A logo of a university

Description automatically generated

COSC2659

iOS Development

Assignment 2

Student: Trinh Van Minh Duc - s3915177

Due Date: 06 September 2023

**Table of Contents**

[**A.** **Introduction** - 1 -](#_Toc144912265)

[**B.** **Project Description** - 1 -](#_Toc144912266)

[**a.** **How To Play** - 1 -](#_Toc144912267)

[**b.** **Tips and Tricks** - 1 -](#_Toc144912268)

[**C.** **Implementation Details** - 2 -](#_Toc144912269)

[**a.** **Main features** - 2 -](#_Toc144912270)

[**1.** **Welcome view** - 2 -](#_Toc144912271)

[**2.** **Game view (Main View)** - 3 -](#_Toc144912272)

[**3.** **Instruction view** - 4 -](#_Toc144912273)

[**4.** **Leaderboard view** - 5 -](#_Toc144912274)

[**5.** **Stats and Achievements view** - 6 -](#_Toc144912275)

[**6.** **Setting view** - 7 -](#_Toc144912276)

[**7.** **Background music/Sound effects** - 7 -](#_Toc144912277)

[**8.** **Device compatibility** - 8 -](#_Toc144912278)

[**b.** **Advance features** - 8 -](#_Toc144912279)

[**1.** **Save/Resume** - 8 -](#_Toc144912280)

[**2.** **Game progression and levels** - 9 -](#_Toc144912281)

[**3.** **Multi-language** - 10 -](#_Toc144912282)

[**4.** **Theme setting** - 11 -](#_Toc144912283)

[**D.** **Demonstration Video** - 11 -](#_Toc144912284)

[**E.** **Limitations/Bugs** - 11 -](#_Toc144912285)

[**F.** **Conclusion** - 11 -](#_Toc144912286)

[**G.** **References** - 11 -](#_Toc144912287)

# **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”.

# **Project Description**

## **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.

## **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.

# **Implementation Details**

## **Main features**

### **Welcome view**

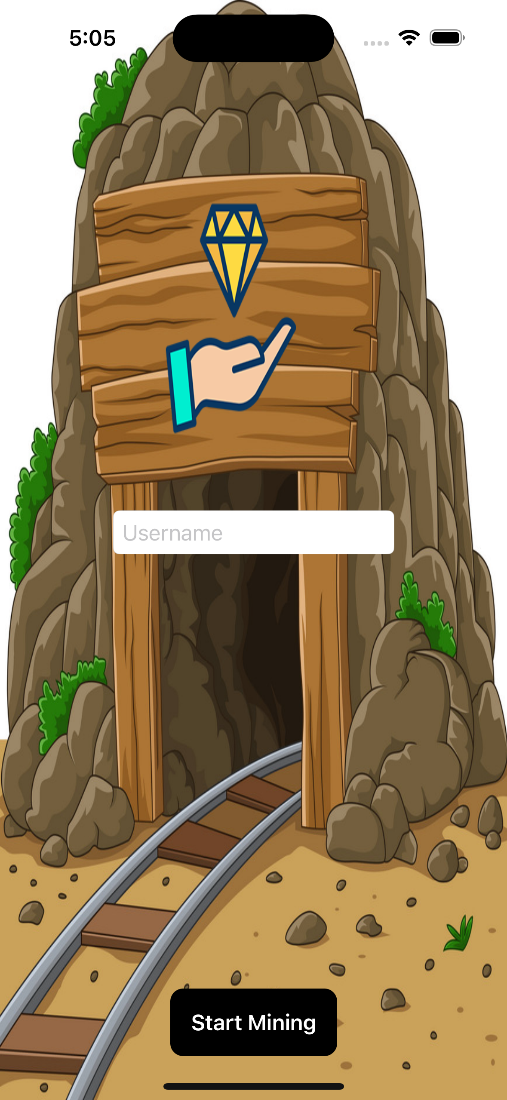


Figure 1: Welcome View

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**”.

### **Game view (Main View)**

 A screenshot of a game

Description automatically generated

Figure : Game View Figure : 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.

