Title: **Technical Assessments of UNRWA Mobile apps**

Date: June 2020

Versions

Version	Create Date	By	Notes
1.0	29.06.2020	Yasmeen Helles	Created the first version of the document
1.1	30.09.2020	Adham Enaya	Reviewed and enhanced version 1.0

Table of Contents

Introduction

This document assesses the current releases of UNRWA Mobile applications for two platforms Android and iOS and then it proposes enhancements of application in many sides such as user interface (UI), source code, architectural presentation pattern if used in the current release, Application Program Interface (API), local Database used, third-party library and Localization of application (Support to many languages).

1. Contact Mobile Application

1.1Introduction

This section assesses the current release of Contact Mobile application for two platforms Android and iOS and then it proposes enhancements of application in many sides such as user interface (UI), source code, architectural presentation pattern if used in the current release, Application Program Interface (API), local Database used, third-party library and Localization of application (Support to many languages).

1.2Android Platform

1.2.1 User Interface

The current design for the current release of contact mobile application is very simple and flat so we would to enhance it and make more interaction between it and user using a material design which is framework works with all the android platforms and is known for a flat graphical interface, takes care of the user experience and It deeply interacts with a user. Material Design has responsive and meaningful interactions. Google's new visual design framework is flat, elegant, and vibrant creating a unified Android experience.

1.2.2 Source Code

Kotlin is the language used in the current version which is efficient and presents a familiar development tooling that is meant to boost developers' productivity, good compiler and provides an enhanced run-time performance so we don't need to change anything in this side.

1.2.3 Architectural Presentation Pattern

The current release of the application doesn't use any architectural pattern so the software components aren't grouped in meaningful way.

There are three types of most commonly used architectural UI design patterns such as MVC, MVP, and MVVM. MVP is an abbreviation of Model-View-Presenter. MVC is an abbreviation of Model-View-Controller. Whereas MVVM stands for Model-View-View Model. Here are the benefits of these patters:

- It plays a significant role in developing an application as best practices formalize them that are loosely combined
- Easier to test & maintain and facilitate reusable object-oriented development.

- These architecture patterns are designed to do moderate the complex codes and make the UI code cleaner and manageable.

1.2.4 Application Program Interface (API)

- Contact Mobile App use the third-party library such as Retrofit with Gson because it can make your app simple and clear to write REST code. Retrofit is used to manipulate the HTTP web service API communication, Gson is used to parse web service API return data in JSON format.
- Retrofit can configure which converter is used for the data serialization.
- Retrofit uses the OkHttp library for HTTP requests.

So, we would to keep depending on it in the application and we wouldn't change it.

1.2.5 Third-party Library

In the current release of contact mobile apps use a variant of the third-party libraries such as Retrofit and we would use library-related UI and Design of application.

1.2.6 Local Database

SQLite Database used in a current release which is primarily geared toward storing all data on devices and moving away from common client-server architecture. Thousands of mobile apps were built using this tool. If we want to use a new tool rather than SQLite, we can use Realm that is absolutely free of charge. Realm appeared much later than SQLite. Its development began in 2010, this tool works on its own persistence engine. The realm is widely used for mobile development, especially among newbies in Android development. This DB works seamlessly with Java, Swift, Objective-C, Xamarin, and React Native. Now, this tool continues to gain popularity among mobile developers.

1.2.7 Localization of application

App localization is the process of changing and refining an app to appeal to all users in different languages. You want to make sure that your app is as appealing and easy to use in Arabic as it is in English. We want to make sure if the contact mobile application supports Arabic and English Language and add a new feature to control and change application language.

1.3 Android Enhancement Proposal

Item	Current Release	Proposed Release	Benefits of proposed Release
User Interface (UI)	Flat design	We would use material design	• Material Design is known for flat graphical interface. • Takes care of the user experience and it deeply interacts with user. • Material Design has responsive & meaningful interactions.

			Google's new visual design framework is flat, elegant, and vibrant creating a unified Android experience
Architectural Presentation Pattern	Not used	We would use one of the design patterns types mentioned ealier.	• play a significant role in developing an application as best practices formalize them that are loosely combined • Easier to test & maintain and facilitate reusable object-oriented development. • These architecture patterns are designed to do moderate the complex codes and make the UI code cleaner and manageable.
Local Database	SQLite	Realm	• Realm that is absolutely free of charge. • Faster in fetching records than SQLite

1.4iOS Platform

1.4.1 <u>User Interface</u>

Attractive and interactive user interface designs are all the important for smartphone, tablet and other tech device users, therefore the appearance of an application is of most importance when it comes to how popular it is going to be. The more well designed, interactive and simple to use an app's user interface design is, the more downloads it will have.

