Semester	2023B
Course	COSC2659 - iOS Development
Course Coordinator	Tom Huynh
Assignment	Assignment 2 - Technical Report
Student	Nguyen Quang Duy - s3877991

"I declare that in submitting all work for this assessment I have read, understood and agreed to the content and expectations of the Assessment Declaration."

TABLE OF CONTENTS

I. Introduction **3**

II. Project description **3**

III. Implemented features **3**

1. Main features *3*

2. Advanced features *7*

3. Live demo *8*

IV. References **9**

V. Appendix **10**

I. Introduction

Blackjack is one of the most popular and challenging card games thanks to its entertainment, relaxation, and especially randomness. Even if there are numerous ways to win, beating the dealer in terms of profit is still not easy to achieve.

The idea of this game is mainly inspired by “Xi Dach", the Vietnamese variant of Blackjack which is mostly played on the holidays, especially Tet. During the game, the players bet with their real money and they will gain all of the bet from the dealer and their opponents if they win, which is however considered illegal as a risk of social corruption. Besides, the people nowadays are constanly under pressure with dozens of working tasks and deadlines, which leads to their increasing needs of relaxation and entertainment between their working hours. As a result, those problems have motivated me to develop an iOS game named “Blackjack” with a view to making users feel relaxed during their working hours along with helping them widen knowledge and improve gaming stategies without losing any money before playing the real game.

II. Project description

The main goal of the “Blackjack” iOS game application is to make the total of cards in the player's hand as close to 21 as possible without exceeding it and beat the dealer. Otherwise, the player’s money in the game will be deducted.

To play the game, firstly the user has to tap the Play button in the MenuView, redirecting to the GameView. Next, the user is prompted to enter a unique username, if the inputted username is empty or already taken in the application, an error message appears and the user is prompted to enter again. Then, at the beginning of the round, both the user and the dealer have two cards in their hands, and they can tap on their cards to view them. If the total of their cards is already 21, the user taps “STAY” button immediately. Otherwise, they are allowed to draw at most 3 cards by tapping “HIT” button as well as make sure that the current total of their cards is between 16 and 21, and they can tap “STAY” button if they don’t want to draw any cards. After that, the dealer automatically performs the same operation as users and then shows their cards to compare with users’ cards. If the users win, they’ll win some money along with the current total score of their cards; otherwise, they’ll lose some money but their highscores still remain. The game ends with the user or the dealer running out of their money.

III. Implemented features

1. *Main features*

a. MenuView

The MenuView contains 4 buttons that contain the navigation links to other views, including a “Play” button to the GameView, a “Leaderboard” button to the list of user rankings, highscores and achievement badges, a “How to play” button to the view that contains gaming instructions and information about difficulties and achievements, and finally a “Settings” button to a control view that allows user to change application theme and game difficulty based on their preferences. And to customize those buttons with the same style, I implemented a custom ViewModifier and then applied it to them.

Figure 1. *Menu view.*

b. GameView

In terms of mechanism, firstly, every time a user plays a new game, a Register modal view appears, prompting them to enter their username ***(Appendix 2)***. Next, there are two sections, one on the top is for the player and one on the bottom is for the dealer. And each section includes a game statistics displaying current amount of money and highscores and a stack of cards, which is always hidden if it belongs to the dealer and can be flipped via a tap if it belongs to the player. At the bottom of the view, there are two action buttons: "HIT" and "STAY". The "HIT" button allows user to draw more cards, and it will be hidden if the current total of the player's cards is 21 or more, or the number of cards is already 5. The "STAY" button allows the user to stop drawing any cards as well as end their turns, and it will be hidden if the current total is smaller than 16 and the number of cards is smaller than 5. Then, when the user ends their turn, the dealer will automatically perform the same operations as the users and then flip their cards, showing the hand status of player and dealer's cards for comparison before displaying a RoundResult modal view. If the player wins, they will gain some money along with their total score of their cards as the bonus score; if they lose, they will lose some money but their highscore still remains; if there is a tie, the player won't lose any money as well as their score. And the amount of money the user gains or loses along with the bonus score during the game varies depending on the difficulty they choose. For example, in "Easy" mode, the amount of money they win doubles the amount of money they lose; in "Medium" mode, the amount of money they win and lose is the same and the bonus score is doubled; in "Hard" mode, the amount of money they win halves the amount of money they lose and the bonus score is tripled. Besides, during the game, when the user reaches a certain score, they will achieve a new badge as an achievement, which is displayed in an Achievement modal view. Finally, when the user or the dealer runs out of money, a GameResult modal view appears, displaying the username along with achievements and highscore as well as ending the current game. ***(Appendix 2)***

To implement that mechanism, I declared state variables and functions in this view to display the current progress of the game such as the current amount of money and highscore, toggle the display of the model views, validate user input, and check if the user wins or loses, improving gaming experience as well as ensuring the accuracy of the game logic.

