COSC2659 - iOS Development

ASSIGNMENT 2: NoiTu Game

Full name: Vo Thanh Thong

Student ID: s3878071

Lecturer: Tom Huynh

Teaching Assistant: Bao Ho

Table of Contents

I. Introduce your iOS game

Welcome to the interactive and engaging word association game! Developed as part of the COSC2659 iOS Development course at RMIT University Vietnam, this app is designed to test and enhance your word association skills while providing hours of fun and entertainment.

To guide you through the gameplay, this app provides a comprehensive set of instructions in the "Rule" tab. It explains the game mechanics, scoring system, and how to navigate through the various levels. Additionally, you will find helpful visual representations and descriptive texts to ensure an intuitive and enjoyable user experience.

In the "Username," "Level," and "Play" tabs, you will find interactive features that allow you to register your username, select different levels of difficulty, and actively participate in the game by entering words and checking their correctness. The app provides real-time feedback, turning the answer text to green when it matches the criteria, indicating your success.

Get ready to embark on an entertaining journey of word discovery and quick thinking! Challenge yourself, improve your word association skills, and strive for high scores. Whether you're a casual player or a word enthusiast, this app guarantees an engaging and educational experience. Enjoy the game and have fun connecting words in creative ways!

II. Why did I choose this topic and motivation

The choice of the topic of word association games stems from the recognition of their widespread appeal and the potential they hold for both entertainment and cognitive development. These games have been enjoyed by people of all ages and have proven to be engaging and intellectually stimulating. The motivation behind choosing this topic lies in the desire to explore the educational value of word association games and the opportunity to create a mobile app that combines fun gameplay with cognitive benefits.

Furthermore, the inspiration for this topic comes from the understanding that language and wordplay are fundamental aspects of human communication and cognitive development. Word association games provide a unique platform to explore the intricacies of language, expand vocabulary, and foster creative thinking. Witnessing the positive impact these games have on individuals, both in terms of linguistic abilities and cognitive skills, has further fueled the motivation to delve deeper into the topic and develop an app that can provide an enjoyable and enriching experience for players.

By choosing the topic of word association games, the goal is to create an app that not only entertains but also serves as a tool for learning and cognitive growth. The inspiration lies in the belief that games can be more than just a form of leisure; they can be a means to engage the mind, expand knowledge, and foster intellectual development.

III. Explain how to play the game. Rules of the game or tips/tricks to play

Tips and Strategies:

To excel in WordConnect, it's essential to think both quickly and creatively. Speed is key as you try to form connections rapidly, but don't rush so much that you sacrifice accuracy. The game rewards you with points for each correct match, so aim for precision. If you find yourself stuck or unsure, don't hesitate to tap the 'Give Up' option to refresh the word and provide you with a new starting point. Additionally, experimenting with synonyms, homophones, and different word forms can be a clever strategy. For instance, if the displayed word is 'jump,' you could respond with 'jumper' or 'jumping' to keep the chain going. Don't be afraid to explore various possibilities.

Challenge Yourself:

WordConnect is not just a game of words; it's a journey of word discovery and quick thinking. Challenge yourself to break free from the obvious choices and push your creativity to the limit. As you progress through the levels, the game will become more challenging, requiring you to think even more critically about word associations. So, get ready to embark on this exciting adventure of linguistic exploration and see just how far your word-linking abilities can take you. WordConnect is not just about scoring points; it's about expanding your vocabulary and enhancing your cognitive skills while having a great time!

IV. What are the main features and advanced features you have done

Main Features:

1. **Menu View (Welcome View):**

Screenshot:

Description: The implementation of the WelcomeView in SwiftUI involves creating a view that displays a welcome screen with buttons for various actions such as playing the game, accessing the instructions, adjusting settings, and viewing the leaderboard.

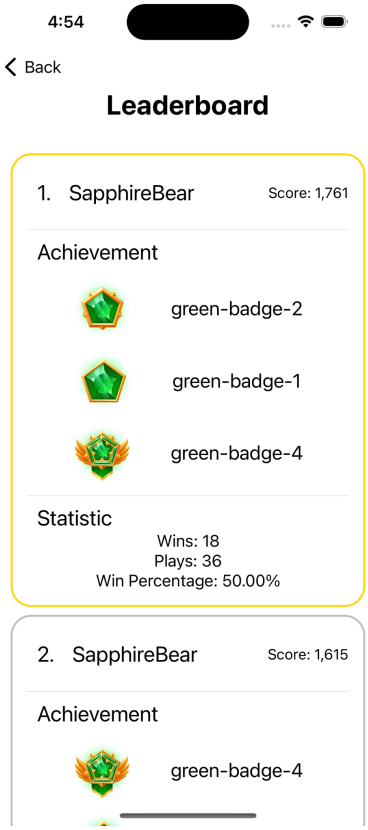
The view uses SwiftUI's @State and @ObservedObject property wrappers to manage the state and observe changes in the game object, which is passed as a parameter to the view. The @State properties are used to track whether the play, how to play, settings, and leaderboard buttons are tapped or not.

