Government mServices

Technical Guidelines

Date: 13th September 2016
Version: Draft 1.3
Department: TDED

Unclassified

Malta Information Technology Agency, Gattard House, National Road, Blata l-Bajda HMR 9010 Malta
Telephone: (+356) 21234710
Web Site: www.mita.gov.mt

Modification History

Version	Date	Author	Comments
Draft 1.0	06/05/2016	MITA- TDED	Initial Release
Draft 1.1	08/07/2016	MITA- TDED	Updates to section *3.4 Platforms Target platform versions*
			Inclusion of section *04.5Contact/ Feedback/ Suggestion/ Comments Form*
			Inclusion of *Appendix A - GovMT Launcher Functionality*
Draft 1.2	12/08/2016	MITA- TDED	Altered text in Appendix A – regarding the unique *URL scheme*
			Inclusion of APP package ID – *Reference Section 04.3.1*
Draft 1.3	13/09/2016	MITA- TDED	Appendix A – Update *SkipSplashScreen parameter set as optional*

Executive Summary

MITA intends to draft and articulate a holistic Mobile Government Strategy which will encompass the final governing direction in this construct. However, in order to further expedite progress in this regard, this document is being drafted to provide an initial set of enabling guidelines to create the opportunities for a number of quick wins in this regard - by delivering a number of mobile applications which have widespread impact on the wider citizenship.

In this context, for the immediate term, this document should act as:

- Catalyst for the design and implementation for a number of quick wins.
- Provide a baseline set of directions on how to engineer native, hybrid and web mobile applications.
- Provide a baseline set of governance measures with which these applications need to adhere to.

In summary, this document makes the following **key recommendations**:

1. The selection of which **deployment model** to consider (i.e. native, hybrid or responsive/form factor adapt) depends on a number of considerations – this document provides **guidelines** on when to adopt which.

2. The native build process will be **federated**[1] (in the sense there will be no central dependencies). Final native binary **signage** (under Government of Malta) and **publishing** (in the respective stores) will be performed and governed centrally by MITA.

3. Specifically, for **web based**, mobile application development which interact with back office services, the primary technology to use needs to be **Javascript**, with focus on **Node.js** and **MongoDB**.

4. Both Web apps and Native apps will capitalise on the same solution architecture including the use of core building blocks such as web services to deliver the required services, irrespective of the medium or channel used.

5. Within MITA, **Xamarin** is the preferred software development tool of choice at this point in time – the reasons relate to:
   - Provides the capability to have a single code base for multiple platforms
   - Leverage's the platform strengths for optimal user experience
   - Integrates well with our mainstream engineering practices and processes
   - Can capitalize on current skill set (C#) and development tools (IDE[2])

   MITA however will not mandate or restrict the use of tools/frameworks on 3rd party suppliers and/or contractors in the context of mobile app development.

6. App deployment to respective stores will be **governed centrally** by MITA
   - This does not inhibit the development (full lifecycle) of apps in a federation fashion
   - There will be a maintenance window for mainstream deployments – exceptions will be governed by a separate process
   - All published government mobile apps will be published under the Government of Malta

[1] This may require an upfront investment in a number of devices required for building and testing apps.
[2] Xamarin is now available with Visual Studio and will be open source in the coming months.

Table of Contents

01. Background

Mobile application engineering is inherently different from traditional desktop applications. Designing and engineering for mobile needs consideration of how and whether to exploit features such as positioning systems, motion sensors, cameras and microphones. Considerations for battery use, computing resources and network disconnections are also challenges that need to be dealt with at the initial stages of the design. For the sake of this document and until the final strategy document is in place, the key considerations and approaches that need to be undertaken in this construct are grouped into 4 dimensions, namely Data, Security, User Experience and Platform *(refer to Chapter 3)*.

02. Process Guidelines

The general engineering discipline is not yet geared fully and mature for mobile app development - this from a skills perspective, enabling platforms as well as internal/external processes perspective. Areas that require specific consideration are the SDLC, Security and general quality control, as well as standardization and streamlining of the development environments/processes to the maximum extent possible. Such challenges will become more amplified in the context where the mandatory federated approach for the engineering of native mobile apps is employed.

This section attempts to rationalize, a priori, a number of considerations as well as propose a baseline approach, from a **process perspective** (the software engineering part is discussed further on) to mitigate these challenges as we go along and mature in this construct.

