Bankrevels – Environment Setup Guide

Contents

[Introduction 3](#_Toc373710314)

[Windows 3](#_Toc373710315)

[IBS & RIB 3](#_Toc373710316)

[Requirements 3](#_Toc373710317)

[Steps 3](#_Toc373710318)

[Building the WAR file 4](#_Toc373710319)

[Deploying to deployment Tomcat 4](#_Toc373710320)

[Starting up to IBS 4](#_Toc373710321)

[Shutting down the IBS and undeploying the WAR file 4](#_Toc373710322)

[RMB 4](#_Toc373710323)

[Requirements 4](#_Toc373710324)

[Steps 5](#_Toc373710325)

[Compiling for Deployment 6](#_Toc373710326)

[OS X 6](#_Toc373710327)

[RMB 6](#_Toc373710328)

[Compiling for iOS 10](#_Toc373710329)

# Introduction

This guide is to assist those who are attempting those who are looking to continue work on our FYP project. We will go through step by step how to set up the environment on both Windows and OS X (Titanium only) to that you will be able to build your own versions of the application.

# Windows

Our team has developed most of the application on the Windows platform. Moreover, since the version of TIBCO ActiveSpaces we are using is for Windows, the IBS can only be built on this platform. For our Android app, we also use the Windows platform to compile it.

## IBS & RIB

### Requirements

1. Java 6 x86
2. TIBCO ActiveSpaces x86
3. Eclipse IDE for Java EE Developers x86
4. Apache Tomcat 7 (zip or x86 version, but I recommend the zip version as it works with both x86 and x64)
5. Google Chrome
6. \*All binaries have to be the 32 bit version so as to emulate the environment on the deployment server

### Steps

1. Install the latest Java 6 x86 code
2. Run and install the TIBCO ActiveSpaces x86 installer
3. Extract the Eclipse IDE zip folder into a folder of your choice
4. Extract or install the Tomcat 7 into your root drive folder
5. Next, we have to set up the environment variables for ActiveSpaces
   1. Environment variables can be accessed from Start > My Computer (right click > Properties) > Advance System Settings > Environment Variables button at the bottom right corner.
6. Under User variables for <username> create the following variables (assuming default configuration during installation)
   1. AS\_HOME – C:\tibco\as\2.0
   2. JAVA\_HOME - C:\Program Files (x86)\Java\jdk1.6.0\_45 <- *or your version number*
7. Add the following directories to your Path variable under System Variables
   1. *C*:\tibco\as\2.0\lib;C:\tibco\as\2.0\bin;
   2. C:\Program Files (x86)\Java\jdk1.6.0\_45\bin; (If you don't already have Java in your path variable)
8. Next, we need to setup Tomcat to load the ActiveSpaces JAR and DLL into the shared classloader.
   1. Copy C:\tibco\as\2.0\lib\as-common.jar to <your tomcat directory>\lib
   2. Copy C:\tibco\as\2.0\lib\ as-common.dll to <your tomcat directory>\bin
9. Now run eclipse.exe in the Eclipse folder to start the IDE and close the Welcome page
10. Right click the left panel called Project Explorer and click Import > Import...
    1. Select General > Existing Projects into Workspace and Next
    2. Under Select root directory, browse to the IBS root directory and click ok
       1. The root directory for the IBS is SMUtBank\_IBS. It can be found under our SVN > App > IBBackend > Server
    3. Click Finish
11. Once the project folder has appeared on the Project Explorer, select the SMUtBank\_IBS folder icon and click Run As > Run on Server
    1. Under Select the server type, choose Apache > Tomcat v7.0 Server and check the Always use this server when running this project, then click Next
    2. Under Tomcat installation directory, click browse and select the folder in which you extracted the Tomcat to earlier on
    3. If you have multiple JREs on your computer, this is the time to set JRE6 32bit to the server, otherwise the Workbench default JRE will do and click Finish
12. Now Windows Firewall will appear, with Eclipse and Java asking for permission to access the network, allow access for both.
13. Go to the Server tab, on the bottom panel, 3rd from the left and click the red stop button on the top right of the panel. Then click the green play button to restart the server.
14. Now go to the following link in your Chrome: http://localhost:8080/SMUtBank\_IBS/ and you should see the IBS Testing page.
15. \*Important Note\* Whenever changes are made outside of Eclipse, E.g. you manually add a file to the project folder or add a lib to the tomcat server, you need to click the project folder under the Project Explorer and hit F5 to force a refresh and restart the Tomcat server.

