# CS673S16 Software Engineering
# Team 1 - FriendZone
# Project Proposal and Planning

Your project Logo here if any

| Team Member | Role(s) | Signature | Date |
|---|---|---|---|
| Ed Orsini | Team Leader & Configuration Leader | *Ed Orsini* | 2/16/2017 |
| Cory Stone | QA Leader | *Cory Stone* | 2/16/2017 |
| Michael Eskowitz | Security Leader | *Michael Eskowitz* | 2/16/2017 |
| Robert Gomez | Environment and Integration Leader | *Robert Gomez* | 2/16/2017 |
| Arpita Vats | Requirement Leader | *Arpita Vats* | 2/16/2017 |
| Ravi K Rajendran | Design Leader | *Ravi K Rajendran* | 2/16/2017 |
| Nick Hattabaugh | Implementation Leader | *Nick Hattabaugh* | 2/16/2017 |
| | | | |
| | | | |

## Revision history

| Version | Author | Date | Change |
|---|---|---|---|
| 1 | **Ed Orsini, Cory Stone, Michael Eskowitz, Robert Gomez, Nick Hattabaugh** | 2/16/2017 | **Initial document** |
| | | | |
| | | | |
| | | | |

# 1. Overview (Nick)

Social networking websites have been around since the early days of the internet. In recent years they have seen explosive growth to the point where Facebook has become the third most popular website in the world with 1.86 billion active users. The purpose of our software system is to capture a fraction of this user base by providing a unique social networking experience. Our application is not being developed to target a specific user group and, as such, the potential user will be any computer friendly individual who wishes to engage both friends and family in a social experience.

(motivation, the purpose and the potential user of the software system etc. )

# 2. Related Work (Nick)

There have been many social networking websites that have been popular over the years. The very earliest social networking website was Geocities which allowed users to create their own website and grouped the sites into “cities” based on the site’s content. The next generation of social sites like America Online developed instant messaging applications which allowed users to

communicate with their friends in addition to providing a platform for creating their own profile and website. Many social websites followed and were often targeted towards various niches like Classmates.com or specific ethnic groups like Asian Avenue. Modern social networking websites still resemble these platforms.

LinkedIn is a popular social site devoted to career building and networking. On this site users are able to create a virtual resume which lists their employment history, educational background and any references that they might have. Users on LinkedIn are able to add friends and coworkers to their network and then can contact people that their contacts know. Companies on this site can post job openings that need to be filled and job hunters can search for companies that are looking to hire.

Another niche type social networking site is OkCupid. This website is primarily devoted to dating and helping users make personal connections. Each user is able to create a profile listing the standard attributes such as their age, height, weight and ethnicity. Further, users are able to differentiate from one another by writing a description of themselves, their life and what they are looking for. Most importantly, however, they can also post pictures that users can look at to determine if they are interested in that person. If one user finds another to be desirable they are able to message them and potentially strike-up a romance.

The largest social networking site is Facebook. Facebook is the third most popular website in the world. It is more generic than the social networking sites we have described as it is not devoted to dating or career building. The purpose of facebook is to keep in touch with your friends and family. Towards this end a Facebook user creates an account that is based on their real name and can upload photos and post events or messages to their timeline. The timeline is a historical chronology of the user's life. As with all modern social networking sites, Facebook includes instant messaging capabilities so that the user can message friends and family.

The social networking platform that we are developing will be generic in focus similar to Facebook. It is our goal to provide a platform for user profile creation and to allow users to interact by messaging one another and posting to their walls. Our website has a casual atmosphere catering to User's social lives and hopes to foster and improve their current social lives. Our mission is to make it easier to connect with your existing friends but also provide an easy way to expand your social circle.

(describe any similar software systems, and the difference from them)

# 3. Proposed High Level Requirements (Cory)

a. Functional Requirements
    i. Essential Features
        1. Profile
            a. Users have a profile where they can add information about themselves:
                i. First name
                ii. Middle name
                iii. Last name
                iv. "About me"

v. DOB
vi. Email address
vii. Profile image

b. Users can view other users' profiles

2. Activity feed:
    a. Show recent posts from friends
    b. Show recent image uploads from friends
    c. Show recent birthdays of friends
3. Friends:
    a. Users can add another user as a friend
    b. Users can remove a friend
4. Images:
    a. Users can upload images to their photo album
    b. Users can remove images from their photo album
    c. Users can designate one of the images in their album as their profile picture.
    d. Users can view other users' photo albums
5. Wall:
    a. Users can add posts to their wall
    b. Users can add posts to others' walls
    c. Users can comment on posts
6. Chat:
    a. Users can send private messages to each other
7. Search:
    a. Users can search for other users by name
    b. Users can search for other users by email
8. Security:
    a. Users can log into app using Facebook OAuth
    b. Users can log into app using Google OAuth

