**(404) Team Name Not Found**

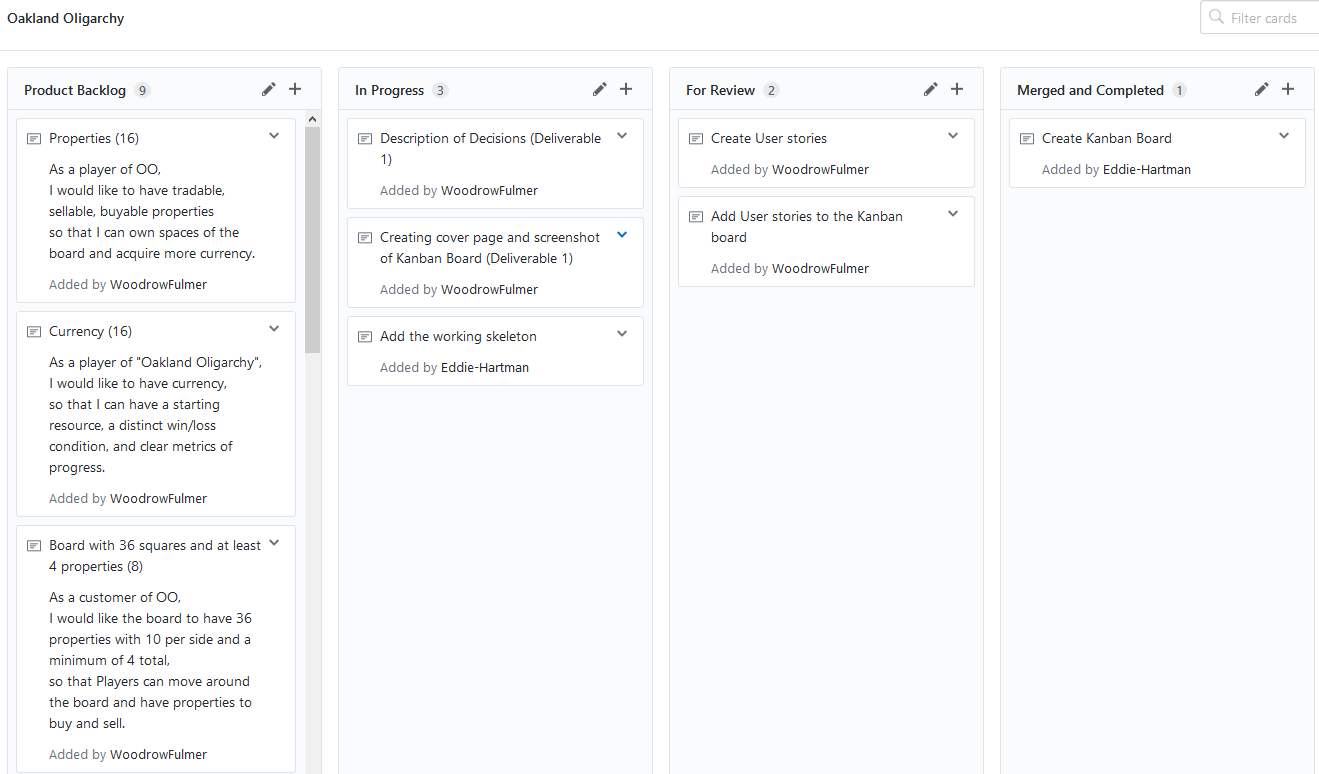
Eddie Hartman Eddie-Hartman

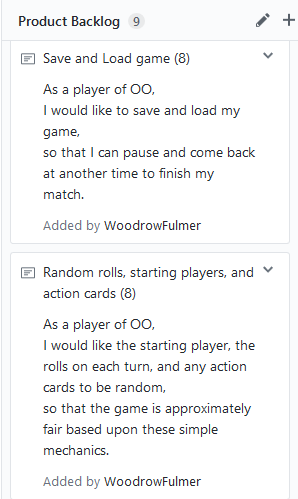
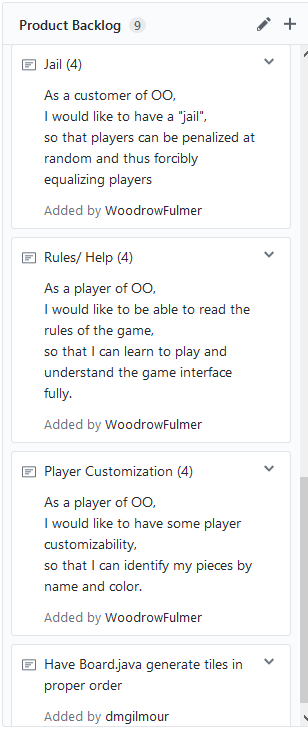
David Haskell dah121

Dan Gilmour dmgilmour

Woodrow Fulmer (Scrum Master) WoodrowFulmer

8 JUNE 2017



# Design Decisions

During this sprint, we made decisions on what significant features to include, major code layout, and the layout for the user interface.

We chose our code layout to include three main objects, Window, Player, and Properties, as most features of the game can be categorized into one of the three. Window holds all our code for the user interface including the frame, all the panels, and currently all the action listeners, but the final location for these is still being discussed. Player stores all information about a given player including money, an array of properties they own, their id, their current location on the board, parameters for being in jail, etc. Properties is an array of all properties on the board, information on who owns them, how much they are worth, how much rent they charge, whether they belong to a monopoly, whether they have improvements, etc.

For our user interface, we chose to have four sections, the top menu, the board, the player status panel, and the action panel. The top menu will include features such as save game, load game, instructions, and options. The board panel will have all the properties arranged in the shape of the board as clickable buttons that will display information about the property. The status panel will list all players with their money as clickable buttons that will display more information on click, such as total worth, properties owned, etc. The actions panel will have buttons for every action that a player can take on their turn, such as roll, trade, end turn, purchase upgrades, etc.

Because we could not decide on a color scheme, Dan just decided to make the color scheme random each time.

# Interactions with Customer

Interactions with the customer/user took place during an in-person interview on May 25th. Our notes are summarized in “requirements.md” and “designReqs.md” located in the project repository. In addition, these notes were used to generate the user stories found in “UserStories.md” or on the project board. We plan to condense and reorganize these notes during the next sprint to make it easier to backlog.

