

# iOS Automation: The Trifecta

Elizabeth Taylor  
Staff QA Engineer, Digimarc Corp.  
@blackstonefinn

# Agenda

- Introduction
- Why automate
- What to automate
- How to automate using Xcode Instruments
  - The Trifecta
  - Sample Code examples
  - Demo

Search

## Digimarc Discover

digimarc discover



+ INSTALL

Category: Lifestyle  
 Updated: Sep 27, 2012  
 Version: 3.7  
 Size: 3.9 MB  
 Language: English  
 Seller: Digimarc Corporation  
 © 1995-2012 Digimarc, All Rights Reserved

Rated 4+

Requirements:  
 Compatible with iPhone 3GS, iPhone 4, iPhone 4S, iPhone 5, iPod touch (4th generation), iPod touch (5th generation), iPad 2 Wi-Fi, iPad 2 Wi-Fi + 3G, iPad (3rd generation) and iPad Wi-Fi + 4G.  
 Requires iOS 5.0 or later.

[Developer Web Site](#)[App Support](#)[Application License Agreement](#)

## Digimarc

# Digimarc Discover

[Developer Page >](#)[Tell a Friend >](#)**Description**

Digimarc® Discover lets you use your iPhone, iPad or iPod Touch to detect imperceptible digital watermarks, QR codes or barcodes in printed materials and to identify songs. Read a watermark or QR code to link you to information, video, and related content through many different types of network services. Identify songs to learn the title and artist, see cover art, and shop for the son... [More ▾](#)

**What's New in Version 3.7** Updated Sep 27, 2012

- + Optimized for iOS 6 and the new iPhone 5
- + Fix for in-app browser scrolling issue

**Customer Ratings**[Current Version](#)[All Versions](#)

Tap to rate

Average rating for all versions: 58 Ratings



Featured



Genius



Top Charts



Categories



Purchased

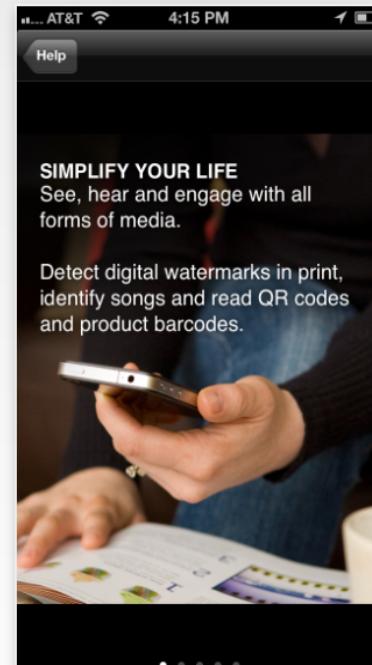


Updates

digimarc discover

**Digimarc Disc...**Digimarc  
Lifestyle

(58)

**OPEN**

1 of 1



Featured



Top Charts



Genius



Search



Updates

# Why Automate?

- Integration testing
- Catches bugs manual testing can miss
- Exploratory Testing
- Automation <3 Stress Tests!
- Tie it to build system for check-in
- Regression testing
- Fast < 10 min per functional area
- And most of all ... it's FUN ☺

# What to automate?

- Ask....      is it BBB?
- Basic App Functionality
- Boring stuff – copy & legal
- “Break it” stuff – stress tests & bugs

# Automation Framework

Choose a Template for the Trace Document:

- iOS**
  - All
  - Memory
  - CPU
  - I/O Activity
  - Graphics
- iOS Simulator**
  - All
  - Memory
  - CPU
  - File System
- Mac OS X**
  - All
  - Memory
  - CPU
  - File System
  - Behavior
- User**
  - All



## Automation

This template executes a script which simulates UI interaction for an iOS application launched from Instruments.



Open an Existing File...

