Intel SWPC-Intel® XDK Lab Script

v1.01: 10.08.2013

# Purpose

The purpose of this document is to describe the tutorial on “Developing a simple mobile app using Intel® XDK” for the lab session at SWPC.

# An Introduction to the Intel® XDK

Please follow the presentation for an introduction to the Intel XDK

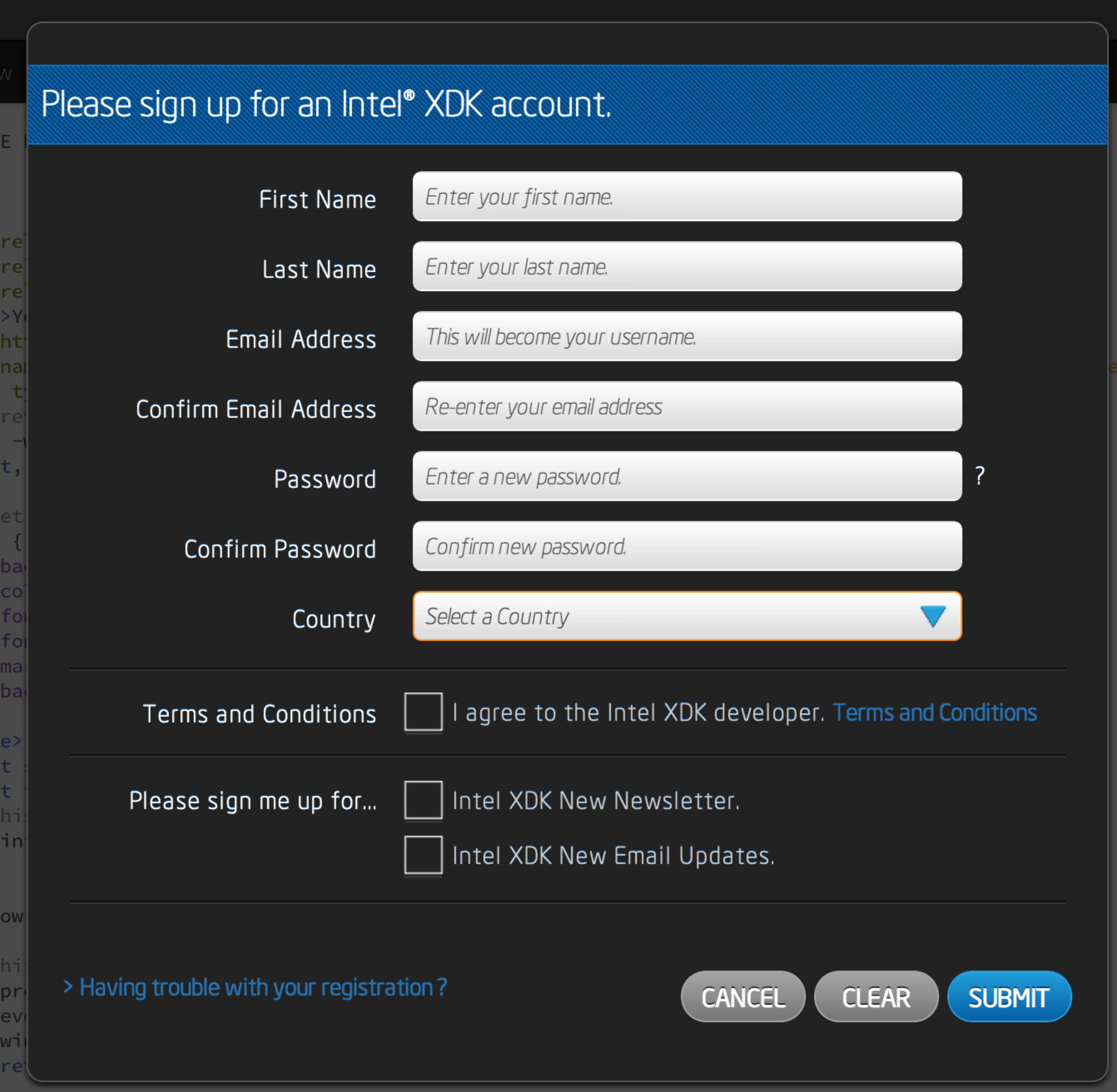
# Hands on lab activities

In this lab we will learn end to end development of a simple mobile app using Intel XDK. The goal of this lab is to explore some device features and build a hybrid app that utilizes simple features like beep, vibrate and playing sound. This activity will provide you hands-on experience with designing the UI using App Designer, programming using Intel XDK APIs to access native features of the device and using the emulator to emulate different devices. You will also learn how to preview your app on the device and build an app package that runs on device.

## Pre-lab activities

### Sign up for an account

* Open XDK New by double clicking on the icon
* Sign up for an account, filling in the necessary details



* Then log in with the credentials you created.

### Download App lab on your phone

* If you have either an iPhone or an Android device, search for the free “App Lab” app in the store to download.
* This app will be used later in the lab to preview your app on the device.

# Develop a simple “My Phone can...” app

Our goal is to develop an app that looks like this:



For simplicity we will divide the app development into 5 parts.

