THE

A

TEAM

Project Plan

<u>Team Member</u>

Jason Acevedo

Vincent Agriesti

Thomas Campus

Ming Lei

Sally Ng

Table of Contents

1. Scope

1.1 Overview

The goal of this project is to produce the software product Clue-Less, based on the popular board game of Clue. As the primary contractor for this project, the A-Team estimates it will take 3 months of calendar time to complete this project with a final delivery on May 5th, 2018. Clue-Less will be supported on PCs initially with potential to have cross-platform support in the future. Clue-Less will allow up to 6 players to play simultaneously from different PCs that have Internet connectivity. In addition to the software, the project will produce all necessary deliverables to support a successful project.

1.2 Deliverables

- **Vision Document** describing the user's needs, the concept of operations, and all associated use cases.
- **Requirement Document** describing the assumptions, constraints, and requirements of the product.
- **Design Document** describing the communication formats, object relationships, and sequence of operations.
- **Target Implementation** of the product.
- **Presentation** of the entire project, its deliverables, and a demonstration of the final product.

1.3 Feature Increments

<u>Skeletal</u>

- This increment will provide the subsystems to establish server and client communications as well as a prototype of any of the interfaces required between the subsystems.

<u>Minimal</u>

- This increment will prove out the architecture and flash any remaining feasibility of the derived software requirements and design documentation. After this increment, a pair of users should be able to play a text based version of Clue-Less within a console.
- Number of players is limited to 2.

<u>Target</u>

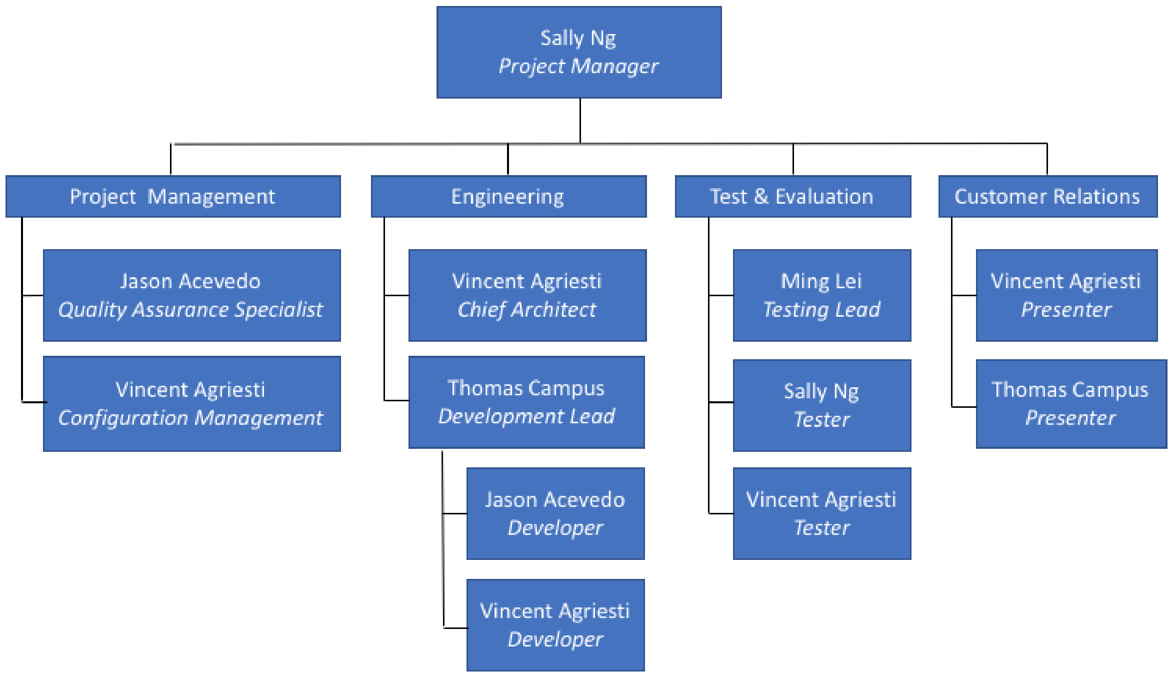
- The final increment will provide the ability to play a graphical version of Clue-Less within the operating system environment.
- All 6 suspects will be remotely playable.