Cancel

Record

Choose

# Script Framework - Trifecta

1. Navigate Functions

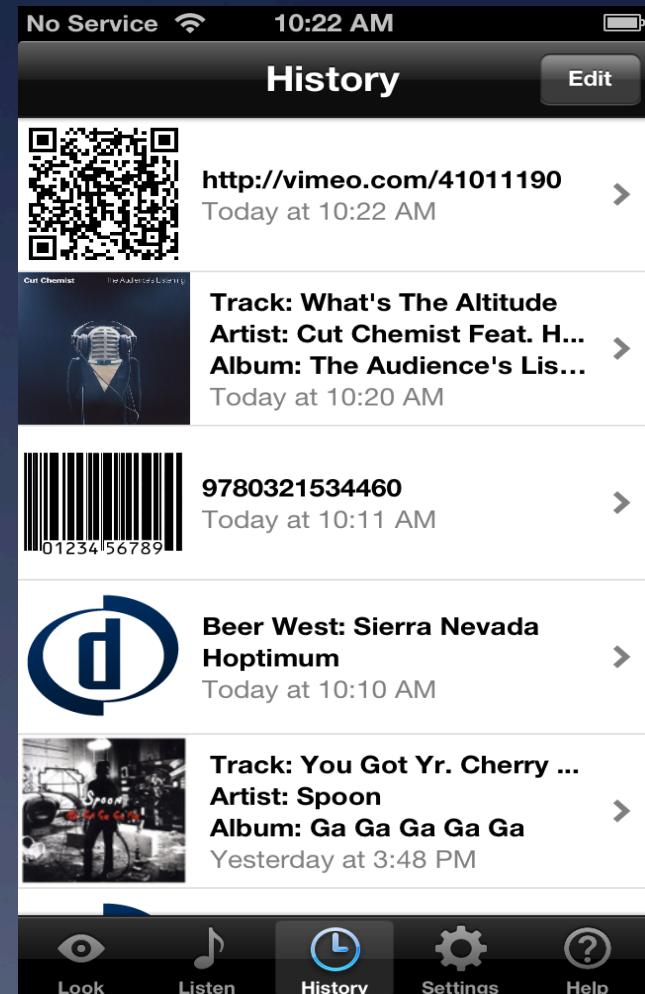
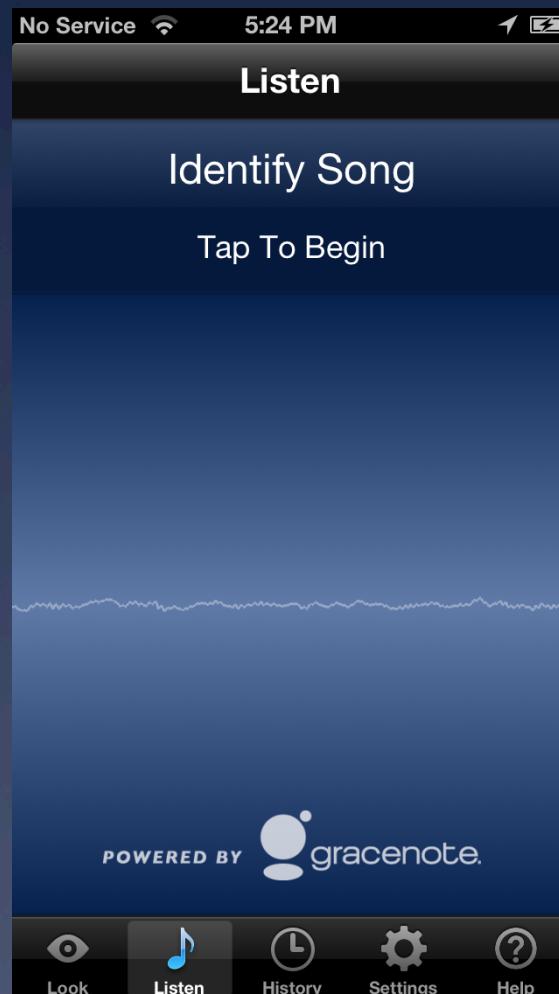
2. Validate Functions

3. Test Suite

# Script Framework - Trifecta

1. Navigate functions
  - Identify UI Elements
  - common navigation paths
  - wrap functions
2. Validate functions
  - UI element validation
  - functional tests
  - stress tests
3. Test Suite
  - import libraries – navigate and validate
  - list validate functions

# Digimarc Discover UI



# Digimarc Discover UI



# Navigation Functions

Navigation functions are used to find the best path from a start state to a goal state.

They are typically used in robotics and game development to solve the navigation problem.

There are several types of navigation functions, including:

- Potential field navigation

- A\* search

- Dijkstra's algorithm

- Depth-first search

- Breadth-first search

# Navigation Functions - Purpose

- Frequently used paths, actions
- Separate test setup v. feature test
- Contain iPhone v. iPad UI differences

# Creating Nav Functions

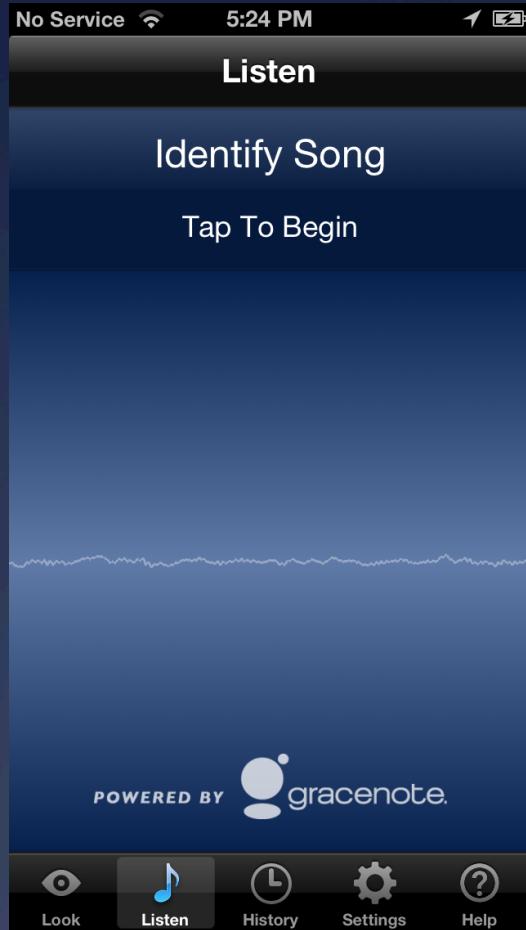
1. ID App UI elements
2. Write functions - code, debug, code
3. Wrap functions

