**CS673F16 Software Engineering** 

**Team 2 - MeetCute**

**Project Proposal and Planning**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Daniel [Shih](mailto:Shih,shih0623@bu.edu) | Team Leader | *Huai Chun Shih* | Oct.1 2016 |
| Rebecca Bell | Env. & Integ. Lead | *Rebecca Bell* | Oct 2, 2016 |
| Bingliang Weng | Requirement Leader | *Bingliang Weng* | 10/1/2016 |
| Gabriel Leake | Security Leader | *Gabriel Leake* | 10/1/2016 |
| Ian Klarman | QA Leader | *Ian Klarman* | 10/2/2016 |
| Adiba Nisa | Design Leader | *Adiba Nisa* | 10/2/2016 |
| Gautam Bhat | Configuration Lead | *Gautam Bhat* | 10/2/2016 |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Change** |
| **0.1** | **cooperating** | **Oct.2 2016** | **initial** |
| **0.2** | **Update Management Plan** | **Oct.1 2016** | **GL** |
| **0.3** | **Update Nonfunctional Reqs** | **Oct 11 2016** | **BB** |
| **0.4** | **Update Implemented Features, Risks** | **Dec 19, 16** | **BB** |
| **0.5** | **Update Github branch management** | **Dec.19 16** | **DS** |

# Contents

[Overview -AN](#_g6igqliy7rm)

[Related Work -DS](#_bf21eadgjj29)

[Proposed High level Requirements - BB](#_rgyo4hi9stmq)

[Management Plan -GL](#_ts358bsdtbcv)

[(For more detail, please refer to SPMP document for encounter example)](#_ds8oyr75pnh1)

[Process Model](#_p6g30deor1wd)

[Objectives and Priorities](#_q05un4fx7l1f)

[Risk Management (need update constantly)](#_5wrnthwbcgjy)

[Monitoring and Controlling Mechanism](#_bbdc8y3y417w)

[Schedule and deadlines (need update constantly)](#_wor2td6kkr7l)

[Quality Assurance Plan -BW](#_f4z84tz7qrf3)

[Configuration Management Plan -GT](#_j5uvivmxqcsp)

[Configuration items and tools](#_qvwj1x7w7phd)

[Change management and branch management](#_911z5t5h2n3a)

[Code commit guidelines](#_pm523od0l757)

[References -EN](#_pd9euov6m4du)

[Glossary ( terminology )](#_ty3i2nqffhtc)

[Overview](#_g6igqliy7rm)

[Related Work](#_bf21eadgjj29)

[Detailed Description](#_rgyo4hi9stmq)

[Management Plan](#_ts358bsdtbcv)

[Process Model](#_p6g30deor1wd)

[Risk Management](#_5wrnthwbcgjy)

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

[Schedule and deadline](#_wor2td6kkr7l)

[Quality Assurance Plan](#_2qmny2mv8poy)

[Metrics](#_vwjduhc9wuah)

[Standard](#_jqsecra47e7v)

[Inspection/Review Process](#_d2l6hlyvabsu)

[Testing](#_l9xnpmd6hh0y)

[Defect Management](#_5amsh8h9f0c7)

[Process improvement process](#_jhct37ebxxpn)

[Configuration Management Plan](#_j5uvivmxqcsp)

[Configuration items and tools](#_qvwj1x7w7phd)

[code commit guidelines](#_pm523od0l757)

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

[Glossary](#_ty3i2nqffhtc)

# Overview -AN

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

Motivation and Purpose:

1. Because they have become the most common way to interact with single people.
2. Learning software engineering. It contains good features like CRUD operation to practice.
3. It is interesting & social with a variety of features.
4. Other dating sites are lacking compared to what we plan to offer

potential user :

single people(usually anyone between the age of 22-35)

