UNIVERSITY OF LONDON

CM3050 Mobile Development
Lecturer: Joe McAlister
Student Number: 190126209

Final Project Report: Recaller App

1. Concept Development

When we were asked to make a project proposal I thought about creating an application that I can use everyday because it is entertaining and above all, didactic. I know of several apps created particularly with this purpose ("Brilliant" being one I like very much). The problem is that most of these apps are paid (if you want a good experience that is), so I decided to create an app for something I truly like. I thought about games that test the speed of your math mental calculations (e.g. sums, subtractions, multiplication, division, percentages), or games that test your memory (remembering names, order of things, recall of adjectives, nouns, places, etc). Since I prefer the games that test your memory, I decided upon the latter.

My proposition was the following: "The application will be a game to exercise your memory, diction and agility. Basically, the game would present you with the definition of a word and a few rows of letters so that the user can choose the letters that spell the word that belongs to that definition. The game will present the user with a defined amount of definitions (6-7) and the game is timed, if the user can not provide the specific amount of words in the allotted time then he/she loses the game. But the user is also able to skip a definition if they don't really know the word for it, but the game will reduce time from the timer every time the user skips a definition.

At the end of the game the user will see a page where it shows the list of words that were practiced in that session and if they correctly gave the word for it, a percentage of correct responses, buttons to close the game or keep playing, and a score. Maybe even a graph to show how they have improved over time and a congratulation alert if they achieved the best score until that point. If time permits, I would also implement some nice animations for the game".

When I talked to some friends about this app they made the suggestion that the game should not be about winning or losing, just about learning. That is why I decided that, even

though the game will be timed, the user will not lose points for skipping words and also there would not be a defined amount of words to be played per game. The user will be able to guess an indeterminate amount of words, as long as it is done in the allotted time.

One of the questions asked during the proposal was “What problem does your application solve? For this I responded: “It solves the problem of people like me that want to take a break and practice these kinds of quick games that help them improve some aspect of their life but that don’t take too much of their time during the day. In my case I like learning new words so that is why I chose this kind of game… This could also help someone trying to learn words for a new language (like Memrise or Duolingo)”.

There are many apps like the one I propose, for example Elevate offers lots of games to improve different aspects of knowledge. There is also PowerVocab, 7 Little Words and many others. Why is my idea better than those already in the market? When I answered this question in the proposal I stated that “... since it is a basic idea, I do not think there is much to improve for these games besides the animations that you use, how you engage the user to really enjoy the app so that they want to use it even multiple times a day. The game could even give you reminders to practice at a particular time of day, show you how you have improved over time so that you can see if you are making any progress and give you “praise” when you do a great job. All of these games pretty much do the same but in a different way”.

Even after implementing my app, I still think the idea is simple. Yet, what makes it enjoyable is the interface that you implement. It could be quite simple but not engaging, or have many interactions and animations and be cumbersome or even annoying to use. I tried to reach that sweet middle point between those two extremes and I think I succeeded.

My main concerns about creating this app were:

- The animations that are needed to make the game engaging.
- How to calculate a score for the game: should difficult words be worth more points? How do I score words according to difficulty?
- How not to ask the same definitions over and over? Just having a really long list and expect that randomly choosing them will do a good job?
- Should I use an API to get the definitions or have my own “dictionary”?

2. Wireframing

When I was explaining my proposal to my friends I created some low fidelity wireframes for the app, which I will show in this section. I discussed my idea with a friend who is a teacher and loves these kinds of games too, so I thought her input would be quite valuable.

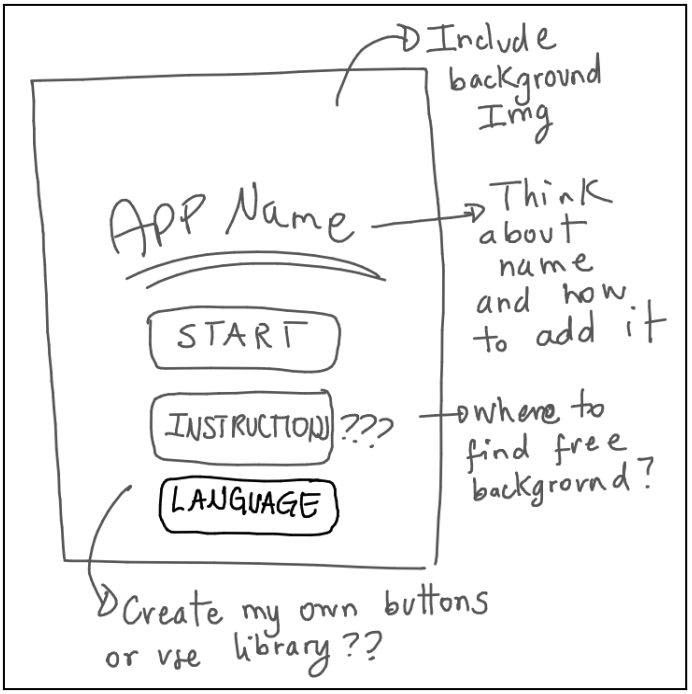

Figure 1. Home screen of the app

As you can see in Figure 1, at the beginning I thought about including words for several languages in this game. I only implemented the game in English though. I named the game “Recaller” because it is a memory game to “recall” words.