# Creating Nav Functions

## 1. Identify UI elements

```
logElementTree();
```

# ID UI Elements



```
var target = UIATarget.localTarget();
var app = target.frontMostApp();
var mainWindow = app.mainWindow();

mainWindow.logElementTree();
```

# Listen – UI Elements

The image shows a screenshot of the Listen app interface on a mobile device. The top status bar indicates 'No Service' and the time '5:24 PM'. The main screen has a dark blue background with a wavy pattern at the bottom. At the top, there's a black navigation bar with the word 'Listen' in white. Below it, a large white button labeled 'Identify Song' with the placeholder text 'Tap To Begin' is centered. In the bottom left corner, there's a 'POWERED BY' logo for 'gracenote' with a small globe icon. At the very bottom, a black tab bar features five icons: 'Look' (eye), 'Listen' (music note), 'History' (clock), 'Settings' (gear), and 'Help' (question mark).  
  
To the right of the screenshot is a detailed hierarchical tree view of the UI elements:

- UIAWindow: rect:{(0, 0), {320, 568}}
  - UIAStaticText: name:Identify Song value:Identify Song rect:{(2...}
  - UIAStaticText: name:Tap To Begin value:Tap To Begin rect:{(2...
  - UIAImage: name:poweredByGracenote.png rect:{(52, 459}, {21...
- UIANavigationBar: name:Listen rect:{(0, 20}, {320, 44})
  - UIAImage: rect:{(0, 20}, {320, 44})
    - UIAStaticText: name:Listen value:Listen rect:{(130, 29}, {60, 27)}
- UIATabBar: rect:{(0, 519}, {320, 49})
  - UIAImage: rect:{(0, 516}, {320, 3})
  - UIAImage: rect:{(0, 519}, {320, 49})
  - UIAButton: name:Look rect:{(2, 520}, {60, 48})
  - UIAButton: name:Listen value:1 rect:{(66, 520}, {60, 48})
  - UIAButton: name:History rect:{(130, 520}, {60, 48})
  - UIAButton: name:Settings rect:{(194, 520}, {60, 48})
  - UIAButton: name:Help rect:{(258, 520}, {60, 48})

# Navigate Function - goToTab

- Frequently used paths, actions

```
function goToTab(tabName)
{
    "use strict";
    var target = UIATarget.localTarget();
    var app = target.frontMostApp();
    var tabBar = app.tabBar();

    // Select Tab if not already selected
    if (tabBar.selectedButton().name() != [tabName])
        tabBar.buttons()[tabName].tap();
    UILogger.logMessage("Tap " + tabName);

}
```

# Navigate Function - Back

```
function back()

{
    "use strict";
    var target = UIATarget.localTarget();
    var app = target.frontMostApp();
    var mainWindow = app.mainWindow();
    var navBar = mainWindow.navigationBar();
    var backButton = navBar.leftButton();

    while (backButton.isValid())
    {
        backButton.tap();
        target.delay(1);

        backButton = navBar.leftButton();
    }
}
```

# ID UI Elements



```
var target = UIATarget.localTarget();
var app = target.frontMostApp();
var mainWindow = app.mainWindow();

mainWindow.logElementTree();
```

# Help UI Elements



Log Messages	Screenshot
▼ UIAWindow: rect:{(0, 0), {320, 568}}	
▼ UITableView: name:Empty list value:rows 1 to 4 of 4 rect:{(0...}	
► UITableViewCell: name:Take a Tour rect:{(0, 74), {320, 46}}	
► UITableViewCell: name:About rect:{(0, 140), {320, 45}}	
► UITableViewCell: name:FAQ rect:{(0, 185), {320, 44}}	
► UITableViewCell: name:Feedback rect:{(0, 229), {320, 45}}	
▼ UINavigationBar: name:Help rect:{(0, 20), {320, 44}}	
► UIImage: rect:{(0, 20), {320, 44}}	
UIAStaticText: name:Help value:Help rect:{(137, 29), {45, 27}}	
▼ UITabBar: rect:{(0, 519), {320, 49}}	
UIImage: rect:{(0, 516), {320, 3}}	
UIImage: rect:{(0, 519), {320, 49}}	
UIAButton: name:Look rect:{(2, 520), {60, 48}}	
UIAButton: name:Listen rect:{(66, 520), {60, 48}}	
UIAButton: name:History rect:{(130, 520), {60, 48}}	
UIAButton: name:Settings rect:{(194, 520), {60, 48}}	
UIAButton: name:Help value:1 rect:{(258, 520), {60, 48}}	

