MET CS 673 S18 Software Engineering
Team 3 - The Wish List
Project Proposal and Planning

Team Member	Role(s)	Signature	Date
Duncan Morrissey	Project Leader, Configuration Leader	*Duncan Morrissey*	3/1/2018
Edward Ryan	Design Leader, QA Leader	*Edward Ryan*	3/1/2018
Yiannis Karavas	Requirements and Implementation Leader	*Yiannis Karavas*	3/1/2018
Zach Lister	Backup Project Leader, Management Plan Leader	*Zach Lister*	3/1/2018
Ben Mitchell	Security leader	*Benjamin Mitchell*	3/1/2018

Revision history

Version	**Author**	**Date**	**Change**
2.0	**Duncan, Edward, Yiannis, Zach, Ben**	**3/1/2018**	**Accepting all changes.**
1.2	**Edward**	**2/28/2018**	**Minor updates to QA plan to point to testing documents**
1.1	**Yiannis, Duncan**	**2/15/2018**	**Added a few**

			function and non functional requirements
1.0	**Duncan, Edward, Yiannis, Zach, Ben**	**2/5/2018**	**As Issued**

1. Overview

For our project we will be creating a gift registry website. The primary use is creating a shareable wishlist. Users will be able to create lists, add or remove items, and organize and rank these items within their list. Furthermore, they will be able to share these lists with others, and counterparties will be able to see what items have been purchased for an individual. The problem this solves is for events, parties or holiday times, we're creating an easily accessible place for people to share what gifts they want, and others can see what gifts have been purchased to help guide their buying behavior. The potential user is essentially anyone who takes place in a gift purchasing event like the holiday times.

2. Related Work

The registry, or even just gift registry space is a fairly crowded market. MyRegistry.com is the largest business here, with a website tailored to wedding, baby and gift registries. Their main value-add is integration with online retailers, as they offer a browser extension which allows you to add to your registry from partner sites seamlessly. Other websites in this space are Wishlistr, who offer basic list creation/sharing, and GiftBuster who offer an integrated phone app/website experience.

Our main differentiator is ideally ease-of-use. MyRegistry offers lots of integration, but their UI is extremely crowded, completely overloaded with content. Wishlistr and GiftBuster offer scaled down applications from MyRegistry, but down offer the full range of benefits that MyRegistry does. Ideally we will strike a happy medium here: a clean, easy to use application that offers some third party integration and dynamic gift suggestions.

3. Proposed High level Requirements

a. Functional Requirements
    i. Essential Features (the core features that you definitely need to finish)
        1. Users create wishlists
            a. As a creator of a list, I want to be able to create a list.
        2. Users modify wishlists
            a. As a creator of a list, I want to be able to modify a list after it has been created already.
        3. Users delete wishlists
            a. As a creator of a list, I want to be able to delete a list.
        4. Share wishlists/control who accesses
            a. As a creator of a list, I want to be able to share my list and control who sees it.
        5. View someone else's wishlist, navigate between wishlists shared with you
            a. As a user of this app, I want to be able to view the lists that have been shared with me.
        6. Seeing what items have been purchased on wishlists shared with you
            a. As a user of this app, I want to be able to view which items have already been purchased on lists that have been shared with me.
        7. Mark as purchased items on others' wishlists
            a. As a user of this app, I want to be able to mark an item as "purchased" on a list which has been shared with me if i have purchased that item.
        8. Users see landing/homepage upon entering the site, with links to users/groups
            a. As a user of this app, I would like to be greeted with a user friendly UI with comprehensible links to other user's lists which have also been shared with me.
    ii. Desirable Features (the nice features that you really want to have too)
        1. Rank/organize wishlist items
            a. As a creator of a list, I want to be able to organize my list in order of, for example, my desire to have those items.
        2. Link to wishlist items on the web

