Best Practice Guidelines

for developing quality mobile applications

version 1: March 15 2011

The Unified Testing Initiative (UTI) has put together some **Best Practice Guidelines** for developing quality mobile applications irrespective of the platform for which they are being developed.

Some are common sense, and others are less obvious, but they're all worth checking off before your app goes live.

The user outcomes aren't supposed to be rigid, and of course applications that are intended to be different by design will always be acceptable, provided that the user experience is not adversely affected.

If anyone is planning to use the best practices as a basis for testing their app, we suggest you use a device to which a factory reset has been applied prior to the installation of the application to be tested. This will ensure that there is a known base with only pre-installed applications and any errors will be attributable to the application under test.

These Best Practice guidelines will be updated on an ongoing basis as a result of input from the community and changing platform requirements. UTI welcomes input via the UTI blog at www.unifiedtestinginitiative.org/blog

We hope you find the guidelines useful.

List of Contents

Limitations and expectations in the mobile environment

Mobile devices typically have limited display areas, a keyboard unsuited to lengthy input, and less processing power and memory capacity than PC-format devices (although the gap is narrowing for high-end devices).

The bandwidth of a mobile connection is also usually lower than a fixed-line connection, and the latency is higher. Also, the user may be limited in the amount of data they can consume without incurring additional charges.

PC-format devices have had many years to develop consistent UI layouts that are used across most applications on a platform, while mobile device UIs are still evolving and some applications may experiment with unique UI layouts that are different to the common visual language of the underlying platform. This can sometimes mean that user expectations are not always aligned with developer intentions.

For all these reasons, it is important that mobile applications operate in a predictable and consistent manner, and limit their consumption of data, system and network resources as much as possible.

The Best Practices

Installing and Launching

OTA install

The Application should install over-the-air (OTA), with the icon for the application found from the device.

Long Launch Time

All applications should notify the user if there's going to be a long launch time. If the Application takes longer than five seconds to be ready for use, a progress bar or a message should be displayed to tell the user what is happening.

Memory and file storage during run

For an application that writes to file system, ensure that it correctly handles out-of- space exceptions during execution, and gives a meaningful warning to the user advising about lack of space when a file is trying to be stored.

Connectivity

For an application using an HTTP network connection ...

Networking should take place in a separate thread, so as not to block other application activities and to allow the app to display progress.

Networking should take place in a separate thread, so as not to block other application activities and to allow the app to display progress.

symbian

It should be able to use simultaneous connections properly and the user should be able to control the connections through the Connection Manager.

Please note: Users might have two types of connectivity - eg WiFi and cellular - and the device might be switching between the two. The app must respond to this.

Blocked Connectivity

Ensure that it can handle the network connection being invalid / unusable (e.g. data connection / APN not properly set up or invalid for current carrier) or the device being switched into offline / flight mode.

The user should be notified with an appropriate message if this occurs: ideally the application should indicate that the device is in Airplane mode (or equivalent) stating that the application cannot run successfully.

Sending and receiving data

Ensure your application can connect via a valid data session setup and send/receive data via an HTTP network session. You need to make sure that the application data is properly sent / received over the network. (Check it for each Application screen or feature that uses data services).

Network delays and loss of connection

When the application uses network capabilities, it should be able to handle network delays and any loss of connection, giving a clear error message to the user indicating there was an error with the connection.

Messaging & calls

Sending or receiving SMS and / or MMS

For an application that sends and/or receives SMS or MMS messages as part of its function, ensure that it can send messages successfully and that:

a) a notification of a new message is given where enabled on the receiving handset
b) the message is in the correct format, and, where MMS is involved, it contains the correct payload.

Handling telephone call when application is in use

The user must be able to accept an incoming phone call while the application is running. It should then be possible to resume from the same point in the application at the end of the call, or a logical re-starting point.

The application should not block the user from making emergency calls on a cellular network.

External Influence

Memory card insertion

For an application on a device which supports removing and replacing memory cards while applications are running, the application should handle this gracefully and continue to operate as designed.

User Interface

Battery Life

Consider battery management, and allow the power-saving features of the device to be used, including sleep functions for the screen and the device itself.

Don't hijack the native experience

Only hide the status bar where the application's user experience is better without it. The back button should always navigate through previous screens. Use native icons consistently. Don't override the menu button. Instead, put menu options behind the menu button.

Respect user expectations for navigation. The back button should always navigate back through previously-seen screens.

Always support trackball navigation. Understand your navigation flow when the entry point is a notification or widget.

Navigating between application elements should be easy and intuitive.

Discrimination

Don't make assumptions about screen size, resolution, orientation or input. Never hard-code string values in code. Use Relative Layouts and device independent pixels. Optimize assets for different screen resolutions, and use reflection to determine what APIs are available.

Read time and readability

There should be a comfortable amount of time for content reading. Each screen should be visible for the time necessary to comfortably read all its information. Everything in the application should be in a font size and type that is readable by the user.

Touch screen use

For applications used in a touch screen device without stylus, on-screen elements should be of sufficient size and responsiveness to provide a good user experience.

