GameNight Mobile Application

by

Reggie Philpott and Kelsey Trusty

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

The author grants to the School of Information Technology permission to reproduce and distribute copies of this document in whole or in part.

Reggie Philpott	4/15/19
Kelsey Trusty	4/15/19
Abdou Fall, Faculty Advisor	4/15/19

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2019

Acknowledgments

The GameNight Mobile Application team would like to personally thank Professor James Scott and Professor Abdou Fall. Their encouragement and enthusiasm throughout this project helped push us to the finish line.

We would also like to thank Aaron Sapp for taking time to mentor us on how to develop our mobile application. With the help of Aaron, we were able to expand our knowledge within mobile applications which lead to us completing the app.

We would finally like to thank our families who were there for us throughout this project. It was not easy, and we had a lot of learning to complete. Thank you all for never giving up on us and our dream application, GameNight.

Table of Contents

List of Figures

Abstract

In 2015, the sale of tabletop games by small publishers topped $1 billion in the United States and Canada for the first time.[1] In 2017, this total skyrocketed above $1.5 billion.[2] In an age where digital screens are often uncomfortably pervasive, people have a growing desire for human contact away from technology. The explosive popularity of tabletop gaming showcases this very clearly. Currently, an average of over 10 tabletop games are being published each day, a staggering number.[3] Due to the ever-growing number of choices, it can be very difficult to select a game that everyone will enjoy playing. To address this issue, the GameNight mobile application helps the members of a group select a range of games from their collection by breaking games down into their mechanical components, and then searching for a common interest for the group by looking for a commonly occurring mechanic. The application then suggests games that utilize this mechanic, providing an experience that everyone at the table will enjoy. The application can be used in multiple ways: for example, a user can designate games they already own as being in their collection, or they can designate games they are considering purchasing, using it as a purchasing aid. GameNight saves time and energy that can be better spent getting straight to gaming!

[1] https://icv2.com/articles/news/view/35150/hobby-games-market-nearly-1-2-billion

[2] https://icv2.com/articles/news/view/41016/hobby-games-top-1-5-billion

[3] https://goo.gl/9gr8Nk

Introduction

Problem

With so many tabletop games being published today in a wide variety of genres, finding the right one can be overwhelming. Within a friend group, it can be a challenge to pick a game from someone's collection that will appeal strongly to everyone. As the range of games becomes broader and broader, it is a chore to simply sort through all the possibilities. Additionally, there is very little published guidance when it comes to what games a player might enjoy. Because of this, many players end up having subpar experiences when being introduced to new games, making them less likely to want to try new games in the future. When a group finds a game that they all like, this can also lead to a lack of desire to try new games out as well, causing people to miss out on other enjoyable experiences. Ultimately, these issues hurt the tabletop gaming hobby as a whole.

Solution

GameNight fosters a positive tabletop gaming experience by analyzing a given collection of games, and providing suggestions on what a given group of players is likely to enjoy when they play together. This collection of games can come from one player's collection or from multiple players' collections, and can also include games that someone in the group is considering purchasing. It does this by distilling games down to their mechanical level, and then searching for a common mechanic among the collections of the group members. The application then provides suggestions based on this information. It then synthesizes data about the games and about the game group to help people avoid negative experiences and reduce deliberation time when choosing a game.

Project Goals/Brief Methodology

The goal of this project was to produce an iPhone application that will assist users in planning board game nights by taking a group's collective interests into account. It is easy to use for all technology comfort levels. Due to our relative inexperience with development, we chose this project to challenge ourselves and learn a new skill. We spent the fall semester collaborating and communicating in order to deliver a mobile application during the spring semester.

Overview

The following report details the GameNight mobile application project from conception to completion. It includes these sections: Discussion of the solution, design objectives, methodology/technical approach; an inclusion of the two user profiles; the use case diagram; the project Gantt chart; analysis of problems encountered; and a conclusion.

Discussion

Project Concept and Solution

Our idea for GameNight came from Reggie Philpott's personal experience as a tabletop gamer. Oftentimes, because games can be so different in their mechanics and theme, he observed that people sometimes had negative experiences with games, and different people don't enjoy all games in the same way. This is true within established gaming groups, and also when it comes to introducing people to a game for the first time. Especially with people who are newer to tabletop gaming, it is crucial to provide them with a positive experience to keep them from being turned off to new gaming experiences in general. Reggie has found that when most people say they don't care for tabletop games, the truth is that they simply haven't yet found games that appeal to them. The GameNight mobile application addresses these issues for both newcomers and established gamers.

