CS673S16 Software Engineering
Silver Serpents - Projects Portal
Project Proposal and Planning

Team Member	Role(s)	Signature	Date
Ben Schwartz	Group Leader	*Ben Schwartz*	2/4/18
Chad Cover	Configuration Leader Environment and Integration Leader	*Chad David Cover*	2/6/2018
Andy O'Connell	Implementation Leader	*Andy O'Connell*	
Kanishka Bagri	QA Leader	*Kanishka Bagri*	
Prabhakar Punj Lal	Design Leader	**Prabhakar Punj Lal**	02/07/2018

Revision history

Version	Author	Date	Change
1	**Ben**	**2/4/18**	**Added member names and roles** **Added project overview and related work**
2	**Andy**	**2/4/2018**	**Completed Proposed High level Requirements**
3	**Chad**	**2/6/2018**	**Completed Configuration Management Plan section.**
4	**Kanishq**	**2/6/2018**	Completed the following section: Quality Assurance Plan Metrics Standard

			Inspection/Review Process Testing Defect Management Process improvement process
5	**Prabhakar**	**2/7/2018**	**Updated Management Plan Section**
6	**Chad**	**2/8/2018**	**Updated Design Document & Coding Standards subsections**

1. Overview

At the beginning of our group's formation, we [Chad and Andy] wanted to do something using the MEAN stack. We started thinking about making a Single Page Application [SPA] to be used by future software engineering

Our plan, for right now, is to make a place that will store students projects for this class. That way, when students start class, you can send our link to them and all the projects will be in one place. Plus, they can use it to search for different tags on the projects. The projects will be tagged by tools or languages used and by type of project (game, web app, etc). The GitHub, pivotal tracker, notes, and possibly even features that the group didn't have time to implement would also be there.

This SPA could be extended to be helpful for companies, too. Within a single company, its projects could be stored here for easy access and/or as a "portfolio" for prospective Clients. Outside of a single office, this could be a paid service that collects projects from multiple firms and give potential clients an idea of what possibilities are out there

2. Related Work

Sadly, there are quite a few other websites like this. Almost all of them offer a similar collection aspect to ours, but the vast majority of them focus on the statistics of the projects. The difference between those and ours is that we want to focus on the projects themselves.We want the collection of projects to be available in the SPA, we want them to be

tagged by their features, by their purposes, and their formats.
https://www.projectportal.net
https://www.liquidplanner.com/support/articles/project-portals/
http://promise.site.uottawa.ca/SERepository/
https://airtable.com
http://openscience.us/repo
http://isbsg.org

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

3. Proposed High level Requirements

a. Functional Requirements
   i. Essential Features (the core features that you definitely need to finish)
      1. Website shall be an overall portal that will store Student projects.
      2. Website shall use user authentication that will be restricted to Boston University Students, Faculty, and researchers.
      3. Website shall use strong encryption to protect Information, Identities, and all other sensitive information.
      4. Users will be given minimum required permissions by default.
      5. Users shall be able to link a GitHub or BitBucket Repository to a created project.
      6. Projects shall be classified as to what department they are relevant to.
      7. User shall be able to search for project by typing keywords into search prompt.

   ii. Desirable Features (the nice features that you really want to have too)
      1. Users shall be able to add project participants based off of the user models from the user authentication system
      2. Unregistered Users shall be able to see high level information about projects
      3. Section for Feedback Comments
      4. Option to Favorite a project, such that it gets added to the Users Profile page under 'Favorites" for future reference
      5. Highlight the search string in the Search Result Set
      6. Paging for Result set greater than 100
      7. A guest Persona to view limited details of the project

iii. Optional Features (additional cool features that you want to have if there is time)
    1. Platform shall Use Machine learning to help classify new projects being created on the system
    2. Platform shall use Machine learning to automatically group similar projects together
iv. Existing Features (Not applied to a brand new project)

b. Nonfunctional Requirements
    i. Tech stack will incorporate Angular 2, MongoDB, Express, Nodejs.
    ii. Development Team shall use Github to host project Files
    iii. Development Team shall Pivotal Tracker for internal Team Tracking of development
    iv. Web Front End shall be visually enhanced with Angular Material

c. Implemented Features (*to be completed at the end of each iteration*)

4. Management Plan

a. Process Model: **AGILE- SCRUM**

The main rules we are going to follow:

- Weekly 30 minutes meetings on Thursdays (Two 15 minutes standup meetings before and after the lecture).
- Weekly 1 hour meetings on Google Hangouts on Mondays discussing the week's plans and the progress achieved till then.

