NUS Information Technology
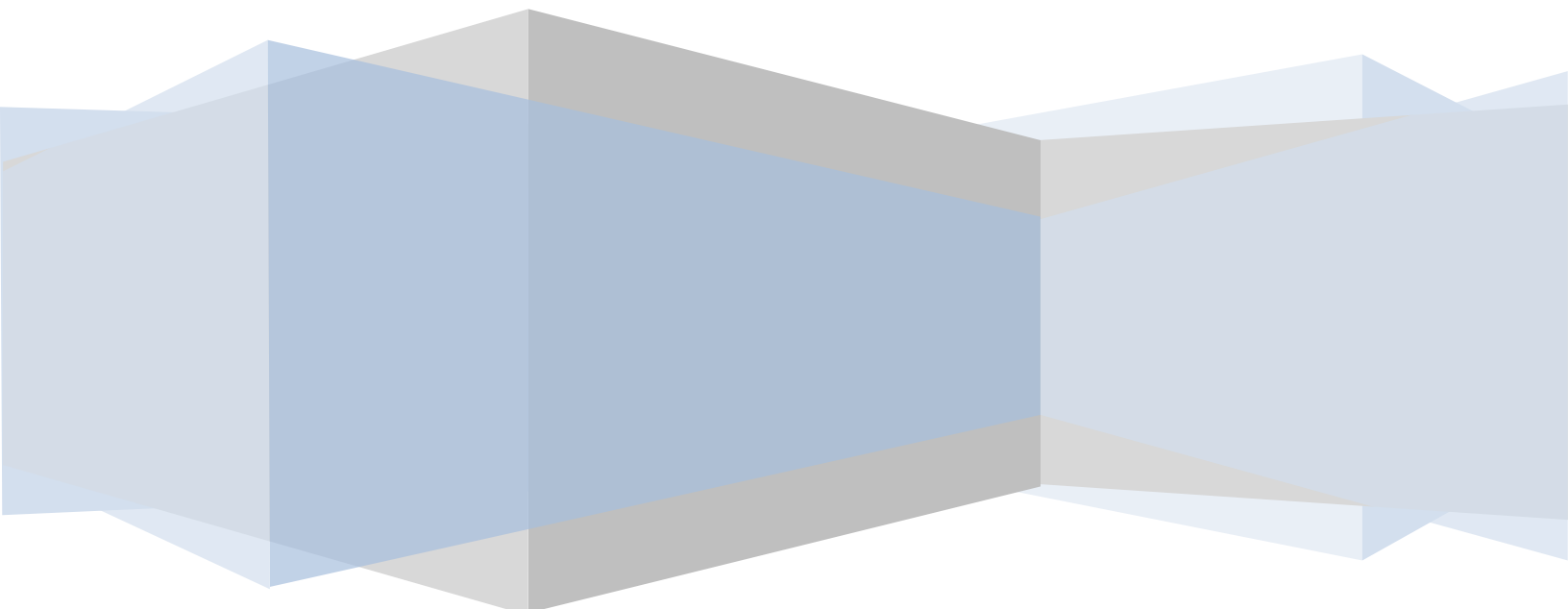
NUS Student Mobile Application Development and Deployment Guide

Version 1.1

Last Update: 10-Dec-2019

Document Change History

Version	Date Changed	Summary of Changes
1.0	19-Jul-2019	First release of the Student Mobile Application Development and Deployment Guide
1.1	10-Dec-2019	Slight modifications made to the Student Mobile Application Development and Deployment Guide

Contents

1. Introduction

The purpose of this Guide is to:
   a) Enable developers to jump start and get quickly assimilated to the recommended development and deployment environment.
   b) Enable developers to understand and perform the applicable processes required in developing mobile applications
   c) Provide a set of guidelines for developing mobile applications
   d) Be used as a handy reference for developers who are handling mobile application development.

1.1 Glossary

1.1.1 Tools
Tools refer to software or utilities (but not limited) that you use to assist in your development process.

1.1.2 Libraries
Libraries refer to software libraries or external packages, written by someone else and provided in an API form for developers to consume and it to accomplish certain functions.