# Navigate Function – Tap Cell

- Separate test setup v. test logic

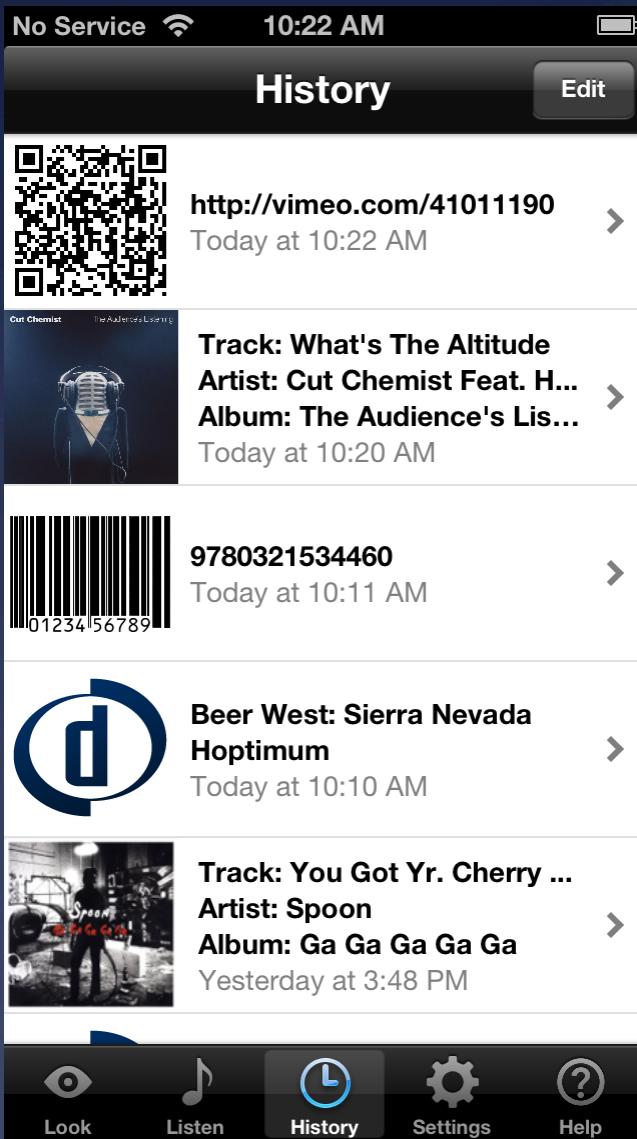
```
function tapCell(cellName)
{
    "use strict";
    var target = UIATarget.localTarget();
    var app = target.frontMostApp();
    var mainWindow = app.mainWindow();
    var table = mainWindow.tableViews()[0];
    var cell = table.cells()[cellName];

    // Select Cell
    if (cell.isValid())
    {
        table.cells()[cellName].tap();
        UIALogger.logMessage("Tap " + cellName);
    }
}
```

# Logging

```
UIAlogger.logMessage("Song Selected");
```

# Navigate Function – History List



View iPad differences

```
function goToHistoryList()
{
    "use strict";
    var target = UIATarget.localTarget();
    var app = target.frontMostApp();

    // Select History Tab
    goToHistory();
    target.delay(2);
    if ((target.model() == "iPad")
        && (appIsPortrait()))
    {
        var histButton = app.navigationBar().leftButton();
        if (histButton.isValid())
            app.navigationBar().leftButton().tap();
    }
    // Wait for List to Populate
    target.delay(2);
}
```

# Wrapping Functions

1. Readability of test script
2. Fix or change in one place

```
function goToHelpFAQ()  
{  
    "use strict";  
    var target = UIATarget.localTarget();  
  
    // Select Help Tab  
    goToHelp();  
  
    // Tap FAQ in Help Menu  
    tapCell("FAQ");  
  
    // allow time for FAQ to render  
    target.delay(2);  
}
```

# Target.delay(#)

Shift + ⌘ to Zoom In, Control + ⌘ to Zoom Out, Option + ⌘ to Time Filter

Index	Timestamp	Log Messages	Log Type	Screenshot
0	5:25:55 PM PDT	Help Test Suite	Start	
1	5:25:55 PM PDT	▶ Validate Help About Legal	Pass	
7	5:25:59 PM PDT	target.captureRectWithName("{orig...")	Debug	
8	5:26:01 PM PDT	HelpAboutLegal	Screenshot	
9	5:26:05 PM PDT	target.frontMostApp().mainWindow()	Debug	
10	5:26:06 PM PDT	target.frontMostApp().mainWindow()	Debug	
11	5:26:14 PM PDT	target.frontMostApp().mainWindow()	Debug	
12	5:27:03 PM PDT	Help Test Suite	Start	
13	5:27:03 PM PDT	▶ Validate Help About Legal	Pass	
19	5:27:04 PM PDT	target.captureRectWithName("{orig...")	Debug	
20	5:27:07 PM PDT	HelpAboutLegal	Screenshot	