02.1 Development approach

The following section provides an overview of how to identify those opportunities which can be exploited as apps as well as introduces a number of considerations from a process perspective in terms of the development tools to be considered, explains the basic deployment models that need to be considered as well as a discussion on how to best approach maintenance and updates.

02.1.1 Establish whether / which applications are good candidates

A simple decision framework is being described to help guide establish the appropriateness level of developing an app in the first place. These guidelines need to be applied in context to determine the relevance of app as well as identify a priori any potential risks that exist.

Perspective	Baseline
Is application graphics intensive?	As such these should be 'avoided' or heavily re-engineered, particularly when a native mobile app is being considered.
Is the app core functionality computationally taxing?	Complex compute should not be processed on the end user terminal. If this precludes the app from having the required functionality, it should be either re-engineered or not considered.
What level of enterprise service integration does it need if at all?	If heavy integration is required, then the best candidate deployment model should be an enterprise class web application with the pre-requisite of being form-factor adapt/ responsive.
Connected vs. disconnected use of app features	Mobile applications might need to cache/ store data to provide the user with limited functionality of the mobile app even when the device is not connected to the internet. The adoption of such an approach should be given due consideration given that it has an impact on the engineering of the mobile app, and indirectly on the costs associated with its development.
What is the designated security profile of the application in the context of identity and data processing?	Complexities surrounding data security and privacy in the context of mobile app development is further amplified vis-a'-vis how data is accessed and how it is stored/ cached (retained on the mobile). Depending on the security profile and classification of the data being accessed, the application might require authentication/ authorization to access such data, and possibly need to encrypt any data/ tokens used to access such information[3]. Retention of such information is also tricky and

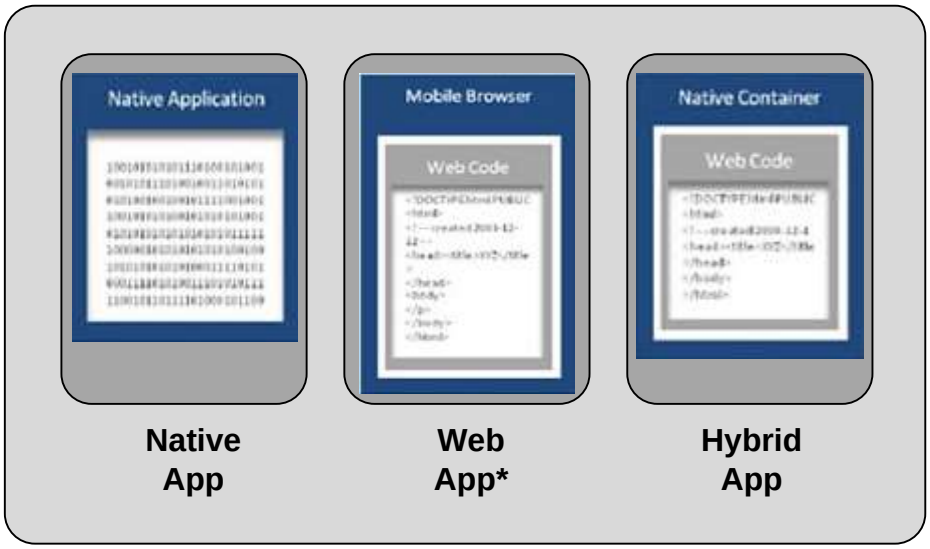
[3] IOS provides a local secure storage called KeyChain - https://developer.apple.com/library/ios/documentation/Security/Conceptual/keychainServConcepts/01introduction/introduction.html#//apple_ref/doc/uid/TP30000897-CH203-TP1

