import SwiftUI

struct AudioControlsView: View {

@StateObject private var audioManager = AudioManager.shared

var body: some View {

HStack(spacing: 20) {

Button(action: {

audioManager.toggleMusic()

}) {

Image(systemName: audioManager.isMusicEnabled ? "music.note" : "music.note.slash")

.foregroundColor(audioManager.isMusicEnabled ? Color(hex: "00E5FF") : .gray)

}

Button(action: {

audioManager.toggleSound()

}) {

Image(systemName: audioManager.isSoundEnabled ? "speaker.wave.2.fill" : "speaker.slash.fill")

.foregroundColor(audioManager.isSoundEnabled ? Color(hex: "00E5FF") : .gray)

}

}

.font(.title2)

}

}

//

// AudioManager.swift

// Trivia

//

// Created by ahmed.elakpawy on 29.10.24.

//

import AVFoundation

import SwiftUI

class AudioManager: ObservableObject {

static let shared = AudioManager()

var backgroundMusicPlayer: AVAudioPlayer?

var correctSoundPlayer: AVAudioPlayer?

var wrongSoundPlayer: AVAudioPlayer?

var gameOverSoundPlayer: AVAudioPlayer?

@Published var isMusicEnabled = true

@Published var isSoundEnabled = true

init() {

setupAudio()

}

private func setupAudio() {

// Setup background music

if let musicURL = Bundle.main.url(forResource: "background\_music", withExtension: "mp3") {

backgroundMusicPlayer = try? AVAudioPlayer(contentsOf: musicURL)

backgroundMusicPlayer?.numberOfLoops = -1 // Loop indefinitely

backgroundMusicPlayer?.volume = 0.5

}

// Setup sound effects

if let correctURL = Bundle.main.url(forResource: "correct", withExtension: "mp3") {

correctSoundPlayer = try? AVAudioPlayer(contentsOf: correctURL)

correctSoundPlayer?.volume = 0.7

}

if let wrongURL = Bundle.main.url(forResource: "wrong", withExtension: "mp3") {

wrongSoundPlayer = try? AVAudioPlayer(contentsOf: wrongURL)

wrongSoundPlayer?.volume = 0.7

}

if let gameOverURL = Bundle.main.url(forResource: "game\_over", withExtension: "mp3") {

gameOverSoundPlayer = try? AVAudioPlayer(contentsOf: gameOverURL)

gameOverSoundPlayer?.volume = 0.7

}

}

func playBackgroundMusic() {

guard isMusicEnabled else { return }

backgroundMusicPlayer?.play()

}

func stopBackgroundMusic() {

backgroundMusicPlayer?.stop()

backgroundMusicPlayer?.currentTime = 0

}

func playCorrectSound() {

guard isSoundEnabled else { return }

correctSoundPlayer?.play()

}

func playWrongSound() {

guard isSoundEnabled else { return }

wrongSoundPlayer?.play()

}

func playGameOverSound() {

guard isSoundEnabled else { return }

gameOverSoundPlayer?.play()

}

func toggleMusic() {

isMusicEnabled.toggle()

if isMusicEnabled {

playBackgroundMusic()

} else {

stopBackgroundMusic()

}

}

func toggleSound() {

isSoundEnabled.toggle()

}

}

//

// CategoryCard.swift

// Trivia

//

// Created by ahmed.elakpawy on 30.10.24.

//

import SwiftUI

struct CategoryCard: View {

let category: QuizCategory

let isSelected: Bool

var body: some View {

VStack(spacing: 15) {

Image(systemName: category.icon)

.font(.system(size: 30))

Text(category.rawValue)

.font(.headline)

.multilineTextAlignment(.center)

Text(category.description)

.font(.caption)

.foregroundColor(.gray)

.multilineTextAlignment(.center)

}

.padding()

.frame(height: 180)

.frame(maxWidth: .infinity)

.background(

RoundedRectangle(cornerRadius: 15)

.fill(Color(hex: "2E1C4A").opacity(0.6))

)

.overlay(

RoundedRectangle(cornerRadius: 15)

.stroke(category.color,

lineWidth: isSelected ? 2 : 0)

)

.foregroundColor(isSelected ? category.color : .white)

}

}

// Preview provider for testing

struct CategoryCard\_Previews: PreviewProvider {

static var previews: some View {

CategoryCard(

category: .general,

isSelected: true

)

.previewLayout(.sizeThatFits)

.padding()

.background(Color.black)

}

}

import SwiftUI

struct CategorySelectionView: View {

@Environment(\.dismiss) private var dismiss

@State private var selectedTimeLimit: Int? = nil

@State private var showingGameView = false

@State private var selectedCategory: QuizCategory?

// Add specific time options

let timeOptions = [30, 60, 120, nil] // nil represents no time limit

var body: some View {

NavigationView {

ScrollView {

VStack(spacing: 25) {

// Time Selection

VStack(alignment: .leading, spacing: 15) {

Text("Select Time Limit")

.font(.title3)

.bold()

.foregroundColor(.white)

HStack(spacing: 15) {

ForEach(timeOptions, id: \.self) { seconds in

TimeOptionButton(

seconds: seconds,

isSelected: selectedTimeLimit == seconds

) {

selectedTimeLimit = seconds

}

}

}

}

.padding(.horizontal)

// Categories Grid

LazyVGrid(columns: [

GridItem(.flexible()),

GridItem(.flexible())

], spacing: 15) {

ForEach(QuizCategory.allCases) { category in

CategoryCard(category: category, isSelected: selectedCategory == category)

.onTapGesture {

selectedCategory = category

showingGameView = true

}

}

}

.padding(.horizontal)

}

.padding(.vertical)

}

.background(

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

)

.navigationTitle("Choose Category")

.navigationBarTitleDisplayMode(.inline)

.fullScreenCover(isPresented: $showingGameView) {

if let category = selectedCategory {

GameView(

category: category,

timeLimit: selectedTimeLimit

)

}

}

}

}

}

// Time Option Button Component

struct TimeOptionButton: View {

let seconds: Int?

let isSelected: Bool

let action: () -> Void

var body: some View {

Button(action: action) {

Text(seconds.map { "\($0)s" } ?? "Γê₧")

.font(.headline)

.foregroundColor(isSelected ? .black : .white)

.frame(height: 44)

.frame(maxWidth: .infinity)

.background(

RoundedRectangle(cornerRadius: 10)

.fill(isSelected ? Color(hex: "00E5FF") : Color(hex: "2E1C4A").opacity(0.6))

)

}

}

}

// Game View that uses the time limit

struct GameView: View {

let category: QuizCategory

let timeLimit: Int?

@StateObject private var game = GameLogic()

@State private var timeRemaining: Int

@State private var timer: Timer?

@State private var isTimeUp = false

@Environment(\.dismiss) private var dismiss

init(category: QuizCategory, timeLimit: Int?) {

self.category = category

self.timeLimit = timeLimit

\_timeRemaining = State(initialValue: timeLimit ?? 0)

}

var body: some View {

VStack {

if timeLimit != nil {

Text(timeRemaining.formatted())

.font(.largeTitle)

.foregroundColor(timeRemaining < 10 ? .red : .white)

}

ContentView()

}

.onAppear {

startTimer()

game.startNewGame(category: category)

}

.onDisappear {

timer?.invalidate()

}

.alert("Time's Up!", isPresented: $isTimeUp) {

Button("OK", role: .cancel) {

game.endGame()

}

}

}

private func startTimer() {

guard timeLimit != nil else { return }

timer = Timer.scheduledTimer(withTimeInterval: 1, repeats: true) { \_ in

if timeRemaining > 0 {

timeRemaining -= 1

} else {

timer?.invalidate()

isTimeUp = true

}

}

}

}

//

// Color+Extension.swift

// Trivia

//

// Created by ahmed.elakpawy on 30.10.24.

//

import SwiftUI

extension Color {

init(hex: String) {

let hex = hex.trimmingCharacters(in: CharacterSet.alphanumerics.inverted)

var int: UInt64 = 0

Scanner(string: hex).scanHexInt64(&int)

let a, r, g, b: UInt64

switch hex.count {

case 3: // RGB (12-bit)

(a, r, g, b) = (255, (int >> 8) \* 17, (int >> 4 & 0xF) \* 17, (int & 0xF) \* 17)

case 6: // RGB (24-bit)

(a, r, g, b) = (255, int >> 16, int >> 8 & 0xFF, int & 0xFF)

case 8: // ARGB (32-bit)

(a, r, g, b) = (int >> 24, int >> 16 & 0xFF, int >> 8 & 0xFF, int & 0xFF)

default:

(a, r, g, b) = (1, 1, 1, 0)

}

self.init(

.sRGB,

red: Double(r) / 255,

green: Double(g) / 255,

blue: Double(b) / 255,

opacity: Double(a) / 255

)

}

}

import SwiftUI

struct ContentView: View {

@StateObject private var game = GameLogic()

@StateObject private var levelManager = LevelManager.shared

@State private var showingLevelUpAlert = false

@State private var newLevelName = ""

@State private var showingCategorySelection = true

@Environment(\.dismiss) private var dismiss

@State private var showingLeaderboard = false

var body: some View {

ZStack {

// Gradient Background

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

// Content

VStack(spacing: 20) {

if game.isGameOver() {

gameOverView

} else {

gamePlayView

}

}

.padding()

}

.onAppear {

setupLevelUpNotification()

}

.alert("Level Up!", isPresented: $showingLevelUpAlert) {

Button("OK", role: .cancel) { }

} message: {

Text("Congratulations! You've reached \(newLevelName)!")

