COSC2659

iOS Development

Assignment 2

Student: Trinh Van Minh Duc - s3915177

Due Date: 06 September 2023

Table of Contents

A. Introduction

For the assignment 2, I created a game called ‘Minetiplier’. ‘Minetiplier’ is a gambling game that bring a nostalgic vibe along with the thrill from the unknow outcome of the game. The game is a combination of gambling and a classic childhood game called “Minesweeper”. “Minetiplier” allows player to test their decision-making skill despite being based purely on luck. By basing on luck, the game provides a thrilling experience as the outcome is random and boosting players excitement with their bet. The thrilling experience from placing a bet with random outcome is one of the primary inspirations for me to make this game. Additionally, my fondness for the childhood game “Minesweeper” is another inspiration to make “Minetiplier”.

B. Project Description

a. How To Play

1. Place a bet: bet: Enter the amount of money to bet on the game.
2. Select number of bombs: Select from 1 to 15 bombs, the higher the risk, the higher the reward.
3. Flip the cards: Select a card to flip out of the 16 cards on the screen. The reward is multiplied by the number of diamonds found. However, if you flip a bomb, you lose.
4. Cash out: Cash out at any time during the game.

b. Tips and Tricks

1. Manage your balance: Avoid going all-in in 1 turn, set a cap limit every time you play.
2. Consider the risk and reward: The more bomb you chose, the higher the risk but it comes with higher reward. Consider the risk and reward wisely with you bet.
3. Cash out at the right time: You might want to cash out when there are more bombs than diamond.
4. Knowing to stop: Don’t be greedy, stop when you can feel the luck is not with you.

C. Implementation Details

a. Main features

1. Welcome view

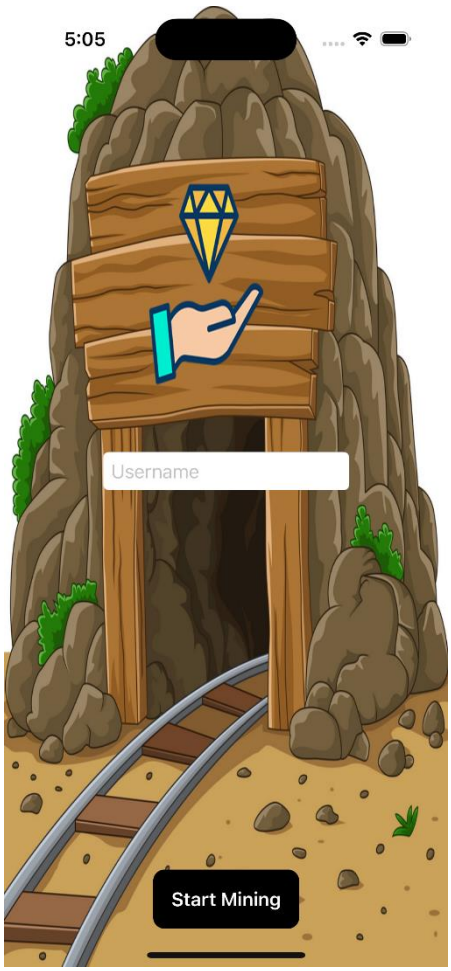

Figure 1: Welcome View

In the welcome view, I use a simple "**ZStack**" to display the background image, a "**TextField**" for user to input their username and a button to load in the username into the "**EnvironmentObject**".

2. Game view (Main View)

Figure 2: Game View

Figure 3: Ongoing Game View

The game view is a combination of 3 other views. The card section is displayed using "**LazyVGrid**" and "**GeometryReader**" to distribute the cards equally. The whole view is controlled using 2 "**State**" variables stored in "**UserDefaults**" along with "**.allowsHitTesting(Boolean)**" to prevent users to interact with the card when they are not in a game and interact with the game setting while they are in a game.