1.4.2 <u>Source Code</u>

We want to rearrange source code and make sure it is following the standard and best practices when writing readable code.

1.4.3 <u>Architectural Presentation Pattern</u>

The same thing in the iOS version that doesn't use any architectural pattern so the software components aren't grouped in a meaningful way.

1.4.4 <u>Application Program Interface (API)</u>

Application communicate with server and fetching data using URLSession which is provide an API for downloading data from and uploading data to endpoints indicated by URLs. When fetch some data using URLSession, here's what we're going to do:

- Set up the HTTP request with URLSession
- Make the request with URLSessionDataTask
- Quickly print the returned response data
- Properly validate the response data
- Convert the response data to JSON

We propose to rely on 3rd-party libraries, such as Alamofire which is have many features like:

- Chainable Request / Response Methods
- URL / JSON Parameter Encoding
- Upload File / Data / Stream / MultipartFormData
- Download File using Request or Resume Data
- Authentication with URLCredential
- HTTP Response Validation
- Upload and Download Progress Closures with Progress
- Dynamically Adapt and Retry Requests
- Network Reachability

1.4.5 Third-party Library

In current release of contact mobile app doesn't use any third-party library.

1.4.6 Local Database

SQLite Database is used in a current release which is primarily geared toward storing all data on devices and moving away from common client-server architecture. If we want to use a new tool rather than SQLite, we can use Core Data or Realm. The following is described for each one:

SQLite:

- SQLite is the most used database engine in the world and its open source as well.
- Independence from a server.
- It implements a transactional SQL database engine with no configuration.
- Safe access from multiple processes and threads.
- Stores data in tables with one or more columns that contain a specific type of data.

Core Data

- Uses more memory than SQLite
- Uses more storage space than SQLite
- Faster in fetching records than SQLite.

Realm

- Realm was designed to be faster and more efficient than the previous database solutions. It is available in Objective-C and Swift, and it's designed for iOS and Android.
- It's absolutely free of charge,
- Fast, and easy to use.
- Unlimited use.
- Work on its own persistence engine for speed and performance.
- Realm is very easy to install and faster to work with compared to SQLite and Core Data. Also, the database files are shareable among iOS and android.

1.4.7 Localization of application

The same thing in the iOS we want to make sure if the mobile application supports Arabic and English language and add a new feature to control and change application language.

1.5 iOS enhancement proposal

Item	Current Release	Proposed Release	Benefits of proposed Release
User Interface (UI)	Use the default interface	We will use a customized interface and make design consistent	• Make application more interactive and attractive
Architectural Presentation Pattern	Not used	we will use one of the design patterns types	• play a significant role in developing an application as best practices formalize them that are loosely combined • Easier to test & maintain and facilitate reusable object-oriented development. • These architecture patterns are designed to do moderate the complex codes and make the UI code cleaner and manageable.

Application Program Interface (API)	URLSession	Alamofire	• Chainable Request / Response Methods • URL / JSON Parameter Encoding • Upload File / Data / Stream / MultipartFormData • Download File using Request or Resume Data • Authentication with URLCredential • HTTP Response Validation • Upload and Download Progress Closures with Progress • Dynamically Adapt and Retry Requests • Network Reachability
Local Database	SQLite	Core Data or Realm	• Faster in fetching records than SQLite

1.6 Project Details and Information

1.6.1 Project General Overview	
Project Name	Staff Contact information
Project Lead	Ahmad Shatarat
Project development Start and end dates	
Project Status & completion percentage	completed
Supported Platform (Android, iOS, Web control panel, API, 3rd party integration)	Android & iOS + web application
Development team size	3 (1 iOS, 2 Android)
Current support team size	1 iOS, 1 Android
Operation field (HQA, GFO, WBF, JFO, LBF)	All fields
Number of users	

