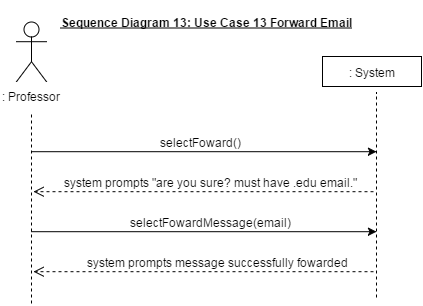


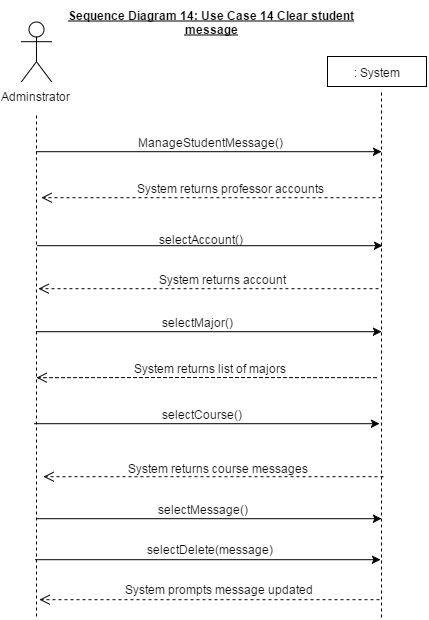












## 4.5 Domain Model Diagrams

## 

## 4.6 System Class Diagrams

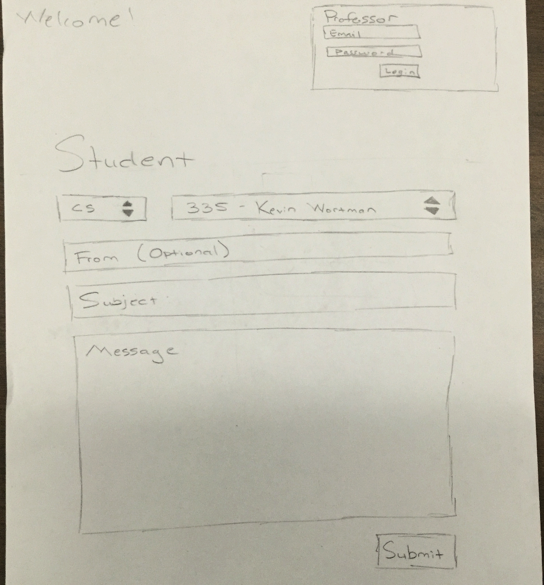
## 4.7 Database Information (If Applicable)

### 4.7.1 Database Tables (If Applicable)

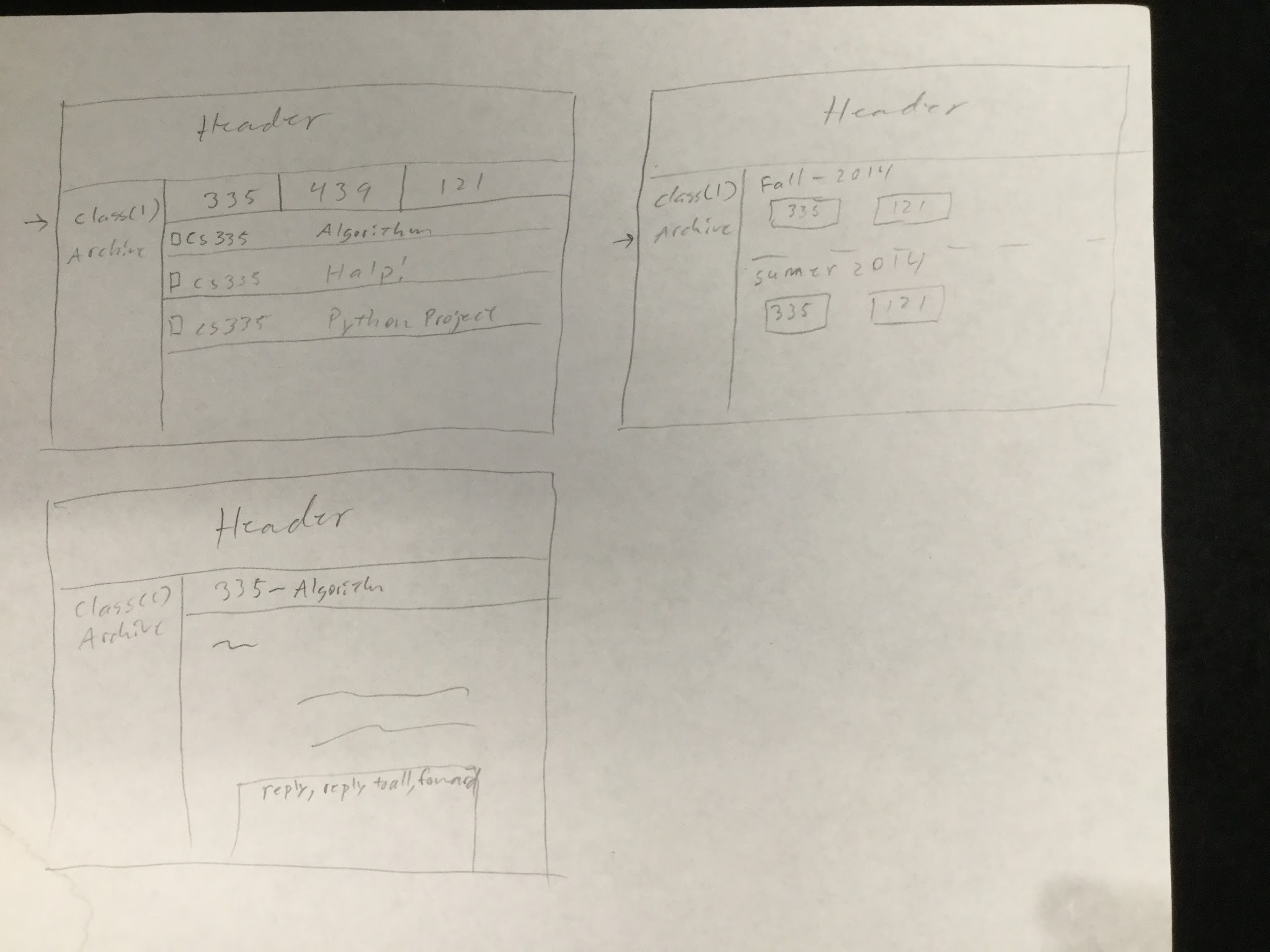
## 

## 

## 4.8 Initial Sketch



## 4.9 Interface Sketch



# 5. Test and Integration Plans and Results

## 5.1 Test Plans

## 5.2 Test Cases

## 5.3 Test Results

# 6. Installation Instructions and User Documentation

## 6.1 Prerequisites

Download and install node and npm from: <https://nodejs.org/en/>

## 6.2 Database Installation (If Applicable)

Install MongoDB for OSX (10.7 or later):