Perspective	Baseline
	might require further investigation as to how the app development should progress; a retention period for such data should be established and identified, and should be established on a case by case basis. Mobile apps which expose services requiring authentication of the citizen (via eID) or public officers (via CORP) should be 'avoided' and not considered for the immediate future (not a quick win). If such functionality is required then the development option should be an enterprise class web application, servicing such needs.
Payments	Applications that require the functionality of payments should be 'avoided' and not considered for the immediate future (not a quick win). If such functionality is required then the development option should be an enterprise class web application, servicing such needs.
What is the anticipated update/refresh lifecycle requirement of the app?	Identifying the refresh period of an app is crucial in determining whether it is viable to have it developed as a native/ cross platform mobile app or develop it as a web app (responsive web site). Such considerations should factor in the complete app deployment cycle which in some cases might take weeks to be officially published in a store. Periodically new updates to frameworks (used for the development and build of apps) should be considered to ensure currency of technology/ frameworks adopted by the published apps – deprecation and framework updates might have serious repercussions on the app features (particularly those dependent on plugins supplied by the framework usage).
App Support	The consideration to move to mobile app development/ production should also take into consideration the periodic review of the published apps themselves. Periodically a revision of the published apps should be undergone to ensure that they are still supported in terms of device support/ compatibility (new releases) as well as ensure that any APIs used by the app are still active (deprecated etc.).
What is the scope (in terms of extent and reach) of the app?	Apps should be developed to add (new) value added services or to extend existing functionality provisioned via traditional desktop/ web applications – it is important to note that this may imply a change in or addition to the existing business processes. Functionality which already exists and which is perhaps already available via a responsive web site should be carefully evaluated prior being 'replicated' onto a mobile platform as a native app. The scope, target audience and the frequency of use from the consumer's perspective is also something which needs careful consideration given the app development/ re-engineering costs incurred.
The use of certain features in mobile app	Depending on the mobile app business nature, certain features might be required to enhance the UX/ UI. Due consideration of such features needs to be carefully evaluated and scrutinized due to 1. Complexities surrounding the mobile app engineering to expose such features (such as the use of GPS); 2. Notifications: Depending on the type of notification required (including whether it is a broadcast or personalized) certain considerations related to the approach of how to identify users or rather subscribers, is something which needs to be looked into on a case by case basis. From a business perspective it is important to ensure that the use of such features will not introduce unnecessary complexities when alternative mechanisms (such as the use of SMS) might be more practical from a

Perspective	Baseline
	consumer perspective (requires data connectivity vs. none) and involve additional costs.
Legal Obligations, Data Persistence and Privacy	Depending on the business nature and target audience, the app might need to satisfy some legal obligations related to the retention of data. Why such data is being retained, its duration and the purpose should be clearly defined and agreed upon with the app owner. Specifically for native/ hybrid mobile app development, end users should be made aware about such information.

02.1.2 Development/Deployment models

A number of **development/ deployment models** are available when it comes to the final delivery and packaging (for distribution/consumption) of mobile applications. These are **Native App**, **Web App** and **Hybrid App**. A summary of the key features of each is provided below.



**Web Site / HTML5 responsive*

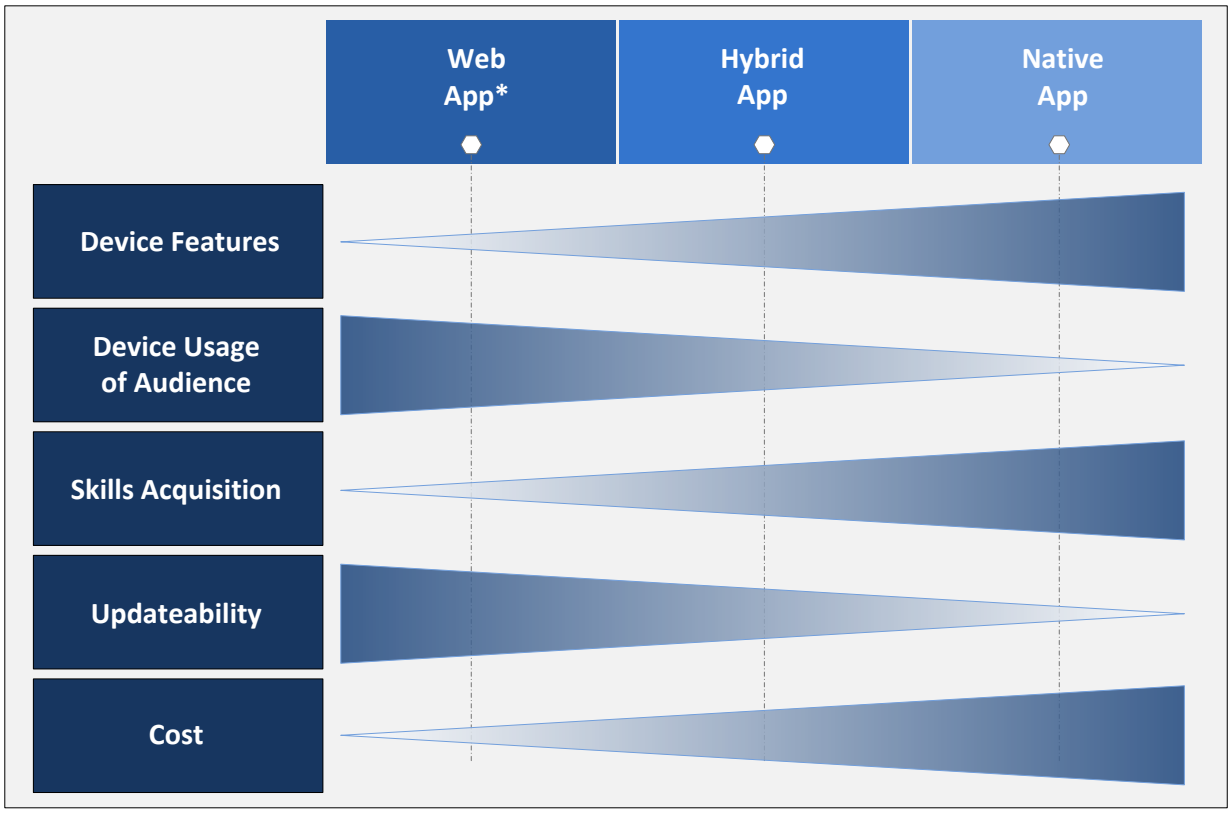
- **Native** apps are specific to a given mobile platform[4] (iOS, Windows or Android) using the development tools and language that the respective platform supports (e.g., Xcode and Objective-C with iOS, Eclipse and Java with Android, Visual Studio and C# with Windows). Native apps look and perform the best.