### Building the WAR file

To build the WAR file for Tomcat deployment, right click the project folder under Project Explorer and click Export > WAR file. Optimize the runtime for Apache Tomcat 7 and select a destination to export the WAR file. Then click Finish.

### Deploying to deployment Tomcat

Open a browser to the Tomcat app manager, usually at <http://10.0.106.169:8080/manager/html/>. Username and password is tomcat. Assuming that the project is already undeployed, scroll down to WAR file to deploy > click choose file and select the WAR file to deploy. Then click Deploy.

### Starting up to IBS

To startup the IBS after deploying, go to the app test page and click Startup IBS. Once startup has been completed, it will return the time it took at the top. It is not compulsory to startup the server before use as the first user which accesses the server will auto trigger the startup code, but it will allow the first user to access the IBS a faster response time.

### Shutting down the IBS and undeploying the WAR file

It is important to shutdown the IBS before undeploying. Otherwise, you will not be able to undeploy the WAR file from the tomcat web app manager and will have to manually remove it from the tomcat server webapps folder after force closing the tomcat server.

In order to undeploy safely, go to the IBS test page and click Shutdown IBS for redeployment. At this point of time, it is important to ensure that no other users attempt to access the server, otherwise it will restart again. After that, go to the tomcat web app manager and and click the number in the sessions column for SMUtBank\_IBS. Check all the boxes in the list and click Invalidate selected Sessions. Then return to the main page. Then click stop. Count to 5, then click undeploy. If you are lucky, it will undeploy. Otherwise, try clicking the undeploy button at once per second. If the button becomes disabled, refresh the page, and continue trying to undeploy. If after 30 seconds of undeploying and it still refuses to budge, try clicking Start, followed by undeploy. Usually, that will solve the problem. If it still does not work, you will have to remove the webapp manually from the tomcat server.

