

Breaksy App - Complete Implementation Guide

> ****CRITICAL****: Brand name is *****"Breaksy"***** - ALWAYS use this exact spelling!

📄 Table of Contents

1. [App Logo Integration](#1-app-logo-integration)
1. [Splash Screen Redesign](#2-splash-screen-redesign)
1. [Welcome Screen Fixes](#3-welcome-screen-fixes)
1. [Login Page Redesign](#4-login-page-redesign)
1. [Profile Card Screen](#5-profile-card-screen-optimization)
1. [Questions Flow - Critical Fixes](#6-questions-flow---critical-fixes)
1. [About You Page](#7-about-you-page---navigation-fix)
1. [Score Analysis Page](#8-score-analysis-page---complex-fixes)
1. [Implementation Strategy](#-implementation-strategy)

1. App Logo Integration

Priority: 🟡 IMPORTANT

Requirements:

- Use app logo from Xcode Project Assets
- Neutral background (no black/white, only logo visible)
- Fallback: Standard Xcode integration if neutral variant not possible

Implementation:

```
``swift
// TASK: Integrate Breaksy logo from Assets

// 1. Search for logo in Assets.xcassets
// Possible names: "BreaksyLogo", "AppIcon", "AppLogo"

// 2. Implementation with transparent background
let logoImageView = UIImageView()
logoImageView.image = UIImage(named: "BreaksyLogo")
logoImageView.contentMode = .scaleAspectFit
logoImageView.backgroundColor = .clear // Neutral background

// 3. Fallback implementation
if logoImageView.image == nil {
    logoImageView.image = UIImage(named: "AppIcon")
}

// 4. Ensure consistency across all screens
// Add to: SplashScreen, WelcomeScreen, LoginPage, ProfileCard, etc.
``
```

Success Criteria:

- ✅ Logo appears with neutral/transparent background
- ✅ Consistent across all screens
- ✅ Proper aspect ratio maintained

2. Splash Screen Redesign

Priority:  CRITICAL +  IMPORTANT

Requirements:

Assets & Design:

- Use Laurel Rating PNG from Assets (will be provided)
- Create ****TWO versions****:
 1. PNG-based (immediate)
 1. Code-based (cleaner, iOS-native, not just emoji)
- PS3-style wave background animation
- Dark color scheme (consistent with app)

UX Improvements:

- Professional design (not childish Apple-style)
- Optimized text/font sizes
- Prominent star rating element

Implementation:

VERSION 1: PNG-Based

```
``swift
```

```
import UIKit
```

```
class SplashScreenViewController: UIViewController {
```

```
    // MARK: - UI Elements
```

```
    private let backgroundGradientLayer = CAGradientLayer()
```

```
    private let logoImageView = UIImageView()
```

```
    private let laurelRatingImageView = UIImageView()
```

```
    private let titleLabel = UILabel()
```

```
    override func viewDidLoad() {
```

```
        super.viewDidLoad()
```

```
        setupBackground()
```

```
        setupLogo()
```

```
        setupRatingStars()
```

```
        setupAnimation()
```

```
    }
```

```
    // PS3-Style Wave Background
```

```
    private func setupBackground() {
```

```
        view.backgroundColor = UIColor(hex: "#1a1a1a")
```

```
        backgroundGradientLayer.colors = [
```

```
            UIColor(hex: "#1a1a1a").cgColor,
```

```
            UIColor(hex: "#2d2d2d").cgColor,
```

```
            UIColor(hex: "#1a1a1a").cgColor
```

```
        ]
```

```
        backgroundGradientLayer.frame = view.bounds
```

```
        view.layer.insertSublayer(backgroundGradientLayer, at: 0)
```

```
        animateWaves()
```

```

}

private func animateWaves() {
    let animation = CAKeyframeAnimation(keyPath: "colors")
    animation.values = [
        UIColor(hex: "#1a1a1a").cgColor, UIColor(hex: "#2d2d2d").cgColor, UIColor(hex:
"#1a1a1a").cgColor],
        UIColor(hex: "#2d2d2d").cgColor, UIColor(hex: "#3a3a3a").cgColor, UIColor(hex:
"#2d2d2d").cgColor],
        UIColor(hex: "#1a1a1a").cgColor, UIColor(hex: "#2d2d2d").cgColor, UIColor(hex:
"#1a1a1a").cgColor]
    ]
    animation.duration = 8.0
    animation.repeatCount = .infinity
    animation.timingFunction = CAMediaTimingFunction(name: .easeInEaseOut)
    backgroundGradientLayer.add(animation, forKey: "waveAnimation")
}

private func setupLogo() {
    logolImageView.image = UIImage(named: "BreaksyLogo")
    logolImageView.contentMode = .scaleAspectFit
    logolImageView.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(logolImageView)

    NSLayoutConstraint.activate([
        logolImageView.centerXAnchor.constraint(equalTo: view.centerXAnchor),
        logolImageView.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor,
constant: 60),
        logolImageView.widthAnchor.constraint(equalToConstant: 120),
        logolImageView.heightAnchor.constraint(equalToConstant: 120)
    ])
}

private func setupRatingStars() {
    // Use provided PNG from Assets
    laurelRatingImageView.image = UIImage(named: "LaurelRatingStars")
    laurelRatingImageView.contentMode = .scaleAspectFit
    laurelRatingImageView.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(laurelRatingImageView)

    NSLayoutConstraint.activate([
        laurelRatingImageView.centerXAnchor.constraint(equalTo: view.centerXAnchor),
        laurelRatingImageView.centerYAnchor.constraint(equalTo: view.centerYAnchor),
        laurelRatingImageView.widthAnchor.constraint(equalToConstant: 200),
        laurelRatingImageView.heightAnchor.constraint(equalToConstant: 80)
    ])
}

private func setupAnimation() {
    // Loading indicator at bottom
    let loadingIndicator = UIActivityIndicatorView(style: .large)
    loadingIndicator.color = .white
    loadingIndicator.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(loadingIndicator)

    NSLayoutConstraint.activate([
        loadingIndicator.centerXAnchor.constraint(equalTo: view.centerXAnchor),
        loadingIndicator.bottomAnchor.constraint(equalTo:
view.safeAreaLayoutGuide.bottomAnchor, constant: -80)
    ])
}