        a. As a creator of a list, I want to be able to add links where items on my list can be purchased.
    3. Search among wishlists shared with you
        a. As a user of this app, I want to be able to search for particular items amongst the lists shared with me.
iii. Optional Features (additional cool features that you want to have if there is time)
    1. Suggest wishlist items to others
        a. As a user of this app, I want to be able suggest items on lists which have been shared with me. In particular, items from the wishlist database or from the web.
    2. Integrate with large vendors websites
        a. As a user of this app, I would like it if the application was integrated with larger vendors such as amazon in order to increase the ease of use and fluidity of the application.
    3. Classify different types of users (charities)
        a. As a user of this app, I would like to be able to tell the difference between and/or search for different types of users such as single users, charities and large organizations' lists.
    4. Associate events with wishlists
        a. As a creator of a list, I want to be able to associate wishlists with events such as birthday parties or weddings.
iv. Existing Features (Not applied to a brand new project)

b. Nonfunctional Requirements
    i. Fewer than 1.5 seconds to load a moderately sized list (10-100 items)
        1. As a user of this app, i would like my list to load quickly.
    ii. Scalability: Add a large amount of items to wishlist (up to 1000 items)
        1. As a user of this app, i would like to be able to add a large amount of items to my wishlist.
    iii. Complexity: Fewer clicks to reach anywhere (maximum of 6 clicks)
        1. As a user of this app, i would like to be able to reach any page of this webapp from another within 6 clicks.
c. Implemented Features (*to be completed at the end of each iteration*)

4. Management Plan

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

a. Process Model

The Scrum process will be used to develop the project. There will be four, 3-4 week sprints in the development of the project. The first sprint will encompass all of the requirements gathering as well as a basic functioning application. The second sprint will be about implementing features to make the application whole and usable. The third sprint will be used to enhancing the basic functionality and using customer feedback to enhance the

user's experience. This process model will not be true Scrum, we will not be having a daily standup. Instead, we will be having weekly meetings accompanied with a slack channel for constant communication within the team. The decision to not have daily stand ups comes from the fact that all of the team members have full time jobs and don't have the time to properly contribute to a daily standup.

b. Objectives and Priorities

(Project Goals can include but not limited to completed all proposed (essential) features, deploy the software successfully, the software has no known bugs, maintain high quality?)

The highest priority will be having a deliverable product at the end of each iteration. The next highest priority will be to complete all essential features. The third priority is to make sure the product is deployed and runs smoothly without crashing bugs.

c. Risk Management (need update constantly)

i. Refer to the CS673S18T3_RiskManagement Sheets file for risk management

d. Monitoring and Controlling Mechanism

The team will meet weekly for an hour to discuss progress and any issues. Independently of the weekly meeting, the team has access to slack to ask questions and communicate small progress. Time after class on Thursdays will also be used to get together in person and make sure things are moving along nicely.

e. Schedule and deadlines (need update constantly)

2/09/18 - end of sprint 0
3/02/18 - end of sprint 1
3/30/18 - end of sprint 2
4/27/18 - end of sprint 3 and project

5. Quality Assurance Plan

Additional information regarding the testing performed can be found in the

a. Metrics

i. Definitions of Quality Metrics

Results for the following metrics will be tabulated at the end of each iteration.

Product - Complexity	
Metric	**Description**

Lines of Code (LOC)	The total lines of code in all application files. An increase in value correlates to an increase in complexity.
Number of Files	The total number of files in the application. An increase in value correlates to an increase in complexity.
Number of Classes	The total number of classes defined in all application files. An increase in value correlates to an increase in complexity.
Number of Methods	The total number of methods/functions defined in all application files. An increase in value correlates to an increase in complexity.