- **Web** apps use standard web technologies—typically HTML5, JavaScript and CSS. This is primarily a write-once-run-anywhere approach to mobile development creates cross-platform mobile applications that work on multiple devices. While reasonably sophisticated apps can be created with HTML5 and JavaScript alone, some vital limitations remain at this point in time such as functionality related to session management, secure offline storage, and access to native device functionality (camera, calendar, geolocation, etc.)

- **Hybrid** apps enable the embedding of HTML5 apps inside a wrapper native container (such as phonegap, intelXDK), combining the best (and worst) elements of native and HTML5 apps.

Figure 1 provides an overview of the pros and cons related to the challenges encountered depending on the type of development/ deployment model adopted.

[4] Latest Blackberry phones are shipped with android.



**Web Site / HTML5 responsive*

Figure 1: Pros and Cons

In general, a key perspective that should be followed during the decision making process is the consideration of **native User Experience (UX) performance versus portability** because these have broader consequences on the application **manageability**, **time** to **market** and **cost**.

In general, during the early project stages, the following key principles need to be considered because they will influence the final selection of the deployment model, namely:

1. As the need for **application portability increases**, the ability to maximize on performance and richness of the native mobile platform and UX decreases.

2. **Code re-use capability decreases** with **maximization of native features** – this in turn will increase the cost of the respective maintenance lifecycle as well as increase the time to market, significantly.

The advantages and disadvantages of each **deployment** approach (*Native/Hybrid/Web*) are very well documented, and summarized in the following table. Mobile development is a constantly moving target. Every six months, there's a 'new' version / flavour mobile operating system, with unique features only accessible with native APIs. The containers bring those to hybrid apps soon thereafter, with the web making tremendous leaps every few years.

Based on current technology, one of the scenarios described here is bound to suit specific requirements.

App / Devices Features	Native	Web	Hybrid
Performance	Faster	Slower	Slower
Distribution	Appstore	Web	Appstore

Connectivity	Online and offline	Mostly online	Online and offline
Development skills	ObjectiveC, Java	HTML5, CSS, Javascript	HTML5, CSS, Javascript
Offline storage	Secure file storage	Shared SQL	Secure file system, shared SQL
Graphics	Native APIs	HTML, Canvas, SVG	HTML, Canvas, SVG
Native look and feel	Native	Emulated	Emulated
Camera/ Video	Yes	No	Yes
Notifications	Yes	No	Yes
Contacts, calendar	Yes	No	Yes
Geolocation	Yes	Yes	Yes
Swipe	Yes	Yes	Yes
Pinch, spread	Yes	No	Yes
Multi touch	Yes	No	Yes (limited)

02.1.3 Software Engineering tools/IDE's

There are various development environments / tool chains which can be considered for use. It is safe to state the primary focus (not necessarily sole) is environments which target Windows Operating environments and mostly follow the Fat Client Server or Web Application/Services pattern. Mobile development introduces another build context and whilst traditional Integrated Development Environments (IDE) compliers are being evolved with such functionality, the most prevalent and popular environments within the industry are specific tools or tool chains. These include **Intel XDK, Xamarin, Kony and others**. Xamarin and Kony are believed to be industry leaders at this stage.