```

```

        loadingIndicator.startAnimating()
    }
}

// Helper Extension
extension UIColor {
    convenience 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 6: (a, r, g, b) = (255, int >> 16, int >> 8 & 0xFF, int & 0xFF)
            default: (a, r, g, b) = (255, 0, 0, 0)
        }
        self.init(red: CGFloat(r) / 255, green: CGFloat(g) / 255, blue: CGFloat(b) / 255, alpha:
CGFloat(a) / 255)
    }
}

```

VERSION 2: Code-Based Stars

```

`swift
class CodeBasedStarsView: UIView {

    override func draw(_ rect: CGRect) {
        guard let context = UIGraphicsGetCurrentContext() else { return }

        // Draw custom star shapes (iOS-native style)
        let starSize: CGFloat = 30
        let spacing: CGFloat = 10
        let totalWidth = (starSize * 5) + (spacing * 4)
        let startX = (rect.width - totalWidth) / 2
        let centerY = rect.height / 2

        for i in 0..<5 {
            let starX = startX + (CGFloat(i) * (starSize + spacing))
            drawStar(in: context, center: CGPoint(x: starX + starSize/2, y: centerY), size: starSize)
        }
    }

    private func drawStar(in context: CGContext, center: CGPoint, size: CGFloat) {
        let path = UIBezierPath()
        let outerRadius = size / 2
        let innerRadius = outerRadius * 0.4
        let angleIncrement = .pi * 2 / 5

        for i in 0..<10 {
            let angle = CGFloat(i) * angleIncrement / 2 - .pi / 2
            let radius = i % 2 == 0 ? outerRadius : innerRadius
            let x = center.x + radius * cos(angle)
            let y = center.y + radius * sin(angle)

            if i == 0 {
                path.move(to: CGPoint(x: x, y: y))
            } else {
                path.addLine(to: CGPoint(x: x, y: y))
            }
        }
    }
}

```

```

    }

    path.close()

    // Gradient fill for modern look
    context.saveGState()
    path.addClip()

    let colors = [UIColor.systemYellow.cgColor, UIColor.systemOrange.cgColor]
    let gradient = CGGradient(colorsSpace: CGColorSpaceCreateDeviceRGB(), colors: colors as
    CFArray, locations: nil)!
    context.drawLinearGradient(gradient, start: CGPoint(x: center.x, y: center.y - outerRadius),
    end: CGPoint(x: center.x, y: center.y + outerRadius), options: [])

    context.restoreGState()
  }
}

```

Success Criteria:

- ☒ Professional, dark-themed design
- ☒ PS3-style wave animation working
- ☒ Both PNG and code-based versions implemented
- ☒ Proper spacing and hierarchy

3. Welcome Screen Fixes

Priority: 🟡 IMPORTANT

Requirements:

- Drastically reduce emoji star sizes
- Remove leaf decorations (keep star positions)
- Cleaner, minimalist look

Implementation:

```

```swift
// TASK: Optimize Welcome Screen

class WelcomeScreenViewController: UIViewController {

 @IBOutlet weak var starLabel1: UILabel!
 @IBOutlet weak var starLabel2: UILabel!
 @IBOutlet weak var starLabel3: UILabel!
 // ... more star labels

 // REMOVE: Leaf decoration views
 // @IBOutlet weak var leafImageView1: UIImageView!
 // @IBOutlet weak var leafImageView2: UIImageView!

 override func viewDidLoad() {
 super.viewDidLoad()
 resizeStars()
 removeDecorations()
 }
}

```



#### #### Navigation:

- Add back arrow (top left)

#### #### Layout:

- “Breaksy” logo at top (near notch)
- Animated welcome text above sign-in buttons:
  - \*\*\*“Welcome to Breaksy. Your journey to freedom starts here. Let’s build your personalized recovery plan.”\*\*
- Animation: Slide-in from right

#### #### Button Styling:

- Google Button: New font + background
- Apple Button: New background
- “Skip for now”: Add confirmation popup → “Do you really want to skip this step?”

#### ### Implementation:

```
``swift
import UIKit
import AuthenticationServices

class LoginPageViewController: UIViewController {

 // MARK: - UI Elements
 private let backButton = UIButton(type: .system)
 private let logoImageView = UIImageView()
 private let welcomeLabel = UILabel()
 private let appleSignInButton = ASAuthorizationAppleIDButton()
 private let googleSignInButton = UIButton(type: .system)
 private let skipButton = UIButton(type: .system)

 override func viewDidLoad() {
 super.viewDidLoad()
 setupUI()
 setupConstraints()
 animateWelcomeText()
 }

 // MARK: - Setup
 private func setupUI() {
 view.backgroundColor = UIColor(hex: "#1a1a1a")

 // Back Button
 backButton.setImage(UIImage(systemName: "chevron.left"), for: .normal)
 backButton.tintColor = .white
 backButton.addTarget(self, action: #selector(backTapped), for: .touchUpInside)

 // Logo
 logoImageView.image = UIImage(named: "BreaksyLogo")
 logoImageView.contentMode = .scaleAspectFit

 // Welcome Text
 welcomeLabel.text = "Welcome to Breaksy. Your journey to freedom starts here. Let's build your personalized recovery plan."
 welcomeLabel.textColor = .white
 welcomeLabel.font = .systemFont(ofSize: 18, weight: .medium)
```

```

welcomeLabel.numberOfLines = 0
welcomeLabel.textAlignment = .center
welcomeLabel.alpha = 0
welcomeLabel.transform = CGAffineTransform(translationX: 50, y: 0) // Start off-screen right

// Apple Sign In Button
appleSignInButton.cornerRadius = 12
appleSignInButton.addTarget(self, action: #selector(appleSignInTapped), for: .touchUpInside)

// Google Sign In Button - REDESIGNED
googleSignInButton.setTitle("Sign in with Google", for: .normal)
googleSignInButton.setTitleColor(.black, for: .normal)
googleSignInButton.titleLabel?.font = .systemFont(ofSize: 17, weight: .semibold)
googleSignInButton.backgroundColor = .white
googleSignInButton.layer.cornerRadius = 12
googleSignInButton.addTarget(self, action: #selector(googleSignInTapped),
for: .touchUpInside)