}

.fullScreenCover(isPresented: $showingLeaderboard) {

LeaderboardView()

}

}

private var gameOverView: some View {

VStack(spacing: 20) {

Spacer()

Text("Game Over!")

.font(.largeTitle)

.bold()

.foregroundColor(.white)

Text("Your score: \(game.score) out of \(game.getTotalQuestions())")

.font(.title2)

.foregroundColor(.white)

if game.score > game.highScore {

Text("New High Score! ≡ƒÄë")

.foregroundColor(Color(hex: "00E5FF"))

.font(.title3)

.padding(.top, 5)

}

Text("High Score: \(game.highScore)")

.font(.subheadline)

.foregroundColor(Color(hex: "B4A5FF"))

.padding(.top, 5)

LevelProgressView()

.padding(.top, 20)

// Navigation Buttons

VStack(spacing: 15) {

Button("Play Again") {

game.restart()

}

.buttonStyle(PrimaryButtonStyle())

Button("Change Category") {

dismiss()

}

.buttonStyle(SecondaryButtonStyle())

Button("View Leaderboard") {

showingLeaderboard = true

}

.buttonStyle(SecondaryButtonStyle())

Button("Share Score") {

shareScore()

}

.buttonStyle(SecondaryButtonStyle())

Button("Back to Home") {

dismiss()

}

.buttonStyle(SecondaryButtonStyle())

}

.padding(.horizontal)

.padding(.top, 20)

Spacer()

}

}

private func setupLevelUpNotification() {

// Your implementation here

}

private func shareScore() {

let scoreText = "I scored \(game.score) out of \(game.getTotalQuestions()) in \(game.currentCategory.rawValue) category on Trivia Master! Current level: \(levelManager.getCurrentLevel().name) ≡ƒÄ«≡ƒºá"

let activityVC = UIActivityViewController(

activityItems: [scoreText],

applicationActivities: nil

)

// Get the window scene and present the share sheet

if let windowScene = UIApplication.shared.connectedScenes.first as? UIWindowScene,

let window = windowScene.windows.first,

let viewController = window.rootViewController {

// For iPad: set the source view for the popover

if let popover = activityVC.popoverPresentationController {

popover.sourceView = window

popover.sourceRect = CGRect(x: window.bounds.midX, y: window.bounds.midY, width: 0, height: 0)

popover.permittedArrowDirections = []

}

viewController.present(activityVC, animated: true)

}

}

private var gamePlayView: some View {

VStack(spacing: 25) {

scoreAndProgressView

.padding(.top, 20)

LevelProgressView()

.padding(.horizontal)

Spacer()

questionView

.padding(.horizontal)

Spacer()

answersView

.padding(.bottom, 30)

}

}

private var scoreAndProgressView: some View {

VStack(spacing: 10) {

HStack {

Text("Score: \(game.score)")

.font(.headline)

.foregroundColor(.white)

Spacer()

AudioControlsView()

Spacer()

Text("Question \(game.questionIndex + 1)/\(game.getTotalQuestions())")

.font(.headline)

.foregroundColor(.white)

}

.padding(.horizontal)

ProgressView(value: Double(game.questionIndex + 1),

total: Double(game.getTotalQuestions()))

.padding(.horizontal)

.tint(Color(hex: "00E5FF"))

}

}

private var questionView: some View {

Text(game.getCurrentQuestion().text)

.font(.title2)

.bold()

.padding(20)

.multilineTextAlignment(.center)

.frame(maxWidth: .infinity)

.foregroundColor(.white)

.background(

RoundedRectangle(cornerRadius: 15)

.fill(Color(hex: "2E1C4A").opacity(0.6))

.overlay(

RoundedRectangle(cornerRadius: 15)

.stroke(Color(hex: "00E5FF").opacity(0.3), lineWidth: 1)

)

)

}

private var answersView: some View {

VStack(spacing: 12) {

ForEach(0..<game.getCurrentQuestion().answers.count, id: \.self) { index in

Button(action: {

game.checkAnswer(index)

}) {

Text(game.getCurrentQuestion().answers[index])

.padding()

.frame(maxWidth: .infinity)

.background(

RoundedRectangle(cornerRadius: 12)

.fill(Color(hex: "00E5FF").opacity(0.1))

.overlay(

RoundedRectangle(cornerRadius: 12)

.stroke(Color(hex: "00E5FF").opacity(0.5), lineWidth: 1)

)

)

.foregroundColor(.white)

}

}

}

.padding(.horizontal)

}

}

struct PrimaryButtonStyle: ButtonStyle {

func makeBody(configuration: Configuration) -> some View {

configuration.label

.font(.headline)

.foregroundColor(.black)

.frame(height: 50)

.frame(maxWidth: .infinity)

.background(Color(hex: "00E5FF"))

.cornerRadius(10)

.opacity(configuration.isPressed ? 0.8 : 1.0)

}

}

struct SecondaryButtonStyle: ButtonStyle {

func makeBody(configuration: Configuration) -> some View {

configuration.label

.font(.headline)

.foregroundColor(Color(hex: "00E5FF"))

.frame(height: 50)

.frame(maxWidth: .infinity)

.background(Color.clear)

.cornerRadius(10)

.overlay(

RoundedRectangle(cornerRadius: 10)

.stroke(Color(hex: "00E5FF"), lineWidth: 1)

)

.opacity(configuration.isPressed ? 0.8 : 1.0)

}

}

struct ContentView\_Previews: PreviewProvider {

static var previews: some View {

ContentView()

}

}

import Foundation

class GameLogic: ObservableObject {

@Published private(set) var score = 0

@Published private(set) var questionIndex = 0

@Published private(set) var showingScore = false

@Published private(set) var currentQuestions: [Question] = []

@Published private(set) var highScore: Int

@Published private(set) var currentCategory: QuizCategory = .general

private let audioManager = AudioManager.shared

private let levelManager = LevelManager.shared

init() {

self.highScore = UserDefaults.standard.integer(forKey: "highScore")

startNewGame()

audioManager.playBackgroundMusic()

}

func startNewGame(category: QuizCategory = .general) {

self.currentCategory = category

let filteredQuestions = filterQuestionsByCategory(category)

currentQuestions = Array(filteredQuestions.shuffled().prefix(10))

score = 0

questionIndex = 0

showingScore = false

audioManager.playBackgroundMusic()

}

private func filterQuestionsByCategory(\_ category: QuizCategory) -> [Question] {

if category == .general {

return QuestionBank.allQuestions

}

return QuestionBank.allQuestions.filter { $0.category == category }

}

func checkAnswer(\_ selectedAnswer: Int) {

if selectedAnswer == currentQuestions[questionIndex].correctAnswer {

score += 1

audioManager.playCorrectSound()

} else {

audioManager.playWrongSound()

}

if questionIndex + 1 < getTotalQuestions() {

questionIndex += 1

} else {

endGame()

}

}

func getCurrentQuestion() -> Question {

return currentQuestions[questionIndex]

}

func getTotalQuestions() -> Int {

return min(10, currentQuestions.count)

}

func isGameOver() -> Bool {

return showingScore

}

func restart() {

startNewGame(category: currentCategory)

}

func endGame() {

showingScore = true

audioManager.stopBackgroundMusic()

audioManager.playGameOverSound()

if score > highScore {

highScore = score

UserDefaults.standard.set(highScore, forKey: "highScore")

}

// Add points to total score - bonus points for non-general categories

let categoryBonus = currentCategory == .general ? 1 : 1.5

let points = Int(Double(score) \* 10 \* categoryBonus)

levelManager.addScore(points)

// Save score to leaderboard

saveScore(score: points, category: currentCategory)

}

// MARK: - Score Management

func saveScore(score: Int, category: QuizCategory) {

let username = UserDefaults.standard.string(forKey: "username") ?? "Unknown Player"

let newScore = PlayerScore(

username: username,

score: score,

category: category,

date: Date()

)

var scores = getStoredScores()

scores.append(newScore)

scores.sort { $0.score > $1.score }

if let encoded = try? JSONEncoder().encode(scores) {

UserDefaults.standard.set(encoded, forKey: "playerScores")

}

}

func getStoredScores() -> [PlayerScore] {

if let data = UserDefaults.standard.data(forKey: "playerScores"),

let decoded = try? JSONDecoder().decode([PlayerScore].self, from: data) {

return decoded

}

return []

}

// Helper Methods

func hasEnoughQuestions(for category: QuizCategory) -> Bool {

let questions = filterQuestionsByCategory(category)

return questions.count >= 10

}

func getQuestionCount(for category: QuizCategory) -> Int {

return filterQuestionsByCategory(category).count

}

func getHighScore(for category: QuizCategory) -> Int {

return UserDefaults.standard.integer(forKey: "highScore\_\(category.rawValue)")