# Navigate Functions ctd.

1. Rotation
2. Set App state at end of test

# Script Framework - Trifecta

- ✓ Navigate functions
  - Identify UI Elements
  - common navigation paths
  - wrap functions
- 2. Validate functions
  - UI element validation
  - functional tests
  - stress tests
- 3. Test Suite
  - import libraries – navigate and validate
  - list validate functions

# Validation Functions

# Validation Functions - Purpose

1. Validate UI elements are correct
  - Buttons, Headers
  - Copy / Strings
  - Images
2. Validate functionality
3. Stress tests

# Validate – UI Elements

No Service 7:02 PM

No Service 11:49 AM

No Service 10:29 AM

## Help

- Take a Tour**
- About**
- FAQ**
- Feedback**

**Look Listen History Settings Help**

## Feedback

Got a suggestion? We'd love to hear from you.

To report an issue, please include as much detail as possible including your phone model and iOS version as well as the name of the watermarked content you were reading or listening to.

Tap "Submit Feedback" to send us an email.

**Submit Feedback**

**Look Listen History Settings Help**

## History

**Cut Chemist The Audience's Listening**

**What's The Altitude**

**Artist:** Cut Chemist Feat. Hymnal

**Album:** The Audience's Listening

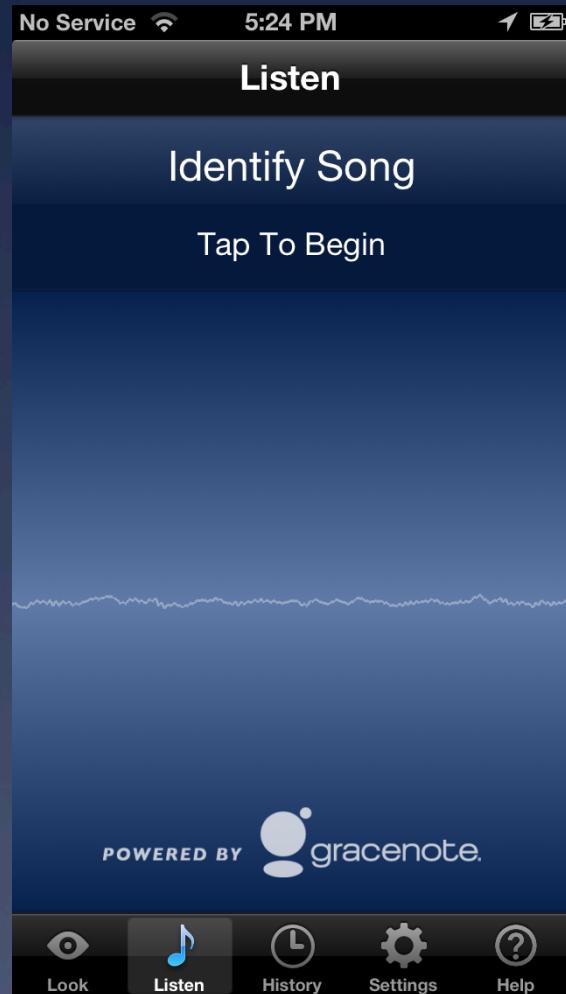
**Download on iTunes**

**Share**

POWERED BY

**gracenote**

# Validate - Listen UI Elements



```
function validateListenUI()
{
    // declare test name
    var testName = "TEST: Validate Listen UI";
    UIALogger.logStart(testName);

    // Call function to get to Listen mode
    goToListen();

    target.delay(2);

    // Validate UI elements: Nav Bar, Buttons, Gracenote Logo and copy
    var graceLogo = mainWindow.images()[GRACENOTE_IMAGE_FILENAME_LISTEN];
    var idSong = mainWindow.staticTexts()["Identify Song"];
    var idTapToBegin = mainWindow.staticTexts()["Tap To Begin"];
    if ((graceLogo.isValid()) &&
        (idSong.isValid()) &&
        (idTapToBegin.isValid())&&
        (app.navigationBar().name() == "Listen"))
    {
        UIALogger.logPass(testName);
        target.delay(2);
        target.captureScreenWithName("Listen UI");
        iPadRotate();
    }
    else
    {
        UIALogger.logFail(testName);
    }

    // reset to look mode
    resetAppToLookModePortrait();
}
```

