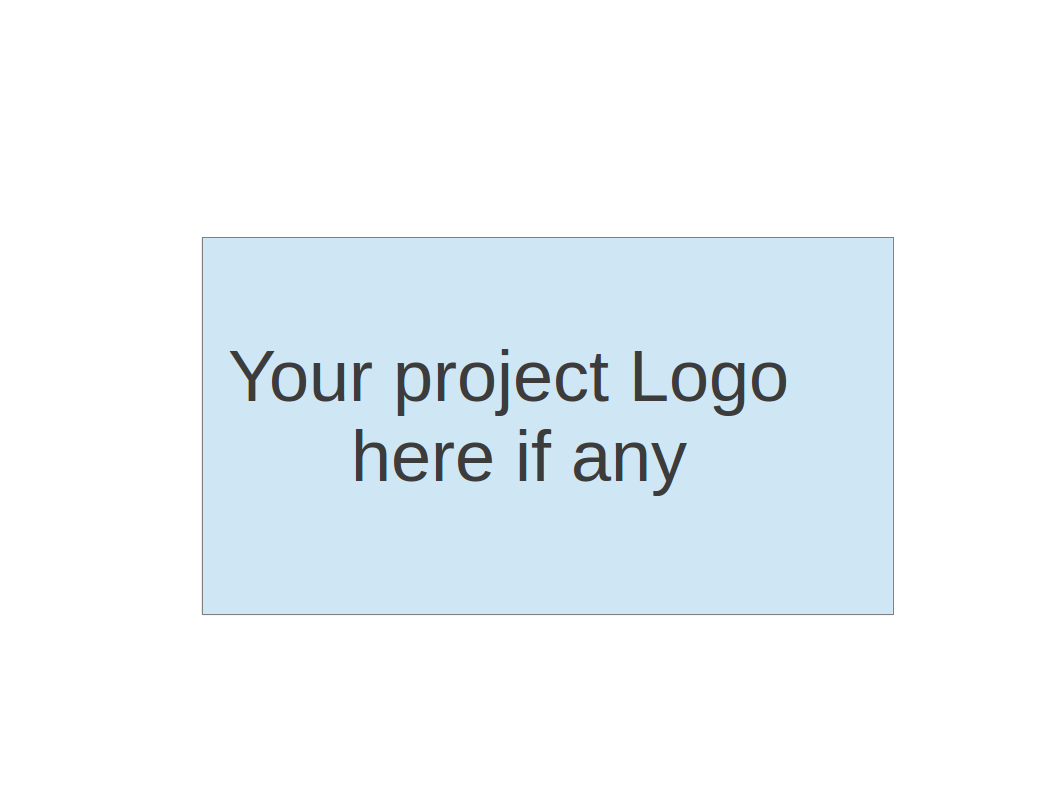
**CS673S16 Software Engineering** 

**Team 3 - Project Name**

**Project Proposal and Planning**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Oliver Nowak | Team Lead | O No | 1/21/2016 |
| William Baldoumas | Environment & Integration | WB | 01/21/2017 |
| Ashutosh Singh | Design | AS | 2/2/2017 |
| Michael Song | Security | MS | 2/2/2017 |
| Emily Birmingham | QA | EB | 2/8/2017 |
| Mike Trinh | Configuration, Requirements | MT | 2/15/2017 |
| Son Nguyen | QA | SN | 02/16/2017 |
| Betsy Berkey | Implementation | BB | 3/12/2017 |
|  |  |  |  |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Overview

Related Work

Detailed Description

[Management Plan](#_2et92p0)

[Process Model](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.27177f40uci)

[Risk Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.a4oqwntk3mw)

[Monitoring and Controlling Mechanism](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.ywdoc2clc9yt)

[Schedule and deadline](#_1t3h5sf)

[Quality Assurance Plan](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.72e1f4uawy2r)

Metrics

[Standard](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.vc72k6dweldv)