The goal of GameNight is to foster a positive tabletop gaming experience by selecting a game for a group that will appeal to all based on the shared preferences of the players. It allows users to add the games in their collection, as well as games they are considering adding to their collection, to their profile. Then, it examines the mechanics of the games in order to make its recommendations. The application looks at commonly occurring mechanics, and it uses this data to look for a trend among a group who are going to get together to play one or more games. In doing so, users can form a gaming group on the application to allow GameNight to provide its suggestions. The main function of GameNight is for the user to enter the length of time and the specific attendees for their upcoming gaming session. GameNight then uses its collected data about users and games to generate suggestions for what game or games to play based on the

interests of the attendees. This helps users have more positive experiences, and reduce deliberation time when choosing a game to play.

Design Objectives

GameNight is a mobile application for iPhones. Figure 1: Game Entries in Database illustrates games entries that are stored in the Firebase database in JSON format. The user interface connects to a database, leveraging the Google Firebase technology for storing data in JSON format as well as for the authentication of users. This allows the application to store user profile entries and tabletop game entries.

Figure 1: Game Entries in Database

Because people of all ages play and enjoy tabletop games, the goal was to make the user interface as accessible and friendly as possible. A user can easily tell GameNight what games

they want to designate as being in their collection at a given time. This gives a baseline picture of what other types of games that user might enjoy, based on the game mechanics that are present across those games.

By searching for a common thread among the games in all the designated collections of people who are getting together to play a game, it is able to give suggestions about what would give the group as a whole the most positive experience.

With more time it would have been our goal to add additional functionality whereas after playing a game, the application would give the user the ability to log their play to better learn about what types of mechanics and themes they enjoy. In this way, it would notice user tendencies regarding how often they play particular games, or other such data points such as how long they might typically go between plays of a certain game. So, the more a user would log plays within GameNight, the more the application would learn about their preferences. This would increase the accuracy of its suggestions for the future.

Above all, GameNight is a very specialized tool. Its purpose is simple and obvious, its interface is approachable, and it exists to solve exactly one problem. Its utility and appeal arise from its commitment to simplicity in function and form which support the specialized task it carries out.

At this initial stage, GameNight was not developed for Android devices. The reason for this was simple: both team members had iPhones, so the design and development work for an iPhone application was readily available. Since one of the main goals of the project was to use Senior Design as an opportunity to learn a new skill, this made it easier for us to work on as students without a lot of development experience. To develop something for Android, we would have needed access to Android phones. While this is possible for the future, it was not feasible

for the scope of the Senior Design project, so we focused our efforts completely on the iOS environment.

Methodology/Technical Approach

By requiring the application to have a simple user interface that made sense for all types of users, it helped us to build a practical application within a reasonable amount of time. This was also necessary as neither of us had much experience with software development. We decided to pick this project to push the boundaries of our technical skills, and still be able to deliver an application at the end of the spring semester.

In addition to having our weekly practicum on campus, we also had regularly scheduled meetings to collaborate, complete assigned tasks, and cast vision for the next segment of our work. Working in a flexible manner allowed us to quickly pivot if needed, or if a change was required due to a new development in our work or research.

User Profile – Application User

Project

GameNight Mobile Application

Potential Users

Person with a tabletop game collection

Person open to playing a tabletop game at a social gathering

Software, Interface, and Related Experience

GameNight users need no special experience other than understanding how to download and open a mobile application.

Experience with Similar Applications

There is currently no application similar to GameNight to the best of our knowledge. There are applications that allow a user to aggregate a collection, and applications that allow a user to connect with others over a common interest, but nothing designed for the express purpose of helping a user plan a game night with their friends and family.

Task Experience

When the user opens the application, the user can add games from the database to their individual collection. They can then link their profiles to those of their friends who have also designated games as being in their own collections. This designation can be for a variety of purposes: games a user owns, games a user is currently interested in playing, games a user is thinking of purchasing, etc. When it comes time to have a game night, the user will be prompted to select the profiles of their friends who will be attending. Then, they will enter the amount of time they would like to spend gaming. The application will then provide suggestions based on these criteria, taking into consideration the preferences and interests of all those in attendance,

and eliminating games that will not work due to time constraints or a number of players outside the range of what a game supports.