Create the UI

Add functionality

Preview the app in Emulator

Preview the app on device

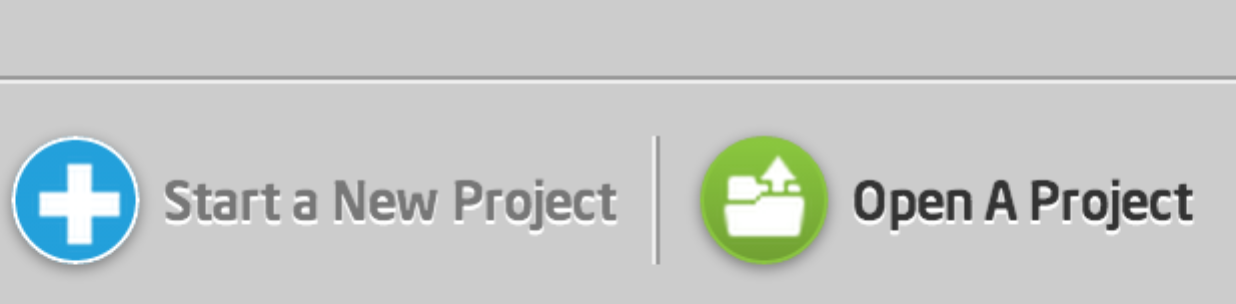
Build a package for app store

**4.1 Create the UI**

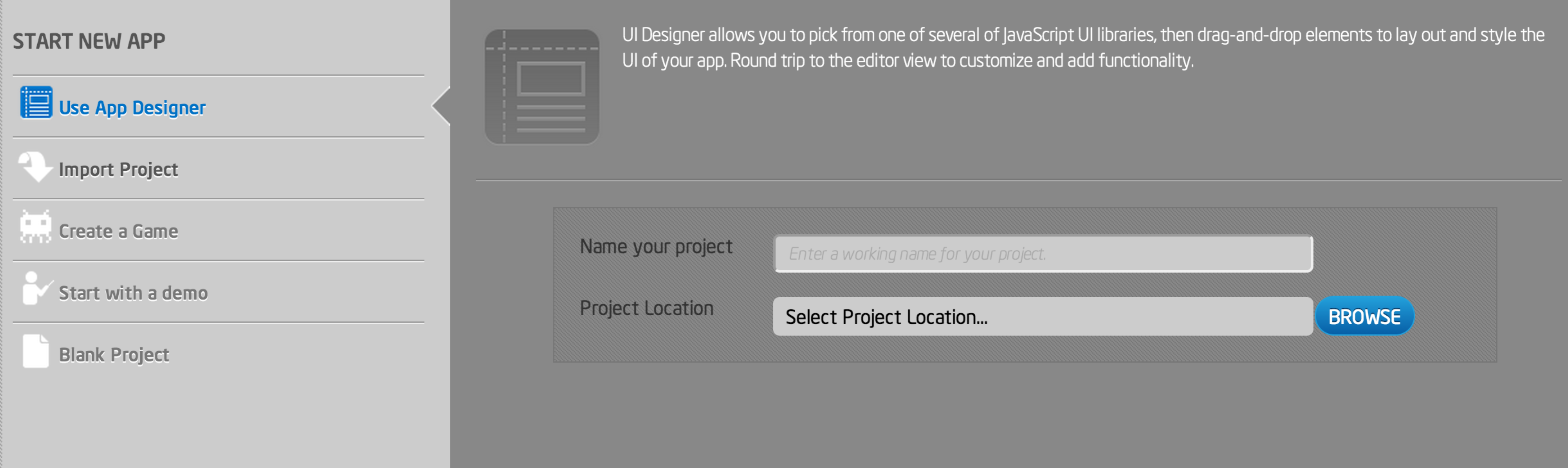
### To build UI that is responsive across multiple devices and form factors, we use libraries or frameworks like jQuery Mobile, Twitter Bootstrap or App Framework. To make design phase easier Intel XDK has a built-in tool called App Designer, which lets you drag and drop components for your UI. You can choose one of the three libraries mentioned above and App Designer will generate code using that library for your UI. For this lab we will build the UI with App Designer using App Framework. App Framework is Intel’s JavaScript library that is specifically designed for mobile UIs.

### Start a new Project

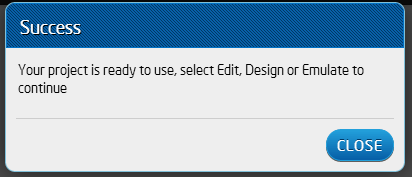
* Open XDK New by clicking on the icon.
* Under the Projects tab click on the Start a New Project to create a new project



* Select Use App Designer for this project.
* Fill in the name of your project and select the folder where you want this project to be saved.

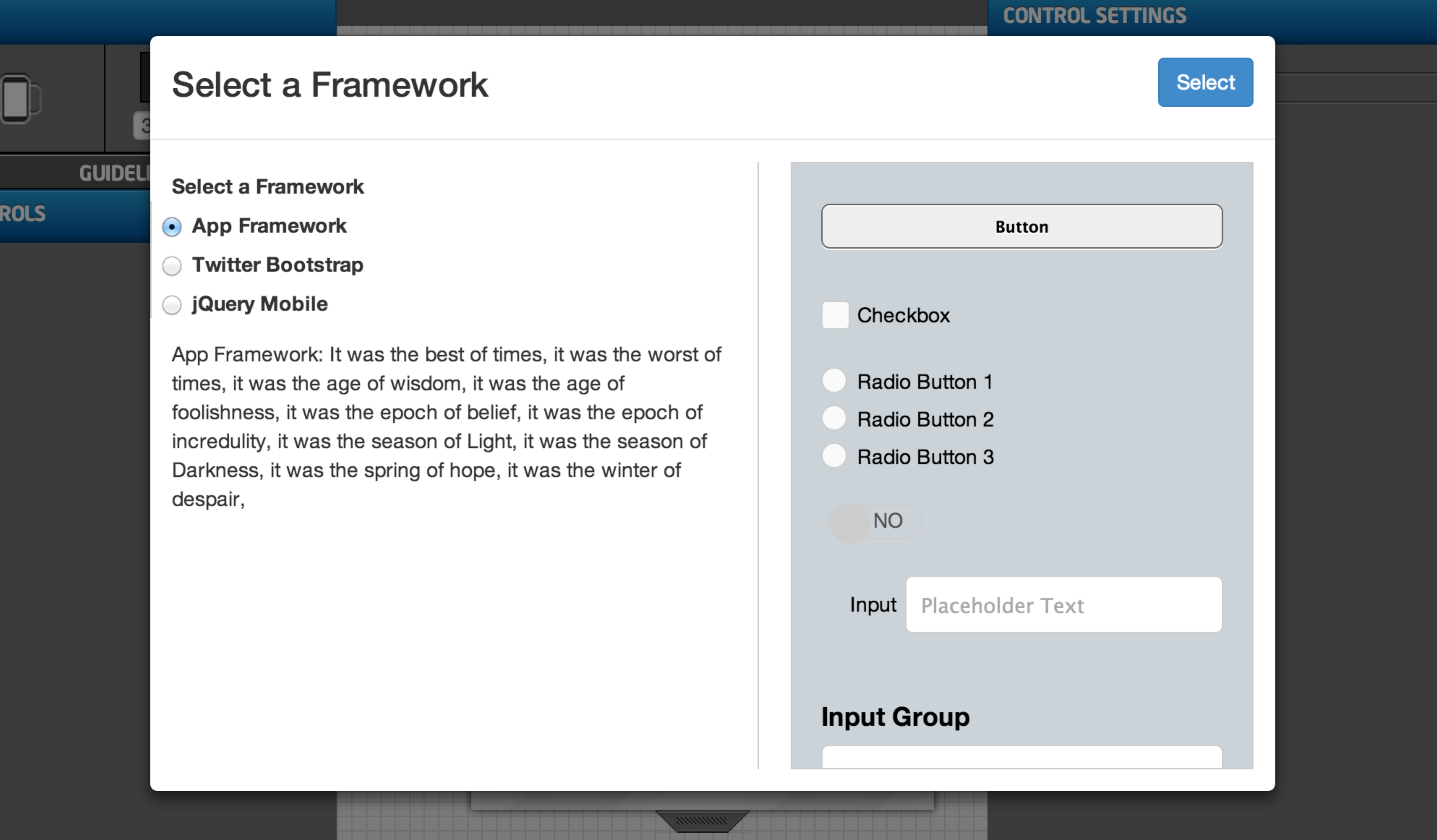


* Create the project. When you see the following message box, you will have a project created in your folder with default files Index.html and project.xdk

