**“Quick Feedback”**

**CS462 Project Version 1.08**

by Group 5: Vinh, Frankie, Sami, Julia, Michael, Yuri

Table of Contents

1. Introduction 1

1.1. Description of the Problem 1

1.2. Goals and Objectives 1

1.3. Purpose 2

1.4. Scope 3

1.5. Definitions, Acronyms and Abbreviations 3

1.6. Development Environment 3

1.6.1. Software 3

1.6.2. Hardware 3

1.7. Operational Environment 4

1.7.1. Software 4

1.7.2. Hardware 4

2. Requirement Description 5

2.1. Functional Requirements 5

2.1.1. Functional Requirements for Users 5

2.1.1.a. Functional Requirements for Users – Student 5

2.1.1.b. Functional Requirements for Users – Professor 5

2.1.2. Functional Requirements for Administrators 5

2.2. Non-Functional Requirement 6

2.2.1. Security Requirements 6

2.2.2. Performance Requirements 6

2.2.3. Reliability Requirements 6

2.2.4. Availability Requirements 7

2.2.5. Efficiency Requirements 7

2.2.6. Usability Requirements 7

2.2.7. Maintainability Requirements 7

2.2.8. Portability Requirements 8

2.2.9. Testability Requirements 8

3. Management Process 9

3.1. Project Schedule 9

3.1.1. Working Days Overview 9

3.1.2. Tasks 11

3.2. Project Plan 13

3.2.1. Phase Plan 13

3.2.2. Iteration Plan 13

3.2.3. Iteration Objectives 15

3.2.4. Releases 15

3.2.5. Project Schedule 15

3.2.6. Project Resourcing 15

3.3 Project Monitoring and Control 16

3.3.1. Budget Control Plan 16

3.3.2. Quality Control Plan 16

3.3.3. Reporting Plan 16

3.3.4. Measurement Plan 17

3.4. Risk Management Plan 17

4. Design Description 19

4.1. Product Perspective 19

4.1.1. Product Vision 19

4.2. Product Feature 19

4.3. Use Cases 20

4.3.1. UC1: 20

4.3.2. UC2: 22

4.3.3. UC3: 23

4.3.4. UC4: 24

4.3.5. UC5: 24

4.3.6. UC6: 25

4.3.7. UC7: 26

4.3.8. UC8: 27

4.3.9. UC9: 28

4.3.10. UC10: 29

4.3.11. UC11: 30

4.3.12. UC12: 30

4.3.13. UC13: 31

4.3.14. UC14: 32

4.4. System Sequence Diagrams 33

4.5. Domain Model Diagrams 33

4.6. System Class Diagrams 33

4.7. Database Information (If Applicable) 33

4.7.1 Database Tables (If Applicable) 33

4.8. Initial Sketch 33

5. Test and Integration Plans and Results 35

5.1. Test Plans 35

5.2. Test Cases 35

5.3. Test Results 35

6. Installation Instructions and User Documentation 36

6.1. Prerequisites 36

6.2. Database Installation (If Applicable) 36

6.3. System Administration User (If Applicable) 36

6.4. User Manual, Operational Manual and Instruction 36

7. Recommendations for the Enhancement 37

8. References and Bibliography 37

Team: **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 Shell/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

1. Allows students to anonymously message 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 professors 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 message 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 professor to review past feedback.

## 1.4. Scope

## To build a system where a student can submit feedback message to a particular professor’s course. Additionally professors will be able to manage messages, respond to messages individually, forward messages to personal email to reply to class, and view archived messages.

## 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

### 2.1.1. Functional Requirements for Users

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 delete a message |
| IRP08 | User shall be able to forward to his/her personal email account |

### 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 clear student email |

## 2.2. Non-Functional Requirement

### 2.2.1. Security Requirements

|  |  |
| --- | --- |
| **Security Requirements** | |
| SR01 | The web app 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 web application, which will improve students’ needs and professor’s teaching performance. Also, student asks professor any question 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

|  |
| --- |
| **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. |
| **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. |
| **September 7, 2015** |
| **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.  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. |
| **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. |
| **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, Michael, 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 state of the system. The goals of the use cases are updated as well. The overall documentation is reviewed and edited after group conversations. |

