ReadySelectPlay Summary Report

Jayanth Gangadhar, Matthew Hantsbarger, Parshva Shah

Problem:

The problem we wanted to solve was that selecting a board game in a group can be a difficult endeavor. Multiple factors can complicate the process, such as keeping track of everybody's preferences, negotiating different wants and needs, and explaining games that people may be unfamiliar with. Deciding what board game to play at a party can be a hassle for participants. Narrowing down which games can be played while taking into account everyone's preferences can be difficult to do in a timely fashion. Selecting from a huge list of available games can take a long time as people have to stop and explain several games to other players not familiar with them, and trying to negotiate clashing preferences can lead to conflict and indecision.Our application was developed with an intent to ease the process of communication, filtering and selection of the game among a huge list of games that is to be played during the game night.

The primary user of our application is the party host - someone who hosts board game nights and has to cater to the preferences of all other board game players while filtering from a list of board games. Secondary users are the other attendees of the board game night who voice their preferences to the host and along with the party host actively take part in the voting process.

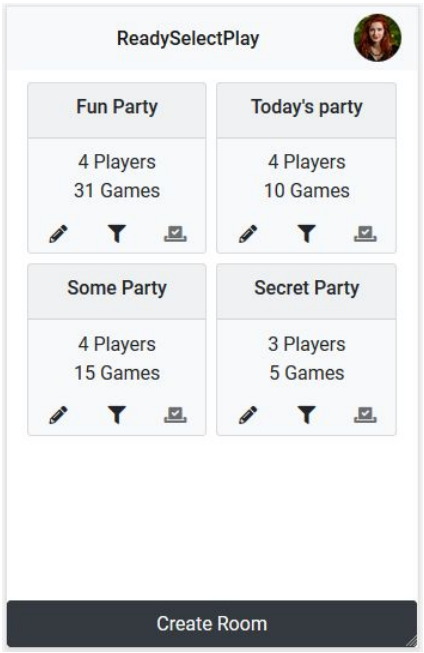
The user tasks can be narrowed down to three main task that we identified:

1. Organizing the game night: This is done prior to the game night where the host creates a game room and adds the potential attendees of the game night and games he/she owns. The attendees can also add the games which they will be bringing to the game night.
2. Filtering games: This is done during the game night. Once all the games have been added, the list of games can be narrowed down by taking into account the different user preferences and applying them on the filter page which contains an extensive list of filters including duration, user count, complexity, categories and mechanics.
3. Voting: The final step is voting. Once the filtering is completed all users are presented with the filtered list of the games and the users can cast their votes for the games of their preference. Once everyone has voted, the host ends the voting and the winning game is displayed where the users can view information of the game which includes details such as a description, player count, category and the game mechanics.

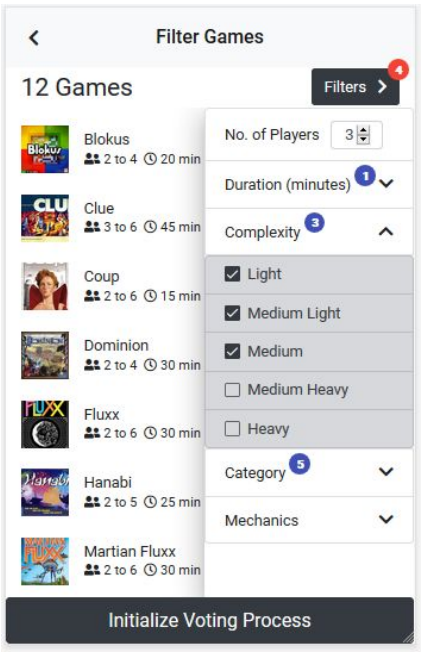
Design:

To solve these problems we decided to create an application that would facilitate the game selection process for our users. The final design of our application involves a "room" system, where users can create and be added to rooms with other users and their collective games can be listed. They can then narrow down the list of games based on a variety of filters, including game duration, the number of players, complexity on a five-level scale, category of game, and mechanics the game uses. We also include a voting process for users where they can vote on a filtered list of games as a method for groups to end up with a game that a majority of the group want to play.