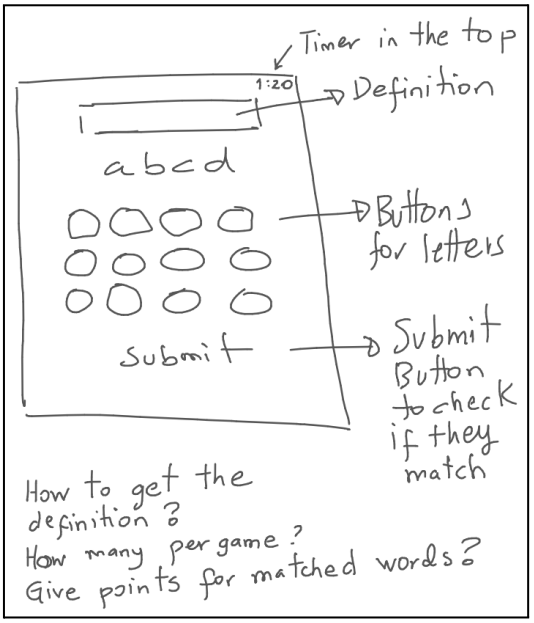

Figure 2. Main Screen of the app.

At the beginning I did not know if it was preferable to use the keyboard provided by the phone or tablet or to create one myself. I decided on the latter because I thought that since it was a game about spelling words, it would be simpler if the user has a more limited number of options to choose from (instead of all the 26 letters). Also at this point I decided that the words should not be excessively long or too short, and chose to use words that were 3-13 characters in length.

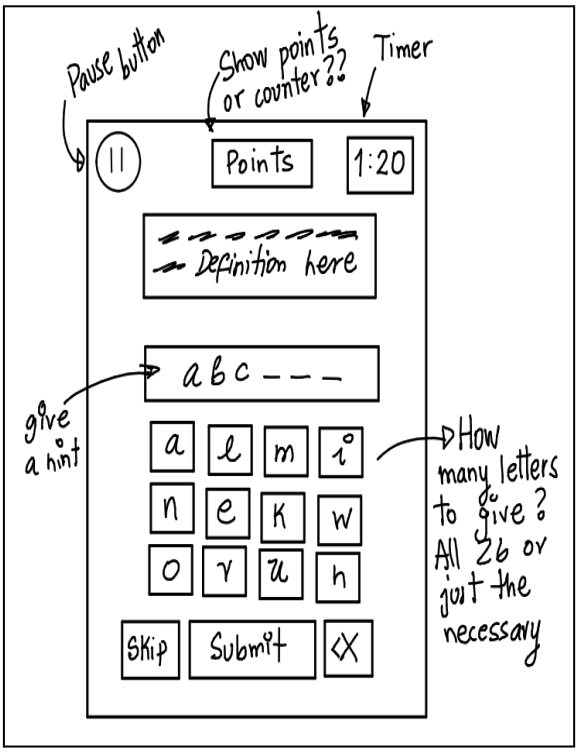

Figure 3. Main Screen of the app (higher fidelity)

There are several things to mention in the Main Screen. I decided to include a skip button because sometimes it happens that the user does not really know the answer, and still this should not be considered wrong. A backspace button is needed because the user can make a mistake while spelling the word and should be able to go back to a previous state.

Since many words can match a given definition, the game should provide a few letters as a hint so that the user can choose between those different options that might exist. A pause button was provided so that the user could pause the game, but also, to be able to restart the game from within the Game Screen. (See Figure 4)

Even though the user has access to the instructions of the game when the app starts (see Figure 1), it should be able to access them from any screen, if for some reason they forget how to play it. The idea is to be able to offer help wherever is needed.

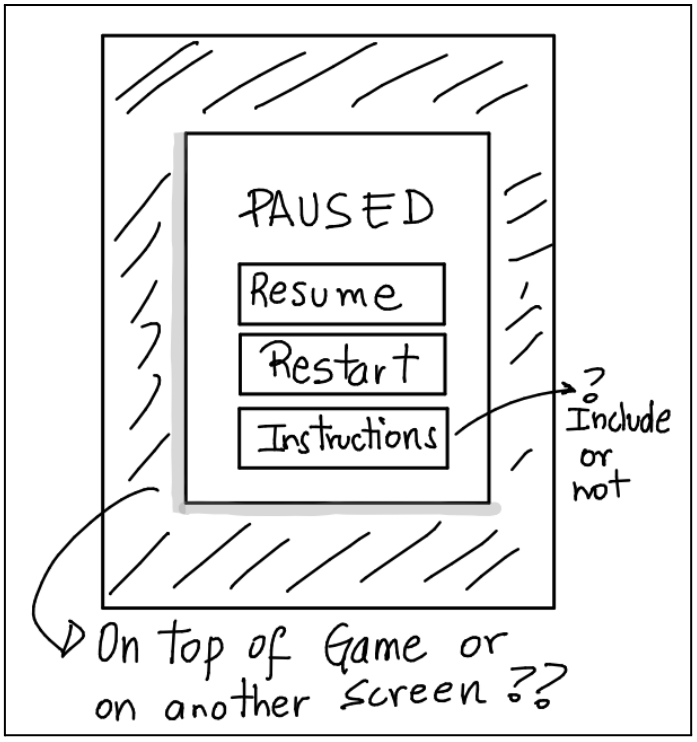

Figure 4. Show when Pause button is pressed.