# Validate – Help UI Elements



```
function validateHelpUI()
{
    // declare test name
    var testName = "TEST: Validate Help UI";
    UIALogger.logStart(testName);

    // Call function to go to help
    goToHelp();
    target.delay(4);

    // declare variable for cell and validate
    var tutorial = mainWindow.tableViews()["Empty list"].cells()["Take a Tour"];
    var about = mainWindow.tableViews()["Empty list"].cells()["About"];
    var faq = mainWindow.tableViews()["Empty list"].cells()["FAQ"];
    var feedback = mainWindow.tableViews()["Empty list"].cells()["Feedback"];
    if (
        (tutorial.checkIsValid())
        &&
        (about.checkIsValid())
        &&
        (faq.checkIsValid())
        &&
        (feedback.checkIsValid())
        &&
        (app.navigationBar().name() == "Help")
    )
    {
        UIALogger.logPass(testName);
        target.delay(2);
        target.captureScreenWithName("Help UI");
        iPadRotate();
    }
    else
    {
        UIALogger.logFail(testName);
    }

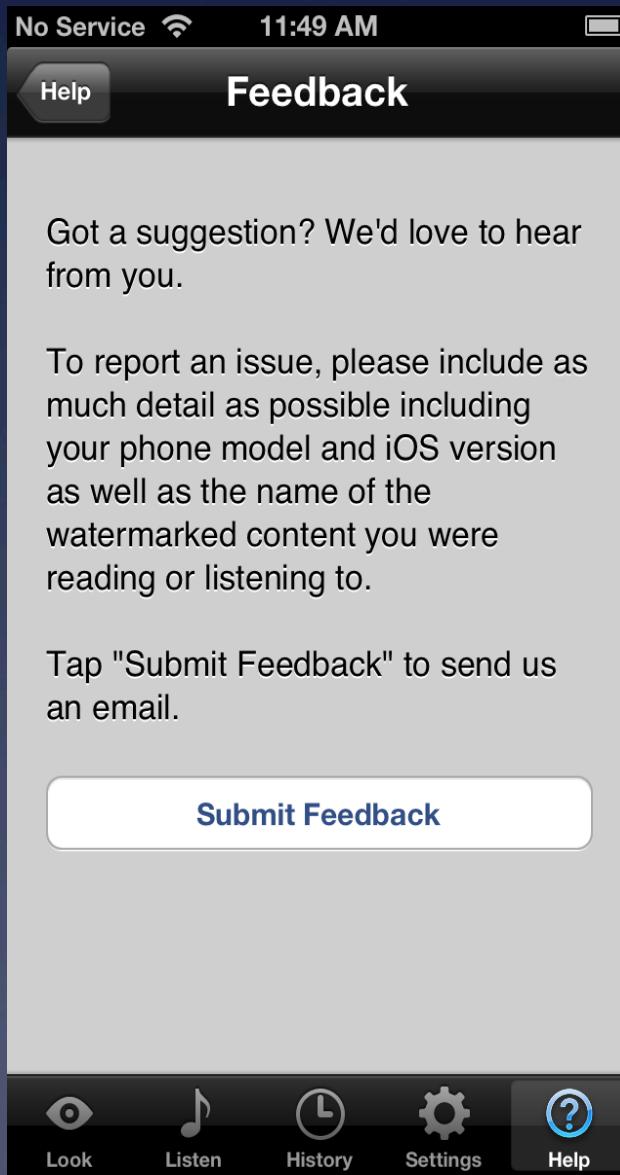
    target.delay(1);

    // reset to look mode
    back();
    resetAppToLookModePortrait();
}
```

# Logging

1. Make results easy to view
  - Test name w/in each function
2. Create using Log Message
3. Pass fail logging
4. Screen shots
5. Output name of test

