I Have This Idea for an App...

Session 203

Jessie Pease, Health Engineering
Tanu Singhal, UIKit Engineering
“To live a creative life, we must lose the fear of being wrong.”

Joseph Chilton Pearce
Our Journey 🚶

Dream big and organize our ideas
Our Journey 🚶‍♂️

Dream big and organize our ideas

Learn to navigate Xcode
Our Journey 🌿

Dream big and organize our ideas
Learn to navigate Xcode
Build a simple game using Swift
Our Journey 🚶

Dream big and organize our ideas
Learn to navigate Xcode
Build a simple game using Swift
Add multiple views within our app
Our Journey 🚶‍♂️

Dream big and organize our ideas
Learn to navigate Xcode
Build a simple game using Swift
Add multiple views within our app
Persist and display data for users
Start Game!
Leaderboard
Start Game!
Leaderboard
Navigating Xcode
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {
    var window: UIWindow?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point to customize application initialization. For the changes to take effect, recompile and run your app.
        return true
    }

    func applicationWillResignActive(_ application: UIApplication) {
        // called when the application is about to move from active to inactive state. This can occur for types of temporary interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.
        // Use this method to perform actions that the application will not be notified about when it reenters the active state.
    }

    func applicationDidEnterBackground(_ application: UIApplication) {
        // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to re-create it when the application re-enters the active state. When your application enters the background, its current state of execution is saved and the application will be re-launched later.
        // This method may be called more than once when the application is terminated.
    }

    func applicationWillEnterForeground(_ application: UIApplication) {
        // Called as part of the transition from the background to the active state; here you can undo many of the changes made on entering the background.
        // Note: If you request core location and if application:didFinishLaunching: was performed, this method will be called again when location services are enabled.
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }
}
class AppDelegate: UIResponder, UIApplicationDelegate {

    // ...
}

func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // ...
    return true
}

func applicationWillResignActive(_ application: UIApplication) {
    // ...
}

func applicationWillEnterForeground(_ application: UIApplication) {
    // ...
}

func applicationWillEnterBackground(_ application: UIApplication) {
    // ...
}

func applicationWillTerminate(_ application: UIApplication) {
    // ...
}

// ...

```swift
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }
}
```
Creating Views with the Storyboard
Demo
Creating the view for our app
Game Logic
Start Screen

Wait for button press

Playing
Start Screen

Wait for button press

Playing

Show🦄 or 💩
Start Screen

Wait for button press

Playing

Show 🦄 or 💩

1s Timer
Playing
Start Screen

Wait for button press

Lonely
Avoided

Playing

Show 🦄 or 💩

1s Timer

Wait for emoji press
Playing
Start Screen
- Wait for button press

Lonely
- Avoided
- 1s Timer
  - Wait for emoji press
  - Show 🦄 or 💩

Playing
- Show 🦄 or 💩
Playing
Start Screen
Wait for button press

Wait for emoji press

Lonely
Avoided

1s Timer
Show 🦄 or 💩
// Game Logic - Start New Game

@IBAction func startPressed(_ sender: Any) {
    startNewGame()
}

Start Screen

Wait for button press
@IBAction func startPressed(_ sender: Any) {
    startNewGame()
}
//Game Logic - Start New Game

@IBAction func startPressed(_ sender: Any) {
    startNewGame()
}

Start Screen

Wait for button press
// Game Logic - Start New Game

@IBAction func startPressed(_ sender: Any) {
    startNewGame()
}

func startNewGame() {
    startGameButton.isHidden = true
    leaderboardButton.isHidden = true
    gamePoints = 0
    updatePointsLabel(gamePoints)
    pointsLabel.textColor = .magenta
    pointsLabel.isHidden = false
    oneGameRound()
}
// Game Logic - Start New Game

@IBAction func startPressed(_ sender: Any) {
    startNewGame()
}

func startNewGame() {
    startGameButton.isHidden = true
    leaderboardButton.isHidden = true
    gamePoints = 0
    updatePointsLabel(gamePoints)
    pointsLabel.textColor = .magenta
    pointsLabel.isHidden = false
    oneGameRound()
}

Start Screen

Wait for button press
/Game Logic - Start New Game

@IBAction func startPressed(_ sender: Any) {
    startNewGame()
}

func startNewGame() {
    startGameButton.isHidden = true
    leaderboardButton.isHidden = true
    gamePoints = 0
    updatePointsLabel(gamePoints)
    pointsLabel.textColor = .magenta
    pointsLabel.isHidden = false
    oneGameRound()
}
/**Game Logic - Start New Game**