3. Instruction view

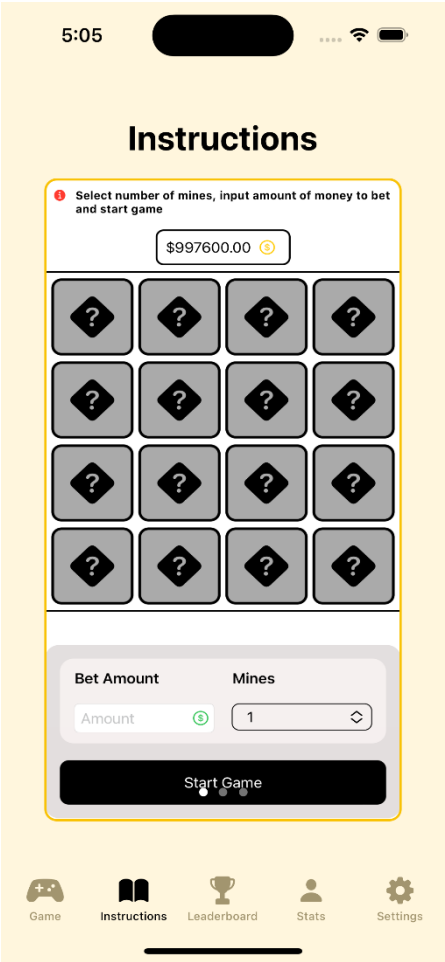

Figure 4: Instruction View

To display the image for the instruction, I used "**PageTabViewStyle**" and String array. The "**TabView**" act as a slider so that users can view each instruction individually with a swipe.

4. Leaderboard view

Figure 5: Leaderboard View

Figure 6: Personal Leaderboard View

To display the leaderboard, I used "**List**" as the data is sorted when it is stored to the array and JSON file. To display a user's highscores, I implemented a filter based on the username into the list.

5. Stats and Achievements view

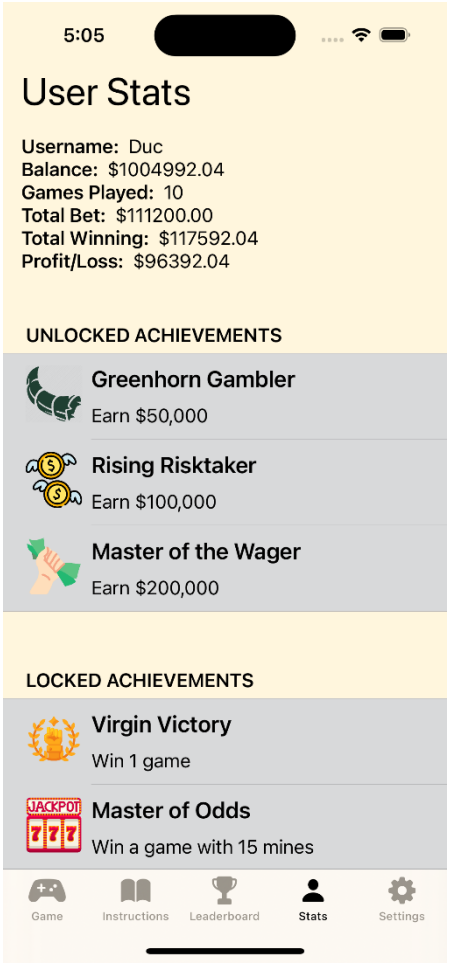

Figure 7: Stats and Achievements View

In this view, I used "**EnvironmentObject**" to store and display the user stats. For the achievements, I implemented a "**List**" with 2 sections to divide it into locked and unlocked achievements.