### 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 communications 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 * Identify use cases * Define project environment * Project management and scheduling * 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 * UML Use Case Diagram * Architecture baseline defined * Demonstration * Finalize Report 1 | | |
| Construction Phase – Iteration 4 | October 7 | October 18 |
|  | | |
| Construction Phase – Iteration 5 | October 19 | October 30 |
|  | | |
| Construction/Transition Phase – Iteration 6 | October 31 | November 11 |
|  | | |
| Transition Phase – Iteration 7 | November 12 | November 23 |
|  | | |
| Transition Phase – Iteration 8 | November 24 | November 28 |

### 3.2.3. Iteration Objectives

Iteration Objectives are stated above 3.2.3 Iteration Plan.

### 3.2.4. Releases

November 29, 2015

### 3.2.5. Project Schedule

### 3.2.6. 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 |
| Other |  |  |
| Sub Total | $1,680 |  |
| Phase II |  |  |
|  |  |  |
|  |  |  |

### 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** |
| Time Constraint | Lay out deadlines for each iteration |
| Scope of the project | Pre-defining what extent of the project. |
| Choosing the right development tools | Experimenting each tool to see what tools work best with |
| Skills on web development | Interview potential members for skills in web development |

**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

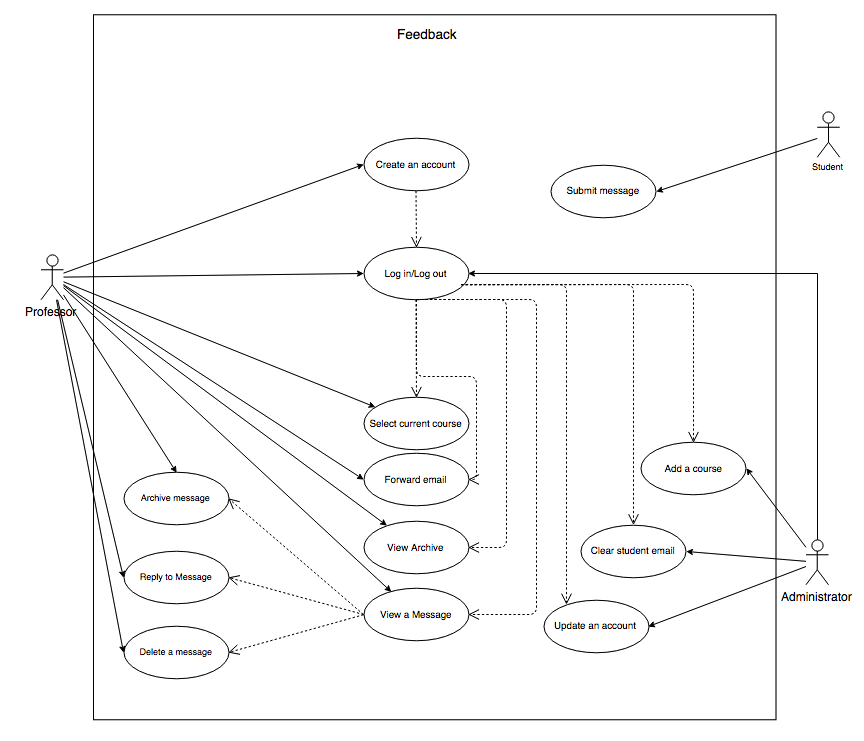
## 4.1.1. Product Vision

“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 app presents a communication link for students freely and anonymously.

## 4.2. Product Feature

## 4.3. Use Cases

Use Case Diagram



### 4.3.1. UC1:

|  |  |
| --- | --- |
| Use Case ID | 01 |
| Use Case Name | Create an account |
| Primary Actor(s) | Professor |
| Goal | User creates his/her account successfully |
| Pre-Conditions | System displays the home web page |
| Post-Conditions | System stores the account information, with an associated unique ID number into the user database |
| Main Flow | 1. User selects “Sign up” button to sign up for an account on “Home” web page 2. System displays a blank creation form 3. User selects their affiliated “Campus” 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 clicks the “Submit” button 9. System verifies input information by sending a verification email containing a secret code to the user 10. User receives the email in their email inbox 11. User will use the secret code within the email and input into the web page 12. System verifies the code 13. System creates an account saving the professor’s information along with a unique ID into the database |
| Alternative Flows | A.5.    System displays an error message: “Account already exists.”  A.5.1. System navigates the user to login page (UC:02)  B.5.    System displays an error message: “Email is not in .edu format.”  B.5.1. System navigates user back to Main Flow #5  A.8.    System prompts “Password must be 8 characters long”  A.8.1. Return to Main Flow #7  B.8. System prompts “Password must contain at least 1 digit.”  B.8.1. Return to Main Flow #7  C.8. System prompts “Password must contain at least 1 letter.”  C.8.1 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: 04). 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) |

