**CS673 Software Engineering** 

**Team 4 - Project Name (**Project Portal )

**Project Proposal and Planning**

|  |  |  |  |
| --- | --- | --- | --- |
| Team Member | Role(s) | Signature | Date |
| Zachary Kysar | Backup Team Leader and Requirements Leader | *Zak* | 09/12/2019 |
| Connor Richmond | Team Lead | *CR* | 9/15/2019 |
| Rattikan Dudley | Security Lead | *Rk* | 9/15/2019 |
| Molla Negash | Configuration management plan lead | *MN* | 09/16/2019 |
| Juan P Sanchez | Quality Lead | *JPS* | 9/15/2019 |

**Revision history**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author Signature** | **Date** | **Change** |
| 1 | Zak | 09/12/2019 | Added notes in red |
| 2 | Zak | 09/15/2019 | Started adding high level requirements |
| 3 | CR | 9/16/2019 | Adding Project Overview and Management Plan |
| 4 | JPS | 9/17/2019 | Added QA plan |
| 5 | Rk | 9/18/2019 | Adding security feature in high level requirement |
| 6 | MN | 9/18/2019 | Adding notes on configuration plan |
| 7 | Zak | 9/18/2019 | Zak took read over all, making changes where needed. |
| 8 | CR | 10/16/2019 | Address Comments and Feedback from  Professor |
| 9 | CR | 10/30 | Update for current iteration and address feedback |
| 10 | CR | 12/3 | Update requirements to match project state |
| 11 | CR | 12/12 | Final update and polish |

[Overview](#_87t9hln2vjz0)

[Related Work](#_mps353x5ezyl)

[Detailed Description](#_fg3z0hpd4q9v)

[Management Plan](#_ds8oyr75pnh1)

[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](#_tadq5mb0pici)

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

[Metrics](#_b2haznn3yyz2)

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

[Inspection/Review Process](#_f1c69ifi68h7)

[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](#_jhct37ebxxpn)

[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](#_yyauft6zr9hw)

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

[Glossary](#_ty3i2nqffhtc)

# Overview

With the overall number of Computer Science related projects growing each year at different academic institutions the need for a centralized and searchable solution for project publication and discussion is growing. The goal of the Computer Science Project Portal is to provide a scalable, integrated platform for project development and publication. Active projects can post preliminary results while opening discussions to address issues or ask experts for their opinions as the project progresses. Projects later in the development cycle can utilize our platform to distribute results on a larger scale, and potentially seek collaborators for future work as new potential use cases and applications for the project emerge. Additionally, for more entry level coding projects, professors can use the Project Portal to consolidate and assess final projects/assignments in their courses. Every project submitted will be open source and searchable via keyword tagging extracted manually from user submission and from keyword extraction in the project description.

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

# Related Work

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

CS Project Portal is a centralized searchable project specifically designed for environmental education purposes. CS Project is very user friendly. We strive to create a better UI experience than our competitor’s applications. We feature a user question and feedback options from a project submitter as well as the user can contact a submitter directly via the given email. We will differentiate ourselves from other similar applications by providing a single centralized platform for computer science project hosting with a focus on collaboration and open source development.

Those are a sample from the similar applications:

* 1. Computer Science Portal for Geeks: <https://www.geeksforgeeks.org/>
     1. Lacks actual; projects or is too crowded by other aspects.
  2. CSEstack.org :[https://www.csestack.org](https://www.csestack.org/)
     1. Lacks actual: same as Geeks but has a nice feature of form discussion.
  3. CodeProject.com : <https://www.codeproject.com/>
     1. Public access

# Proposed High level Requirements

* 1. Functional Requirements
     1. Essential Features (the core features that you definitely need to finish)
        1. Add Project to Share
           1. As a Computer Science student, I want to be able to share my projects with other computer science students so that I can contribute to the community for other computer science students.
        2. Search Projects
           1. As a Computer Science guru, I want to be able to generate ideas for projects and have best in class examples.

I would be interested in searching all variables in the object.

Acceptance test would be to set parameters of search, then actually search tuples from the project table.