ii. Desirable Features
1. Chat:
    a. Users can have a real-time chat with other users
2. Privacy:
    a. Users can designate profile elements as friends-only
    b. Users can designate posts as friends-only
    c. Users can designate images as friends-only
3. Invitations:
    a. Send an email inviting a friend to sign up and use the app

iii. Optional Features
1. Interests:
    a. Users can tag their profile with interests
    b. Searching for an interest will show users with that interest
2. Introduction guide:

        a. Tutorial with popups explaining how to use the app
    3. Dating feature:
        a. Have a component of the app geared towards dating
b. Nonfunctional Requirements
    i. Application should support latest version of major web browsers
    ii. Application layout and functionality should be familiar to social media users
c. Implemented Features (to be completed at the end of project)

# 4. Management Plan (Rob)

(For more detail, please refer to SPMP document for encounter example)

## a. Process Model

We are implementing the Agile methodology in our project. We will have 2 meetings a week to discuss our current status and to identify any issues that may occur and handle them early. We will demo our software to each other as soon as it is complete so we can provide feedback and make sure all of our different features come together well as a whole. After each iteration we will have a retrospective meeting where we will discuss what went well and what didn't go so well in the last iteration, this way we will be able to improve our software development process as we go.

## b. Objectives and Priorities

Our main objective is to be able to complete the core features we laid out for our project. Along with that we want to make sure this is a good learning experience for those who are not familiar with the MEAN stack or with this software development process. If we finish our features with time to spare we will look to implement some additional features that we agreed would be nice to have.

## c. Risk Management (need update constantly)

Our meetings will be crucial in identifying anything that could set our project back, so communication and openness is a must. We will use our slack channel to alert the other teammates of news that could affect meeting a deliverable or affect their current feature work. We will create multiple feature branches to work on to avoid any erasing of code or breaking of the build because of incompatible code. We will have code review when submitting a merge request to avoid any obvious flaws in code. Committed code should be covered and have passed Unit Tests. QA will then test the code as soon as it is committed.

## d. Monitoring and Controlling Mechanism

Our main tool to monitor the status of current iterations will be pivotal tracker. Our meetings will also be used in keeping track of everyone's current status and be used

to identify any delays in our current iteration. Finally, our git merging process will help keep a constant quality of code throughout the project.

### e. Schedule and deadlines (need update constantly)

Deadlines for our project

Iteration 0 - 2/16/2017
Iteration 1 - 3/16/2017
Iteration 2 - 4/6/2017
Iteration 3 - 4/27/2017

## 5. Quality Assurance Plan (Cory)

(For more detail, please refer to SQAP document for encounter example)

### a. Metrics

i. Definition
   1. Product Metrics
      a. Size: Number of lines of code (LOC).
      b. Defect density: Number of defects per KLOC.
   2. Process Metrics
      a. Story points: Number of story points completed during an iteration, number story points planned for the iteration that were not completed.
      b. Defect detection rate: Number of defects detected during an iteration.
      c. Defect fix rate: Number of defects fixed during an iteration.
      d. Cost: Time spent on testing and defect resolution compared to overall time spent on project.

ii. Results (to be completed at the end of each iteration)

### b. Standard

i. Documentation: We will use the document outlines provided for this course.
   1. Information about the project and its planning will be documented in this document.
   2. Product design details will be documented in the Software Design Document (SDD).
   3. Test results will be documented in the Tests Report document (Testing).

ii. Coding:
   1. Code style: We will not have strict enforcement of a particular code style, but recommend that each developer try to follow the style guides for HTML, CSS, and JS presented by W3Schools

(http://www.w3schools.com/).

2. Code documentation: As much as possible, code should have descriptive naming, clear syntax, and be structured in such a way as to make it self-documenting (https://en.wikipedia.org/wiki/Self-documenting_code).

## c. Inspection/Review Process

i. Requirements will be decided upon and reviewed by the team as a whole.
ii. Architecture designs will be reviewed by the Design leader.
iii. Test plans will be reviewed by the QA leader.
iv. Code reviews will be performed by the Configuration leader or another member of the team (see section 6b).