### 4.3.2. UC2:

|  |  |
| --- | --- |
| Use Case ID | 02 |
| Use Case Name | Logging in/Logging out |
| Primary Actor(s) | Professor |
| Goal | The user logs into their account using their email/password combination  The user logs in/out into his/her account |
| Pre-Conditions | System has an existing account for user (UC:01) |
| Post-Conditions | The system forwards the user to their account page |
| 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 |
| Alternative Flows | A.5.    System alerts user that the input was an incorrect  A.5.1. System navigates the user back to Main Flow #2  A.7. User selects “Log out”  A.7.1. System logs user out of their 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) |

### 4.3.3. UC3:

|  |  |
| --- | --- |
| Use Case ID | 03 |
| Use Case Name | Select current course |
| Primary Actor(s) | Professor |
| Goal | User selects the courses that they are currently teaching |
| Pre-Conditions | System has an existing account for user (UC:01)  System has forwarded the user to their account home page (UC:02) |
| Post-Conditions | System will reflect the courses selected by the user in the database  System will display the user’s name and their course for students in order for the students to send a message to the user |
| 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 |
| Alternative Flows |  |
| Exceptions | A.3.    System does not display desired course  A.3.1. User selects “Not seeing your course?” hyperlink  A.3.2. System renders: “Submit a course ticket.” form  A.3.3. User inputs desired course  A.3.4. User clicks “Submit” |
| 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) |

### 4.3.4. UC4:

|  |  |
| --- | --- |
| Use Case ID | 04 |
| Use Case Name | View a message |
| Primary Actor(s) | Professor |
| Goal | The user views a message |
| Pre-Conditions | System has an existing account for user (UC:01)  System has forwarded the user to their account home page (UC:02) |
| Post-Conditions | System displays the messages the students have sent to the user then move the message into the archive with the user’s approval |
| Main Flow | 1. User selects “View Messages” screen 2. The system lists a page of the courses they are currently teaching 3. User clicks on the message they wish to view 4. System lists the anonymous messages the student have sent 5. System displays the message |
| Alternative 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) |

### 4.3.5. UC5:

|  |  |
| --- | --- |
| Use Case ID | 05 |
| Use Case Name | Reply to message |
| Primary Actor(s) | Professor |
| Goal | The user replies to a specific student without compromising the student’s identity. |
| Pre-Conditions | System has an existing account for user (UC:01)  System has forwarded the user to their account home page (UC:02)  System has placed student’s messages into user’s inbox  System displays the “View Messages” screen |
| Post-Conditions | 1. System displays the messages that user’s reply to the student has been sent (without compromising the student’s identity) 2. System moves the message into the archive |
| Main Flow | 1. User selects desired message to view 2. User clicks on “Reply” button 3. System renders “Reply” form 4. User fills in their response in the “Response” text box 5. User selects “Send” 6. System sends the message to the student |
| Alternative Flows |  |
| Exceptions | A1.     Message does not contain a return email address  A.1.1. System’s reply button is replaced with “Return address was not provided.” text  B.5.    System fails to send reply to the student  B.5.1. System displays: “Failed to send message, the email may be invalid.”  B.5.2. 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). |

### 4.3.6. UC6:

|  |  |
| --- | --- |
| Use Case ID | 06 |
| Use Case Name | Archive message |
| Primary Actor(s) | Professor |
| Goal | The user archives message into the appropriate position |
| Pre-Conditions | System has an existing account for user (UC:01)  System has forwarded the user to their account home page (UC:02) |
| Post-Conditions | System archives message into the appropriate position |
| Main Flow | 1. User navigates into “View Message” screen 2. User selects the desired message 3. System renders the desired message information 4. User selects “Archive Message” button 5. System prompts “Are you sure you want to archive this message?” 6. User selects “Archive” 7. System archives message |
| Alternative Flows | A.6.    User selects “Cancel”  A.6.1. User is forward to Main Flow #3 |
| Exceptions | A.1.    System does not display the desired message to archive |
| Notes | The importance of allowing a user to “archive a message” is that it satisfies the functional requirement  IRP04. This requirement is important in allowing the user to archive a message to view again at a later date (Use case ID: 07). |