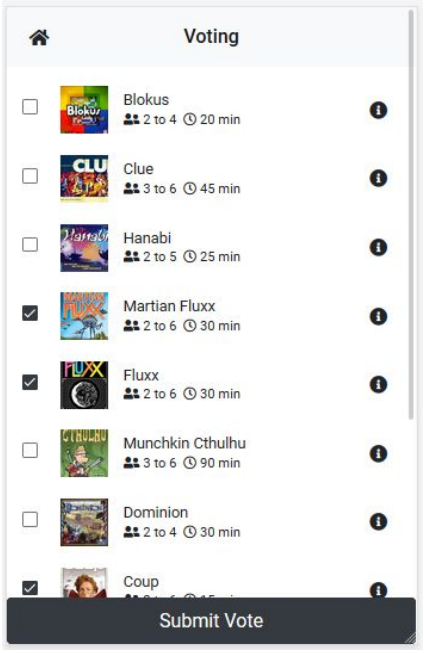
We chose to use a room system for a few reasons. The first was that we anticipated that users would want to use the app for multiple game nights and so providing a way for users to save their friend groups and accompanying games was paramount to our initial designs. The



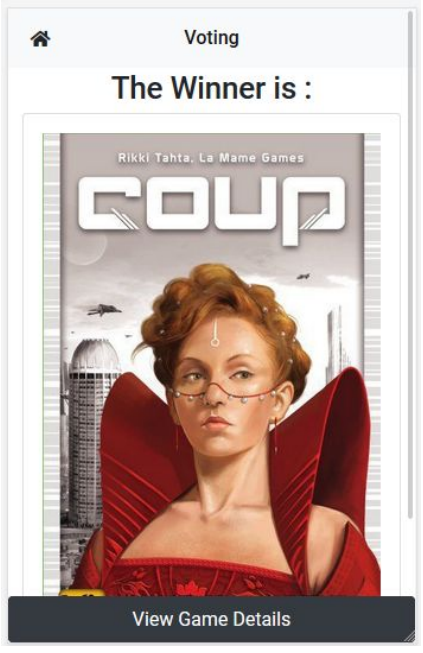
Home page with several rooms.



Filter page with the filters menu opened and several filters applied.



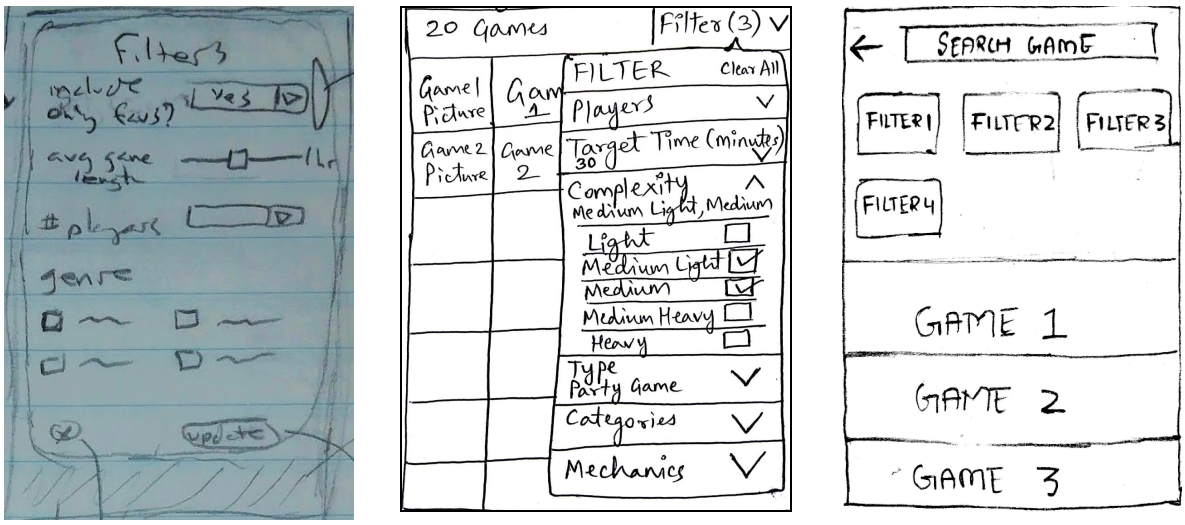
Voting screen where user can vote for multiple games to play.



The result screen with the game that received the most votes and the ability to view more information about the game.

second was that creating a list of games a user-owned was going to be a time-consuming task that only really needed to be performed once, whereas other filters were dependant on the preferences or limitations of the users at that particular event. The third was that it allowed us the flexibility to explore other features for facilitating communication, such as the potential to plan

events with reminders or the ability to chat with other participants in a group chat room. Ultimately we decided against those features that aided asynchronous digital communication that was already covered by other popular applications such as instant messaging apps or group event planners because they would be expensive in terms of development time and would not easily replace applications that users already use and are familiar with. At multiple stages of our design, we elected to pursue or not pursue certain features based on the reasoning that we weren't seeking to supplant already effective methods of communication, but rather augment them. We only sought to implement features that we felt were not addressed sufficiently by other technologies or methods. For example, we added features like the ability to look up game details
because we could implement them in a way that was more convenient for users than the alternatives, such as looking at the boxes or rules for several games or searching each game online for information.