At this stage, in a MITA construct (based on market trends, current internal skill set and strategic direction) the development tool of choice should be **Xamarin**. The rationale for **Xamarin** being the initial preferred choice because it provides mature cross platform capabilities from a single code base – thus mitigating considerably issues related to maintenance lifecycles because of multiple code basis. This beside the tight integration with the prevalent IDE/Tool Chains/Operating environments in use within MITA's software development arm.

For **web** or **hybrid** apps which are heavily baselined in **HTML5**, **CSS** and **JavaScript**, the consideration for technologies such as **Cordova/PhoneGap, MongoDB** *(or platform specific alternatives)* and **Node.js**[5] is believed to be sensible. This approach allows developers to apply the same skill set in both front and back end constructs.
MITA however will not mandate or restrict the use of tools/ frameworks on 3rd party suppliers and/or contractors in the context of mobile app development.

02.1.4 Development/Testing deployment to respective app stores

In the context of MITA, a federated build process will be established. A federated Android/Windows/IOS development/testing environment (*based on the development environments described above*) will need to be set-up with any specific configuration requirements (including any investments required) established and made available a priori within each internal development team.

[5] Event-driven I/O server-side JavaScript environment.

Uploads to stores will be performed centrally and governed accordingly. Final native binary **signage** (under Government of Malta) and **publishing** (in the respective stores) will be performed and governed centrally by MITA (Mobile Initiatives Team).

02.1.5 Maintenance and updates

Updates cycles to mobile apps should be controlled. Whilst a degree of flexibility is to be in place for specific scenarios, a set of pre-defined update windows will be put into place and developers will be expected to schedule deployment to app stores within this schedule.

Updates must be concurrent with non-mobile app counterpart (such as web site and back office process) to ensure service delivery coherency across different consumption channels. This becomes a critical consideration in the context of the limited ability to influence the publishing timeframes of mobile apps to respective stores, especially when dealing with IOS variants.

03. Software Engineering Guidelines

There are 4 key dimensions that need to be looked into considerably differently when developing mobile apps when compared to traditional client server approaches. These dimensions, Data, Security, UX and Platform, are discussed in the following sections with an associated set of guiding principles provided to ensure a coherent and sound approach and which should form a baseline pattern for the first wave of mobile apps – keeping in mind the opportunity for future evolution whilst still remaining relevant.

03.1 Data

Online/offline operation – The mobile experience would be badly affected if it depends on persistent internet connection. Mobile apps need to leverage local caching mechanism to deliver a smooth user experience while negotiating "dead-zones" and network signal variance.

03.1.1 Persistence and Privacy[6]

Data persistence or caching allows users to use the app (possibly with limited features) while their connection to the World Wide Web is down. Data persisted on the device should be considered and addressed on a need basis. Depending on the security and sensitivity of the data related to the particular scenario, the developed application should resort to encryption to store sensitive data, and decide which best storage mechanism to adopt (session or otherwise). Data persisted on the device should have a retention period set, to ensure that data is retained for the intended scope (for example for the duration of the active session or time period) and for a predefined period of time, thus reducing risks associated with data security.

Depending on the scenario/ target audience of the mobile app there might be instances which mandate abidance to/ with a set of rules/ legal acts. Such as the case of the enforcement regulations concerning children's online privacy, referred to as Children's Online Privacy Protection Act (COPPA[7]).

03.1.2 Latency-design for intermittent connectivity

Mobile applications should assume that the connection to any internet service is unknown, thus handle any failure to connect to an external service. In such cases the application should deny access to specific features which require such access. Depending on the case and the application features, the use of cached data might be considered to allow smooth running of the application in such

03.1.3 Push & Sync

Apps should refrain from being intrusive on any level. They should offer the user to block any notifications and updates, informing the user with the due implications. Apps should support automatic, manual and hybrid refresh functionalities as endorsed by the respective platform vendors, depending on the nature and features provided by the mobile app. Data synchronization depends on the nature of the application – data can be refreshed when the application is launched or through the provision of data refresh/ sync features, initialized directly by the user.