### 4.3.7. UC7:

|  |  |
| --- | --- |
| Use Case ID | 07 |
| Use Case Name | View archive |
| Primary Actor(s) | Professor |
| Goal | The user views their archives of questions previously submitted |
| Pre-Conditions | System has an existing account for user (UC:01)  System has forwarded the user to their account home page (UC:02) |
| Post-Conditions | System displays user’s archived messages |
| 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 |
| Alternative Flows | A.2.    System does not display all the courses taught by the user  B.5. System does not display all the semesters within the course  C.7. The message to view is not listed within the semester |
| Exceptions | A.2. System does not display all the courses taught by the user  B.5. System does not display all the semesters within the course  C.7. The message to view is not listed within the semester |
| 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) |

### 4.3.8. UC8:

|  |  |
| --- | --- |
| Use Case ID | 08 |
| Use Case Name | Delete a message |
| Primary Actor(s) | Professor |
| Goal | The user deletes an existing message |
| Pre-Conditions | System has an existing account for user (UC:01)  System has forwarded the user to their account home page (UC:02)  System has placed student’s messages into user’s inbox |
| Post-Conditions | System deletes message from the database |
| Main Flow | 1. User selects “View Message” button 2. System renders “View Message” web page 3. User selects desired message from list of message 4. System renders the desired message 5. User selects “Delete” 6. System prompts a message: “Are you sure you want to delete?” 7. User selects “Delete” on the prompt message 8. System deletes message from database 9. System prompts another message: “Message successfully deleted.” 10. System removes a message from “View Messages” screen |
| Alternative Flows | A.6.    Professor selects “Cancel”  A.6.1. System prompts message: “Message was not deleted.”  A.6.2. System forwards User back to Main Flow #2 |
| Exceptions | B.2. System does not display the desired message |
| Notes | The importance of this use case satisfies the requirement of IRP07. This is necessary to free space in the account’s database |

### 4.3.9. UC9:

|  |  |
| --- | --- |
| Use Case ID | 09 |
| Use Case Name | Forward email (to personal email) |
| Primary Actor(s) | Professor |
| Goal | The user forwards a message to a personal email account |
| Pre-Conditions | System has an existing account for user (UC:01)  System has forwarded the user to their account home page (UC:02)  System has placed student’s messages into user’s inbox |
| Post-Conditions | System forwards the message to the user’s personal email account |
| Main Flow | 1. User selects “View Message” 2. User select desired message 3. System renders the desired message 4. User selects “Forward Email” 5. System prompts: “Are you sure? This message shall be forwarded to your .edu email.” 6. User selects “Forward” 7. System forwards the message 8. System prompts a message: “Message has been forwarded.” |
| Alternative Flows | A.5.    Professor selects “Cancel”  A.5.1. System prompts: “Forward message canceled.”  A.5.2. System forwards user back to Main Flow #3  A.6.    System fails to forward the message  A.6.1. System prompts:  “Message failed to send. Would you like to try again “Yes”, “No”.”  A.6.2. User selects “Yes”  A.6.3. System navigates user to Main Flow #5 and forwards the message again  B.6.    System fails to forward the message  B.6.1. System prompts: “Message failed to send. Would you like to try again “Yes”, “No”.”  B.6.2. User selects “No”  B.6.3. System navigates user to Main Flow #3 |
| Exceptions |  |
| Notes | The importance of this use case satisfies the requirement of IRP08. This allows professor to view the message outside of web application |

