# Bowl Picks

Zackery Griesinger

UMKC CS 449

Table of Contents

[Vision Statement 3](#__RefHeading___Toc492807790)

[Requirements 4](#__RefHeading___Toc492807791)

[Categories of Actors 4](#__RefHeading___Toc492807792)

[Actor-Goal List 4](#__RefHeading___Toc492807793)

[User Stories 5](#__RefHeading___Toc492807794)

[Product Backlog 6](#__RefHeading___Toc492807795)

[Sprint #1 7](#__RefHeading___Toc492807796)

[Review 7](#__RefHeading___Toc492807797)

[Retrospective 7](#__RefHeading___Toc492807798)

[Design 9](#__RefHeading___Toc492807799)

[Coding Standards 10](#__RefHeading___Toc492807800)

# Vision Statement

**For any Android users who have an interest in sports, especially fantasy sports who want to compete against their friends in an assortment a NCAA bowl pick competition. Unlike other fantasy sports apps that require you to learn every player and draft a well thought out team our product allows for simpler competition in that users only have to pick the bowl game winners.**

# Requirements

## Categories of Actors

**Players** – Adults with sufficient knowledge to navigate the app and sufficient knowledge of NCAA football to be able to pick winners of NCAA bowl games in a specific league.

**Commissioners –** Players that lead a group of players in a league. They manage the settings available to a league and facilitate the league play

**Administrator –** Not necessarily associated with a league but they can edit the database to set the game winners and point spreads accordingly.

## Actor-Goal List

|  |  |
| --- | --- |
| **Actor** | **Goal** |
| Player | Pick NCAA bowl game winners |
|  | Browse who else is in their league |
|  | See the picks of other people in their league |
|  | See live updates of winners and current leaderboard |
|  | Run simulated game winners to see resulting leaderboard |
| Commissioner | Create a league – with a name and logo |
|  | Set the league settings, like what bowl games count, how much point spreads are worth and report winnings. |
|  | Add people to managed league |
|  | View all picks and leaderboard |
| Administrator | Set up all available games and available settings for leagues |
|  | Modify errors in user and team database |

## User Stories

**As a player I would like to create an account and be able to sign into my account and see my picks and have them be saved for when I come back.**

**Story Points: 8**

**As a player in a league I would like to be able to make picks on the upcoming NCAA football bowl games.**

**Story Points: 5**

**As a player in a league I would like to be able to see the other picks of the players in my league once I have locked in my picks.**

**Story Points: 3**

**As a commissioner I would like my account to be able to handle all of the management aspects of a commissioner but also be able to act as a player in a league.**

**Story Points: 10**

**As a commissioner of a league I want to be able to see and manage the picks of the players in my league.**

**Story Points: 3**

**As an administrator of the app I want to be able to see all the commissioners of all the leagues and manage the logistics of the player and commissioner user settings.**

**Story Points: 8**

### Product Backlog

The collection of stories makes up your project’s product backlog:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Story ID** | **Story** | **Story Points** | **Priority** | **Status** |
| S1 | Allow a user to create an account that can save their preferences/picks (Firebase DB) | 8 | 1 | COMPLETED 9/24/17 |
| S2 | Set Up Database to Record all users picks and correct picks | 5 | 2 | COMPLETED  10/1/17 |
| S3 | Allow User to pick and then display all their picks | 5 | 3 | COMPLETED  10/7/17 |
| S4 | Allow players to see the other picks of other players once they have locked in their picks | 3 | 7 | COMPLETED  10/8/17 |
| S5 | Commissioner feature, allowing management and creation of leagues | 10 | 4 | COMPLETED 10/15/17 |
| S6 | Commissioner can add players to league and see their picks. | 3 | 6 | COMPLETED  10/22/17 |
| S7 | Global app administrator feature that allows someone with access to manipulate the global games/winners that all leagues will use. | 8 | 5 | COMPLETED  11/05/17 |
| S8 | UI overhaul, Remembering login states and misc. tiny additions | 6 | 8 |  |

# Sprint #1

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S1 | Set up Firebase DB for authentication | 2 | 3 |
|  | UI for signing in | 3 | 1 |
|  | Create DB table to hold other user info | 1 | 1 |
|  | Link Firebase to Android Studio | 1 | .5 |
|  | [STRETCH] OAuth providers for Facebook/Google Login | 3 | n/a |
|  | Test all workflows with valid/invalid login/signup | 2 | 1 |
| S2 | Setup UI for picking games | 2 | 2 |
|  | Add DB procedure to store picks in DB | 1 |  |
|  | App UI for viewing picks easily | 2 |  |
|  | Test and validate proper picks and data | 4 |  |
| S3 | [STRETCH] Begin setting up logic for Leagues then for Commissioners | 4 |  |

## Review

[Minutes/notes from feature demo (product review).]