// Google Icon
let googleIcon = UIImageView(image: UIImage(named: "GoogleIcon"))
googleIcon.contentMode = .scaleAspectFit
googleButton.addSubview(googleIcon)
googleIcon.translatesAutoresizingMaskIntoConstraints = false
NSLayoutConstraint.activate([
 googleIcon.leadingAnchor.constraint(equalTo: googleSignInButton.leadingAnchor,
constant: 16),
 googleIcon.centerYAnchor.constraint(equalTo: googleSignInButton.centerYAnchor),
 googleIcon.widthAnchor.constraint(equalToConstant: 24),
 googleIcon.heightAnchor.constraint(equalToConstant: 24)
])

// Skip Button - REDESIGNED
skipButton.setTitle("Skip for now", for: .normal)
skipButton.setTitleColor(.systemGray, for: .normal)
skipButton.titleLabel?.font = .systemFont(ofSize: 16, weight: .regular)
skipButton.backgroundColor = UIColor(white: 0.2, alpha: 0.5)
skipButton.layer.cornerRadius = 12
skipButton.addTarget(self, action: #selector(skipTapped), for: .touchUpInside)

// Add to view
[backButton, logoImageView, welcomeLabel, appleSignInButton, googleSignInButton,
skipButton].forEach {
 $0.translatesAutoresizingMaskIntoConstraints = false
 view.addSubview($0)
}

private func setupConstraints() {
 NSLayoutConstraint.activate([
 // Back Button
 backButton.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 16),
 backButton.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor,
constant: 8),
 backButton.widthAnchor.constraint(equalToConstant: 44),
 backButton.heightAnchor.constraint(equalToConstant: 44),

 // Logo - NEAR NOTCH
 logoImageView.centerXAnchor.constraint(equalTo: view.centerXAnchor),
 logoImageView.topAnchor.constraint(equalTo: view.safeAreaLayoutGuide.topAnchor,
constant: 20),

```



```

 logImageView.widthAnchor.constraint(equalToConstant: 100),
 logImageView.heightAnchor.constraint(equalToConstant: 100),

 // Welcome Text - ABOVE BUTTONS
 welcomeLabel.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 32),
 welcomeLabel.trailingAnchor.constraint(equalTo: view.trailingAnchor, constant: -32),
 welcomeLabel.bottomAnchor.constraint(equalTo: appleSignInButton.topAnchor, constant:
-40),

 // Apple Button
 appleSignInButton.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 32),
 appleSignInButton.trailingAnchor.constraint(equalTo: view.trailingAnchor, constant: -32),
 appleSignInButton.centerYAnchor.constraint(equalTo: view.centerYAnchor),
 appleSignInButton.heightAnchor.constraint(equalToConstant: 56),

 // Google Button
 googleSignInButton.leadingAnchor.constraint(equalTo: appleSignInButton.leadingAnchor),
 googleSignInButton.trailingAnchor.constraint(equalTo: appleSignInButton.trailingAnchor),
 googleSignInButton.topAnchor.constraint(equalTo: appleSignInButton.bottomAnchor,
constant: 16),
 googleSignInButton.heightAnchor.constraint(equalToConstant: 56),

 // Skip Button
 skipButton.leadingAnchor.constraint(equalTo: appleSignInButton.leadingAnchor),
 skipButton.trailingAnchor.constraint(equalTo: appleSignInButton.trailingAnchor),
 skipButton.topAnchor.constraint(equalTo: googleSignInButton.bottomAnchor, constant:
16),
 skipButton.heightAnchor.constraint(equalToConstant: 56)
])
}

// MARK: - Animations
private func animateWelcomeText() {
 UIView.animate(withDuration: 0.6, delay: 0.3, options: .curveEaseOut) {
 self.welcomeLabel.alpha = 1
 self.welcomeLabel.transform = .identity
 }
}

// MARK: - Actions
@objc private func backTapped() {
 navigationController?.popViewController(animated: true)
}

@objc private func appleSignInTapped() {
 // Apple Sign In logic
}

@objc private func googleSignInTapped() {
 // Google Sign In logic
}

@objc private func skipTapped() {
 showSkipConfirmation()
}

private func showSkipConfirmation() {
 let alert = UIAlertController(
 title: "Skip Sign In",
 message: "Do you really want to skip this step?",

```

```

 preferredStyle: .alert
)

 alert.addAction(UIAlertAction(title: "Yes, Skip", style: .destructive) { [weak self] _ in
 self?.navigateToNextScreen()
 })






 alert.addAction(UIAlertAction(title: "Cancel", style: .cancel))

 present(alert, animated: true)
}

private func navigateToNextScreen() {
 // Navigation logic
 let profileCardVC = ProfileCardViewController()
 navigationController?.pushViewController(profileCardVC, animated: true)
}
}
}

```

### ### Success Criteria:

-  Back button functional
-  Logo positioned near notch
-  Welcome text animates in
-  Buttons redesigned with new styling
-  Skip confirmation popup works

-----

## ## 5. Profile Card Screen Optimization

### Priority:  IMPORTANT

### ### Requirements:

- Center everything upward (too much space to notch currently)
- Remove welcome text (already on login page)
- Move logo + arrow up
- Extend profile card height
- Move "Start my journey" button down
- Focus on profile card (less text)

### ### Implementation:

```

```swift
class ProfileCardViewController: UIViewController {

    @IBOutlet weak var backButton: UIButton!
    @IBOutlet weak var logoImageView: UIImageView!
    @IBOutlet weak var welcomeLabel: UILabel! // TO BE REMOVED
    @IBOutlet weak var profileCardView: UIView!
    @IBOutlet weak var startButton: UIButton!

    // Constraints to modify
    @IBOutlet weak var logoTopConstraint: NSLayoutConstraint!
    @IBOutlet weak var backButtonTopConstraint: NSLayoutConstraint!