The body of the view is composed of a VStack that contains the app logo, followed by the buttons wrapped in a Button view. Each button has an action closure that is triggered when the button is tapped. The action closure updates the respective @State property, and a navigationDestination modifier is applied to navigate to the corresponding view when the @State property changes.

The view also includes additional features such as loading players and game state when the view appears, supporting different locales and dark mode, and managing the accent color.

2. **Leaderboard View:**

Screenshot:



Description: The LeaderboardView contains a VStack that includes a title ("Leaderboard") and a ScrollView. Inside the ScrollView, there is a VStack that iterates over the game's players array using a ForEach loop. For each player, an instance of the LeaderboardItemView is created, passing the player's information such as their position, name, score, achievements, total wins, and total plays.

The LeaderboardItemView is a separate view that represents an individual leaderboard item for a player. It includes the player's position, name, and score in the top row. It also displays the player's achievements in a loop, showing an image and the achievement's name. Finally, it presents the player's statistics, including the number of wins, number of plays, and win percentage.

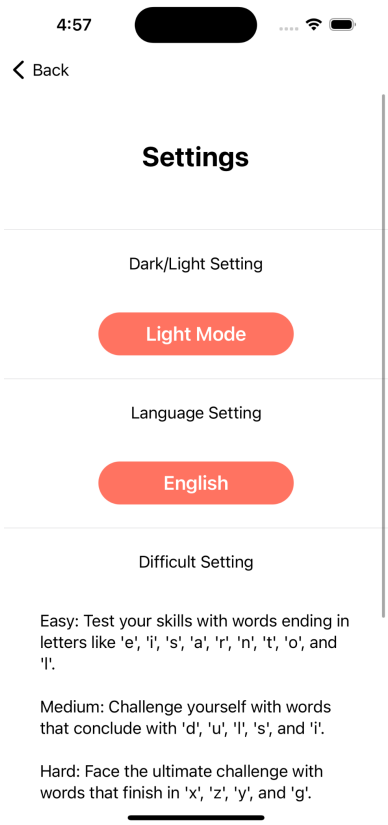
3. **Game View:**

Screenshot:

Description: The body of the GameView is a VStack that contains different views based on the game's state. If there is no currentPlayer in the game, a registration view is displayed. It includes a Text view asking the player to enter their name, a TextField for username input, and a Register Button. When the Register Button is tapped, the currentPlayer is created, added to the game's players array, and a random word from the wordsList is selected as the initial wordGuess. If the currentLevel is not within the range of 1 to 12, a GameLevelView is displayed. This view allows the player to select the level they want to play. Otherwise, the game play view is shown. It includes a Text view greeting the player with their username and the current level, a Text view displaying the score, the wordGuess, a TextField for the player to enter a word, and Check and Give Up buttons. The Check button triggers the checkWord() function in the game logic, updates the score and wordGuess, and handles correct and incorrect word inputs. The Give Up button displays a Game Over pop-up and updates the player's totalPlays.

4. **Game Settings View:**

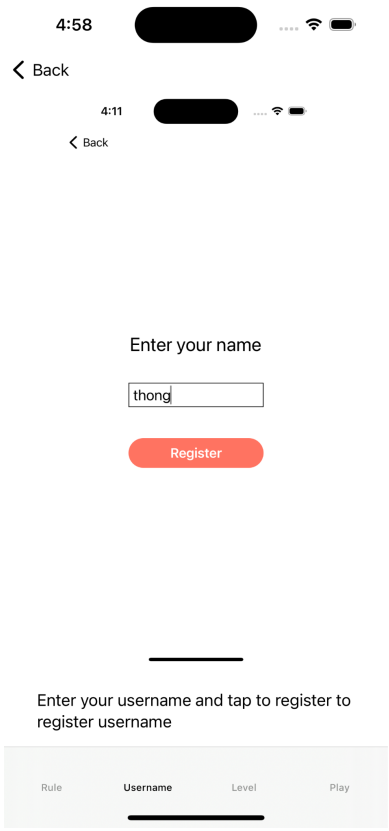
Screenshot: https://youtu.be/JFLHHBxS72E



Description: Dark/Light Setting: This section allows the user to switch between dark and light mode. It displays a button labeled "Dark Mode" or "Light Mode" based on the isDark boolean binding. When tapped, the isDark binding is updated to toggle the mode. Language Setting: This section allows the user to switch between English and Vietnamese languages. It displays a button labeled "English" or "Tiếng Việt" based on the currentLocale binding. When tapped, the currentLocale binding is updated to toggle the language. Difficulty Setting: This section provides information about different difficulty levels and allows the user to select a difficulty level. It includes three buttons: "Easy," "Medium," and "Hard." When a button is tapped, the corresponding action is performed. For example, tapping the "Easy" button updates the lastLetter array and filters the wordsList based on the last letter of each word.