Product - Stability	
Metric	**Description**
Defects found (iteration)	The number of defects found in the iteration. An increase in value correlates to a decrease in stability.
Defects found (total)	The number of defects found across all iterations. An increase in value correlates to a decrease in stability.
Defects fixed (iteration)	The number of defects fixed in the iteration. An increase in value correlates to an increase in stability.
Defect backlog	The number of defects that are waiting to be fixed in the current release. An increase in value correlates to a decrease in stability.
Residual defects	The number of defects that have been elected to either not be fixed or to be fixed in a future release. An increase in value correlates to an decrease in stability.
Defect fix rate	The ratio of defects fixed to defects found in the iteration. An increase in value correlates to an increase in stability.
Distribution of defect severities	The percentage of defects associated with each severity category. A distribution favoring lower severities correlates to an increase in stability.
Unit test pass rate	The percentage of unit tests that pass on the build at the end of the iteration. An increase in value correlates to an increase in stability.

Product - Cost	
Metric	**Description**
Total hours spent on project	The total hours spent on any aspect of the project. An increase in value correlates to an increase in cost.
Distribution of hours by product activity	The percentage of hours spent on each project activity. This metric shows what activities are incurring cost, rather than affecting the cost

	by its value alone.

Process - Testing	
Metric	**Description**
Unit test code coverage	The percentage of the application code tested by at least one unit test. A higher value is better.
Ratio of Lines of Code to Lines of Test Code	The ratio lines of “production” code to lines of code in testing. This value should be somewhere in the vicinity of 1:1.0-2.0.

Process - Defect Tracking	
Metric	**Description**
Defects awaiting disposition	The total number of defects awaiting disposition at the end of the iteration. At the end of each iteration, this value should be zero.
Defects missing severity/probability	The total number of defects missing either the severity or probability categorization at the end of the iteration. At the end of each iteration, this value should be zero.

ii. Results (by Iteration)

The results for quality metrics are updated at the end of each iteration. Please refer to the Software Test document for more detailed information regarding test results.

b. Standard

Requirements Documentation

Requirements are to be documented in such a way that they can be mapped to either verification or validation testing results. Any requirements added must be done so at the consent of all team members.

Design Documentation

Design documentation will be finalized as a part of the final documentation. Classes and methods are expected to have comments describing their purpose and any special considerations to be made when using or interacting with them.

Quality Assurance Documentation

Quality Assurance documentation will be produced at the end of each iteration, providing the data for product and process metrics. Refer to the Metrics section for more detailed information regarding what metrics are reported.

Coding Guidelines

Coding guidelines are those as enforced by the Rubocop Ruby gem - a static code analyzer used in conjunction with the project. Additional coding guidelines may be instated upon agreement amongst team members.

c. Inspection/Review Process

Requirements Review

Requirements are to be reviewed by all team members prior to acceptance. Any changes to the priority/inclusion of any requirements must be agreed upon by all team members.

Documentation Review

Documentation reviews are to be performed at the end of each iteration to ensure all content is testable and is covered by one or more test cases. Any issues identified in this review must be addressed in the next iteration. In the case of the final iteration, any issues identified must be corrected prior to product release.

Code Review

Code review can be informally requested at any time during the implementation process. However, code review by at minimum one other individual is required in order to mark the implementation of a feature as ready for inspection/verification.

Verification

In order to mark a feature or bug as completed, an individual other than the implementer of the code checked in against that work item must verify the functionality/defect described in the work item has been addressed.

d. Testing

i. Unit Testing

Unit tests should be implemented by a developer either (a) in conjunction with the implementation of a feature or (b) in conjunction with implementing corrective action for a defect. Unit tests can be run at any time by any team member, but their pass/failure state will only be formally reported at the end of each iteration by the QA leader.

ii. Integration Testing

Integration testing occurs when features are implemented to ensure that features implemented previously or concurrently do not introduce defects. Failures found during integration testing are reported as defects and may be reported by any team member.

iii. System Testing

System testing occurs at the end of each iteration, ensuring the product functions in the target production environment. Failures during system testing are reported as defects and may be reported by any team member.

iv. Functional Testing

Functional testing takes two forms: ad-hoc and formal functional testing.
Ad-hoc functional testing can be performed at any time by any team member, and failures found are reported as defects.
Formal functional testing is the execution of test cases confirming the product functions as specified. Formal functional testing is performed at the end of each iteration. Failures during formal functional testing are reported as defects and as test cases being marked as failed. Formal functional testing is lead by the QA leader, but may be performed by any team member.

v. Performance Testing

Performance testing ensures the product functions within the specified performance requirements. Performance tests can be run informally at any time by any team member, but their pass/failure state will only be formally reported at the end of each iteration by the QA leader.