### **Instruction view**

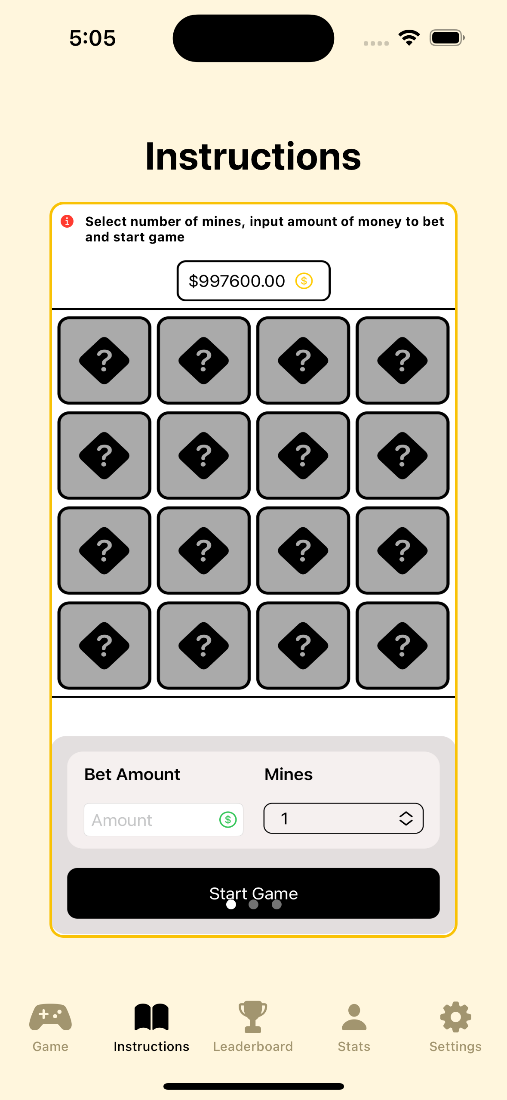
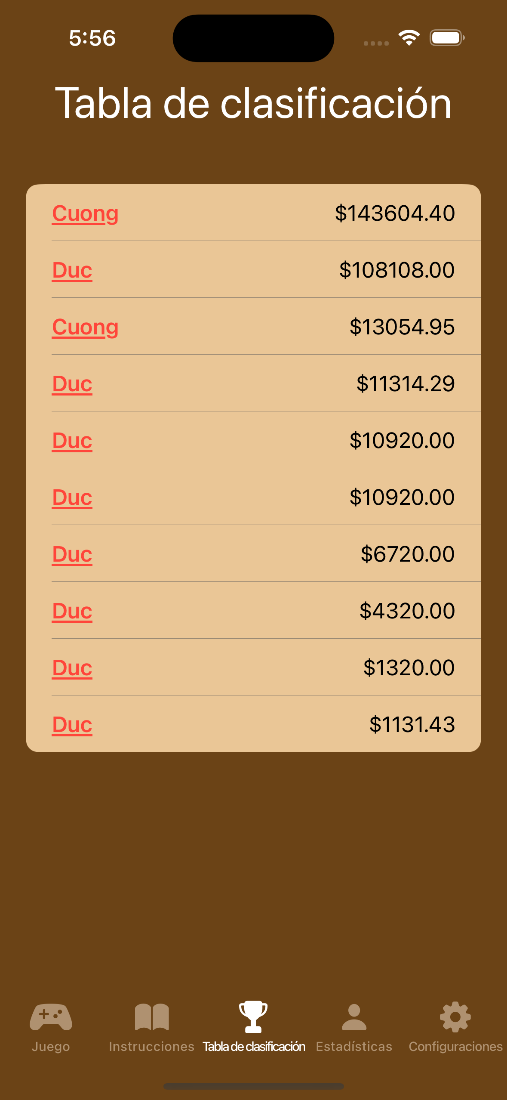


Figure : Instruction View

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.

### **Leaderboard view**

 A screenshot of a phone

Description automatically generated

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.