}

private func updateHighScore(for category: QuizCategory) {

let currentCategoryHighScore = getHighScore(for: category)

if score > currentCategoryHighScore {

UserDefaults.standard.set(score, forKey: "highScore\_\(category.rawValue)")

}

if score > highScore {

highScore = score

UserDefaults.standard.set(highScore, forKey: "highScore")

}

}

func getAvailableCategories() -> [QuizCategory] {

return QuizCategory.allCases.filter { hasEnoughQuestions(for: $0) }

}

}

// Add this to your Models file if not already present

struct PlayerScore: Codable, Identifiable {

var id = UUID()

let username: String

let score: Int

let category: QuizCategory

let date: Date

}

import SwiftUI

// First, define the GameGuide structure

struct GameGuide: Identifiable {

let id = UUID()

let title: String

let icon: String

let sections: [String]

}

// QuickTipView component

struct QuickTipView: View {

let icon: String

let title: String

let tip: String

var body: some View {

VStack(alignment: .leading, spacing: 8) {

HStack {

Image(systemName: icon)

.foregroundColor(Color(hex: "00E5FF"))

Text(title)

.font(.headline)

.foregroundColor(.white)

}

Text(tip)

.font(.subheadline)

.foregroundColor(.gray)

.fixedSize(horizontal: false, vertical: true)

}

.padding()

.frame(maxWidth: .infinity, alignment: .leading)

.background(Color.white.opacity(0.1))

.cornerRadius(12)

}

}

// GuideDetailView

struct GuideDetailView: View {

@Environment(\.dismiss) private var dismiss

let guide: GameGuide

var body: some View {

NavigationView {

ZStack {

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

ScrollView {

VStack(alignment: .leading, spacing: 20) {

ForEach(guide.sections, id: \.self) { section in

HStack(alignment: .top, spacing: 12) {

Image(systemName: "checkmark.circle.fill")

.foregroundColor(Color(hex: "00E5FF"))

Text(section)

.foregroundColor(.white)

}

.padding()

.frame(maxWidth: .infinity, alignment: .leading)

.background(Color.white.opacity(0.1))

.cornerRadius(12)

}

}

.padding()

}

}

.navigationTitle(guide.title)

.navigationBarTitleDisplayMode(.inline)

.toolbar {

ToolbarItem(placement: .navigationBarLeading) {

Button("Close") {

dismiss()

}

.foregroundColor(Color(hex: "00E5FF"))

}

}

}

}

}

// Main HelpSupportView

struct HelpSupportView: View {

@State private var selectedGuide: GameGuide?

@State private var showingGuideDetail = false

let guides = [

GameGuide(

title: "Getting Started",

icon: "star.fill",

sections: [

"Choose a category that interests you",

"Select a comfortable time limit",

"Read questions carefully",

"Use process of elimination for tough questions"

]

),

GameGuide(

title: "Scoring System",

icon: "trophy.fill",

sections: [

"Base points: 10 points per correct answer",

"Time bonus: Up to 5 extra points for quick answers",

"Category bonus: 50% extra points for specialized categories",

"Streak bonus: Extra points for consecutive correct answers"

]

),

GameGuide(

title: "Leveling Up",

icon: "arrow.up.circle.fill",

sections: [

"Gain experience points from each game",

"Higher difficulties give more points",

"Complete achievements for bonus points",

"Maintain streaks for faster progression"

]

),

GameGuide(

title: "Pro Tips",

icon: "lightbulb.fill",

sections: [

"Practice different categories to find your strengths",

"Start with shorter time limits to improve speed",

"Use the 50/50 method when unsure",

"Learn from incorrect answers to improve"

]

)

]

var body: some View {

ZStack {

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

ScrollView {

VStack(spacing: 24) {

// Game Guides

VStack(alignment: .leading, spacing: 5) {

Text("Game Guides")

.font(.headline)

.foregroundColor(.gray)

.padding(.horizontal)

ForEach(guides) { guide in

Button {

selectedGuide = guide

showingGuideDetail = true

} label: {

HStack(spacing: 16) {

Image(systemName: guide.icon)

.font(.title2)

.foregroundColor(Color(hex: "00E5FF"))

.frame(width: 30)

Text(guide.title)

.foregroundColor(.white)

Spacer()

Image(systemName: "chevron.right")

.foregroundColor(.gray)

}

.padding()

.background(Color.white.opacity(0.1))

.cornerRadius(12)

}

.padding(.horizontal)

}

}

// Support Links

VStack(alignment: .leading, spacing: 5) {

Text("Support")

.font(.headline)

.foregroundColor(.gray)

.padding(.horizontal)

VStack(spacing: 1) {

Link(destination: URL(string: "mailto:support@triviaapp.com")!) {

HStack {

Image(systemName: "envelope.fill")

Text("Contact Support")

Spacer()

Image(systemName: "arrow.up.right")

}

.foregroundColor(.white)

.padding()

}

Link(destination: URL(string: "https://twitter.com/triviaapp")!) {

HStack {

Image(systemName: "bubble.left.fill")

Text("Follow us on Twitter")

Spacer()

Image(systemName: "arrow.up.right")

}

.foregroundColor(.white)

.padding()

}

}

.background(Color.white.opacity(0.1))

.cornerRadius(12)

.padding(.horizontal)

}

}

.padding(.vertical)

}

}

.navigationTitle("Help & Support")

.navigationBarTitleDisplayMode(.inline)

.sheet(isPresented: $showingGuideDetail) {

if let guide = selectedGuide {

GuideDetailView(guide: guide)

}

}

}

}

//

// HomeView.swift

// Trivia

//

// Created by ahmed.elakpawy on 29.10.24.

//

import SwiftUI

struct HomeView: View {

@StateObject private var levelManager = LevelManager.shared

@State private var showingLevelInfo = false

@State private var selectedLevel: Level?

var body: some View {

ScrollView {

VStack(spacing: 25) {

// Current Progress Section

VStack(spacing: 15) {

HStack {

VStack(alignment: .leading, spacing: 8) {

Text("Current Level")

.font(.subheadline)

.foregroundColor(.gray)

HStack {

Image(systemName: levelManager.getCurrentLevel().image)

.font(.title2)

Text(levelManager.getCurrentLevel().name)

.font(.title3)

.bold()

}

.foregroundColor(.white)

}

Spacer()

Text("Score: \(levelManager.currentTotalScore)")

.font(.headline)

.foregroundColor(Color(hex: "00E5FF"))

}

if let nextLevel = levelManager.getNextLevel() {

ProgressView(value: levelManager.getProgressToNextLevel()) {

HStack {

Text("Next Level: \(nextLevel.name)")

.font(.caption)

.foregroundColor(.gray)

Spacer()

Text("\(nextLevel.minScore - levelManager.currentTotalScore) points to go")

.font(.caption)

.foregroundColor(.gray)

}

}

.tint(Color(hex: "00E5FF"))

}

}

.padding()

.background(

RoundedRectangle(cornerRadius: 15)

.fill(Color(hex: "2E1C4A").opacity(0.6))

)

.padding(.horizontal)

// Level Map

VStack(alignment: .leading, spacing: 15) {

Text("Level Journey")

.font(.title3)

.bold()

.foregroundColor(.white)

.padding(.horizontal)

ScrollView(.horizontal, showsIndicators: false) {

HStack(spacing: 20) {

ForEach(levelManager.levels, id: \.name) { level in

LevelCard(

level: level,

isCurrentLevel: level.name == levelManager.getCurrentLevel().name,

isLocked: level.minScore > levelManager.currentTotalScore

)

.onTapGesture {

selectedLevel = level

showingLevelInfo = true

}

}

}

.padding(.horizontal)

}

}

// Leaderboard Section

VStack(alignment: .leading, spacing: 15) {

Text("Top Players")

.font(.title3)

.bold()

.foregroundColor(.white)

.padding(.horizontal)

ForEach(Array(samplePlayers.prefix(10).enumerated()), id: \.element.id) { index, player in

TopPlayerRow(rank: index + 1, player: player)

}

}

.padding(.horizontal)

}

.padding(.vertical)

}

.background(

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

)

.sheet(isPresented: $showingLevelInfo) {

if let level = selectedLevel {

LevelInfoView(level: level)

}

}

}

}

struct LevelCard: View {

let level: Level

let isCurrentLevel: Bool

let isLocked: Bool

var body: some View {

VStack(spacing: 12) {

Image(systemName: level.image)

.font(.system(size: 30))

.foregroundColor(isLocked ? .gray : Color(hex: "00E5FF"))

Text(level.name)

.font(.caption)

.bold()

.multilineTextAlignment(.center)

Text("\(level.minScore)pts")

.font(.caption2)

.foregroundColor(.gray)

}

.frame(width: 100, height: 120)

.padding()

.background(

RoundedRectangle(cornerRadius: 15)

.fill(isCurrentLevel ? Color(hex: "00E5FF").opacity(0.2) : Color(hex: "2E1C4A").opacity(0.6))

.overlay(

RoundedRectangle(cornerRadius: 15)

.stroke(

isCurrentLevel ? Color(hex: "00E5FF") : Color.clear,

lineWidth: 2

)

)

)

.overlay(

Group {

if isLocked {

Color.black.opacity(0.4)

.cornerRadius(15)

.overlay(

Image(systemName: "lock.fill")