2. Use of 3rd party tools and libraries
Developers are encouraged to use or utilize the list of recommended tools or libraries.

If there is a better tool or library that is not mentioned in the recommended list, and with good reason to use it, please justify your decision to your team.

3. Choice of Mobile Platforms

Two main types of platforms:
   - Mobile Web: web-based applications formatted specifically for use in web browsers of mobile devices.
   - Native Apps: applications that can only run or execute in the environment which it is designed for, such as iOS and Android.

3.1 Mobile Web
A mobile web app is basically a web based application formatted for use in mobile web browsers. It is recommended for:
   - Transactional based applications
   - Applications displaying information
   - Applications requiring login. Can make use of existing login session handling of application server which is proven and secured.
   - Target is to cover as many different types of platforms as possible.

A mobile web app can be developed as a HTML5 based app. HTML5 is a standard for markup language used for structuring and presenting web content. On top of the traditional web-based contents, web browsers that implement the HTML5 standard will provide support for additional functions like:
- Geolocation
- Animation
- 2D canvas for graphics
- Offline Storage

The challenges of HTML5 development:
- HTML5 support varies across browsers and devices

3.2 Native Apps

Native Apps can be further divided into:
- iOS apps (iPad and iPhone)
- Android apps

Native Apps are recommended for:
- Apps that need to utilize mobile device's native features, example:
  o Geolocation (detection of coordinates of locations)
  o Gyrometer and Accelerometer (sense of orientation and position in space)
  o Image or Video capturing
- A light weight version of existing web-based applications with selected functions that can be useful on the go.

3.3 Native Apps vs Mobile Web

	Native Apps	Mobile Web Apps	Remarks
Functionality	Native apps can interface with the device's native features to implement functions based on geolocation, gyrometer, accelerometer, image capturing, etc. ✔️	Mobile web apps can access a limited amount of the device's native features.	Does the app need to tap on any of the device's native features?
Development Effort	Need to develop the app for multiple platforms. Development for each platform is like developing an independent app. Currently both iOS and Android platforms are developed as they are the most popular to date.	Only need to develop a single app. ✔️	Mobile web app is preferred as development effort is lower. Note that this is in comparison with pure native development platforms.

Supported Platforms	Native app can only run in the platform it is designed for.	Basically a web-based application. Works across multiple platforms and devices as long as the mobile web browser supports functions used by the app. ✓	Mobile web app is preferred as it can reach a bigger group of users.
Supported OS	Need to test in multiple versions of the native OS. Certain functions or libraries may work in higher OS version but not the lower, and vice versa. There is also a need to determine the minimum and maximum versions of the OS/SDK your app can support.	Basically a web-based application. Works across multiple OS as long as the mobile web browser supports functions used by the app. ✓	Mobile web app is preferred as effort to ensure app works across multiple OS is less.
Design and User Interface	Allows richer and more complex design and UI by using the platform's native software library and toolkit. ✓	Though javascript libraries like jQuery Mobile can help to provide a richer look-and-feel, the UI still falls behind that of a native app in terms of interactivity and complexity.	Native app is preferred if it requires a rich and interactive interface that cannot be fulfilled by a mobile web app. Mobile web app is preferred if it is a basic app displaying information in standard list or table format.
Performance	A native app that access mainly local data with little or no network access required can typically launch and run faster than a mobile web app. ✓	Transactions mainly networked based, performance closely tied to speed of network.	Native app is preferred if performance is an issue and little or no network access is required.
Deployment	When submitting the native app to the app store, it is not transparent to us how long it will take for the app to be approved and published.	Deployment of apps will be to your own application servers. The deployment time can be under the control of your team or an internal team.	For native apps, especially for iOS apps, there may be a need to factor in deployment time in the project timeline, as it may take up a

	For Apple Appstore, it may take up to one or two weeks, and even more if the app is rejected and fixes are required.	✓		significant part of the project schedule.

4.   Version Control System

The source code should be managed by a version control system. The recommended version control is Git, and we highly recommend using GitHub Desktop for a more streamlined version control management process.

