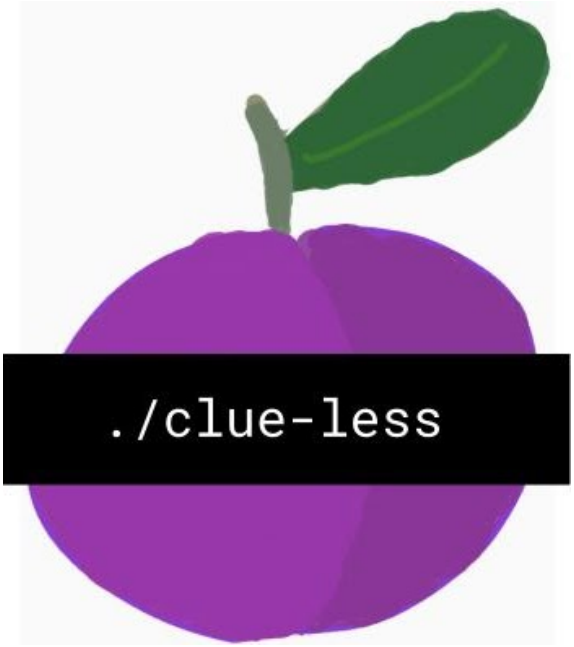
Johns Hopkins University



Project Plan

Professor Plum's Programmers

Foundations of Software Engineering
Section 81
Spring 2021

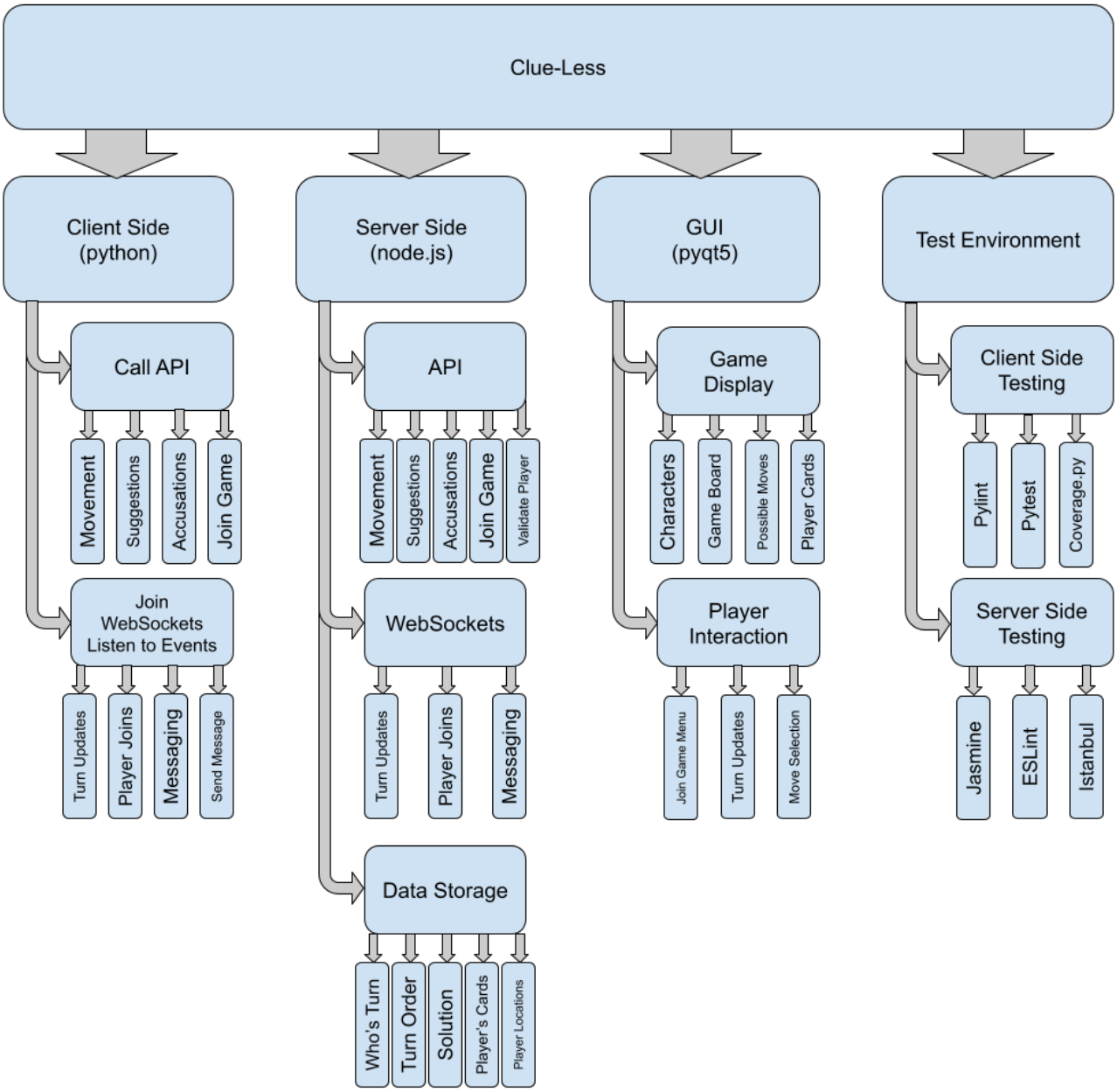
Project Description & Deliverables

Clueless is a simplified computer version of the classic Clue game. Clueless is a game that the player must download and have internet access to play. Clue is a game where Mr. Boddy is murdered in Tudor mansion and the suspects have to work together to determine the details of the crime before they too could be killed. The details include who, where, and how.

In Clueless there are nine rooms, six weapons, and six suspects. Players first choose one of the suspects. The suspects, weapons, and rooms that are not the solution are divided amongst the players. Then they take turns, where they can move between the rooms (which are separated by hallways) and make suggestions. Suggestions are when a player makes a guess at the solution to the crime and the other players (in turn order starting with the next player to move) get an opportunity to show a card to disprove the guess. In this version, once a player shows a card, the suggestion is considered void and no more players must show a card. At the end of a player's turn, they can make an accusation. An accusation is a final guess where if it is correct the player wins but if it is wrong, then the player is eliminated from the game. If a player is eliminated, then they still can disprove guesses, but they are unable to take their turn. Completion of the Clueless project will result in the following deliverables, with accompanying due dates:

	Deliverable	**Delivery Date**
1.	Team Charter	9 February 2021
2.	Project Plan	23 February 2021
3.	Vision Document	9 March 2021
4.	Requirements Document	16 March 2021
5.	Skeletal Increment	23 March 2021
6.	Design Document	6 April 2021
7.	Minimal Increment	23 April 2021
8.	Final Increment	11 May 2021

Work Breakdown Structure



Increment Features

Skeletal Increment Delivery