.foregroundColor(.gray)

)

}

}

)

}

}

struct TopPlayerRow: View {

let rank: Int

let player: Player

var body: some View {

HStack(spacing: 15) {

Text("#\(rank)")

.font(.headline)

.foregroundColor(rank <= 3 ? Color(hex: "00E5FF") : .gray)

.frame(width: 40)

Image(systemName: "person.circle.fill")

.font(.title2)

.foregroundColor(Color(hex: "00E5FF"))

VStack(alignment: .leading, spacing: 4) {

Text(player.name)

.font(.subheadline)

.foregroundColor(.white)

Text(player.level)

.font(.caption)

.foregroundColor(.gray)

}

Spacer()

Text("\(player.score)")

.font(.headline)

.foregroundColor(Color(hex: "00E5FF"))

}

.padding()

.background(

RoundedRectangle(cornerRadius: 10)

.fill(Color(hex: "2E1C4A").opacity(0.6))

)

}

}

struct LevelInfoView: View {

let level: Level

@Environment(\.dismiss) private var dismiss

var body: some View {

NavigationView {

VStack(spacing: 20) {

Image(systemName: level.image)

.font(.system(size: 60))

.foregroundColor(Color(hex: "00E5FF"))

Text(level.name)

.font(.title2)

.bold()

VStack(alignment: .leading, spacing: 15) {

InfoRow(title: "Required Score", value: "\(level.minScore) points")

InfoRow(title: "Max Score", value: "\(level.maxScore) points")

InfoRow(title: "Perks", value: "Special achievements and badges")

}

.padding()

.background(

RoundedRectangle(cornerRadius: 15)

.fill(Color(hex: "2E1C4A").opacity(0.6))

)

Spacer()

}

.padding()

.navigationBarTitleDisplayMode(.inline)

.navigationTitle("Level Details")

.toolbar {

Button("Done") {

dismiss()

}

}

}

}

}

struct InfoRow: View {

let title: String

let value: String

var body: some View {

HStack {

Text(title)

.foregroundColor(.gray)

Spacer()

Text(value)

.foregroundColor(.white)

}

}

}

import SwiftUI

struct LeaderboardView: View {

@Environment(\.dismiss) private var dismiss

@State private var timeFrame: TimeFrame = .weekly

@State private var scores: [PlayerScore] = []

@StateObject private var game = GameLogic()

enum TimeFrame: String, CaseIterable {

case daily = "Daily"

case weekly = "Weekly"

case allTime = "All Time"

}

var filteredScores: [PlayerScore] {

let calendar = Calendar.current

let now = Date()

return scores.filter { score in

switch timeFrame {

case .daily:

return calendar.isDateInToday(score.date)

case .weekly:

let weekAgo = calendar.date(byAdding: .day, value: -7, to: now)!

return score.date >= weekAgo

case .allTime:

return true

}

}

}

var body: some View {

NavigationView {

ZStack {

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

VStack(spacing: 20) {

Picker("Time Frame", selection: $timeFrame) {

ForEach(TimeFrame.allCases, id: \.self) { frame in

Text(frame.rawValue).tag(frame)

}

}

.pickerStyle(SegmentedPickerStyle())

.padding()

ScrollView {

if filteredScores.isEmpty {

VStack(spacing: 10) {

Image(systemName: "trophy.fill")

.font(.system(size: 50))

.foregroundColor(Color(hex: "00E5FF"))

.padding()

Text("No Scores Yet")

.font(.title2)

.foregroundColor(.white)

Text("Complete a game to see your score here!")

.font(.body)

.foregroundColor(.gray)

.multilineTextAlignment(.center)

}

.padding(.top, 50)

} else {

VStack(spacing: 10) {

ForEach(Array(filteredScores.enumerated()), id: \.1.id) { index, score in

LeaderboardRow(rank: index + 1, score: score)

}

}

.padding(.horizontal)

}

}

}

}

.navigationTitle("Leaderboard")

.navigationBarTitleDisplayMode(.inline)

.toolbar {

ToolbarItem(placement: .navigationBarLeading) {

Button("Close") {

dismiss()

}

.foregroundColor(Color(hex: "00E5FF"))

}

}

}

.onAppear {

scores = game.getStoredScores()

}

}

}

struct LeaderboardRow: View {

let rank: Int

let score: PlayerScore

var body: some View {

HStack {

Text("#\(rank)")

.font(.headline)

.foregroundColor(rank <= 3 ? Color(hex: "00E5FF") : .gray)

.frame(width: 40)

Image(systemName: "person.circle.fill")

.font(.title2)

.foregroundColor(Color(hex: "00E5FF"))

VStack(alignment: .leading) {

Text(score.username)

.foregroundColor(.white)

Text(score.category.rawValue)

.font(.caption)

.foregroundColor(.gray)

}

Spacer()

Text("\(score.score)")

.font(.headline)

.foregroundColor(Color(hex: "00E5FF"))

}

.padding()

.background(

RoundedRectangle(cornerRadius: 10)

.fill(Color.white.opacity(0.1))

)

.cornerRadius(10)

}

}

//

// LevelManager.swift

// Trivia

//

// Created by ahmed.elakpawy on 29.10.24.

//

import Foundation

struct Level {

let name: String

let minScore: Int

let maxScore: Int

let image: String // System image name for each level

}

class LevelManager: ObservableObject {

static let shared = LevelManager()

let levels: [Level] = [

Level(name: "Baby Brain", minScore: 0, maxScore: 199, image: "brain.head.profile"),

Level(name: "Curious Kid", minScore: 200, maxScore: 399, image: "questionmark.circle"),

Level(name: "Eager Student", minScore: 400, maxScore: 599, image: "book.fill"),

Level(name: "Junior Scholar", minScore: 600, maxScore: 799, image: "graduationcap.fill"),

Level(name: "Knowledge Explorer", minScore: 800, maxScore: 999, image: "map.fill"),

Level(name: "Brainiac Beginner", minScore: 1000, maxScore: 1199, image: "sparkles"),

Level(name: "Trivia Enthusiast", minScore: 1200, maxScore: 1399, image: "star.fill"),

Level(name: "Quiz Whiz", minScore: 1400, maxScore: 1599, image: "bolt.fill"),

Level(name: "Smarty Pants", minScore: 1600, maxScore: 1799, image: "crown.fill"),

Level(name: "Bright Mind", minScore: 1800, maxScore: 1999, image: "lightbulb.fill"),

Level(name: "Trivia Challenger", minScore: 2000, maxScore: 2399, image: "trophy.fill"),

Level(name: "Clever Sage", minScore: 2400, maxScore: 2799, image: "wand.and.stars"),

Level(name: "Mastermind", minScore: 2800, maxScore: 3199, image: "medal.fill"),

Level(name: "Professor", minScore: 3200, maxScore: 3599, image: "books.vertical.fill"),

Level(name: "Brainiac Genius", minScore: 3600, maxScore: 3999, image: "brain"),

Level(name: "Einstein", minScore: 4000, maxScore: Int.max, image: "atom")

]

@Published var currentTotalScore: Int {

didSet {

UserDefaults.standard.set(currentTotalScore, forKey: "totalScore")

}

}

init() {

self.currentTotalScore = UserDefaults.standard.integer(forKey: "totalScore")

}

func getCurrentLevel() -> Level {

return levels.first { level in

currentTotalScore >= level.minScore && currentTotalScore <= level.maxScore

} ?? levels[0]

}

func getNextLevel() -> Level? {

let currentLevel = getCurrentLevel()

return levels.first { $0.minScore > currentLevel.maxScore }

}

func getProgressToNextLevel() -> Double {

let currentLevel = getCurrentLevel()

let scoreInLevel = Double(currentTotalScore - currentLevel.minScore)

let levelRange = Double(currentLevel.maxScore - currentLevel.minScore)

return scoreInLevel / levelRange

}

func addScore(\_ score: Int) {

let oldLevel = getCurrentLevel()

currentTotalScore += score

let newLevel = getCurrentLevel()

if newLevel.name != oldLevel.name {

NotificationCenter.default.post(name: .levelUp, object: nil)

}

}

}

extension Notification.Name {

static let levelUp = Notification.Name("levelUp")

}

import SwiftUI

struct LevelProgressView: View {

@StateObject private var levelManager = LevelManager.shared

var body: some View {

VStack(spacing: 8) {

HStack {

Image(systemName: levelManager.getCurrentLevel().image)

.font(.title2)

Text(levelManager.getCurrentLevel().name)

.font(.headline)

Spacer()

Text("Total Score: \(levelManager.currentTotalScore)")

.font(.subheadline)

}

.foregroundColor(.white)

if let nextLevel = levelManager.getNextLevel() {

ProgressView(value: levelManager.getProgressToNextLevel()) {

HStack {

Text("Next: \(nextLevel.name)")

.font(.caption)

.foregroundColor(.gray)

Spacer()

Text("\(nextLevel.minScore - levelManager.currentTotalScore) points to go")

.font(.caption)

.foregroundColor(.gray)

}

}

.tint(Color(hex: "00E5FF"))

}

}

.padding()

.background(

RoundedRectangle(cornerRadius: 15)

.fill(Color(hex: "2E1C4A").opacity(0.6))

)