<u>Dream</u>

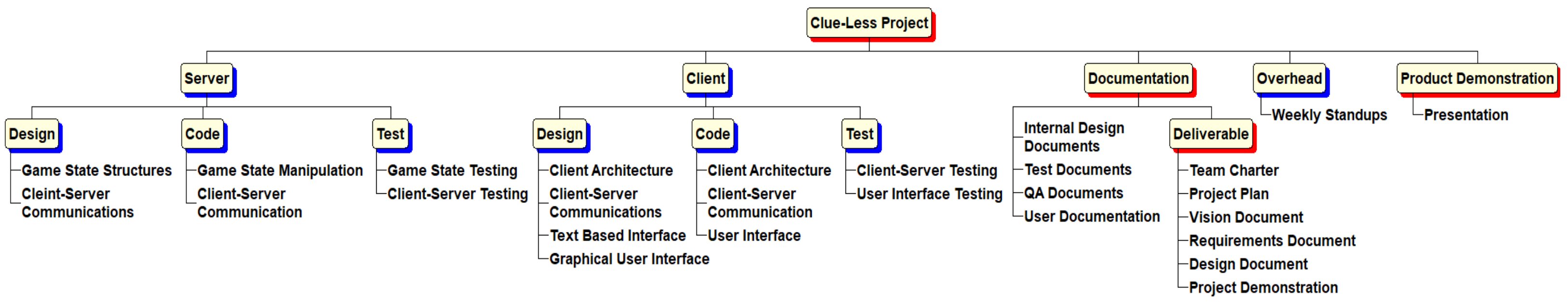
- As a stretch goal, and looking forward, another increment of development of Clue-Less may allow porting of the code to work across platforms (iOS, Android, Windows, Linux, Mac).
- The Dream increment would introduce better graphics in the GUI.

2. Organization

2.1 Team Structure



2.2 Work Breakdown Structure



2.3 Responsibility Matrix

Work	Vincent	Sally	Jason	Thomas	Ming
Team Charter					R
Project Plan					R
Vision Document					R
Requirements Document	N	R			N
Design Document	G	R			
Weekly Meetings	S	R	S	S	S
Client Design, Code, Test	S		S	R	
Server Design, Code, Test	S		S	R	
User Documentation	S		S	R	
QA Documents					R
Test Documents	S	S	S	S	R
Packaging	R	A	A	R	A
Presentation	R	A	A	R	A

R - Responsible for Work

S - Supporting Work

A - Approval Required for Work Completion

G - Gate Reviewer

N - Notification Required

3. Process

3.1 Project Development Process

Clue-Less will use an incremental/iterative software development life cycle process with agile principles to lineup with the three required increments of the project. The programming language being used in the project is Java due to its portability and familiarity within the team. The team will use GitHub for version control and bug tracking throughout the development process.

3.2 Project Monitoring

The Clue-Less project will monitor progress by using GitHub project tools, in combination with our milestone dates and WBS structure, in order to keep track of where we are in the project. Project progress will be communicated within the team at our weekly standup. Due to the nature of the stakeholder, we do not anticipate providing project progress reports unless they are requested.