Iteration 0: Phase 1-Planning and Documentation:

- Decide the technologies that we will use for front, middle and back end development.
- Learn and understand the technologies that will be used in the project-The MEAN Stack.
- Learn REST API.
- Discuss **a)** the features that *must* be included in the MVP, **b)** the features that adds additional functionalities to the application but are not absolutely necessary **c)** the features that beautifies the project (front end).
- Identify potentials risks.
- Plan the schedule.
-

Iteration 1: Phase 2- Development of the MVP-1:

- Work on the wireframes and the stories.
- Develop necessary data flow representations/diagrams.
- Plan tasks for each sprint of the iterations.
- Develop '*Registration/Signup'*, '*Login/Sign in'* functionalities and implement '*Access Controls'*.

- Develop ‘New Project’ functionality using which a user can upload a project (thinking about the tags that we need to associate with them during uploading).
- Test what is developed so far.

Iteration 2: Phase 3- Development of MVP-2

- Plan and start developing the *‘Project Tracker’* feature.
- Integrate and Test what is developed so far.

Iteration 3: Phase 4- Add ‘Search’ functionality and finalize MVP

- Implement the ‘Search’ functionality based on tags, email, username, department/course.
- Work on the ‘*My Projects*’ (or ‘*View Projects*’) feature.
- Integrate and Test what is developed so far.

b. Objectives and Priorities

Essential goals of the project include the following feature:

- A user will be able to upload a project which will be available for others to see.
- Limited access controls based on 3 types of user- Student, Faculty and Guest.
- Search system based on various attributes.

Ambitious goals/Stretch Goals:

- Email verification (if possible mobile number verification too) to avoid spammers and bots.
- Comment section where people can comment on the projects.
- Feedback from the professors/client, where they can upload their comments on specific parts.

c. Risk Management (need update constantly)

RISK ID	CATEGORY	DESCRIPTION	PROBABILITY	IMPACT	RESPONSE
SS-101	LEARNING	Lack of expertise in the development stack (here the MEAN stack).	3	4	Collaborative learning/online learning.
SS-102	MEMBER DROPS	Possibility of a member dropping the course.	1	5	Re-plan schedules and assign tasks

SS-103	REQUIREMENTS	Requirement not taken carefully or properly.	2	4.5	Re-think and arrange meeting with the client
SS-104	CODE CONFLICTS	There is a possibility of conflicts between codes written by two members i.e. they do not conform to each other.	3	4	Discuss the code in detail step by step
SS-105	BUGS	Bugs in access controls, design bugs, etc. which hinder the advancement of the project.	3	5	Group discussion.
SS-106	MEETINGS	Member skips the meeting/Meeting is canceled due to some reasons/	2	3	Someone from the team arranges a session to help him catch up with others.
SS-107	COMMUNICATION	A member doesn't clarify his doubts.	3	5	Group meeting dedicated to doubts' clarification.

d. Monitoring and Controlling Mechanism

To keep the team on track we need to:

- Regularly conduct group meetings to discuss problems and issues that we encounter during development (lack of proper communication can be disastrous).
- Regularly contact each other regarding any issues or for clarification of a query.
- Share with the group any imperative changes that need to me made (not discussed previously in meetings).

Tools for monitoring and communication:

1) **Monitoring tools:**
    - Pivotal Tracker

- GitHub (Code Integration)
- Mocha & Chai (Testing)
- Google Drive (Document Sharing)

2) **Meetings and communication tools:**
   - Google hangouts
   - Slack

e. Schedule and deadlines

We will try to complete the tasks two days before the due date of each iteration so that we have additional time for bug fixing and code review as well as for preparing the presentations.