### 4.3.10. UC10:

|  |  |
| --- | --- |
| Use Case ID | 10 |
| Use Case Name | Submit message |
| Primary Actor(s) | Student |
| Goal | The student submits message to the professor |
| Pre-Conditions | System displays feedback submission web page |
| Post-Conditions | 1. The system sends the message to the recipient specified by user 2. The system displays that the message is submitted |
| Main Flow | 1. User navigates to the student web page 2. User selects their desired “College” 3. User selects their desired “Major” 4. User selects their desired “Course” 5. User inputs a subject in the subject field 6. User inputs a message in the message field 7. User presses the “Submit” button 8. System sends a message to the professor’s account |
| Alternative Flows | A.5.    User inputs their personal email into “Email” textbox  A.5.1. User returns back to Main Flow #6 |
| Exceptions | A.2.   System does not provide the desired “College”  A.3.   System does not provide the desired “Major”  A.4.   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. |

### 4.3.11. UC11:

|  |  |
| --- | --- |
| Use Case ID | 11 |
| Use Case Name | Logging in for administrator |
| Primary Actor(s) | Administrator/admin |
| Goal | The user successfully logs in his/her account |
| Pre-Conditions | System has an existing account for user (UC:01) |
| Post-Conditions | System forwards the user to their account page |
| Main Flow | 1. User selects “Administrator Login” 2. System forwards user to the administrator login web page 3. User types in their user name in the “Username” 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 |
| Alternative Flows | A.5.    System alerts user that email/password combination  A.5.1. System forwards user back to Main Flow #2 |
| Exceptions | B.5.1. System alerts user that the account does not exist in the database  B.5.2. System forwards user back to Main Flow #2 |
| 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. |

### 4.3.12. UC12:

|  |  |
| --- | --- |
| Use Case ID | 12 |
| Use Case Name | Update an account |
| Primary Actor(s) | Administrator |
| Goal | The user modifies a professor account in the system |
| Pre-Conditions | User has access to the web application  System has an existing account for the professor (UC:01)  System has an existing account for user (UC:01)  System has forwarded the user to their account home page (UC:02) |
| Post-Conditions | The professor’s account is updated |
| Main Flow | 1. User clicks on “Users” on their administrator page 2. System will render 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 |
| Alternative Flows | A.5.    User selects “Delete”  A.5.1. System prompts message: “Are you sure? This account will permanently be deleted?”  A.5.2. User selects “Yes”  A.5.1. System deletes the account from the database |
| Exceptions | A.2 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. |

### 4.3.13. UC13:

|  |  |
| --- | --- |
| Use Case ID | 13 |
| Use Case Name | Add a course |
| Primary Actor(s) | Administrator |
| Goal | The user adds courses for professor selection |
| Pre-Conditions | System has an existing account for user (UC:01)  System has forwarded the user to their account home page (UC:02)  System has granted user access control prior to adding a course |
| Post-Conditions | A new courses is added in the system |
| Main Flow | 1. User navigates to the administrator web page 2. User selects the “Add Course” tab 3. User selects the desired college 4. User selects the desired major 5. User inputs the course name 6. User selects the “Add” button 7. System adds the course into the database |
| Alternative Flows | A.7   “College” not properly selected  A.7.1   System prompts an error message  A.7.2   User selects “Okay” within prompt  A.7.3   Return to Main Flow #3  B.7   “Major” not properly selected  B.7.1   System prompts an error message  B.7.2   Administrator selects “Okay” within the prompt  B.7.3   Return to Main Flow #4  C.7   “Professor” not properly selected  C.7.1   System prompts an error message  C.7.2   User selects “Okay” within prompt  C.7.3   Return to Main Flow #6 |
| Exceptions |  |
| Notes | This use case is important because it allows the professor to select the courses they are currently taking. This in turn allows student to view which course-professor relation they wish to submit the message to. The use case satisfies the functional requirement of administration AR04 |

### 4.3.14. UC14:

|  |  |
| --- | --- |
| Use Case ID | 14 |
| Use Case Name | Clear student email |
| Primary Actor(s) | Administrator |
| Goal | The user clears a student email from the list of student emails |
| Pre-Conditions | System has forwarded the user to their account home page (UC:02) |
| Post-Conditions | System removes student email from database |
| Main Flow | 1. User navigates to the “Manage Student Email” web page 2. User chooses the specific student email address 3. User clicks “Remove” 4. System removes the email address |
| Alternative Flows |  |
| Exceptions |  |
| 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. |