6. Setting view

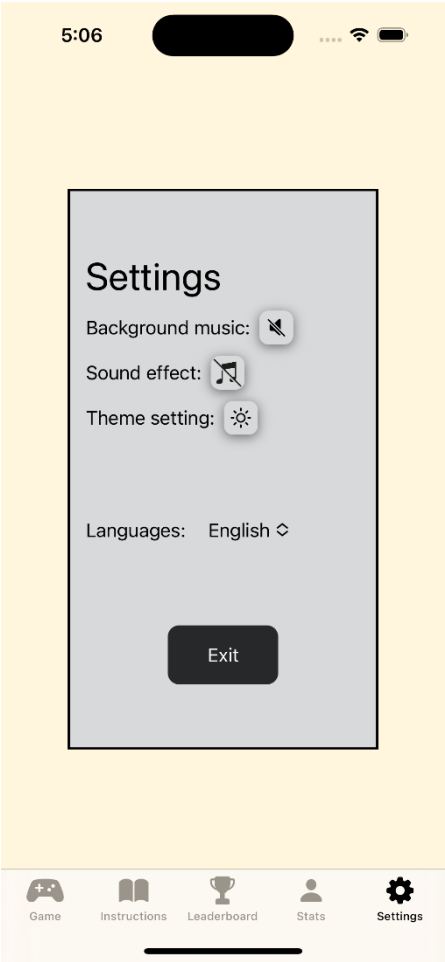

Figure 8: Setting View

For the Setting View, I implemented “GeometryReader” with “offset” to display the frame in the centre of the screen. There are 3 buttons for user to toggle the theme and sound setting with a picker to select the language of the game. A button is implemented for the exit and a function that will store the user’s data to a JSON file is called upon action.

7. Background music/Sound effects

To implement both background music and sound effects, I created a class called “**SoundManager**” with 2 “**AVAudioPlayer**” so that background music and sound effects can be play at the same time. Furthermore, to enable sound effect and background music toggle, I created 2 buttons that will toggle 2 Boolean values that is declare as a “**State**” and stored using “**UserDefaults**”. Please refer to “**Figure 8**” for UI demonstration.

8. Device compatibility

Figure 9: iPad Game View

For device compatibility, I used "**GeometryReader**" for most of the views. "**GeometryReader**" allows the views to read the screen size to manipulate the size of items accordingly. For the rest of the view, refer to Appendix.

b. Advance features

1. Save/Resume

To implement the Save and Resume feature, I created an "**ObservableObject**" that allow notify views when a data is changed and save it into "**UserDefaults**" as "**UserDefaults**" will be store in the app and load upon view loaded.

2. Game progression and levels

Figure 10: Game difficulty

For the game difficulty, "**Menu**" is implemented instead of "**Picker**" so that I have more control over the UI. The difficulty level increases as the number of bomb increases.

3. Multi-language

Figure 11: Multi-language

To support multi-language, I used "**Localization**" and pass it into the environment using "**.environment(\.locale, .init(identifier: lang)**". This dynamic feature allows the game to support more language in the future.