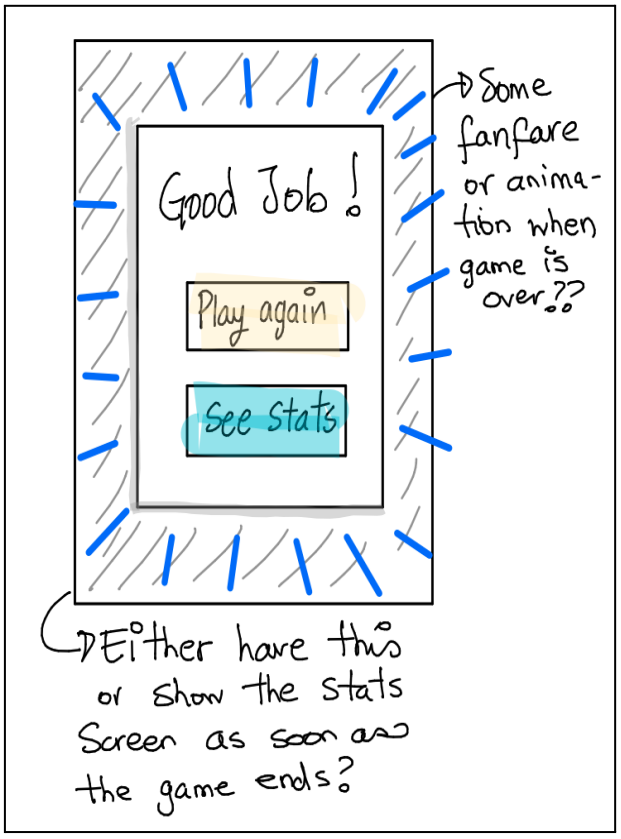

Figure 5. Show when game ends (time runs out)

When the time runs out, the game should indicate it with a screen like the one shown in Figure 5, from here the user should be able to start a new game or see the stats that he/she achieved during the game. I also decided to include a sound to be played when there are

five seconds left in the timer, because it happens that the user might be too concentrated and not notice that the time is running out until it is too late.

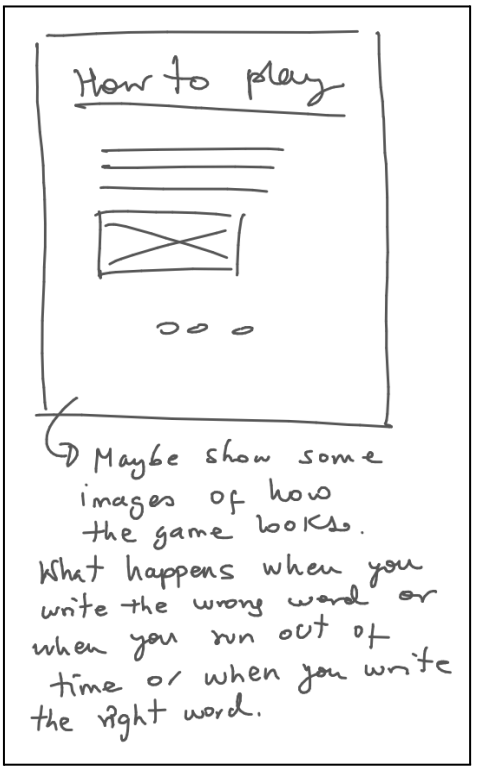

Figure 6. Instructions Screen (1)

The instructions page was the last screen I implemented since I wanted to include pictures of how the game actually looks. Nonetheless, I knew that I needed to decide whether the instructions would be scrollable sideways or downward. I decided on doing it sideways because that is the choice I usually see in apps, and because I think it is easier to see in this structure how much is left to finish reading them.

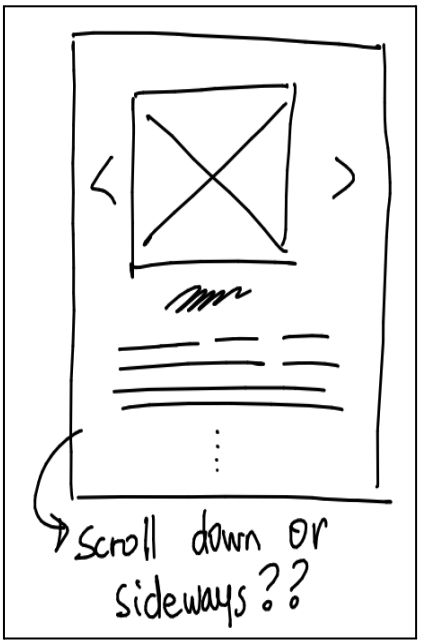

Figure 7. Instructions Screen (2)

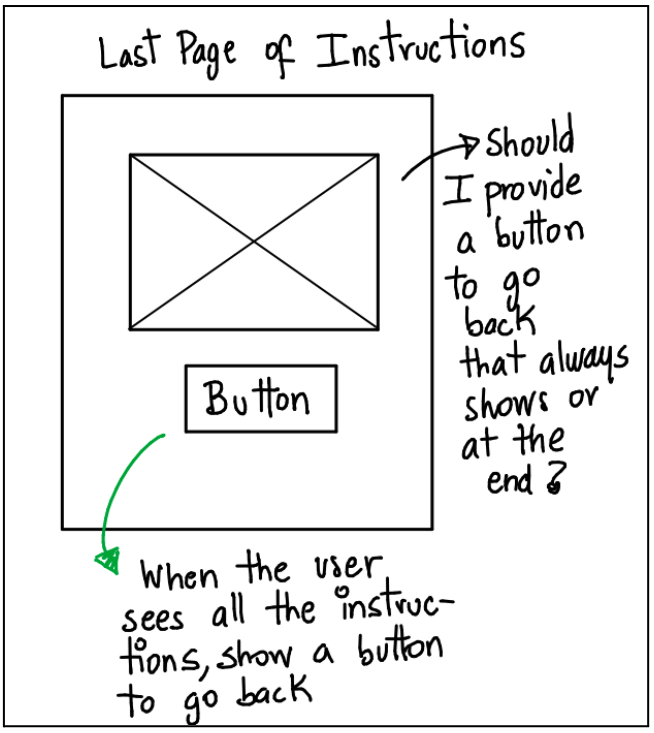

FIgure 8. Instructions Screen (3)

Usually when you reach the end of the instructions you have a way to go back, but in my app's case, you can reach the Instructions Screen from several Screens, so the user should be able to go back to the screen from which the Instructions were called.