4.1 Guidelines

The followings are some guidelines when using Git.

    a.   A typical application has two main types of branches: **master** and **supporting branches.** The **master** stores stable releases of the application. Our source code at **origin/master** should always be at production-ready state; whereas **the supporting branches** are used to store unfinished versions whose enhancements or bug-fixes require relatively longer time to complete.
        Read more at https://nvie.com/posts/a-successful-git-branching-model/. *Not sure if we are allowed to use links to third-party websites?*

    b.   Git is primarily meant for storing application's source codes and other deployment-requisite files (HTML, JSP, JavaScript, XML, API jar files, etc); developers should abstain from using it to store huge binary files, or as storage for office documents such as Excel, Word, etc. Also practice using .gitignore if there are any files in your source code that are large, but are expected to remain unchanged throughout your development process, for example: node_modules in your npm package. Read more at https://github.com/github/gitignore/tree/master/Global.

The followings are some guidelines when using SVN.

    c.   A typical application has 3 SVN folders, namely **tag**, **branch** and **trunk**.
        The **tag** stores stable releases of the application; whereas **branch** is used to store unfinished versions whose enhancements or bug-fixes require relatively longer time to complete. Lastly, the **trunk** stores current production copy of application and is used for auto-deploying the application to various operating environments.

    d.   The SVN is primarily meant for storing application's source codes and other deployment-requisite files (HTML, JSP, JavaScript, XML, API jar files, etc); developers should abstain from using it to store huge binary files, or as storage for office documents such as Excel, Word, etc.

4.2 Folder Structure of Subversion Repository

Following are the recommended folder structures of the Subversion Repository:

a.   Mobile Web
    Structure should be the same as any web-based application in Java or .NET:
    -   Project/
        o   trunk/

- o   tags/
- o   branch/

b.   Native Apps
   Under the project root, the app should be further branched into Android and iOS:
   -   Project/
      - o   Android/
         - ▪   trunk/
         - ▪   tags/
         - ▪   branch/
      - o   iOS/
         - ▪   trunk/
         - ▪   tags/
         - ▪   branch/

5.   Development Environment

5.1 Mobile Web

The mobile web app is basically a web-based application. It is recommended to develop the application either in the Java or .NET platform and deploy in the Java or .NET application server. The development environment should follow the recommendations in the Developer Guides of the respective platforms.

For the presentation layer of the app, it is recommended to adopt the jQuery and jQuery Mobile frameworks. jQuery is a HTML5-based JavaScript library for desktop browsers to create rich interactive websites and web apps. jQuery Mobile is a framework built on top of jQuery that provides a range of user interface elements and features for you to use in your mobile apps.

The features of jQuery Mobile Framework include:
-   Supports a wide range of devices (iOS, Android, Windows Phone, etc) and mobile browsers (Opera Mobile/Mini, Firefox Mobile, Chrome, etc).
-   Layout and theming engine – quickly style and extend styles.
-   Touch friendly inputs and widgets – improved user experience for form inputs.
-   Ajax –based navigation – faster loading of pages.

5.1.1   Software packages to install to develop Mobile Web applications

#	Software	Details
1	Mac Operating System	Mainly for running iOS Simulator to test the Mobile Web app in iOS mobile browser.

2	Windows Operating System	Mainly for running Android Emulator to test the Mobile Web app in Android mobile browser. It is also possible to run the Android Emulator in Mac and Linux environments.
3	Google Chrome (Desktop)	Desktop version of Google Chrome web browser. Good for functional testing of the mobile web app. The look and feel may differ from mobile browser, but it should be very close to the Android default browser.
4	Safari (Desktop)	Desktop version of Safari web browser. Good for functional testing of the mobile web app. The look and feel may differ from mobile browser, but it should be very close to the iOS default browser.
5	iOS Simulator	Pls refer to the appropriate section and version to install This is to test your mobile web app works on iOS default browser.
6	Android Emulator	Pls refer to the appropriate section and version to install This is to test your mobile web app works on Android default browser.

Please refer to the Java or .Net developer guides for the relevant software to install, based on your target environment to host your web-based mobile application

5.2 Android App

You are strongly recommended to test the application in the latest version of Android or the most widely used version to date. Minimum version of Android supported should be as close to the latest version as possible, while at the same time should be low enough such that a sufficient percentage of devices can still be supported.

