Pyroferus
Technologies

Testing Approach

For Mobile Application

Introduction

This document can be used as a starting guide towards developing a more comprehensive mobile application test strategy specifically for a mobile application, product or a mobile application development project. It merely highlights a few important points to keep in mind while developing a testing methodology for a mobile application.

MOBILE APPLICATION

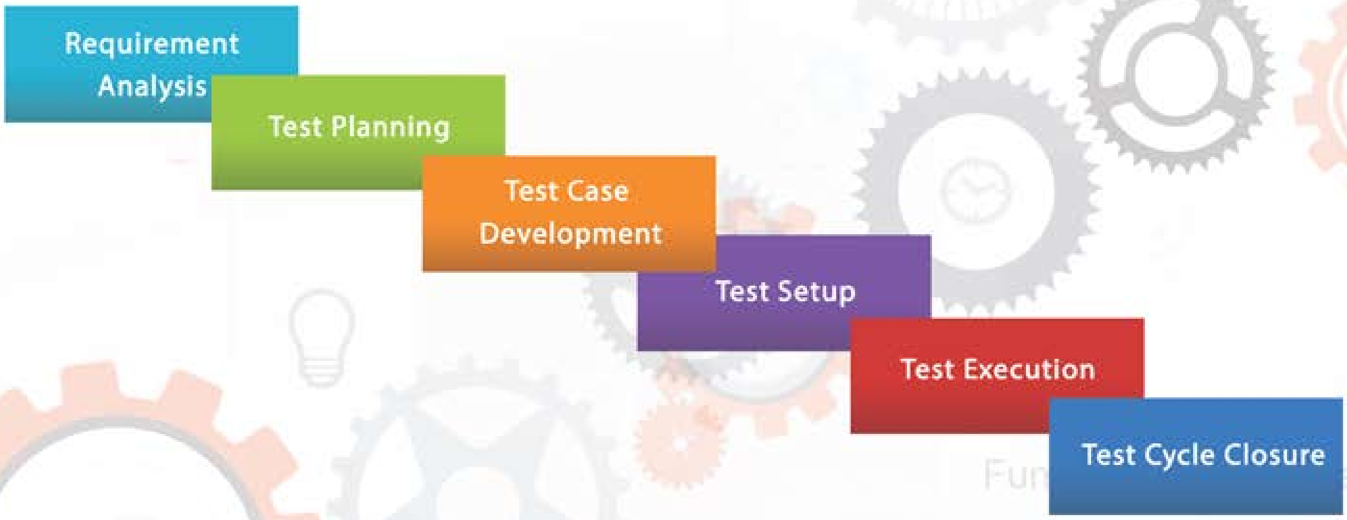
Testing Coverage

The mobile application testing can be covered with the following testing…

- Functional Testing
- User Experience Testing
- Interoperable Testing
- Security Testing
- Performance Testing

1. Functional Testing

This is the traditional testing method used to validate compliance of the mobile application with the functional requirements and business needs. Based on the developed test cases, mobile testers should do manual testing to test the mobile app manually as a "black box" to see if the functions provided are correct and work as designed…



1.1 Our Approach to Functional Testing

- Requirement analysis through FSD & Design document
- Preparation of Test plan
- Preparation of Test cases
- Preparation of RTM
- System Testing to cover the major functional areas
- Execution of Test cases
- Bug reporting and tracking
- Test summary report

2.User Experience Testing

The coverage of User Experience Testing is as follows…

- Graphic design – Layout, Navigation, Position of icons, Different display format, Readability, Text display (alignment, overlap, text wrapping), styles, fonts, content, images, look & feel
- Interaction – Touch screen, Motion sensor, Gestures, Input, Error messages and warnings
- Transitions between screens
- Usability – Screen setup, action chains, Progress bars, Multitasking
- Logic for the presentation of data and behaviour of the forms that collect data
- Data consistency
- Language – Grammar, Spelling