.overlay(

RoundedRectangle(cornerRadius: 15)

.stroke(Color(hex: "00E5FF").opacity(0.3), lineWidth: 1)

)

}

}

import SwiftUI

import AuthenticationServices

struct LoginView: View {

@State private var email = ""

@State private var password = ""

@State private var isShowingSignUp = false

@State private var showingAlert = false

@State private var alertMessage = ""

@State private var showingUsernamePrompt = false // Add this line

@AppStorage("isLoggedIn") private var isLoggedIn = false

var body: some View {

ZStack {

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

ScrollView {

// Keep all your existing content exactly the same

VStack(spacing: 25) {

Image(systemName: "brain.head.profile")

.font(.system(size: 80))

.foregroundColor(Color(hex: "00E5FF"))

.padding(.top, 60)

Text("Welcome Back")

.font(.title)

.bold()

.foregroundColor(.white)

VStack(spacing: 15) {

Text("Email")

.foregroundColor(.white)

.frame(maxWidth: .infinity, alignment: .leading)

.padding(.horizontal, 20)

TextField("Enter your email", text: $email)

.textFieldStyle(TriviaTextFieldStyle())

.textContentType(.emailAddress)

.autocapitalization(.none)

Text("Password")

.foregroundColor(.white)

.frame(maxWidth: .infinity, alignment: .leading)

.padding(.horizontal, 20)

SecureField("Enter your password", text: $password)

.textFieldStyle(TriviaTextFieldStyle())

Button("Forgot Password?") {

// Handle forgot password

}

.font(.footnote)

.foregroundColor(Color(hex: "00E5FF"))

.frame(maxWidth: .infinity, alignment: .trailing)

.padding(.trailing, 20)

}

.padding(.horizontal, 20)

Button("Sign In") {

handleLogin()

}

.font(.headline)

.foregroundColor(.black)

.frame(height: 50)

.frame(maxWidth: .infinity)

.background(Color(hex: "00E5FF"))

.cornerRadius(10)

.padding(.horizontal, 20)

VStack(spacing: 20) {

HStack {

Rectangle()

.frame(height: 1)

.foregroundColor(.gray.opacity(0.5))

Text("OR")

.foregroundColor(.gray)

.font(.caption)

Rectangle()

.frame(height: 1)

.foregroundColor(.gray.opacity(0.5))

}

.padding(.horizontal, 20)

// Apple Sign In Button

SignInWithAppleButton { request in

request.requestedScopes = [.email, .fullName]

} onCompletion: { result in

handleAppleSignIn(result)

}

.frame(height: 50)

.frame(maxWidth: .infinity)

.cornerRadius(10)

.padding(.horizontal, 20)

}

HStack {

Text("Don't have an account?")

.foregroundColor(.gray)

Button("Sign Up") {

isShowingSignUp = true

}

.foregroundColor(Color(hex: "00E5FF"))

}

}

.padding(.bottom, 30)

}

}

.alert("Error", isPresented: $showingAlert) {

Button("OK", role: .cancel) { }

} message: {

Text(alertMessage)

}

.sheet(isPresented: $isShowingSignUp) {

SignUpView()

}

.sheet(isPresented: $showingUsernamePrompt) { // Add this sheet

UsernamePromptView(isLoggedIn: $isLoggedIn)

}

}

private func handleLogin() {

if email.isEmpty || password.isEmpty {

alertMessage = "Please fill in all fields"

showingAlert = true

return

}

isLoggedIn = true

}

private func handleAppleSignIn(\_ result: Result<ASAuthorization, Error>) {

switch result {

case .success(let authorization):

if let appleIDCredential = authorization.credential as? ASAuthorizationAppleIDCredential {

// Handle successful Apple sign in

print("Successfully signed in with Apple")

if let email = appleIDCredential.email {

print("User Email: \(email)")

UserDefaults.standard.set(email, forKey: "userEmail")

}

if let fullName = appleIDCredential.fullName {

let firstName = fullName.givenName ?? ""

let lastName = fullName.familyName ?? ""

print("User Name: \(firstName) \(lastName)")

}

// Instead of directly setting isLoggedIn, show username prompt

showingUsernamePrompt = true

}

case .failure(let error):

alertMessage = error.localizedDescription

showingAlert = true

}

}

}

//

// Models.swift

// Trivia

//

// Created by ahmed.elakpawy on 29.10.24.

//

import Foundation

struct Player: Identifiable {

let id = UUID()

let name: String

let score: Int

let level: String

}

// Sample data

let samplePlayers = [

Player(name: "John Doe", score: 2500, level: "Einstein"),

Player(name: "Jane Smith", score: 2300, level: "Brainiac Genius"),

Player(name: "Mike Johnson", score: 2100, level: "Professor"),

Player(name: "Sarah Williams", score: 2000, level: "Mastermind"),

Player(name: "Tom Brown", score: 1900, level: "Clever Sage"),

Player(name: "Lisa Anderson", score: 1800, level: "Trivia Challenger"),

Player(name: "James Wilson", score: 1700, level: "Quiz Whiz"),

Player(name: "Emma Davis", score: 1600, level: "Smarty Pants"),

Player(name: "David Miller", score: 1500, level: "Knowledge Explorer"),

Player(name: "Olivia Taylor", score: 1400, level: "Junior Scholar")

]

import SwiftUI

struct OnboardingView: View {

@State private var currentPage = 0

@State private var username = ""

@State private var showUsernameAlert = false

@AppStorage("hasCompletedOnboarding") private var hasCompletedOnboarding = false

@AppStorage("username") private var storedUsername = ""

private let pages = [

OnboardingPage(

image: "brain.head.profile",

title: "Welcome to Trivia Master",

description: "Challenge yourself with exciting questions across various topics"

),

OnboardingPage(

image: "trophy.fill",

title: "Earn Achievements",

description: "Level up and unlock new ranks as you progress"

),

OnboardingPage(

image: "person.3.fill",

title: "Compete Globally",

description: "Join players worldwide and climb the leaderboard"

)

]

var body: some View {

ZStack {

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

VStack(spacing: 20) {

if currentPage == pages.count {

// Username Input View

VStack(spacing: 30) {

Image(systemName: "person.circle.fill")

.font(.system(size: 100))

.foregroundColor(Color(hex: "00E5FF"))

.padding()

Text("Choose Your Username")

.font(.title)

.bold()

.foregroundColor(.white)

Text("This will be displayed on the leaderboard")

.font(.body)

.foregroundColor(.gray)

.multilineTextAlignment(.center)

.padding(.horizontal)

TextField("Enter username", text: $username)

.textFieldStyle(TriviaTextFieldStyle())

.padding(.horizontal, 40)

.autocapitalization(.none)

Button("Start Playing") {

if username.isEmpty {

showUsernameAlert = true

} else {

storedUsername = username

hasCompletedOnboarding = true

}

}

.font(.headline)

.foregroundColor(.black)

.frame(height: 50)

.frame(maxWidth: .infinity)

.background(Color(hex: "00E5FF"))

.cornerRadius(10)

.padding(.horizontal, 40)

}

} else {

// Regular Onboarding Pages

TabView(selection: $currentPage) {

ForEach(0..<pages.count, id: \.self) { index in

VStack(spacing: 20) {

Image(systemName: pages[index].image)

.font(.system(size: 100))

.foregroundColor(Color(hex: "00E5FF"))

.padding()

Text(pages[index].title)

.font(.title)

.bold()

.foregroundColor(.white)

Text(pages[index].description)

.font(.body)

.multilineTextAlignment(.center)

.foregroundColor(.gray)

.padding(.horizontal)

}

.tag(index)

}

}

.tabViewStyle(PageTabViewStyle(indexDisplayMode: .automatic))

.indexViewStyle(PageIndexViewStyle(backgroundDisplayMode: .always))

Button(currentPage == pages.count - 1 ? "Set Up Profile" : "Next") {

withAnimation {

currentPage += 1

}

}

.font(.headline)

.foregroundColor(.black)

.frame(height: 50)

.frame(maxWidth: .infinity)

.background(Color(hex: "00E5FF"))

.cornerRadius(10)

.padding(.horizontal, 20)

.padding(.bottom, 30)

}

}

}

.alert("Username Required", isPresented: $showUsernameAlert) {

Button("OK", role: .cancel) { }

} message: {

Text("Please enter a username to continue")

}

}

}

struct OnboardingPage {

let image: String

let title: String

let description: String

}

//

// ProfileView.swift

// Trivia

//

// Created by ahmed.elakpawy on 29.10.24.