Frequency of Use

The application will be used whenever a user wants to plan a game night. This will vary from person to person and group to group.

Key Interface Design Requirements that the Profile Suggests

- Simple to use for anyone
- Aesthetically pleasing appearance

User Profile – Application Administrator

Project

GameNight Mobile Application

Potential Users

Administrator of the GameNight mobile application

Software, Interface, and Related Experience

GameNight administrators will need to be familiar with the Firebase database, the application code, and how to update both.

Experience with Similar Applications

There is currently no application similar to GameNight to the best of our knowledge. However, administrators will be familiar with other applications that use the Swift programming language and that use Firebase as a database and user authentication system.

Task Experience

Administrators will be responsible for the upkeep of game entries within the database, as well as the maintenance of user accounts within the application. Administrators can add games to the database so users will be able to find them and add them to their individual collections. Additionally, they can make corrections to game data and user profile data, such as if data was entered incorrectly or something fundamentally changed, like the title of a game. Administrators will also be responsible to update the code and keep up with the state of Swift and Firebase, making necessary changes to remain up to date.

Frequency of Use

The application will be administrated on a regular basis, as games are continually being released, and data is continually changing.

Key Interface Design Requirements that the Profile Suggests

- Familiarity with the Swift programming language

- Familiarity with the functionality of Google Firebase

- General knowledge of board game new releases/data changes

Use Case Diagram

Figure 2: Use Case Diagram shows the actions that both a user and an administrator of GameNight would perform. Please note: for the User, “Create Account” and “Log Game Played” represent future work; additionally, “Edit Game Preferences” and “Edit Game Mechanic Preferences” were eliminated as tasks, but remain as artifacts to detail our earlier process. The code logic determines preferences rather than a user needing to enter them.