# Prioritization

To prioritize user stories, we categorized each as either primary, secondary, or low-priority features. Primary features included currency, properties, and placing properties on the board as they are core features that make Oligopoly playable. Next, secondary features were comprised of jail, randomization, and saving games because they are necessary features, but the game is still playable without them. Low-priority items were help documentation and player customization as they are completely unneeded to play the game and rather just items proposed by the user or our team.

# Points of Debate

We were unsure how to let classes such as our ActionListeners access the information they need to. For instance, with trading we want a player to be able to initiate a trade from the actions panel on the left and to be able to start a trade by clicking on a property, then clicking trade under its expanded information.

We disagreed about how much we should be abstracting our code. For instance, one team member thought we should call a function that would make the frame, while another thought we could do this in the main function.

# Team Effectiveness

Overall, our use of GitHub as a development tool has been rather relaxed. This led to some conflicting commits. While these conflicts were easily resolved, we would like to use branches more effectively in future sprints. In addition, we should make better use of the project board as a means of designating who is working on what (and when).

We decided as a team to use the Slack messaging service. However, due to errors with its notification system we have been having difficulty communicating promptly. In the future, we would like to seriously discuss resolving this problem or finding a new means of communication. In addition, we can partially solve this problem by agreeing to all check Slack with increased frequency.

We tried splitting our team of four into two groups of two for planning and coding. This worked well because both members always had something to do and could be productive when planning, or coding. When planning, it was helpful to have someone else to give a second perspective on our ideas, and having only two members made conversations go quicker than if they had been with the whole team. When coding, having a second perspective and someone to answer questions is useful, and so far with dividing labor, we have always been able to find two disjoint parts of the code that members can work on in tangent without interfering, but expect splitting work much further will be difficult.