```

```

@IBOutlet weak var profileCardHeightConstraint: NSLayoutConstraint!

override func viewDidLoad() {
    super.viewDidLoad()
    optimizeLayout()
}

private func optimizeLayout() {
    // REMOVE: Welcome text (already on login page)
    welcomeLabel.removeFromSuperview()

    // MOVE UP: Logo and Back Button (closer to notch)
    logoTopConstraint.constant = 40 // Was ~120
    backButtonTopConstraint.constant = 40 // Aligned with logo

    // EXTEND: Profile Card
    profileCardHeightConstraint.constant = view.bounds.height * 0.5 // Was 0.35

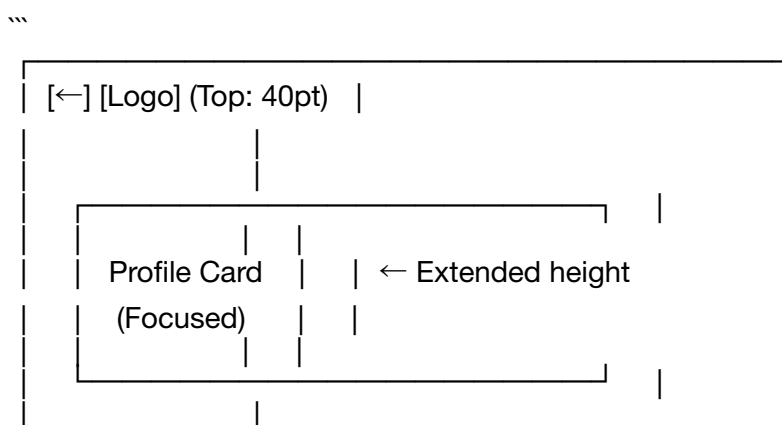
    // REPOSITION: Start Button (sticky at bottom)
    startButton.translatesAutoresizingMaskIntoConstraints = false
    NSLayoutConstraint.activate([
        startButton.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 32),
        startButton.trailingAnchor.constraint(equalTo: view.trailingAnchor, constant: -32),
        startButton.bottomAnchor.constraint(equalTo: view.safeAreaLayoutGuide.bottomAnchor,
constant: -30),
        startButton.heightAnchor.constraint(equalToConstant: 56)
    ])

    // IMPROVE: Profile Card UI
    profileCardView.layer.cornerRadius = 20
    profileCardView.layer.shadowColor = UIColor.black.cgColor
    profileCardView.layer.shadowOpacity = 0.2
    profileCardView.layer.shadowOffset = CGSize(width: 0, height: 4)
    profileCardView.layer.shadowRadius = 12
}

@IBAction func startButtonTapped(_ sender: UIButton) {
    // Navigate to questions
    let questionsVC = QuestionViewController()
    questionsVC.questionIndex = 1
    navigationController?.pushViewController(questionsVC, animated: true)
}
}

```

Visual Hierarchy:



| [Start my journey] | ← Bottom: 30pt

Success Criteria:

- ☒ Welcome text removed
- ☒ Logo/arrow moved up to 40pt from top
- ☒ Profile card extended (50% screen height)
- ☒ Button sticky at bottom
- ☒ Cleaner, more focused appearance

6. Questions Flow - Critical Fixes

Priority:  CRITICAL (Navigation broken)

Requirements:

Navigation (BROKEN):

- Back button must work:
 - Question 2 → Question 1
 - Question 1 → Login Page
- “Skip Test” confirmation: “Do you really want to skip?”
 - Yes: “Yes, I really want to skip this step” (red)
 - No: “Cancel”

Layout:

- Questions positioned higher (not centered)
- Optional: Green checkmark in number circles for answered questions

Content:

- “X (Twitter)” → only “X”

Implementation:

```
``swift
class QuestionViewController: UIViewController {

    var questionIndex: Int = 1
    let totalQuestions: Int = 10

    @IBOutlet weak var backButton: UIButton!
    @IBOutlet weak var questionLabel: UILabel!
    @IBOutlet weak var answerStackView: UIStackView!
    @IBOutlet weak var skipButton: UIButton!
    @IBOutlet weak var questionTopConstraint: NSLayoutConstraint!

    // Question data
    var questions: [Question] = [] // Load from model
```

```

var answeredQuestions: Set<Int> = []

override func viewDidLoad() {
    super.viewDidLoad()
    setupNavigation()
    optimizeLayout()
    loadQuestion()
}

// MARK: - CRITICAL FIX: Navigation
private func setupNavigation() {
    backButton.addTarget(self, action: #selector(backButtonTapped), for: .touchUpInside)
}

@objc private func backButtonTapped() {
    if questionIndex > 1 {
        // Go back to previous question
        navigationController?.popViewController(animated: true)
    } else {
        // Go back to Login Page
        navigationController?.popToViewController(ofClass: LoginPageViewController.self,
animated: true)
    }
}

// MARK: - Layout Optimization
private func optimizeLayout() {
    // Position questions HIGHER on screen
    questionTopConstraint.constant = 80 // Was ~150+
}

// MARK: - Skip Confirmation
@IBAction func skipTestTapped(_ sender: UIButton) {
    let alert = UIAlertController(
        title: "Skip Questions",
        message: "Do you really want to skip this step?",
        preferredStyle: .alert
    )

    alert.addAction(UIAlertAction(
        title: "Yes, I really want to skip this step",
        style: .destructive
    ) { [weak self] _ in
        self?.navigateToNextScreen()
    })

    alert.addAction(UIAlertAction(title: "Cancel", style: .cancel))

    present(alert, animated: true)
}

// MARK: - Answer Selection
@objc private func answerButtonTapped(_ sender: UIButton) {
    let answerIndex = sender.tag

    // Mark as answered
    answeredQuestions.insert(questionIndex)

    // Optional: Show checkmark
    showCheckmark(forQuestion: questionIndex)
}

