# Project Plan

Project Title: Code Swap

**Team Members: Andrew Michaelis, Brandon Knight, Caleb Drake, Jackson Melling**

**Client: Alex Lo**

# Contents

# Project Description

The main purpose of this project is to provide an efficient peer-reviewing tool for a classroom setting. This tool will allow professors and teachers the ability to add their students to the system by uploading a formatted file with the student information or through a wizard provided by the system. When the teacher adds a new assignment, the system will then match up each student to a previously specified number of peers and allow peer-reviewing of the peer material. The system will be very versatile, allowing for any number of users and a variety of different types of assignments -- while the system will mostly be used for performing peer reviews based on code quality, it can also be used to peer review written work such as papers or essays.

# Feature List

Features are listed in order of priority.

## 1. User Access Levels

Each user will be able to log into the system to access the system. The system will have a variety of roles, including Admin, Faculty, Teaching Assistant, and Student. Each of these roles will have a specific set of user access roles -- Admin can add, view, and modify all users in the system; Faculty can add students, but can only view or modify students that they have added or that belong to their courses. Teaching Assistants can only view students (and submissions) that belong to their courses. Students are the most limited, and can only view themselves and the anonymous peers assigned to them.

## 2. Secure Login with OpenID

The system will allow users to sign in via OpenID, which allows users to use their accounts from other websites such as Google, Yahoo, and Facebook.

## 3. Administrative Tools

As mentioned, the system will allow Administrators to add new users to the system, modify some traits of other user's in the system (such as their user access level, their visible classes, and profile if one exists), and remove users from the system.

## 4. Installer for Deployment

The system will be bundled into an easy-to-use installer that can be downloaded and executed on a computer that will be used as the host machine for the Code Swap system. The only OS being targeted at this time is Ubuntu 12.04.

## 5. File Transfers

Teachers will be allowed to upload a formatted file containing student information (Most likely Name, e-mail address, and Role) and have the system automatically invite these students to the system and add them to the appropriate course.

Students will be able to upload their completed coursework for peer review, and their assigned peers will be able to download it, review it on their own computer, and then either re-upload it with their comments attached or add comments directly on the site.

## 6. E-mail Notifications

The system will be able to generate e-mails to users.

- All users will receive an email in each of the following scenarios:

- ○ A user is invited to the system.
- ○ A user registers a new account to the system.
- A user with Student access will receive an email when:
  - ○ A peer completes a review of the student's submission.
  - ○ The student uploads a file to the system.
  - ○ The teacher or TA comments on their submission.
- A user with Faculty access will receive an email when:
  - ○ They have the option for updates enabled and a student submits a review or uploads a file.
- A user with Administrator access will receive an email when:
  - ○ A faculty member is added to the system.
  - ○ A system error occurs.

## 7. Grade Reporting

A user with Faculty or TA access levels will be able to generate a grade report for a particular class or assignment. The system will then gather the information from the database including peer comments, average peer score, and other relevant information and display it in a tabular format.

## 8. Peer Assigning Algorithm

The system will be able to match students with anonymous peers in such a way that certain constraints are met. The known criteria at this time are as follows:
- ○ Students should not be matched with the same peer twice unless they've been matched with every other student once already.
- ○ Each student should have the same number of peers

## 9. Class Set-up Wizard

A Faculty user will be able to create a new course in the system by following a 'wizard' that will walk them through the steps. They will be able to name the course, add students (via upload or manual entry), and add TAs or other faculty.

# Metrics

The following metrics will be tracked during each sprint:

### 1. Number of Commits

The team will track number of commits to the master branch. This will be used to track not only how often features get worked on, but who is working on features.

### 2. Emails

The number of emails to the client and between group members will be tracked as a way of ensuring constant communication. Email will be the primary form of communication outside of group meetings so information can be logged and reviewed later if necessary.

### 3. Use Cases Completed

The team will develop use cases, and track how many use cases to a given feature are completed when that feature is worked on.

### 4. Hours per Week

Hours of work a week, including debugging, research, prototyping, and actual development. This will be categorized in the end-of-sprint report (number of hours in each category per week).

### 5. Test Coverage

Tracking the number of tests written for the features in a sprint and what parts of the features are covered.

# Risks

## Risk Management

Risks will be managed throughout the process of developing our Senior Project.  The riskiest situations have been detailed below.

## Risk Categorization

Risks will be organized based on severity using the following categories:

4. Catastrophic – If this risk were to occur, the project may take major losses of time or functionality

3. Critical – If this risk were to occur, the project may suffer losses of time, functionality, or cost, but to a lesser extreme than a catastrophic risk.

2. Marginal – If this risk were to occur, the project may suffer slight setback on the development timeline.

1. Negligible – If this risk were to occur, the project may suffer minimal loss of time.

## Risk Table

| Risks | Probability of Occurrence | Impact | Risk Value |
|---|---|---|---|
| Integration with various systems | 60% | 4 | 2.4 |
| Difficulties deploying on AWS | 40% | 3 | 1.2 |
| Team troubles with Ruby/ Rails | 35% | 2 | 0.7 |

*Integration with various systems*

We will have to work to make sure that we are able to deploy to multiple platforms, and are comfortable with writing instructions, or a script that will install our product on such platforms. These platforms include: Linux servers, AWS (see next point), and other server technologies.

*Difficulties deploying on AWS*

We must deploy our product on AWS as a requirement for our client, and as none of us are familiar with AWS or how to deploy an application to it, we may have unforeseen troubles with the technologies.

*Troubles with Ruby/Rails*

Throughout development our team may have troubles with learning / developing in rails and this may impede our progress on certain parts of the project. We don't foresee this as a huge risk, but it is definitely a possibility.

# Deployment Plan

The project will be deployed on a weekly basis to a VM hosted by Rose-Hulman as well as the system set up on AWS. Source code will be available on a public GitHub page [1]. The tentative schedule for the next 8 weeks is below; Other features not mentioned below will be implemented in future sprints.

Sprint 1:  Login with user access levels [Feature 1], Secured with OpenID [Feature 2], Installer [Feature 4]

Sprint 2: Admin Tools [Feature 3]

Sprint 3: E-mail notifications system [Feature 6]

# Appendix

[1] GitHub Repository:  https://github.com/alexlo03/CodeSwap/
[2] Trello KanBan:  https://trello.com/b/HOdiXfRn