### **Stats and Achievements view**

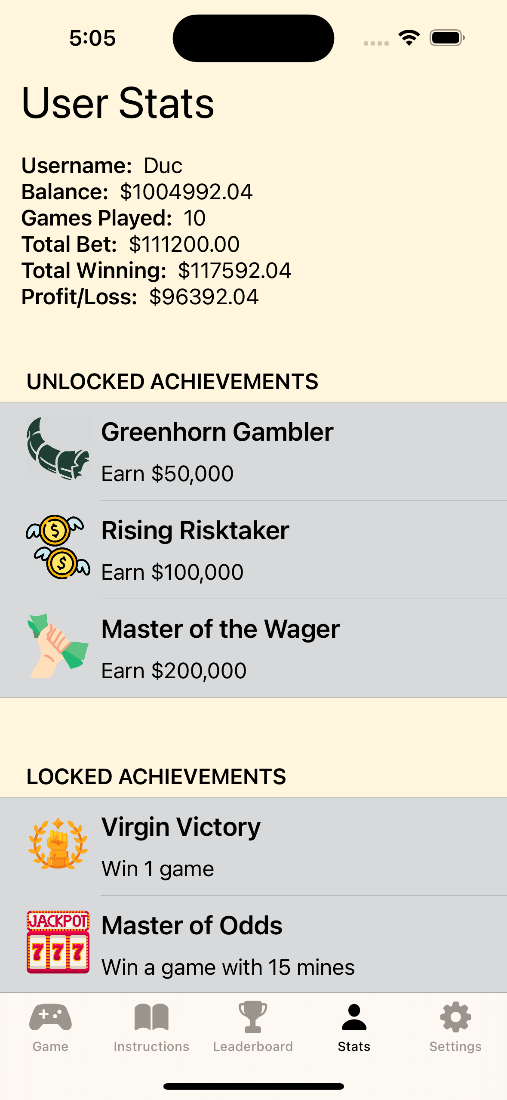


Figure 7: Stats and Achievements View

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.

### **Setting view**

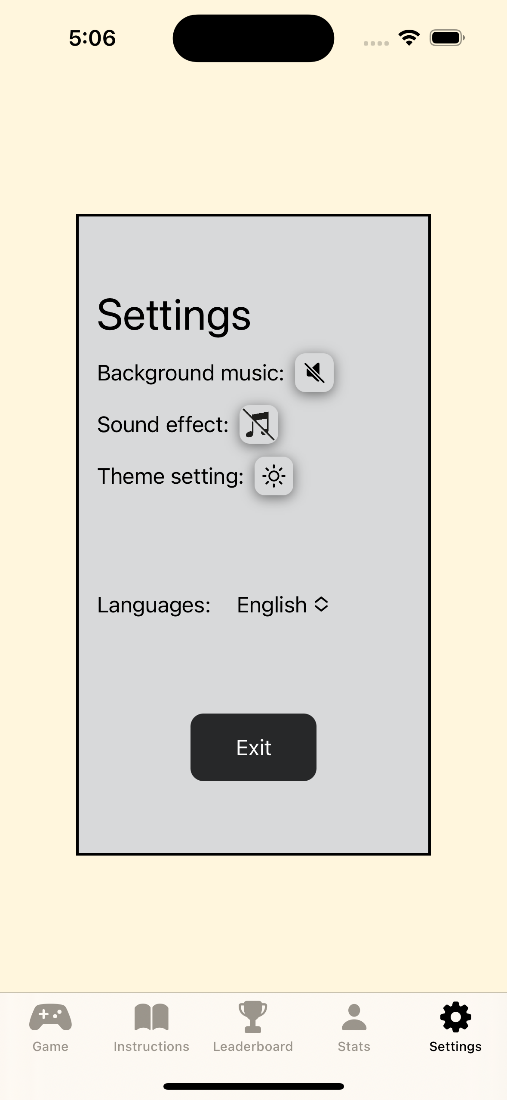


Figure 8: Setting View

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.