# Related Work -DS

|  |  |  |
| --- | --- | --- |
|  | MeetCute (our app) | OkCupid |
| Applications | Responsive Web App for all browsers. | Non-Responsive Web App  IOS & Android App |
| Client side framework | React.js | React.js / Native mobile app  Web app use React, a similar approach, but not Responsive. |
| Server side framework | Node.js (JavaScript)  Runs fast.  Easy to implement. | OKWS (C++)  Runs faster a little, but harder to implement. |
| Database | MongoDB (non-SQL)  Scalable, easy to use, but not as mutual as SQL. It is good enough for a dating site but not a banking account.etc. | SQL  More mutual and stable. |
| System Design | MVC (bound by Meteor.js)  Developing is a way faster. | MVC |
| Product Design | Mobile-First, Responsive,  Follow Google’s Material Design.  Single-page-web-applications  (Rendering faster, better UX, like Gmail)  Latency compensation  (app changes instantly without waiting for http response)  Simple & Easy to create an account and get started | Not sure. I guess it was designed by pure engineers. It needs better designers. |
| Functionality | Provides real time chatting with a user  Provides single sign in only | Provides messaging (i.e. email)  Allows for other types of registration besides Facebook |

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

# Proposed High level Requirements - BB

* 1. Functional Requirements
     1. Essential Features
        1. Ability to register an account (sign up)
        2. Ability to login and logout
        3. Profile page creation
           1. Profile Edit
        4. “Discover” profile list page
        5. Chat Functionality
     2. Desirable Features
        1. Log in with Facebook
        2. Optimized for mobile view / Responsive
     3. Optional Features
        1. Profile Search
        2. Ability to favorite or friend other profiles
  2. Nonfunctional Requirements
     1. User should be able to view the app using different browsers (Chrome, Safari, Firefox)
     2. Performance Optimization
        1. Implement caching so image-heavy Discover page loads quickly upon repeated viewing
  3. Implemented Features
     1. Ability to register an account (sign up)
     2. Ability to login and logout
     3. Profile page creation
     4. Profile Edit
     5. “Discover” profile list page
     6. Chat Functionality
     7. Emoji support
     8. Log in with Facebook
     9. Optimized for mobile view / Responsive

# Management Plan -GL

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

## Process Model

* + 1. We are following an Agile style process:
       1. 3 week iterations
       2. Weekly standup (cannot be daily due to time constraints)
       3. Retrospective/lessons learned at the end of each sprint
       4. Weekly planning phase (end of class)
       5. Using Pivotal Tracker to manage and assign tasks

## Objectives and Priorities

* + 1. Learn new JavaScript framework - MeteorJS
    2. Build responsive, mobile-friendly CRUD webapp
       1. Due to the time constraints and new technology, our focus will be on core features, not a lot of frills, and attempting to build a quality/stable app.

## Risk Management (need update constantly)

* + 1. Knowledge risks:
       1. JavaScript
          1. Med due to paired programming
       2. Git/Github
          1. Med due to tutorial availability, but not enough time for tutorials
       3. Meteor framework
          1. Med - availability of examples / literature but short on time
       4. Complex features in a too short time frame
          1. Medium - Will need to manage scope to prevent feature creep
    2. Schedule risks
       1. Some group members are either remote, in multiple classes & conflicts due to midterms/work,etc..
          1. Med - Mitigated by use of online tools for meetings / communication, and collaboration via internet.

## Monitoring and Controlling Mechanism

* + 1. Slack communication (#team-2 channel)
    2. Use of Pivotal Tracker to create User Stories and file Bugs
    3. Github code reviews
    4. Everyone will ramp up to a standard level by doing Meteor website’s tutorials, in addition we are doing pair programming. This will allow for multiple resources to be available.
    5. If someone is sick or does not complete their tasks, this will be caught at the weekly meeting and then the upcoming iterations tasks will be adjusted accordingly.