## Retrospective

What went well? What could have gone better? What lessons did you learn? What do you plan to do differently on the next iteration?

|  |
| --- |
| This iteration I was able to complete the first story and begin the setup for story 2. Overall my estimated hours were fairly close I just found that creating a basic UI was easier than normal but most of the coding required to get the activities to do what I intended was quite difficult sometimes. I great underestimated the work required for story 2. It is the core functionality of the app and should have been broken down into smaller sub stories. This is because while I set up the initial database and the UI for the bowl picks I didn’t fully finish the story so much velocity is quite low. I am going to break down story 2 and focus on it for the next iteration. Setting up the user authentication was difficult in some ways and more complex in others. I should have spent more time scouring the documentation rather than coding up solutions and debugging them, which took quite a lot of time to get the workflow moving between four activities. However, once I had got the whole solution working testing the features was a breeze due to the robustness of the integration with Android Studio. |

**Project velocity: 8**

# Sprint #2

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S2 | Write procedures/input information into the DB to hold wins | 2 | 1 |
|  | When a user logs in if they have picks display them and if not display let them pick | 3 | 4 |
| S3 | Display Radios from a list of games and teams | 1 | 2 |
|  | After a user picks the games submit them to DB | 2 | 2 |
|  | Create user interface to show the user their old picks for each game | 3 | 2 |
| S4 [STRETCH] | Explore options to see other picks of other users (Anonymously vs UserID?) | 3 | 3 |

## Review

This sprint was very successul, however I faced significant unforseen challenges in some aspects. Creating the list of games to pick was quite challenging because the list had to be created dynamically from whatever the database returns. The struggle was that when you create a form in Android programmatically it can be difficult to generate and then reference the id’s to get the state of the picks. I was able to use a formula in a “for” loop to reference the ID’s dynamically. The biggest thing that I need to refine next sprint is the UI of the app. It is very barebones and simplistic in its current form.

## Retrospective

What went well? What could have gone better? What lessons did you learn? What do you plan to do differently on the next iteration?

|  |
| --- |
| This sprint I was able to accomplish everything in my backlog, even the stretch goal. This was because however, I didn’t estimate some of the stories properly. Some things that I thought was going to take quite some time were quite easy and vice versa. Next iteration hopefully I manage to correctly guess and maintain the amount of time each process will take. Note that the screenshot of the new bowl pick UI is different, it is dynamically created instead of statically defined in the layout file. |

**Project velocity: 13**

# Sprint #3

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S5 | Give every player in the DB a status: “Admin” “Commissioner” “Player” | 1 | .5 |
|  | After a user picks, let them “Create a League” | 2 | 2 |
|  | Allow Commissioner’s to add players to the league by playername (or email?) | 3 | 4 |
| S6 | Commissioner can see everyone in his league’s picks | 1 | 1 |
| S8  [STRETCH] | Overhaul the UI to make it look better | 3 | - |

## Review

This sprint went fairly well. I set out pretty much everything I set out to do and I added a few things here and there. A couple things were unforeseen and they haven’t been implemented quite the way I would like (The UI for the picks) and one of my tasks S6 ended up being irrelevant as everyone should be able to see people in their league. I also added some logic checking on the entry into the app and removed the useless header bar on the front landing activity.

## Retrospective

What went well? What could have gone better? What lessons did you learn? What do you plan to do differently on the next iteration?

|  |
| --- |
| Overall I would say this sprint went well, I’m coming upon having all of the necessary features finished and then I will be able to refine the error checking and rectifying the god-awful UI that my app currently has. I did learn that the project will have to be flexible and sometimes the features that you think are worth while and good are actually pretty useless. Despite the fact that I had this app all planned out much more thoroughly than most other projects I’ve done there were still many unforeseen challenges and things that I hadn’t thought about. I’m not very familiar with Java as a language so many times I was hung up on syntactical discrepancies from what I am used to (looking at you String.equals instead of ==). The sprints are getting more and more effect and manageable as the project goes on. Hopefully I can continue to improve my process. I’ve come to learn that concise smaller stories, even if they take awhile are much more manageable to complete |