2.1 Our Approach for User Experience Testing

The approach for User Experience Testing is, based on the UI checklist pass/fail criteria user experience testing result should be published to all the stake holders

Preparation of UI checklist

Execution of UI checklist

Publishing the UI checklist result

2.2 User Experience Testing Methods

One-on-one usability testing where users are tested in a controlled 'laboratory' environment. This is the most effective method used to identify why our mobile application is not performing and how it can be improved

One-on-one usability testing

Prototyping where users are observed interacting with a design document / paper mock-up of the mobile application to ensure you get the design right before start testing the mobile application

Prototype Verification

3.Interoperable Testing

When doing testing of actual devices, it is necessary to test the application on every possible device platform. To avoid this scenario, we can utilize a Platform Matrix in our testing. This approach can actually reduce the number of combinations that you have to test.

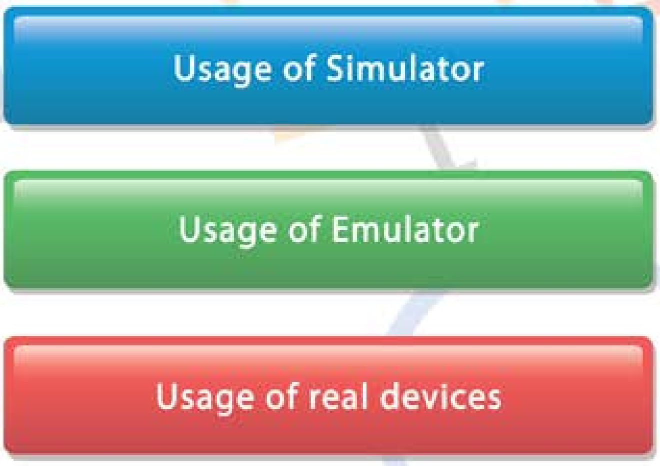
Testing needs to ascertain that the app can be successfully downloaded to the device, executed on the device and interact with the supporting back content infrastructure. When updates are made we need to be sure that the application can to be pushed out to and accepted by the end user. There's a misperception that successful testing of app functionality on one device provides assurance across all others of the same operating system

This kind of testing assesses the application/web using a wide variety of operating systems, device types, variations in screen resolution and its screen size, frequent OS changes and multiple standards with protocols

Ensure the compatibility testing for the mobile application installation/uninstallation and the transaction request/response initiated from the mobile banking application

3.1 Our Approach for Interoperable Testing

Using the bible of "Platform Matrix" the mobile application should be tested using the Simulator (Platform specific), Emulator (Device specific) and Actual Devices



3.1.1 Compatibility Testing under Screen Resolution

Resolution - The total number of physical pixels on a screen

Screen density - The quantity of pixels within a physical area of the screen, usually referred to as DPI (dots per inch)

Density-independent pixel (DP)

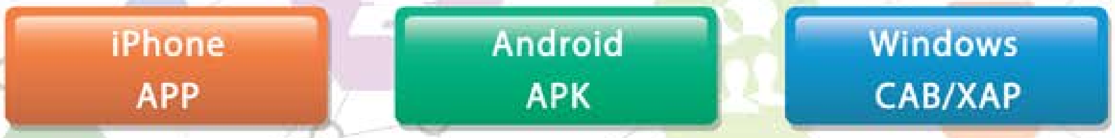
This is a virtual pixel unit that we would use when defining a layout's UI in order to express the layout's dimensions or position in a density-independent way. The density-independent pixel is equivalent to one physical pixel on a 160 DPI screen, which is the baseline density assumed by the system of a "medium" density screen. At runtime, the system transparently handles any scaling of the DP units as necessary, based on the actual density of the screen in use. The conversion of DP units to screen pixels is simple: pixels = DP * (DPI / 160).