3.3 Project Quality Plan

- **Quality Assurance Plan**

  The A Team is committed to produce on-time and high-quality software. The team will work in the following three areas, Compliance Requirements, Clean Design and Code as well as Effective Quality Control, to ensure product quality[1]. To achieve the best quality, the quality team is not associated with developing group directly because it may be a conflict of interest. However, they will work together to ensure product quality so that they meet their schedule and requirements.

  a. Compliance Requirements

  Quality Assurance Plan requires the developing team has fully understanding of project requirements. The completeness and correctness understanding of requirements are the premise of the quality of all work products that follow.

b. Clean Design and Code

The team assesses every element of design to meet requirements. The source code shall conform industry best-practice programming principles and techniques. The quality team will test the code based on maintainability, reusability and understandability.

c. Effective Quality Control

The A team will use limited resources in order to achieve the highest likelihood of quality. The developing team may use game engines for project, such as Unity. The quality team will test whether they run in the most effective manner. The quality team will measure each existing process and its output to determine quality performance. Defects and errors will be collected and categorized. Each defect should be traced back to its cause.

- **Testing Plan**
  - Testing is performed within an Agile – Scrum software process during the Testing phase.

Weekly Scrum meetings include discussion of test results and addressing methods of improvement. The testing distribution show as follows:

10% - *Alpha Testing*. Identify all possible issues or defects from backlog items before releasing it into the customer.

10% - *Acceptance Testing*. Performed during demos and verifies whether the end to end flow of the system is as per customer requirements.

10% - *Black Box Testing*. Tests are based on the requirements and functionality.

10% - *White Box Testing.* Testing internal logic of an application's code.

20% - *Regression Testing*. Retesting an area where a bug was fixed.

20% - *System Testing.* The entire system is tested as per the requirements.

20% - *Unit Testing*. Testing of individual software components.

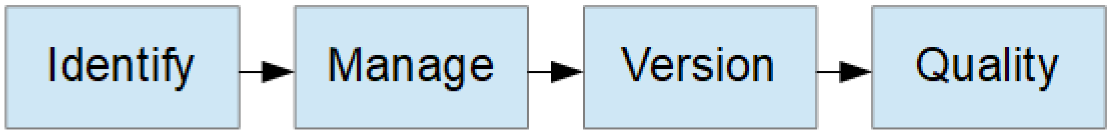
- List of big test events:
  1) Overall System Test before an incremental Release.
  2) Daily tests as changes are checked-in.
  3) Nightly automated system test.

- List of what will and will not be tested
  1) The following will be tested: Interface, System Input, System Output, and Functional units (arithmetic, logic, classes, file handling, data handling, miscellaneous/support functions).
  2) Database and software Development Application will not be tested.

- List of testing tools will be used
  1) Python test scripts
  2) JAVA test scripts
  3) Human computer interaction

Test functions will implement a 0,1 - Pass/Fail method of report.

- **Configuration Management Plan**
  - The process of identifying and managing changes to the software program is a 4-step process:



1) Identify all items that define the software configuration.
2) Manage changes to one or more of these items.
3) Facilitate the construction of different versions of an application.
4) Ensure that software quality is maintained as the configuration evolves over time.

- List of products will be under CM control and how they will be versioned
  1) Products under CM control include: Overall System Release, Interface, Functional units (arithmetic, logic, classes, file handling, data handling, miscellaneous/support functions) and Database.
  2) All components (functions) of the system are documented, and changes to that component is versioned with an Item description, Baseline version number of the release using Semver.org format. Ex: text_box_input V1.1.0, text_box_input V1.2.0, text_box_input V1.3.0...
  3) The Baseline program, or initial release of the Clue-Less program, is versioned using Semver.org format. Ex: Clue-Less V1, Clue-Less V2, ...

- Description of the process used to perform changes
  1) Changes are implemented through the Git Hub Repository of checking-out / checking-in code.
  2) Incremental changes are performed on a weekly basis during the Agile coding stage.

3) Backlog bug fixes are addressed during Scrum meetings and fixed during the weekly coding stage.