Figure 2. *Game view, Round result and Achievement view.*

c. LeaderboardView

This view contains a list of users having been signed up to the game, which is loaded from a JSON file and then sorted descending by their highscores by a View Model called "UserViewModel". Each user row includes username, highscores, achievement badge and espesically a button that displays further information in modal view, including join date, number of rounds played and winning rate.

To display that modal view dynamically, I declared and published "selectedUser" and "showUser" variables in UserViewModel, which would be used in UserRow component and UserModal view. When tapping on the button of a specific user row, the object of that user is assigned to "selectedUser" variable and the value of "showUser" is toggled, showing the modal view that displays the information of the selected user. [2]

Figure 3. *Leaderboard and User modal view.*

d. HowToPlayView

This view contains several sections which contains information about the instruction to play the game, difficulties, achievement badges that users can achieve when reaching its milestone score, and so forth. Most of those sections were created by implementing an InformationSection component which accepts a String as title and an array of Strings as paragraphs.

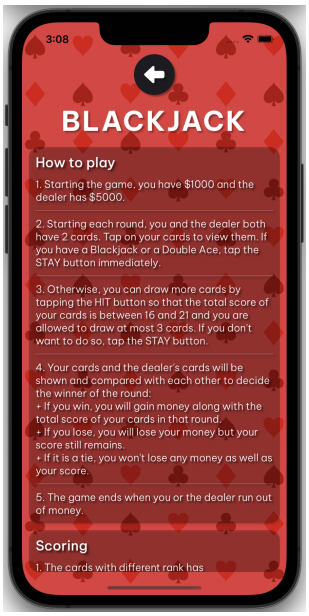


Figure 4. *How to play view.*

e. SettingsView

This view contains some selection controls such as game difficulty and appearance, which allow the user to change game settings as their preference, along with a button to clear the user data. To create those selection controls, I implemented a custom SegmentControl component which accepts a custom Enumerator as an option instead of Picker from SwiftUI framework. And when the options from the selection controls are selected, their value is stored in two AppStorage variables in MenuView. Based on those variables, the game settings will change.

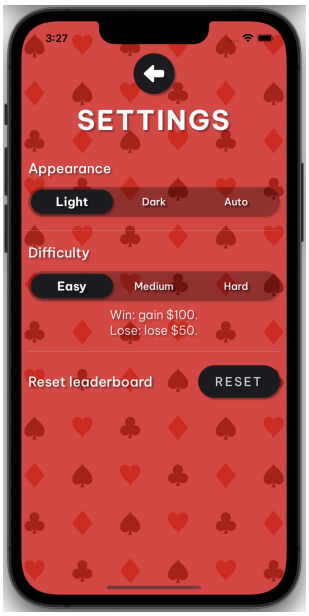


Figure 5. *Settings view.*

f. Others

Appart from the mentioned features, I added sound effects for winning, losing announcements and user actions such as button tap, flip the cards. I also add background music, one for GameView and one for other views. And to make the game play those sounds smoothly without any interuptions, I implemented 4 playing sound functions, each of which plays different types of sounds with different volumes.

Besides, the game is compatible in not only iPhone devices from iPhone 14 onwards but also iPad devices, especially iPad Pro.