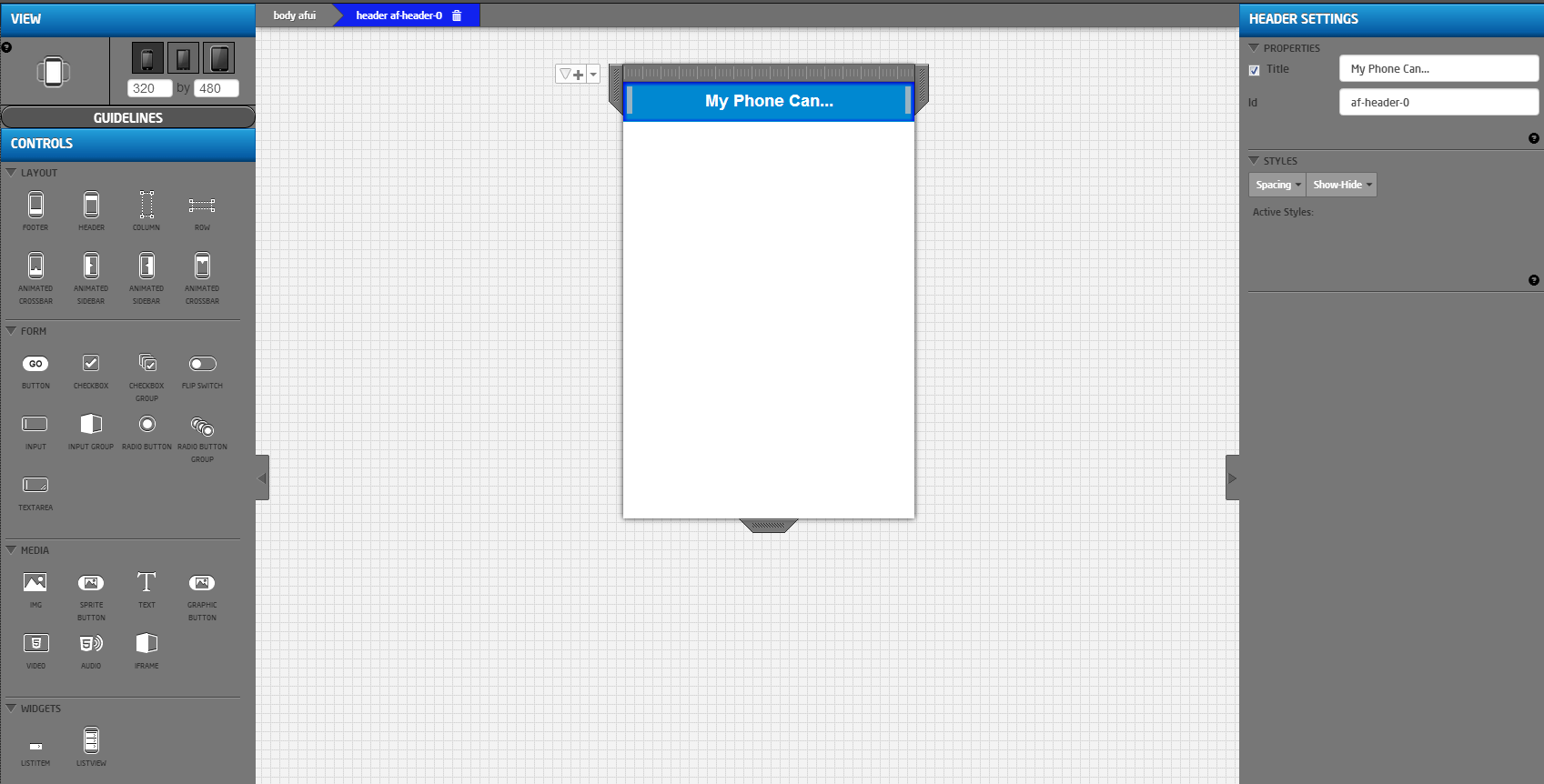


### Using App Designer

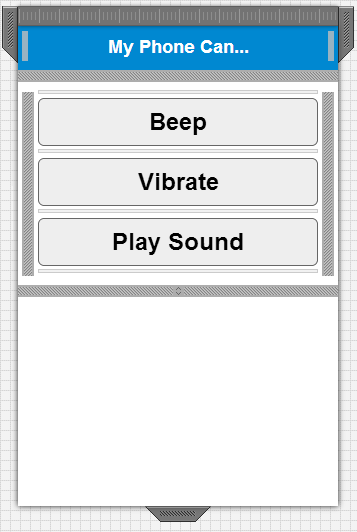
* Click on index.html in the file manager and click on App Designer tab, this will open up App Designer for you to design your UI.
* Select App Framework to start with.



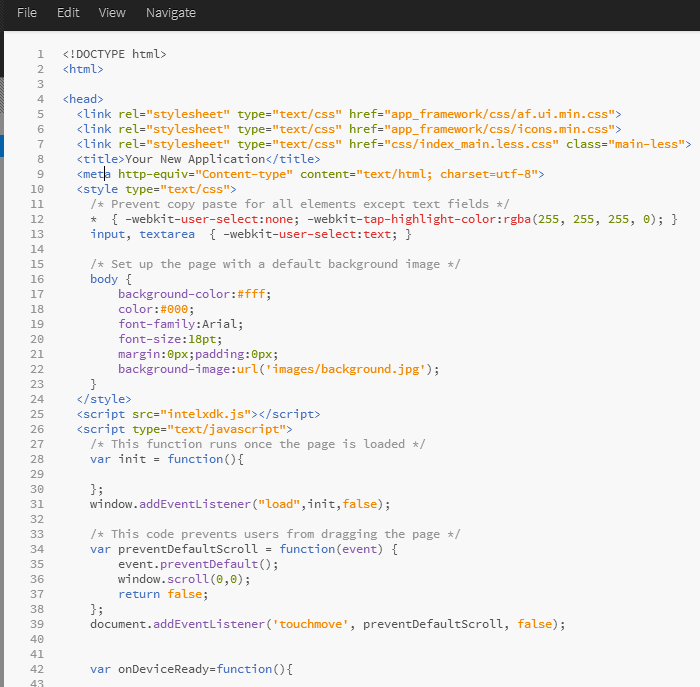
* From the controls panel drag and drop a header component into the design panel.
* Add a title by selecting the Title on the Properties menu to the right.



* Drag and drop a button. Change the text of label under the properties to “Beep”
* Similarly add two more buttons for “Vibrate” and “Play Sound”



* Press Ctrl S to save any UI changes
* View the source using Brackets Editor



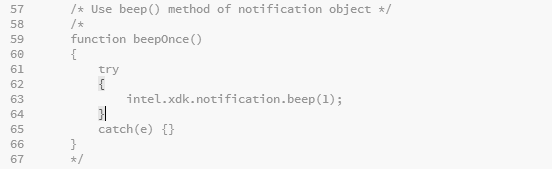
## Add functionality

For the sake of consistency and completeness, please [download](https://github.com/xdklab/sample-notifiers) the project from github that has been created beforehand. It will provide a skeleton for you to insert the functionality at proper places and it also has all the assets necessary for this project.

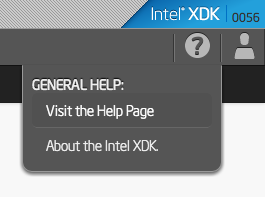
Unzip the project and open the project using “Open A Project” icon under Projects tab .

### Open and Edit project

* Select the MyPhoneCan project in the Projects tab
* This is the same UI as we created earlier, but with some extra skeleton for adding the functionality.
* Uncomment code for beep.