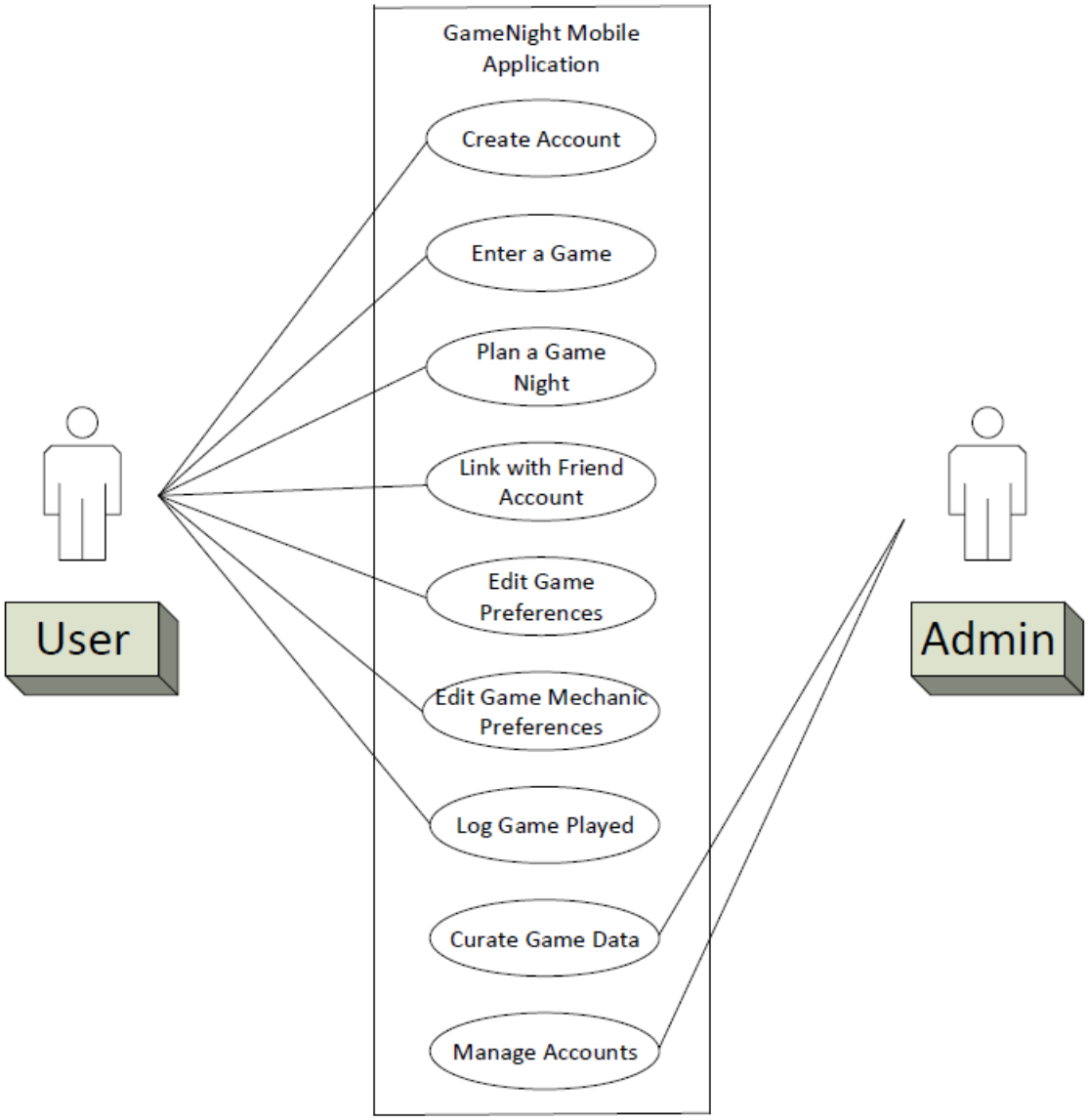

Figure 2: Use Case Diagram

Technical Architecture

Application

The application was written in Swift using Xcode. Swift is Apple's proprietary programming language for its operating system, and Xcode is Apple's Integrated Development Environment.

The main function of the application was to allow a user to select friends who will be attending a tabletop gaming event, designate the length of time for gameplay, and then receive recommendations about what games from the attendees' designated game collections would provide the most positive experience for everyone.

Users can designate games for their collection from the GameNight database. In the same way, users can also designate other users as friends. The application uses this information when making recommendations. It breaks games down to their mechanical level, searching for a mechanic that is common across the attendees. We defined game mechanics as "the component parts of a game that drive the action from the beginning to the end of the game." For example, Monopoly has five mechanics: roll dice and move your piece; set collection (same color properties); trading with other players; player elimination (risk of bankruptcy); and bidding via auction, if a player who lands on an available property chooses not to purchase it. By searching for a mechanical commonality among players it filters out all the noise, eliminating indecision and saving time, all the while providing a better experience for everyone involved.

Although it is technically a social application, there is not a lot of typical social functionality. This was by design: the application was designed to encourage users to put their phones away and spend time with their friends by interacting over a tabletop game. Thus, the app was designed with the intention that a user would not spend more than a few minutes with it in

order to get their recommendations. The developers were interested in finding a way to leverage technology in such a way that it would discourage the use of technology and create a space for people to spend intentional time together.

Database

The backend of GameNight was hosted on Google's Firebase Realtime Database. Firebase has many backend functions, but GameNight utilized it only as a realtime database. Firebase is a NoSQL database that lives in the cloud. It is storing all tabletop game data, user data, and image data. All of the game and user data is stored as JSON objects. The images are actual images that are saved in the cloud.

Utilizing Firebase allowed the developers to focus on learning Xcode and Swift, which form the frontend of our project. Without a lot of development experience, this allowed the creation of a functional application without a lot of complex backend code. Since the goal for this project was to take the opportunity to learn a new skill, this made it much easier to focus on learning Swift in much greater depth than would have otherwise been possible if splitting time with backend development. A focus on the user experience also provided a much better application within the allotted timeframe for Senior Design.

Network

GameNight observes the Firebase realtime database for changes in the data, such as when a user adds or deletes a game from their current collection. This happens as communication between the application and the cloud server, through observing/listening functions within the application code itself. Firebase is highly scaleable thanks to its home in the cloud. As such, it would still be able to support GameNight if it grew to a large scale. Because the data is stored as a JSON tree, no logic would have to change when scaling up.

Security

In a production environment, user authentication would be handled by Firebase on the backend. Additionally, no users have access to add games to the database. This was done to prevent any malicious or careless activity related to the user, game, and image information. Additionally, this supports the integrity of the data by allowing administrators to maintain full control over information entering the database.

