Cross-Platform Development

CS 4720 – Mobile Application Development
Which Platform?

• We’ve discussed this!
• Where are the users you want to target?
• How much do you plan to charge?
• How much do you want to invest?
• Where do people expect a premium experience?
• What sort of payment structure do you want to use?
Which Platform?

• Some thoughts:
  – Are you trying to keep down costs for developing for both platforms?
  – Do you have one development team?
  – Who will be maintaining this system?
Which Platform?

• The answer to this could end up being
  – We want to reach everyone!
  – We want it as cheap as possible!
  – We want it easy to maintain!

• The old adage – Cheap, Quick, Good: Pick 2

• Cross-platform development used to definitely be “cheap” and “quick”... and definitely not “good”
Early Cross-Platform

- Early cross-platform development wasn’t very good at all
  - Poor user interface
  - Poor performance
  - Overly-large files/binaries
  - Platform holders weren’t a big fan of it
Cross-Platform Development Now

• Things have improved!
• You can now:
  – Create custom, native UIs for each platform
  – Use existing toolchains (compiling with VS, Android Studio, or Xcode, for instance)
  – Binary sizes have diminished (extra steps added in compile process to remove excess code)
  – Tools are easier to use
Cross-Platform Development Now

• Cross-platform development comes in two flavors
  – Build with an API/toolchain that exposes the native SDK of the platform for a single programming language/system (i.e. C# with Xamarin)
  – Build a hybrid HTML5 web app that can execute as an app on the device (i.e. Apache Cordova)
SDK API - Xamarin

https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_0__overview/
SDK API - Xamarin

• Uses C#
• Build on Mono Framework
• Build native UIs
• Use custom plugins to get access to device and platform-specific features
• iOS: compiled directly to native ARM assembly
  – Mac is still needed to push to device or store
• Android: compiled to run on MonoVM
SDK API - Xamarin

```csharp
namespace CrossOne
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();
            MainPage = new CrossOne.MainPage();
        }

        protected override void OnStart()
        {
            // Handle when your app starts
        }

        protected override void OnSleep()
        {
            // Handle when your app sleeps
        }

        protected override void OnResume()
        {
            // Handle when your app resumes
        }
    }
}
```
SDK API - Xamarin

```csharp
namespace CrossOne.Droid
{
    [Activity (Label = "CrossOne", Icon = "@drawable/icon", Theme="@style/MainTheme", MainLauncher = true)]
    public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        protected override void OnCreate (Bundle bundle)
        {
            TabLayoutResource = Resource.Layout.Tabbar;
            ToolbarResource = Resource.Layout.Toolbar;

            base.OnCreate (bundle);

            global::Xamarin.Forms.Forms.Init (this, bundle);
            LoadApplication (new CrossOne.App ());
        }
    }
}
```
namespace CrossOne.iOS
{
    // The UIApplicationDelegate for the application. This class is responsible for launching the
    // User Interface of the application, as well as listening (and optionally responding) to
    // application events from iOS.
    [Register("AppDelegate")]
    public partial class AppDelegate : global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
    {
        // This method is invoked when the application has loaded and is ready to run. In this
        // method you should instantiate the window, load the UI into it and then make the window
        // visible.
        //
        // You have 17 seconds to return from this method, or iOS will terminate your application.
        //
        public override bool FinishedLaunching(UIApplication app, NSDictionary options)
        {
            global::Xamarin.Forms.Forms.Init();
            LoadApplication (new CrossOne.App ());

            return base.FinishedLaunching (app, options);
        }
    }
}
HTML5 Hybrid - Cordova

• Uses HTML5 – HTML, CSS, and JavaScript
• Runs in a custom app container for each platform
• Plugins are available that allow you to access device specific features using JavaScript (jQuery)
• Rendering HTML5 like this can use more processing/power than native
HTML5 Hybrid - Cordova
HTML5 Hybrid - Cordova

• Example project!
Things to Consider

- With some planning, 50-80% of the code base can be reused
  - User interface code is the least reusable for SDK API systems
  - Plugins and platform-native code for HTML5
- Maintenance can be easier if problem is in the business logic and not UI
- Testing business logic can be easier / more centralized

https://www.infoq.com/articles/mobile-cross-platform-app-development
Things to Consider

• If you’re team already knows/uses C# or HTML5, then your costs may be reduced

• UI design, particularly to follow human interface guidelines for the platform, gets significantly harder

• Platforms are constantly changing and these toolsets may not keep up

• Performance is still an issue

• Cost vs. return on investment is the question