4. Theme setting

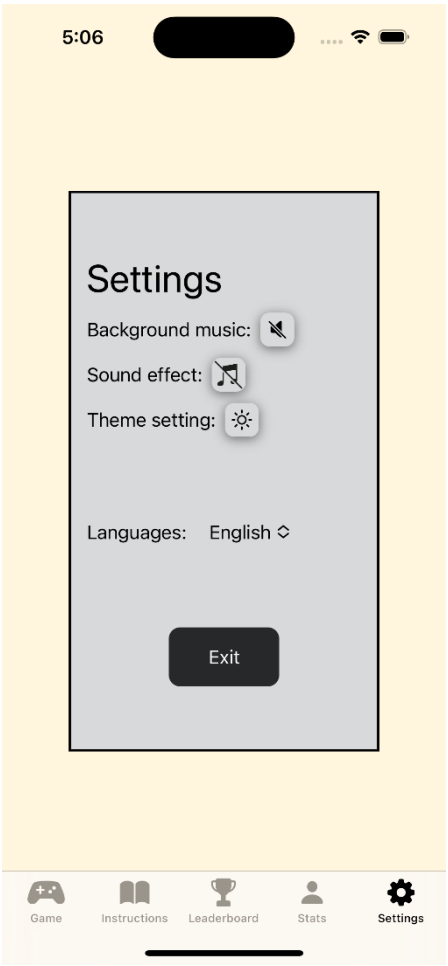

Figure 12: Light mode

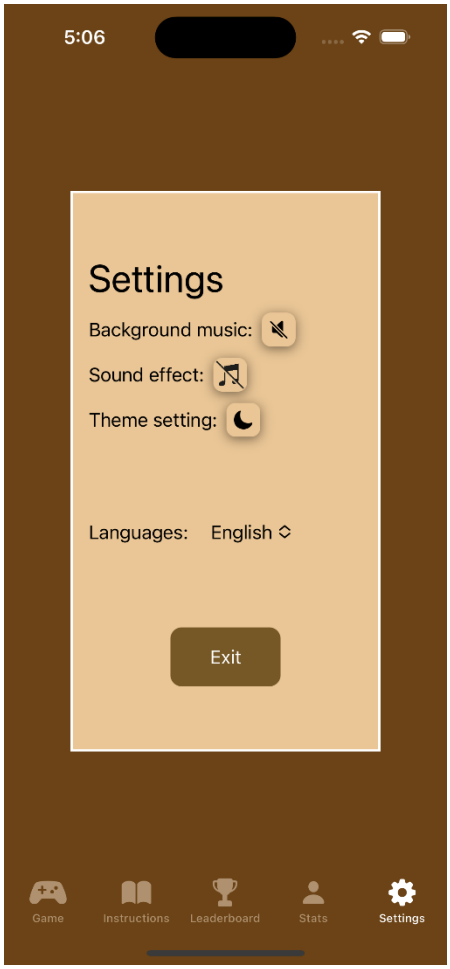

Figure 13: Dark mode

To implement the theme setting, I create an "**AppStorage**" variable and pass it into the environment using "**.environment(\.colorScheme, bool ? .dark : .light)**". To switch the theme, a button is implemented for user to toggle the variable.

D. Demonstration Video

- https://youtu.be/Y0ei-rCwdkw

E. Limitations/Bugs

- Sometime the Save and Resume run into a bug where the last flipped card doesn't load when re-open the app despite being save to the "**UserDefaults**".

F. Conclusion

In conclusion, the game work as intended. The game is capable of bring a thrill and exciting experience for the players with purely random outcomes and initiate their decision-making skill

to be in profit. While working on this project, I was able to learn more advanced SwiftUI features such as **“AppStorage**” and “**UserDefaults**” to store variable in the app, “**ObservableObject**” to make real-time change to variable, “**GeometryReader**” to design layout based on the screen size and “**AVAudioPlayer**” to play sound in the game. I also learned to use “**Localization**” and manipulate the environment locale to change the language of the app dynamically. However, there are limitations to the app that I could improve in the future. The first improvement I could make is to have a better color palette for the app to be more appealing. Another improvement is to have user register for the account with their information and implement real money into the game. Lastly, I could improve animation of the game to be more engaging.

G. References

[1] O. Baumeister. “SwiftUI – Memory Game.” Youtube.com. https://www.youtube.com/watch?v=aJ9kKX6Ak3k (accessed Aug. 15, 2023).

[2] dbestech. “SwiftUI Image Slider | Auto Play.” Youtube.com. https://www.youtube.com/watch?v=VYxxzrlS8q0 (accessed Aug. 16, 2023).

[3] K.C. Chen. “SwiftUI Tups 1 – Dynamic Localiztion without Change system’s Language.” Medium.com. https://kowei-chen.medium.com/swiftui-dynamic-localization-tricks-87c37a6db3e7 (accessed Aug. 27, 2023).

[4] P. Hudson. “How to provide relative sizes using GeometryReader.” Hackingwithswift.com. https://www.hackingwithswift.com/quick-start/swiftui/how-to-provide-relative-sizes-using-geometryreader (accessed Aug. 15, 2023).

[5] Stackoverflow. “How to use UserData Observable Object in SwiftUI?.” Stackoverflow.com https://stackoverflow.com/questions/62372188/how-to-use-userdata-observable-object-in-siftui (accessed Aug. 21, 2023).

[6] P. Hudson. “How to disable taps for a view using allowsHitTesting().” Hackingwithswift.com. https://www.hackingwithswift.com/quick-start/swiftui/how-to-disable-taps-for-a-view-using-allowshittesting (accessed Aug. 25, 2023).

[7] P. Hudson. “How to show a menu when a button is pressed.” Hackingwithswift.com. https://www.hackingwithswift.com/quick-start/swiftui/how-to-show-a-menu-when-a-button-is-pressed (accessed Aug. 20, 2023).