[Inspection/Review Process](#_35nkun2)

[Testing](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.r5d5mhtlf0kq)

[Defect Management](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.54a4wuncjg1c)

Process improvement process

[Configuration Management Plan](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.hw41vg4ykxen)

[Configuration items and tools](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.bwlb4d4vdox2)

[code commit guidelines](#_4i7ojhp)

[References](https://docs.google.com/document/d/107bVcXdAG-ogRr90PquFB8-aWGvTwSua8pu_O4Kmz6c/edit#heading=h.8mva2050iy7t)

[Glossary](#_1ci93xb)

# Overview (Ashu)

Our motivation behind this project is to have a web app that can streamline the process of team communication and help increase the efficiency and productivity of the team. (real motivation being getting good grades). The target user base for this web app are working professionals, small / medium sized businesses who need a free software for team collaboration. In addition to the professional use it can also be used for personal use.

**Communication Tool existing features:**

1. Users can log in to the app using their username and password.
2. Users can create rooms and send messages within rooms.
3. All users who are logged into the app are able to view all rooms and all messages.
4. The sidebar displays all rooms as well as the users that are currently logged in.
5. Users can upload files.
6. Users can send emoticons in their messages.
7. Users can search for messages using a keyword or phrase. They will receive information about the messages that contain that keyword or phrase.

# Related Work (Michael)

Our main focus is to build a web-based chat system that facilitates communication in the form of text. Slack may be one of the similar software system.

**The essential functions of Slack:**

1. Group communication in discussion forums, called channels, but also allows one-on-one communication via private message as well as chat and data transfer, including file sharing.
2. All content inside Slack is searchable, including files, conversations, and people. (Contextual Search)
3. All communications are always in Sync.
4. Mac, Windows, Android and iOS apps available.
5. Open API for users to build own integrations
6. All data transfer is encrypted.
7. Desktop & Mobile messaging
8. Real time notification allowed.
9. Tags, keywords & @mentions.
10. Audio/Video conferencing
11. Personalized profiles.
12. Slack integrates with a large number of third-party services and supports community-built integrations. Nearly 100 integrations.
13. Allow communities, groups, or teams to join through a specific URL or invitation sent by a team admin or owner.

**Essential features that we are different from Slack:**

1. Group communication in discussion forums, called channels, but also allows one-on-one communication via private message. We will also include file sharing feature in our system.
2. We do not involve the search function in our system.
3. Our chatting system is available on website. Not on mobile devices.
4. We do not have tags. We have @ mentions.
5. We do not do audio/video conferencing.
6. Allow communities, groups, or teams to join through a specific URL or invitation sent by a team admin or owner.
7. Our application requires BU email to access.

# Proposed High level Requirements (Mike)

Functional Requirements

* + - **Essential Features**
* Chat tool works without console errors.
* Grammar and English usage of user-facing text is correct.
* Refactor the REST API url structure
* Require BU email to access Communication Tool.
* Password requirements - at least 8 characters using at least one uppercase letter, one lowercase letter, and one number.
* Users can invite other users to rooms and groups.
* Users receive notification of room/group invite.
* Users can create groups within rooms and invite other users. Messages in those groups can be viewed by invited users.
* Messages written to a room or group remain for a month unless the limit of 1,000 messages is reached. Then, the older ones will be deleted first.
* The sidebar shows groups that users have been invited to - within the room that they’re in.
* Create documentation that clearly defines all steps to install app
  + - **Desirable Features**
* Integration with other apps in the suite - Allow users working within the RequirementTracker or IssueTracker apps to send messages directly to the CommunicationTool.
* Users can mark their messages as bold or code using special symbols.
  + - **Optional Features**
* Improve website with a more distinct color scheme and a more liquid design.
* Use BU’s authentication process to verify users.

Nonfunctional Requirements

* Remove folders and files from directory structure that are not used
* Dependency of “home” directory for building/installing code base is removed
* Documentation for how to add a “test” user added.