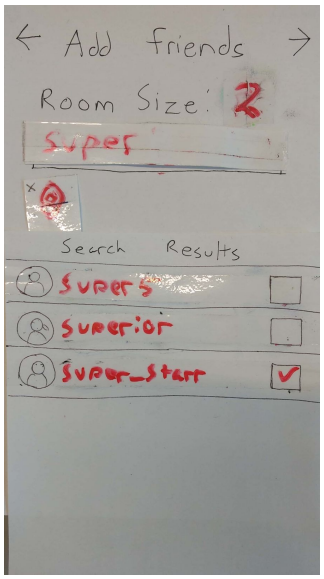


Our three initial filter menu designs. We eventually settled on the second design for our application

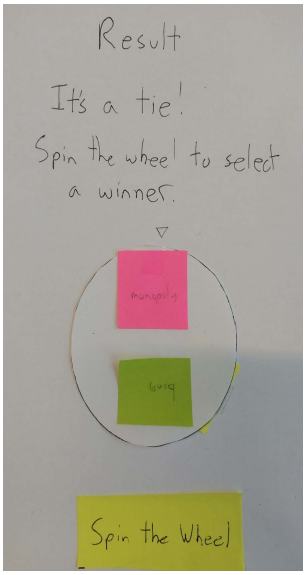
The system of filtering games was by far the aspect of our design that received the most scrutiny from our design process because it was the main feature that was intended to solve the issues we were addressing. It served as a way to help partygoers keep track of each other's wants and needs with regards to what games they could and wanted to play and inform each member of what games fit those criteria and details about those games if the user so wished. The decision that the list of games visibly changed as filters were added or modified was made because we wanted to keep a user constantly informed about what games still met their criteria, especially if they filtered based on wants that would drastically cut down on the number of available games.

The voting feature was adjusted quite a bit in design. This came partially from the acknowledgment that it would be practically impossible to design a one-size-fits-all method for helping groups settle on a single game. Our previous research into Group Decision-Making showed us that there were multiple methods that groups used to create consensus and since the intent was to come to a decision that maximized individual satisfaction for every member of the group there was no way we could have a voting method where the final result would have a 100% satisfaction rate for every participant. Initially, we decided on a single vote system where every user had a single vote with the reasoning that implementing the simplest method would

have the broadest applicability and ease of use for users. After our paper prototype we decided to change it to allow for multiple voting, as a way to give users more freedom in having “soft” preferences without ballooning the complexity of the voting system, which is a concern we had with alternatives like ranked voting allowing users to express levels of preference or scoring equalizers intended to give larger bonuses to users who had not had the game they voted for selected in previous votes.



Paper prototype Add Friends screen. The forward arrow and room size indicator were changed based on feedback.



A spinning wheel screen for settling ties in voting. Was removed in later iterations because it created negative user perception and wasn't necessary for the process of selecting a game.

A large number of changes were made in response to user feedback and our observation from our paper prototype. We made the conscious decision to remove extraneous information, primarily in removing the “room size” indicator and the tiebreaker wheel. The tiebreaker wheel was also removed because testers didn’t like the application creating “winners” and “losers” and drawing attention to it. We also made sure to create more consistency in buttons that transitioned from page to page, choosing to drop the forward arrow in exchange for labeled buttons at the bottom of the screen indicating the action performed or the page it led to. This was also where we changed to the “multiple voting” system.

Another design considerations we had to address when adapting our paper prototype into a software prototype was the decision to make category and mechanic filters work in a way where a game only needed to match at least one of the selected categories or mechanics, since games only fell under a single category and since we predicted that users would want to view all games that included any selected mechanics, as opposed to only games that included all the selected mechanics. This wasn’t an design decision we had anticipated when creating the paper prototype but something that arose when performing internal testing of our software prototype.

Our major design decisions from Heuristic Evaluation feedback centered around making account implementation more obvious and improving signposting of how the pages linked together, this time through addressing toast notifications. We added usernames under their profile icons in the add users screen and added the “users” profile icon to the home page to better communicate that the user was logged into their account and to give them more visibility in ensuring they had selected their friends. Updating toast notifications to improve feedback

centered on indicating how to start voting when voting was not in progress, and reducing the duration of the message to not get in the way of other functionality.

