Mobile App TDD

The purpose of this document is to describe the technical design of the Mobile App.

Table of Contents

[Project Layout 1](#_Toc419122042)

[Architectural Design 1](#_Toc419122043)

[Springboard 2](#_Toc419122044)

[Top Navigation 2](#_Toc419122045)

[Container 2](#_Toc419122046)

[Nav Toolbar 2](#_Toc419122047)

[Task 2](#_Toc419122048)

[TaskFragment / TaskUIViewController 2](#_Toc419122049)

[UserManagement 3](#_Toc419122050)

[App.Shared 3](#_Toc419122051)

[App.Shared.Notes 3](#_Toc419122052)

[App.Shared.RockApi 3](#_Toc419122053)

[App.Shared.UI 3](#_Toc419122054)

[App.Shared.Analytics 3](#_Toc419122055)

[Config.cs 4](#_Toc419122056)

[ConnectLink 4](#_Toc419122057)

[FacebookManager 4](#_Toc419122058)

[GroupFinder 4](#_Toc419122059)

[PrivateConfig.cs 4](#_Toc419122060)

[Strings.cs 4](#_Toc419122061)

[VimeoManager 4](#_Toc419122062)

## Project Layout

App.Shared

Code that is specific to Mobile App but used across both iOS and Android.

Droid

Android specific Mobile App code

iOS

iOS specific Mobile App code

Rock.Client.Models

Models from the Rock project

Rock.Mobile

Collection of platform agnostic code that can be reused in any app.

(Similar to App.Shared, except not specific to Mobile App)

## Architectural Design

While there are certain differences due to the nature of iOS and Android being different platforms, the general design is shared between both, although implemented specifically for each platform.

### Springboard

This is the ‘core’ class that owns all of the Task instances of the app.

A Task is the general term for the major “features”, like News, Connect, Prayer, etc.

The Springboard manages activating and deactivating Tasks as appropriate, and is loosely

modeled off the design of the iOS and Android home screens, where you use a Springboard to launch apps.

### Top Navigation

This class is defined in MainUINavigationController.cs on iOS, and NavbarFragment.cs on Android. It is the top “bar” that stores the hamburger icon as well as the app logo. It also acts as the parent to whatever UI is being currently displayed.

### Container

In iOS, this is defined in ContainerViewController.cs. On Android, it is defined main.axml as a “FrameLayout” with the id “activetask”. Conceptually they do the same thing. They are the parent view to any UI that wishes to display itself. Doing this allows the **Top Navigation** to control and communicate with the UI pages in an abstract way.

### Nav Toolbar

This is the bottom navigation bar that slides up and down. The idea is that it is a contextually aware bar that displays appropriate buttons that let a user perform an action that makes sense.

It is defined in NavToolbar.cs on both platforms.

For example, in the Prayer section, there is an “Add” button that lets them add a new Prayer Request. When reading a message, there is a “Share” button that lets them export the current note.

**On iOS only**, there is also a “back” button that lets the user navigate back a page within the current task.

### Task

This is a base class that all the main feature classes of the app derive from. NewsTask, ConnectTask, PrayerTask, etc. It acts as the parent and manages switching between the various UI screens that make up the task.

It specifically is defined in Tasks/Task.cs on both platforms.

### TaskFragment / TaskUIViewController

These are effectively the same thing, but specific to Android / iOS respectively. They act as the base UI class from which all other UI pages within a Task derive. Let’s look at the NewsTask as an example:

**NewsTask**

*Derives from Task*

**NewsPrimaryUIViewController / NewsPrimaryFragment**

*Derives from TaskUIViewController / TaskFragment*

**NewsDetailsUIViewController / NewsDetailsFragment**

*Derives from TaskUIViewController / TaskFragment*

### UserManagement

Certain features don’t warrant being in a Task. These are “stand alone” UIs that are launched by the **Springboard** and are modal. Examples of this include Profile Editing, Image Cropping, and User Login.

They derive from standard UIViewController / Fragment and are owned and instanced by the **Springboard**.

### App.Shared

As described above, App.Shared is a project containing code shared between iOS and Android. The more code in here, the better, because it works across platforms.

### App.Shared.Notes

Contains the implementation of the Note System, including the Note Database and all components of NoteScript.

NoteScript is built on top of Rock.Mobile.UI which is our platform abstracted UI system.

### App.Shared.RockApi

Contains all the code related to communicating with Rock. It defines all end points and models needed for pulling down and sending up data to Rock.

### App.Shared.UI

As Rock.Mobile.UI matured, it became possible to build full UI pages that are then shared between Android and iOS. The advantage to this is only having to implement and maintain one UI page for both platforms.

While there are only a handful of UI pages written like this, the goal is to build all future UIs on Rock.Mobile.UI and get away from implementing them in the platform specific **Tasks** section.

### App.Shared.Analytics

This defines the Analytics tracking used by Mobile App. If you want to add more analytics, this is the place to do it. It currently uses **Localytics** as the analytics engine, but our Analytics class was designed in an abstract way such that there is only one actual call to Localytics, and it should be fairly easy to replace with something else.

### Config.cs

This defines all of the configuration settings that someone might want to adjust in order to customize the app. There should be no hard-coded values throughout Mobile App. If there are, it’s a mistake, and they should either be in **Config.cs** or **PrivateConfig.cs**.

### ConnectLink

To make it easy to add more options for the Connect section of the app, there are ConnectLinks. Defined in an array within **Config.cs**, these are read by the Connect task and used to populate the list of Connect options.

### FacebookManager

Provides encapsulated functionality for parsing Facebook responses. Basically works to consolidate Facebook code into one place.

### GroupFinder

Because getting a group is a multi-step process, this class uses RockApi to receive Groups using the correct API calls. Additionally it parses responses and handles error checking, so that the process is opaque to users working at the UI level.

### PrivateConfig.cs

Works exactly like Config.cs, except denotes settings that a general user will probably *not* want to change. You can, but it requires a good understanding of Mobile App’s design.

### Strings.cs

Centralizes ALL STRINGS displayed to a user. Do not ever hardcode a string in the app. It should go here. This allows quick customization for all terminology and branding in the app. Additionally, it lays the groundwork for multi-language support.

### VimeoManager

Not currently used, but parses metadata associated with a Vimeo video. Was used to get thumbnails of each video to display in the **Messages** section.

### Additional Resources

For details regarding Rock.Mobile, see:

[Rock.Mobile\_Documentation](../../../Rock.Mobile/Rock.Mobile_Documentation.docx)

[Rock.Mobile.UI\_Documentation](../../../Rock.Mobile/Rock.Mobile.UI_Documentation.docx)

For details regarding Notes, see:

[Notes - Getting Started](NoteScript/Notes%20-%20Getting%20Started.docx)

[Notes - Database](NoteScript/Notes%20-%20Database.docx)

[Notes - NoteScript](NoteScript/Notes%20-%20NoteScript.docx)