Screen repainting

The application screens should be correctly repainted, including cases when edit boxes and dialog boxes are dismissed. Also, there should be no blinking of moving objects and background. If the application objects overlap they must still render correctly.

Consistency

The application UI should be consistent and understandable throughout, e.g. displaying a common series of actions, action sequences, terms, layouts, soft button definitions and sounds that are clear and understandable

Application speed

The application should work in the device it was targeted for, and should be usable on the device: the speed of the application should be acceptable to the purpose of the application and must not alter the user experience by being uncontrollable.

The speed of the application should be good enough for the application usage (i.e. the frame rate or response to user input should remain adequate, and not compromise the application usage, or prevent the user from progressing normally). For games, to ensure a smooth looking experience for your user, a minimum generally accepted frame rate is 15 frames per second. Also, make sure not to base animations on the frame rate. In other words, you may not want to paint and adjust drawing locations every time the drawing loop is cycled through. You should introduce timers into your code to ensure that animation is consistent across devices. On some devices, you may need to adjust drawing locations less frequently than on others.

Error messages

Any error messages in the application should be clearly understandable. Error messages should clearly explain to a user the nature of the problem, and indicate what action needs to be taken (where appropriate).

Function progress

Any function selected in the Application should give evidence of activity within five seconds. There should be some visual indication that the function is being performed. The visual indication can be anything that the user would understand as a response, e.g.

- prompting for user input;

- displaying splash screens or progress bars;

- displaying text such as “Please wait...”, etc.

Actions while performing calculations or image rendering processes

The user display and application activity should remain consistent and stable when calculations or image-rendering processes are being performed.

Multiple display format handling

Where the device and application can display in multiple formats (e.g. portrait / landscape, internal / external display), the elements of the application should be correctly formatted in all display environments, with the application displaying correctly without obvious errors in all formats.

If the application is supposed to be used ONLY in landscape mode, then the application should be forced so that it cannot be rotated to portrait mode. The same applies to applications intended to be run in ONLY portrait mode.

Differing screen sizes

Where the application is designed to work on multiple devices it must be able to display correctly on differing screen sizes.

Multiple format input handling

Where the device and application can accept input in multiple formats (e.g. external touch screen / external keypad / internal touch screen / internal keypad / QWERTY layout / 12-key layout and others), it should work correctly with all supported input methods.

Accelerometer/motion sensor responses

Where both the device and the application use accelerometer / motion sensor support, the response of the application to movement or change of alignment of the device should not impair use of the application, nor be likely to confuse the user. The application should change between portrait and landscape modes without confusing errors being displayed to user.

Spelling errors

The Application should be free of spelling or language errors unless they are part of a deliberate design concept.

Technical text errors

The text in the application should be clear and readable. The application should be free of technical text display issues such as: Text cut off / Text overlapping, and all text in each target language should be displayed without corruption, distortion or other display problems.

Problems to avoid are:

- Menu item text labels incorrectly aligned with cursor
- Button text label over-running the button area or truncated such that its meaning is not clear
- Text over-running or being truncated in other bounded text display areas (e.g. speech bubbles, user interface elements etc)
- Text not wrapping at the edge of the screen resulting in words being cut off
- Multiple pieces of text overlapping each other, or text overlapping user interface elements
- Text being cut horizontally.

Language

Correct operation

Ensure that the application works correctly with all appropriate languages and allows the user to select languages if appropriate, with the correct rendering.

Supported formats

Verify that date, time, time zone, week start, numeric separators and currency, are formatted appropriately for the implemented language's target country and supported throughout the application.

International characters and units of measurement

Ensure that the application accepts and displays all appropriate international characters and units of measurements correctly, according to the local market needs.

Performance

Responsiveness

Always update the user on progress. Render the main view and fill in data as it arrives. Always respond to the user input within 5 seconds. Users perceive a lag longer than 110-200ms adversely.

Suspend/resume

Where an OS environment supports 'suspend / resume', ensure that the application suspends and resumes at a point that does not impair the user experience.

Multi-tasking and effect on other functions

In a multi-tasking environment, remember to release used resources or functionality for other applications to use when not in use by that application.

An application should correctly handle situations where - following user input, or some external event (e.g. a phone call) - it is switched to the background by the terminal. Upon returning to the foreground the application should resume its execution correctly.

While in the background the application should not intrude in any way on the operation of other applications or handset functions, unless its explicit design is to do so and that is clearly explained in an accessible Help file.

When pauseApp()/hideNotify() is called by the system, the MIDlet should do the following:

1. Pause the application.
2. Save application state.
3. Release any resources the app won’t need while paused (like sound resources).

Upon they system calling startApp() should reallocate the resources it needs and renew the app from its saved state.

For apps which are meant to run in the background, it’s not necessary to do all of the above, but apps should at least stop the drawing loop as drawing while in the background is a waste of system resources.

Resource sharing – database

Where there are multiple applications that make use of a common database (for example, calendar synchronisation and calendar viewing), the database should be properly shared between those applications.