1.6.2 Project Documentations	Attach all relevant documents
Software requirements specification	N/A
Software design document	N/A
User interfaces (wireframes)	N/A
Description of key components (i.e. algorithms) (if exits)	
Mobile database design document	Offline DB
Backend database design document	N/A
API design document	N/A
1.7 Development Details	
1.7.1 Backend Control panel	
Web technology stack	.Net
Database technology	Sql server
Source code	TFS
1.7.2 Backend API	
Service Architecture (API, SOAP…Others?)	API
Technology used	Retrofit
1.7.3 Android	
Supported devices (mobile, tablet)	Mobile
Target OS & SDK version	Android SDK Platform (API level 26)
Programming language	Kotlin
Key libraries & frameworks	Retrofit, Kotlin, hdodenhof/CircleImageView, Firebase-Messaging

Source code	
Project Architecture	None
1.7.4 iOS	
Supported devices (iPhone, iPad)	Yes
iOS deployment target	iOS SDK (iOS 11.1)
Programming language	Swift
Key libraries and frameworks	None
Source code	
Project Architecture	None
1.8 Testing and Maintenance	
Test cases and test results	
Number of open incidents	
Number of pending requirements	
1.9 Integration with other systems	
What systems is the application integrated to	
Integration architecture design	
1.10 Deployment	
Android deployment location (play store, our server, by email, other)	Intranet
iOS deployment location (apple store)	N/A
Backend deployment location(server location)	
Deployment plan	
Deployment keys, token?	

1.11 **Other Comments**

2 ED/VTCs Students Mobile App

2.1 Introduction

This document is to compare between current release of VTCs mobile application and the proposed enhancements of application in many sides such as user interface (UI), source code, and architectural presentation pattern if used in current release, Application Program Interface (API) and third-party library.

2.2 User Interface

We would to enhance the design screens and components of current release using material design for many reasons such as:

- Google's Material Design guidelines, which third-parties are encouraged to use, are designed so that each app has a consistent layout, look and feel.
- Material design framework works with all the android platforms and is known for flat graphical interface. Google's new visual design framework is flat, elegant, and vibrant creating a unified Android experience.

Top four benefits of using Material Design

Branding

Material design uses flat and light objects. It requires just two to three colors to brand the entire APP. material design is easy to brand the APPs you develop.

User Engagement

It is very important to engage users. You have to constantly communicate with users through your APP. User Engagement is one of the key factors to success. 35 % of the Smart-Phone APPS are used just once in the life time of the user who installed the APP. Material design provides a visual language between user and you.

BETTER UI/UX

UI/UX is medium of communication between you and the User. Bad UI will make the user to uninstall your APP. Material Design takes care of the user experience. It deeply interacts with user. Material Design has responsive &meaningful interactions. You can delight the user by integrating material design in your app.

Cost Effective

Google Introduced material design for web interfaces too. Just One prototype design can be used for web, mobile, and APP.

This reduce the overhead cost of designer, and UX developer for multiple platforms. It offers lot of free resource available such as icon, color pallets, templates and wire frames.

2.3 Source Code

Java is the language used in the current version, but today most programmers are directed towards the Kotlin language because of its many advantages.

Some of the main goals of Kotlin's development team are rapidly becoming some of the biggest advantages of using Kotlin over Java. Kotlin is:

- Efficient and presents a familiar development tooling that is meant to boost developers' productivity
- A good compiler
- A seamless integration with the existing infrastructure (Kotlin's compatible with all Java frameworks and libraries, and it's designed to integrate easily with Marven and Gradle build systems, as well)
- It provides an enhanced run-time performance.

2.4 Architectural Presentation Pattern

In current release of application doesn't use any architectural pattern so the software components aren't group and views aren't useful. There are three types of most commonly used architectural UI design patterns such as MVC, MVP, and MVVM. MVP is an abbreviation of Model-View-Presenter. MVC is an abbreviation of Model-View-Controller. Whereas MVVM stands for Model-View-View Model.

All of these design patterns

- Play a significant role in developing an application as best practices formalize them that are loosely combined
- Easier to test & maintain and facilitate reusable object-oriented development.
- These architecture patterns are designed to do moderate the complex codes and make the UI code cleaner and manageable.