Implemented Features (to be completed at the end of project)

# Management Plan (Michael, Ashu, Son)

## Process Model

* + Agile via weekly Scrum meetings.
  + Scrum meetings should be ~15mins with answers to the following 3 questions:
    - What work has been completed?
    - What work remains?
    - Is the work progress blocked ?
  + Longer conversations (especially around “blocked” items) should occur after the ~15min meeting with the relevant people
  + At the beginning of every iteration there will be a “retrospective” meeting. This will be about figuring out what processes worked; what processes didn’t work; how can we improve; what issues came up.
  + The team will vote on one “issue” from the retrospective that we will try to improve on for the next iteration.
  + At the end of every iteration, we will have a demo of features by the team.
  + Towards the end of every iteration, the requirements lead and team lead will need to plan user stories for the following iteration, based on the team’s progress and sprint velocity.
  + Code that is ready to be merged will need an approved “Pull Request” via Github.

## Objectives and Priorities

* + Communication is priority #1. We can’t fix what we don’t know is broken. ;)
  + Provide “professional” software engineering experience
  + Provide experience for working as a team on a common set of problems

## Risk Management

All project involve a degree of risk. These are issues that can potentially cause problems such as a delay of the schedule or increased project costs.

We try to confront risks as soon as possible rather than waiting for them to confront us in the process of building the application.

Our risk management includes the following tasks:

* Identify risks and their triggers
* Classify and prioritize all risks
* Craft a plan that links each risk to a mitigation
* Monitor for risk triggers during the project
* Implement the mitigating action if any risk materializes
* Communicate risk status throughout project

Our Monitoring includes:

* Update SPPP reports and meeting minutes and include risk management issues
* Revise risk plans according to any major changes in project schedule
* Review and reprioritize risks, eliminating those with lowest probability

Mitigating options include:

* Accept: Acknowledge that a risk is impacting the project. Make an explicit decision to accept the risk without any changes to the project. Project management approval is mandatory here.
* Avoid: Adjust project scope, schedule, or constraints to minimize the effects of the risk.
* Control: Take action to minimize the impact or reduce the intensification of the risk.
* Transfer: Implement an organizational shift in accountability, responsibility, or authority to other stakeholders that will accept the risk.
* Continue Monitoring: Often suitable for low-impact risks, monitor the project environment for potentially increasing impact of the risk.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RiskId** | **Risk** | **Probability** | **Impact** | **Action** | **Result** |
| 1 | None of us knew the required technology (Node & Django) | Frequent | Critical | Do tutorials.  Scrum Meeting.  Pair Programming.  (Accept) | Risk mitigated significantly |
| 2 | Someone dropped the class | Occasional | Critical | Change Management  Pair Programming  (Transfer) | Risk mitigated significantly |
| 3 | Deep diving in the previous code | Frequent | Critical | File system and Infrastructure  (Control) | Results are hard to evaluate |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **RiskId** | **Risk** | **Probability** | **Impact** | **Action** | **Result** |
| 4 | Merge Conflicts | Frequent | Critical | Pair Programming.  Oliver did quadruple work.  (Continue Monitoring) | Risk mitigated significantly |
| 5 | Too much user stories to handle | Frequent | Marginal | Prioritization  (Control,  Avoid) | Risk still presented |
| 6 | Team member missing scrum meetings | Occasional | Marginal | Extra meetings in sub-teams  (Continue Monitoring) | Results are hard to evaluate |

Other implementations we have included in our project are:

|  |  |
| --- | --- |
| **Implementation** | **Risk mitigated** |
| Documentations (SPPP, SDD, Meeting Minutes, Testing, Architecture Overview) | Responsibility and delegation  Process monitoring |
| Sub-team meetings | Overriding works  Schedule estimation |
| Iteration Presentation and Review | Process monitoring  Schedule estimation |

Not every risk can be dealt with earlier than its natural occurrence. However, we were doing proactive planning for our software project via risk management. The top-ranking risks were planned and mitigated. All risks were monitored. And it is important for us to focus on our project’s critical success factors while keeping an eye on its risk factors.