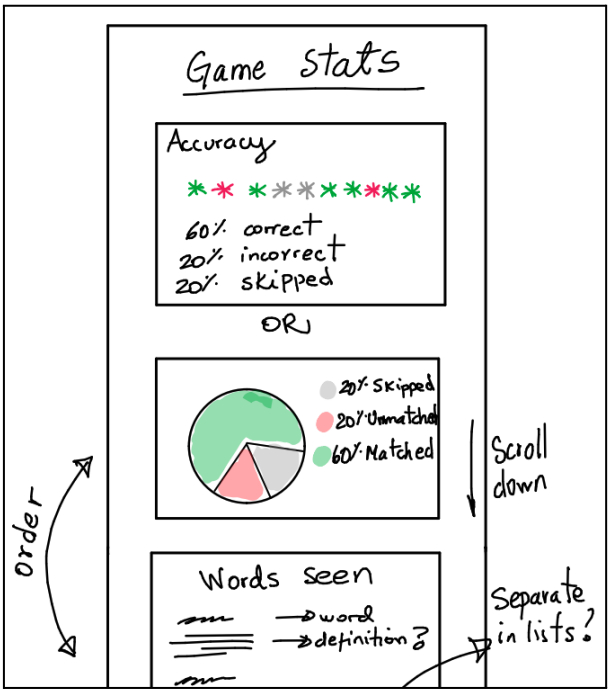

Figure 9. Stats Screen (1)

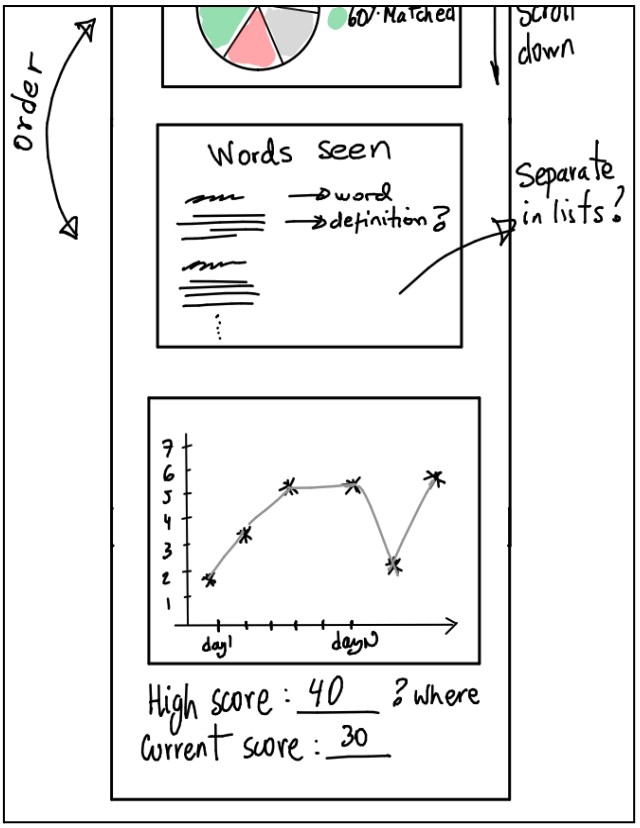

Figure 10. Stats Screen (2)

The Stats Screen was the most difficult to decide upon. There is plenty of information that could be shown about the game, and many ways to show it. I wanted something simple, yet informative. For this type of game, accuracy is quite important: From all the questions shown, how many were answered correctly. In Figure 9 you can see that there were two ways which I decided I could show the accuracy, and upon requesting feedback, I chose the pie chart (apparently it was easier to read/understand).

Since the end of the game is to help you remember words, I thought it was important that the game showed a list of the words and definitions “learned” during that session, that is why I wanted to include them in the Stats screen even though this is not a statistic of the game. I also decided upon creating a separation between the words matched, unmatched and skipped in this section of the Stats Screen.

Lastly, I wanted to include a graph that showed the points achieved during the game. For simplicity, I decided to only include data for the last 10 games. Ideally, this section of the graph would also include a congratulating portion, if the user achieved a high score, but this was not included in the end (for time reasons).

I did not create some high fidelity wireframes, I only used these few drawings to have an idea of the elements that should be included in the game. In this way I was able to assess the type of libraries I should use, the amount of screens to be created and how they were to be connected.

To choose a color palette I looked at different apps and realized that for this type of game (learning games) blue hues are usually employed, so I decided that when the skeleton of the app was ready I was going to include background colors and images with that characteristic.