**Project velocity: 13**

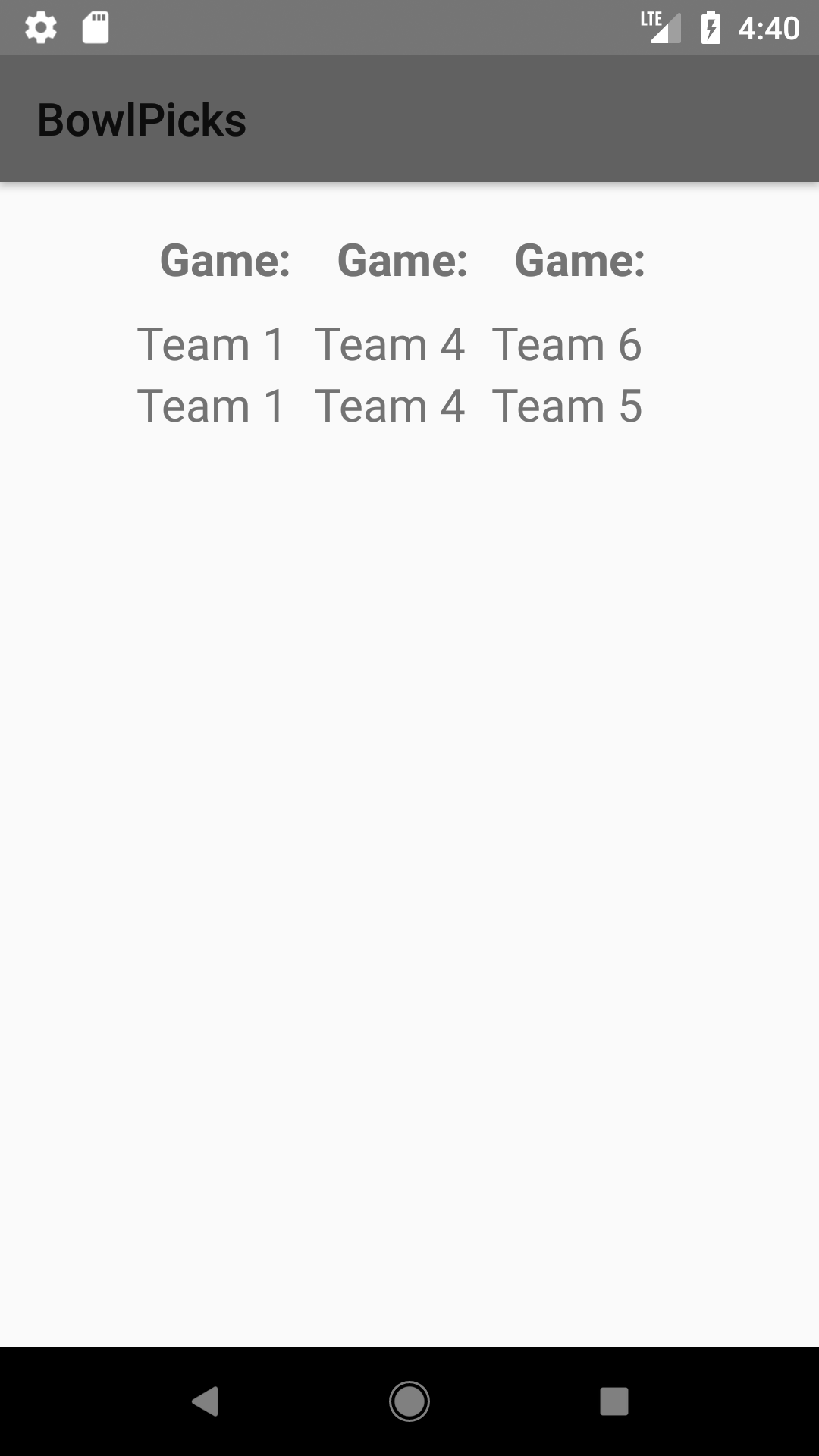