<https://docs.mongodb.org/manual/tutorial/install-mongodb-on-os-x/>

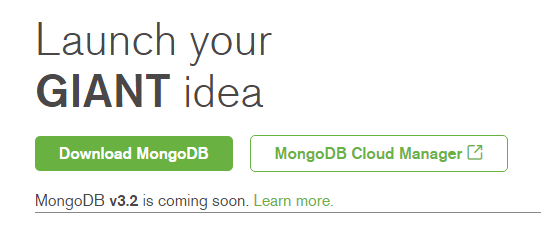
1. ruby -e “$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install>)”
2. brew update
3. brew install mongodb
4. sudo mkdir /data/db
5. sudo mkdir -p /data/db
6. sudo chown -R $USER /data/db

## 

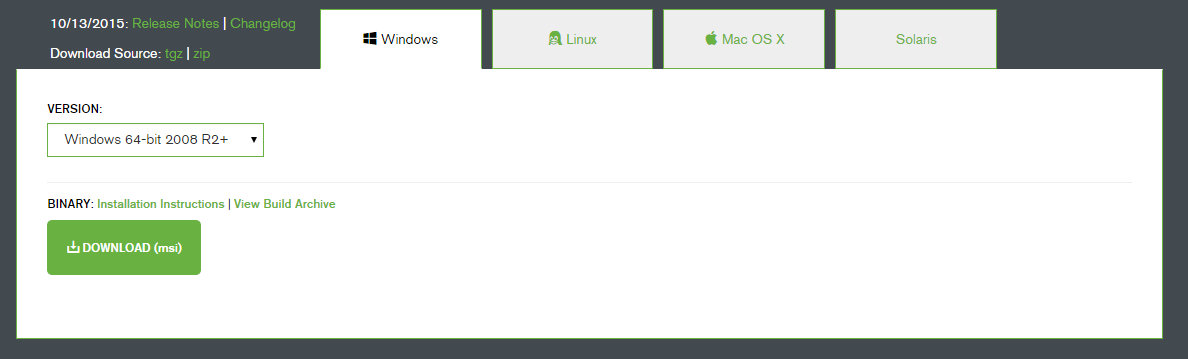
Installing MongoDB for windows 8/10 64 bit processor:

[http://mongodb.org/](http://mongodb.org)

1. Click the Download MongoDB button .



1. Download the windows compatible version of Mongodb for a 64 bit processor "Windows 64-bit \*2008R2+".



1. Click on the download (msi) button.
2. Run through the standard windows install.
3. By default windows will install MongoDB in c:\Program Files\MongoDB

## 6.3 System Administration User (If Applicable)

The system Administrator is responsible for adding an available course to the database. To get started as an Administrator refer to the manual below.

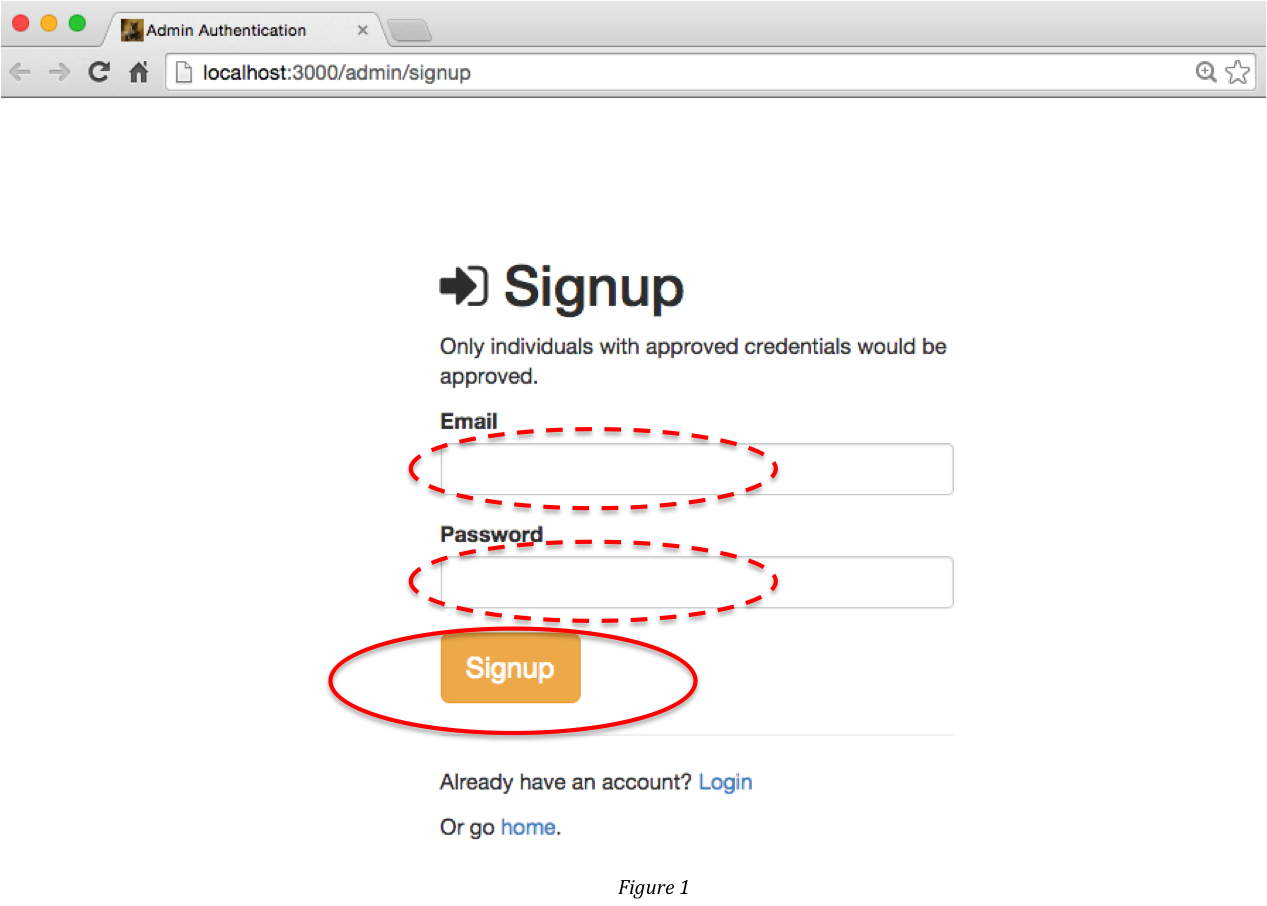
**Navigation**

1. To navigate throughout the web app as an Administrator, use the navigation bar at the top of browser.

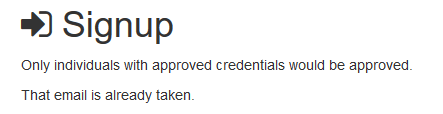


**Sign Up**

1. To sign up as an administrator, navigate to <http://localhost:3000/admin/signup>
2. The Administrator sign up requires an email address and a password refer to Figure 1

.

1. First enter an email address into the email textbox shown in Figure 1.
2. Second enter a password in the password textbox shown in Figure 1.
3. Once both the email and password textbox are filled out, click the signup button
4. If an account under same email address inputted is already in the system will prompt the user “The email is already taken.”