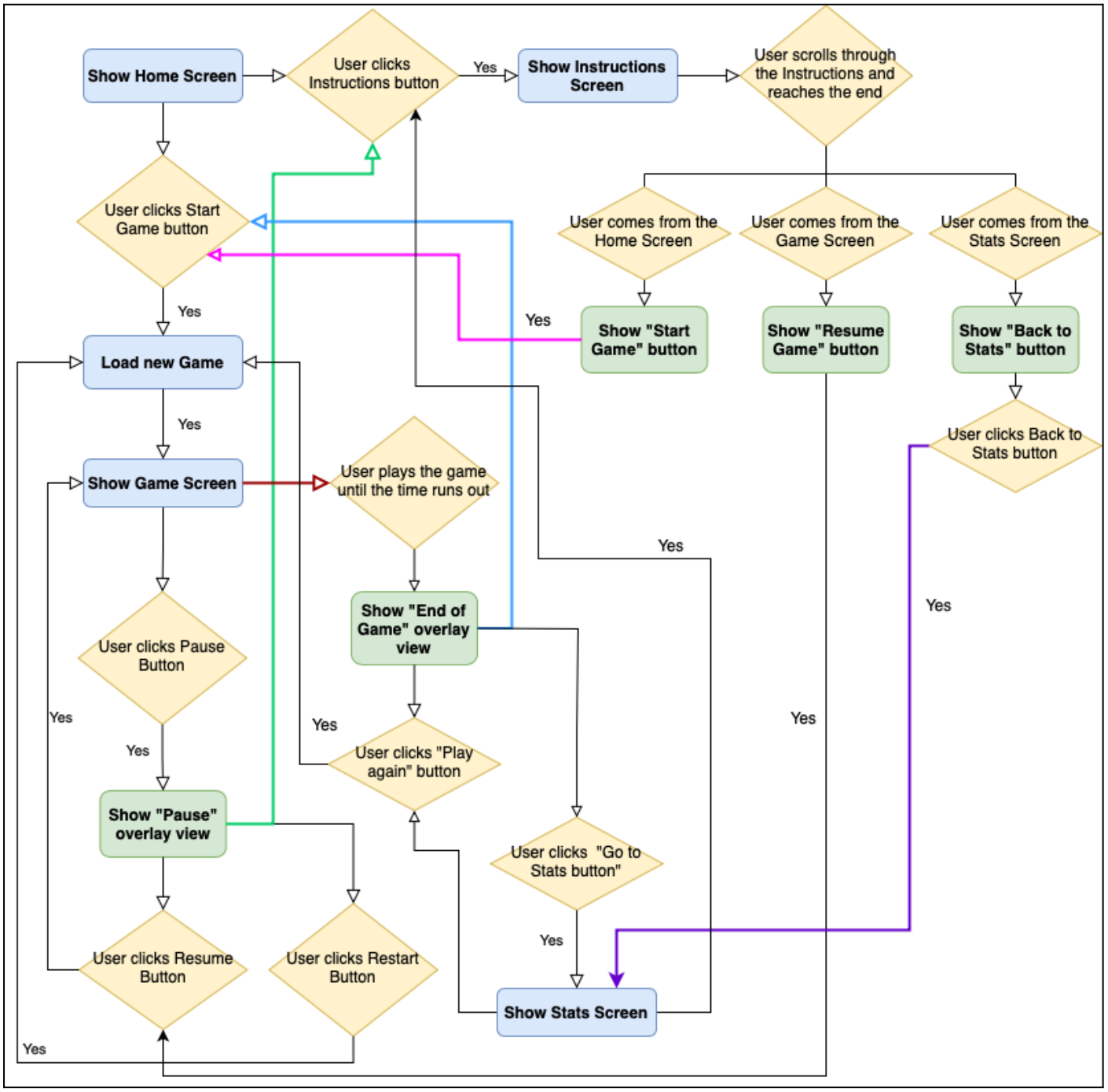

Figure 11. User Flow Diagram.

Something I did create was Figure 11, because I needed to establish how the screens linked to each other. Particularly, how the Instructions Screen was going to be shown when called from the Home, Game and Stats Screen. The idea was that if the user goes to the Instructions from the Home Screen, then at the end of the Instructions there would be a button to start a new game. If the user sees the instructions through the Game Screen, then at the end of the instructions a button will be shown to resume the game. Lastly, if the user sees the Instructions through the Stats Screen, the button at the end of the instructions should allow the user to go back to the Stats screen.

In summary:

- The game consists of spelling a word that matches a given definition.
- Four screens will be created: Home, Game, Instructions and Stats screens.
- Users will be able to pause, resume and restart a game.
- The game will be timed and give points to the user according to the difficulty of the words.

3. Prototyping and Development

After creating the User Flow diagram and before starting to code, I used post-it notes to write the most important elements of the app that should be coded, as seen in Figure 12.

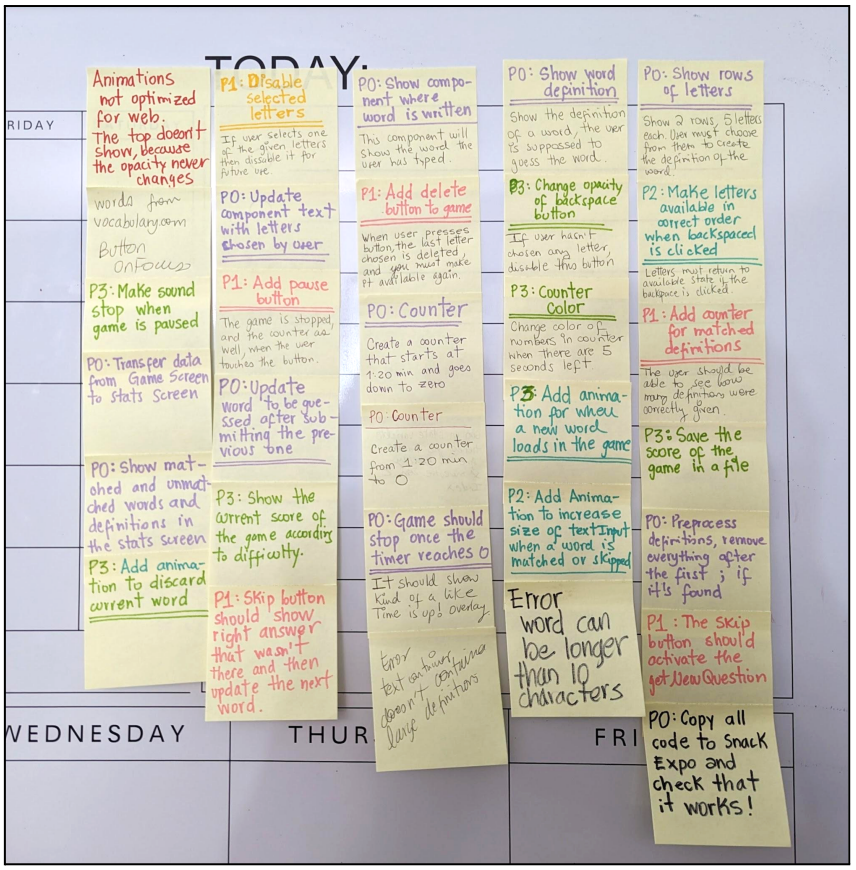

Figure 12. App's log

For this I assigned priority to elements of the page that should be implemented, P0 through P3, being P0 the top priority. Once I implemented a functionality described in one of these post-it, I removed it from my list of “to-do”s and put it in my list of “finished”.