4. Assumptions & Constraints

- Constraints

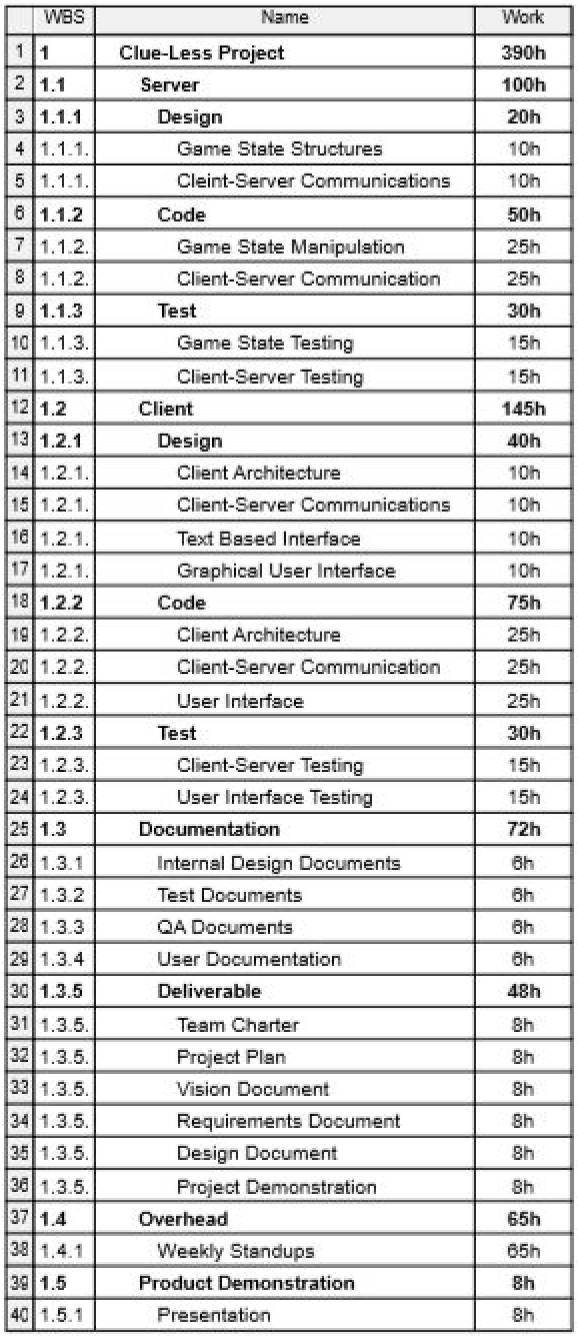
1) Due to the short development time of the Clue-Less project, we are constraint to the target system in terms of functionality. With more time, the features specified in the Dream increment could be implemented.
2) Due to the budget of the project team, we are constrained to using open source (free) software during the project.
3) The Clue-Less project must comply with academic integrity and policies of John Hopkins University.

- Assumptions

1) The Clue-Less project assumes the user is competent enough in the English language, as that is the language of choice used by the A-Team.
2) Work on the Clue-Less project will only be probono during the semester.
3) The user will not have accessibility concerns.