Testing Plan and Report

Overview/Methodology

The plan included manual tests due to the relative simplicity of the application. The manual tests were as follows:

- Log in with a username and password
- Add a game to a user's collection
- Remove a game from a user's collection
- Enter attendees and time for a game night
- Link a friend's profile
- Unlink a friend's profile

Future manual tests will include:

- Register a new user account
- Log a game play

Scope of Testing

The application is iPhone only at this point in time, so iPhones of various versions were used in the testing, both simulated within the Xcode environment and on various physical devices.

Objectives

The objective of testing was to ensure that the functions of the application worked as expected. The tests were constructed to isolate each function so it could be viewed on its own, in series with the other functions. Each verified test could then be brought together to ensure the application worked from the time a user logged in through planning a game night.

Reporting

Test Name	Planned Result	Actual Result	Pass/Fail	Reported Bug
Logging in	**User connects to application**	**Login screen freezes while connecting**	**Fail**	**Entered email/password not in database**
Logging in	**User connects to application**	**Main application appears**	**Pass**	**None**
Add game to user collection	**Game appears in user collection**	**List of all games doubles in size**	**Fail**	**Code observing database appends all data to games array**
Add game to user collection	**Game appears in user collection**	**Screen does not return to games list**	**Fail**	**Incorrect code for push/pop view vs. present/dismiss view**
Add game to user collection	**Game appears in user collection**	**Game added to user collection**	**Pass**	**None**
Remove game from user collection	**Game is deleted from user collection**	**List of all games replicates multiple times**	**Fail**	**Code observing database appends all data to games array**
Remove game from user collection	**Game is deleted from user collection**	**Screen does not return to games list**	**Fail**	**Incorrect code for push/pop view vs. present/dismiss view**
Remove game from user Collection	**Game is deleted from user collection**	**Game is deleted from user collection**	**Pass**	**None**
Select guests and time	**User receives game recommendations**	**User receives expected recommendations**	**Pass**	**None**
Linking user profile	**Friend appears in user friend list**	**List of all users replicates multiple times**	**Fail**	**Code observing database appends all data to users array**

Linking user profile	**Friend appears in user friend list**	**User profile is linked with friend's profile**	**Pass**	**None**
Unlinking user profile	**Friend is removed from user friend list**	**User profile is unlinked from friend's profile**	**Pass**	**None**
User selects game entry information	**Game information appears**	**Information from a different game appears**	**Fail**	**UI Table Cells are not corresponding to the correct value**
User selects game entry information	**Game information appears**	**The correct game information appears**	**Pass**	**None**
User selects user entry information	**User information appears**	**Information from a different user appears**	**Fail**	**UI Table Cells are not corresponding to the correct value**
User selects user entry information	**User information appears**	**The correct user information appears**	**Pass**	**None**

Figure 3: Testing Report

What we learned during testing

Testing was instructive in learning how to debug code logic. Finding bugs was a process of simply using the application and discovering errors as time went on and the code became more and more fleshed out. It also became evident that the application was simple and intuitive to use, which was perfect for an application that was meant to encourage users to engage with it briefly and then set their phones aside. Due to the application's relative simplicity and the fact that it serves users by accomplishing one very specific purpose, bugs were easy to detect.

Testing also confirmed that we accomplished our goal of making something that was easy to use, for people of all comfort levels with technology. Through the process of testing new features, it became apparent that some features that were initially discussed did not actually fit with the refined vision of what the application should accomplish. For example, it quickly became clear that there was no need to include chat functionality. For one thing, many other applications are already in use that allow people to communicate through messaging. More

importantly, chat functionality would go against the core mission of GameNight: to encourage people to put their phones to the side and enjoy spending time together. In this way, testing and gaining experience with the application helped tremendously in narrowing the scope of GameNight and making it more effective at its stated purpose.

Budget

When this project began, one of the goals was to try to accomplish the entire project with resources that were already available, or free to acquire. Apple makes the Xcode IDE available free of charge, which made it free to begin learning Swift in the same environment that was used to do all of the software development work. Because the project was completed using only resources that were either free or already available, the original and final budget were the same. Figure 4: Project Budget shows the budget for the project.