Iteration 0	02/09/2018
Iteration 1	03/02/2018
Iteration 2	03/30/2018
Iteration 3	04/27/2018

5. Quality Assurance Plan

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

a. Metrics

Product Complexity:

Lines of Code (LOC)	The total number of lines of code in the application
Number of Files	The total number of classes or functions in the entire application
Number of classes:	The total number of classes or functions in the entire application
Number of methods	The total number of methods in the application

Process Metrics:

Team Velocity:	This measures the number of User Stories the team complete in an Iteration

Open/Close Rates	This measures the number of Defects reported in a sprint
Active Days	This is a measure of the time a team member spends contributing to the product development cycle
Code Churn	Code churns indicates the number of Lines of code that were modified during an iteration. This would include the lines of code that have been added or deleted.
Defect per n LOC	This is a ratio of the number of defects reported for every n lines of code
QA kickback rate	Number of defects that were re-opened by QA in an Iteration

Testing:

Unit Test	The percentage of code that is tested by Unit Tests
Average Lifetime of Open Defects	Average lifetime of reported defects. This should be less than 3 days
Defects reported for each User Story	The number of defects reported for a user story. This number is a direct indication of the quality of the story
Automation Coverage	The percentage of identified P1/P2 test cases that are being tested via Automation.

Product Cost:

Man hours	The total number of hours spent on the Project, including development, Test, Management

b. Standard
(e.g. documentation standard, coding standard etc.)

Scope Document:
This document summarizes the Scope of the project as decided by the team. Any modifications to the scope would be agreed upon by the Customer, and the team

members

Design Documents:

There will be three design documents:
1. A series of wireframes that give a rough approximation of the User Interface;
2. A series of workflow diagrams that depict the flow of data through the front, middle, and backends
3. A series of UML diagrams that show the data models

Quality Test Plan Document:

QA Test Plan would be ready in the first week of each Iteration. The Test Plan should document the Strategy, Test scenarios, Automation Plan, Metrics etc for each sprint.

Coding Standard:

1. Code must be Typescript compiled into ECMAScript 6 or fully linted valid ECMAScript 6. NPM will automate compilation and linting for those who code in Typescript.
2. Variables should be named meaningfully, begin with a lowercase letter, and conform to camelCasing.
3. Global variables, constants and class names should begin with a letter, and be in UPPERCASE.
4. Code should be indented to assist readability.
5. Functions whose names and parameters are not immediately obvious, should include documentation that explains functional usage, expected parameters and type, and return values and type.
6. Data models will conform to the standards established at http://www.json.org/, so as to be importable into MongoDB. Developers are encouraged to enable JSON syntax highlighting in their IDEs or to use the online JSONLint tool:
7. HTML models will conform to the W3C's HTML5 standards at https://www.w3.org/TR/html52/.
8. Stylesheets will conform to CSS3 standards: https://www.w3.org/standards/techs/css#w3c_all
9. Developers are encouraged to enable Typescript, Javascript, JSON, HTML, and CSS syntax highlighting in their IDEs, or to use online validators such as:

    - JSON: https://jsonlint.com/
    - HTML5: https://validator.w3.org/#validate_by_upload+with_options
    - CSS3: https://jigsaw.w3.org/css-validator/#validate_by_upload

c. Inspection/Review Process

(e.g. describe what are subject to review, when to conduct review, who do the reviews and how ?)

Scope Document:

The Scope document needs to be reviewed and accepted by all the team members. The scope of the project could be subject to change with the progress, hence all the future modifications to the scope will also go through Review by all the team members

Code Review:

- As a guideline, every checking to the master branch will go through a code review by 2 peers from the team.
- The code review would include the following:
    - A review of the pull request
    - Unit Test coverage

Further guidelines of the development process would be available in the SDD of the project

User Story Review and Acceptance:

Every Story would go through a Demo with the team (and the customer) ensuring all the Acceptance Criteria for the Stories have been completed. The story would be accepted/marked as completed by the story owner when these criteria are met.