5. Project Estimates

5.1 Effort



	WBS	Name	Work
1	**1**	**Clue-Less Project**	**390h**
2	**1.1**	**Server**	**100h**
3	**1.1.1**	**Design**	**20h**
4	1.1.1.	Game State Structures	10h
5	1.1.1.	Cleint-Server Communications	10h
6	**1.1.2**	**Code**	**50h**
7	1.1.2.	Game State Manipulation	25h
8	1.1.2.	Client-Server Communication	25h
9	**1.1.3**	**Test**	**30h**
10	1.1.3.	Game State Testing	15h
11	1.1.3.	Client-Server Testing	15h
12	**1.2**	**Client**	**145h**
13	**1.2.1**	**Design**	**40h**
14	1.2.1.	Client Architecture	10h
15	1.2.1.	Client-Server Communications	10h
16	1.2.1.	Text Based Interface	10h
17	1.2.1.	Graphical User Interface	10h
18	**1.2.2**	**Code**	**75h**
19	1.2.2.	Client Architecture	25h
20	1.2.2.	Client-Server Communication	25h
21	1.2.2.	User Interface	25h
22	**1.2.3**	**Test**	**30h**
23	1.2.3.	Client-Server Testing	15h
24	1.2.3.	User Interface Testing	15h
25	**1.3**	**Documentation**	**72h**