```

```

        // Save answer and proceed
        saveAnswer(answerIndex)
        goToNextQuestion()
    }

    private func showCheckmark(forQuestion index: Int) {
        // Optional feature: Green checkmark in number badge
        guard let numberBadge = view.viewWithTag(100 + index) else { return }

        let checkmark = UIImageView(image: UIImage(systemName: "checkmark.circle.fill"))
        checkmark.tintColor = .systemGreen
        checkmark.frame = numberBadge.bounds
        checkmark.contentMode = .scaleAspectFit
        numberBadge.addSubview(checkmark)

        UIView.animate(withDuration: 0.3) {
            checkmark.alpha = 1
        }
    }

    // MARK: - Content Fix
    private func loadQuestion() {
        let question = questions[questionIndex - 1]
        questionLabel.text = question.text

        // FIX: "X (Twitter)" → "X"
        let fixedOptions = question.options.map { option in
            option.replacingOccurrences(of: " (Twitter)", with: "")
        }

        // Populate answer buttons
        for (index, option) in fixedOptions.enumerated() {
            let button = UIButton(type: .system)
            button.setTitle(option, for: .normal)
            button.tag = index
            button.addTarget(self, action: #selector(answerButtonTapped), for: .touchUpInside)
            answerStackView.addArrangedSubview(button)
        }
    }

    private func goToNextQuestion() {
        if questionIndex < totalQuestions {
            let nextVC = QuestionViewController()
            nextVC.questionIndex = questionIndex + 1
            nextVC.questions = questions
            navigationController?.pushViewController(nextVC, animated: true)
        } else {
            navigateToNextScreen()
        }
    }

    private func navigateToNextScreen() {
        let aboutYouVC = AboutYouViewController()
        navigationController?.pushViewController(aboutYouVC, animated: true)
    }
}

// MARK: - UINavigationController Extension
extension UINavigationController {

```

```

func popToViewController(ofClass: AnyClass, animated: Bool = true) {
    if let vc = viewControllers.first(where: { $0.isKind(of: ofClass) }) {
        popToViewController(vc, animated: animated)
    }
}
}

```

// MARK: - Question Model

```

struct Question {
    let text: String
    let options: [String]
}

```


Success Criteria:

- ☒ Back button navigates correctly (Q2→Q1, Q1→Login)
- ☒ Skip confirmation popup works
- ☒ Questions positioned higher on screen
- ☒ “X (Twitter)” changed to “X”
- ☒ Optional: Checkmarks show for answered questions

7. About You Page - Navigation Fix

Priority:  CRITICAL

Requirements:

- Back button leads to Question 10 (currently goes to Question 1) 
- Logo + arrow closer to notch
- Age validation: Input >100 → Error message “Number can only be 1 to 100”
 - Display: Red/yellow text below input
 - Auto-correction to 100 already exists

Implementation:

```

``swift
class AboutYouViewController: UIViewController, UITextFieldDelegate {

    @IBOutlet weak var backButton: UIButton!
    @IBOutlet weak var logoImageView: UIImageView!
    @IBOutlet weak var ageTextField: UITextField!
    @IBOutlet weak var errorLabel: UILabel!

    @IBOutlet weak var logoTopConstraint: NSLayoutConstraint!
    @IBOutlet weak var backButtonTopConstraint: NSLayoutConstraint!

    override func viewDidLoad() {
        super.viewDidLoad()
        setupNavigation()
        setupValidation()
        optimizeLayout()
    }
}

```

```

// MARK: - CRITICAL FIX: Back to Question 10
private func setupNavigation() {
    backButton.addTarget(self, action: #selector(goBackToQuestion10), for: .touchUpInside)
}

@objc private func goBackToQuestion10() {
    // Method 1: Find existing Question 10 in nav stack
    if let questionVC = navigationController?.viewControllers.first(where: {
        ($0 as? QuestionViewController)?.questionIndex == 10
    }) {
        navigationController?.popToViewController(questionVC, animated: true)
        return
    }

    // Method 2: Create Question 10 if not in stack
    let question10VC = QuestionViewController()
    question10VC.questionIndex = 10

    // Insert before current VC and pop
    var viewControllers = navigationController?.viewControllers ?? []
    viewControllers.insert(question10VC, at: viewControllers.count - 1)
    navigationController?.setViewControllers(viewControllers, animated: false)
    navigationController?.popViewController(animated: true)
}

// MARK: - Layout Optimization
private func optimizeLayout() {
    // Move logo and back button closer to notch
    logoTopConstraint.constant = 40 // Was ~80+
    backButtonTopConstraint.constant = 40
}

// MARK: - Age Validation
private func setupValidation() {
    ageTextField.delegate = self
    ageTextField.addTarget(self, action: #selector(ageTextChanged), for: .editingChanged)

    // Setup error label
    errorLabel.isHidden = true
    errorLabel.textColor = .systemRed
    errorLabel.font = .systemFont(ofSize: 14, weight: .medium)
}

@objc private func ageTextChanged() {
    guard let text = ageTextField.text, let age = Int(text) else { return }

    validateAge(age)
}

func textFieldDidEndEditing(_ textField: UITextField) {
    guard textField == ageTextField,
        let text = textField.text,
        let age = Int(text) else { return }

    validateAge(age)
}

private func validateAge(_ age: Int) {
    if age > 100 {
        // Auto-correction (already implemented)
    }
}

```



```

ageTextField.text = "100"

// Show error message
errorLabel.text = "Number can only be 1 to 100"
errorLabel.isHidden = false

// Animate in
errorLabel.alpha = 0
UIView.animate(withDuration: 0.3) {
    self.errorLabel.alpha = 1
}

// Auto-hide after 3 seconds
DispatchQueue.main.asyncAfter(deadline: .now() + 3) {
    UIView.animate(withDuration: 0.3) {
        self.errorLabel.alpha = 0
    } completion: { _ in
        self.errorLabel.isHidden = true
    }
}

// Haptic feedback
let generator = UINotificationFeedbackGenerator()
generator.notificationOccurred(.error)
} else if age < 1 {
    ageTextField.text = "1"
    showError("Number can only be 1 to 100")
} else {
    errorLabel.isHidden = true
}
}

private func showError(_ message: String) {
    errorLabel.text = message
    errorLabel.isHidden = false
    errorLabel.alpha = 0

    UIView.animate(withDuration: 0.3) {
        self.errorLabel.alpha = 1
    }

    DispatchQueue.main.asyncAfter(deadline: .now() + 3) {
        UIView.animate(withDuration: 0.3) {
            self.errorLabel.alpha = 0
        } completion: { _ in
            self.errorLabel.isHidden = true
        }
    }
}
}
}