[8] Apple. “Picker” Developer.apple.com. https://developer.apple.com/documentation/swiftui/picker (accessed Aug. 15, 2023).

H. Appendix

Figure 14: Loss Game View

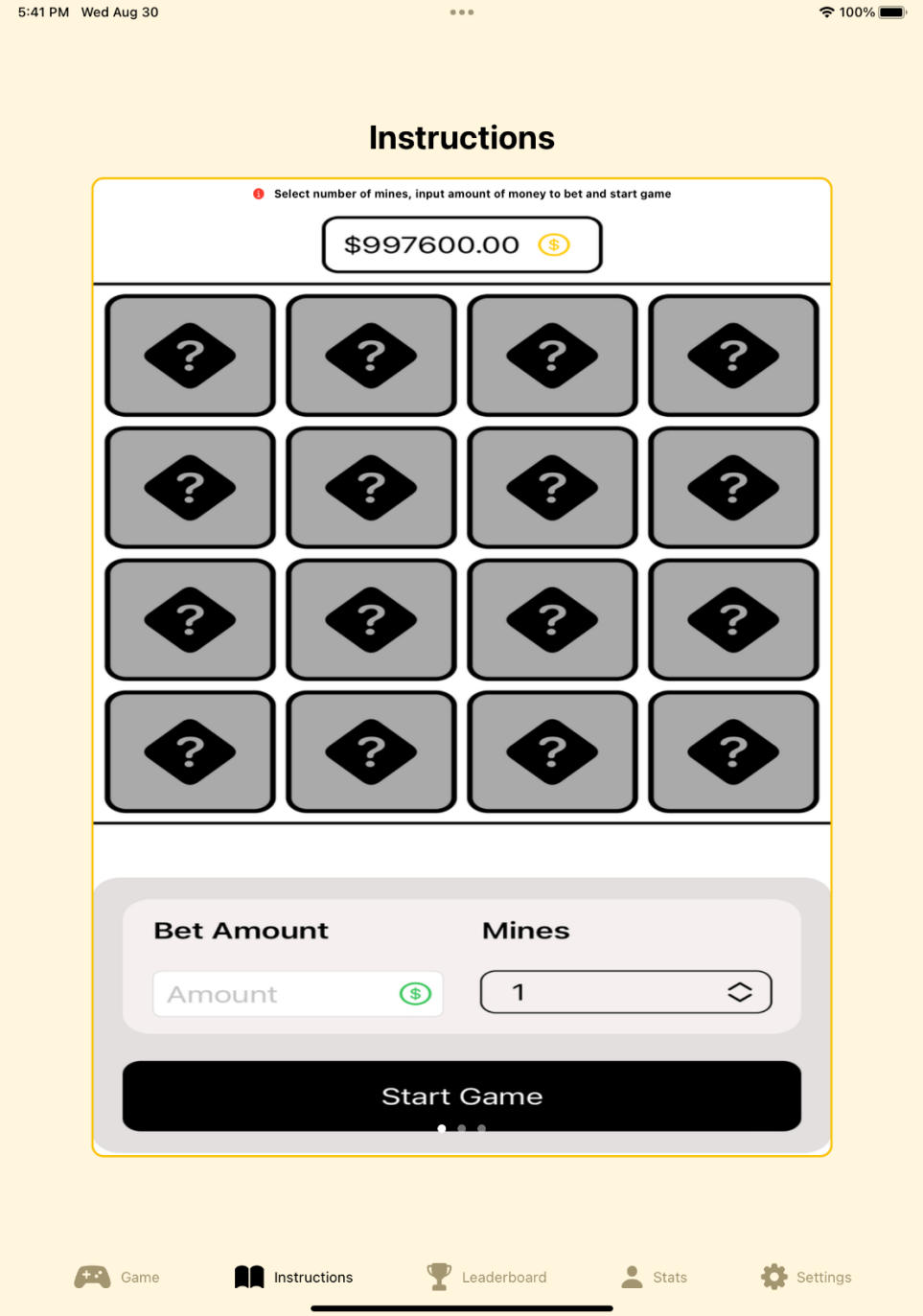