For the skeletal increment delivery, we will have a server setup for storing game state information. This information will include the solution to the clueless game. This server will also manage the player turns, dictating who's turn it is. For this iteration we only plan to provide players with the cards they are dealt and let them make accusations. We will not eliminate the players for a wrong guess in this iteration based on the limited functionality. We also plan to allow multiple players to connect to a single instance of the game. We are not planning on allowing more than one game to be happening concurrently in this stage of the product.

Minimal Increment Delivery

For the minimal increment delivery, we will have all the minimal viable features implemented for a text based variant of the game. This will have all the rules implemented along with the ability to move, suggest, or make an accusation. A wrong accusation will not eliminate the player so they can no longer take their turn, but they can still select a card to show other players when making suggestions. Additionally the product will have a testing structure and framework set up.

Target Increment Delivery

For the target increment delivery, we will add a graphical user interface (GUI) to work with the game. This will display the game board, options, and a way to interact with the game. This includes possible movements, making suggestions, making accusations, and picking a card to show another player. The user should be able to use the GUI for all their interactions with the game. There should also be enough information to understand what is currently happening in the game, like who's turn it is, where they moved, what they suggested, and who showed them a card. Testing will be complete and have a high amount of coverage of non-GUI code.

Dream Increment Delivery

Our dream increment delivery consists of features we would like to have, but don't foresee having time to complete in the project timeline without adding additional risk. This increment would allow the server to be able to host more than one game on the server concurrently. Messaging will be allowed between the players. The GUI will feature real cards from the original clue game. Player names should be customizable. The GUI will also allow the player to move via dragging and dropping their token. Lastly, we would like to allow the board to be rearranged, so rooms can be in different positions.