2.5 Application Program Interface (API)

System use REST API where create JSON API web service project which will have the Interface and Handler web service. This API will connect with RestAPI class
Mobile application the request is sent through the RestAPI, we are calling the REST API from an android application. In the server side, the API service reads data from the database and sends the response in JSON format. After receiving the response, the Android application displays the data of items in UI by parsing the JSON data.
Then create the URL of the post to be performed, followed by establishing the Http connection. Next, set the HttpURLConnection options to perform the Get and read return JSON data from connection. Finally, we just check if the post/get was successfully sent/received by the server's reply

- We propose use the third-party library such as Retrofit with Gson because it can make your app simple and clear to write REST code. Retrofit is used to manipulate the http web service api communication, Gson is used to parse web service API return data in JSON format.
- In Retrofit can configure which converter is used for the data serialization
- Retrofit uses the OkHttp library for HTTP requests.

Item	Current Release	Proposed Release	Benefits of proposed Release
User Interface (UI)	Flat design	We would use material design	• Branding • User Engagement • BETTER UI/UX • Cost Effective
Source Code	Used Java Language	We will use Kotlin	• efficient and presents a familiar development tooling that is meant to boost developers' productivity • a good compiler • a seamless integration with the existing infrastructure (Kotlin's compatible with all Java frameworks and libraries, and it's designed to integrate easily with Marven and Gradle build systems, as well) • provides an enhanced run-time performance
Architectural Presentation Pattern	Not used	we will use one of design patterns types	• play a significant role in developing an application as best practices formalize them that are loosely combined • Easier to test & maintain and facilitate reusable object-oriented development. • These architecture patterns are designed to do moderate the complex codes and make the UI code cleaner and manageable.
Application Program Interface (API)	REST API with JSON Parser Class	We would to use Retrofit with Gosn	• I can make your app simple and clear to write REST code. • Retrofit is used to manipulate the http web service api communication • Gson is used to parse web service API return data in JSON format. • In Retrofit can configure which converter is used for the data serialization

Third-party Library	Not used	We propose use some third-party libraries such as Retrofit with Gson.	• They ease the developer's job through code re-use

2.6 Third-party Libraries

In current release we don't use any third-party library so we suppose to use it because it provides many benefits.

Top third-party libraries can be used in Android development are:

- **Font Awesome -** Font Awesome gives you scalable vector icons that can instantly be customized - size, color, drop shadow, and anything that can be done with the power of CSS.
- **Material Design icons -** material-design-icons are the official open-source icons featured in the Google Material Design specification.
- **Universal Image Loader for Android -** aims to provide a reusable instrument for asynchronous image loading, caching and displaying.
- Volley, developed by Google engineers, used for simple http requests.
- Retrofit, by Square. Competes with volley, but volley can be used even to download images, which retrofit is not developed for.
- Picasso, by Square. Peer library of Retrofit specifically for image downloading.
- OkHttp again by Square, which is even used in Android HttpUrlConnection from KitKat.
- Gson, by Google. To convert Java objects into Json and back.

2.7Summary of Comparison

3 Electronic Mother and Child Health (eMCH)

3.1 Introduction

Electronic Mother and Child Health (eMCH): This mobile application is a digitalized version of the paper Mother and Child Health Handbook distributed to all Palestine refugee mothers attending UNRWA health centers. The application was jointly developed by UNRWA and JICA (Japan International Cooperation Agency). It is now designated as the Electronic Mother and Child Health Booklet (e-MCH) (كتيب صحة الأم والطفل الإلكتروني). This application was developed to help you in getting access to your electronic maternal and child healthcare information stored in UNRWA's electronic health system, and it allows you to view your record and the records of your children with all healthcare information and data including medical, nursing or other professional health care advice, diagnosis, preventive services and results of diagnostic services. In addition, it aims to educate you on issues that are important for your health and the health of your children.

This document assesses the current release of application on two platforms (Android and iOS) and proposes enhancements if needed.

3.2 User Interface

3.2.1 <u>Android Version</u>

Application in Android platform include libraries such as support library package that contains the largest set of APIs compared to the other libraries, including support for application components, user interface features, accessibility, data handling, network connectivity, and programming utilities.

We would add some enhancement on the design screens and components of current release using material design for many reasons such as:

- Google's Material Design guidelines, which third-parties are encouraged to use, are designed so that each app has a consistent layout, look and feel.
- Material design framework works with all the android platforms and is known for flat graphical interface. Google's new visual design framework is flat, elegant, and vibrant creating a unified Android experience.

3.2.2 <u>iOS Version</u>

The Interface Builder editor within XCode makes it simple to design a full user interface without writing any code. Simply drag and drop windows, buttons, text fields, and other objects onto the design canvas to create a functioning user interface.