26	1.3.1	Internal Design Documents	6h
27	1.3.2	Test Documents	6h
28	1.3.3	QA Documents	6h
29	1.3.4	User Documentation	6h
30	**1.3.5**	**Deliverable**	**48h**
31	1.3.5.	Team Charter	8h
32	1.3.5.	Project Plan	8h
33	1.3.5.	Vision Document	8h
34	1.3.5.	Requirements Document	8h
35	1.3.5.	Design Document	8h
36	1.3.5.	Project Demonstration	8h
37	**1.4**	**Overhead**	**65h**
38	1.4.1	Weekly Standups	65h
39	**1.5**	**Product Demonstration**	**8h**
40	1.5.1	Presentation	8h

Total Team Hours is 390 hours.

5.2 Cost

Category	Budget for Period in US Dollar
Human Resources(internal)	$0
Human Resources(external)	$0
Purchases(COTS)	$0
Equipment	$0
Premises	$0
Tools	$0
Travel Costs	$0
Training	$0
Review Activities	$0
Other	$0
Total	$0

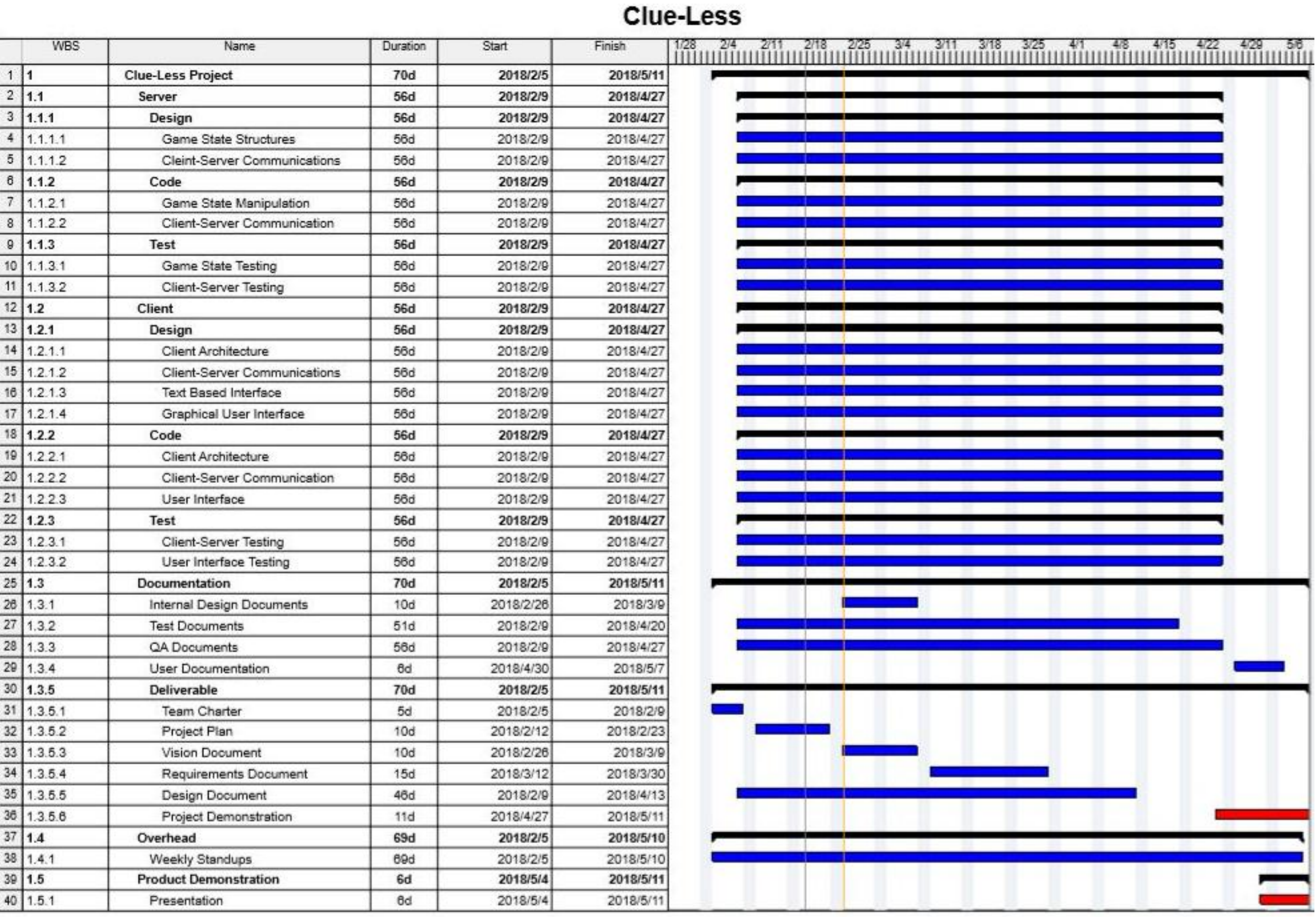
Total cost is $0. As a sign of good faith, we are undertaking this project at no cost to the customer, in hopes of securing larger and future efforts.