Implementation:

ReadySelectPlay was developed as a MEAN stack web application (Mongo, Express, Angular, Node) and is hosted on https://ready-select-play.herokuapp.com. While deciding on which technology stack to use for developing ReadySelectPlay, the viable options were:

- MEAN stack
- Android Studio
- Ionic framework

Using MEAN stack was the final collective decision because of the following reasons:

- We were more familiar with MEAN stack as compared to the other two.
- MEAN stack is used for developing web applications whereas the other two were for developing mobile applications. Users would have needed to download and install mobile applications before they could use it.
- There is no need to take care of the platform dependency of the application if it is running as a MEAN one. The OS platform (Android/iOS) affects the implementation and support in case of Android Studio and Ionic framework.

About the components of MEAN stack:

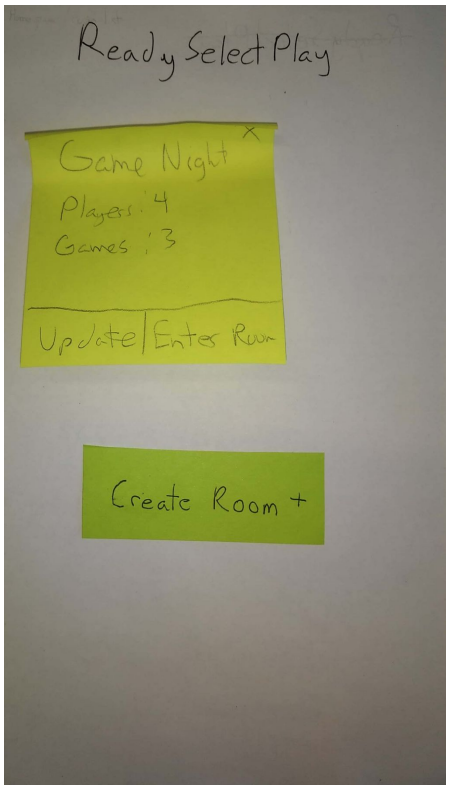
- Mongo (MongoDB) is a database where we store information of our users, games, and rooms.
- Express (Express.js) is a JavaScript library which acts as a middle tier and handles communication between the frontend and the backend.
- Angular (AngularJS) is a JavaScript framework which helps to develop the frontend of the application.
- Node (Node.js) is a server environment which communicates with the database to resolve API requests and acts a the backend of the application.

We used the BoardGameGeek API - http://bgg-json.azurewebsites.net to get the details of about 60 games which we added to our MongoDB manually. There were some downsides of using this API with respect to the requirements of our application:

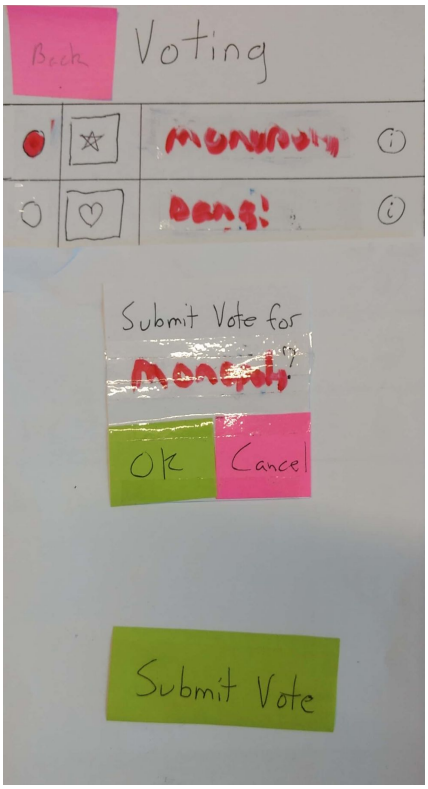
- Obtaining detail for a particular game using the API had to be done one at a time.
- The API did not provide the category and complexity for games, meaning we had to add them manually to our database.
- The API did not facilitate a proper search feature where users could provide the search keyword and receive a list of matching games from its database.
- The descriptions of some games were too long which were not suitable for reading when at a party. Also, it would have been better if the API also facilitated links of youtube video tutorials of the games as most of the people at a party may be unaware of how to play the game.

We designed our application with a mobile-first and responsive design due to the obvious fact that the users in a party would have easy access to their mobile phones and tablets. They can use any mobile browser and go to ReadySelectPlay website and they are good to go. As far as users who wish to use a PC / laptop is concerned, they can also just go to the URL, but the interface won't be as comfortable to interact with as it would be if they switch to a mobile view in their browser. Similarly, switching to a Desktop site while opening the website on a mobile phone would affect the usability of the interface.