Figure 15: iPad Instruction View

Figure 16: iPad Leaderboard View

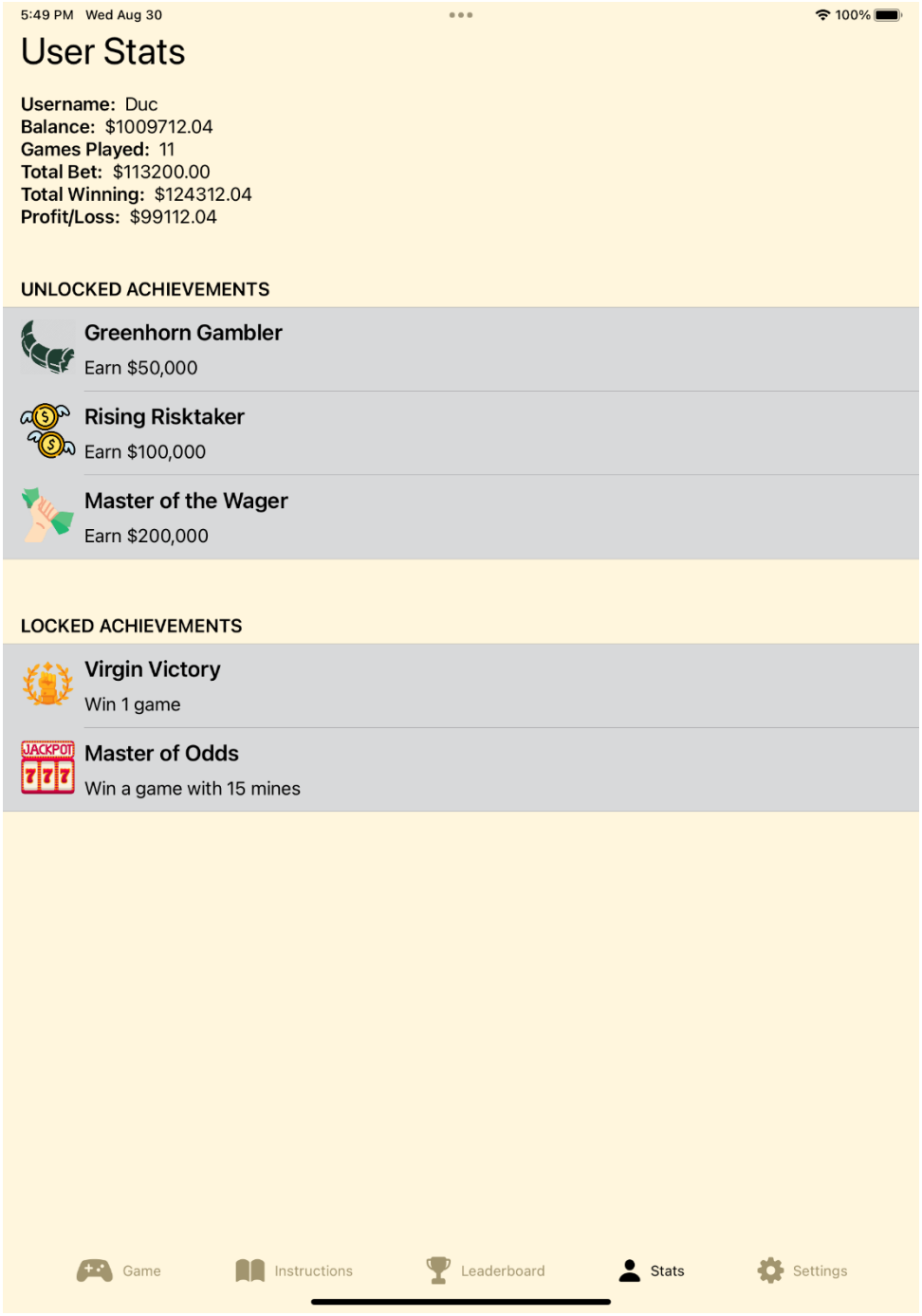

Figure 17: iPad Stats and Achievements View

Figure 18: iPad Setting View