## RMB

### Requirements

1. Java 6 x86
2. Appcelerator Titanium IDE
3. An Appcelerator Titanium Account
4. \*Titanium only supports 32bit Java 6

### Steps

1. After downloading and creating an account on the Appcelerator website, install the Titanium executable.
2. After installation, launch Titanium and login with your Titanium Account.
3. Once the Dashboard loads, click Getting Started > Configure Native SDKs, select Android SDK and click the button Install or Update Android SDK.
   1. You may also wish to follow the Titanium guide here: http://docs.appcelerator.com/titanium/latest/#!/guide/Quick\_Start-section-29004949\_QuickStart-LaunchingTitaniumStudio
   2. Also of importance is the Titanium Compatibility Matrix listed here: http://docs.appcelerator.com/titanium/latest/#!/guide/Titanium\_Compatibility\_Matrix-section-29004837\_TitaniumCompatibilityMatrix-Xcode
   3. Depending on which version of Titanium SDK you are running, there are minimum and maximum version for both Android and XCode SDKs
   4. You will need to do a full update of Titanium and the SDK before moving on to the next step. Click Next for any popups.
   5. Remember to allow the IDE through the firewall if it so requests
   6. You should also switch the Titanium UI Studio to the Advance layout, under Customize your Environment
4. This will launch the Android SDK installer, set the installer to install to a folder in your root drive E.g. C:\Android
5. Once it has downloaded, the Android SDK will launch. Select the following to install
   1. All items under Tools
   2. Android 4.0.3 > SDK Platform
   3. Android 4.0.3 > ARM EABI v7a System Image
   4. Android 4.0.3 > Intel x86 Atom System Image
   5. Android 4.0.3 > Google APIs
   6. Android 2.3.3 > SDK Platform
   7. Android 2.3.3 > Intel x86 Atom System Image
   8. Android 2.3.3 > Google APIs
   9. All items under Extras
   10. \*You may wish to download the newer versions of Android to test with, but it is not guaranteed to work
6. Click Install and Accept the license for all items (scroll down and ensure there are all Ticks) and click Install
7. Once all the downloads are completed, you may close the Android SDK.
   1. At this point, if you are running a Intel machine with a Intel processor that supports virtualization, you should install the Intel HAXM so you can run the Intel Android image which will provide a much improved experience on the Android emulator
   2. Go to the Android SDK folder >extras > intel > Hardware\_Accelerated\_Execution\_Manager and install IntelHaxm.exe
   3. If the installation completes successfully, you may use the Intel Android image instead to run the emulator
8. Now you will need to import the necessary modules into Titanium.
   1. From the SVN folder > App > Libraries, extract the contents of ti.nfc-android-1.2.0.zip and paste it in C:\Users\<your windows username>\AppData\Roaming\Titanium
   2. Restart Titanium
9. Now click Import Project on the left hand panel. Select Titanium > Existing Mobile Project and click Next
10. Browse to the correct Project directory and click Finish
    1. SVN folder > App > tBankMobileApp
11. Now, on the left hand panel, the project should have been imported and a few items will appear. Just above, there is an icon with a green play icon with a folder. Click that and choose Run Configurations.
12. Click Titanium Android Emulator and click the New icon near the top, first to the left
13. On the right hand panel, you can choose between the Android 4.0.3 ARM v7a version or the x86 version. If you successfully installed the Intel HAXM, choose the x86 version, else choose the ARM v7a version
14. Set the log level to trace
15. Set the Screen to WVGA 800
16. Check the Re-build Project on Launch box and click Apply then click Run
17. The first time it runs, it will take a while for the emulator to appear. Even after it appears, it will take some time for it to load. Give it around up to 15 minutes, depending on your computer speed.
    1. Sometimes, it will hang at a screen with a status bar on the top. That is good enough, close the emulator and the next run will be fine
    2. Note that the first time you run the emulator, Titanium will report that the emulator is not running. It is fine, once the emulator has started up, you can run the project again and it will detect the emulator.
18. After the first launch, go to the android sdk folder and run AVD Manager. Select the only item on the list and click Edit
    1. Under the Hardware box, change the Max VM Application Heap to 128
    2. If your computer has a dedicated graphics chip, you may enable GPU emulation by clicking New and choosing GPU emulation from the properties list and change the value to yes. \*This will provide a much faster experience on the emulator but if you find your emulator crashes, turn off this option.
    3. When you are done, click Edit AVD and close the AVD manager
    4. If your emulator is running, close it and launch it again from Titanium
19. Once your emulator has reached the locked screen, drag the lock icon to the right to enter the home screen. The go back to Titanium and Run the project again
20. If it doesn't work the first time, try and try again. You may be second time lucky.

### Compiling for Deployment

In the previous steps, we have launched the app onto the emulator for development purposes. Now if we intend to create an APK file for deployment to Android phones, we have to build the application.

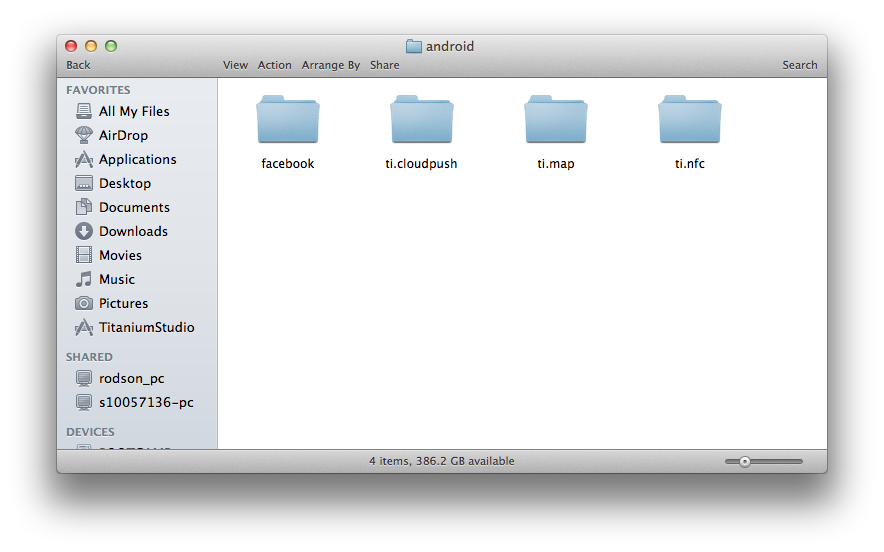
1. Click the Brown package icon 2 buttons right of the Run icon and choose Distribute – Android App Store
2. Set the Distribution Location to where you want to save the APK to
3. For Keystore Location, click browse and go to SVN Folder > App > APK Builds > BankRevels.keystore
   1. The password is: yanniistheleader
4. Click Publish
5. In the future, you can click the newly generated button under the build icon to build it automatically without configuring the settings
6. In order to deploy it to the IBS, you will need to copy the APK file to <Eclipse project folder>/WebContent/and
7. For iOS, <Eclipse project folder>/WebContent/ios