d. Testing

Testing Type	Description
Unit Testing	Unit testing is a part of the software development process, where the developers writing the code also develop corresponding Tests to test the individual classes/methods. These tests will be automated and run as a part of each build.
Integration Testing	Integration testing is a part of the development process. In this phase individual modules are combined and tested as a group. Eg: Testing of a REST endpoint which communicates with the backend and returns a valid response
Functional Testing	Functional testing is a type of black box testing where a slice of functionality of the whole system is tested. Functional testing is based on the Test Planning that is done during the feature development with an objective to identify and list all the test cases to test a functionality. During this phase, the above mentioned test cases are executed to ensure the features conform to the defined expected behavior

Performance and Scale Testing	Performance testing is a form of software testing in which the team focuses on testing the performance of the application under heavy load. The objective is not to find functional defects, rather capture the numbers against the standard identified metrics. This testing is usually done by a specialized team

(e.g. who, when and what type of testing to be performed? How to keep track of testing results?)

A separate document about testing result would be linked here.

e. Defect Management

Defect Management is the process to identify defects in a software and the sequence of phases it goes through before the fix for the defect is merged, verified and validated.

Defect Management Roles and Responsibilities

Role	Responsibility
Reporter	The team member who found and reported the bug in the Defect tracking system. This can be a QA, Developer, Support Or the Customer
Dev Owner	The owner of the defect who is responsible for analysing the root cause and providing a fix for it
QA Owner	The QA team member who is responsible for verifying the fix once the code change is merged in to the master branch
Observer	Other team members who are watching the issue to keep a track of its status.

Defect Management System

- We will be using the **XYZ** for our Defect Management.
- All the defects found during the development phase will be reported in the Defect Management System.
- The defects will go through the Life cycle of the defect:
  Open -> Development (for fixing) -> QA (Validation) -> Closed

Defect Logging guidelines

All the defects being logged into the system need the following information.

- Defect Summary:
- Defect Priority
- Defect Severity
- Development Owner
- QA Owner:
- STRs: Steps to Reproduce
- Expected Behavior
- Actual Behavior
- Attachments: Any screen grabs or Log files

Note: Defects with missing information will be sent back to the reporter

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

6. Configuration Management Plan

a. Configuration items and tools

i. Bug control: Github issues
ii. Requirement analysis and progress monitor tools: Pivotal tracker
iii. Deployment: Webpack deployment to a hosted Apache server. (ISP and plan to be determined, possibly gwiddlefoundation.org.uk.)
iv. Communication tools: Google Chat, Slack Channel.

b. Change management and branch management

i. Each Pivotal Issue ticket will be assigned a number.
ii. The developer who picks up the features implementation ticket will create a working branch: “features/ticket-number.”
iii. Each GitHub bug will be assigned a number.
iv. The developer who picks up the bug fix ticket will create a working bug branch: “bugs/bug-number.”
v. Branches should be pushed to GitHub only after the feature is implemented or bug fixed. (See Code commit guidelines below.)
vi. A ‘‘master’’ branch is created to hold published/functional version of project.
vii. Features and bug branches will be merge into master after they receive at least two PR approvals. (See Code commit guidelines below.)

c. Code commit guidelines

i. Pull and merge the remote master branch into local working branches on a daily basis.
ii. Commit changes to the local branch frequently, with meaningful commit messages that describe the changes made.
iii. A features or bugs branch may be checked in only after it passes unit testing on the developer’s local machine.

iv. Push the local branch to the GitHub repository after it passes local tests. Make a pull request to all team members.
v. At least two team members must review a PR before it is approved. Reviewers must pull the to-be-reviewed branch to their local machine, and review the code and run the tests successfully before approving.
vi. The developer who checked in the branch is responsible for merging it to master and deleting the features or bugs branch in the GitHub repository after the merge is complete. (The choice of keeping or pruning local features branch shall be left up to the individual developer.)

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

MEAN: Acronym describing the technology stack consisting of MongoDB, ExpressJS, Angular, and Node.js.

PR: Pull Request

SPA: Single Page Application