//

import SwiftUI

struct ProfileView: View {

@AppStorage("isLoggedIn") private var isLoggedIn = true

@StateObject private var levelManager = LevelManager.shared

@State private var showingLogoutAlert = false

@State private var selectedTimeFrame: TimeFrame = .allTime

enum TimeFrame: String, CaseIterable {

case week = "This Week"

case month = "This Month"

case allTime = "All Time"

}

// Sample statistics - In a real app, these would come from your backend

let statistics = [

Statistic(title: "Games Played", value: "127"),

Statistic(title: "Correct Answers", value: "891"),

Statistic(title: "Accuracy", value: "76%"),

Statistic(title: "Best Streak", value: "15")

]

let achievements = [

Achievement(title: "First Victory", description: "Win your first game", isAchieved: true, image: "star.fill"),

Achievement(title: "Perfect Game", description: "Score 100% in a game", isAchieved: true, image: "crown.fill"),

Achievement(title: "Knowledge Seeker", description: "Play 50 games", isAchieved: true, image: "book.fill"),

Achievement(title: "Trivia Master", description: "Reach level Einstein", isAchieved: false, image: "trophy.fill"),

Achievement(title: "Speed Demon", description: "Answer all questions under 30 seconds", isAchieved: false, image: "bolt.fill"),

Achievement(title: "Social Butterfly", description: "Share 10 results", isAchieved: false, image: "person.2.fill")

]

var body: some View {

NavigationView {

ScrollView {

VStack(spacing: 20) {

// Profile Header

VStack {

Image(systemName: "person.circle.fill")

.font(.system(size: 80))

.foregroundColor(Color(hex: "00E5FF"))

Text("John Doe")

.font(.title2)

.bold()

.foregroundColor(.white)

Text(levelManager.getCurrentLevel().name)

.font(.subheadline)

.foregroundColor(Color(hex: "00E5FF"))

}

.padding()

// Statistics Grid

LazyVGrid(columns: [

GridItem(.flexible()),

GridItem(.flexible())

], spacing: 15) {

ForEach(statistics) { stat in

StatisticCard(statistic: stat)

}

}

.padding(.horizontal)

// Achievement Section

VStack(alignment: .leading, spacing: 15) {

Text("Achievements")

.font(.title3)

.bold()

.foregroundColor(.white)

.padding(.horizontal)

ForEach(achievements) { achievement in

AchievementRow(achievement: achievement)

}

}

.padding(.top)

// Settings Buttons

VStack(spacing: 15) {

NavigationLink(destination: SettingsView()) {

SettingsButton(title: "Settings", icon: "gear")

}

NavigationLink(destination: HelpSupportView()) {

SettingsButton(title: "Help & Support", icon: "questionmark.circle")

}

Button {

showingLogoutAlert = true

} label: {

SettingsButton(title: "Logout", icon: "rectangle.portrait.and.arrow.right")

}

}

.padding()

}

}

.background(

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

)

.alert("Logout", isPresented: $showingLogoutAlert) {

Button("Cancel", role: .cancel) { }

Button("Logout", role: .destructive) {

isLoggedIn = false

}

} message: {

Text("Are you sure you want to logout?")

}

.navigationTitle("Profile")

}

}

}

struct Statistic: Identifiable {

let id = UUID()

let title: String

let value: String

}

struct Achievement: Identifiable {

let id = UUID()

let title: String

let description: String

let isAchieved: Bool

let image: String

}

struct StatisticCard: View {

let statistic: Statistic

var body: some View {

VStack(spacing: 10) {

Text(statistic.value)

.font(.title2)

.bold()

.foregroundColor(Color(hex: "00E5FF"))

Text(statistic.title)

.font(.caption)

.foregroundColor(.gray)

}

.frame(maxWidth: .infinity)

.padding()

.background(

RoundedRectangle(cornerRadius: 15)

.fill(Color(hex: "2E1C4A").opacity(0.6))

)

}

}

struct AchievementRow: View {

let achievement: Achievement

var body: some View {

HStack {

Image(systemName: achievement.image)

.font(.title2)

.foregroundColor(achievement.isAchieved ? Color(hex: "00E5FF") : .gray)

.frame(width: 40)

VStack(alignment: .leading, spacing: 4) {

Text(achievement.title)

.font(.subheadline)

.foregroundColor(achievement.isAchieved ? .white : .gray)

Text(achievement.description)

.font(.caption)

.foregroundColor(.gray)

}

Spacer()

if achievement.isAchieved {

Image(systemName: "checkmark.circle.fill")

.foregroundColor(Color(hex: "00E5FF"))

}

}

.padding()

.background(

RoundedRectangle(cornerRadius: 10)

.fill(Color(hex: "2E1C4A").opacity(0.6))

)

.padding(.horizontal)

}

}

struct SettingsButton: View {

let title: String

let icon: String

var body: some View {

HStack {

Image(systemName: icon)

.frame(width: 30)

Text(title)

Spacer()

Image(systemName: "chevron.right")

.font(.caption)

.foregroundColor(.gray)

}

.foregroundColor(.white)

.padding()

.background(

RoundedRectangle(cornerRadius: 10)

.fill(Color(hex: "2E1C4A").opacity(0.6))

)

.padding(.horizontal)

}

}

import Foundation

struct Question: Hashable {

let text: String

let answers: [String]

let correctAnswer: Int

let category: QuizCategory // Added category property

// Implementing Hashable

func hash(into hasher: inout Hasher) {

hasher.combine(text)

hasher.combine(answers)

hasher.combine(correctAnswer)

hasher.combine(category)

}

// Implementing Equatable (required by Hashable)

static func == (lhs: Question, rhs: Question) -> Bool {

return lhs.text == rhs.text &&

lhs.answers == rhs.answers &&

lhs.correctAnswer == rhs.correctAnswer &&

lhs.category == rhs.category

}

}

struct QuestionBank {

static let allQuestions = [

// Geography Questions

Question(

text: "What is the capital of Japan?",

answers: ["Beijing", "Seoul", "Tokyo", "Bangkok"],

correctAnswer: 2,

category: .geography

),

Question(

text: "Which ocean borders California?",

answers: ["Atlantic", "Indian", "Pacific", "Arctic"],

correctAnswer: 2,

category: .geography

),

Question(

text: "What is the capital of Australia?",

answers: ["Sydney", "Melbourne", "Canberra", "Brisbane"],

correctAnswer: 2,

category: .geography

),

Question(

text: "Which language is primarily spoken in Brazil?",

answers: ["Spanish", "Portuguese", "English", "French"],

correctAnswer: 1,

category: .geography

),

Question(

text: "What is the currency of the United Kingdom?",

answers: ["Euro", "Dollar", "Pound", "Franc"],

correctAnswer: 2,

category: .geography

),

Question(

text: "What is the main language spoken in Canada?",

answers: ["English", "French", "Spanish", "German"],

correctAnswer: 0,

category: .geography

),

Question(

text: "Which country is famous for the Great Wall?",

answers: ["India", "China", "Japan", "South Korea"],

correctAnswer: 1,

category: .geography

),

Question(

text: "What is the largest continent by land area?",

answers: ["Africa", "Asia", "Europe", "Australia"],

correctAnswer: 1,

category: .geography

),

Question(

text: "What country is home to the kangaroo?",

answers: ["South Africa", "India", "Australia", "New Zealand"],

correctAnswer: 2,

category: .geography

),

Question(

text: "What is the currency used in Japan?",

answers: ["Dollar", "Yuan", "Won", "Yen"],

correctAnswer: 3,

category: .geography

),

Question(

text: "What is the largest island in the world?",

answers: ["Australia", "Greenland", "New Guinea", "Borneo"],

correctAnswer: 1,

category: .geography

),

Question(

text: "What is the tallest building in the world?",

answers: ["Shanghai Tower", "Burj Khalifa", "One World Trade Center", "Eiffel Tower"],

correctAnswer: 1,

category: .geography

),

// Science Questions

Question(

text: "Which planet is closest to the Sun?",

answers: ["Earth", "Venus", "Mercury", "Mars"],

correctAnswer: 2,

category: .science

),

Question(

text: "What is the largest organ in the human body?",

answers: ["Heart", "Liver", "Brain", "Skin"],

correctAnswer: 3,

category: .science

),

Question(

text: "How many bones are in the adult human body?",

answers: ["206", "205", "208", "210"],

correctAnswer: 0,

category: .science

),

Question(

text: "What is the chemical symbol for water?",

answers: ["H2O", "O2", "CO2", "HO"],

correctAnswer: 0,

category: .science

),

Question(

text: "What is the hardest rock?",

answers: ["Granite", "Marble", "Quartz", "Diamond"],

correctAnswer: 3,

category: .science

),

Question(

text: "What is the smallest planet in our solar system?",

answers: ["Mars", "Mercury", "Venus", "Pluto"],

correctAnswer: 1,

category: .science

),

Question(

text: "How many days are in a leap year?",

answers: ["364", "365", "366", "367"],

correctAnswer: 2,

category: .science

),

Question(

text: "What is the freezing point of water in Fahrenheit?",

answers: ["0┬░F", "32┬░F", "100┬░F", "212┬░F"],

correctAnswer: 1,

category: .science

),

Question(

text: "Who discovered penicillin?",

answers: ["Marie Curie", "Isaac Newton", "Albert Einstein", "Alexander Fleming"],