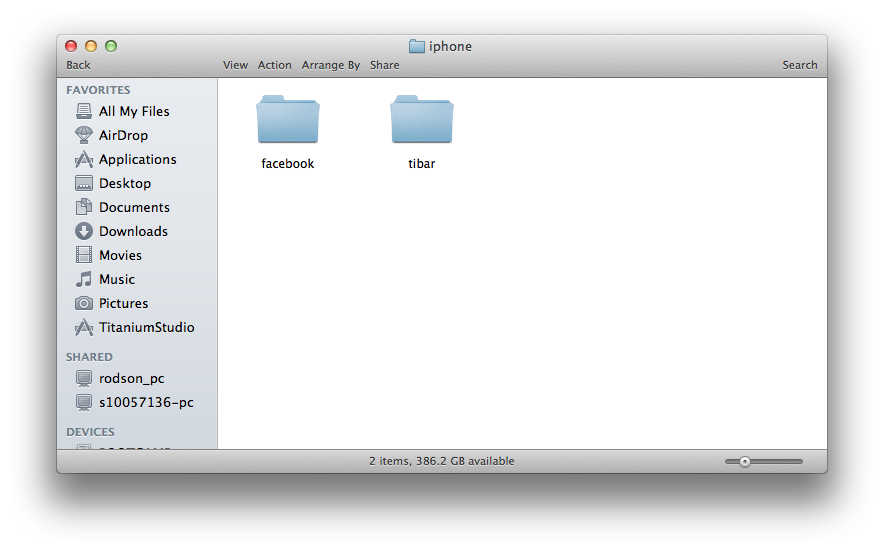
# OS X

## RMB

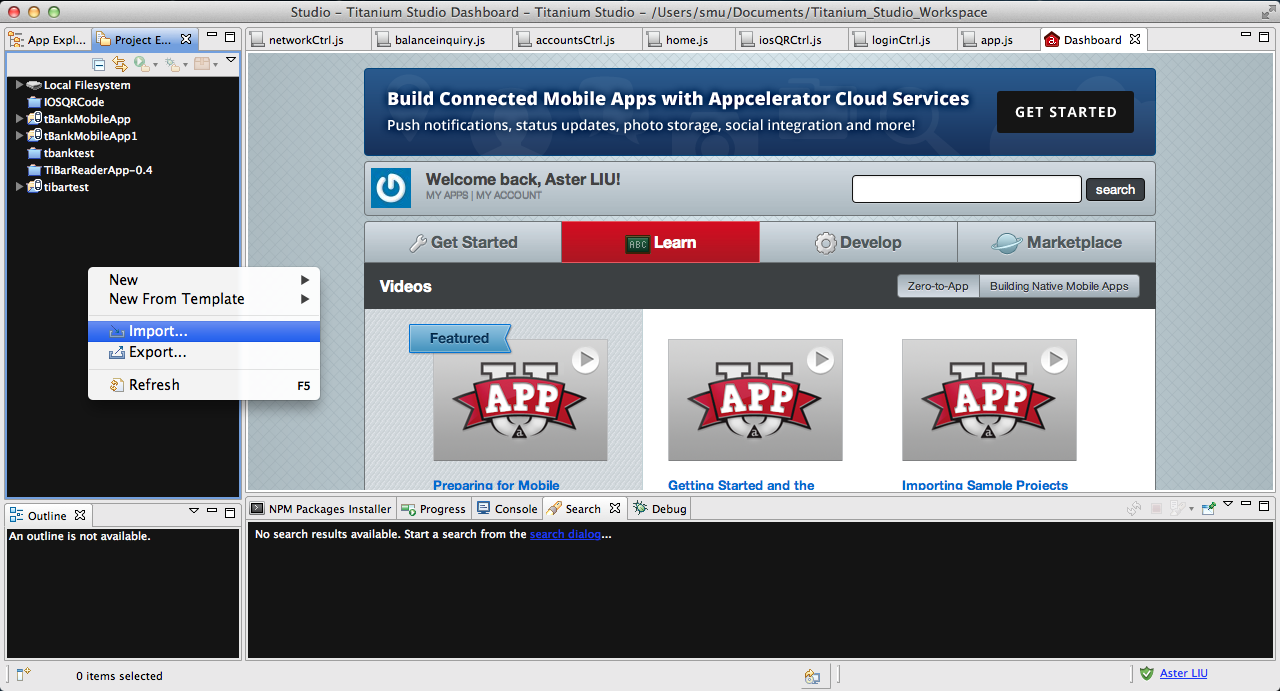
Titanium for OS X is pretty straight forward, install the DMG file from the Titanium website, and then install Xcode from the App Center. Once Titanium launches, go to Getting Started and follow the instructions to install the iOS SDK. Following which import the project as specified in Android.