# Validate – Feedback UI Elements

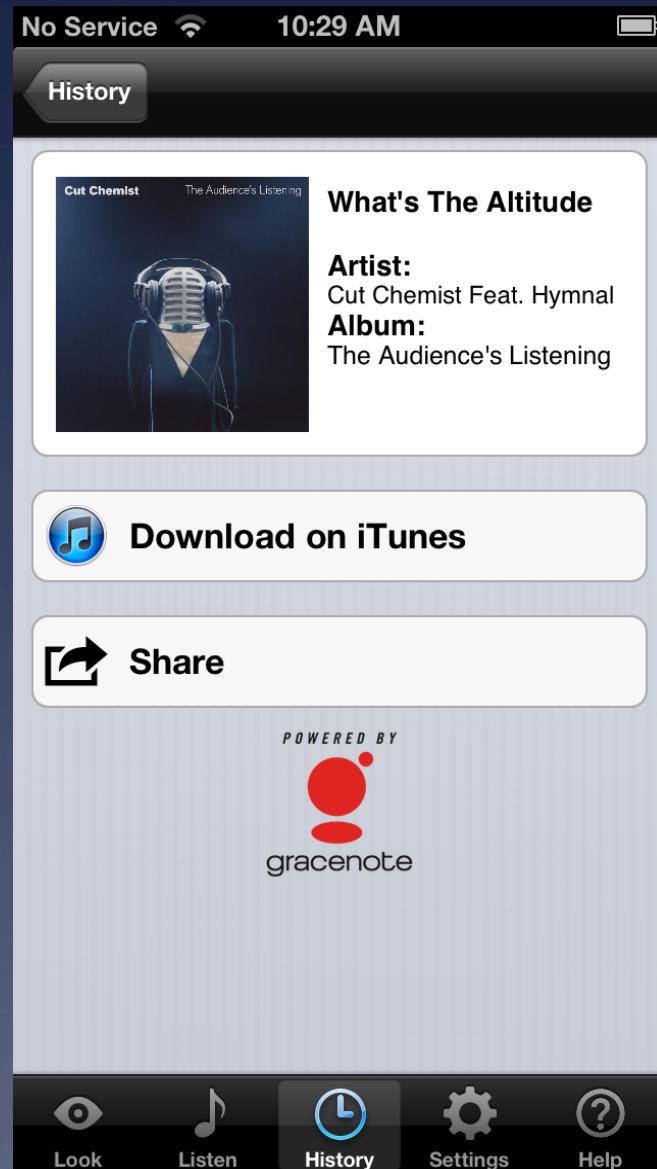
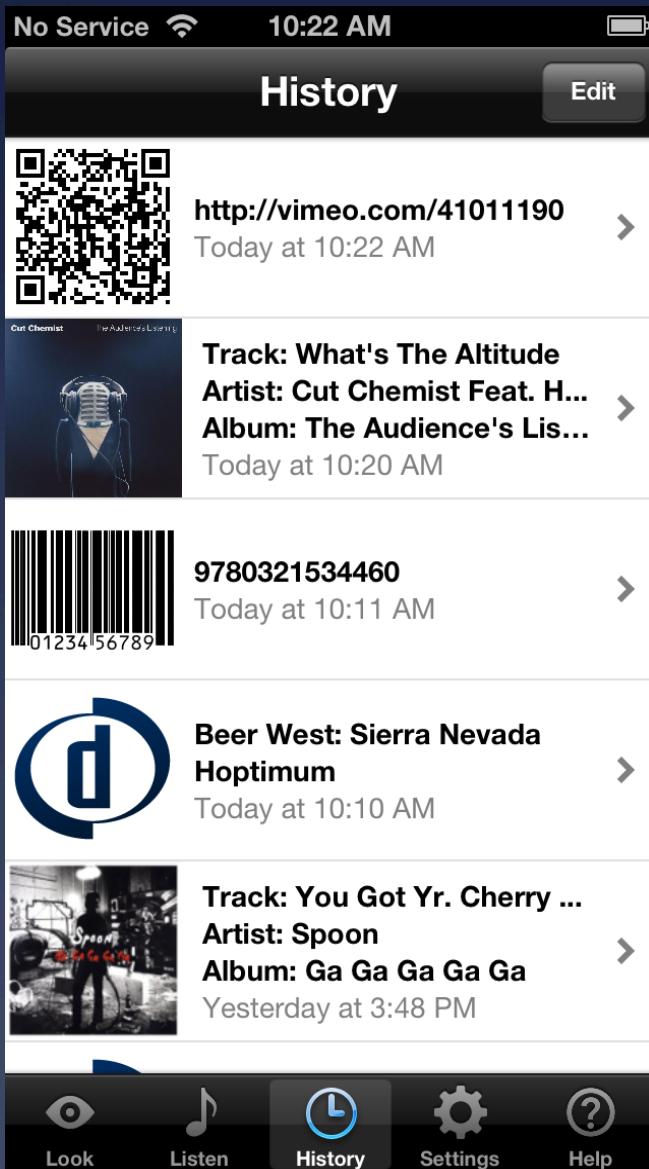


```
function validateSubmitFeedbackUI()
{
// Declare Test Name
var testName = "TEST: Validate submitFeedbackUI";
UIALogger.logStart(testName);

// Go to feedback page
goToHelpFeedback();
target.delay(1);

// Validate Feedback elements: Nav Bar, Buttons and Copy
var body = mainWindow.staticTexts()["Got a suggestion? We'd love to hear from you.\n\nTo report an is:
if (
    (body.checkIsValid())
    &&
    (app.navigationBar().name() == "Feedback")
    &&
    (app.navigationBar().leftButton().name() == "Help")
    &&
    (mainWindow.buttons()[0].name() == "Submit Feedback")
)
{
    UIALogger.logPass(testName);
    target.delay(2);
    target.captureScreenWithName("Submit Feedback UI");
    iPadRotate();
}
else
{
    UIALogger.logFail(testName);
}
    target.delay(1);
// reset for next test
back();
resetAppToLookModePortrait();
}
```

# Validate – UI Elements



# Validate Functions ctd.

1. Help –
  - Submitting Feedback email
2. History –
  - Payoff rendering in-app browser
  - Sharing functionality
  - Deleting history list items
3. Stress Tests

# Stress Test Candidates

1. Repeated navigation across the app
2. Drilling down into nested views
3. Adding Rotation into the mix
4. Exit and returning to the application
5. Repeated item selection - history list\*

# Script Framework - Trifecta

- ✓ Navigate functions
  - Identify UI Elements
  - common navigation paths
  - wrap functions
- ✓ Validate functions
  - UI element validation
  - functional tests
  - stress tests
- 3. Test Suite
  - import libraries – navigate and validate
  - list validate functions

# Test Suites

# Test Suites

1. Import Navigation lib
2. Import Validation lib for area
3. Organize by functional areas\*
4. Simple list of validation functions
5. Logging for output readability