### **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.

### **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.

## **Advance features**

### **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.

### **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.

### **Multi-language**

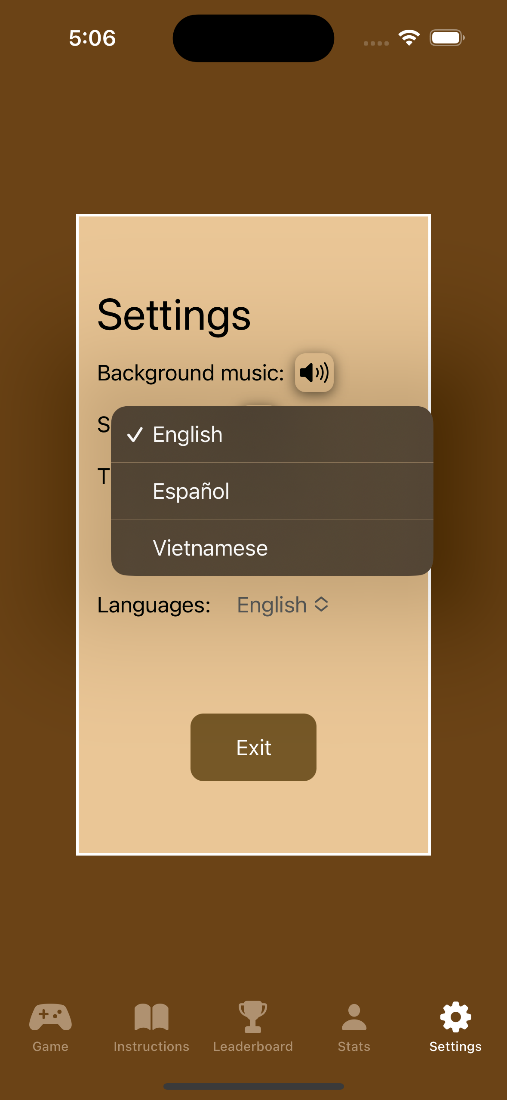
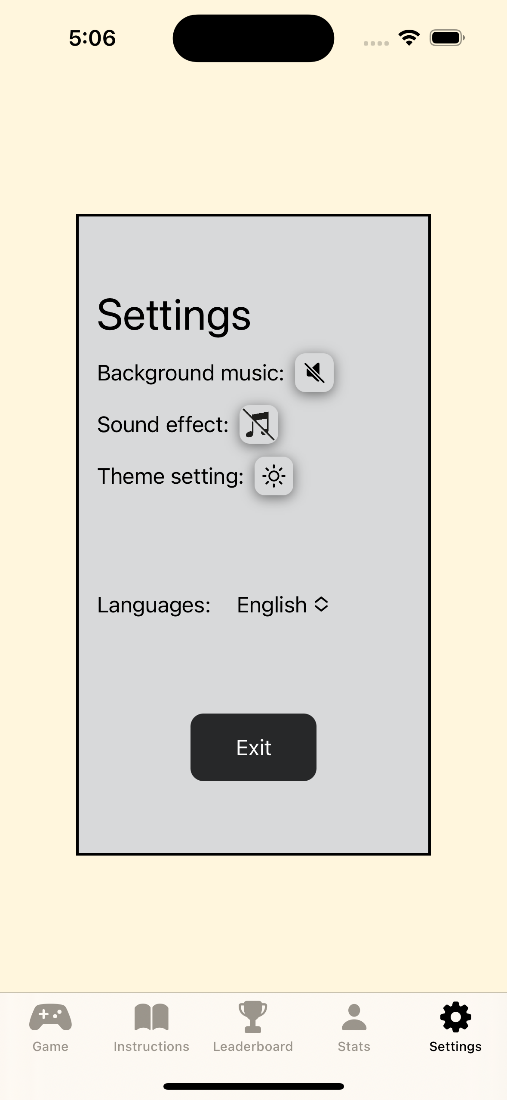


Figure 11: Multi-language

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.

### **Theme setting**

 A screenshot of a phone

Description automatically generated

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.

# **Demonstration Video**

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

# **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**”.

# **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.

# **References**