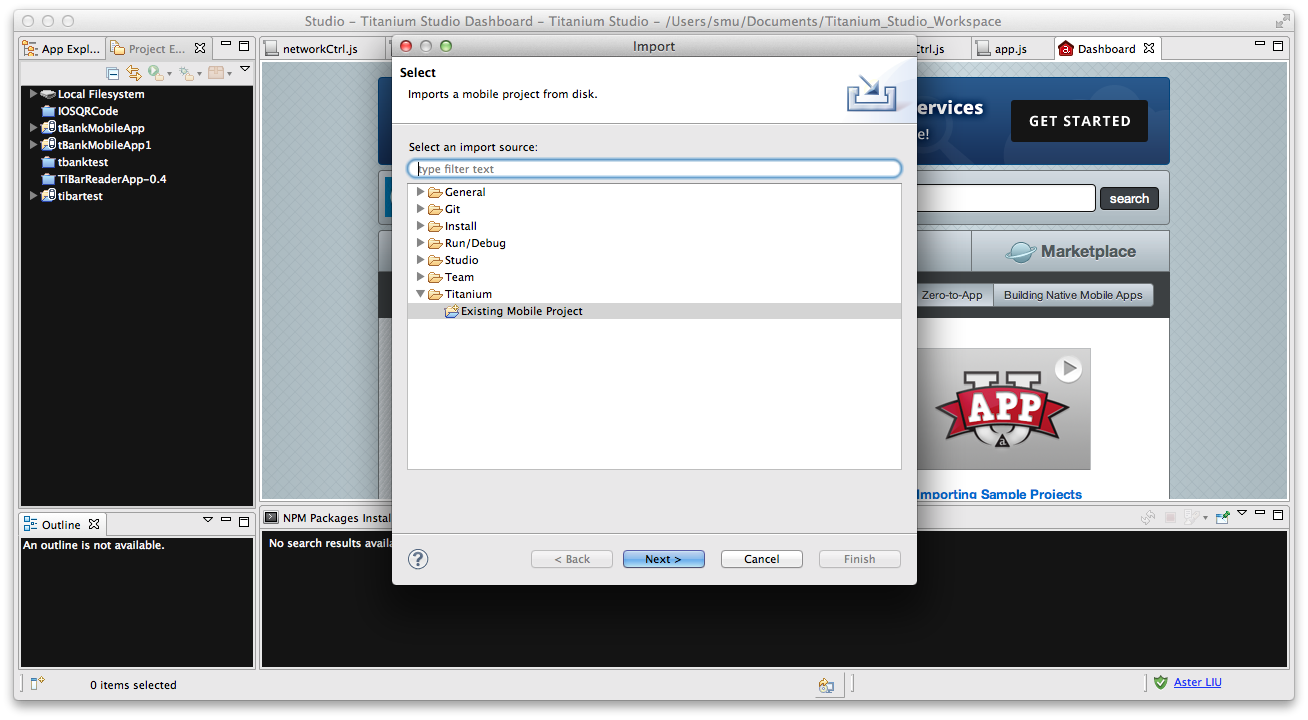
1. Go to folder ~/Library/Application Support/Titanium/modules. Put ti.nfc into android folder and tibar into iphone folder.