1. Once you sign up with a valid Email and Password, you will be allowed access to an Administrator profile page shown in figure 2

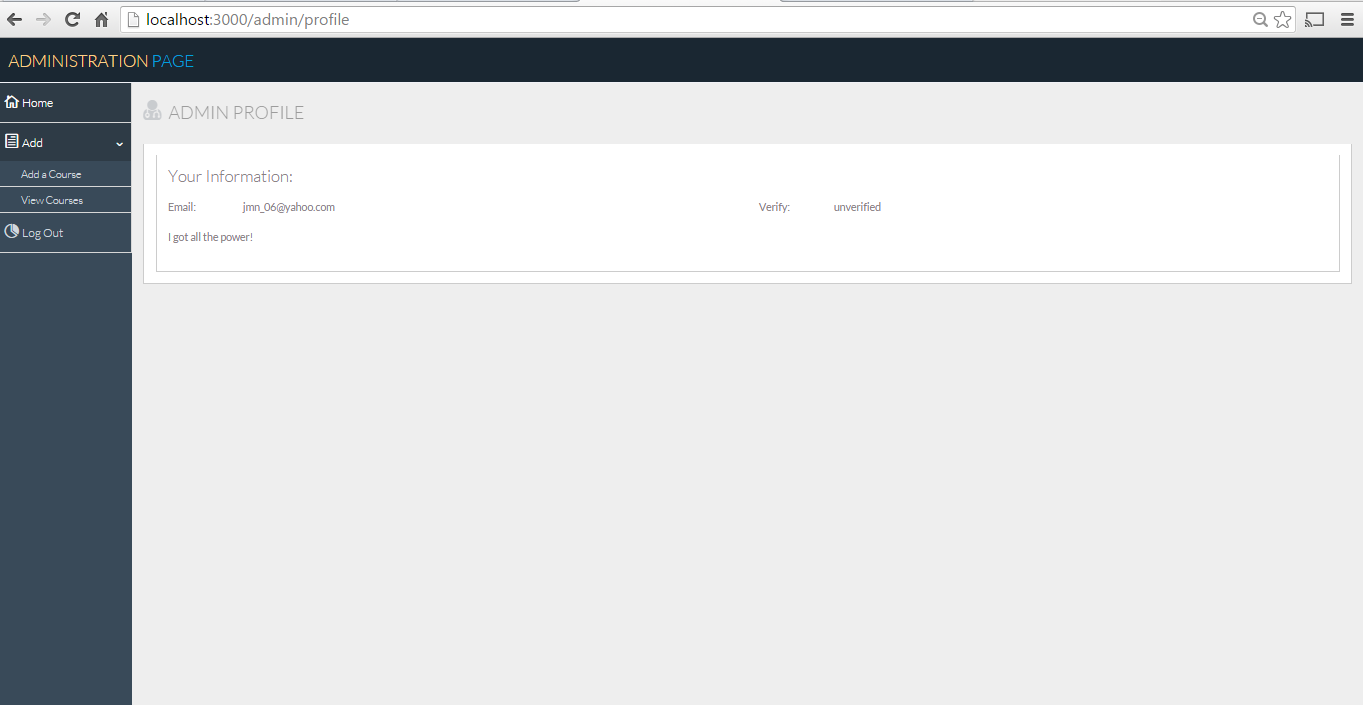
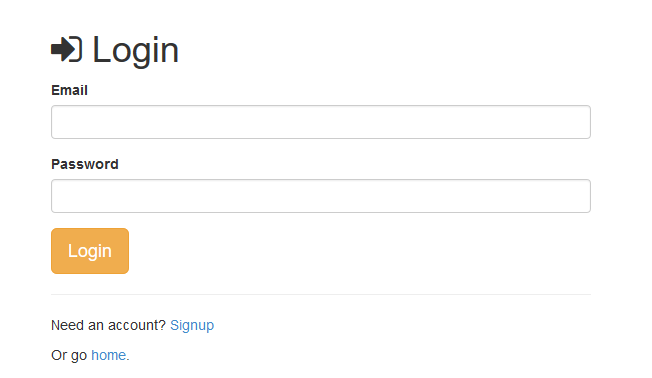


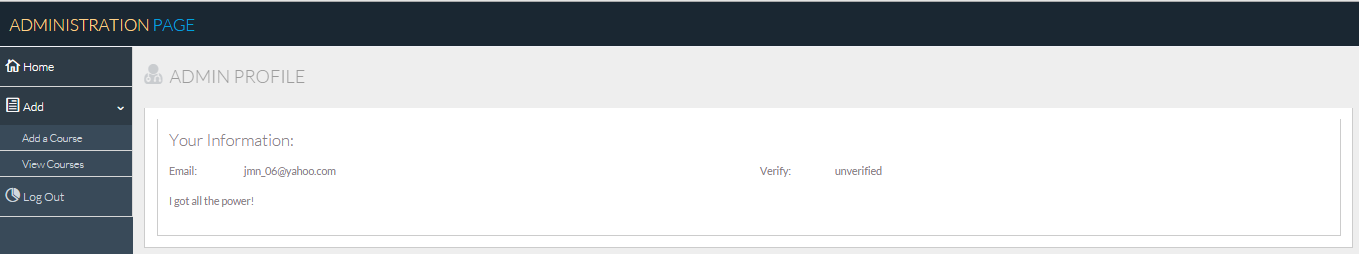
figure 2

**Login**

1. To login to an existing Administrator account navigate to: <http://localhost:3000/admin/login>

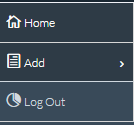


1. Type in the Email address that was used during the adminstrator signup
2. Type in the Password that was used during the adminstrator signup
3. Click the Login button
4. Once you enter the same Email and Password that you specified during sign up, you will be allowed access to the Administrator profile page.

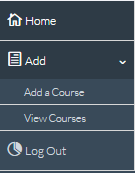


**Add a Course**

1. To add a course, Click “Add” in the menu on the left hand side of the page.



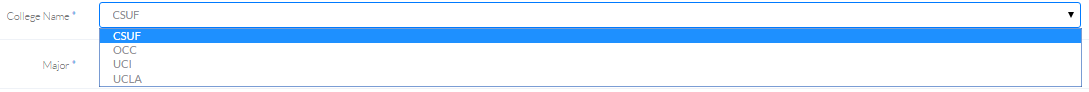
1. The menu will expand giving you the additional option of “Add a course” and “View a Course”



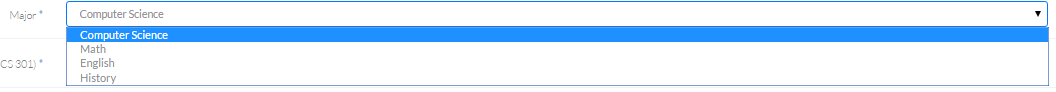
1. Click on “Add a Course”
2. To add a course you will need to fill a Add course form shown in the figure below. The fields that are required to be filled out in order to add a course are marked with an ‘\*’.



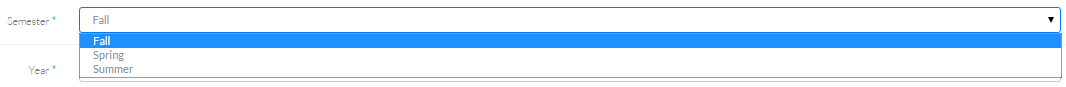
1. First select a college. When you click on college, a drop down list of available colleges are provided. Select a college in which the course is offered at.