correctAnswer: 3,

category: .science

),

Question(

text: "What is the speed of light?",

answers: ["299,792 km/s", "150,000 km/s", "300,000 km/s", "350,000 km/s"],

correctAnswer: 0,

category: .science

),

Question(

text: "What is the most abundant gas in Earth's atmosphere?",

answers: ["Oxygen", "Carbon Dioxide", "Nitrogen", "Hydrogen"],

correctAnswer: 2,

category: .science

),

Question(

text: "Which mammal is known to have the most powerful bite?",

answers: ["Lion", "Crocodile", "Jaguar", "Hippopotamus"],

correctAnswer: 3,

category: .science

),

Question(

text: "What is the main ingredient in chocolate?",

answers: ["Cocoa", "Vanilla", "Cinnamon", "Sugar"],

correctAnswer: 0,

category: .science

),

Question(

text: "What color are emeralds?",

answers: ["Blue", "Green", "Red", "Yellow"],

correctAnswer: 1,

category: .science

),

Question(

text: "What is the chemical symbol for potassium?",

answers: ["P", "Po", "K", "Pt"],

correctAnswer: 2,

category: .science

),

Question(

text: "How many planets are in our solar system?",

answers: ["7", "8", "9", "10"],

correctAnswer: 1,

category: .science

),

Question(

text: "What is the main ingredient in sushi?",

answers: ["Rice", "Bread", "Noodles", "Tortilla"],

correctAnswer: 0,

category: .science

),

// History Questions

Question(

text: "What year did the Titanic sink?",

answers: ["1910", "1912", "1914", "1916"],

correctAnswer: 1,

category: .history

),

Question(

text: "Who was the first President of the United States?",

answers: ["Thomas Jefferson", "George Washington", "Abraham Lincoln", "John Adams"],

correctAnswer: 1,

category: .history

),

Question(

text: "What year did man first land on the moon?",

answers: ["1965", "1969", "1970", "1972"],

correctAnswer: 1,

category: .history

),

Question(

text: "Who was the 16th president of the United States?",

answers: ["George Washington", "Abraham Lincoln", "Theodore Roosevelt", "Thomas Jefferson"],

correctAnswer: 1,

category: .history

),

// Technology Questions

Question(

text: "What does 'www' stand for in a web address?",

answers: ["Wide World Web", "Web World Wide", "World Wide Web", "Wide Web World"],

correctAnswer: 2,

category: .technology

),

Question(

text: "Who invented the light bulb?",

answers: ["Thomas Edison", "Nikola Tesla", "Benjamin Franklin", "Alexander Bell"],

correctAnswer: 0,

category: .technology

),

Question(

text: "Which company created the iPhone?",

answers: ["Google", "Samsung", "Apple", "Nokia"],

correctAnswer: 2,

category: .technology

),

Question(

text: "What does 'ATM' stand for?",

answers: ["Automatic Teller Machine", "Automated Transaction Machine", "Automatic Time Machine", "Automated Teller Machine"],

correctAnswer: 3,

category: .technology

),

Question(

text: "Who is known as the father of computers?",

answers: ["Albert Einstein", "Charles Babbage", "Isaac Newton", "Thomas Edison"],

correctAnswer: 1,

category: .technology

),

// Art Questions

Question(

text: "Who painted 'Starry Night'?",

answers: ["Pablo Picasso", "Vincent van Gogh", "Claude Monet", "Salvador Dali"],

correctAnswer: 1,

category: .art

),

Question(

text: "Who painted the Sistine Chapel ceiling?",

answers: ["Raphael", "Michelangelo", "Leonardo da Vinci", "Donatello"],

correctAnswer: 1,

category: .art

),

Question(

text: "Who wrote 'To Kill a Mockingbird'?",

answers: ["Harper Lee", "J.K. Rowling", "Ernest Hemingway", "Mark Twain"],

correctAnswer: 0,

category: .art

),

// Entertainment Questions

Question(

text: "What is the highest-grossing film of all time?",

answers: ["Avatar", "Titanic", "Avengers: Endgame", "The Lion King"],

correctAnswer: 2,

category: .entertainment

),

// Sports Questions

Question(

text: "What is the tallest animal in the world?",

answers: ["Elephant", "Giraffe", "Rhino", "Horse"],

correctAnswer: 1,

category: .sports

),

Question(

text: "Which country hosted the 2016 Summer Olympics?",

answers: ["Brazil", "China", "Japan", "Russia"],

correctAnswer: 0,

category: .sports

)

]

static func getQuestions(for category: QuizCategory) -> [Question] {

if category == .general {

return allQuestions

}

return allQuestions.filter { $0.category == category }

}

// Helper method to get question count for a category

static func getQuestionCount(for category: QuizCategory) -> Int {

return getQuestions(for: category).count

}

}

//

// QuizCategory.swift

// Trivia

//

// Created by ahmed.elakpawy on 30.10.24.

//

import SwiftUI

enum QuizCategory: String, CaseIterable, Identifiable, Codable {

case general = "General Knowledge"

case science = "Science"

case history = "History"

case geography = "Geography"

case entertainment = "Entertainment"

case sports = "Sports"

case technology = "Technology"

case art = "Art & Literature"

var id: String { self.rawValue }

var icon: String {

switch self {

case .general: return "brain.head.profile"

case .science: return "atom"

case .history: return "clock.fill"

case .geography: return "globe.americas.fill"

case .entertainment: return "film.fill"

case .sports: return "sportscourt.fill"

case .technology: return "laptopcomputer"

case .art: return "paintpalette.fill"

}

}

var color: Color {

switch self {

case .general: return Color(hex: "00E5FF")

case .science: return Color(hex: "FF6B6B")

case .history: return Color(hex: "4ECDC4")

case .geography: return Color(hex: "45B7D1")

case .entertainment: return Color(hex: "96CEB4")

case .sports: return Color(hex: "FF9F1C")

case .technology: return Color(hex: "9B5DE5")

case .art: return Color(hex: "F15BB5")

}

}

var description: String {

switch self {

case .general: return "Test your knowledge across various topics"

case .science: return "Explore the wonders of science and nature"

case .history: return "Journey through time and historical events"

case .geography: return "Travel the world through exciting questions"

case .entertainment: return "Movies, TV shows, music, and pop culture"

case .sports: return "Challenge yourself with sports trivia"

case .technology: return "Dive into the world of tech and innovation"

case .art: return "Discover art, literature, and culture"

}

}

}

import SwiftUI

struct SettingsView: View {

@Environment(\.dismiss) private var dismiss

@AppStorage("username") private var username = ""

@AppStorage("difficultyLevel") private var difficultyLevel = "Normal"

@AppStorage("questionTimeLimit") private var questionTimeLimit = 15

@StateObject private var audioManager = AudioManager.shared

@State private var showResetAlert = false

@State private var showUsernameChangeSheet = false

let difficultyLevels = ["Easy", "Normal", "Hard"]

let timeLimits = [10, 15, 20, 30]

var body: some View {

BackgroundView {

ScrollView {

VStack(spacing: 24) {

ProfileSectionView(

username: username,

showUsernameChangeSheet: $showUsernameChangeSheet

)

GameSettingsSectionView(

difficultyLevel: $difficultyLevel,

questionTimeLimit: $questionTimeLimit,

difficultyLevels: difficultyLevels,

timeLimits: timeLimits

)

AudioSettingsSectionView(audioManager: audioManager)

DataManagementSectionView(showResetAlert: $showResetAlert)

}

.padding(.vertical)

}

}

.navigationTitle("Settings")

.navigationBarTitleDisplayMode(.inline)

.alert("Reset Progress", isPresented: $showResetAlert) {

Button("Cancel", role: .cancel) { }

Button("Reset", role: .destructive) {

resetProgress()

}

} message: {

Text("This will reset all your progress, including scores, achievements, and level. This action cannot be undone.")

}

.sheet(isPresented: $showUsernameChangeSheet) {

UsernameChangeView(username: username)

}

}

private func resetProgress() {

UserDefaults.standard.set(0, forKey: "totalScore")

UserDefaults.standard.set(0, forKey: "highScore")

UserDefaults.standard.removeObject(forKey: "playerScores")

difficultyLevel = "Normal"

questionTimeLimit = 15

}

}

// Background View Component

struct BackgroundView<Content: View>: View {

let content: Content

init(@ViewBuilder content: () -> Content) {

self.content = content()

}

var body: some View {

ZStack {

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

content

}

}

}

// Profile Section Component

struct ProfileSectionView: View {

let username: String

@Binding var showUsernameChangeSheet: Bool

var body: some View {

VStack(alignment: .leading, spacing: 5) {

SectionHeader(title: "Profile")

Button(action: { showUsernameChangeSheet = true }) {

HStack {

VStack(alignment: .leading, spacing: 4) {

Text("Username")

.font(.subheadline)

.foregroundColor(.gray)

Text(username)

.foregroundColor(.white)

}

Spacer()

Image(systemName: "pencil")

.foregroundColor(Color(hex: "00E5FF"))

}

.padding()

.background(Color.white.opacity(0.1))

.cornerRadius(12)

}

}

.padding(.horizontal)

}

}

// Game Settings Section Component

struct GameSettingsSectionView: View {

@Binding var difficultyLevel: String

@Binding var questionTimeLimit: Int

let difficultyLevels: [String]

let timeLimits: [Int]

var body: some View {

VStack(alignment: .leading, spacing: 5) {

SectionHeader(title: "Game Settings")

VStack(spacing: 16) {

VStack(alignment: .leading, spacing: 8) {

Text("Difficulty Level")

.font(.subheadline)

.foregroundColor(.gray)

Picker("Difficulty", selection: $difficultyLevel) {

ForEach(difficultyLevels, id: \.self) { level in

Text(level).tag(level)

}

}

.pickerStyle(SegmentedPickerStyle())