## Monitoring and Controlling Mechanism

* + Use Pivotal Tracker software to track story progress and sprint velocity
  + Set up a Continuous Integration pipeline for running unit/functional tests upon checking

## Schedule and deadlines (need update constantly)

* + Iterations are the unit of time for a set of features
    - Iteration 0 - 2/16/2017
    - Iteration 1 - 3/16/2017
    - Iteration 2 - 4/6/2017
    - Iteration 3 - 4/27/2017

# Quality Assurance Plan (Son and Emily)

## Metrics

* **Product metrics**
* Purpose: to keep track of the website usage, user patterns, etc
* Possible tool for front end:
  + Google analytics: using Google Analytics to have it track accesses and devise graphical analysis
* Basic log file system: store number of files, number of classes, time spent to keep track of project size and manpower

* Process improvement metrics  
  Some possible metrics that help record team’s process:
* Actual stories completed / committed stories
* Technical debt: number of bugs found / number of bugs fixed
* Retrospective process improvement: ability to revise development process to make it more effective for next iteration / sprint

Reference:<http://pragmaticmarketing.com/resources/9-scrum-metrics-to-keep-your-team-on-track>

## Standard

## Variable names descriptive and concise

## single quotations for string

## Comments on all methods

## Lines no longer than 100 characters

## Braces always follow if statements, etc?

## Comments on every method

## files should have a header

## File

## Date

## Author

## purpose

* <https://github.com/airbnb/javascript>
* <https://make.wordpress.org/core/handbook/best-practices/coding-standards/javascript/>

## 

## Inspection/Review Process

* Using Github pull request or maybe Reviewboard

The review process:

* Requirement inspection: at least once every iteration / sprint, a requirement inspection review is required to verify requirements priorities, checking with team 2 for integrations plan and avoid any duplication.
* Design inspection: general designs and approaches for each feature should be report to the team leader / team sub leader before actual implementation.
* Code review: a code review must be requested before actual commit to the master branch.
  + Less than 200 lines at a time: commit often to your branches. (Easier for reviewers, and able to fix bugs early)
  + The committing reviews should include at least implementation leader, and maybe the team leader. Only do a master commit when a “ship it!” is granted.
  + Reviewer must record inspection rate and defect rate.
  + Review submitter must record defect-fixing time.

## Testing

* Unit testing is required for each method
* At least one integration testing is required for each feature
* All tests must be passed before any commit

Reference:<http://alistapart.com/article/writing-testable-javascript>

## Defect Management

* Report defect immediately when it’s found, put defects on backlog and handle them the same way as user stories.
* Priority of the defect must be more severe for on-going defects in master to minor defects in feature branches.
* Integration test is needed to make sure the defect is completely fixed.
* Tests must be rerun after fixing defects to ensure consistency.

References:<https://www.agilecockpit.com/wp-content/uploads/2014/11/Whitepaper_Managing_Defects_in_an_Agile_environment.pdf>

# Configuration Management Plan (William)

## Configuration items and tools

* + - We will be using github on a virtual machine running an Ubuntu distribution of Linux as our version control system. Both GUI and command-line use of git is acceptable. Everyone will also be using Sublime Text to do their coding in order to keep a unified file system structure and uniform line endings.

## Change management and branch management

* + - Team members should branch off of our updated master branch for each new feature that is to be implemented. In order to avoid merge conflicts, team members should pull the most recent code from the develop branch often, and merge it into their working code, either by rebasing or by merging manually.

## Code commit guidelines

* + - Team members may commit to their feature branches as needed, but code will need to be reviewed and marked with a “ship-it” before pushing to the develop branch. The master branch will be saved for deployments.

# References

(For more detail, please refer to encounter example in the book or the software version of the documents posted on blackboard. )

# Glossary