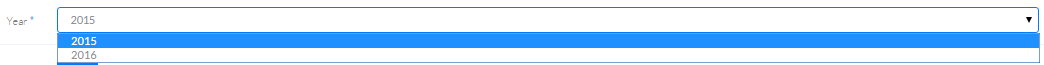
1. Then Select a Major in which the course is offered in the drop down menu shown below.



1. Type in a course number into the course textbox.
2. Select a semester in which this course was offered in the drop down menu shown below.



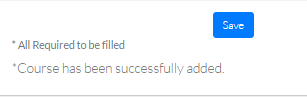
1. Select the year in which the course is offered in the drop down menu shown below.



1. Once all the fields are filled out click the save button
2. If one of the fields are not filled out you will see a “! Please fill out this field”

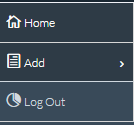


1. Once all the fields are filled out properly, you will see a “\*Course was successfully added” at the lower left hand side of the page.

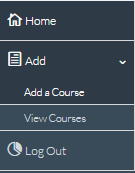


**View a Course**

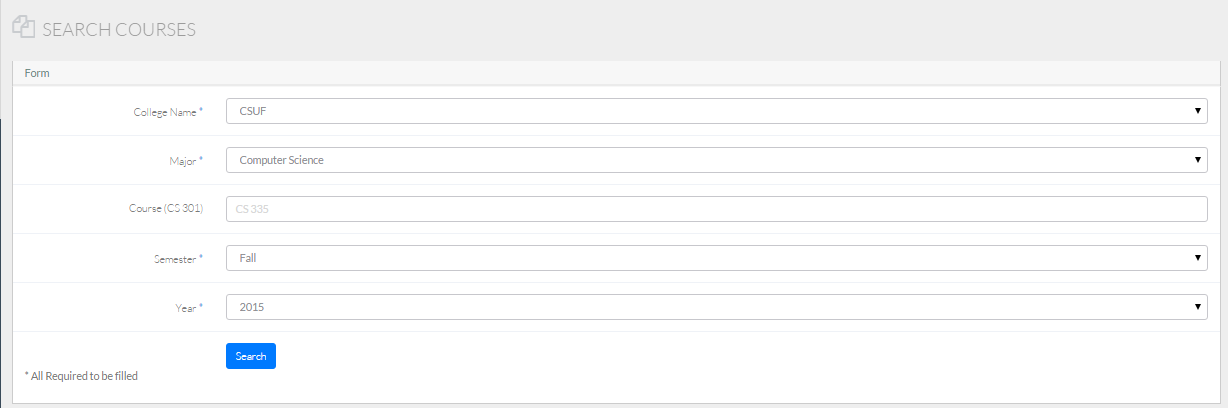
1. To view a list of courses that you or another Administrator have added, click on “Add” on the menu towards the left of the page.



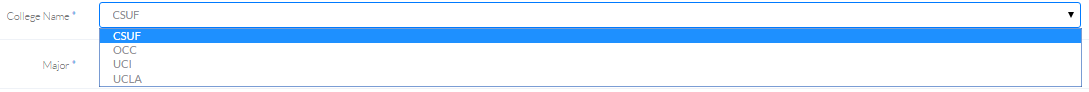
1. The menu will expand giving you the additional option of “Add a course” and “View a Course”



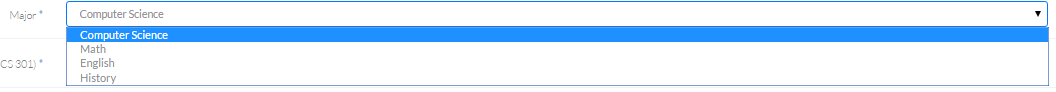
1. To view a course fill out a form to narrow down your search of the available courses. The fields marked as ‘\*’ are required fields.



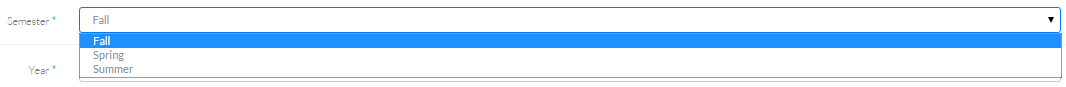
1. First select a college. When you click on college, a drop down list of available colleges are provided. Select a college in which the course is offered at.



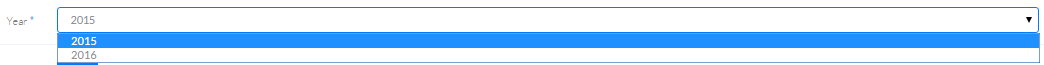
1. Then Select a Major in which the course is offered in the drop down menu shown below.



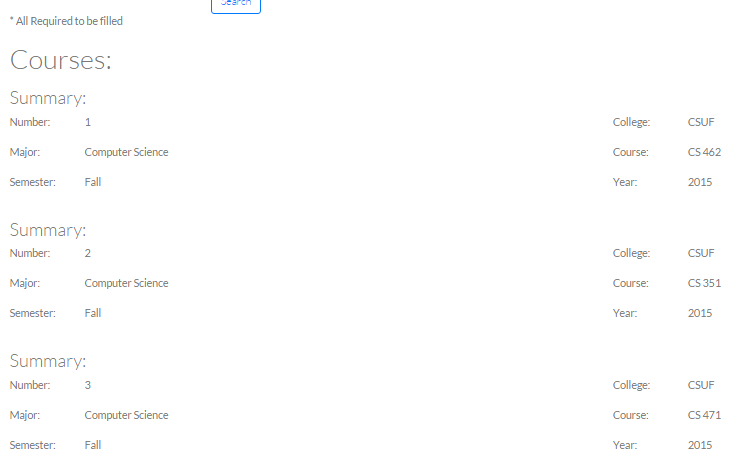
1. Type in a course number into the course textbox. This field is optional.
2. Select a semester in which this course was offered in the drop down menu shown below.



1. Select the year in which the course is offered in the drop down menu shown below.



1. Once all the fields are filled out click search button.
2. Once all the fields are filled out properly, you will see a list of courses that are available underneath the search form.



## 6.4 User Manual, Operational Manual and Instruction

### 

**Introduction**

The purpose of the ‘Student Feedback’ web app is to provide professors with an open communication link between his or her students so students can ask questions they feel too embarrassed to ask about in person. The ‘Student Feedback’ web app allow students to remain anonymous behind the packet switching network. The web app is a tool intended for professors to get student feedback on issues students may be having with a concept or an assignment. The web app provides a professor with easy access to any student’s messages as well as a the capability to send a reply to the entire class. For professors, please continue reading for more information on how you can get started using the ‘Student Feedback’ web app. If you are a student, you can skip over this section and jump to ‘User Manual.