Project Schedule

Below is the initial project schedule in table format. Appendix A contains a full Gantt chart of the project schedule, including task dependencies.

Name	Begin date	End date
Project Plan	2/17/21	2/23/21
Vision Document	2/24/21	3/9/21
Requirements	3/10/21	3/16/21
Design Document	3/17/21	4/6/21
Skeletal Presentation	3/22/21	3/23/21
Minimal Presentation	4/12/21	4/13/21
Final Presentation	5/10/21	5/11/21
Server Prototype	2/24/21	3/7/21
Client Prototype	2/24/21	3/7/21
Single Client Join	3/8/21	3/11/21
Multi Client Join	3/12/21	3/17/21
Turn Structure	3/18/21	3/21/21
Accusation T/F	3/8/21	3/14/21
Provide Card Info	3/8/21	3/14/21
Suggestions	3/15/21	3/21/21
Suggestion Response	3/22/21	3/28/21
Player Elimination	3/29/21	3/31/21
Board Structure	3/8/21	3/14/21
Player Positions	3/15/21	3/21/21
Find Allowed Moves	3/22/21	3/26/21
Websocket Connections	3/22/21	3/28/21
Player Up Alerts	3/29/21	4/4/21
Full Player Turn	4/5/21	4/11/21
Concurrent Games	4/12/21	4/25/21
GUI Prototype	4/12/21	4/21/21
Gameboard GUI	4/22/21	4/27/21
Game Piece GUI	4/28/21	5/2/21
Interactive Movement	5/3/21	5/9/21
Rearrangeable Board	5/3/21	5/7/21
Custom Names	5/3/21	5/7/21
Card GUI	4/22/21	4/27/21
Clue Tracker GUI	4/28/21	5/2/21
Player Messaging	4/22/21	4/26/21

Risk Assessment & Mitigation Plan

Risk 1

Work amount exceeds small team capabilities

Mitigation

We understand as a group that each member's contributions will need to be substantial in each phase and deliverable. We will keep each other motivated and up-to-speed with any questions and concerns often and start well ahead of deadlines to ease the stress of last minute work. Our team chemistry will be critical to our success and we will strive to maintain a positive and lighthearted rapport throughout the semester.

Risk 2

Illness among team members

Mitigation

In the unfortunate event that a team member falls sick, we will have planned areas of that member's work to delegate to the others. In general, we will cover each others' responsibilities when necessary in order to meet all important project deadlines. To facilitate this, we will start collective work on phases of the project well in advance and raise any red flags as soon as possible.

Risk 3

Unfamiliarity with Project Tools

Mitigation

Several project team members do not have experience using node.js for server-side programming development. In order to mitigate the risks of lack of knowledge, the team will rely on knowledge sharing from more experienced team members, and weigh switching to other development techniques if the knowledge gap becomes problematic. Team members who are less knowledgeable in node will also help reduce this risk by doing their own research on the side to become more familiar with the tools we are using.

Quality Plan

Quality Assurance

We will employ several standard methods to ensure we sustain a desirable level of quality throughout our codebase. Linting will be performed regularly to reduce error potential and overall code quality, likely through a chosen lint software. Code reviews will be required prior to merging, as described in Configuration Management below. We will use a bug tracking system for easy access to reported software issues throughout development. Finally, we will perform manual black box integration testing prior to each delivery to test our code against external factors that could compromise our code from functioning.

Configuration Management

To manage our project source code, we will use an Organization GitHub repository with a simple Main/Develop/Feature branch Git workflow. Commits to the Main branch will be limited to documentation uploads for team deliverables, and merges from the second branch, Develop. The Develop branch will be a continuous branch that contains the currently implemented features for the project, and is pulled into Main when a Version is generated, after testing has verified that it is bug-free. From Develop, developers will create branches to implement individual features. Once a feature is complete, a Pull Request will be generated by the main developer of the feature for full team input before merging into Develop. Once all comments have been addressed by the devs or accepted by the commenters, the feature branch will be merged into Develop, and removed. The Main branch will be protected from commits, and the Develop branch will be protected to require that Pull Requests are complete before merging. Figure 1 gives an example Git network diagram for this type of workflow, using “Master” as the Main branch.