3. Create an account and log-in. Gather data on the user. This would

aid in searching as well.: Student Object should contain:

* + - * 1. Name
        2. Focus of Study
        3. School Name
        4. Experience
    1. Desirable Features (the nice features that you really want to have too)
       1. Comment on projects, and save commenting history.
          1. Would need new table for this.
       2. Upload files to the project
       3. Edit Project info after upload
       4. Remove Project if Project owner
       5. Open applications for contributors on projects
          1. Project owner could choose to publish project on a contributor board
          2. Contributors could search and view projects that they are interested in
          3. Once they found an interesting project they could request to be a contributor to gain access to the project
       6. Sending projects to other users
          1. When a user comes across an interesting project they could choose to share (via a button) with another user
          2. We could potentially have a “friends list” type functionality or they could search for other users by name.
          3. Shared projects could either be received as a notification or it could start a message thread between users.
    2. Optional Features (additional cool features that you want to have if there is time)
       1. Be able to contact project submitter
          1. Through email.
       2. Analytics of projects and demographics and display on view
       3. Add multiple authors to project
       4. Ability to add non users as authors
       5. Add, this is my favorite project, button
       6. Keywords that can be used to search the project
  1. Nonfunctional Requirements
     1. The application will be scalable to other browsers
     2. The application will be able to support up to 500 users or projects
     3. The Project pages can store up to 100 user comments and questions
     4. Speed - keep our code light, and simple.
     5. Security - make sure any sensitive data is not exposed.
  2. Implemented Features

Iteration 1:

* Add project
* Basic project tagging
* View projects in project table

Iteration 2:

* Rudimentary search functionality that traverses project table and returns matching projects
* View project description page- User can click button next to project name and be redirected to that projects page

Iteration 3:

* Focused primarily on building out site flow with view development coming in future iterations
* Further refined search process to implement popularity based return criteria in addition to matching search keywords

Iteration 4:

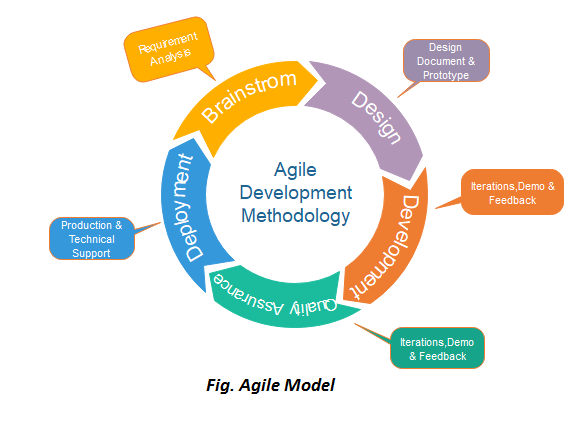
* Sign up and create new accounts( Professor/Faculty Distinction implemented)
* Log in and Log out by user type with password hashing
* Complete redesign and responsive design deployed to server

Iteration 5 (final iteration):

* All essential and most optional requirements implemented
* Student and faculty accounts with respective functionality
* Search projects with new drop down
* View project pop up with comments displayed
* Ability to add additional comments to any projects.
* Remove projects that you have uploaded
* Finalized testing and ensured all test cases pass

# Management Plan

## Process Model



We’ll be using an agile process model which is designed to help our project to adapt to change requests quickly.

## Objectives and Priorities

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

Complete all required/essential features to our live website. Have a functional and intuitive user interface that has no known bugs. Consistently test our code to ensure new deploys don’t cause breaks in production.

## Risk Management (need update constantly)

Risk management is going to be an ongoing priority throughout the project’s lifetime. Starting early the team will go over what development will look like for the project to make sure everyone is aligned in their own development process. After standardizing the process the initial workflow will begin with a focus on small and rapid iterations. Each change must be incremental to ensure an easy reverting process if any unseen bugs emerge. Having consistent unit testing is a requirement but each production deploy can introduce an untested condition/state. With every new bug that emerges we will actively incorporate new tests into our pre-deploy process. Lastly, having our weekly standups cover the basics of code review we will maximize the number of eyes that are reviewing code before we ship it!

## Monitoring and Controlling Mechanism

* + 1. All version control will be done through git
    2. Development of code will be done on individual dev branches and thoroughly tested and reviewed before being deployed
       1. We’ve standardized our merge and code review process to reduce bug risk post merge
       2. Only code that has been tested fully after merging with master can be pushed to origin/master and deployed to the live site
    3. We’ll use pivotal project tracker to monitor team progress with tickets and dev work
       1. Adding in weekly progress checkins and status updates while consulting pivotal tracker
       2. Each team member updates their relevent user story depending on what tasks they’re picking up
    4. Weekly stand ups to get briefed on team updates and current status of development
       1. This briefings have become streamlined to discuss not only current progress, but blockers and next steps as well.
    5. Ideally we’ll continuously test new code to catch any unforeseen bugs in production