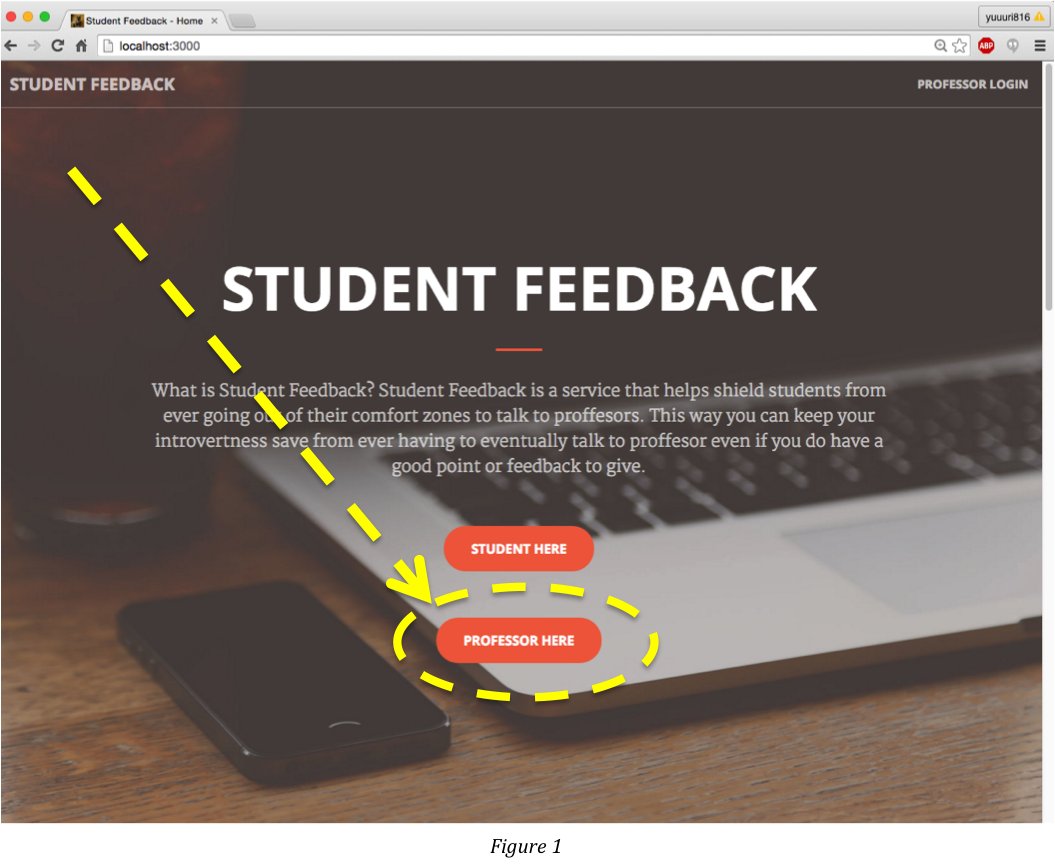
### 6.4.1 User Manual - Professor

**Navigation**

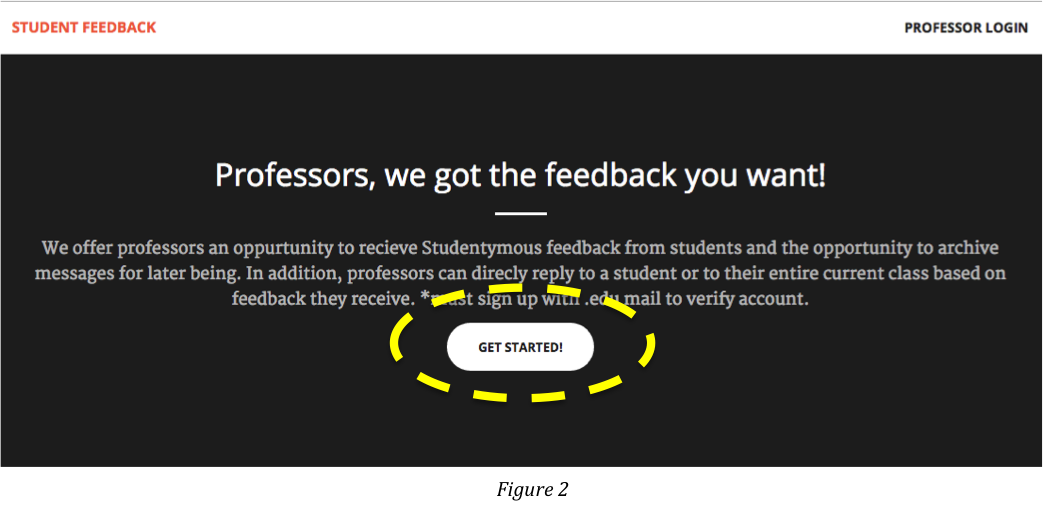
1. To navigate throughout the web app as an Administrator, use the navigation bar at the top of browser.