```

Storyboard Setup:

```

```xml
<!-- Add error label below age text field -->
<label opaque="NO" userInteractionEnabled="NO" contentMode="left"
horizontalHuggingPriority="251" verticalHuggingPriority="251" text="" textAlignment="natural"
lineBreakMode="tailTruncation" baselineAdjustment="alignBasedlines"
adjustsFontSizeToFit="NO" translatesAutoresizingMaskIntoConstraints="NO" id="errorLabel">

```

```

<constraints>
 <constraint firstAttribute="height" constant="20" id="..." />
</constraints>
<fontDescription key="fontDescription" type="system" weight="medium" pointSize="14"/>
<color key="textColor" systemColor="systemRedColor"/>
<nil key="highlightedColor"/>
</label>
```

```

Success Criteria:

- ☒ Back button leads to Question 10
- ☒ Logo/arrow positioned at 40pt from top
- ☒ Age validation shows error for >100
- ☒ Auto-correction to 100 works
- ☒ Error message auto-hides after 3s

8. Score Analysis Page - Complex Fixes

Priority:  CRITICAL (Wrong calculations)

Requirements:

Layout:

- Logo + arrow closer to notch (consistent)
- Continue button ****sticky**** at bottom

Chart Logic (BROKEN):

- ☒ "Your Score" must NEVER be smaller than "Average"
- ☒ Percentage calculation not connected
- ☒ Fix: If user 22% worse → "Your Score" bar = Average + 22%

Content:

- Key Insights: Shorten text
- Recommended Next Steps: "Get yourself in touch with Breaksy" → Feature marketing
 - Example: "Do a daily check-in to stay on your porn-free streak"
- Important Disclaimer: Compress field (smaller but readable)

Implementation:

Part 1: Chart Calculation Fix

```

```swift
import UIKit
import Charts // If using Charts library

class ScoreAnalysisViewController: UIViewController {

 @IBOutlet weak var backButton: UIButton!
 @IBOutlet weak var logoImageView: UIImageView!

```

```

@IBOutlet weak var chartContainerView: UIView!
@IBOutlet weak var yourScoreBar: UIView!
@IBOutlet weak var averageScoreBar: UIView!
@IBOutlet weak var percentageLabel: UILabel!
@IBOutlet weak var keyInsightsStackView: UIStackView!
@IBOutlet weak var recommendedStepsStackView: UIStackView!
@IBOutlet weak var disclaimerView: UIView!
@IBOutlet weak var continueButton: UIButton!
@IBOutlet weak var scrollView: UIScrollView!

@IBOutlet weak var logoTopConstraint: NSLayoutConstraint!
@IBOutlet weak var yourScoreBarHeightConstraint: NSLayoutConstraint!
@IBOutlet weak var averageScoreBarHeightConstraint: NSLayoutConstraint!

// Score data
var userScore: Int = 0
var averageScore: Int = 0

override func viewDidLoad() {
 super.viewDidLoad()
 optimizeLayout()
 calculateAndDisplayScore()
 setupContent()
 setupStickyButton()
}

// MARK: - Layout Optimization
private func optimizeLayout() {
 logoTopConstraint.constant = 40
 // Back button aligned with logo (same constraint)
}

// MARK: - CRITICAL FIX: Chart Calculation
private func calculateAndDisplayScore() {
 let scoreData = ScoreData(yourScore: userScore, averageScore: averageScore)
 updateChart(with: scoreData)
}

private func updateChart(with data: ScoreData) {
 let heights = data.displayBarHeight

 // Update bar heights
 yourScoreBarHeightConstraint.constant = heights.yours
 averageScoreBarHeightConstraint.constant = heights.average

 // Update percentage label - PROPERLY CONNECTED
 let percentage = data.percentageDifference

 if percentage > 0 {
 percentageLabel.text = String(format: "%.0f%% higher dependence", percentage)
 percentageLabel.textColor = .systemRed
 percentageLabel.isHidden = false
 } else {
 // User is at or below average
 percentageLabel.isHidden = true
 }

 // Animate bars
 UIView.animate(withDuration: 0.8, delay: 0.2, options: .curveEaseOut) {
 self.view.layoutIfNeeded()
 }
}

```

```

 }
}

// MARK: - Sticky Continue Button
private func setupStickyButton() {
 // Remove from scroll view and add to main view
 continueButton.removeFromSuperview()
 view.addSubview(continueButton)

 continueButton.translatesAutoresizingMaskIntoConstraints = false
 NSLayoutConstraint.activate([
 continueButton.leadingAnchor.constraint(equalTo: view.leadingAnchor, constant: 20),
 continueButton.trailingAnchor.constraint(equalTo: view.trailingAnchor, constant: -20),
 continueButton.bottomAnchor.constraint(equalTo:
view.safeAreaLayoutGuide.bottomAnchor, constant: -20),
 continueButton.heightAnchor.constraint(equalToConstant: 56)
])

 // Adjust scroll view bottom inset
 scrollView.contentInset.bottom = 80
}

// MARK: - Content Setup
private func setupContent() {
 setupKeyInsights()
 setupRecommendedSteps()
 setupDisclaimer()
}

private func setupKeyInsights() {
 // SHORTENED text
 let insights = [
 "High dependence detected",
 "Social isolation risk present",
 "Impulse control needs improvement"
]

 keyInsightsStackView.arrangedSubviews.forEach { $0.removeFromSuperview() }

 for insight in insights {
 let label = UILabel()
 label.text = "• \(insight)"
 label.font = .systemFont(ofSize: 15)
 label.textColor = .white
 label.numberOfLines = 1
 keyInsightsStackView.addArrangedSubview(label)
 }
}

private func setupRecommendedSteps() {
 // BREAKSY FEATURE MARKETING
 let steps = [
 RecommendedStep(
 icon: "checkmark.circle.fill",
 title: "Daily Check-ins",
 description: "Stay accountable with Breaksy's daily progress tracking"
),
 RecommendedStep(
 icon: "bolt.shield.fill",
 title: "Emergency Support",

```