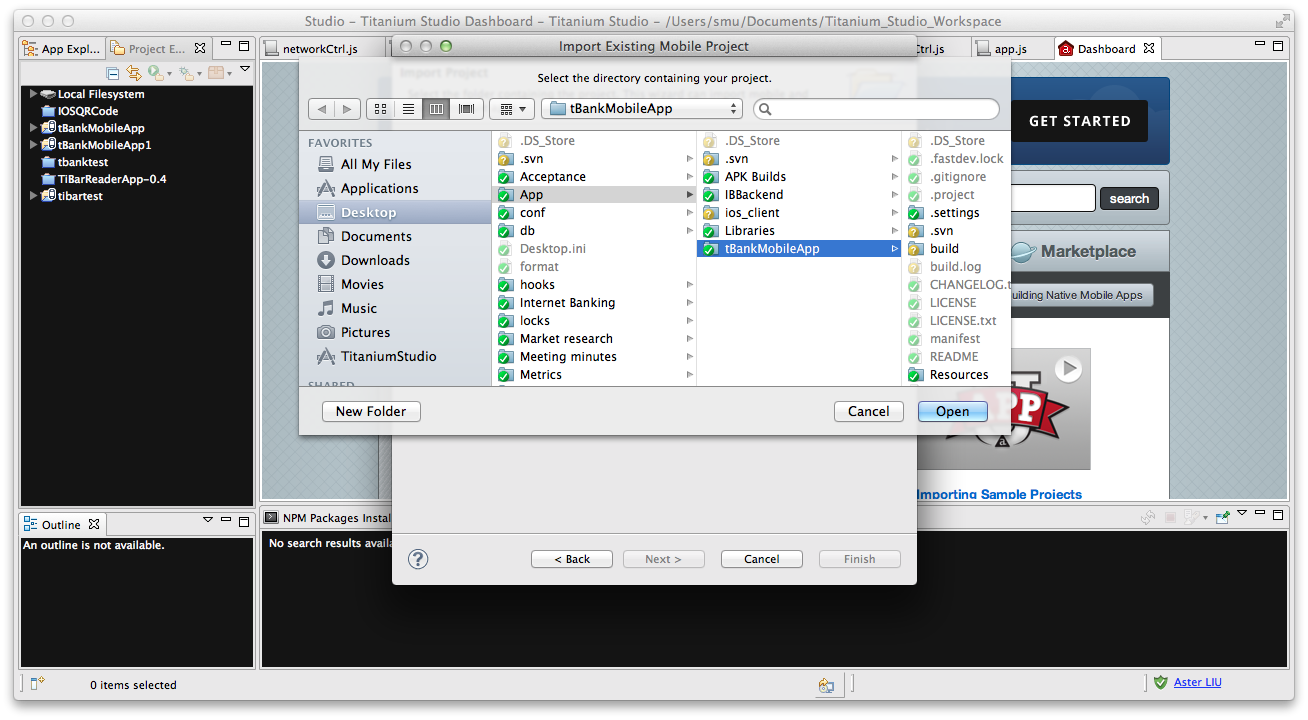




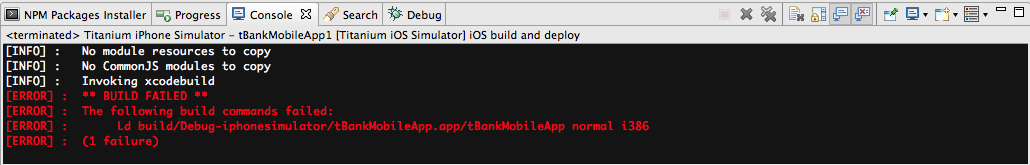
1. Open Titanium Studio, import the mobile project.