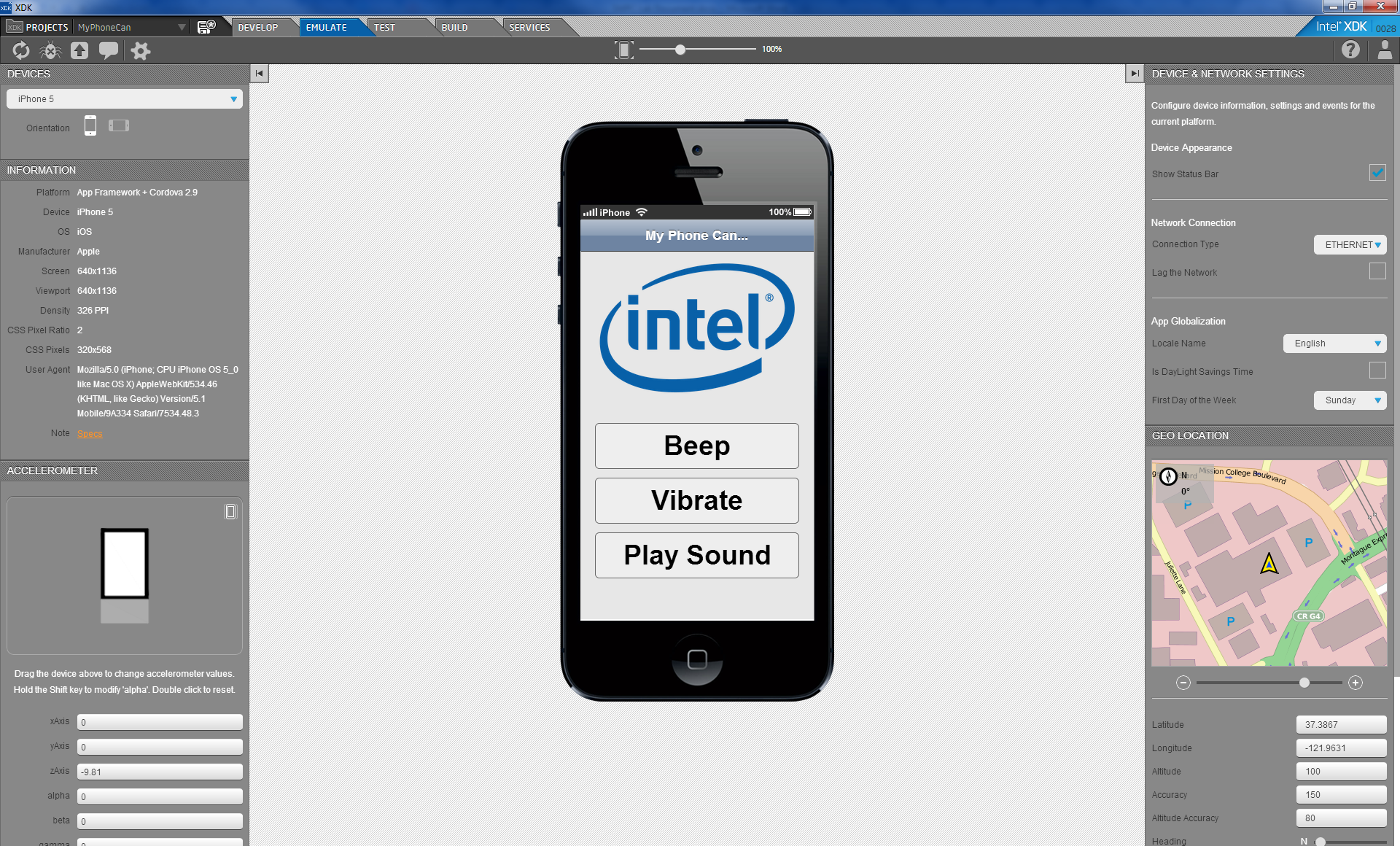
* Use emulator to emulate the beep functionality.
* Lookup API for vibrate in the documentation by clicking “?” at the top right of XDK.



* Add code for vibrate on your own
* Add code for playing sound on your own. APIs will be explained.
* Solution files on github (link and QR code will be given)

## Preview in Emulator

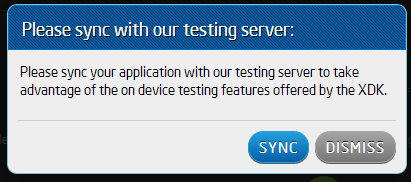
* View the project in the emulator.
* Change devices to see how the app looks on different devices



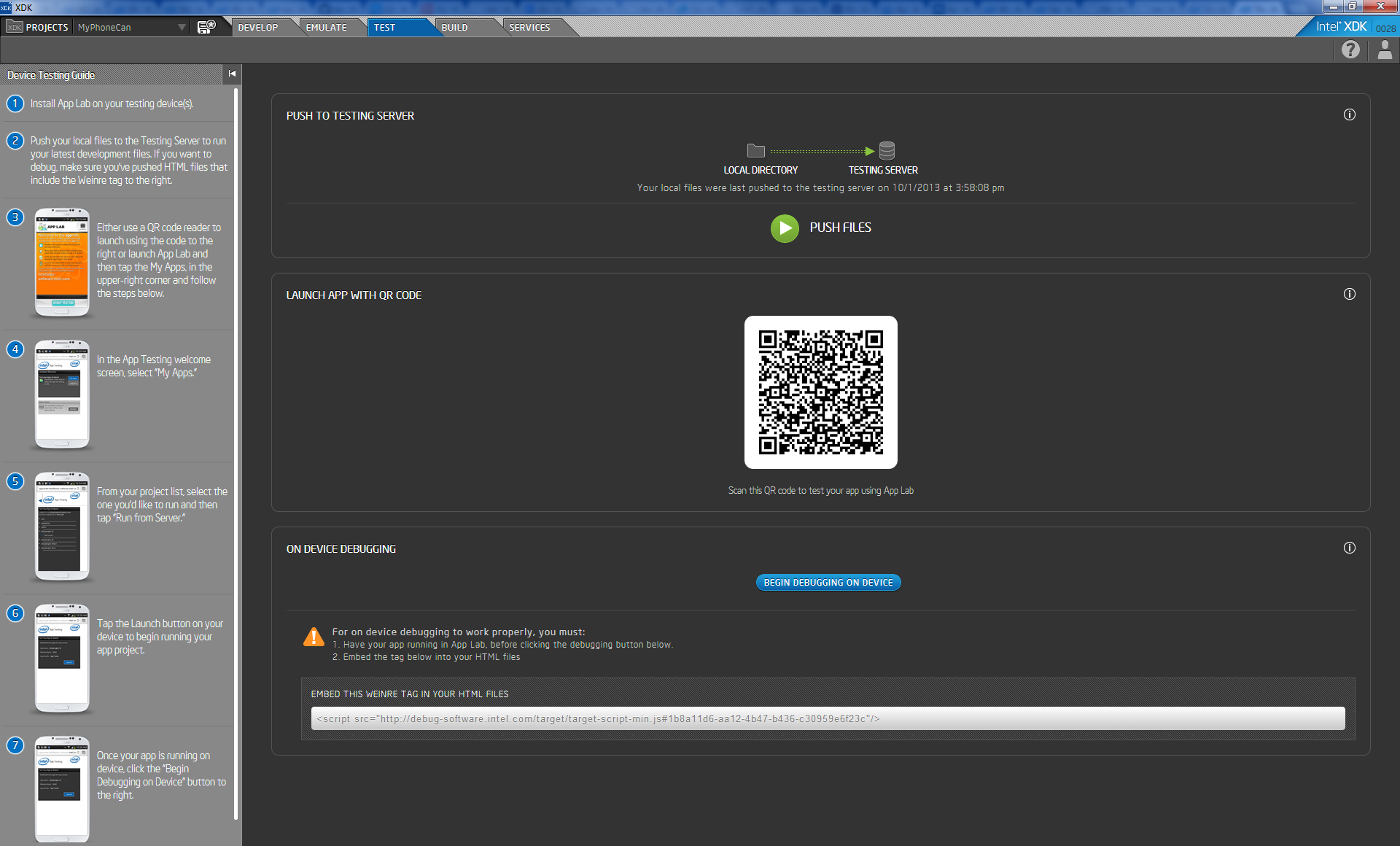
## Preview on Device

To preview your app on your device:

* Sync the project to the cloud by navigating to the Test Tab.