@IBAction func startPressed(_ sender: Any) {
    startNewGame()
}

func startNewGame() {
    startGameButton.isHidden = true
    leaderboardButton.isHidden = true
    gamePoints = 0
    updatePointsLabel(gamePoints)
    pointsLabel.textColor = .magenta
    pointsLabel.isHidden = false
    oneGameRound()
}
// Game Logic - Start New Game

@IBAction func startPressed(_ sender: Any) {
    startNewGame()
}

func startNewGame() {
    startGameButton.isHidden = true
    leaderboardButton.isHidden = true
    gamePoints = 0
    updatePointsLabel(gamePoints)
    pointsLabel.textColor = .magenta
    pointsLabel.isHidden = false
    oneGameRound()
}
// Game Logic - Start New Game

@IBAction func startPressed(_ sender: Any) {
    startNewGame()
}

func startNewGame() {
    startGameButton.isHidden = true
    leaderboardButton.isHidden = true
    gamePoints = 0
    updatePointsLabel(gamePoints)
    pointsLabel.textColor = .magenta
    pointsLabel.isHidden = false
    oneGameRound()
}
func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
        repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        }
    }
}
func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                                  repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        }
    }
}
// Game Logic - One Game Round

```swift
func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                                  repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        } else {
            self.oneGameRound()
        }
    }
}
```
/Game Logic - One Game Round

func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                                repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        } else {
            self.gameOver()
        }
    }
}
func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                                  repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        } else {
            self.gameOver()
        }
    }
}
// Game Logic - One Game Round

func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                                  repeats: false) { _ in 
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.gameOver()
            }
        } else {
            self.oneGameRound()
        }
    }
}
// Game Logic - One Game Round

func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                                  repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.gameOver()
            }
        } else {
            self.oneGameRound()
        }
    }
}
func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
        repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        } else {
            self.oneGameRound()
        }
    }
}
// Game Logic - One Game Round

func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                                  repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        } else {
            self.gameOver()
        }
    }
}
//Game Logic - One Game Round

func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                               repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.gameOver()
            }
        } else {
            self.oneGameRound()
        }
    }
}

Playing

Lonely

Show 🦄 or 💩

Avoided 🦄

1s Timer
func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
        repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        } else {
            self.oneGameRound()
        }
    }
}
// Game Logic - One Game Round

func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                                repeats: false) { _ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        }
    }
}
func oneGameRound() {
    updatePointsLabel(gamePoints)
    displayRandomButton()

    timer = Timer.scheduledTimer(withTimeInterval: 1.0,
                                  repeats: false) {_ in
        if self.state == GameState.playing {
            if self.currentButton == self.goodButton {
                self.gameOver()
            } else {
                self.oneGameRound()
            }
        }
    }
}
/Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed

@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}

// Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed
@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}
// Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed
@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}
// Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed
@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}
//Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed

@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}
// Game Logic – Good or Bad button pressed

// Add a point, and update the score when the good button is pressed
@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}
// Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed
@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}

@IBAction func badButtonPressed(_ sender: Any) {
    badButton.isHidden = true
    timer?.invalidate()
    gameOver()
}
// Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed
@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}

@IBAction func badButtonPressed(_ sender: Any) {
    badButton.isHidden = true
    timer?.invalidate()
    gameOver()
}
// Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed
@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}

@IBAction func badButtonPressed(_ sender: Any) {
    badButton.isHidden = true
    timer?.invalidate()
    gameOver()
}
// Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed
@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}

@IBAction func badButtonPressed(_ sender: Any) {
    badButton.isHidden = true
    timer?.invalidate()
    gameOver()
}
// Game Logic - Good or Bad button pressed

// Add a point, and update the score when the good button is pressed
@IBAction func goodButtonPressed(_ sender: Any) {
    gamePoints = gamePoints + 1
    updatePointsLabel(gamePoints)
    goodButton.isHidden = true
    timer?.invalidate()
    oneGameRound()
}

@IBAction func badButtonPressed(_ sender: Any) {
    badButton.isHidden = true
    timer?.invalidate()
    gameOver()
}
Demo
Bring in the game logic
Accomplishments 🎉
Accomplishments 🎉

Navigating Xcode to learn how to create fun apps
Accomplishments 🎉

Navigating Xcode to learn how to create fun apps
Creating a simple UI using the Storyboard
Accomplishments 🎉

Navigating Xcode to learn how to create fun apps
Creating a simple UI using the Storyboard
Connecting our UI to code
Accomplishments 🎉

- Navigating Xcode to learn how to create fun apps
- Creating a simple UI using the Storyboard
- Connecting our UI to code
- Writing game logic code in Swift
Taking It Further! 🚀
Taking It Further! 🚀

Use SpriteKit to give more life to your good and bad buttons
Taking It Further! 🚀

Use SpriteKit to give more life to your good and bad buttons

Add MusicKit integration to incorporate sound into the game play
Taking It Further!