For example, on a 240 DPI screen, 1 DP equals 1.5 physical pixels. Always use DP units when defining your application's UI to ensure that the UI displays properly on screens with different densities

	Low density (120), *ldpi*	Medium density (160), *mdpi*	High density (240), *mdpi*	Extra high density (320) *xhdpi*
Small screen	QVGA (240x320)		480x640	
Normal screen	WQVGA400 (240X 400) WQVGA (240X432)	HVGA (320x480)	WVGA (480x800) WVGA854 (480X854) 600X1024	640X960
Largescreen	WVGA800 (480X800) WVGA854 (400X854)	WVGA800 (480X800) WVGA854 (400X854) (600X1024)		
Extra large screen	1024X600	WXGA (1280X800) 1024X768 1280X768	1536X1152 1920X1152 1920X1200	2048X1536 2560X1536 2560X1600

3.1.2 Platform Matrix

Platform matrix should be ensured for all the testing devices



3.1.3 Android Build Installation

Android specific Mobile Application build is delivered as APK file to the mobile devices.

APK

Android Application Package file is used to distribute and install application software on the android operating system available in the mobile devices.

Each Android application is compiled and packaged in a single file that includes all of the application's code (.dex files), resources, assets, and manifest file. The application package file can have any name but must use the .apk extension

3.1.4 Windows Build Installation

Windows Phone specific Mobile Application build is delivered as XAP file to the mobile devices for the windows phone OS 7.x to 8.x and CAB file to the windows phone OS 5.x to 6.x.

XAP

XAP is the application package - it is the distributable unit that allows us to install the application on a device and it is a compressed output file for the Windows Phone application. Basically, it is a ZIP file with a different extension. If we change XAP to ZIP, we'll be able to read its contents fairly easy. The application contains Windows Phone solution components such as the compiled XAML and code-behind, an application manifest and possibly one or more assemblies.

CAB

A cabinet (.cab) file is a library of compressed files stored as a single file. Cabinet files are used to organize installation files that are copied to the user's system

3.1.5 Apple iOS build Installation

iPhone/iPAD specific Mobile Application build is delivered as APP file to the mobile devices from the App store and
IPA file to the mobile device from the iTunes as a developer edition.

IPA/APP

iOS App Store Package(IPA) which stores an iOS application in which each .ipa file is compressed with a binary for the ARM architecture that can only be installed in iOS devices, which is extracted from the APP file product folder. If we change the extension to .zip we will be able to unzip it and view the contents

3.1.6 Native, Mobile Web and Hybrid application

Native apps are built for a specific platform with the platform SDK, tools and languages, typically provided by the platform vendor (e.g. xCode/Objective-C for iOS, Eclipse/Java for Android, Visual Studio/C# for Windows Phone).

Mobile Web apps are server-side apps, built with any server-side technology (PHP, Node.js, ASP.NET) that render HTML that has been styled so that it renders well on a device form factor.

Hybrid apps, like native apps, run on the device, and are written with web technologies (HTML5, CSS and JavaScript). Hybrid apps run inside a native container, and leverage the device's browser engine (but not the browser) to render the HTML and process the JavaScript locally. A web-to-native abstraction layer enables access to device capabilities that are not accessible in Mobile Web applications, such as the accelerometer, camera and local storage

4.Performance Testing

The performance of the mobile web, or mobile app, is the most significant factor affecting conversion rates for mobile device users. Load and performance testing is critical in spotting load and performance issues that can have a negative effect on user conversion.
For the mobile web, HP Load Runner/Performance Center can be used for load and performance testing. This product tests mobile web browser performance by getting a native browser to simulate a mobile browser.

For mobile apps, it depends on the platform and architecture of the mobile apps. Most mobile apps get their data through the service layer. One way we can do performance testing is by stressing the service layer of the mobile app while manual users simultaneously access the apps. For example, if the data layer is through web services or a REST services call, the performance of these services is tested while manual testers simultaneously access the mobile app. With this approach, something close to real metrics can be achieved.