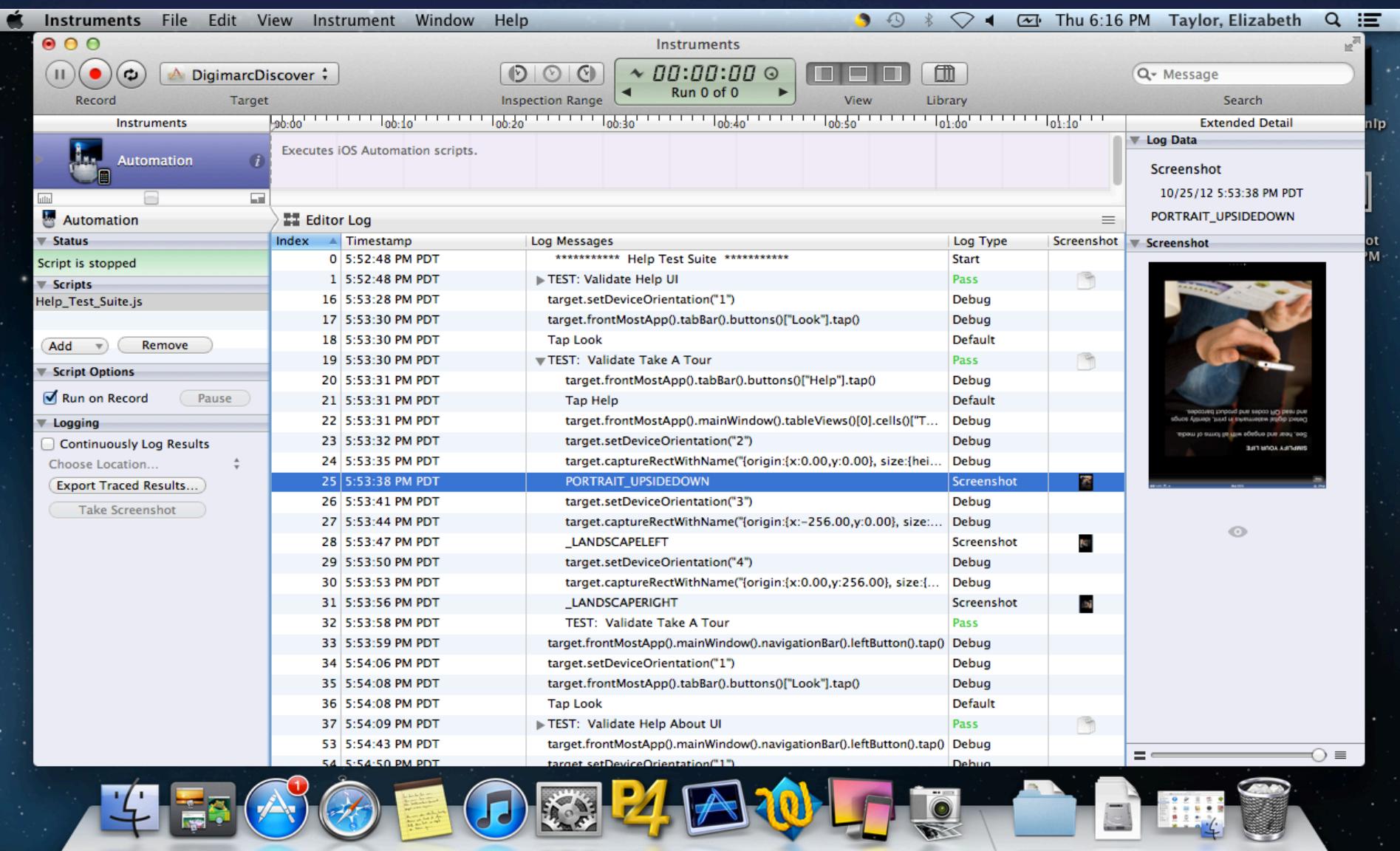
# Help - Test Suite

```
// import functionlibs
#import "../FunctionLibs/navigateFunctionLib.js";
#import "../FunctionLibs/validateHelp_FunctionLib.js";

// Declare Test Suite Name
var suiteName = " ***** Help Test Suite *****";
UIALogger.logStart (suiteName);

// Call Validation Functions
validateHelpUI();
validateTakeATour();
validateHelpAbout();
validateHelpAboutLegal();
validateDigiEULA();
validateFBEULA();
validateZXingEULA();
validateFAQ();
validateSubmitFeedbackUI();
validateSubmitFeedbackEmail();
```

# Help Test Suite Results



# Script Framework - Trifecta

- ✓ Navigate functions
  - Identify UI Elements
  - common navigation paths
  - wrap functions
- ✓ Validate functions
  - UI element validation
  - functional tests
  - stress tests
- ✓ Test Suite
  - import libraries – navigate and validate
  - list validate functions

# Demo

# Questions & Feedback

# Thank you!

@blackstonefinn