03.1.4 Web Service API's

Mobile apps should leverage the use of **REST** based web service APIs to exchange data from/ to the mobile device. This approach would allow users to multitask within the app itself, optimizing user experience while the data is being transferred. These web service APIs are utilized for all end-user applications, irrespective of the medium so as to ensure that there is seamless (data) conformance between the mobile app (native or hybrid) and the web app version (responsive web site).

[6] http://www.gsma.com/publicpolicy/wp-content/uploads/2012/03/gsmaprivacydesignguidelinesformobileapplicationdevelopmentv1.pdf

[7] COPPA - http://www.iubenda.com/blog/guide-coppa-mobile-apps/

03.2 Security

The general guidelines reflected in the document published by ENISA titled **Smartphone Secure Development Guidelines for App Developers**[8] should be followed. Furthermore, the following considerations also apply.

03.2.1 Encryption at rest and in transit

As outlined in the previous section (4.1), sensitive data persisted on the device's local storage and during transit should be encrypted. Ideally sensitive information (such as passwords) is not stored on the device. In case passwords need to be stored on the device, the encryption and key-store mechanisms provided by the mobile OS can be used to securely store passwords, password equivalents and authorization tokens. Instead of passwords consider using authorization tokens that can be securely stored on the device (as per the OAuth model). The tokens should be time bounded to the specific service as well as revocable (if possible server side), thereby minimizing the damage in loss scenarios.

03.2.2 Authentication

In the case of authentication various protocols like OAuth2 can be used to securely authenticate/ authorize the user to consume the web service APIs servicing the mobile app. Depending on the type of service/ data being consumed, appropriate authentication mechanisms should be used. Session management is another aspect which needs careful consideration to ensure that any requests from the device are seamless and legitimate to the authenticated user.

Mobile apps which expose services requiring authentication of the citizen (*via eID*) or public officers (*via CORP*) should be **'avoided'** and not considered for the **immediate future** (not a quick win). If such functionality is required then the development option should be an enterprise class web application, servicing such needs.

03.3 User Experience

In order to optimize and deliver an intuitive app, it is important to identify and limit the scope of functionality provided by the app – this means that the apps delivered should ideally focus on providing specific, focused services as opposed to mimic and/or replicate all the functionality provided by the entity's core business application/s.

User experience is heavily influenced and quality controlled by the respective mobile platform vendor. In this respect, first and foremost, the engineering roles must acquaint themselves with a number of guidelines presented by the respective platform[9] vendors. It is important to outline that although there are commonalities between the respective stores, there might be some specific requirements pertaining to individual stores.

Initial design considerations should be performed via **wireframes** or **mockups** using tools such as **Balsamiq**, **Mockingbird**, **Visio** etc. These allow the designer to quickly design UX without having to worry about the actual UI design. Key considerations to be kept in mind are:

1. Design based on **device capability** and be able adapt the content to be delivered.
    a. Content scaling to optimize use of screen real estate and resolutions.

[8] https://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-applications/smartphone-security-1/smartphone-secure-development-guidelines

[9] **Apple**: http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html; **Android**: http://developer.android.com/design/index.html; **Windows**: Design library for Windows Phone - http://msdn.microsoft.com/en-US/library/windowsphone/design/fa00461b-abe1-41d1-be87-0b0fe3d3389d(v=vs.105).aspx

   b. Video formats need to be adapted to the device supported formats Interaction mechanisms need to be customized based on device capabilities.

2. **Simplicity** – An effective mobile layout needs to simple. The information presented needs to be structured into a layout that minimizes scrolling and the need for zooming. For example, a single column layout is generally preferred to avoid cluttering and horizontal scrolling.

3. **Navigation** – Mobile applications need to provide an easy way of navigating between 'pages/tabs' and make optimum use of the device capabilities.
   a. The navigation mechanisms in touch based smartphone are primarily icon and tab driven, while the navigation is menu driven for devices with keyboards. Reducing the number of touches to perform an action is necessary to enhance user experience.
   b. Minimize gestures/input– adopt OS specific conventions.

4. **Performance** – Mobile apps need to provide fast response times to the user. This can be achieved by keeping their size small, local caching and usage of background processing to provide a seamless experience.