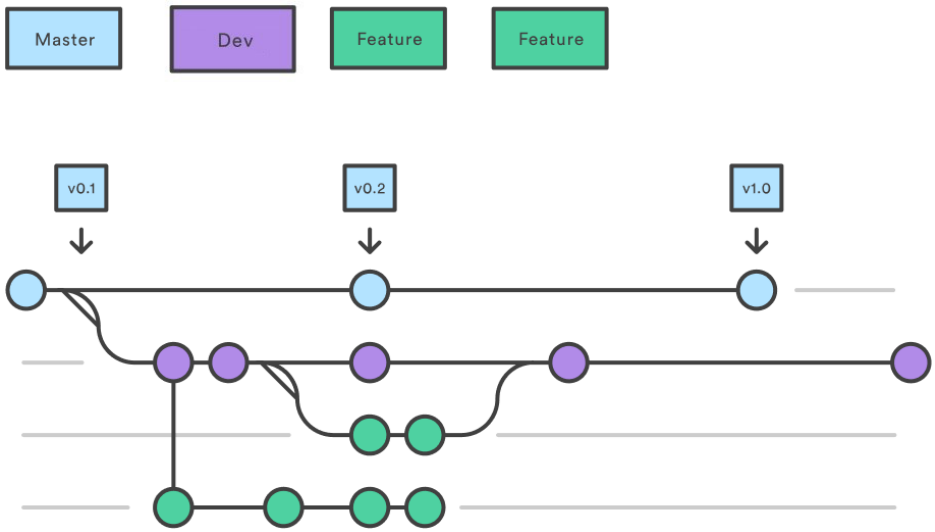


Figure 1: Master/Dev/Feature Git Workflow [1]

Feature branches will be generated to address Issues, which will be tracked using a GitHub project board. Issues in the board will be sorted in order of priority, and the developer leading the implementation of the feature/bug fix will be assigned to the Issue. Commits should include links to the issue they are addressing by including the issue number in the commit message, and issues will be closed by commit keywords when merging the branch.

Testing

Unit tests will be generated alongside code for both the server and client side development. Developers will be responsible for updating these tests as new features are implemented in order to maintain full coverage for unit tests. Unit tests will be set up to run using GitHub Actions whenever a Pull Request is generated for feature branches into Develop, and on pushes to both the Main or Develop branches. The existing tests must pass before features can be implemented into the Develop branch. In addition, Developers will be responsible for running Linter tests on their code before merging into the Develop branch in order to verify that code follows adequate professional standards.

Appendix A: Gantt Chart

Name	Begin date	End date
Project Plan	2/17/21	2/23/21
Vision Document	2/24/21	3/9/21
Requirements	3/10/21	3/16/21
Design Document	3/17/21	4/6/21
Skeletal Presentation	3/22/21	3/23/21
Minimal Presentation	4/12/21	4/13/21
Final Presentation	5/10/21	5/11/21
Server Prototype	2/24/21	3/7/21
Client Prototype	2/24/21	3/7/21
Single Client Join	3/8/21	3/11/21
Multi Client Join	3/12/21	3/17/21
Turn Structure	3/18/21	3/21/21
Accusation T/F	3/8/21	3/14/21
Provide Card Info	3/8/21	3/14/21
Suggestions	3/15/21	3/21/21
Suggestion Response	3/22/21	3/28/21
Player Elimenation	3/29/21	3/31/21
Board Structure	3/8/21	3/14/21
Player Positions	3/15/21	3/21/21
Find Allowed Moves	3/22/21	3/26/21
Websocket Connections	3/22/21	3/28/21
Player Up Alerts	3/29/21	4/4/21
Full Player Turn	4/5/21	4/11/21
Concurrent Games	4/12/21	4/25/21
GUI Prototype	4/12/21	4/21/21
Gameboard GUI	4/22/21	4/27/21
Game Piece GUI	4/28/21	5/2/21
Interactive Movement	5/3/21	5/9/21
Rearrageable Board	5/3/21	5/7/21
Custom Names	5/3/21	5/7/21
Card GUI	4/22/21	4/27/21
Clue Tracker GUI	4/28/21	5/2/21
Player Messaging	4/22/21	4/26/21

Sources

[1] Negar Jamalifard. *Use Git More Efficiently: A Simple Git Workflow*. 6 June 2019. <https://miro.medium.com/max/1867/1*BgjBdnC82OR5sh73nvxtxQ.png> Digital Image.