iOS App Anatomy

Dr. Sarah Abraham
Anatomy of a Mobile App

(worklight.com)
Apps built on a common set of phone features

- Libraries provided to use these features
- Standardized API calls access these libraries
- Third-party apps built upon these calls
- Libraries optimized and reusable in memory
- Less code to write and better performance
Frameworks

- Key pieces of code that make mobile applications easy to build and stable (ideally)

- Bundle (structured directory) contains:
  - Dynamic, shared library
  - Associated resources (images, headers etc)

- Frameworks shared between applications

- Fast access, reduced memory, consistent look ‘n’ feel
Using Frameworks

- Frameworks are designed for specific functionality
- Native code should fit its framework (not the other way around)
- Native code should make use of frameworks
- iOS development based on frameworks (Foundation, UIKit etc)
• Framework has default set of behaviors / functionality (i.e. methods)

• Programmer uses these methods to output desired behavior for app

• Developer code written to specialize the framework’s behavior
iOS App Anatomy

Windows layers of an iOS app

(UI/UX design for iOS 7, Vu Tran Lam)
Common iOS Frameworks

- **Foundation**: Low-level management of strings, collections, primitive data types, containers etc
- **UIKit**: Class-level management of iOS user-interface layer
- **CoreData**: Interfaces for managing app and user data
- **CoreGraphics**: Interfaces for 2D vector-based drawing engine
Other Systems in iOS

- Storyboard: Defines user interface and app flow
- Delegate: Coordinates multiples pieces and systems in the app
- View Controller: Manages user interface and display
- Views: Elements of the user interface
Storyboard

- Lays out user’s path
- Defines scenes
- Defines segues between scenes
- Uses Auto Layout for nice formatting
Delegate

- Pattern of development where one object in a program acts on another object’s behalf
- Coordinates between objects by passing messages
- Can return values to determine how to handle event
View Controller

- At least one per app
- Manages defined part of user interface
- Handles interactions between interface and underlying data
- Central to app development
Views

- Display presented to user
- Controls layout and subviews
- Handles drawing and animation
- Responds to events
- Created programmatically or through Storyboards
MVC Pattern

- Pattern guiding all iOS development
- Model-View-Controller
  - Model includes app-specific data, classes etc
  - View includes interface and screens presented to user
  - Controller dictates how model and view should change based on user input
MODEL

UPDATES

VIEW

MANIPULATES

CONTROLLER

USER

(Wikipedia)
Working in Xcode on Campus

- PCL Media Lab has 44 iMacs with Xcode installed
  - [http://www.lib.utexas.edu/services/media-labs/](http://www.lib.utexas.edu/services/media-labs/)
- To use Xcode:
  1. Open Xcode (use Spotlight Search or go through Applications folder)
  2. Check “Don’t Enable” when pop asks if you want to enable developer mode
  3. Enter your EID and password when it provides a prompt
  4. Xcode should now run
Xcode Layout Demo