5. **Accessibility** – Given that accessibility guidelines are platform specific it is imperative that the mobile app (native/ hybrid) is also available as a responsive web application which is designed with accessibility in kind. This approach would ensure that the mobile app (native/ hybrid) is also available as a web app (running on the mobile device web browser as a responsive web site) providing the necessary mechanisms to make it accessible to all intended users. Bundling separate modules, providing easier interaction with the application, could also be an option of consideration.

03.4 Platform

The development of mobile apps should target the mainstream, latest vendor platform versions particularly:

- Android 4.0 and above
- iOS 6 and above
- Windows 8

04. Other considerations

04.1 UI Design – Branding

A degree of consistency in terms of look and feel is considered fundamental. MITA shall provide a reference guideline to aid developers achieve a coherent branding approach, specifically for the development of mobile apps. It is advisable to reference the targeted /leading platform vendors' UI design guidelines in order to ensure the mobile app is compliant with the respective store/s requirements.

Key areas that need a degree of standardization in this construct are:

1. **Splash Page** – A splash screen for all government published mobile apps shall be designed so as to ensure a common user experience when loading government mobile apps.

2. **Logos** – Mobile apps should refrain from displaying the logo on various screens. Users need to clearly distinguish the icon from other apps in the launch screen, possibly show it again on a splash screen and hide it throughout other screens.

3. **Icons and Images** – All images and icons should be tailored to mirror the native look and feel of the respective platform.

4. **Colour Scheme** – The main colour scheme should be defined per application. Colours should be used to enhance usability rather than define it. This makes it easier to cater for a number of borderline cases such as (but not limited to) colour-blindness and using the app in direct sunlight. The schemes used for the mobile app should be similar to the colour schemes used by the government/ department/ entity website.

5. **Font** – All government mobile apps should follow the standard font types and sizes identified and established in the mobile app branding guidelines. The latter guidelines will be based on the major targeted platforms' guidelines.

04.2 Testing

Because the deployment lifecycle is much more taxing than typical fat client server and even more web based application development, testing and quality control become a fundamental requirement. The development team shall design, develop, build and test the full functionality of the mobile app, prior submitting the application for publishing (to the respective app stores) which will be done by MITA to sign the application under Government of Malta.

04.3 Stores

There are various ways how government mobile apps can be distributed. Two popular methods are via app stores, either public or private. The decision of which store should be used depends on a number of considerations which typically include aspects surrounding security (for the type of service/ app being provisioned), consumer reach and the level of app management required. In the context of the current maturity landscape as well as objectives related to app publishing (at least for the immediate future), the public app stores are potentially the most viable solution – cheap to use, leverage a model that targeted consumers (citizen) are familiar with and is less intrusive (i.e. do not require consumers to connect to a different 'corporate' app store). MITA shall centrally govern the publishing of the respective apps in the public stores, in order to sign and certify the application as a legitimate government mobile app published by the Government of Malta.

It is important to note that the public app stores are not designed to provide device management required to deploy/ push apps on restricted devices – hence restrictions to use the app should be

handled by the app UI. Such stores typically provide basic statistical features including but not limited to device types, number of downloads, ratings etc.

It is being envisaged that each developed app should ideally (and as a minimum) measure/ track user activity. Typical user activity logging might include login/ logout events, authentication issues (exceptions) and methods of authentication, method of access (WiFi/ 3G), and duration of activity within app. Retention of such activity logs, will differ according to the applied scenario and app in question.

04.3.1 Publishing the APP with a unique APP Package ID

The mobile apps that shall be published through MITA, shall follow the following naming convention

mt.gov.*<procuring entity>*.*<application name>*, thus ensuring that the package ID is unique and conforms with a naming convention which is associated with the Government of Malta.

Example of Package ID include:

App Package ID	Sector/ Ministry/ Entity
mt.gov.mita.appname	MITA
mt.gov.opm. appname	OPM
mt.gov.gozo. appname	Gozo
mt.gov.energy. appname	Energy
mt.gov.health. appname	Health
mt.gov.educ. appname	Education

Note: should the respective App need to be included in the *Government Launcher* mobile app – refer to **Appendix A** for additional details.