}

VStack(alignment: .leading, spacing: 8) {

Text("Default Question Time")

.font(.subheadline)

.foregroundColor(.gray)

Picker("Time Limit", selection: $questionTimeLimit) {

ForEach(timeLimits, id: \.self) { time in

Text("\(time)s").tag(time)

}

}

.pickerStyle(SegmentedPickerStyle())

}

}

.padding()

.background(Color.white.opacity(0.1))

.cornerRadius(12)

}

.padding(.horizontal)

}

}

// Audio Settings Section Component

struct AudioSettingsSectionView: View {

@ObservedObject var audioManager: AudioManager

var body: some View {

VStack(alignment: .leading, spacing: 5) {

SectionHeader(title: "Audio")

VStack(spacing: 1) {

Toggle("Background Music", isOn: .init(

get: { audioManager.isMusicEnabled },

set: { \_ in audioManager.toggleMusic() }

))

.padding()

.tint(Color(hex: "00E5FF"))

Toggle("Sound Effects", isOn: .init(

get: { audioManager.isSoundEnabled },

set: { \_ in audioManager.toggleSound() }

))

.padding()

.tint(Color(hex: "00E5FF"))

}

.background(Color.white.opacity(0.1))

.cornerRadius(12)

.foregroundColor(.white)

}

.padding(.horizontal)

}

}

// Data Management Section Component

struct DataManagementSectionView: View {

@Binding var showResetAlert: Bool

var body: some View {

VStack(alignment: .leading, spacing: 5) {

SectionHeader(title: "Data Management")

Button(action: { showResetAlert = true }) {

HStack {

Image(systemName: "trash")

.foregroundColor(.red)

Text("Reset Game Progress")

.foregroundColor(.red)

Spacer()

}

.padding()

.background(Color.white.opacity(0.1))

.cornerRadius(12)

}

}

.padding(.horizontal)

}

}

// Section Header Component

struct SectionHeader: View {

let title: String

var body: some View {

Text(title)

.font(.headline)

.foregroundColor(.gray)

.padding(.horizontal)

.padding(.bottom, 8)

}

}

// Username Change View

struct UsernameChangeView: View {

@Environment(\.dismiss) private var dismiss

@State private var newUsername: String

@AppStorage("username") private var storedUsername = ""

init(username: String) {

\_newUsername = State(initialValue: username)

}

var body: some View {

NavigationView {

BackgroundView {

VStack(spacing: 20) {

TextField("Enter new username", text: $newUsername)

.textFieldStyle(TriviaTextFieldStyle())

.padding(.horizontal)

Button("Save") {

if !newUsername.isEmpty {

storedUsername = newUsername

dismiss()

}

}

.font(.headline)

.foregroundColor(.black)

.frame(height: 50)

.frame(maxWidth: .infinity)

.background(Color(hex: "00E5FF"))

.cornerRadius(10)

.padding(.horizontal)

}

.padding(.top)

}

.navigationTitle("Change Username")

.navigationBarTitleDisplayMode(.inline)

.toolbar {

ToolbarItem(placement: .navigationBarLeading) {

Button("Cancel") {

dismiss()

}

}

}

}

}

}

//

// SharedStyles.swift

// Trivia

//

// Created by ahmed.elakpawy on 31.10.24.

//

import SwiftUI

struct TriviaTextFieldStyle: TextFieldStyle {

func \_body(configuration: TextField<Self.\_Label>) -> some View {

configuration

.padding()

.background(Color.white.opacity(0.1))

.cornerRadius(8)

.foregroundColor(.white)

.tint(.white)

.accentColor(.white)

}

}

import SwiftUI

struct SignUpView: View {

@State private var email = ""

@State private var password = ""

@State private var confirmPassword = ""

@State private var username = ""

@Environment(\.dismiss) private var dismiss

@State private var showingAlert = false

@State private var alertMessage = ""

@AppStorage("isLoggedIn") private var isLoggedIn = false

@AppStorage("username") private var storedUsername = ""

var body: some View {

ZStack {

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

ScrollView {

VStack(spacing: 25) {

Text("Create Account")

.font(.title)

.bold()

.foregroundColor(.white)

.padding(.top, 50)

VStack(spacing: 15) {

Text("Username")

.foregroundColor(.white)

.frame(maxWidth: .infinity, alignment: .leading)

.padding(.horizontal, 20)

TextField("Enter your username", text: $username)

.textFieldStyle(TriviaTextFieldStyle())

.autocapitalization(.none)

Text("Email")

.foregroundColor(.white)

.frame(maxWidth: .infinity, alignment: .leading)

.padding(.horizontal, 20)

TextField("Enter your email", text: $email)

.textFieldStyle(TriviaTextFieldStyle())

.textContentType(.emailAddress)

.autocapitalization(.none)

Text("Password")

.foregroundColor(.white)

.frame(maxWidth: .infinity, alignment: .leading)

.padding(.horizontal, 20)

SecureField("Enter your password", text: $password)

.textFieldStyle(TriviaTextFieldStyle())

Text("Confirm Password")

.foregroundColor(.white)

.frame(maxWidth: .infinity, alignment: .leading)

.padding(.horizontal, 20)

SecureField("Confirm your password", text: $confirmPassword)

.textFieldStyle(TriviaTextFieldStyle())

}

.padding(.horizontal, 20)

Button("Sign Up") {

handleSignUp()

}

.font(.headline)

.foregroundColor(.black)

.frame(height: 50)

.frame(maxWidth: .infinity)

.background(Color(hex: "00E5FF"))

.cornerRadius(10)

.padding(.horizontal, 20)

Button("Already have an account? Sign In") {

dismiss()

}

.foregroundColor(Color(hex: "00E5FF"))

}

.padding(.bottom, 30)

}

}

.alert("Error", isPresented: $showingAlert) {

Button("OK", role: .cancel) { }

} message: {

Text(alertMessage)

}

}

private func handleSignUp() {

// Validate inputs

if username.isEmpty || email.isEmpty || password.isEmpty || confirmPassword.isEmpty {

alertMessage = "Please fill in all fields"

showingAlert = true

return

}

if password != confirmPassword {

alertMessage = "Passwords don't match"

showingAlert = true

return

}

// Store user data

storedUsername = username

UserDefaults.standard.set(email, forKey: "userEmail")

UserDefaults.standard.set(password, forKey: "userPassword")

// Set login state to true

isLoggedIn = true

// Dismiss the sign up view

dismiss()

}

}

import SwiftUI

@main

struct TriviaApp: App {

@AppStorage("hasCompletedOnboarding") private var hasCompletedOnboarding = false

@AppStorage("isLoggedIn") private var isLoggedIn = false

init() {

// Reset the login state for testing

// Remove these lines later when you have proper authentication

UserDefaults.standard.set(false, forKey: "isLoggedIn")

UserDefaults.standard.set(true, forKey: "hasCompletedOnboarding")

}

var body: some Scene {

WindowGroup {

if !hasCompletedOnboarding {

OnboardingView()

} else if !isLoggedIn {

LoginView()

} else {

TabView {

HomeView()

.tabItem {

Image(systemName: "house.fill")

Text("Home")

}

// Replace ContentView with CategorySelectionView

CategorySelectionView()

.tabItem {

Image(systemName: "gamecontroller.fill")

Text("Play")

}

LeaderboardView()

.tabItem {

Image(systemName: "trophy.fill")

Text("Leaderboard")

}

ProfileView()

.tabItem {

Image(systemName: "person.fill")

Text("Profile")

}

}

}

}

}

}

import SwiftUI

struct UsernamePromptView: View {

@Environment(\.dismiss) private var dismiss

@State private var username = ""

@State private var showAlert = false

@Binding var isLoggedIn: Bool

@AppStorage("username") private var storedUsername = ""

var body: some View {

ZStack {

LinearGradient(

gradient: Gradient(colors: [

Color(hex: "2E1C4A"),

Color(hex: "0F1C4D")

]),

startPoint: .topLeading,

endPoint: .bottomTrailing

)

.ignoresSafeArea()

VStack(spacing: 30) {

Image(systemName: "person.circle.fill")

.font(.system(size: 100))

.foregroundColor(Color(hex: "00E5FF"))

.padding()

Text("Choose Your Username")

.font(.title)

.bold()

.foregroundColor(.white)

Text("This will be displayed on the leaderboard")

.font(.body)

.foregroundColor(.gray)

.multilineTextAlignment(.center)

.padding(.horizontal)

TextField("Enter username", text: $username)

.textFieldStyle(TriviaTextFieldStyle())

.padding(.horizontal, 40)

.autocapitalization(.none)

Button("Continue") {

if username.isEmpty {

showAlert = true

} else {

storedUsername = username

isLoggedIn = true

dismiss()

}

}

.font(.headline)

.foregroundColor(.black)

.frame(height: 50)

.frame(maxWidth: .infinity)

.background(Color(hex: "00E5FF"))

.cornerRadius(10)

.padding(.horizontal, 40)

}

.padding()

}

.alert("Username Required", isPresented: $showAlert) {

Button("OK", role: .cancel) { }

} message: {

Text("Please enter a username to continue")

}

.interactiveDismissDisabled() // Prevents dismissing by swipe

}

}