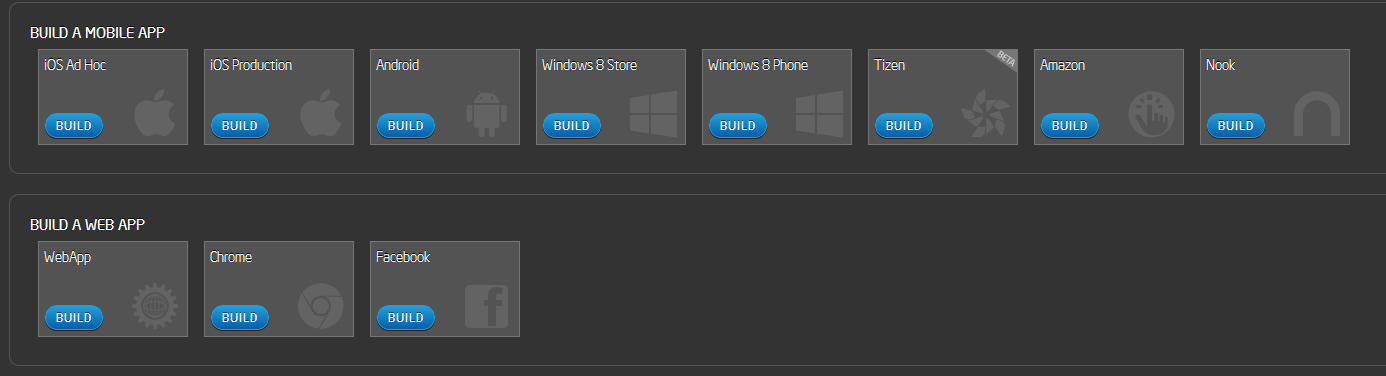


* On your phone log into the App Lab app with your XDK credentials and follow the steps on the Test tab.
* Select the app to preview
* Alternatively you can scan the QR code displayed in the Test tab with your phone and it will launch the app on your phone.

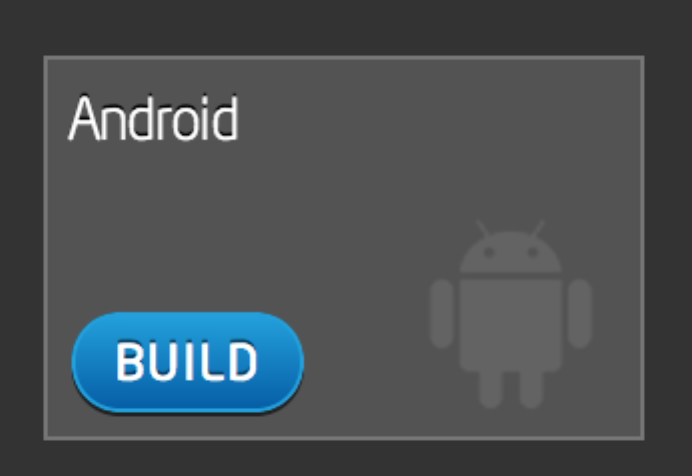


## Build a Package

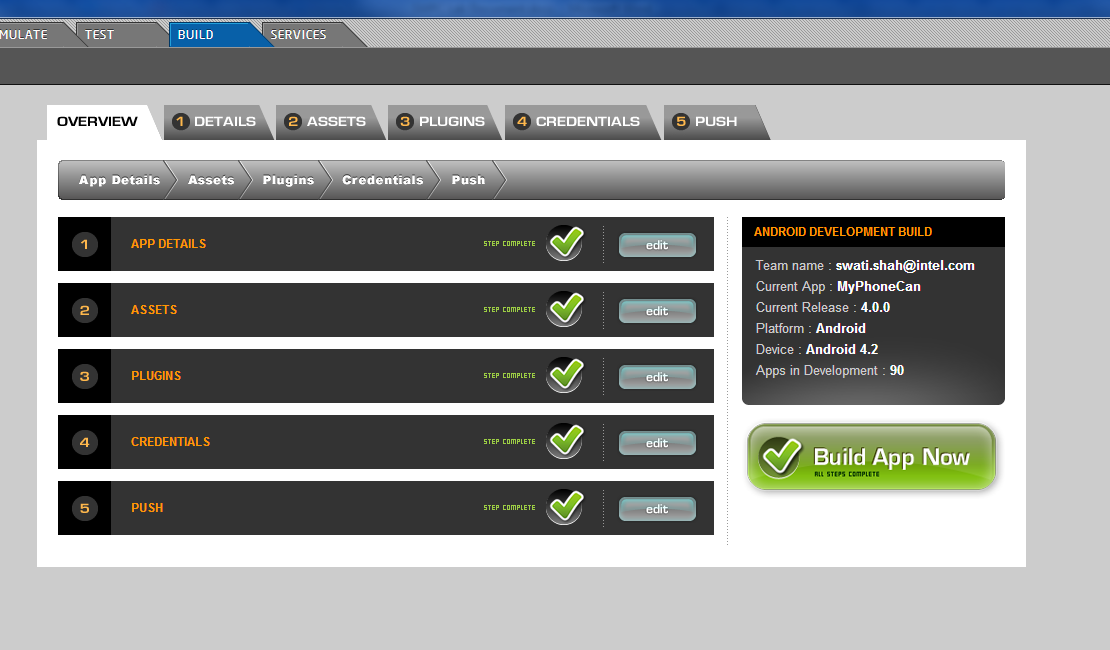
* Go to the Build tab



* Build an example Android\* app by selecting Android button.



* Follow the steps.



* After the build is complete, you will receive an email pointing to the download link for your built app.
* Download the app and install it.