One of the first decisions I had to make was how to access a dictionary of words and how to assign a difficulty to them. I knew I could make use of some APIs to get the “dictionary” functionality (e.g. vocabulary.com, wordsapi.com, GNU collaborative International), but the question of the difficulty was still an issue. There were plenty of libraries I could have used that had a dictionary of English words, but I could not find one which assigned a difficulty level to them, so I decided to create my own using lists of words used for English tests. For this I scrapped the words from www.vocabulary.com (see an example here).

What I did was scrape a list of words for levels A1, A2, B1, B2, C1 and C2 and assigned them a difficulty of easy, standard, intermediate, difficult, hard and expert respectively. The easiest level assigns 500 points, the second easiest 1000 and so on (in increments of 500). I had to do quite a bit of preprocessing to this data to make it ready to be used in my app. For this I created a Jupyter Notebook and used Python, Pandas and Numpy to pre-process the data: remove words less than 3 characters long, remove words longer than 13 characters, assign only one definition per word, removing proper nouns, removing “words” that contain spaces (phrases), remove words for which the definition is too long (as for scientific naming), adding the difficulty column, etc. I included the PreprocessingWords.ipynb file (the jupyter notebook) along with this project as well as the data source I used so that you could see the whole process.

Finally I created a *word_list.js* file which I used in my app to obtain the functionality I desired.

I searched for libraries I could use to make some interesting animations in my app, although I only ended up using one for the buttons on my Home Screen ('react-native-really-awesome-button'), one for the slides in the Instructions Screen (react-native-slideshow-improved) and one for the timer in the Game Screen (react-native-countdown-circle-timer). The other animations showed in the app I coded them myself.

I coded and tested everything in my own computer first, and was only testing the code in the iOS simulator. When I thought the game was almost finished I tried running it in the Android emulator and it did not work. I thought it had something to do with the emulator so I updated it, but to not avail. Then I decided to test the code in an expo snack, and it immediately worked for the iOS emulator but kept failing to load in the Android emulator. After hours of trying to find the reason for this I was able to realize that the problem had to

do with the type of values I was giving to the text elements. They were dynamic (percentage values), and this seemed to work for iOS but it did not work in Android. Once I changed every text size to a fixed number the app could load in both iOS and Android. That is the reason why some sizes are fixed values and some others are dynamic.

I also realized at this point that when I opened the app on the Web, it did not load some of the elements. The app did not show any errors, it just did not show the top elements in the Game Screen. After searching for hours the reason for this bug I found out that the problem was the Animations library, which is “not optimized for web”. If I had realized this sooner I would probably have tried to use another library, but at this point I was too far along the project to start over. That is the reason why the app is not apt for the web.

When I created the Screens, of which there are four total, I was in doubt of whether the Game Screen should be divided into several screens or not. This is because I did not know if I should include a “Paused” and “Game Over” Screens or to somehow include an overlay when the game was paused or finished.

Upon trying both options I decided that the one that felt more natural was the second option, using overlays when the user paused the game. For this I had to make sure that if the game was paused, the timer should be paused too, and if any sound was being played at the time (during the last five seconds of the game) then that sound should also be paused and resumed once the game was resumed.

4. User feedback and testing

I was able to get friends with different devices to test my application. This is the list of phones were the app was tested and the comments/ suggestions they gave me:

- Pixel 6: this friend found two bugs. If you went to the Instructions Screen from the Home Screen, and then once you started the game and hit the Pause button and went to the Instructions Screen again, then instead of showing the instructions from the start, the app would show the last page of the instructions since that was its last state. The same thing happened when she tried to access the Instructions from the Stats Screen. I fixed both issues. She also told me that she did not find it annoying to have a sound played when the time was running out (which was something I was worried about) and said that it would have been fun to have some kind of fanfare played when the game was over if you got a high score. I did want to include something like this, and I was going to use a library called react-confetti-explosion, but unfortunately I did not have the time to add this functionality. Another comment

was that I might play different sounds when a word was spelled correctly and another for incorrect or skipped words, but I thought it was enough with the animation I added (increasing the size of the word container and changing its background color depending on the match).

- Pixel 7: This one friend found two bugs as well. One had to do with the library I used for the slider animation in the Instructions Page. I used the library named 'react-native-slideshow-improved' for this functionality. But I had to make some modifications to the Slideshow.js file in that library after installing it in my project since it did not look as I had planned when I created my wireframes. This user noticed that when you swipe the instructions the little gray dots at the button do not change their color as they should, but they do if you do not swipe the screen but click those buttons in order to see the next page. I had to check that the changes I made to the Slideshow.js file did not damage the proper functionality of the slider, but when I checked the example given in the npm page of the project, I realized this error was already there, so it was not a bug I introduced (but I did not fixed it). The second bug had to do with the Instructions page as well. When I fixed the issue for my first tester I somehow introduced another bug where if the user paused an ongoing game and then went to the instructions screen, the game would not resume when the user went back to the game Screen, but it restarted. I was able to find and fix this bug as well.

- Samsung S22: No bugs were found by this user but he did have a comment about the size of the given letters, apparently they were too small. This was my dad, I know he has really big fingers and always has troubles with apps because of this. I asked my other testers if they thought this was a problem and none of them had the same issue. This is why I decided to keep the size of the letter buttons as they were.