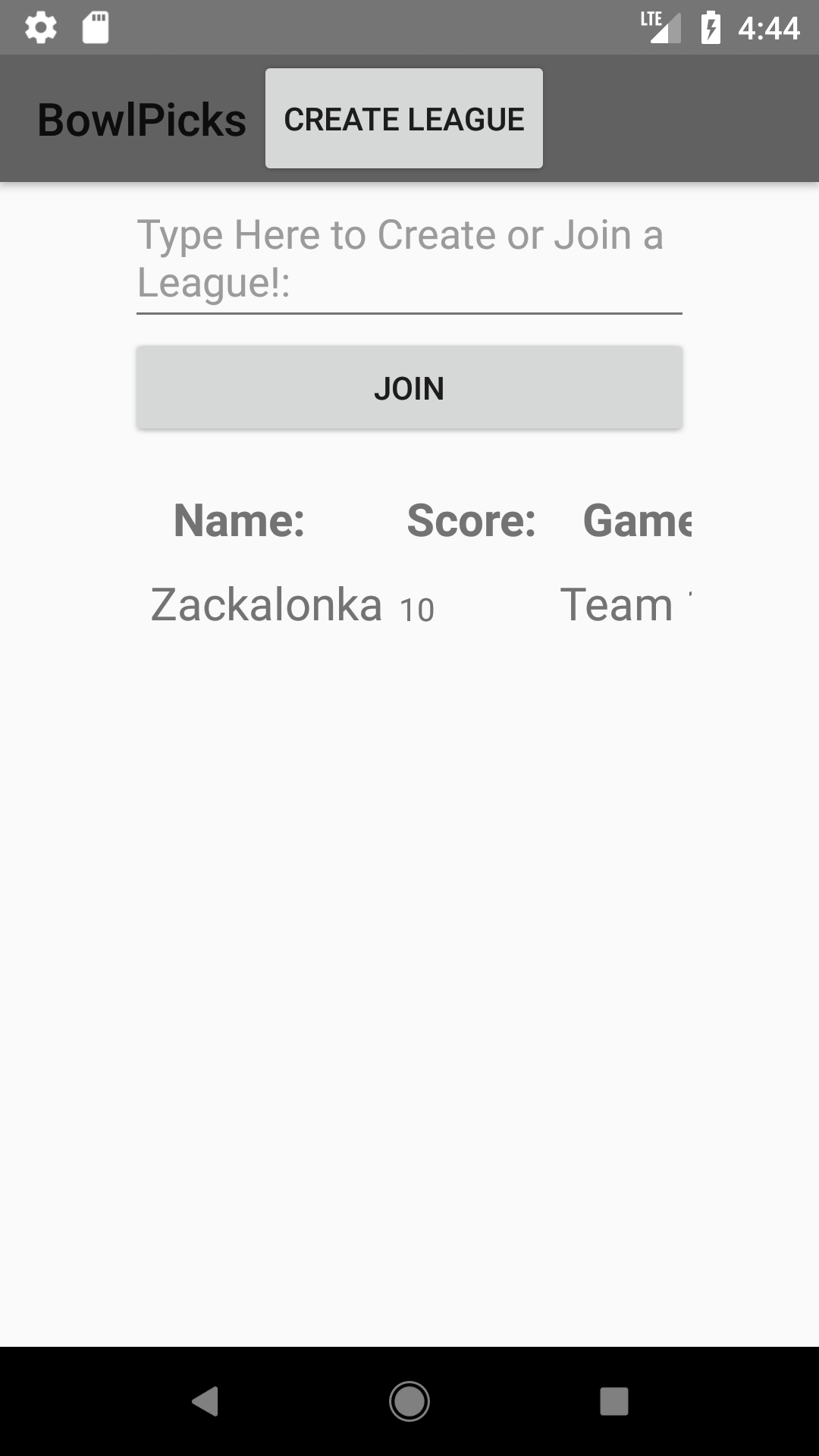
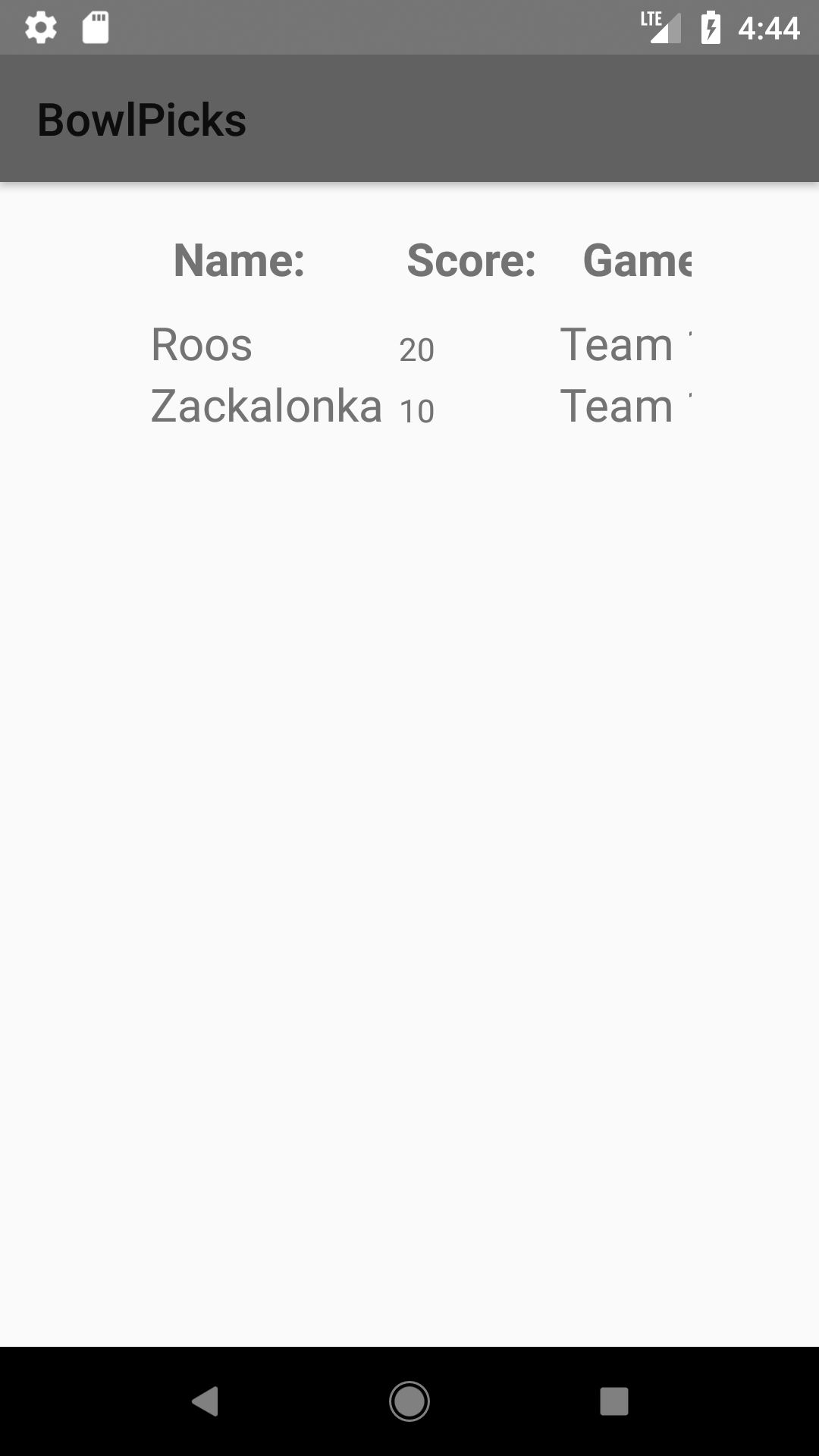
# Sprint #4