Line Item	Cost for Project
2x Macbook Pro Laptops	$3,600.00
1x iPhone SE	$300.00
1x iPhone 6s	$150.00
Xcode IDE Developer Program	$99.00
Google Firebase Account	$2,000.00
Software development (250 hours) ($50/hr.)	$12,500.00
Testing (10 hours) ($50/hr.)	$500.00
Project administration/classwork (100 hours) ($50/hr.)	$5,000.00
Subtotal	$24,149.00
Group Total	$0.00

Figure 4: Project Budget

Since another main goal was to learn a new skill, it made sense to code a mobile application in Swift, since both team members had iPhones. Developing for Android would have required learning twice as much from scratch, which would not have been feasible for this project, since neither team member had experience creating a real application outside the context

of a classroom project. It helped that two different models of iPhone were available as well as many others that were able to be simulated, so the application could be visualized easily on two real devices that were distinct from one another, and on demand on virtually any iPhone potentially available to users capable of supporting the application.

Gantt Chart

Figure 5: Gantt Chart shows the planned work for GameNight.

ID	Task Mode	Task Name	Duration	Start	Finish
1		**1. GameNight Mobile Application P**	**166 days**	**Mon 8/27/18**	**Mon 4/15/19**
2		**2. Project Initiation**	**21 days**	**Mon 8/27/18**	**Mon 9/24/18**
3		2.1 Form Team	4 days	Mon 8/27/18	Thu 8/30/18
4		2.2 Draft Team Contract	16 days	Thu 8/30/18	Thu 9/20/18
5		2.3 Project Approval	1 day	Mon 9/17/18	Mon 9/17/18
6		2.4 Submit Team Contract Draft	1 day	Thu 9/20/18	Thu 9/20/18
7		2.5 Submit Team Contract Final	1 day	Mon 9/24/18	Mon 9/24/18
8		**3. Groundwork/Requirements**	**16 days**	**Mon 9/24/18**	**Mon 10/15/18**
9		3.1 IDE Research	1 day	Mon 9/24/18	Mon 9/24/18
10		3.2 Application Requirements	5 days	Tue 9/25/18	Mon 10/1/18
11		3.3 Coding Research	11 days	Mon 10/1/18	Mon 10/15/18
12		3.4 Image Coding Research	11 days	Mon 10/1/18	Mon 10/15/18
13		**4. User Interface Development**	**16 days**	**Mon 10/15/18**	**Mon 11/5/18**
14		4.1 Main Page Development	6 days	Mon 10/15/18	Mon 10/22/18
15		4.2 Board Game Page	4 days	Mon 10/22/18	Thu 10/25/18
16		4.3 Collection Input Page	3 days	Thu 10/25/18	Mon 10/29/18
17		4.4 User Profile Page	4 days	Mon 10/29/18	Thu 11/1/18
18		4.5 Friend Profile Page	3 days	Thu 11/1/18	Sun 11/4/18
19		4.6 Code Review	1 day	Mon 11/5/18	Mon 11/5/18
20		**5. Database Development**	**31 days**	**Mon 11/5/18**	**Sat 12/15/18**
21		5.1 Database Selection	11 days	Mon 11/5/18	Mon 11/19/18
22		5.2 Database Learning	11 days	Mon 11/19/18	Mon 12/3/18
23		5.3 Database Coding	11 days	Mon 12/3/18	Sat 12/15/18
24		5.4 Code Review	1 day	Sat 12/15/18	Sat 12/15/18
25		**6. Code Revisions**	**31 days**	**Mon 1/14/19**	**Mon 2/25/19**
26		6.1 Product Enhancements	6 days	Mon 1/14/19	Mon 1/21/19
27		6.2 Code Refinement	16 days	Mon 2/4/19	Mon 2/25/19
28		**7. Rollout**	**36 days**	**Mon 2/25/19**	**Mon 4/15/19**
29		7.1 Final Testing	6 days	Mon 2/25/19	Mon 3/4/19
30		7.2 Final Code Revisions	11 days	Mon 2/25/19	Mon 3/11/19
31		7.3 App Store Deployment Resea	6 days	Mon 3/18/19	Mon 3/25/19
32		7.4 IT Expo Preparation	11 days	Mon 3/25/19	Mon 4/8/19
33		7.5 Presentation/Demo	1 day	Mon 4/1/19	Mon 4/1/19
34		7.6 IT Expo	1 day	Tue 4/9/19	Tue 4/9/19
35		7.7 Final Paper Due	1 day	Mon 4/15/19	Mon 4/15/19