## Schedule and deadlines (need update constantly)

* + 1. Iteration 0 - Finish proposal document. 9/18/2019
    2. Iteration 1 - Initial site framework with some pages implemented 9/28/2019
       1. Will need domain and server information with IDEs setup.
       2. Choose bootstrap theme and get hello world page up.
       3. Add project functionality implemented
    3. Iteration 2 - Fully developed, tested and deployed MVP with barebones features 10/14/2019
       1. Search functionality implemented
       2. View project functionality with individual project pages
    4. Iteration 3 - Build out of functional requirements and preliminary testing suite for future deploys 10/30/2019
       1. Some testing implemented
       2. Site flow implemented with all pages developed for MVP
    5. Iteration 4 - Complete new site design and some functionality regressed. Old functionality will be re-implemented after new site redesign is fully tested. 11/14
       1. Total site redesign for latest functionality.
    6. Iteration 5 - Complete deployed project meeting a bug free standard for all essential requirements fully tested and incorporated. Some optional features implemented and focus on test implementation and bug fixing. All test cases passed and site is robust with all current features implemented 12/05

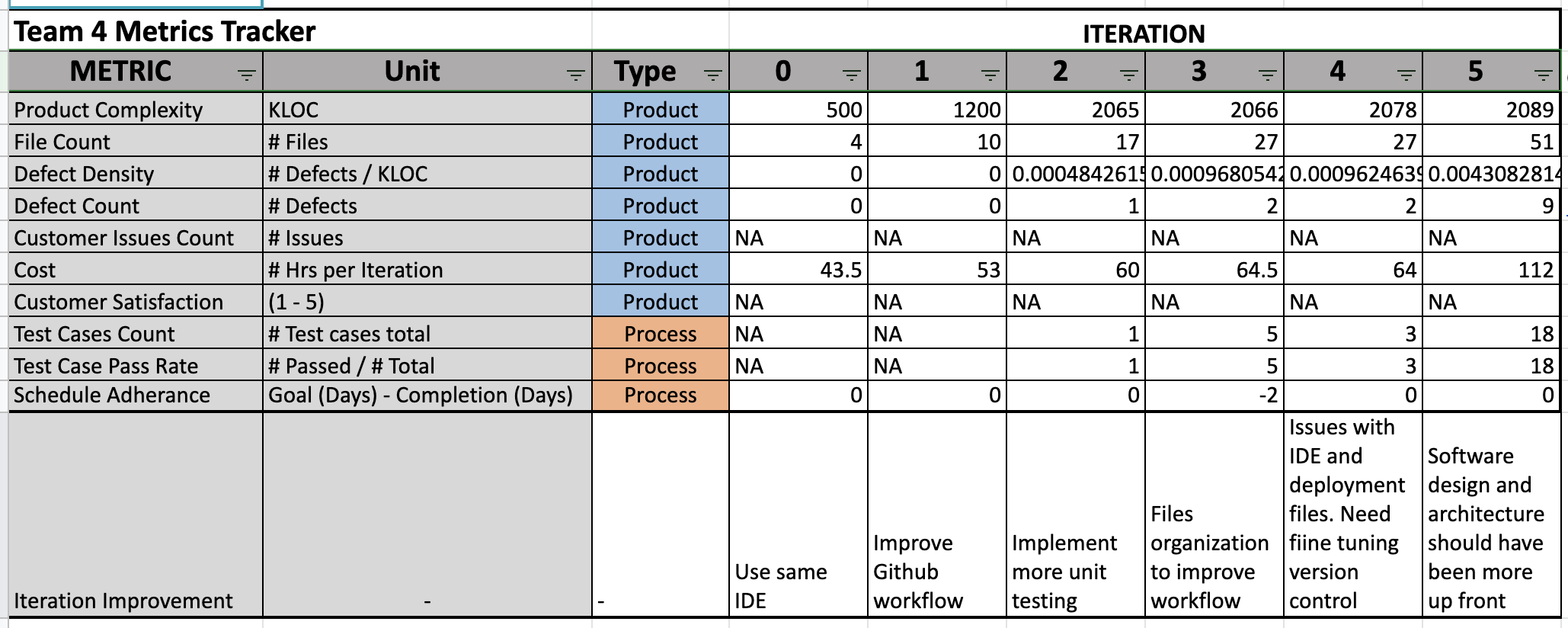
# Quality Assurance Plan

## Key Metrics - Will be monitored and evaluated every iteration

* Product Complexity/Size (KLOC): Count of lines of code generated for back, front end. Also include number of classes and methods.
* Defect Density (Defects/KLOC): Number of defects relative to the software size.
* Number of test cases and pass rate (# passed/ # Total)
* Defect Fix Rate(time in days): Determine the timing from open to close for defects.
* Number of Defects: Number of defects by iteration maintained in shared spreadsheet. Categorize defects by severity, priority, type and source.
* Cost (man hrs / week): Measure cost of the project, team productivity, and process improvement actions.
* Schedule adherence: Establish clear milestones and deliverables for each stage of development. Create a Gantt chart with timing and dependencies.
* Customer Satisfaction: Collect customer perception and feedback through survey. 1-5 rating system. Include areas like Functionality, Usability, Reliability, Performance, and Service.

b. Results:

Results will be compiled in the tracker posted in the Google Docs “Quality Metrics Team 4” excel file and reviewed at the end of each iteration during a lessons learned session.



## Standard

## Coding standards and name scheme:

## Classes: Class name should be a noun starting with uppercase letter. If it contains multiple words, every inner word should start with uppercase. Example: ProjectFolder.

## Interfaces: Interface name should be an adjective starting with uppercase letter. If it contains multiple words, every inner word should start with uppercase. Example: Runnable.

## Methods: Method name should be a verb verb/noun combination starting with lowercase. Example: print(), setUserName()

## Variables: Variable name should be a noun starting lowercase. If it contains multiple words, every inner word should start with uppercase. Example: userName, password.

## Constants: Constant name should be a noun in uppercase. Use public static and final modifiers. Example: final double PI = 3.1416

## Setter/Getter methods: Methods should have the word “set” or “get” and be public. Get method should not take any argument. Set method should take an argument and be void return type.

## SQL special words: Use all uppercase for SQL special words.

## Use consistent indentation and aim for code grouping.

* + - 1. Types of code example:
         1. CSS
         2. Javascript
         3. JQuery
         4. ASP
         5. SQL
    1. Comment standard:
       1. Classes/Methods: Use the following format to comment

*/\*\**

*\* Divides two numbers.*

*\* Precondition: num2 is not zero.*

*\* Postcondition: Returns the quotient of num1 and num2.*

*\* @param num1 the numerator*

*\* @param num2 the denominator*

*\* @return the quotient of num1 and num2*

*\*/*

*public double divide(double num1, double num2)*

*{*

*return num1 / num2;*

*}*

* + - 1. Leave comments work in process features and avoid obvious comments
    1. Same IDE You can use anything you want that connects to Git. Try to get something that is web html focused. Here is a list, for example: [https://ourcodeworld.com/articles/read/200/top-7-best-free-web-development-ide-for-javascript-html-and-css](https://slack-redir.net/link?url=https%3A%2F%2Fourcodeworld.com%2Farticles%2Fread%2F200%2Ftop-7-best-free-web-development-ide-for-javascript-html-and-css).
    2. Testing standardized instruction will be released, which includes a list of steps, features to be tested, inputs and outputs.
    3. Documents Standards
       1. Documents will be updated weekly and be reviewed by multiple team members before each iteration
       2. Any conflict can be marked via comment and discussed in our next team meeting or via slack

## Inspection/Review Process *(e.g. describe what are subject to review, when to conduct review, who do the reviews and how ?)*

* + 1. Review stories deliverables, metrics, schedule.
       1. To be done in our weekly team meeting
    2. Product - make sure the final product is correct based on the product requirements before opening merge request
    3. Conduct a biweekly code review in pairs before promoting branch/feature to development branch. Test functionality and collect defects. Once we’re confident in the new code create a new pull request highlighting the below:
       1. What has changed between this feature branch and the master branch
       2. Highlight what documents/processes have changed overall - added documentation, added tests, new behavior on site enabled
       3. How to use the feature. If fixing bugged code highlight what your fix will enable.
       4. How the design has changed and how it compares to the SDD
       5. Additional notes the reviewers should be made aware of
    4. Keep track of review expenses (man/hrs) for metrics reporting and process improvement actions

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

## Document testing results using “Testing Results Template” stored in Google docs

* + 1. The entire team will perform testing by splitting the types of testing up.
       1. UI testing
       2. System and integration testing
       3. Acceptance testing and beta testing
       4. Implement automated testing using Selenium Webdriver software
    2. We also have a specific development/testing domain to keep separate from out live “production” site

Current Status of Testing:

* Currently testing process is mostly manual and transitioning to automated unit testing in coming iterations
* Search involves testing new features affect on all aspects of the site. Testing results of search involves adding tags and confirming the given project is returned when its tag/author/title is entered in the search field
* Adding projects is simply checking the project table once a user submits a project
* Removing project test - tests that project is removed when a projects author clicks on the delete button
* View project tests that we can pull relevant project information from the project table (likes, comments, project info)

## Defect Management *(e.g. describe the criteria of defect, also in terms of severity, extend, priority, etc. The tool used to management defect, actions or personnel for defect management)*

* + 1. A defect will be considered as any deviation from the expected requirement. Mostly likely to be introduced via un caught bug that is deployed live on site
    2. Defect will be categorized in terms of:
       1. Severity (High / Mid / Low)
       2. Priority (High / Mid / Low)
       3. Type (Unnecessary / Omission / Inconsistency / Unclassified)
    3. Identify at which stage of the process of the development process was created and close as early as possible to reduce repair cost due to increased complexity.
    4. Once bug is identified and we know when it was introduced, draw up plan for resolving defect
    5. Document defects using Github functionality. Talk about planned steps to resolve defect
    6. Review quality scapes and impact to customer and document on user stories the impact and fix date goal.
    7. Implement fix of bug to resolve defect. Perform post mortem analysis to prevent future defects depending on severity (probably mid-high defects only)

# Configuration Management Plan

This is the road map in the software design which is important to monitor and evaluate the project cycle.

## Configuration items and tools

In software project management process, configuration management plan contains configuration items and tools. It includes, but not limited to implementation plan, the software or well commented source code, documentation/pseuod code and test plans, software tools and specifications.

## Change management and branch management

Change management is a procedural method which ensures quality and consistency when changes are made in the configuration object. In this step, the change request is submitted to software configuration leader.

## Change management

* Changes are committed to the repository
* In this project, each team members might make some changes on their working platform (eclipse, android) and then the change should be committed at local repository.

## Branch management

* + 1. Branching shema: This is a version control concept. The project has the master and dev branch that are maintained to have a live and dev version of the site. The master branch is meant to be a completely clean and bug free version of the site that could be deployed at anytime. The master branch will be tagged with different releases to track iterations. This will also enable us to revert to functional versions of the site at any point by reverting to a previous release.
    2. In addition to our master branch each team member currently has their own development branch where they can implement new features and fix existing bugs.
    3. When multiple users are working on a similar issue a communal remote branch will be created for the workstream that each user can push, pull, and merge with. Once both users have tested, committed and pushed their code the feature branch can be merged to master
    4. Once a branch is no longer being developed on it should be removed to keep our repository clean and easily searchable.
    5. <https://medium.com/@patrickporto/4-branching-workflows-for-git-30d0aaee7bf>
    6. workflow

**GitHub flow:**

* Anything in the master branch is deployable
* create a branch off from master for new version release and commit to the repository.
* Every new feature must be tested extensively for any unforeseen bugs
* Once a new feature is ready to be in productions merge master branch into your feature branch
* Test that all functionality is still performing as expected. Run all unit tests to ensure current master performs as excited
* Once all tests have passed merge branch to master
* If all work on branch is completed delete branch from remote.

The git-flow process is designed largely around the “release”. We don’t really have “releases” because we deploy to production every day - often several times a day (Scott, 2011 ).

Therefore this project uses gitHub branching as a workflow. Each team will use it in the development life cycle.

## Code commit guidelines

* + 1. Github, use their version control and branches by various methods mentioned above.
    2. Try to commit at end of each two week iteration after a period of testing has been completed.
    3. Suggest we split into 2 or 3 groups, instead of having everyone working on their own feature, have two or three people on one feature. There are more support and communication this way, and less risk. OR maybe front end or back end?

## Integration and deployment plan.

* + 1. Bought domain name for $10 a year: http://www.computerscienceprojectportal.com
    2. Was already subscribed to a hosting service form Arvixe. The deploy the site to the domain, and give us:
       1. FTP to files of the server
       2. SQL databases
    3. There are separate logins for the FTP and Database. As well as different software to access them. We use Microsoft SQL Server Management Studio for sql and Atom as an IDE for html pages.
    4. Every time we are happy with our code in our master branch, we migrate the files to our server.
    5. Then we test the new files on the test server.
    6. If all looks good on the test server deploy to our live site (production)

# References

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

* Berlack, H.R., 2002. Software configuration management. *Encyclopedia of Software Engineering*.
* Braude, E. Berstein, M. 2016. Software Engineering Modern Approaches