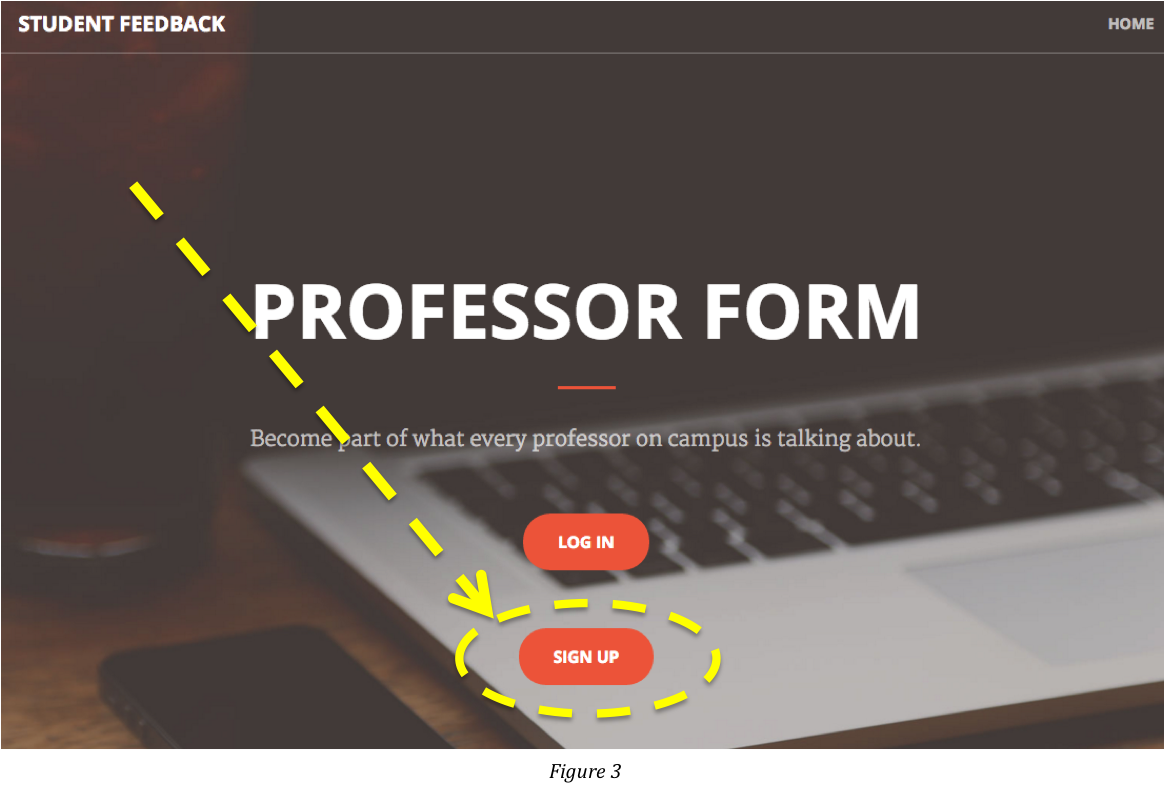
**Main Page**



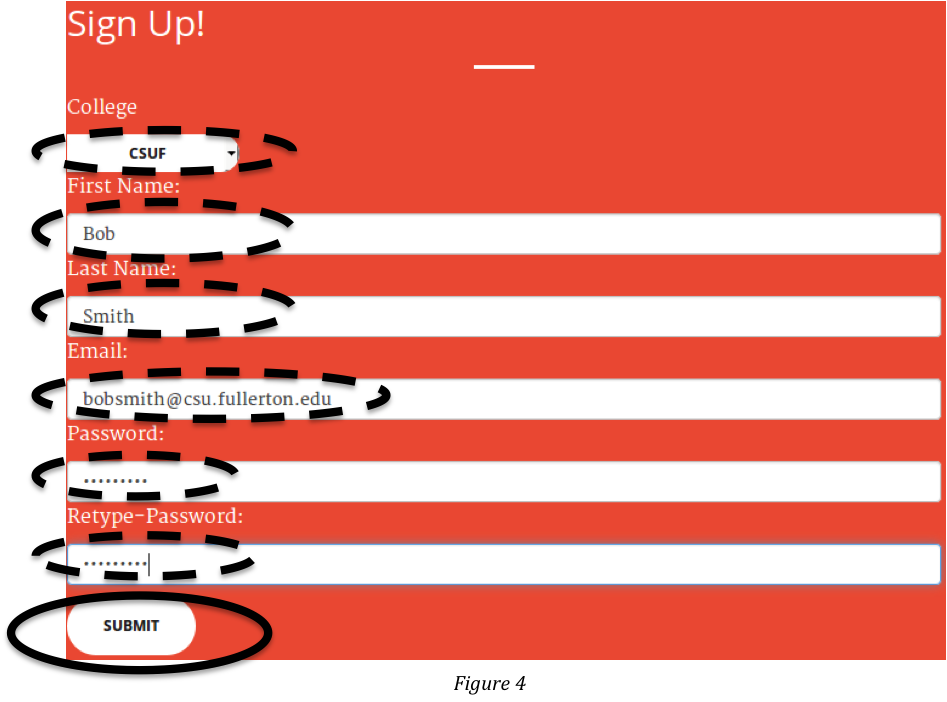
User clicks ‘Professor Here’ on main page. (Figure 1)



User is navigated to this page. (Figure 2)



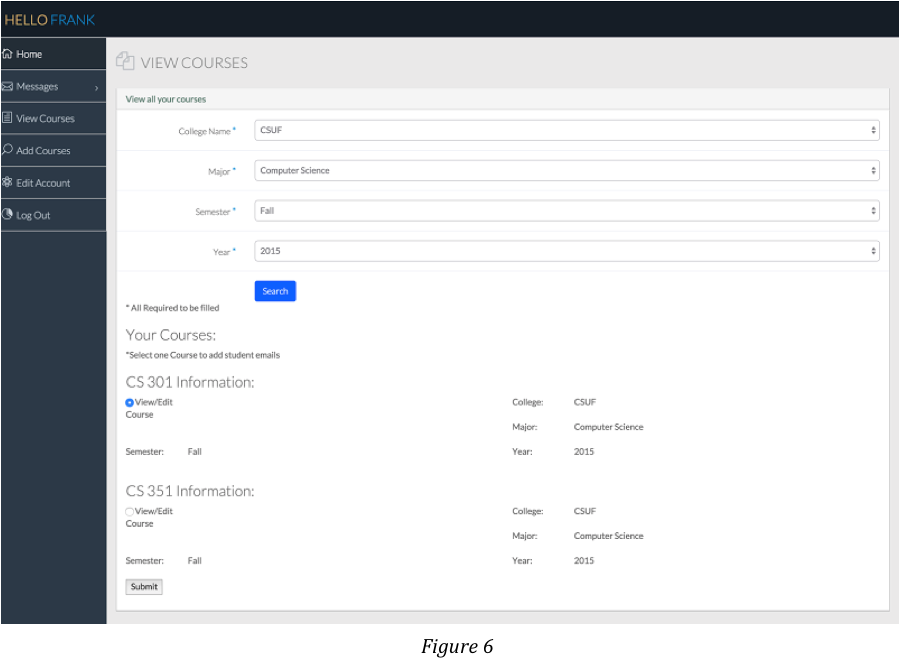
User is now navigated to ‘Professor Form’ page. User can create ‘Professor Account’ by clicking ‘SIGN UP’ button (Figure 3) in which takes you to ‘Sign Up’ page (Figure 4).



If user has already signed up, user can log in by clicking ‘LOG IN’ button shown in Figure 3 instead of clicking ‘SIGN UP’ button. For the convenience, user can always log in by clicking ‘PROFESSOR LOGIN’ on the top right corner of the main page (Figure 1)