5.2.1   Software packages to install to develop Android applications

#	Software	Details
1	Windows (Recommended)	It is also possible to develop in Mac and Linux environments.
2	Android Studio	https://developer.android.com/studio/

3	Java Platform	http://www.oracle.com/technetwork/java/javase/overview/index.html

5	Subversion or Git/GitHub	Subversion tools already comes with Android Studio.
		Tortise SVN
		    - Get from http://tortoisesvn.net/downloads.html
		Command line subversion.
		    - http://subversion.apache.org/packages.html
		
		Developers can use Git or Apache Subversion (SVN) to manage their source code versions. It's highly recommended to use GitHub Desktop https://desktop.github.com/ to simplify your version control management process.

Please refer to the Java or .Net developer guides for the relevant software to install, to develop the respective backends/proxy for the mobile applications.

5.3 iOS App

You are strongly recommended to test the application in the latest version of iOS. The Apple App Store may reject the app if it does not run properly in the latest OS. Target or minimum version of iOS supported should be as close to the latest version as possible, while at the same time should be low enough such that a sufficient percentage of devices can still be supported.

5.3.1   Software packages to install to develop iOS applications

#	Software	Details
1	Mac (Mandatory)	To date, iOS apps can only be developed in Mac OS environment.
2	Xcode (Mandatory)	It is packaged with iOS SDK. Obtainable from https://developer.apple.com/ or the Mac App Store. Comes with the iOS Simulator.
3	Subversion	Command line subversion.
4	Git	Developers can use Git or Apache Subversion (SVN) to manage their source code versions. It's highly recommended to use GitHub Desktop https://desktop.github.com/ to simplify your version control management process.
5	CocoaPods	Dependency manager for Swift and Objective-C Cocoa projects.

Please refer to the Java or .Net developer guides for the relevant software to install, to develop the respective backends/proxy for the mobile applications.

6.   Application Architecture – A Recommendation

In broad terms and with respect to application design and development context, Application Architecture is about structures and patterns. It defines the key components that are typically required to support enterprise application. For further reading on Application Architecture, you may check out:
https://medium.com/ios-os-x-development/ios-architecture-patterns-ecba4c38de52
https://developer.android.com/jetpack/docs/guide
https://www.freecodecamp.org/news/my-crypto-coins-app-series-part-3-df57f86daf74/

6.1 Benefits of an Application Architecture

a.   Flexible Architecture

By having a well-layered and component design, it allows the architecture to be evolved over time. Having loose coupling and separation of concerns (SoC) within an architecture, the components can be swap in and out with other technologies without affecting the entire architecture.

b.   Ease of Maintenance

With a good component design, it is also easier to maintain the application. Complexity is largely hidden by abstraction and encapsulation. Complicated logic are written once but used many times. Isolation of bugs is made easier because culprit components can be more easily identified.

c.   Performance and Scalability

By adopting proven frameworks, the application architecture can be assured to be designed with performance and scalability in mind. These frameworks, implemented with best practice design patterns, form a strong foundation for an enterprise application implementation.

d.   Lower Development Costs and Risks

With the application architecture, development costs and risks will also be significantly lower. Through the use of encapsulation, developers will focus on implementing business functionality while a team of senior developers will develop the architecture framework to support the application. Reusability is high and bad programming practices are more or less confined to the business layer.

6.2 Mobile Web

For Mobile Web, please follow the MVC design guidelines defined in the Developer Guides of the respective Java or .Net platforms.

6.3 Native Apps

MVC design for both iOS and Android platforms. The development environment for iOS and Android already enforces a standard MVC design.

6.4 Same source for TEST and PROD environments

The objective for having the same source codes for SIT (Test) and PROD (Production) environment is to ensure the integrity and consistency of the application working in both environments.

7.   Design & Coding Guidelines

7.1 Mobile Web

For Mobile Web, please follow the Design and Coding Guidelines defined in the Developer Guides of the respective Java or .Net platforms.