# Task 2: “Rolling Can” accelerometer app

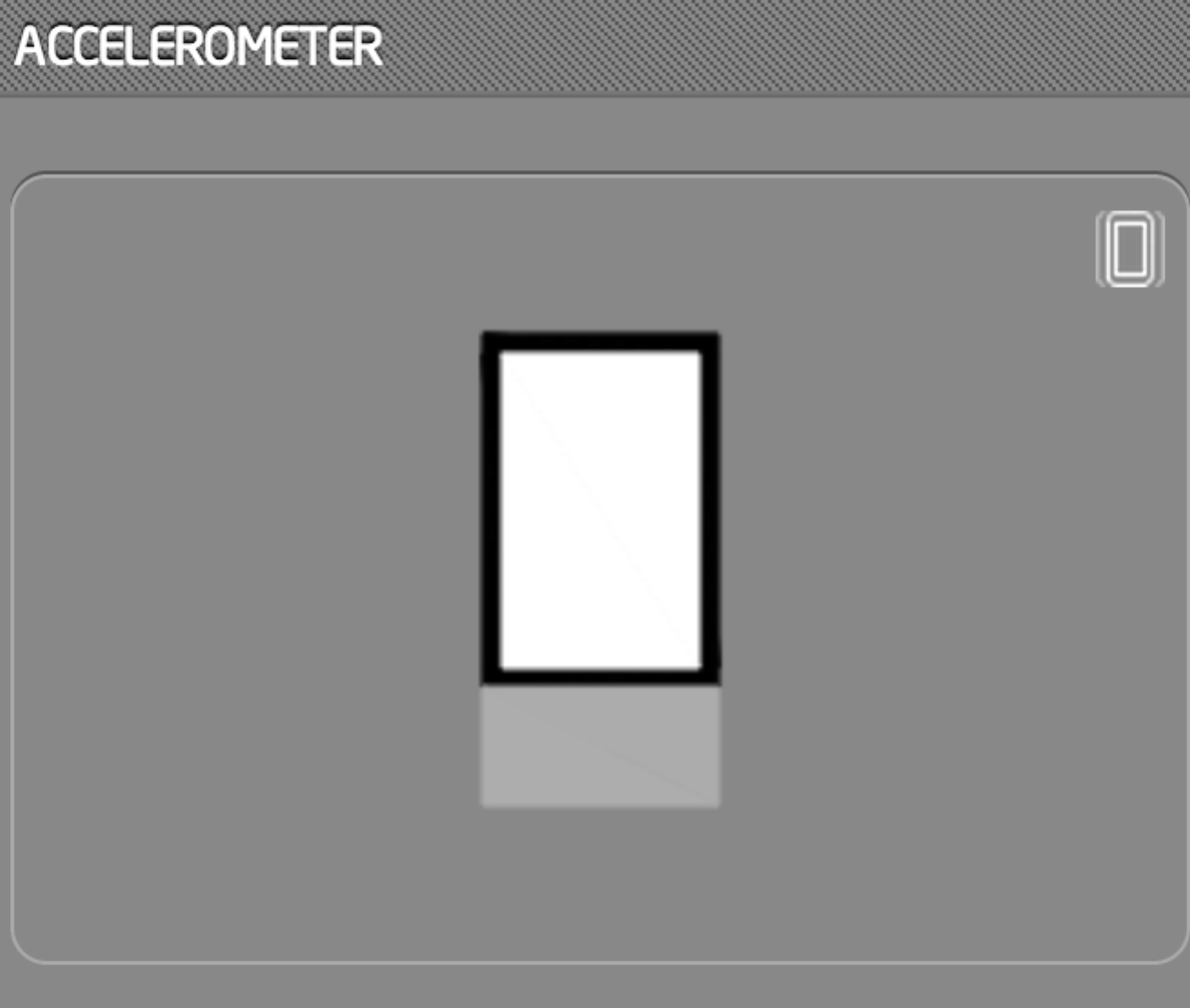
This part of the lab will show you how to start with an existing demo available in the XDK New. We will explore the accelerometer API using this example.

## Start with a demo

* In the Projects tab, select “Start a new project”
* Select “Start with a demo” and select “Rolling Can” demo

## Preview in Emulator

* View the project in the emulator.
* Change devices to see how the app looks on different devices



* Drag around the device in the accelerometer tab in emulator to emulate accelerometer



## Make changes in the code

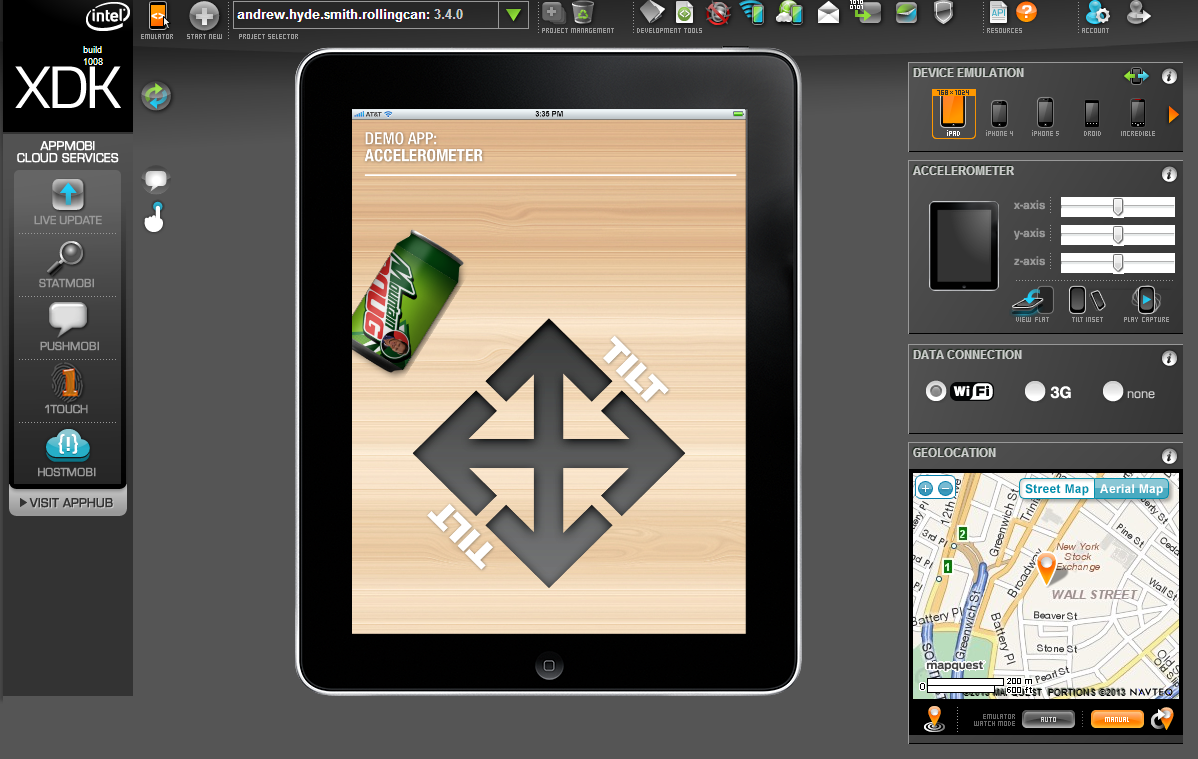
* Uncomment the part of code in the index.html. You can find this piece of the code in lines 349-361. Uncomment the part with can\_doug.png both at 351 & 360



* And comment the code above it – the one with can.png at lines 349 and 358. It should change to look like:



* Preview the changes in Emulator by following instructions from 4.2



* You can notice the can image in the demo has changed

## Make further changes to code

* Notice the intel.xdk.watchAcceleration takes in an option opt.
* Play around with increasing the opt.frequency. And preview the changes as visible in the emulator.