5. **How To Play View:**

Screenshot:



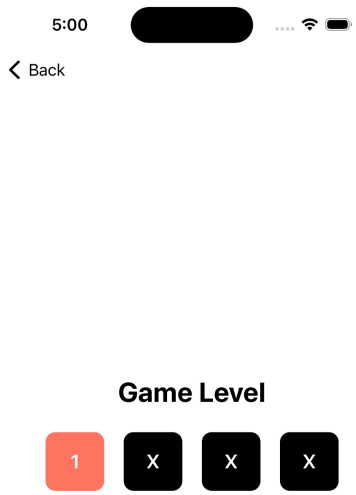
Description: The view includes four tabs: Rule: This tab provides information on how to play the game. It includes a title, followed by a description of the gameplay. The description explains the objective, the task of guessing words based on a specific letter sequence, and the rules for entering guesses. It also mentions the text field for entering guesses and the "Give Up" option. The description emphasizes the importance of speed, accuracy, and creative thinking. Username: This tab displays an image and provides instructions for registering a username. It includes an image showing a username input field and a button to register the username. The accompanying text instructs the user to enter their username and tap the register button. Level: This tab displays an image and provides instructions for choosing a level. It includes an image showing a level selection interface with buttons labeled "1" and other level options. The accompanying text instructs the user to tap on a level button to choose the desired level. Play: This tab displays an image and provides instructions for playing the game. It includes an image showing a word input field and a check button. The accompanying text instructs the user to enter a word and tap the check button to validate their answer. It mentions that the text color will turn green if the answer is correct.

6. **Sound and UI Requirements:**

Advanced Requirements:

1. **Game Progression and Levels:**

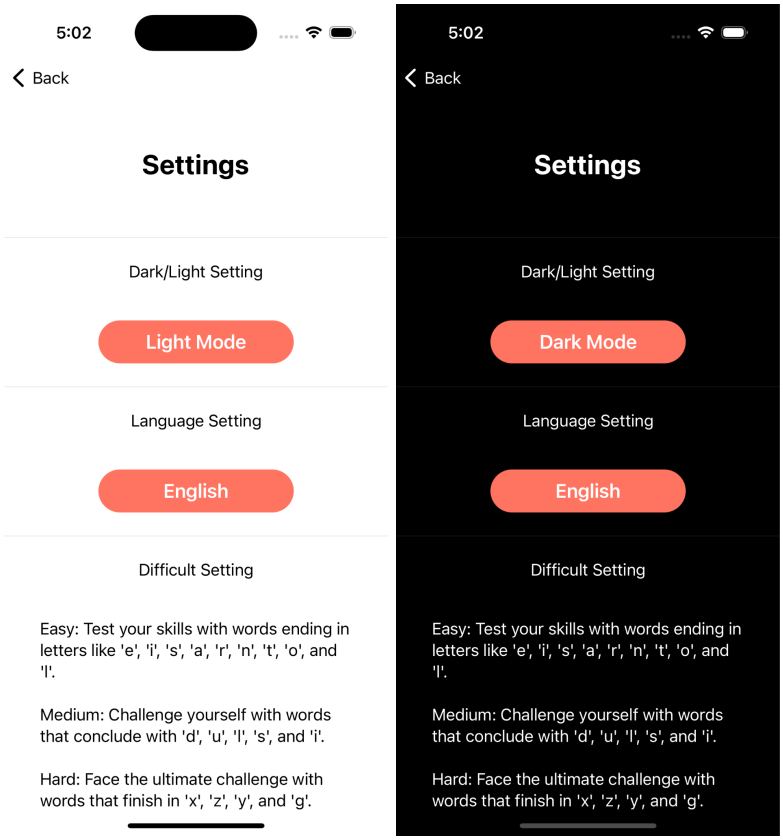
   Screenshot:



Description: Title: A Text view displaying the title "Game Level". It is styled with a large, bold font and some bottom padding. Level Buttons: The level buttons are displayed in a horizontal stack (HStack) with a spacing of 20 points. Each level button is represented by a Button view. The button's label is a Text view displaying the level number. The button's appearance and functionality depend on the game.passedLevel value: If the game.passedLevel is greater than or equal to the level number, the button is enabled and displayed with a coral background color. If the game.passedLevel is less than the level number, the button is disabled and displayed with a black background color. When a level button is tapped, the currentLevel property is updated with the corresponding level number, and the game.targetScore is set accordingly.Each button has a fixed width and height, is rounded with a corner radius of 10, and includes some padding.