Use SpriteKit to give more life to your good and bad buttons

Add MusicKit integration to incorporate sound into the game play

Read from sensors and change the speed of the game based on user movement

<table>
<thead>
<tr>
<th>Introduction to SpriteKit</th>
<th>WWDC 2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing MusicKit</td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>Creating Immersive Apps with Core Motion</td>
<td>WWDC 2017</td>
</tr>
</tbody>
</table>
Enhancing Your App
Start Game!

Leaderboard
Beam Seilaudom
300
Nick Jones
280
Chad Isanhart
250
Mike Valentine
221
Christopher Foss
200
Jeanne Fox
192
Kevin Will Chen
188
Lexi Torres
149
Sarah Milos
130
Tamsin Vantress
129
Tanu Singhal
120
Areas of Focus
Areas of Focus

Data
Areas of Focus

Data

User Interface
Areas of Focus

Data
User Interface
Logic
Model-View-Controller

- Controller
- View
- Model
Model-View-Controller

UIViewController

View

Model
Model-View-Controller

UIViewController

(UIView)

(Model)
Demo
Setting up a data model
# Saving Data

<table>
<thead>
<tr>
<th>Core Data Best Practices</th>
<th>Executive Ballroom</th>
<th>Thursday 2:00PM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networking with NSURLSession</td>
<td></td>
<td>WWDC 2015</td>
</tr>
<tr>
<td>Introducing CloudKit</td>
<td></td>
<td>WWDC 2014</td>
</tr>
</tbody>
</table>
Leaderboard

UINavigationBar
Leaderboard

- UITableView
  - UINavigationBar
  - UITableViewCell
Leaderboard

- UINavigationBar
- UIImageView
- UITableView
- UITableViewCell
Jeanne Fox
Rank #6
192 Points
Details

A mobile screen showing a leaderboard. The leaderboard contains a profile picture of a woman named Jeanne Fox, ranked #6 with 192 Points.
Demo
Creating the user interface
View Controllers
View Controllers
View Controllers

Segue
Navigation Controllers and Table View Controllers

Unicorn Game
Navigation Controllers and Table View Controllers
Navigation Controllers and Table View Controllers
Navigation Controllers and Table View Controllers
<table>
<thead>
<tr>
<th>Section</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>View Controller Programming Guide for iOS</td>
<td>Apple Developer Documentation</td>
</tr>
<tr>
<td>Table View Programming Guide</td>
<td>Apple Developer Documentation</td>
</tr>
<tr>
<td>Navigation Controllers</td>
<td>Apple Developer Documentation</td>
</tr>
</tbody>
</table>
# Auto Layout

<table>
<thead>
<tr>
<th>Session Title</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mysteries of Auto Layout, Part 1</td>
<td>WWDC 2015</td>
</tr>
<tr>
<td>Auto Layout Techniques in Interface Builder</td>
<td>WWDC 2017</td>
</tr>
<tr>
<td>High Performance Auto Layout</td>
<td>Hall 2</td>
</tr>
<tr>
<td></td>
<td>Wednesday 3:00PM</td>
</tr>
</tbody>
</table>
Demo

View controller and logic
Next Steps
Next Steps

Test your app using XCTest framework
Next Steps

Test your app using XCTest framework

Review App Store Review Guidelines
Next Steps

Test your app using XCTest framework

Review App Store Review Guidelines

Enroll in the Apple Developer Program
Next Steps

Test your app using XCTest framework
Review App Store Review Guidelines
Enroll in the Apple Developer Program
Submit for App Review
Next Steps

Test your app using XCTest framework

Review App Store Review Guidelines

Enroll in the Apple Developer Program

Submit for App Review

Tell the world!
Summary
Summary

Explore Xcode
Summary

Explore Xcode

Build a user interface
Summary

Explore Xcode

Build a user interface

Think about the data
Summary

Explore Xcode

Build a user interface

Think about the data

Create good experiences for all devices
Summary

Explore Xcode
Build a user interface
Think about the data
Create good experiences for all devices
Follow best practices
Summary

Explore Xcode

Build a user interface

Think about the data

Create good experiences for all devices

Follow best practices

Congratulations! You’re an iOS App Developer!
More Information