## 

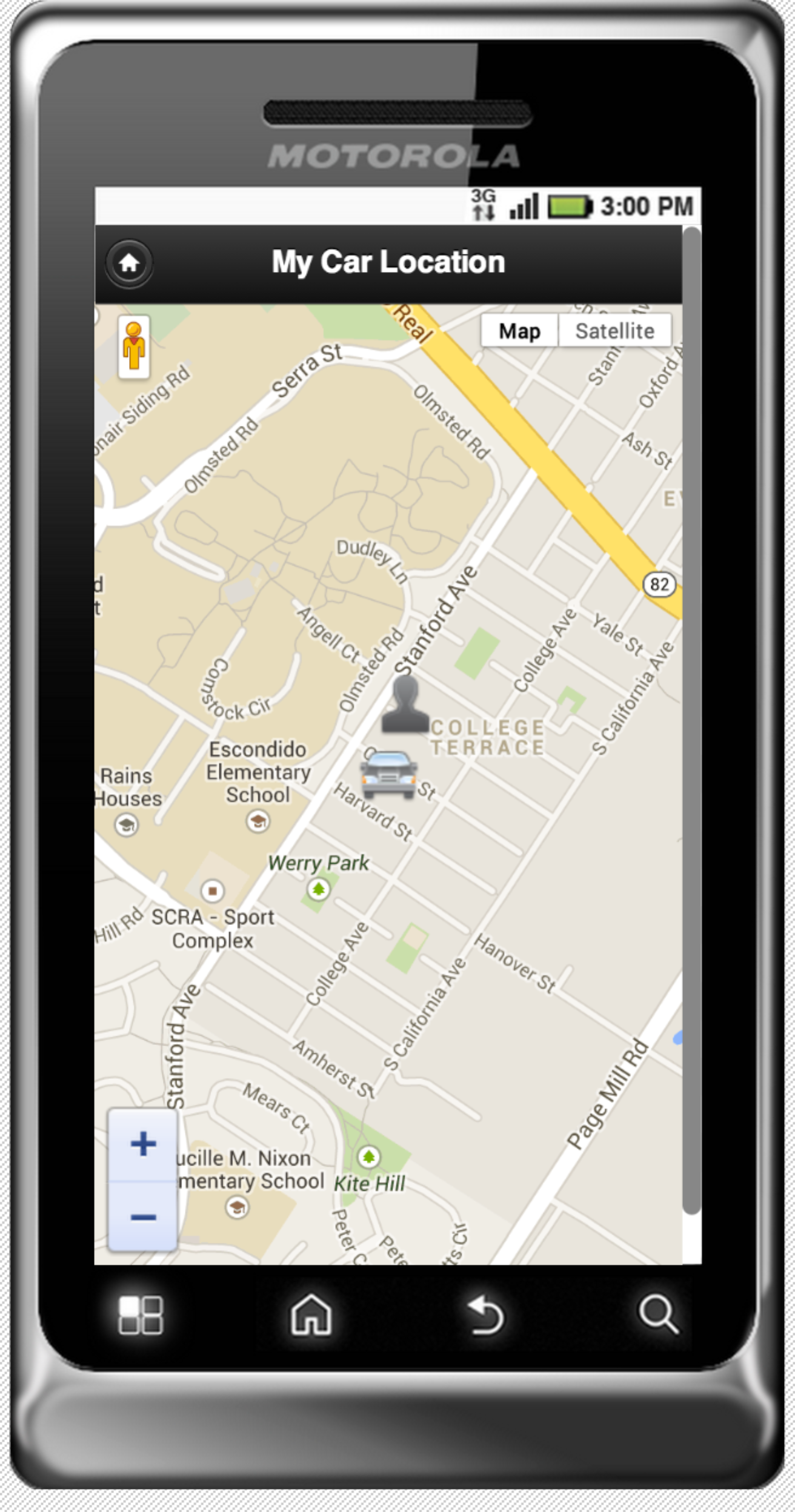
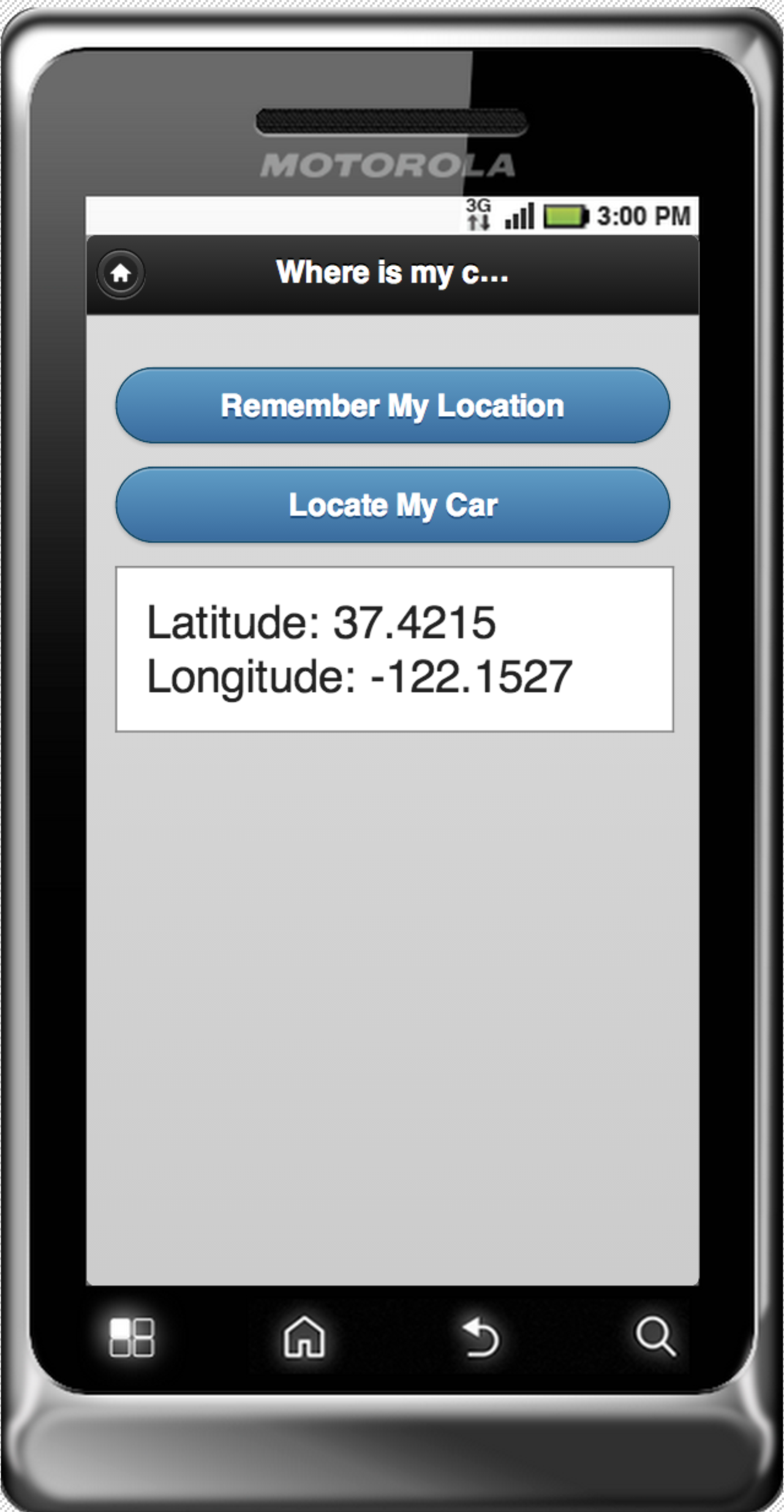
## Preview in App Device

* Follow the instructions in 4.4

## Build a Package

* Follow the instructions in 4.5

# (Additional) Task 3: Develop “Where is my Car?” app



This part of the lab is focused on working with Cordova APIs.