2. **Toggle Theme Setting:**

   Light / Dark Theme Screenshot:

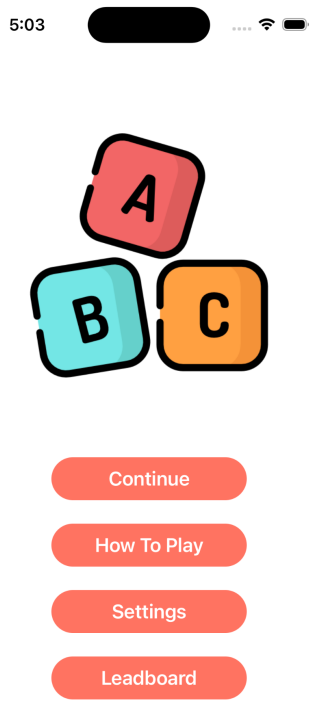


Description: To implement a dark/light theme in SwiftUI, use the .environment(\.colorScheme, isDark ? .dark : .light) modifier. Set the isDark boolean variable to determine the theme, and adjust your view's appearance and colors accordingly.

3. **Save and Resume:**

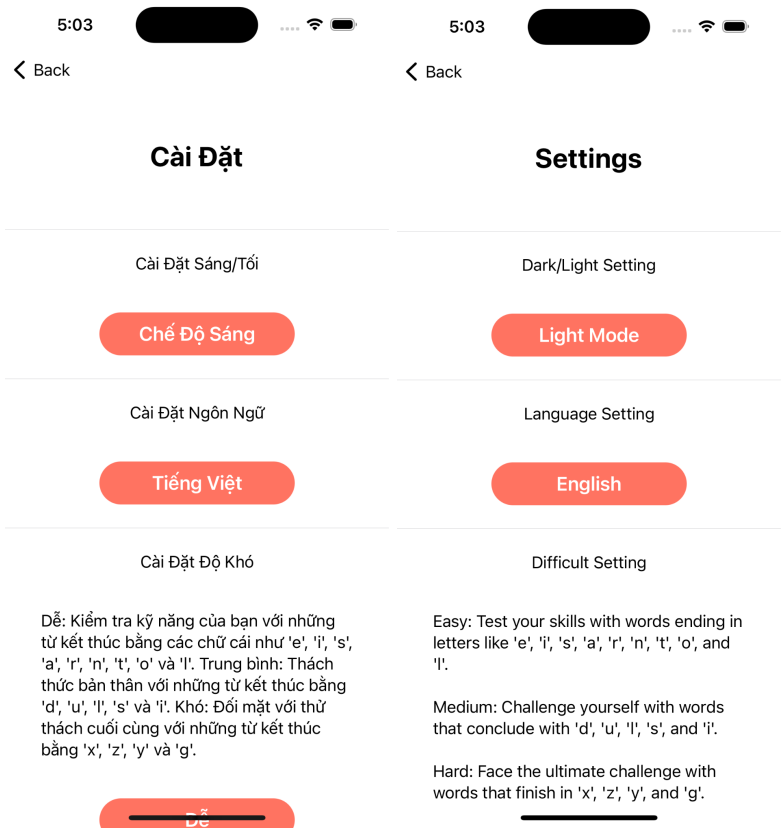
Description: To implement the "save and continue game" feature: Add a saveGameState() method to the Game class that saves the current game state to UserDefaults. Add a loadGameState() method to the Game class that loads the saved game state from UserDefaults. Modify the savePlayers() and loadPlayers() methods to use JSONEncoder and JSONDecoder for encoding and decoding player objects. Call saveGameState() at appropriate points to save the game state. Call loadGameState() when starting or loading a saved game to load the previous state.

Screenshot:



4. **Multiple Language Support:**

   Screenshot (Language Selection in Settings):



Describe: To implement multiple languages in the provided code: Add a currentLocale state variable of type Locale to track the currently selected locale. Use the environment(\.locale, currentLocale) modifier to set the current locale for the view hierarchy. Modify the text strings in the buttons and other UI elements to use localized strings. You can use the Localizable.strings file to provide translations for different languages. Create a Localizable.strings file for each supported language. Add the translations for the text strings used in the UI elements. Use the NSLocalizedString function to retrieve the localized strings in the appropriate places. For example, Text(NSLocalizedString("Play", comment: "")) instead of Text("Play"). Update the currentLocale variable based on the user's language preference or allow the user to manually select the language. To handle localization for dynamically loaded data, you can create localized versions of the data or retrieve localized content from a server based on the current locale.

Link Demo: https://youtu.be/JFLHHBxS72E