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S7 | Admin status for special users | 2 | 3 |
|  | Set Winners | 3 | 2 |
|  | Calculate points from odds (Realtime) | 3 | 4 |
| S8 | UI OVERHAUL | 4 |  |

## Review:

This sprint I added the admin status that can be controlled through the Database. It allows for someone to in real time update the scores of users. As they update the winners string once new users log in they will have the live scores. I also started the UI overhaul, however I’m not completely happy for the way it turned out so I will wait and count this on the next sprint. I also implemented a scrollable view for the picks and scores, this was a much needed update and part of the UI overhaul, however it took much longer than I anticipated and because I didn’t have a story for it specifically it doesn’t contribute to my project velocity. I also removed the join league button because it was serving the same function as create so I incorporated them into one.



## Retrospective

What went well? What could have gone better? What lessons did you learn? What do you plan to do differently on the next iteration?

|  |
| --- |
| I got a lot done for the main features this week and everything is in working condition. The app still needs a lot of work in just little things and error checking. Next sprint I will focus on handling some of the edge cases and exceptions (ie: if two users try to have the same display name) and as I do this I will need to do even more updates to the UI. Android is notably more difficult to design good UI as the web development I’m used to. Especially sense most of my data is dynamic and I can’t exactly plan it out in Android Studio. I attached the screen shot of the scrollable view as well to try to best represent it. I also am learning more and more Java, it only took me 15 minutes to remember its String.equals() instead of String == String instead of 30. Next iteration I hope to have a lot of things ironed out as displayed by the added stories for iteration 5. |

**Project Velocity: 8**

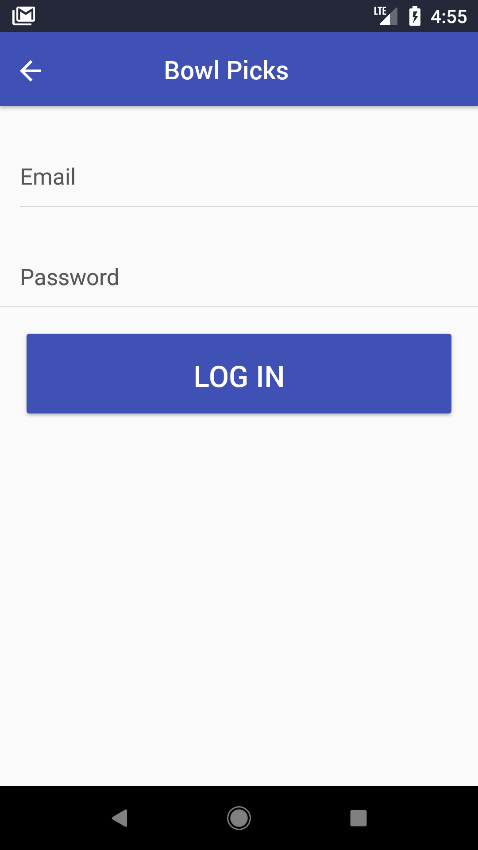
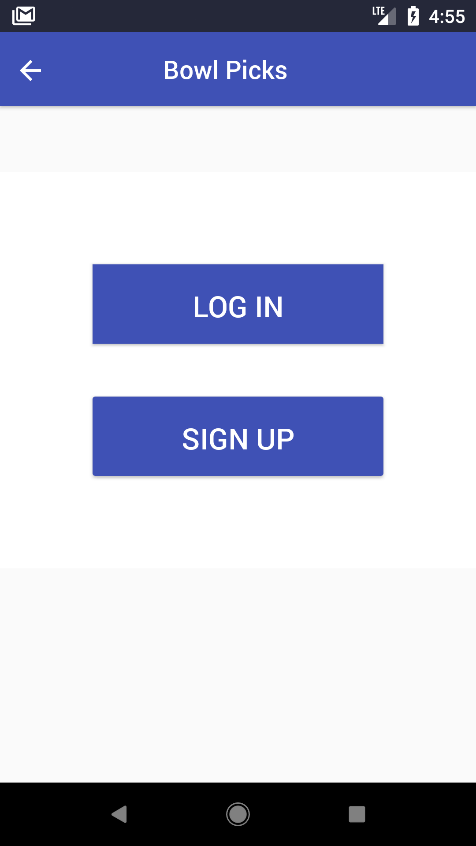
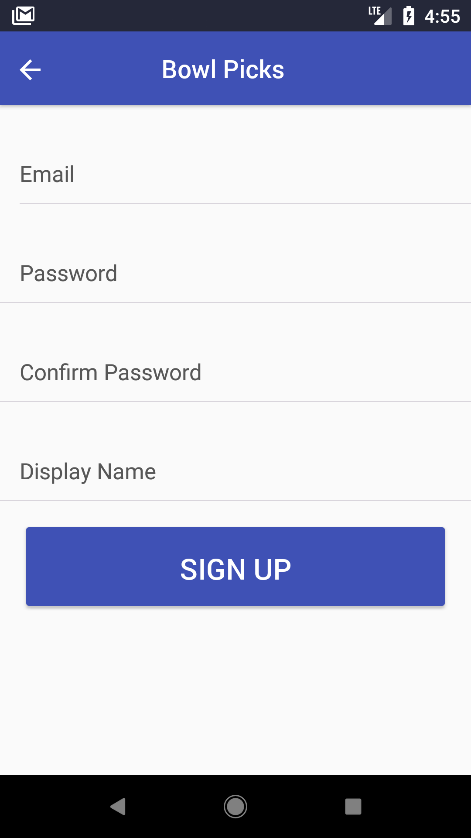
Sprint #5

Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| S8 | Login/Signup page with logo | 1 | 4 |
|  | Better padding on dashboard | 2 | 3 |
|  | Header displaying league name on dashboard | 1 | - |
|  | Error display and checking for duplicate names and such | 4 | - |

## Review:

I wasn’t able to complete my story this week, this was due to a change in my project structure and underlying framework. I decided to move my project to a framework called React Native. It provided many challenges. But I was able to replicate the sign in process much easier with toasts that notify the user of their login errors. It also looks much nicer and provides a more responsive user experience. Next sprint I will focus on finishing up the UI overhaul providing the same functionality as the old app. To look at the source code for the app now, it is mainly in the App.js file and the compiled/packaged down android files are in the android directory.