Performance of the mobile application is having dependencies of device hardware, web server, application server and network bandwidth for data calls

4.1 Our Approach for Performance Testing

We have to assess the following components to ensure the performance of the mobile application with the help of test conditions

Application Response Time

Load Test

Network Conditions Wi-Fi, 2G, 3G, 4G

CPU/Memory usage

Application Error Handling

4.1.1 Testing on various network strengths

- No network, Low, Medium, High and testing during changes of network strengths Low/High and High/Low
- Test intermittent network scenarios that a user might encounter in the real world:

  - Walk out of Wi-Fi range so the connection automatically switches to 3G/2G (for example, in a large building like a hospital or airport, or outdoors)
  - Ride in an elevator or on a train where the network connection may go up and down
  - No network connection available at all

4.1.2 Testing on various network types

- 2G (GPRS, CDMA, EDGE)
- 3G, Wi-Fi, Different types of plans based on service providers
- Only Wi-Fi connection

4.1.3 Testing in various battery strengths

High, Low strengths. Observe the battery consumption rate as the application is run in background or foreground

4.1.4 Monitoring memory usage patterns

- Check on application launch, when app runs in the background and foreground, on app exit and when app runs for a long time.
- When no other 3rd party applications are installed, some third party applications are installed, lots of third party applications that may use some of the same hardware resources as our app (such as the mic) are installed to test interactions
- SD Card interactions – If our application can potentially store or retrieve items on the device's SD card, then it is important to test the application behavior when there is an SD card present and when it isn't.

4.1.5 Interruptions

examples of interruptions – incoming call, receiving message, receiving alerts, camera activated, device shutdown, remove battery, lose and recover network connection

4.1.6 Stress & Performance testing

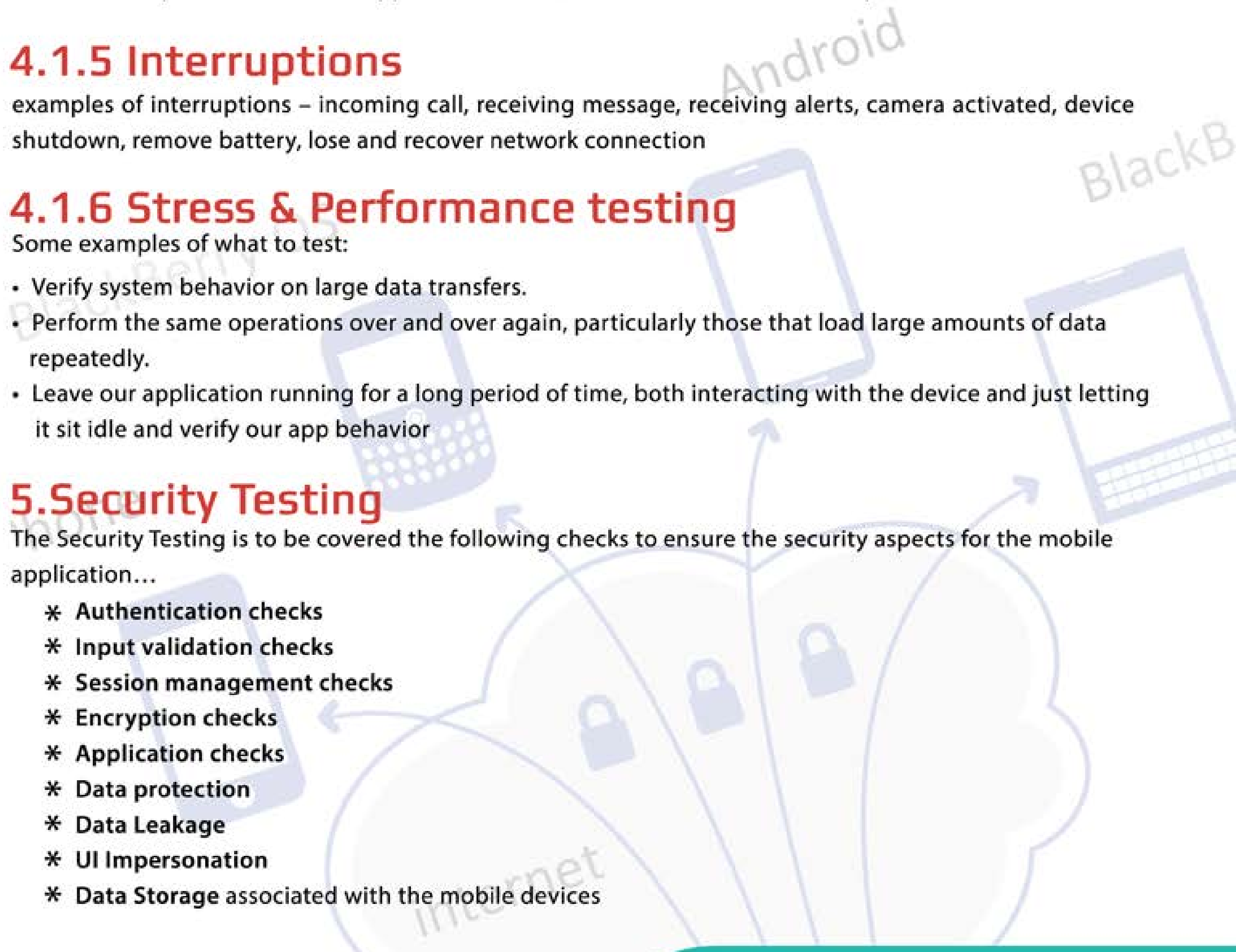
Some examples of what to test:

- Verify system behavior on large data transfers.
- Perform the same operations over and over again, particularly those that load large amounts of data repeatedly.
- Leave our application running for a long period of time, both interacting with the device and just letting it sit idle and verify our app behavior

5.Security Testing

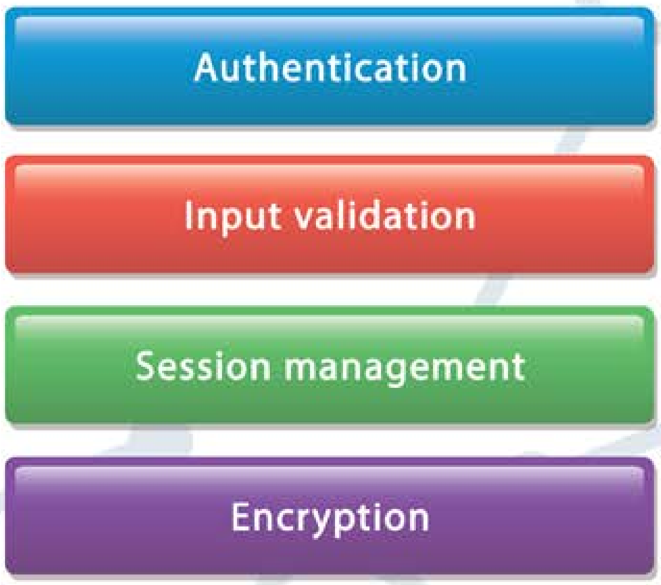
The Security Testing is to be covered the following checks to ensure the security aspects for the mobile application…

- **Authentication checks**
- **Input validation checks**
- **Session management checks**
- **Encryption checks**
- **Application checks**
- **Data protection**
- **Data Leakage**
- **UI Impersonation**
- **Data Storage** associated with the mobile devices