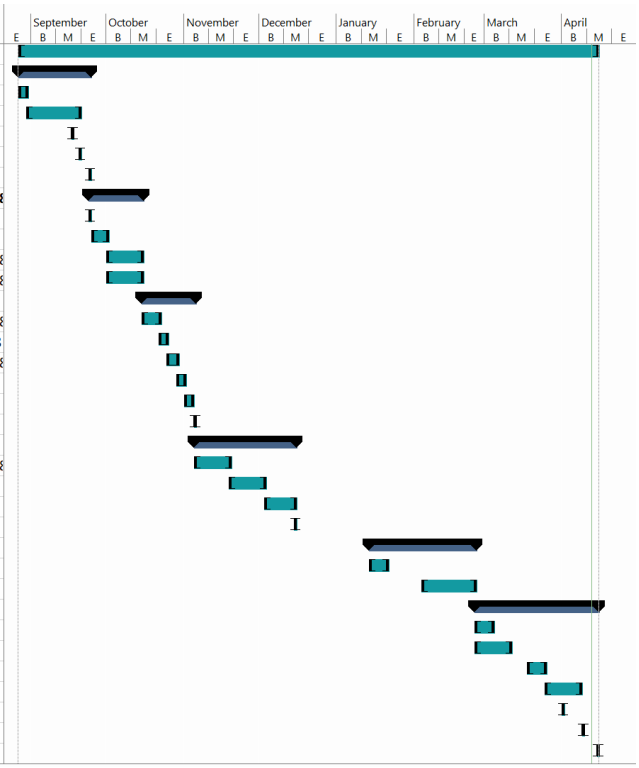

Figure 5: Gantt Chart

From the beginning of the project, the planned work had to change because the complexities of learning how to develop an application that used a database when coming from a non-development background were underestimated. Therefore, development work superseded other areas in order to deliver the project within the timeframe of the senior project. This meant that certain areas of development work greatly expanded, while others were minimized or

eliminated. Because the team members had never built a complex application, there was always the next item on the list that needed to be developed. That continued to be the case even after delivering the senior project.

What this meant in practice was that user testing at the end was mostly eliminated, and we conducted the testing ourselves. We felt comfortable with this, as the application was relatively simple. Additionally, choosing Firebase as a backend was a key decision. Because Firebase eliminated the need for a lot of backend code, this created an environment where the focus was on building the frontend of the project. In retrospect, the task would have certainly been overwhelming for two first-time developers had there been a need to develop the backend portion of the project from scratch as well as the frontend.

Problems Encountered and Analysis of Problems Solved

The challenges we faced when developing the project related to a lack of knowledge base of software development. Because neither team member focused on software development in school, the challenge was in refining the skills necessary to bring the project to fruition based on a limited amount of experience in programming. As the project progressed, encouraging progress was made and the team members' development skills improved. It was entirely possible that developers would have been more cognizant of challenges with more experience in programming, but the old adage applies: "you don't know what you don't know."

Problem: Getting data from Firebase to display in user interface

The main problem encountered at first while developing was getting data to come back from Firebase and display for a user to view it in the application itself. The data was coming back, because it could be printed and viewed in the console. However, it was not appearing in the user interface of the application itself. Other problems occurred after this one, because a

functional user interface made it easier to see the visual results of errors in the code written up to that point. Additionally, there was a lot of code that was being blocked by the fact that we needed to see our data.

As a method of solving this problem, a simultaneous attempt was undertaken by both team members as a means of exploring different ways to write the application code to communicate with Firebase. Both team members wrote separate versions of the application in different ways. However, because neither team member had much development experience, this was an issue that we were not able to solve on our own. Aaron Sapp, an iOS developer, was approached for help. He provided mentorship on how to get data from Firebase to display in the user interface.

Problem: List of Games or Users replicates when item is added/removed from collection

When modifying data within a user's game collection or friend list, the first problem encountered was that information would snowball and continually add itself again and again from the database into the list of all users, or the list of all games within the application itself. Several full sets of data would append themselves to the list within the application whenever data changed by way of an item being added or deleted to/from the user's personal collection. At first, this was very puzzling as it seemed arbitrary as to the number of times that a full set of data would be added to the list of all users, or the list of all games. Several full sets of data could be appended to the list after just one change.