```

 description: "Access instant help when urges strike"
),
 RecommendedStep(
 icon: "person.3.fill",
 title: "Community Support",
 description: "Join others on the same recovery journey"
),
 RecommendedStep(
 icon: "chart.line.uptrend.xyaxis",
 title: "Progress Tracking",
 description: "Visualize your streak and celebrate milestones"
)
]

```

```
recommendedStepsStackView.arrangedSubviews.forEach { $0.removeFromSuperview() }
```

```

// Header
let headerLabel = UILabel()
headerLabel.text = "Recommended: Go with Breaksy"
headerLabel.font = .systemFont(ofSize: 20, weight: .bold)
headerLabel.textColor = .white
recommendedStepsStackView.addArrangedSubview(headerLabel)

```

```

// Add steps
for step in steps {
 let stepView = createStepView(step)
 recommendedStepsStackView.addArrangedSubview(stepView)
}
}

```

```

private func createStepView(_ step: RecommendedStep) -> UIView {
 let containerView = UIView()
 containerView.translatesAutoresizingMaskIntoConstraints = false

```

```

 let iconImageView = UIImageView(image: UIImage(systemName: step.icon))
 iconImageView.tintColor = .systemBlue
 iconImageView.contentMode = .scaleAspectFit
 iconImageView.translatesAutoresizingMaskIntoConstraints = false

```

```

 let titleLabel = UILabel()
 titleLabel.text = step.title
 titleLabel.font = .systemFont(ofSize: 16, weight: .semibold)
 titleLabel.textColor = .white
 titleLabel.translatesAutoresizingMaskIntoConstraints = false

```

```

 let descriptionLabel = UILabel()
 descriptionLabel.text = step.description
 descriptionLabel.font = .systemFont(ofSize: 14)
 descriptionLabel.textColor = .systemGray
 descriptionLabel.numberOfLines = 2
 descriptionLabel.translatesAutoresizingMaskIntoConstraints = false

```

```

 containerView.addSubview(iconImageView)
 containerView.addSubview(titleLabel)
 containerView.addSubview(descriptionLabel)

```

```

 NSLayoutConstraint.activate([
 iconImageView.leadingAnchor.constraint(equalTo: containerView.leadingAnchor, constant:
8),
 iconImageView.topAnchor.constraint(equalTo: containerView.topAnchor, constant: 8),

```

```

 iconImageView.widthAnchor.constraint(equalToConstant: 28),
 iconImageView.heightAnchor.constraint(equalToConstant: 28),

 titleLabel.leadingAnchor.constraint(equalTo: iconImageView.trailingAnchor, constant: 12),
 titleLabel.topAnchor.constraint(equalTo: containerView.topAnchor, constant: 8),
 titleLabel.trailingAnchor.constraint(equalTo: containerView.trailingAnchor, constant: -8),

 descriptionLabel.leadingAnchor.constraint(equalTo: titleLabel.leadingAnchor),
 descriptionLabel.topAnchor.constraint(equalTo: titleLabel.bottomAnchor, constant: 4),
 descriptionLabel.trailingAnchor.constraint(equalTo: titleLabel.trailingAnchor),
 descriptionLabel.bottomAnchor.constraint(equalTo: containerView.bottomAnchor,
constant: -8)
])

 containerView.heightAnchor.constraint(greaterThanOrEqualToConstant: 60).isActive = true

 return containerView
}

private func setupDisclaimer() {
 // COMPRESS disclaimer
 guard let disclaimerLabel = disclaimerView.subviews.first as? UILabel else { return }

 disclaimerLabel.font = .systemFont(ofSize: 12) // Was 14+
 disclaimerLabel.numberOfLines = 3
 disclaimerLabel.lineBreakMode = .byTruncatingTail

 // Reduce container height
 disclaimerView.constraints.forEach { constraint in
 if constraint.firstAttribute == .height {
 constraint.constant = 80 // Was ~120
 }
 }
}

// MARK: - Score Data Model
struct ScoreData {
 let yourScore: Int
 let averageScore: Int

 var percentageDifference: Double {
 guard averageScore > 0 else { return 0 }
 return Double(yourScore - averageScore) / Double(averageScore) * 100
 }

 var displayBarHeight: (yours: CGFloat, average: CGFloat) {
 let baseHeight: CGFloat = 100 // Average bar base height
 let averageBarHeight = baseHeight

 // CRITICAL: Your Score bar MUST ALWAYS be >= Average
 let yourBarHeight: CGFloat

 if yourScore >= averageScore {
 // Normal case: calculate proportional height
 let ratio = CGFloat(yourScore) / CGFloat(averageScore)
 yourBarHeight = baseHeight * ratio
 } else {
 // Edge case: User score lower than average (shouldn't happen based on requirements)
 // Still show at least equal height

```

```

 yourBarHeight = baseHeight
 }

 // If user is X% worse, add that percentage to the bar
 // Example: User 22% worse → yourBarHeight = averageBarHeight * 1.22
 if percentageDifference > 0 {
 let adjustedHeight = averageBarHeight * (1 + CGFloat(percentDifference) / 100)
 return (adjustedHeight, averageBarHeight)
 }