Other programming dos and don’ts

Don’t update widgets too frequently, or update your location unnecessarily or use Services to try to override users or the system as these adversely impact data usage and battery life.

But do share data to minimize duplication and let users manage how often they update, and whether or not they want to allow updates to happen at all.

Media

Application mute option

For applications with sound settings, there should be a Mute or Sound On / Off setting, unless the Application does not have Application mute facility by design it respects the settings of the handset volume buttons.

Settings statuses understandable

Where the application has settings options, ensure that the settings statuses are easily understandable at any stage in the application’s journey.

Saving settings

Where the application has settings options, ensure that the application saves all settings on exit and that restarting the application will restore the saved settings.

Specific functions

For applications with sound, ensure application sounds have specific functions and should not be over utilised - e.g. a game completing with a minute of random noise should be avoided.

Menu

Help and About

An application with user interface capable of displaying information to the user should contain standard Menu items Help & About (or equivalent information in a format easily found and understood by the user) to explain to the user how the Application works.

If it is clear that the application's purpose requires network coverage to operate, then it would be sufficient for the Help to be provided through a browser connection rather than being contained in the application. In the opposite case, where most functions of the application can be used while the device is offline, then the application should have Help that can be accessed without needing a data connection.

Menu items like Help and About should be presented on the main menu or other easily-found screen of the application.

About functions should contain the application name, application version number and author information.

Help should include the aim of the application, usage of the keys (e.g. for games) and other instructions. If the text of the help is too long, it should be divided into smaller sections and/or organized differently.

Help must be accurate and consistent with the application functionality and the handset specifics.

Valid actions

All application items that can be selected and/or changed by user should do what the application is intended to do.

Functionality

Functionality sanity check

All specific application functionality such as algorithms, calculations, measurements, scoring, etc. should be implemented correctly.

Application hidden features

The application should not introduce any hidden features: its functionality set should be consistent with the help and it should not harm the data on the device. However, the following hidden functions are OK: Cheat codes and the unlocking of an application to upgrade from a demo version to a full version.

Keys

Scrolling in menus

For an application with user interaction, when the keypad or other navigation device is used to scroll vertically and (if applicable) horizontally in the Main menu item list, there should be no adverse effect on the application. In addition, an application should be able to lock itself in a vertical or horizontal view if seen as important from application-use point of view.

Selection key

For an application with user interaction, pressing the primary selection key or device equivalent in the main menu item list should select the menu item with no unwanted effects on the application.

Text field scrolling

For an application with user interaction, the scrolling functions of the keypad or other navigation device in a text dialog (for example: About and Help) should scroll vertically and (if applicable) horizontally in the dialog.

Pause

For an application where time-sensitive immediate user intervention is needed, the application should support a pause feature in the areas of the application where this is applicable (for example in-game).

The user should - where necessary - be able to easily pause and resume the application.

All time-specific features of the application should be disabled at the time of the pause.

There should be a clear indication that the application is in a paused state.

There should be a clear indication of how the user can return from the paused state.

(You should use the potentially-available APIs for pause and continue methods if possible.)

Simultaneous key presses

For an application with user interaction, ensure that it copes with simultaneous key presses by not putting the application into an unusable or incomprehensible state by simultaneous key presses. Any error messages generated should be meaningful.

Multi key presses

For an application that supports multi key press actions (on a device that also supports this), they should perform as predicted and should not leave the application in an unusable state.

Device Specific Tests

Device opening and closing

For an application on a device with open / close functionality, ensure that it handles opening and closing of the device correctly while launching and returns to the same state before the interruption.

Stability

Application stability

The application should not crash or freeze at any time while running on the device.

Application behaviour after forced close by system

The application should preserve sufficient state information to cope with forcible close by the system. It should not lose any information that it implies would be preserved, nor become difficult to use subsequently, as a result of a forcible closure by the system.

Data Handling

Save game state

For an application where the user may exit part completed game or where a player high score value is identified, ensure that it can save its game state/high score table information into persistent memory.

Data deletion

Where an application has a function to delete data, it should indicate whether data will be permanently deleted or offer easy reversal of the deletion.

The user should always be required to confirm deletion of data, or have an option to undo deletion, to reduce risk of accidental loss of information through user error.

Security

Encryption

All sensitive information (personal data, credit card & banking information etc.) must be encrypted during transmission over any network or communication link.

Passwords

If an application uses passwords or other sensitive data, the passwords or other sensitive data should not be stored in the device and not echoed when entered into the application. Sensitive data should always protected by password. If possible, all stored password should be encrypted

With passwords, the desired approach is that the application shows which character the user selected and then changes that to an asterisk (*).

If the user is explicitly asked for permission, a password can be stored to the device memory.

It’s important to minimise the risk of access to sensitive information should the device be lost, by ensuring that no authentication data can be re-used by simply re-opening the application.

Once sensitive data has been entered, it should not be displayed in plain text anywhere in the application. However it is generally acceptable to have no more than 25% of a sensitive value displayed in plain text (e.g. 4 of the 16 digits of a card number) where this assists the user to distinguish between multiple cards or accounts.