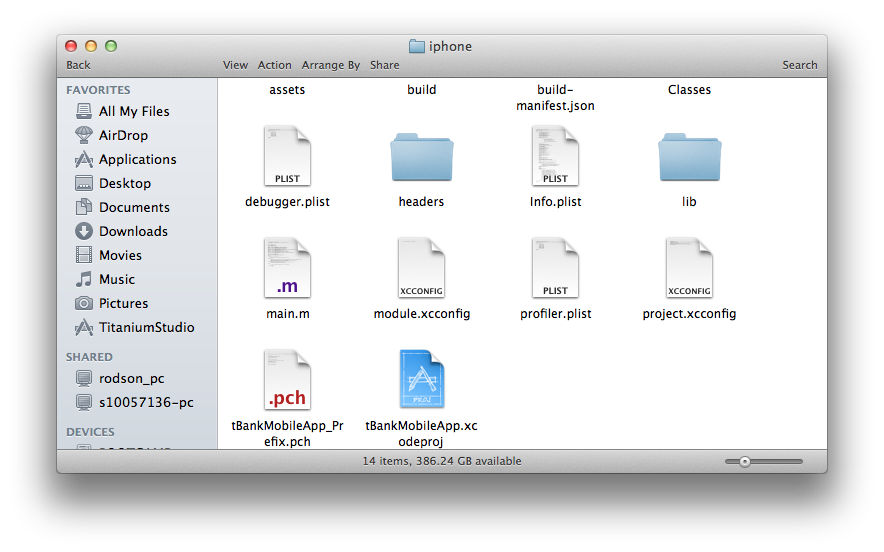




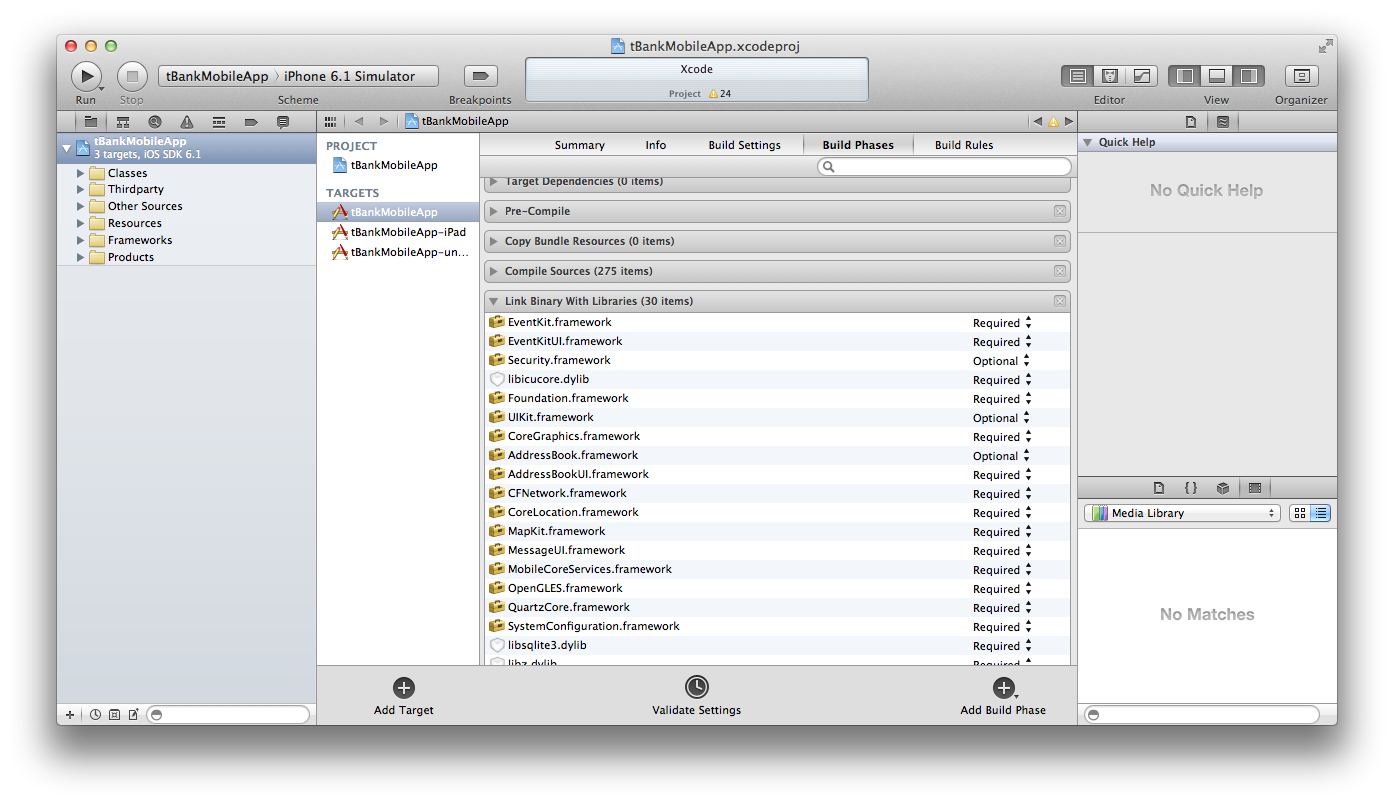
1. Click the  button and run the project on an iphone simulator to build for xcode. It will show you build failed.