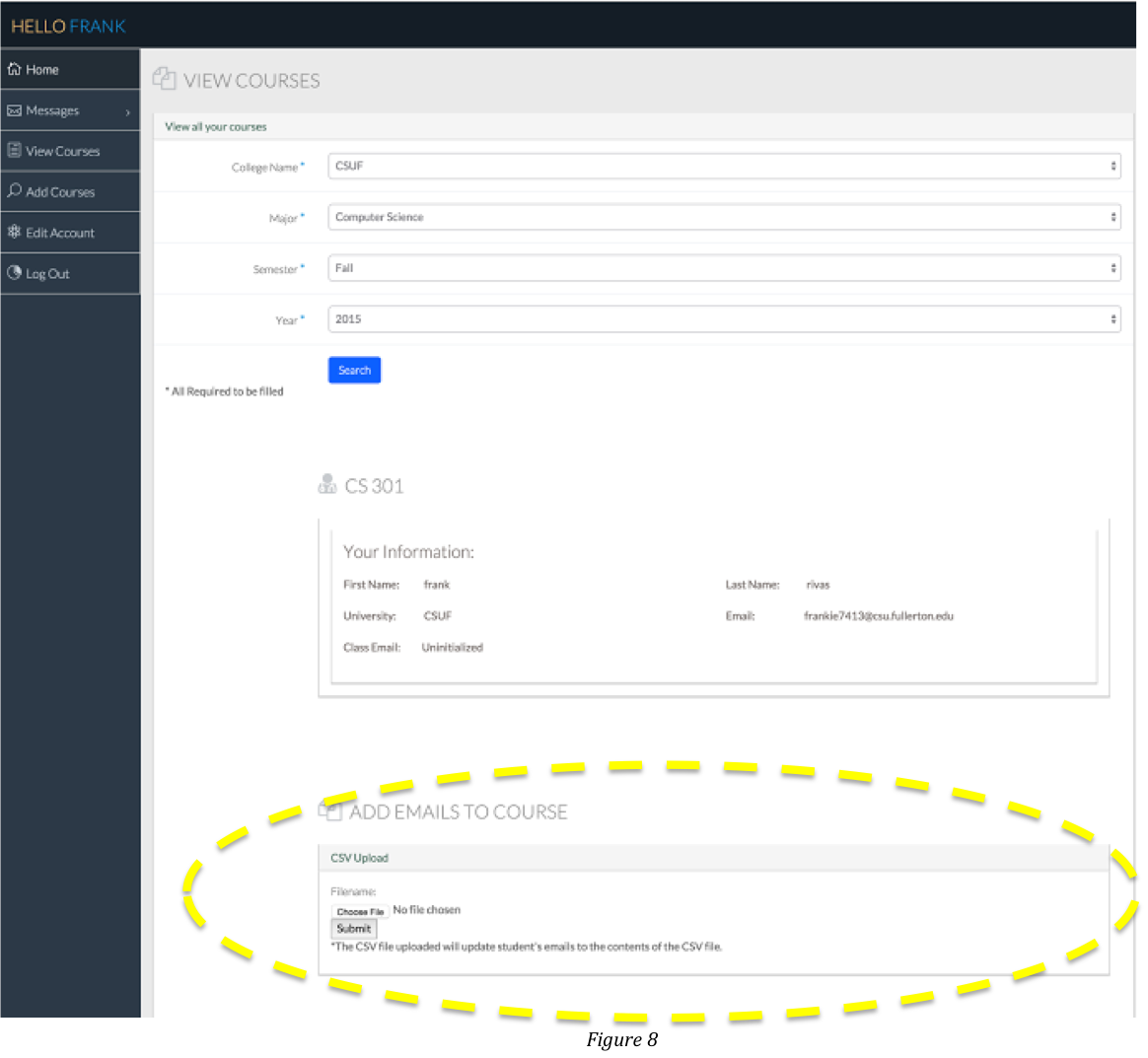
### fig5.png

Once user logs in, the user’s profile page is displayed (Figure 5). From this page, user can select to add/view the courses currently being taught (Figure 6) and can check out the anonymous ‘Student Feedback’ messages (Figure 7). User verification link is sent through by email.

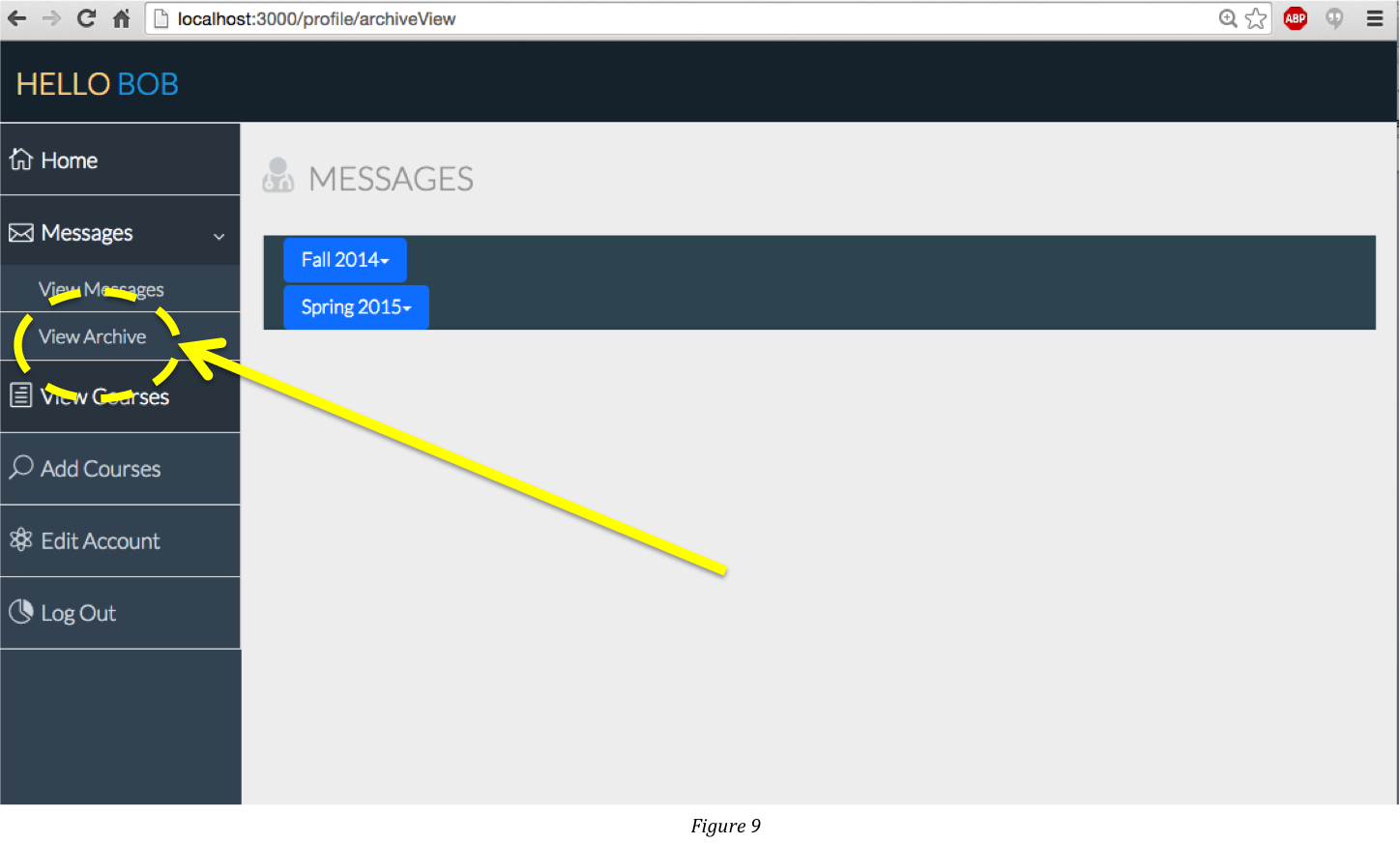




Would you like to reply to all of the students in class? No problem! See Figure 8 - You can set it up by uploading CVS file that contains all students’ email addresses. After setup is done, you can click ‘Reply All’ to do mass emailing!