- Iphone 14: This tester was the teacher that helped me with the game idea. She found the game challenging enough, and quick enough for a person like her (who does not have too much free time with a small child at home as well). She commented that it would be great if the game also had levels, so you could try easy or harder levels on demand. I did not add this functionality but I certainly think it would have been a great addition. One thing that this particular friend helped me with was checking that the sound was actually played when the time was running out. I was afraid it might not work because in the Iphone simulator in Expo no sound is emitted (although it works on the Android emulator in expo and in both Android and iPhone simulators in my computer).

- Iphone 12: This friend found no bugs. Her comment was that for someone who did not have too much practice with this kind of game, sixty seconds seemed too short a time to get the hang of it. So, maybe the learning curve of this game is steeper than I had anticipated, though no other tester gave me the same insight so I decided to leave the timer with 60 seconds.

- Samsung Galaxy Tab S6 and Pixel 6 pro: These were my devices, and since I found and fixed more bugs that I can remember, I will just state that the game works properly on these devices.

Some final comments about the application:

- All pictures used in the game came from an open source.

- For some reason, the Expo Snack could not automatically add the 'react-native-slideshow-improved' library, so I had to copy the Slideshow.js file directly to my app's folder and use it from there. That is the reason why you will not see this dependency in the expo snack but it is present in the package.json file in my Project folder.

- The game seems to play slower when it is run on the emulators directly from the snack expo page, but runs properly in tablets

I will finish the report with some pictures of how the game's Screens look:

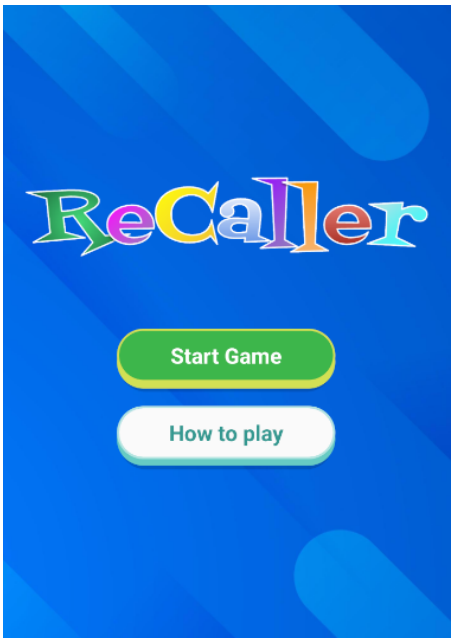

Figure 13. Home Screen

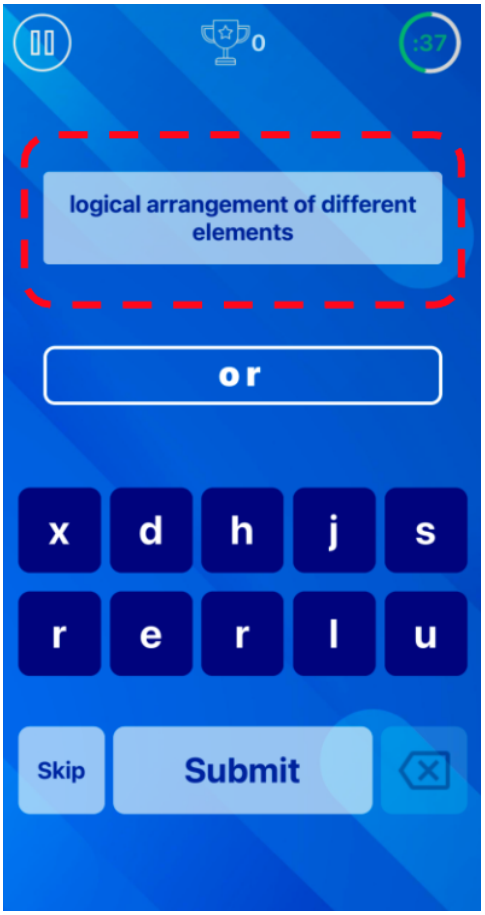

Figure 14. Game Screen (definition section highlighted)

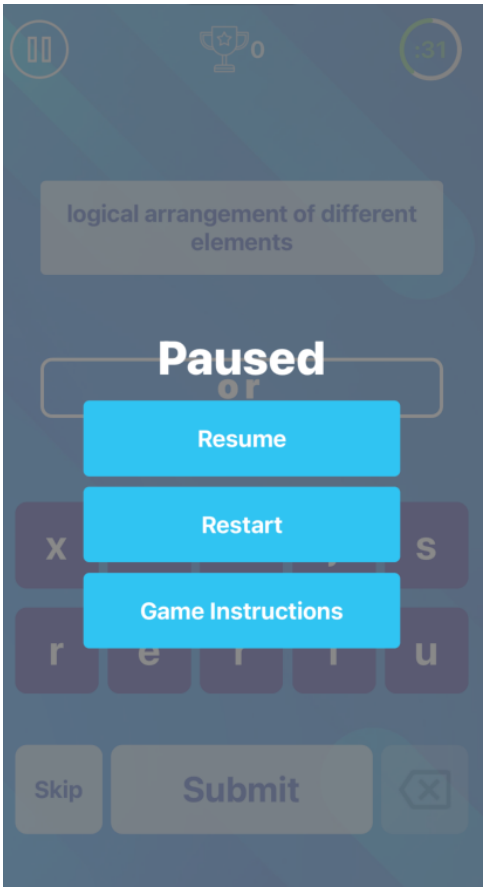

Figure 15. Game Screen (paused state)