04.4 Languages

The majority of applications, that will be developed showcasing a number of government mservices, will use English as their default/ preferred language of choice. Although in most cases English will suffice, it is envisaged that there are cases where the mobile app (or responsive web site) might need to support at least an alternative language – Maltese. This rationale conditions the landscape of the mobile app development in that any engineered apps should take into consideration localization aspects form the very start of their development, thus providing the ability for the mobile app to change the apps base language with minimal or no effect to the source code. Typical engineering considerations and prerequisites supporting such rationale include:

- **Externalisation of resources** - one of the very first steps in the app localization process is to externalise resources (language files) – each language having a separate language resource file. Various leading platform vendors provide details and toolkits on how to approach this matter. *Note*: Details related to the app published in the app store should also be taken into consideration, however the preferred language for that should be English.

- **Layout** – the design should take into consideration the length of words and sentences in languages other than English. Such considerations might include spacing, left-to-right and right-to-left support, as well as other factors – hence a flexible layout that scales according to the language used is something which needs careful consideration. Other considerations linked to the design and layout of the app relate to the system-provided formatting methods for dates, times, number and perhaps currency.

The option to choose the language to use in the mobile app will be determined from within the app itself – i.e. it will not be determined or chosen through the device's language settings.

04.5 Contact/ Feedback/ Suggestion/ Comments Form

Each published mobile app should provide a mechanism of allowing citizens to provide feedback/ suggestions to the service owner. In this context each published app should feature a simple form/ screen allowing users to exchange such feedback/ comments related to the mobile app itself.

05. Appendix A - GovMT Launcher Functionality

The mServices on smartphones will be accompanied by an application acting as a central hub for all government applications. The application will allow users (citizens) to outline the respective mServices listed under specific government sectors and provide the ability to launch the native app or the responsive website relating to the chosen service.

Specifically for applications (native and hybrid), the launcher will make use of deeplinking to launch the app installed on the device. Thus, each application submitted to the store will need:

1. **A unique URL Scheme**

   The convention being adopted will assimilate "**govmt*Name***", where *Name* is the name of the app.

   Note: MITA will only need the *App Name*[10] (supplied in the table below) to generate the URL Scheme. Once this has been generated the URL scheme shall be supplied back to the owner/ supplier to be configured in the respective app.

2. **Allowing remote intents**

   The application must request the appropriate permissions (from the OS) allowing the launcher to trigger the remote intent. This can be done in various ways, depending on the tools being used. In general, it consists of a simple checkbox for the appropriate permissions.

3. **Inclusion of *SkipSplashScreen* parameter *[OPTIONAL]***

   A query parameter will be provided (SkipSplashScreen) from the invoking launcher application to indicate that the target application should skip the splash screen and route users directly to the content page. Thus, the app must be kept stable if the splash screen is skipped. The launcher application will send *SkipSplashScreen* to tell the target launching application not to load the splash screen. In the absence of such a parameter, the normal application flow is to be triggered.

   When invoked through the launcher the app will skip the splash screen (*SkipSplashScreen*), thus a loading page might be required to allow for any data synchronisation required at that stage by the respective app.

In view of the above, in order to publish the app in the respective stores MITA Mobile Initiatives Team would need the following details.

[10] The *App Name* can be supplied in advance of the screen shots and icons required – i.e. prior the completion of development of the actual app.

Info required	Required for
Application (Display) Name *This will be the same name used within the launcher*	**Publishing in stores**
Icons/ Screens (for all stores as per store requirements) zipped *Subdivide icons and screen shots in respective OS platforms folders*	**Publishing in stores**
Description of app (for respective stores)	**Publishing in stores**
Service owner email to show in stores	**Publishing in stores**
Service owner website link to show in stores	**Publishing in stores**
Terms of Use / Privacy Policy *Such as https://www.gov.mt/en/Pages/Terms-of-use.aspx* *Note: To include any additional details related to the specific app*	**Publishing in stores**
Platforms (including minimum version and any known differences) targeted by the application *Android, IOS, Windows*	**Publishing in stores/ Launcher**
Responsive Website link	**Launcher**
Government Sector *Supplied by OPM*	**Launcher**
App Project Manager Contact Details *Supplied by Ministry/ Entity*	**Internal back office use**
Technical Lead Contact Details *In case of issues during publishing / bugs etc.*	**Internal back office use**
Development Tools/ Technologies used *Brief description of tools/ technologies used*	**Internal back office use**
Planned refresh cycles *Visibility of the plans for future updates to the app itself*	**Internal back office use , publishing**

Kindly send the above information (and any queries) to mobile-app.mita@gov.mt at your earliest.

Following the submission of the above details MITA shall provide the respective application *Deeplink URL scheme* and any other details concerning the publishing process of the application (such as package name).