 return (yourBarHeight, averageBarHeight)
}
}

```

```

struct RecommendedStep {
 let icon: String
 let title: String
 let description: String
}

```

### Success Criteria:

- ☒ Chart calculation correct (Your Score never < Average)
- ☒ Percentage properly connected to formula
- ☒ Continue button sticky at bottom
- ☒ Key Insights shortened
- ☒ Recommended steps show Breaksy features
- ☒ Disclaimer compressed but readable

-----

## 🎯 Implementation Strategy

### Phase 1: Critical Fixes (Week 1)

**\*\*Priority: Restore functionality\*\***

...

Day 1-2: Navigation Fixes

- ☒ Questions Back Button (Q2→Q1, Q1→Login)
- ☒ About You → Question 10 Navigation
- ☒ Login Page Back Button

Day 3-4: Chart & Calculations

- ☒ Score Analysis Chart Logic
- ☒ Percentage Formula Connection
- ☒ Bar Height Calculations

Day 5: Testing

- ☒ End-to-end navigation flow

- ✓ All popups and confirmations
- ✓ Data persistence

### ### Phase 2: UI/UX Polish (Week 2)

**\*\*Priority: Professional appearance\*\***

#### Day 1-2: Major Screens

- ✓ Splash Screen Redesign (PS3 waves + stars)
- ✓ Login Page Styling (animated text, new buttons)

#### Day 3-4: Content Screens

- ✓ Profile Card Optimization
- ✓ Score Analysis Content (insights, recommendations)

#### Day 5: Consistency

- ✓ Logo positioning (40pt from top everywhere)
- ✓ Color scheme verification
- ✓ Font consistency

### ### Phase 3: Refinement (Week 3)

**\*\*Priority: Details & polish\*\***

#### Day 1: Assets

- ✓ App Logo Integration
- ✓ Laurel Stars (PNG + Code versions)

#### Day 2: Minor Screens

- ✓ Welcome Screen Cleanup
- ✓ Questions Layout Optimization

#### Day 3-4: Final Details

- ✓ Age validation UI
- ✓ Skip confirmations
- ✓ Checkmarks (optional)

#### Day 5: QA

- ✓ Device testing (various screen sizes)
- ✓ Dark mode verification
- ✓ Accessibility audit

### ## Testing Checklist



### ### Navigation Flow

- [ ] Welcome → Login → Profile Card → Questions 1-10 → About You → Score Analysis
- [ ] Back buttons work correctly at each step
- [ ] Skip confirmations appear and function
- [ ] Question 1 back → Login Page
- [ ] About You back → Question 10

### ### UI Consistency

- [ ] Logo at 40pt from top on all screens
- [ ] Back arrow aligned with logo
- [ ] Dark color scheme throughout
- [ ] Font sizes appropriate and readable
- [ ] Sticky buttons at bottom where specified

### ### Data & Calculations

- [ ] Age validation shows error for >100
- [ ] Chart bars display correctly (Your Score ≥ Average)
- [ ] Percentage calculation accurate
- [ ] Answered questions marked (if checkmarks implemented)

### ### Content

- [ ] All instances say "Breaksy" (not "Breakzy")
- [ ] "X (Twitter)" changed to "X"
- [ ] Welcome text only on Login Page
- [ ] Recommended steps show Breaksy features
- [ ] Key Insights concise
- [ ] Disclaimer compressed but readable

-----

## ## 🚀 Claude Code Usage Guide

### ### For Each Screen:

```
```bash
# Step 1: Analyze current implementation
claude code --files "ScreenName.swift" --prompt "Analyze current implementation and identify issues from requirements"

# Step 2: Implement fixes
claude code --files "ScreenName.swift" --prompt "[Copy relevant prompt from guide above]"

# Step 3: Verify changes
claude code --files "ScreenName.swift" --prompt "Verify all requirements met and test edge cases"

# Step 4: Iterate if needed
claude code --continue --prompt "Fix [specific issue] based on testing"
```
```

### ### Recommended Order:

1. **\*\*Start with Critical Fixes\*\*** (Phase 1)

- `QuestionViewController.swift` (Navigation)
- `AboutYouViewController.swift` (Navigation)
- `ScoreAnalysisViewController.swift` (Calculations)
- 1. **\*\*Move to Visual Impact\*\*** (Phase 2)
- `SplashScreenViewController.swift`
- `LoginPageViewController.swift`
- 1. **\*\*Polish Details\*\*** (Phase 3)
- `WelcomeScreenViewController.swift`
- `ProfileCardViewController.swift`

-----

## ## ⚠ Important Notes

### ### Brand Consistency

- **\*\*ALWAYS\*\*** use “Breaksy” (exact spelling)
- Never use: “Breakzy”, “breakzy”, “BREAKZY”
- Verify with: ``grep -r "reakz" .`` (should return 0 results)

### ### Asset Names

```

- ✅ Correct:
- BreaksyLogo
 - LaurelRatingStars
 - BreaksyApplcon

❌ Incorrect:

- BreakyLogo
- Breakzylcon

```

### ### Navigation Rules

- Question N → Question N-1 (back button)
- Question 1 → Login Page (back button)
- About You → Question 10 (back button)
- Skip buttons → Confirmation popup → Next screen or Cancel

### ### Chart Logic

```
```swift
// ALWAYS TRUE:
yourScoreBarHeight >= averageScoreBarHeight

// Formula:
if userScore > averageScore {
    percentageDifference = (userScore - averageScore) / averageScore * 100
    yourBarHeight = averageBarHeight * (1 + percentageDifference/100)
}
```
```

-----

## ## 🛠 Troubleshooting

### ### Common Issues:

#### \*\*Navigation not working?\*\*

- Check if view controllers in navigation stack
- Verify `popToViewController(ofClass:)` extension exists
- Ensure proper initialization of questionIndex

#### \*\*Chart bars incorrect?\*\*

- Verify data source (userScore, averageScore)
- Check constraint connections in Storyboard
- Confirm calculation in ScoreData struct

#### \*\*Animations not smooth?\*\*

- Use `view.layoutIfNeeded()` inside animation block
- Check constraint priorities
- Verify duration and easing function

#### \*\*Text not updating?\*\*

- Search entire project for "Breakzy" → replace with "Breaksy"
- Check IBOutlet connections
- Verify string localization files

-----

### ## Final Verification

Before submitting each screen:

```
```swift
// Run this verification
func verifyImplementation() {
    // 1. Brand Check
    assert(!codeContains("Breakzy"), "Found 'Breakzy' - should be 'Breaksy'")

    // 2. Navigation Check
    assert(backButtonWorks(), "Back button not functional")
    assert(skipConfirmationShows(), "Skip confirmation missing")

    // 3. Layout Check
    assert(logoTopConstraint.constant == 40, "Logo not at 40pt from top")
    assert(stickyButtonAtBottom(), "Button not sticky at bottom")

    // 4. Data Check
    assert(chartBarsCorrect(), "Chart calculations wrong")
    assert(percentageConnected(), "Percentage not connected to formula")

    print("
```

If Claude Code encounters issues:

1. ****Check this guide**** for exact requirements
1. ****Verify assets exist**** in Xcode project
1. ****Test incrementally**** - one feature at a time
1. ****Use debug prints**** to trace issues
1. ****Review constraint conflicts**** in console

****Ready to implement? Start with Phase 1, Screen 1! 🚀****