5.1 Security Checking Process

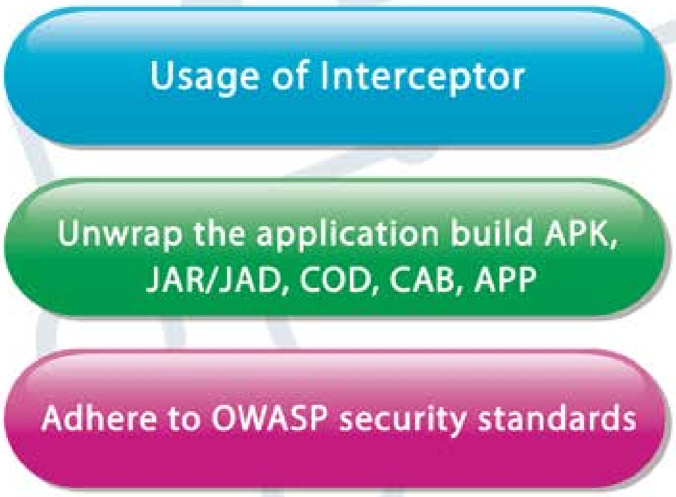
The Security Checking process can be considered as follows:



5.2 Our Approach to Security Testing

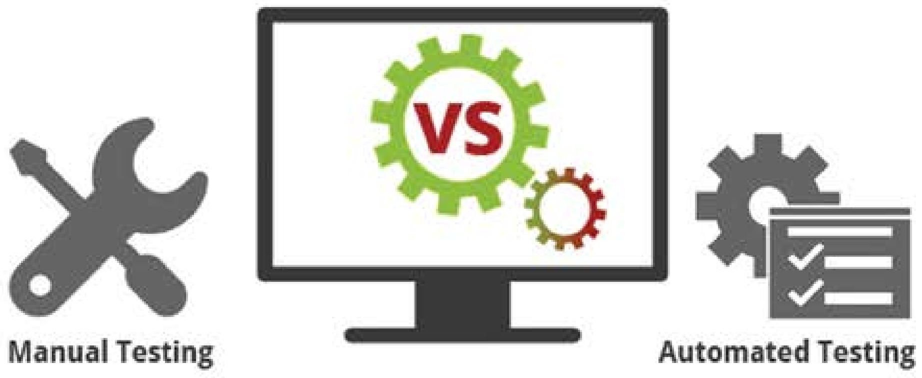
The following testing techniques should be implemented to ensure the application security.

Ensure the security cases should be executed to adhere the **OWASP standards** and test scenarios should be used to cover-up the interceptor and unwrapping.


Usage of Interceptor
Unwrap the application build APK, JAR/JAD, COD, CAB, APP
Adhere to OWASP security standards

6.Manual Vs Automated Testing

Automated testing is highly effective in consistently repeating a test procedure in regression testing as well as testing during the development stages. However, test automation requires significant amount of initial investment.



Therefore, test automation should be done only in the scenarios when:

- The solution lifecycle is long, and the application is growing and evolving
- The scale and frequency of regression testing is high
- A large chunk of test cases includes existing functionality test cases

In reference to mobile application testing, automation should be used to:

- Verify application compatibility when a new OS version is released
- Check backward compatibility when the application is upgraded

Few recommendations for testing with tools such as emulator and automation…

Type of testing	Manual testing (Device)	Manual testing (Emulator)	Automated testing
Unit testing	No	Yes	No
Integration testing	No	Yes	No
System testing	Yes	No	No
Regression testing	Yes	No	Yes
Compatibility testing	Yes	No	Yes
GUI Testing	Yes	No	No
Performance testing	Yes	No	Yes
Security testing	Yes	No	No

6.1 Expected Automation Tools

Following automation tools are expected to be used for our testing…

LogCat – For Logs, Every Actions, Performance testing
Appium – For Functional testing
OWASP – For Security testing

7.Skill Set

1. Strong knowledge in testing techniques
2. Knowledge to understand the Requirements
3. Writing the test cases based on the Requirements for mobile apps
4. Strong knowledge in testing tools
5. Strong knowledge in Java / Eclipse to view the LogCat for error logs
6. Strong Knowledge to understand the mobile functionalities
7. Strong Knowledge to understand the mobile App behaviour in backend
8. Strong knowledge to test the Android / iOS / Windows apps
9. Need to test the performance / compatibility of the Mobile apps
10. Capable to handle the critical situation