## d. Testing

i. Continuous Integration and Deployment (CICD): The team will use Jenkins (https://jenkins.io/) for automated build, testing, and deployment of code. The system will monitor our code repository and trigger a new build whenever code is committed to the master branch. For each build we will record unit test results, automated system test results, and code coverage. The QA leader will work with the Configuration and Implementation leaders to set up this system.
ii. Unit testing: Each developer should include unit tests for their JavaScript code. We will use Mocha (http://mochajs.org/), Chai (http://chaijs.com/), and Sinon (http://sinonjs.org/) for unit testing. Code coverage will be calculated using Istanbul (http://gotwarlost.github.io/istanbul/).
iii. System testing: After the work for a story is deployed, it will be tested by a team member who was not involved in the development. The QA leader will add tests to the test plan for each story to be completed during an iteration. Before the end of an iteration, team members will execute tests in the test plan that have not been automated, and results will be recorded in the Testing document.
    1. Automated system testing: System testing will be automated as much as possible using either the WebDriverJS implementation of Selenium (http://www.seleniumhq.org/) or WebDriver.io (http://webdriver.io/) (TBD). This will be driven primarily by the QA leader, with contributions from other team members.
iv. Acceptance testing: A subset of the system tests will be designated as acceptance tests. These tests must pass for the product to be accepted (i.e. ready for demo at end of iteration).

See the Testing document for test results of each iteration.

## e. Defect Management

i. Criteria: A defect is any error in functionality, behavior that doesn't meet requirements, or behavior that interferes with product usability.
ii. Severity: Defect severity will be categorized as:
    1. Critical: Renders the product unusable (e.g. application crash).
    2. Serious: Causes a requirement to be unmet (e.g. broken feature).
    3. Trivial: Minor issue that does not impede product usability (e.g. cosmetic issues).
    4. Medium: Between serious and trivial (e.g. extraneous behavior)
iii. Priority: Defects will be prioritized as High, Medium, Low.
iv. Defect management: We will use Pivotal Tracker (https://www.pivotaltracker.com) to track defects. Team members will create Bugs in the Icebox as defects are detected. During iteration planning (and mid-iteration if required) the team will triage new defects and decide on severity and priority before moving to the backlog. Labels will be used to indicate the severity and priority of a Bug.
v. Defect correction: Defects will be assigned to developers along with stories during iteration planning. Critical defects may be brought into the iteration scope after planning if necessary.

# 6. Configuration Management Plan (Ed)

(For more detail, please refer to SCMP document for encounter example)

## a. Configuration items and tools

i. ***Git*** - as a version control system (VCS) for tracking changes in our files and coordinating work with our team members
ii. ***GitHub*** - online project hosting using Git. Includes source-code browser, in-line editing, wikis, and ticketing.
iii. ***Command line*** - Although there are plugins and tools for working with Git and IDEs, we'll be working with Git through the command line.

## b. Change management and branch management

i. The team has decided for team members to create and work on feature-related branches that they are responsible for. The team member that is responsible for a particular feature will create a new branch to work on such feature. Commits will be made as often as possible to ensure that all work is saved. When the team member working on the branch determines that the work is satisfactory, the same team member will create a Pull Request. The Pull Request will indicate to the configuration leader that the branch is ready to be merged into the master branch. At this point, the configuration leader will review the proposed changes, test, and finally merge the code if all tests pass successfully. The configuration leader may ask other team members to also perform a code review when needed. When the code merge is complete, the configuration leader will alert the team

members. Finally, a final test of the master branch will be done. This test will include identifying any regressions, if any exist. If any issues are found, the configuration leader will alert the owner of the branch and work with team members to get the issues resolved before merging the code.

### c. Code commit guidelines

i. As stated above, the team members are encouraged to commit to their branches as often as possible to reduce the loss of work. However, team members will need to create a Pull Request in order for their code to make it to the master branch. The Pull Request will trigger a brief code review by team members where the features will also be tested.

## 7. References (for all of us to use if needed)

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

W3Schools - http://www.w3schools.com/
Self-Documenting Code on Wikipedia - https://en.wikipedia.org/wiki/Self-documenting_code
Mocha JavaScript testing framework - http://mochajs.org/
Chai JavaScript assertion framework - http://chaijs.com/
Sinon JavaScript mocking framework - http://sinonjs.org/
Istanbul code coverage tool - http://gotwarlost.github.io/istanbul/
Selenium browser automation - http://www.seleniumhq.org/
WebDriver.io browser automation - http://webdriver.io/
Jenkins continuous integration - https://jenkins.io/
Project on Pivotal Tracker - https://www.pivotaltracker.com/n/projects/1957973
Project on GitHub - https://github.com/edorsini/social-networking-app

## 8. Glossary (for all of us to use if needed)

Git - https://en.wikipedia.org/wiki/**Git**
GitHub - https://**github**.com/
KLOC - 1000 lines of code
Story - One unit of work to be completed by a developer towards delivering a feature.
Story Points - An estimation of complexity for a story. Helps in estimating what work can be completed in an iteration.
Velocity - The number of story points completed in an interation. Helps in planning work for the next iteration.