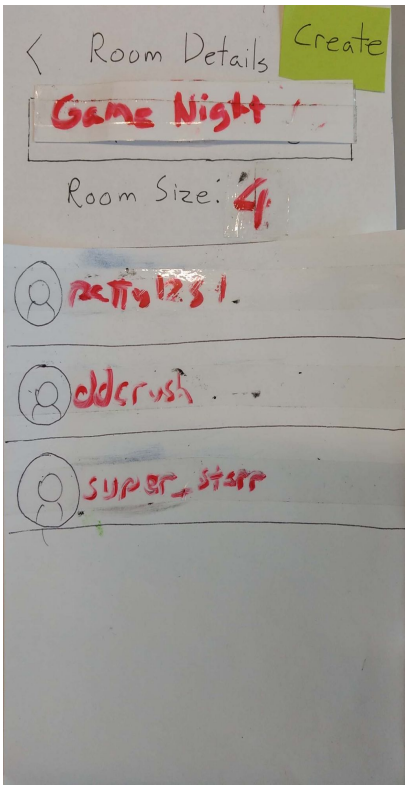
Evaluation:
Paper Prototype lessons:



Home Page



Voting Page



Room Details Page

1. Removing extraneous information
    - We eliminated “Room size” in Room Details page which users interpreted as being an input field to “Enter Room Size”.
    - We removed the popup to confirm user’s votes because we deemed it was not a critical decision that required the user to check.
    - We removed the spinning wheel to randomly select a game as winner in cases where a tie occurred.
2. Consistency and standards
    - There were inconsistencies in navigation using right arrows, next and submit buttons. That led to confusion for users in navigation and overall app structure.
3. Preventing User Errors
    - There was a lack of clarity with the words used (“Update” and “Enter Room”) on the home page for editing room, filtering games and voting.
    - We changed the edit room process to save room changes on the go without waiting for completion of whole process of room creation to ensure users would not lose any changes.
4. Allow for more user freedom and expression

    - User could vote only for a single game, but in the interest of allowing for more nuance in expression of user preference we decided that multiple voting was a more effective alternative.
5. Visibility of system status
    - There was no list of games at the room description page because of which users felt that the games did not get added and gave them misinformation about the system status.

Heuristic Evaluation lessons:

- In our initial software prototype we didn't have text corresponding to avatars of members and games. We found that "recognition is better than recall" and placed an avatar on the home screen to remind users that they were "logged in" to their account. We also added usernames underneath added friends so that users did not have to rely on just the icon alone.
- We didn't have a specific order in which friends and games were displayed while adding them to a room which made it difficult for the users to find and add them manually.
- The duration of the toast notification to alert user that voting process has not been initialized was longer than required, in the interest of implementing "aesthetic and minimalist design" we shortened the duration and clarified the notification.

User Testing lessons:

- We didn't have a "Reset" feature for categories and mechanics filters which meant that users had to manually search for the filters applied and uncheck them one by one if they wanted to clear the filter.

As far as future scope is concerned, the following tasks would be added to the existing application if we had more resources:

- Live refreshing on the Filters page so that everyone can observe the filters applied by the party host in real-time.
- Live refreshing on the Voting count page to show the exact live count of the number of people who submitted their votes to aid the party host in the decision of ending the voting process.
- The functionality of login and sessions to differentiate between party host and other party members. This would prevent other party members from applying filters and initiating and ending the voting process.
- The functionality to add your own game which can be your own version of some existing game or just some game missing in the database.
- While creating a new room, at the Room Details page, there can be an edit option corresponding to the list of Members and Games to direct them to the edit page of the corresponding list.

Reflection:

Our main take-away from the different testing phases was that even with a good reason for a design decision, If the user doesn't understand the function, it needs to change. We had features (for example: User count in the rooms) which we thought would help the users give a better understanding of the different tasks but during testing we realised that some of the features did not aid the users and instead ended up complicating the system for them.

We had difficulty getting the application's overall structure in all our testing. So one thing we would have definitely changed in our testing phases was having the entire structure of the application ready for the users to test. We could have additionally performed quantitative testing to mainly measure the differences in time needed to filter and choose the game that would be played during the game night with and without our application to have better statistics to reflect the impact of our system.

We could have come up with better statistics to reflect the success/failure of our system if we had more time for user testing, especially so if we could perform user testing with larger and also diverse group of testers. We ended up with testing only with grad students who were not all very active board game players. We feel the results could have been a better reflection had we tested with user groups who play board games frequently and also bigger user groups (10-15 people) to see how our application eased the communication and decision process. We were not able to find a "real" game night party that was already happening that we could observe, and we ended up simulating a game-night with a smaller group for the purposes of testing.