Upon investigation, it became apparent that the issue arose with how the code was observing the database for changes. There was an array that was placed inside of the observing function for game data, and another inside of the observing function for user data. The purpose of these arrays was to load the game data and user data into the user interface, so a user could

search and select from it and add games or users to their collection by linking items to their profile. However, because the array was included within the code that was observing the database, it was just growing larger and larger with the same data each time there was a change observed. Figuring this out was the difficult part. Once the problem was identified, it was a simple matter of moving the array declaration elsewhere in the function outside of the observation, so every time the function was called the array would empty itself out and reconstruct itself instead of appending more and more data to itself.

Problem: View does not hide properly when item is added/removed from collection

This issue was also related to adding and removing items from the database, but this time it focused on a bug within the Swift code itself. The issue was that when a user selects to have a game added to their personal collection or another user linked to their profile, the view of the single game or user was not dismissing properly. This meant that there was no indication if the button had worked or not, although the data in Firebase was indeed being modified.

This turned out to be because there are multiple ways to send a view to the back or the front within Swift, and they must be paired together correctly in order to function properly. A developer can either push (show) and pop (hide) a view, or they can present (show) and dismiss (hide) a view. Once this was understood, the issue was able to be addressed and corrected.

Future Recommendations

Future Enhancements

In the future, there are several enhancements that could be made to the product. One thing that would be great to add is a set of utilities to help players with common issues surrounding games that aren't necessarily a part of the games themselves.

For example, it would be relevant to the application to include a “random first player” feature. Many games give rules for determining a first player, but oftentimes these rules are very lazy or extremely unsatisfying. One such common rule is that the oldest (or youngest) player at the table gets to go first. This is not only a lazy rule, it is often irresponsible since in many games turn order can have a very large impact on the strategy a player needs to employ in order to do well. By including a random first player utility, this could be used for every game, giving each player an equal chance of taking the first turn.

Another example would be a dice rolling utility. Players could choose a die with however many sides they need, as well as how many dice of that type they need to roll. Common types of dice are four-, six-, eight-, ten-, twelve- and twenty-sided. This would prevent some situations in which players either do not have the dice they need on hand, or would need to rummage around in other games in order to scavenge the dice they need.

Another future enhancement would be a form where users could submit games that they would like to be added to the database. Since users don’t have access to modify the database for a variety of reasons, this would be an effective way to help keep the game data up to date. Because there are an overwhelming number of games being released, it would be easy for some games to be overlooked by the administrators. In addition, the application is only useful to a user if it contains the games they would like to include in their collection.

One suggestion we received that was not considered over the course of the project was the functionality to limit games by a recommended age. In certain contexts, this would be very important, such as when adults and children are playing a game together. While not a perfect indicator for what children can handle, recommended ages provide a guideline for the

complexity level of a game and a child's ability to not only understand it, but to have a good time while playing.

Project Poster

Figure 6: Project Poster shows the poster for GameNight, which was used at the IT Expo.

Figure 6: Project Poster

Conclusion

Lessons Learned

Fall semester, most of our learning focused around the Swift programming language, the Xcode Integrated Development Environment (IDE), and how to connect a database to a user-facing application. A big takeaway has been that we shouldn't be intimidated by programming, as long as we have a willingness to learn and an enthusiasm about the subject matter.

We developed technical skill as programmers as we have worked on the GameNight mobile application. As spring semester unfolded, this skill advanced as the application came together. Countless hours were spent in making the application as good as it could be. We enhanced existing functionality, added more functionality, and improved the look and feel of the application. The goal of the project was to leverage the Senior Design class to gain skill in a new area before graduation. This goal was accomplished.

References

Griepp, Milton. 2016. “Hobby Games Market Nearly $1.2 Billion.” *ICv2* (Madison, WI), July 31, 2016. https://icv2.com/articles/news/view/35150/hobby-games-market-nearly-1-2-billion

Griepp, Milton. 2018. “Hobby Games Top $1.5 Billion.” *ICv2* (Madison, WI), July 30, 2018. https://icv2.com/articles/news/view/41016/hobby-games-top-1-5-billion