A separate document about testing result should be linked here:

e. Defect Management

The defect management process applies to any change request created for behaviors that deviate from the documented product requirements and/or behaviors. Portions of it may be used for other types of change requests, but all components must be followed for defects.

Defect Management Roles and Responsibilities

Role	Description
CR Originator	Individual that has entered an anomaly into the defect tracking system. Responsible for including required information when initially creating the CR.
CR Owner	Responsible for determining, implementing and testing the corrective action for the CR. It is possible for a CR to have more than one owner.
Verifier	Responsible for testing the fix implemented by the CR owner to ensure the efficacy of the fix and that it has not resulted in additional anomalies.
QA Leader	Responsible to ensure the defect tracking process is followed.

I. Defect Tracking System

All defects are maintained in the Pivotal Tracker defect management system. All supporting information for the defect should be associated with the change request created.

Defect Documentation Components

a. New Defect Template

All new defect work items must be submitted with the following information:

- A brief description of the problem/error
- The expected behavior
- Steps to reproduce
- How reproducible the problem/error is

b. Fix Template

All defects with corrective actions implemented against them must contain the following information for verification:

- Steps taken by the developer to test the corrective action.
- Steps to be taken by the verifier to test the corrective action (if different).

c. Defect Severity

All defects must be assigned a severity as a part of the justification for how a defect is dispositioned. A defect may be assigned one of the following severities based on the associated criteria:

Severity	**Problem Type**
4 - Critical	Problem will cause the product to become unresponsive and unable to recover. Unintended data loss is likely or guaranteed.
3 - Serious	Problem severely restricts use of the product for frequently or for a prolonged duration. Work around is nonexistent or difficult. OR Problem could result in the product being unresponsive. OR Problem could result in unintended data loss.
2 - Medium	There is a simple work around for the problem and product use is acceptable. AND Problem does not result in the product being unresponsive.
1 - Low	Problem is a minor inconvenience to user, and product use is acceptable. AND Problem does not result in the product being unresponsive.

d. Defect Probability

All defects must be assigned a probability as a part of the justification for how a defect is dispositioned. A defect may be assigned one of the following probabilities based on the associated criteria:

Probability	Problem Type
4 - Guaranteed	All users (100%) are expected to encounter this defect.
3 - Very Likely	The majority of users (50%-99%) are expected to encounter this defect.
2 - Likely	Some users (10%-49%) may encounter this defect.
1 - Unlikely	A small percentage of users (<10%) may encounter this defect.
0 - Very Unlikely	No user is ever expected to encounter this defect.

e. Defect Priority

All defects must be assigned a probability as a part of the justification for how a defect is dispositioned. The defect's assigned priority should be correlated to its severity and probability. A defect may be assigned one of the following priorities based on the associated criteria:

The following table should be used as guidance for assigning defect priority. Deviations from this guidance should have documented rationale associated with the defects explaining why the deviation is acceptable.

Severity / Probability	0 - Very Unlikely	1 - Unlikely	2 - Likely	3- Very Likely	4 - Guaranteed
1 - Low	Optional	Optional	Desirable	Desirable	Desirable
2 - Medium	Optional	Desirable	Desirable	Desirable	Required
3 - Serious	Desirable	Desirable	Desirable	Required	Required
4 - Critical	Desirable	Required	Required	Required	Required

6. Configuration Management Plan

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

a. Environment management

i. Application will exist in an AWS Elastic Beanstalk environment

1. Simple configuration
2. Simple deployment
3. High level of abstraction

ii. SQL Server Express will be hosted in AWS RDB

b. Code commit guidelines

i. Group has chance to review pull requests before they are merged

7. References

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

8. Glossary