## Schedule and deadlines (need update constantly)

* + 1. Meetings:
       1. Weekly - Sunday @ 2pm - Updates & pair programming
       2. Adhoc - Scheduled based on workload & availability
    2. Deadlines
       1. Sunday 12/11 - end of Iteration 3
          1. Complete testing
          2. Finalize SPPP and SDD
       2. Sunday 12/4 - no specific deadlines
       3. Sunday 11/20 - no specific deadlines
       4. Sunday 11/13 - end of Iteration 2
       5. Sunday 11/6 - Finish individual coding task
       6. Sunday 10/30 - Finalize coding tasks
       7. Sunday 10/23 - end of Iteration 1
          1. Update SPPP
          2. Powerpoint presentation for retro for iteration 1 (coding)
          3. Do more Meteor/Git tutorials
       8. Sunday 10/16
          1. Finish UML diagram
          2. Finish React/Meteor tutorials
          3. Finalize Requirements
       9. Monday 10/10 - Complete:
          1. Everyone have skeleton app running locally
       10. Sunday 10/2 - Complete:
           1. SPPP
           2. Powerpoint presentation for retro
           3. Meteor/Git tutorials
           4. Workspace setup

# Quality Assurance Plan -BW

Please see separate document:

<https://docs.google.com/document/d/1x_obTiRuhoH1Cy5bERJ-zziVA3EUDDmeD3xTc40GMQk/>

# Configuration Management Plan -GT

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

## Configuration items and tools

* + 1. Git/Github
    2. Google Docs
    3. Pivotal Tracker

## Change management and branch management

* + 1. Change Management
       1. Changes can be made when adding new features/functionality, or after detecting a bug in the code.
       2. Before making change, approval of team leader and story leader should be sought.
    2. Branch Management - Branches
       1. Master: Master branch for project, original release branch.
       2. Subgroup branches: Each module/functionality/member working on these would fork out separate branches to work on. The format should be [name]-task\_name. e.g. “ds-profile\_card” , “ds” is the name of Daniel Shih. “profile\_card” is the task for this branch.
       3. Incomplete code, or code not approved by respective leaders, should not be merged with master.
       4. product branch: steady version for product

## Code commit guidelines

* + 1. Only commit code when a functionality/module is fully working, or someone else’s work depends on it.
    2. It is required to have your work and changes approved by another team member, before proceeding to commit.
    3. Always commit changes with appropriate messages describing the changes made with respect to the previous version of code/documents. Include your initials along with commit message.
    4. Committed code should be neatly indented, and commented properly for other team members to understand.

# References -EN

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

Product defects. Retrieved Oct 1, 2016 from mbalib Web site:

“产品缺陷.” - MBA智库百科, http://wiki.mbalib.com/wiki/%e4%ba%a7%e5%93%81%e7%bc%ba%e9%99%b7.

Alan Zhou. Software Defects Classification Standard. Retrieved Oct 1, 2016 from Google search Web site:  
http://read.pudn.com/downloads12/ebook/49472/%E8%BD%AF%E4%BB%B6%E7%BC%BA%E9%99%B7%E5%88%86%E7%B1%BB%E6%A0%87%E5%87%86.pdf  
   
Methods of Software Test. Retrieved Oct 1, 2016 from Baidu baike Web site:

“软件测试方法.” \_百度百科, http://baike.baidu.com/view/3617317.htm.

xjbclz. (2016, June). Document Standard of Software Development. Retrieved Oct 1, 2016 from CSDN blog Web site:  
<http://blog.csdn.net/xjbclz/article/details/51586586>

* “OkCupid.” Wikipedia, Wikimedia Foundation, https://en.wikipedia.org/wiki/okcupid.
* Okcupid Careers https://www.okcupid.com/careers.
* Software Configuration Management Plan (SCMP) for Encounter version 1.0
* Software Project Management Plan (SPMP) for Encounter version 1.2
* Software Quality Assurance Plan (SQAP) for Encounter version 1.0
* A Survey of Agile Development Methodologies, Laurie Williams, 2007
* Security:
  + https://www.youtube.com/watch?v=xAdwQ4UAoj0
  + https://atmospherejs.com/meteor/insecure
  + https://www.discovermeteor.com/blog/meteor-and-security/