## Retrospective

What went well? What could have gone better? What lessons did you learn? What do you plan to do differently on the next iteration?

|  |
| --- |
| React Native is a JavaScript framework that allows for native JS code to be ran cross platform on Web, iOS, and Android. I decided to switch to this framework because I have done a lot of web development using it and I am very familiar with the component life cycle and it is a lot easier to create a responsive UI. I began the process at the beginning of the spring and didn’t finish the initial migration and setup until the end of the sprint. The very initial setup was easy, I used a package called create react native app to streamline the creation of the package. However I had to “eject” from this package pretty early on to integrate the app with my Firebase database. After I ejected I had a lot of trouble getting the app to build as the gradle system wasn’t working very coherently with the JavaScript packager. I used an integrated Firebase build process to eventually get the app working and I began styling the app. Once I integrate over t |

Project Velocity : 0

# Sprint #6

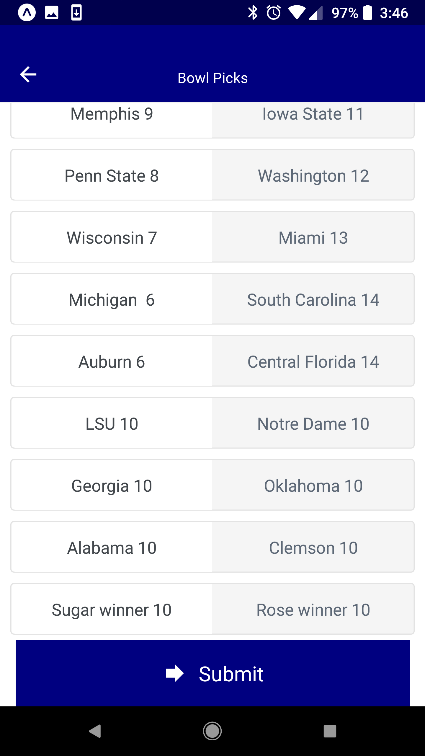
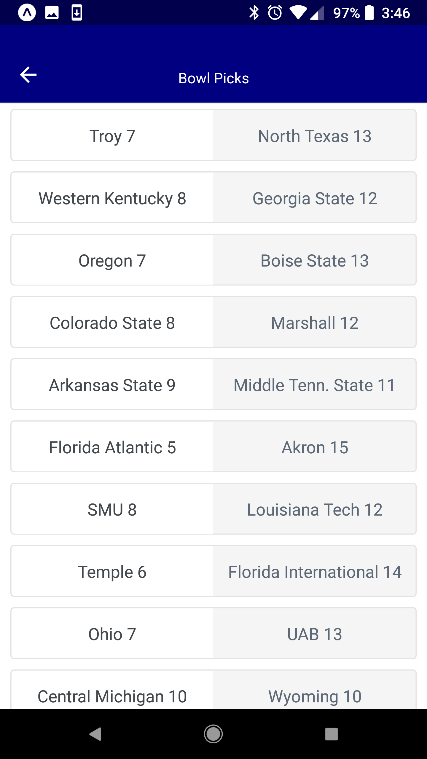
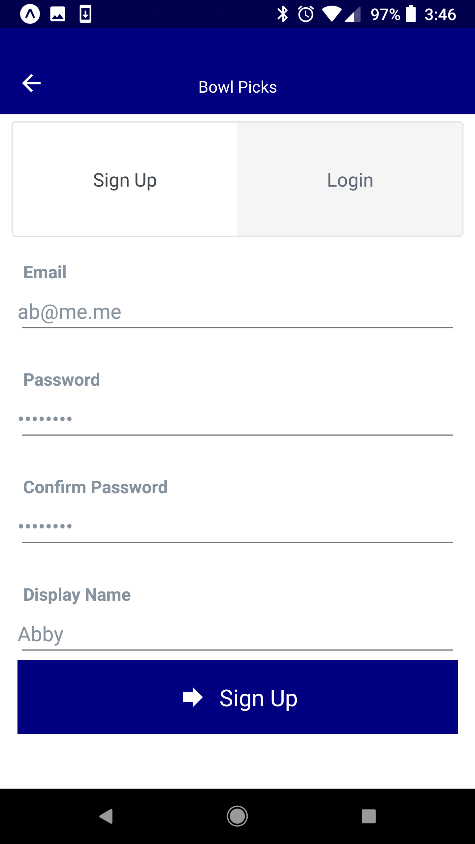
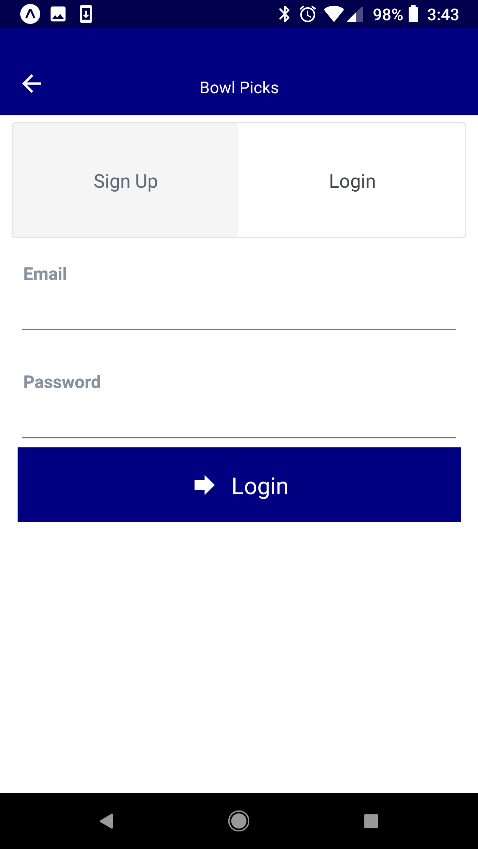
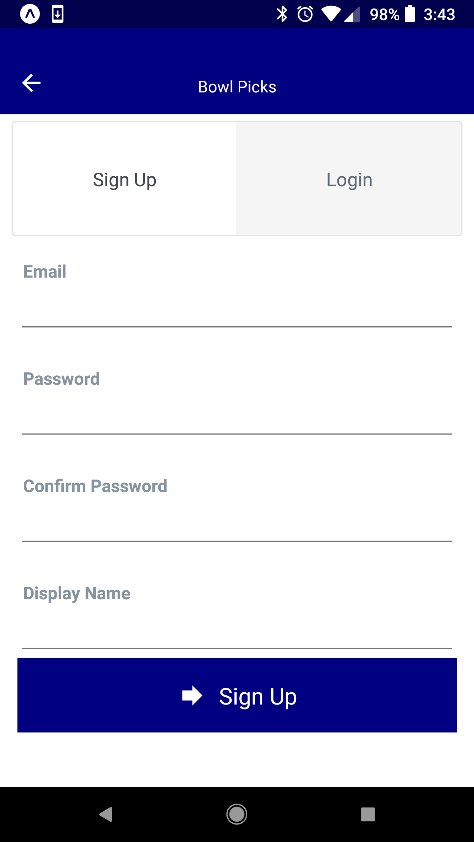
Sprint Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| **Story ID** | **Story / Task** | **Estimated**  **Hours** | **Actual**  **Hours** |
| **S8** | Better picklist UI | **2** | **4** |
|  | Better Leaderboard UI | **2** | **4** |
|  | Some kind of logo and color theme branding | **3** | **1** |