1. Go to the project folder, open build/iphone. Open the \*.xcodeproj file with xcode.



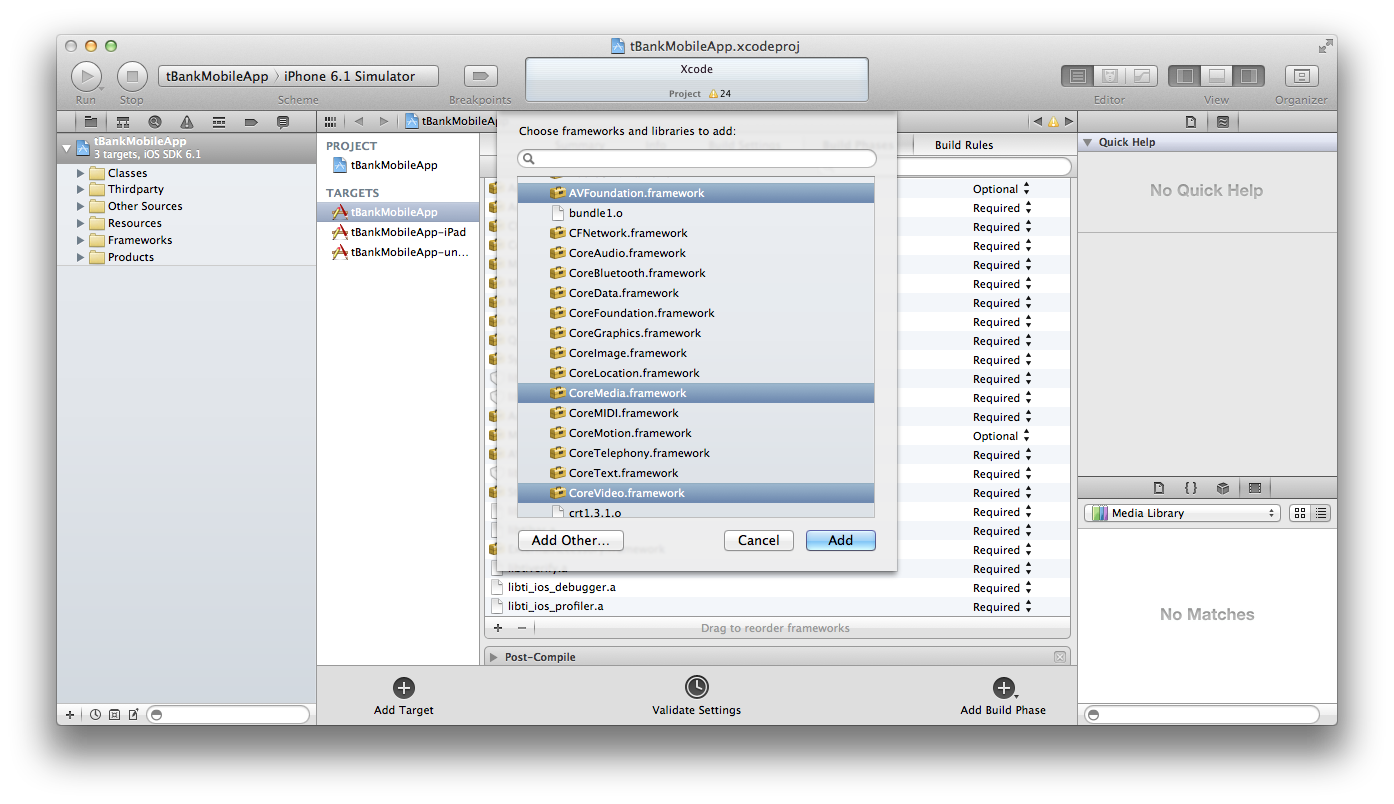
1. Select the project in xcode, go to Build Phases, and then choose Linked Binary With Libraries.



1. Add following frameworks:

* AVFoundation.framework
* CoreMedia.framework
* CoreVideo.framework
* QuartzCore.framework
* libiconv.dylib

These are the frameworks that zbar library need.



Build and run the project in xcode. It will launch the iphone simulator. You can run the project in Titanium Studio as well.

### Compiling for iOS

1. To build a new version of .ipa file and distribute to smu students, contact LE Gia Hai [giahaile@smu.edu.sg](mailto:giahaile@smu.edu.sg) at LiveLabs and pass him the entire source code to be built.