7.2 Native Apps

For Android apps, there are some guidelines in the Android Developer website on the design and functionality. These guidelines help to ensure the quality of the app when it is published in the app store. Please refer to the following link for the guidelines:

http://developer.android.com/distribute/googleplay/quality/core.html

For iOS Apps, there are guidelines on the various design and functionality aspects of the app in the Apple Developer website:

https://developer.apple.com/develop/
https://developer.apple.com/design/human-interface-guidelines/
https://developer.apple.com/app-store/review/guidelines/

7.2.1   Backend Server Application

Native mobile apps may sometimes require a backend server application for the following purposes:
- As a middle layer to interface with other systems
- To access data in databases
- To process and store data pass in from frontend mobile apps

The following applies to backend server application to be developed by your team:
- It is recommended to develop the backend server application either in the Java or .NET platform and deploy in the Java or .NET application server.
- The development process should follow the recommendations in the Developer Guides of the respective platforms.
- Webservice should be used as the communication channel between the frontend mobile app and the backend server application.
    - It is strongly recommended to use REST-based webservice with json data format
- HTTPS should be used for the communication channel between the frontend and the backend.

7.2.2   Data Handling

Following are some points to take note in data handling pertaining to Native Apps:
- For native app transmitting data to and from a server, HTTPS or any secure communication channel should be used.
- Storing of data locally in the mobile device is not recommended. If there is a need to do so, it is mandatory to protect sensitive data by encryption.
- If the app store data on any cloud services:
    - For sensitive data, it is mandatory to be protected by encryption.
    - For normal data, it is recommended to be protected by encryption.

8. Testing

8.1 Unit Testing
Test small bits of your programme e.g. a function call.

8.2 Regression Testing
When you just add in a new feature, test to check that the new codes have not affected your existing working features.

8.3 .System Integration Test
System Test tests the behavior of a complete and fully integrated software product based on the software requirements specification (SRS). It is black box type of testing which does not require knowledge of internal design, structure or code and totally based on the user's point of view.

Processes:
- Create System Test Plan.
- Create Test Cases.
- Create test data used for System Test.
- Run the Test cases.
- Bug Reporting, Bug verification & Regression testing.

8.4 User Acceptance Test
User Acceptance Test ensures end users accept the delivered system. It is based on user requirements specification.

Processes:
- Analyse business requirements.
- Create UAT test plan.
- Create UAT test cases.
- Prepare test data (production like data).
- Run the test cases.
- Record the results.
- Confirm business objectives.

8.5 Testing for Different Platforms
For Mobile Web, please refer to the Testing Guidelines defined in the Developer Guides of the respective Java or .Net platforms for the description of the different server environments (Local, SIT, QAT and PROD) and their testing sequence.

For Native Apps, developer needs to test in emulator and physical device for system integration test. For user acceptance test, user needs to test in physical device.

9. Deployment

9.1 Mobile Web

For Mobile Web, please follow the Deployment Guidelines in the Developer Guides of the respective Java or .Net platforms.

9.2 Native Apps

For deployment of native apps, the Android apps will be published to the Google Play Store and iOS apps will be published to the Apple App Store.

9.2.1   Publishing Android Apps to the Google Play Store

For Android apps to be published to the Google Play Store, it may take about 1 or 2 days for apps to be available for downloads after submission. This applies for both new submission and submission of updated apps.

For more details about publishing Android Apps on Google Play, please refer to the Google Play Launch Checklist:
http://developer.android.com/distribute/googleplay/publish/preparing.html

9.2.2   Publishing iOS Apps to the Apple App Store

It may take up to 2 weeks for apps to be available for downloads after submission to Apple App Store. It may take even longer if the app is rejected by Apple due to some issues which need to be fixed before resubmission. This applies for both new submission and submission of updated apps. The waiting time for apps to be available for downloads can be quite significant, and so it should be factored into the development schedule and go-live date of the app.

For more details about publishing iOS Apps on Apple Appstore, please refer to the App Distribution Guide:
https://developer.apple.com/app-store/submissions/
https://developer.apple.com/app-store/review/guidelines/
https://help.apple.com/xcode/mac/current/#/dev8b4250b57