## 4.4. System Sequence Diagrams

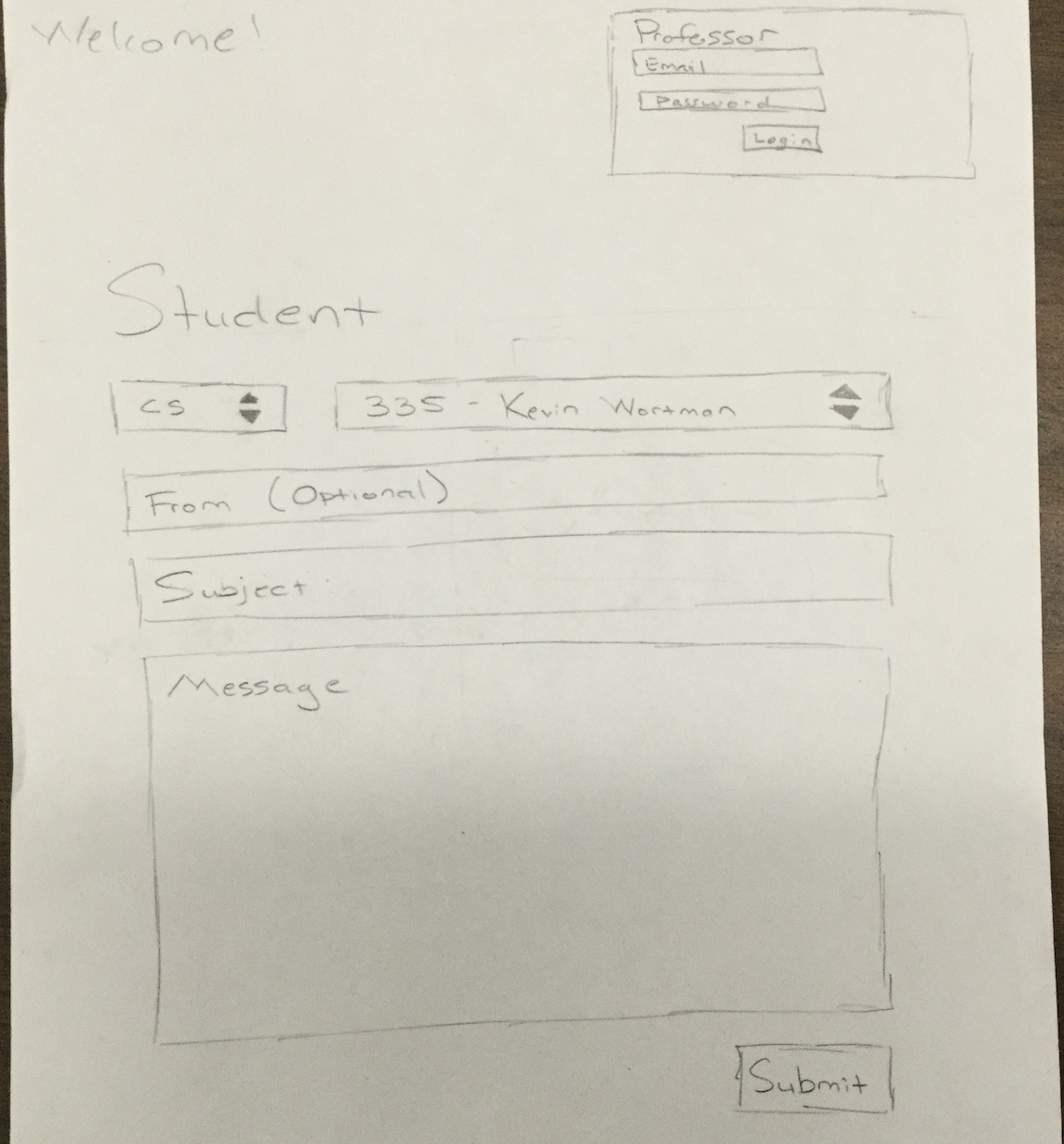
## 4.5. Domain Model Diagrams

## 4.6. System Class Diagrams

## 4.7. Database Information (If Applicable)

## 4.7.1. Database Tables (If Applicable)

## 4.8. Initial Sketch



# 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

## 6.2. Database Installation (If Applicable)

## 6.3. System Administration User (If Applicable)

## 6.4. User Manual, Operational Manual and Instruction

# 7. Recommendations for the Enhancement

# 8. References and Bibliography