6. Project Schedule

6.1 Milestones

Milestone	ETA	Description
Design Started	2/9/2018	Initial design and development discussions begin based on customer's need statements
Requirements Started	2/9/2018	Requirements specification started
Vision Document Completed	3/9/2018	Vision document completed
Initial Design Completed Development Started	3/9/2018	Design revised based on requirements and code development starts
Requirements Completed	3/30/2018	Requirements completed
Skeletal Development Completed	4/1/2018	Minimal implementation of the architecture's sub-system and network communications
Minimal Product Completed	4/15/2018	Fully proves out design and requirement. All priority requirements will be satisfied.
Finial Design Completed	4/12/2018	Design documentation finalized.
Target Implementation Completed	4/27/2018	This is what is demonstrated on the last day of class.
Presentation Design Started	4/27/2018	Presentation documentation started.
Presentation Delivered	5/11/2018	Presentation delivered to customer.

6.2 Gantt Schedule



Clue-Less

	WBS	Name	Duration	Start	Finish
1	**1**	**Clue-Less Project**	**70d**	**2018/2/5**	**2018/5/11**
2	**1.1**	**Server**	**56d**	**2018/2/9**	**2018/4/27**
3	**1.1.1**	**Design**	**56d**	**2018/2/9**	**2018/4/27**
4	1.1.1.1	Game State Structures	56d	2018/2/9	2018/4/27
5	1.1.1.2	Cleint-Server Communications	56d	2018/2/9	2018/4/27
6	**1.1.2**	**Code**	**56d**	**2018/2/9**	**2018/4/27**
7	1.1.2.1	Game State Manipulation	56d	2018/2/9	2018/4/27
8	1.1.2.2	Client-Server Communication	56d	2018/2/9	2018/4/27
9	**1.1.3**	**Test**	**56d**	**2018/2/9**	**2018/4/27**
10	1.1.3.1	Game State Testing	56d	2018/2/9	2018/4/27
11	1.1.3.2	Client-Server Testing	56d	2018/2/9	2018/4/27
12	**1.2**	**Client**	**56d**	**2018/2/9**	**2018/4/27**
13	**1.2.1**	**Design**	**56d**	**2018/2/9**	**2018/4/27**
14	1.2.1.1	Client Architecture	56d	2018/2/9	2018/4/27
15	1.2.1.2	Client-Server Communications	56d	2018/2/9	2018/4/27
16	1.2.1.3	Text Based Interface	56d	2018/2/9	2018/4/27
17	1.2.1.4	Graphical User Interface	56d	2018/2/9	2018/4/27
18	**1.2.2**	**Code**	**56d**	**2018/2/9**	**2018/4/27**
19	1.2.2.1	Client Architecture	56d	2018/2/9	2018/4/27
20	1.2.2.2	Client-Server Communication	56d	2018/2/9	2018/4/27
21	1.2.2.3	User Interface	56d	2018/2/9	2018/4/27
22	**1.2.3**	**Test**	**56d**	**2018/2/9**	**2018/4/27**
23	1.2.3.1	Client-Server Testing	56d	2018/2/9	2018/4/27
24	1.2.3.2	User Interface Testing	56d	2018/2/9	2018/4/27
25	**1.3**	**Documentation**	**70d**	**2018/2/5**	**2018/5/11**
26	1.3.1	Internal Design Documents	10d	2018/2/26	2018/3/9
27	1.3.2	Test Documents	51d	2018/2/9	2018/4/20
28	1.3.3	QA Documents	56d	2018/2/9	2018/4/27
29	1.3.4	User Documentation	6d	2018/4/30	2018/5/7
30	**1.3.5**	**Deliverable**	**70d**	**2018/2/5**	**2018/5/11**
31	1.3.5.1	Team Charter	5d	2018/2/5	2018/2/9
32	1.3.5.2	Project Plan	10d	2018/2/12	2018/2/23
33	1.3.5.3	Vision Document	10d	2018/2/26	2018/3/9
34	1.3.5.4	Requirements Document	15d	2018/3/12	2018/3/30
35	1.3.5.5	Design Document	46d	2018/2/9	2018/4/13
36	1.3.5.6	Project Demonstration	11d	2018/4/27	2018/5/11
37	**1.4**	**Overhead**	**69d**	**2018/2/5**	**2018/5/10**
38	1.4.1	Weekly Standups	69d	2018/2/5	2018/5/10
39	**1.5**	**Product Demonstration**	**6d**	**2018/5/4**	**2018/5/11**
40	1.5.1	Presentation	6d	2018/5/4	2018/5/11

7. Risk Management

Clue-Less will incorporate a risk management plan to ensure the team can contain and mitigate project risks when they arise. Project Risks will be handled using the Risk Mitigation, Monitoring and Management Plan (RMMM)[1]. Each risk is documented individually using a Risk Information Sheet (shown below) with 3 objectives:

1) to assess whether predicted risks do occur,

2) to ensure that risk aversion steps defined for the risk are applied;

3) to collect information that can be used for future risk analysis.

Once a risk has been documented and the project has begun, risk mitigation and monitoring begins.

Risk Information Sheet			
Risk ID:	Date:	Prob:	Impact:
Description:			
Refinement/context:			
Mitigation/monitoring:			
Management/contingency plan/trigger:			
Current Status:			
Originator:		Assigned:	

List of project risks, their assessment, and how to handle them, as well as contingency plans

Risk Description	**Probability**	**Severity**	**Action**	**Status**	**Impact**
Clue-Less is not completed on time	10%	H	N/A	Monitoring	Project failure
Fail to meet follow the schedule	30%	M	Update the schedule	Monitoring	Behind the schedule
Architecture failed to support product	30%	M	Debug	Monitoring	More effort needed to be spent on debugging
Fail to complete the project in the estimated effort	40%	M	Update the schedule	Monitoring	More effort needed than expected

Severity: L - low, M - medium, H - high

Status: potential, monitoring, occurring, or eliminated.

8. Document Versions

The version of Project Plan may be updated during the project process.

Version	**Date**	**Notes**
1	2/25/2018	Initial Release

9. Reference

1. Pressman, R. S. and Maxim, B.R.(2010). *Software engineering: a practitioners approach.* New York: McGraw-Hill Higher Education.