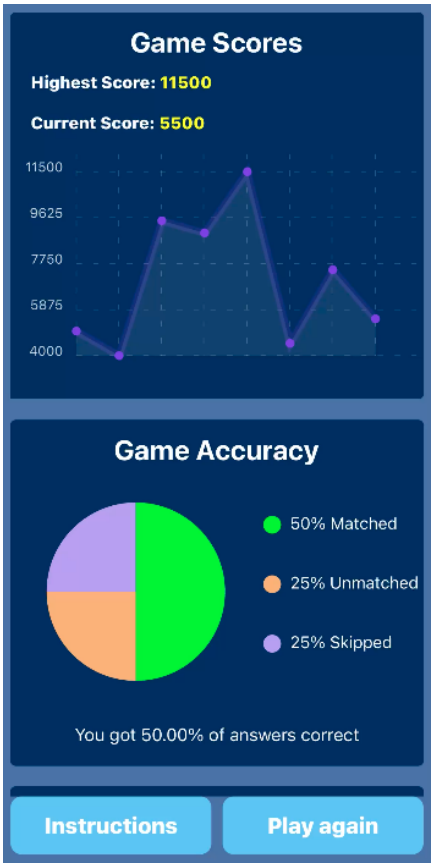

Figure 16. Stats Screen (showing game scores)

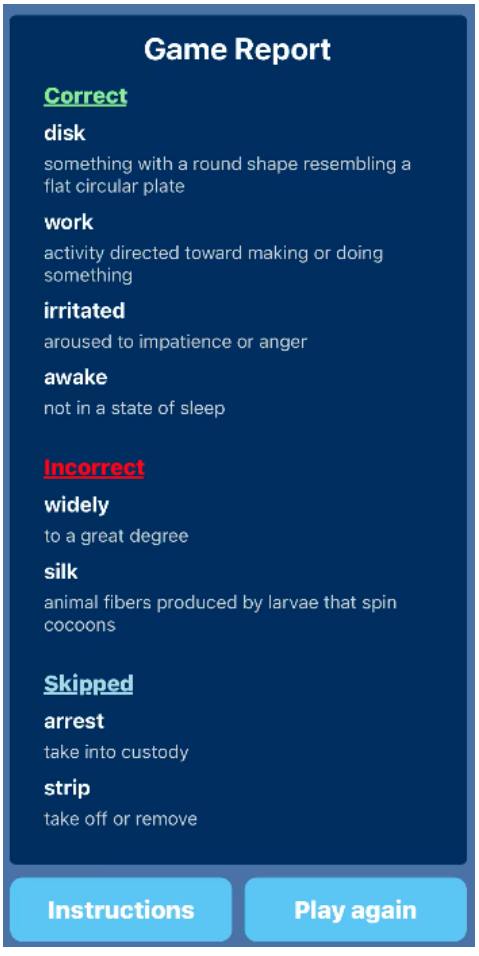

Figure 17. Stats Screen (showing list of words seen during game)

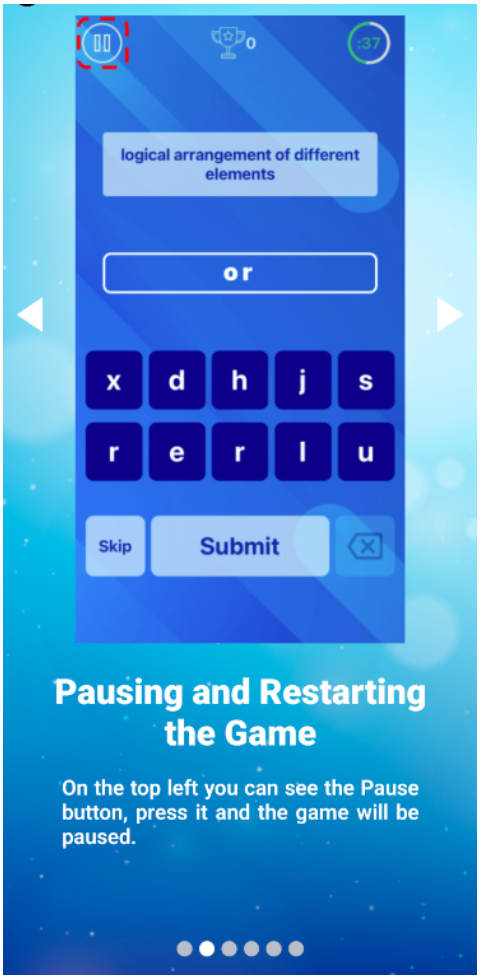

Figure 18. Instructions Screen (showing how to pause the game)

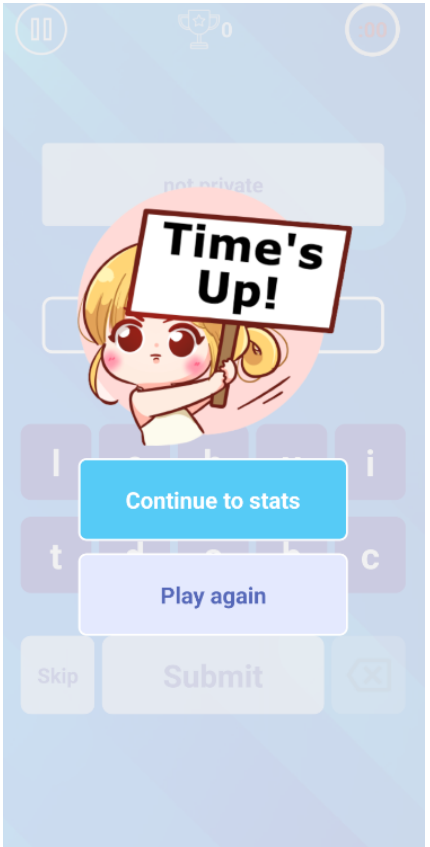

Figure 19. Game Screen (game over state)