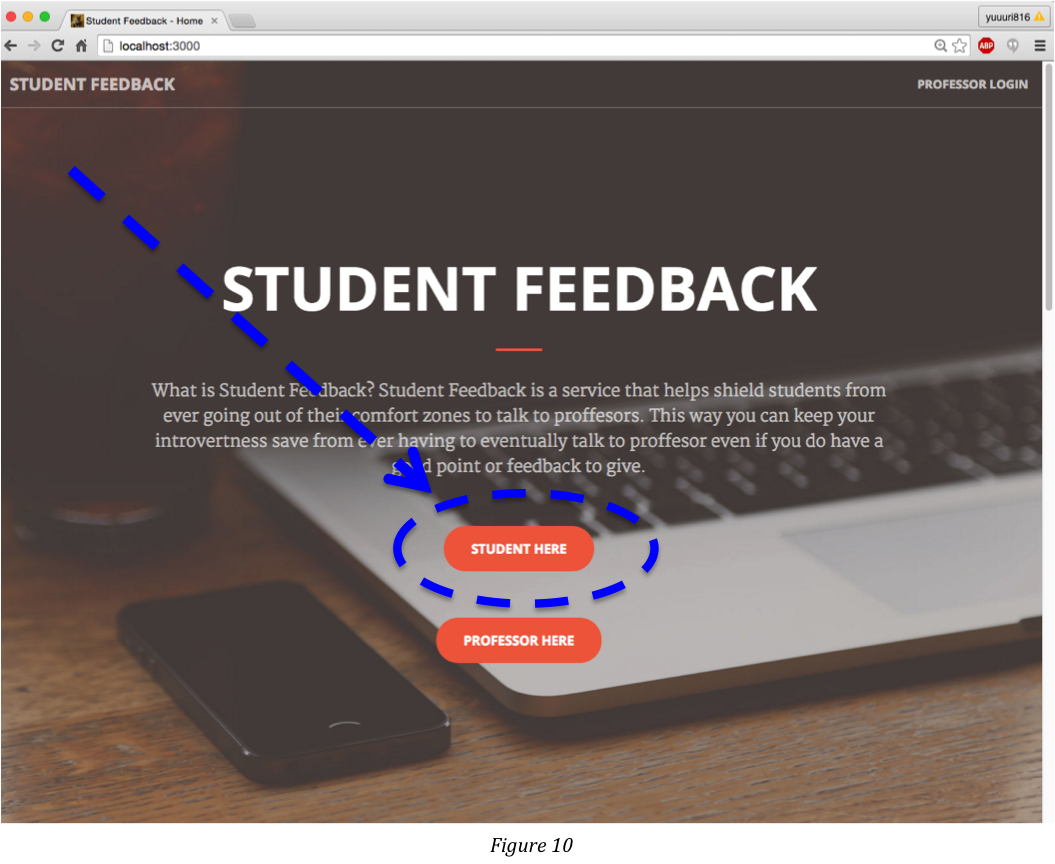


To check out past ‘Student Feedback’ messages, click ‘View Archive’ (Figure 9).

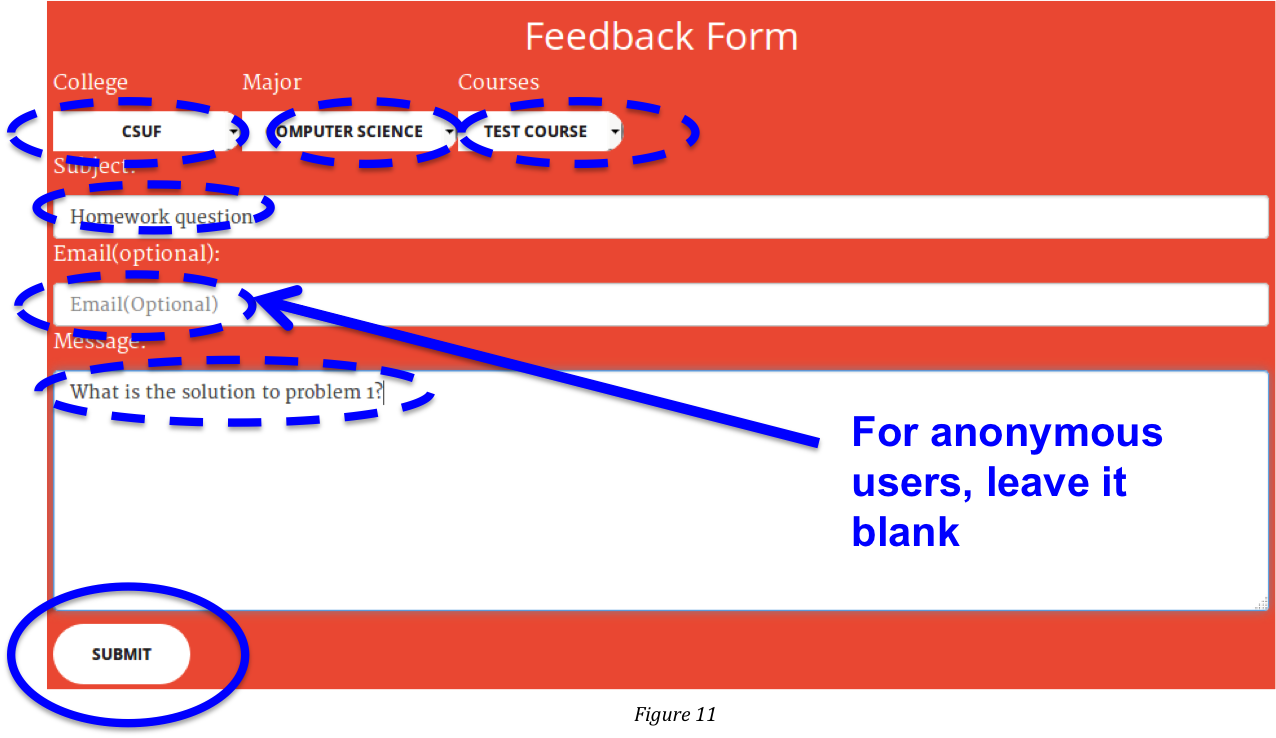


### 6.4.1 User Manual - Student

From the main page (Figure 10), user click ‘STUDENT HERE’ button.



User is navigated to ‘Feedback Form’ page.



User selects and enters all fields as shown in Figure 11 and click ‘Submit’.



Confirmation pops up after user sends his/her ‘Student Feedback’ message!

### 6.4.3 Instruction to run

OSX:

1. Open terminal.
2. Navigate to the desired directory.
3. Type: git clone <https://github.com/frankie7413/462Developer.git>
4. Type: cd 462Developer/myapp
5. Type: npm install
6. Type: cd data
7. Type: mongod
8. Open second terminal
9. Type: cd 462/Developer/myapp
10. Type: npm start
11. Open Chrome or Firefox
12. Navigate to: <http://localhost:3000> on Chrome or Firefox

Windows:

1. Open a Command Prompt.
2. Navigate to the project directory >cd …\462Developer\myapp
3. Type: npm install
4. Open a Second Command Prompt
5. Navigate to the MongoDB folder. By default it is located in /Program Files\MongoDB\Server\3.0\bin
6. Type: mongod --dbpath c:\path\462Developer\myapp\data\

The path is replaced with the path where the 462Developer folder is located

1. In the first Command Prompt, opened in step 1, Type: npm start
2. Open Chrome or Firefox
3. Navigate to: <http://localhost:3000> on Chrome or Firefox

## 6.5 Server Manual

npm start or DEBUG=myapp npm start (For developers)

# 7. Recommendations for the Enhancement

Make compatible to mobile devices such as iPhone, iPad, android.

# 8. References and Bibliography

Larman, Craig. *Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and Iterative Development*. Upper Saddle River, NJ: Prentice Hall PTR, 2005. Print

Purewal, Semmy. Learning Web App Development. Vol. 1. Sebastopol: O’Reilly Media, 2014. Print.

Seguin, Karl. “The Little MongoDB Book.” The Little MongoDB Book. N.p., 20 Apr. 2011. Web. 28 Nov. 2015.

Javascript and ajax: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Express: <http://expressjs.com/>

Template for the home page: <http://startbootstrap.com/template-overviews/creative/>

Profile page: <http://www.bootstrapzero.com/bootstrap-template/nice-admin>

Sign up, Log in, and Authentication:

<https://scotch.io/tutorials/easy-node-authentication-setup-and-local>

Mailing: <http://adilapapaya.com/docs/nodemailer/>