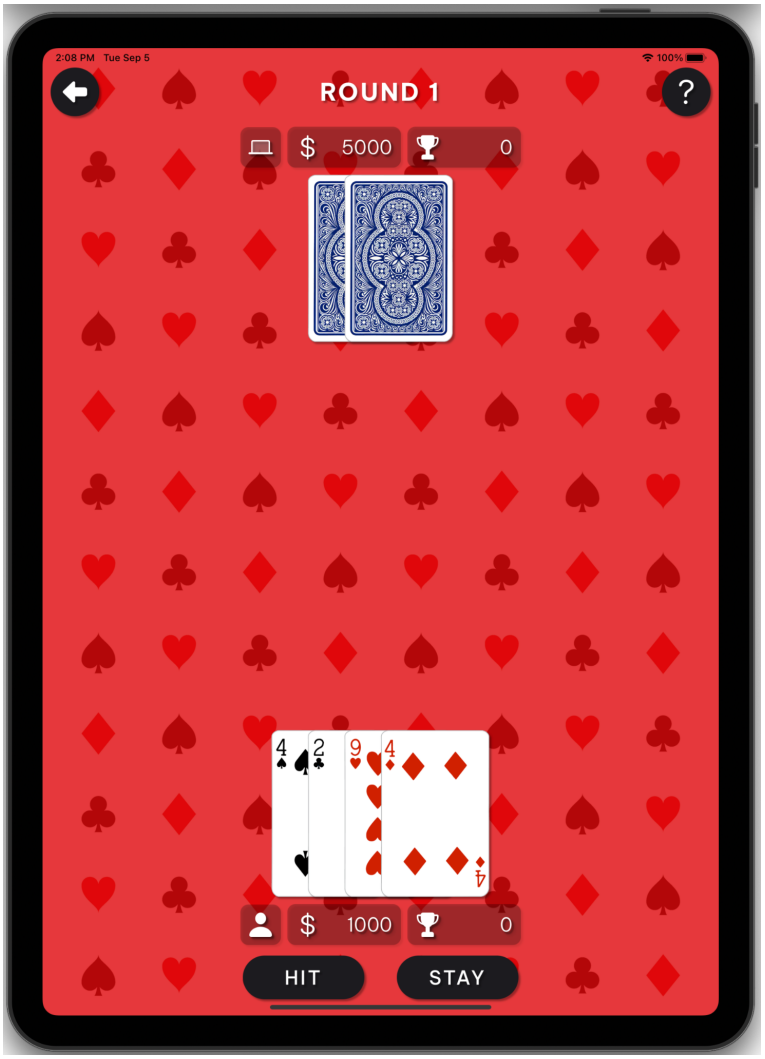


Figure 6. *The interface of the application on iPad devices.*

2. *Advanced features*

a. Save and resume

To allow the user to continue their current progress, I declared a boolean variable named "resume" and assigned it with AppStorage annotation in MenuView. Also, I declared a "scenePhase" variable and assigned it with Environment annotation in both Menu and Game view to check if the user exits the game. [1] Combining those variables, if the user exits the game while playing it, the value of "resume" variable will be true, triggering the display of a ResumeModal view with two options, "CONTINUE" and "NEW GAME". If the user chooses "CONTINUE", they will be redirected to the GameView and their current money, highscore will be kept the same as they exit the game. Otherwise, they have to enter a new username to start a new game, while the current progress is saved and displayed in LeaderBoardView.

Figure 7. *Resume modal view.*

b. Theme setting

To switch the game appearance between light, dark and auto mode, I declare the variable named "appearance" with custom Appearance enumerator and assign it with AppStorage annotation in MenuView, and then use SegmentControl component in SettingsView to change the value of "appearance" variable, and finally add the preferredColorScheme() modifier that accepts the ColorScheme enumerator. Depending on the value of "appearance" variable, the ColorScheme enumerator will change between "light", "dark", and "nil", changing the game appearance as well. [3]

Figure 8. *Menu, Game, and Leaderboard view in dark mode.*

3. *Live demo*

The application's live demo is recorded and then uploaded on YouTube and Google Drive. Please click on the following links below to access and view the demo.

- YouTube
- Google Drive (for backup purposes)

IV. References

[1] Hudson, P. (2022, December 1). *How to detect when your app moves to the background or foreground with scenephase*. Hacking with Swift. https://www.hackingwithswift.com/quick-start/swiftui/how-to-detect-when-your-app-moves-to-the-background-or-foreground-with-scenephase

[2] Kavsoft. (2021, March 27). *Swiftui 2.0 complex hero animations - custom popup's - swipe down to dismiss - swiftui tutorials*. YouTube. https://www.youtube.com/watch?v=MGwPeyB98ng

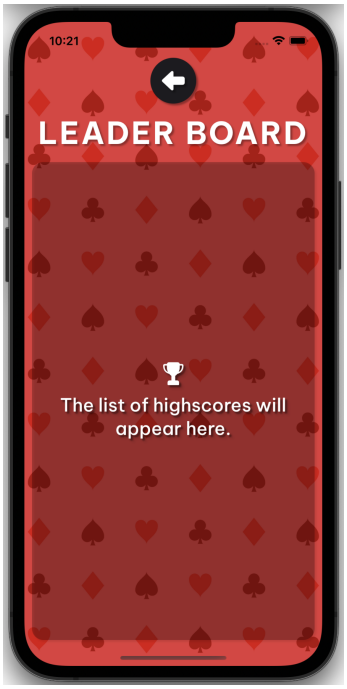
[3] Neovision. (2022, February 5). Ios dark mode preferredcolorscheme works "almost" fine ... .contextmenu still not being changed. Stack Overflow. https://stackoverflow.com/questions/71001858/ios-dark-mode-preferredcolorscheme-works-almost-fine-contextmenu-still-no

V. Appendix

Link to the Github repository of the application: https://github.com/shynkwanchi/Blackjack

Appendix 1. *"Blackjack" application icon.*

Appendix 2. *(From left to right) Registration, Game result modal view.*



Appendix 3. *Empty Leaderboard view.*