Please [download](https://github.com/xdklab/sample-where-is-my-car) the basic skeleton for this app from github.

As part of the lab, you will fill in several missing parts of the app, which involve understanding how to work with some basic Javascript. You will make use of the tools you used in the previous task to test and debug as you develop.

This app will use Cordova’s geolocation APIs to access your phone’s GPS to remember where you parked your car.

## Open “Where is my car” project

* The project already has the UI necessary for the app. There are two buttons as you open the app.
* There are two pages “homePage” and “mapPage”
* There are two buttons
* “Remember My Location” or “curLoc” saves the current location. We make use of HTML5’s localStorage to save the location persistently. This enables us to save the location even when we close and reopen the app afresh.
* “Locate my Car” or “lastLoc” will open a map, showing the position of the car using the location we saved using “curLoc” (localStorage.lat & localStorage.long) and the user’s current location. The user’s current location updates as the user moves around.
* We use the Google Maps API to embed the map and add the markers.

## Fill in the “TODO” items.

There are several blanks in the project corresponding to cordova geolocation APIs. Refer to [Cordova’s documentation](http://cordova.apache.org/docs/en/2.9.0/cordova_geolocation_geolocation.md.html#Geolocation) for the details of the API.

Look for TODO and follow through in the order suggested. FILL\_IN is the parts where you will have to fill in the required APIs or inputs as necessary. To begin with the app does not load

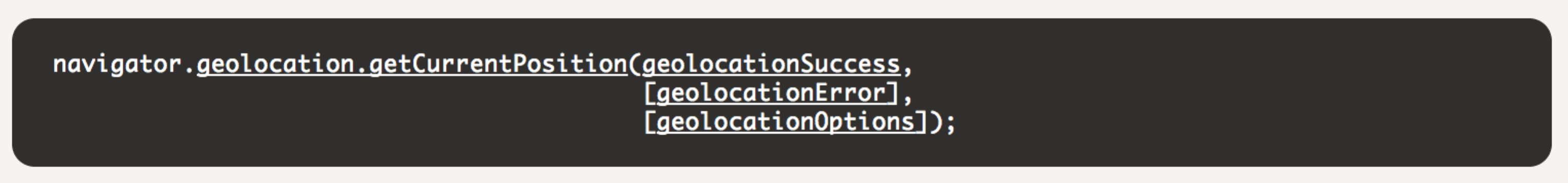
TODO 1: deviceready is an essential to all Cordova applications. It signals that all the Cordova APIs are loaded and the app is ready to use. [Refer to the document](http://cordova.apache.org/docs/en/2.9.0/cordova_events_events.md.html#deviceready) for further details. Fill in the right event in place of FILL\_IN. You will be able to see the home page with two buttons now.

TODO 2: Fill in the Cordova API for getting the current location. [Refer to the document](http://cordova.apache.org/docs/en/2.9.0/cordova_geolocation_geolocation.md.html) for exact details.

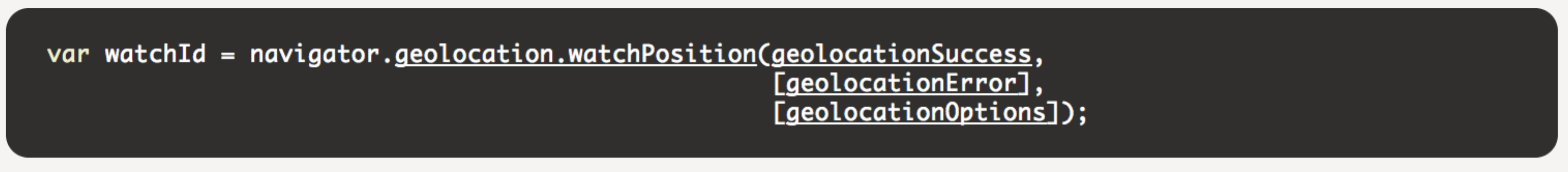
TODO 3: We need an API that updates as the user location changes. While similar to previous API call, this requires an API that watches for user location changes. Imagine moving around with your phone trying to locate your car, this API should watch for user changes and be invoked every time it changes.

Note: There are two different Cordova APIs we primarily use.

* [getCurrentLocation – gets the current location](http://cordova.apache.org/docs/en/2.9.0/cordova_geolocation_geolocation.md.html#geolocation.getCurrentPosition)



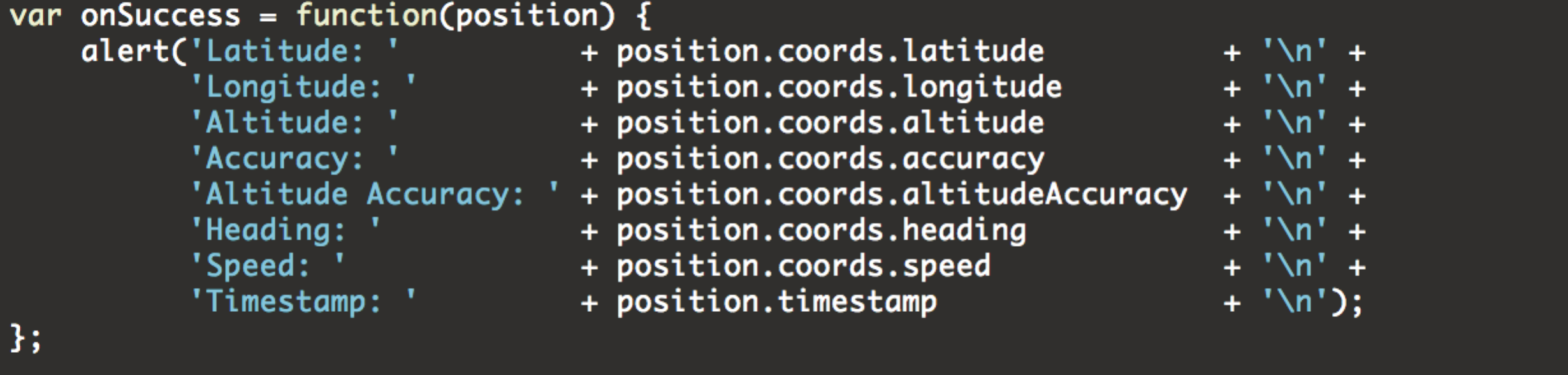
* [watchPosition – watches for changes to user location and is called](http://cordova.apache.org/docs/en/2.9.0/cordova_geolocation_geolocation.md.html#geolocation.watchPosition)



TODO 4: Uncomment the addPin function call and fill in the latitude and longitude values with the car’s latitude and longitude values. This change adds a car marker on the map corresponding to the car location.

TODO 5: Uncomment the addPin function call and fill in the latitude and longitude values with the user’s current location’s latitude and longitude values. This change adds a user marker on the map corresponding to the car location.

Note: This is called within the success handler of the Cordova API call. So, you will need to find the latitude and longitude based on the “position” variable.



## Preview in emulator

Verify all the code changes with emulator. You can emulate geolocation APIs in the XDK in the emulator.

Follow instructions in 4.3

Note: You can also add alert or console.log messages to print out debug messages. See examples in the project

## Preview on Device

Follow the instructions in 4.4

## Build a package

Follow the instructions in 4.5

Note: You will need to set the right permission in the build UI. Open the Details tab in the Build page to set permissions for geolocation access.