## Review

This iteration I finalized the UI of the app, I also was able to get the new React Native build process up to par with the feature set I had originally planned for the app. The picklist UI is much better now and is more in tune with the UX expected from mobile apps. Radio buttons are not normally associated with mobile app forms and generating the pick list using small buttons the user must click is bad UI/UX. I opted for button groupings that force a pick. If a user just clicks submit after they login they will automatically pick all favorites (Which are picks such that the user probably won’t miss every game, but the user is not going to win). The new UI also makes more sense on initial load. When the user fires up the app for the first time they are prompted no longer with two static buttons to login and sing up, but are now prompted with another group button that seamlessly switches between the two views. Once the user signs up or logs in it takes them to the picks page if they have yet to pick their games. On that page it prompts them with the list of games and picks (now fully updated with the 2017-2018 NCAA bowl games) Once they pick their games it forwards them to the page where they can see the leaderboard and the points that each player has. I did make a conscious decision to keep the league system out of the app. It was a feature that made the app too complicated and was not going to be useful for the people that were going to be using the app this year (as the app is 100% topical, as in it will be useless after the first of the year so the main features took priority on the UI rework). I also prevented the users from seeing the picks of the people they are playing against. This was also by design because people could calculate different point differentials based off who was picked. For example: Let’s say Team 1 was worth 5 points and Team 2 was worth 15 points. If 90% of users picked Team 1 to win then the differential on Team 2 becomes greater. If users can see other’s picks it makes it less fair for people who pick first. There are still a lot of things I can improve in the app and it is by no means done, but at this point it is shippable.

Screenshots:

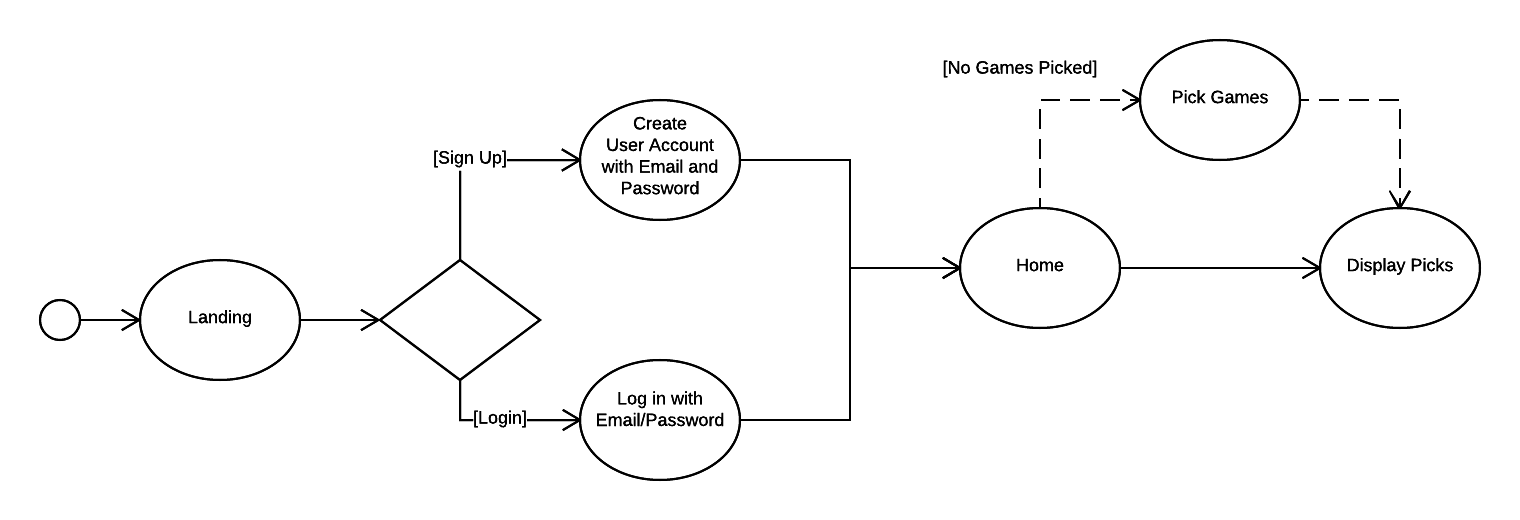


## Retrospective

|  |
| --- |
| I had a lot of issues this iteration, it was mostly due to the switch to React Native. It led to a lot of refactoring and changing old code that worked because of bugs that popped up. I think this was due to the nature of React Native. The framework is still very much in development and despite the many testimonies that I had heard from coworkers that it was perfectly stable for developing apps, there were a lot of random bugs and undescriptive error messages that made developing a huge headache.  The first of these was the way styling is done in React Native. I would say that I am sufficient in basic CSS however all the styling in React Native is done with CSS Flexbox. This is a new tag and something that I am not used to working with. There were a lot of times when things didn’t show up because I had not flexed the view properly. I believe that CSS Flexbox was probably the way to go but it takes a significant ramp-up time to understand how it works properly, and even after this project I would say that I don’t understand it as well as I should.  Second was the inconsistency of component libraries. The React Native default component library is very useful, however much of the contents of the components weren’t quite was I was looking for in terms of UI. At first, I thought that to be no problem as I knew from my experience with React.js that there were a ton of libraries out there that could provide me with pre-styled components. I found two of the biggest ones for React Native early on: “react-native-elements” and “native-base”. Both component libraries were decent however I needed to use their powers combined to get everything I needed in the app. This led to a lot of debugging that was simply resolved by namespace reduction and editing the styles of the components themselves.  Last were the erroneous error messages that React Native kept throwing that were undescriptive. In class we talked a lot about how descriptive error messages are necessary for good projects. In my project I have toasts that pop up if the database has an error and handle all user input errors gracefully by throwing toasts at the user and highlighting incorrect input boxes. However, the error messages thrown by some of the component libraries and the framework itself were not descriptive at all. I dealt with an error involving something called “YogaNodes” and their children. I found on Stack Overflow that this is resolved by switching the package you are using to a different one. Something that has absolutely nothing to do with the code of my project but with an external library. React Native also threw many warnings when it interacted with the Firebase database. I chose to use Firebase because I knew it integrated seamlessly with Android, and Android Studio, and this is true. Firebase integrates even more seamlessly with websites and JavaScript (which is what React Native is typed in). I was expecting the normal workflow and component lifecycle to work fine with React Native however, an inconsistency in the way React Native compiles to Android caused a performance degradation based on a set timeout that resided in a package.  I got these fixed to make the app work properly. In hindsight I probably should not have set out on a new framework so late into the lifecycle of the semester. However, I do believe that if this was an actual project the switch to React Native has been largely beneficial. It is something that lines up with the sunk costs fallacy we talked about in class. If I was going to maintain this codebase going forward the JavaScript-typed UI makes it so much easier, so despite the fact that I had to restart from square one, it was a long-term change that was for the better. Last iteration I ejected from the bootstrapping framework “create-react-native-app” and that turned out to be a mistake, once I refactored everything and rewrote all the functionality in a non-ejected project it was much easier to develop going onward. Over the course of the project I learned a lot about the development process, as well as developing on mobile platforms in general. While I may not have achieved all the features I set out to implement I do think the app fulfills the vision statement at the beginning of this document.  **Project Velocity: 8** |

# Design

[What is the overall structure of the solution? What are the major modules of code? What are the dynamics of communication between these modules? The most common way of depicting this information is with static and dynamic models augmented with short narrative descriptions of design.]



# Coding Standards

* Camel case all function calls
* Lower case with underscores for all constants and variable names
* All strings defined in strings.xml
* Develop branch gets moved into master at the end of the sprint/iteration