Because Cocoa and Cocoa Touch are built using the Model-View-Controller pattern, it is easy to independently design your interfaces, separate from their implementations. User interfaces are actually archived Cocoa or Cocoa Touch objects (saved as .nib files), and macOS and iOS will dynamically create the connection between UI and code when the app is run

A complete iOS app is composed of multiple views through which the user navigates. The relationships between these views are defined by storyboards, which show a complete view of your app's flow.

Interface Builder’s storyboard designer makes it easy to create and design new views, and chain them together to create a complete user interface that’s ready for your custom code.

3.3 Source Code

Java is the language used in the current android version, but today most programmers are directed towards the Kotlin language because of its many advantages.

Some of the main goals of Kotlin's development team are rapidly becoming some of the biggest advantages of using Kotlin over Java. Kotlin is:

- Efficient and presents a familiar development tooling that is meant to boost developers' productivity
- A good compiler
- a seamless integration with the existing infrastructure (Kotlin's compatible with all Java frameworks and libraries, and it's designed to integrate easily with Marven and Gradle build systems, as well)
- Provides an enhanced run-time performance.

In **iOS Platform** swift language version 4.2 used and the last version from apple is swift 5.1.

Swift 5.1 now makes it easier to create and share binary frameworks with others. It also includes features that make it easier to design better APIs and reduce the amount of common boilerplate code.

Key Features

- Module stability defines a new text-based module interface file that describes the API of a binary framework.
- Property wrappers introduce a general-purpose syntax for defining custom access patterns for property values.
- Opaque result types help hide implementation details in APIs.
- Self can now be used for classes and value types.
- Support for handling and updating diffs on collections of appropriate types.
- Improvements to SIMD and String types.

3.4 Architectural Presentation Pattern

In the current release of the application (**Android** and **iOS**) using MVC architectural pattern so the software components grouped and separated with a useful way that allow modification and extended as easy.
There are three types of most commonly used architectural UI design patterns such as MVC, MVP, and MVVM. MVP is an abbreviation of Model-View-Presenter. MVC is an abbreviation of Model-View-Controller. Whereas MVVM stands for Model-View-View Model.
All of these design patterns:

- It plays a significant role in developing an application as best practices formalize them that are loosely combined
- Easier to test & maintain and facilitate reusable object-oriented development.
- These architecture patterns are designed to do moderate the complex codes and make the UI code cleaner and manageable.

3.5 Application Programming Interface

3.5.1 Android Version

Application uses square OkHttp3 library to establish connection network. HTTP is the way modern applications network. It's how we exchange data & media. Doing HTTP efficiently makes your stuff load faster and saves bandwidth.

OkHttp is an HTTP client that's efficient by default:

- HTTP/2 support allows all requests to the same host to share a socket.
- Connection pooling reduces request latency (if HTTP/2 isn't available).
- Transparent GZIP shrinks download sizes.
- Response caching avoids the network completely for repeat requests.

OkHttp perseveres when the network is troublesome: it will silently recover from common connection problems. OkHttp supports modern TLS features (TLS 1.3, ALPN, certificate pinning). It can be configured to fall back for broad connectivity.

Using OkHttp is easy. Its request/response API is designed with fluent builders and immutability. It supports both synchronous blocking calls and asyncounse calls with callbacks.

- We propose use the third-party library such as Retrofit with Gson because it can make your app simple and clear to write REST code. Retrofit is used to manipulate the http web service API communication, Gson is used to parse web service API return data in JSON format.
- In Retrofit can configure which converter is used for the data serialization
- Retrofit uses the OkHttp library for HTTP requests.

3.5.2 iOS Version

Alamofire is used in the current version, Alamofire is an elegant and composable way to interface to HTTP network requests. It builds on top of Apple's URL Loading System provided by the Foundation framework. At the core of the system *URLSession* and the *URLSessionTask* subclasses. Alamofire wraps these APIs, and many others, in an easier to use interface and provides a variety of functionality necessary for modern application development using HTTP networking.

3.6 Third-party Libraries

In the current **Android** version, we use some third-party libraries so we suppose to use it because it provides many benefits.

Top third-party libraries can be used in Android development are

1. **Font Awesome -** Font Awesome gives you scalable vector icons that can instantly be customized - size, color, drop shadow, and anything that can be done with the power of CSS.
2. **Material Design icons -** material-design-icons are the official open-source icons featured in the Google Material Design specification.
3. **Universal Image Loader for Android -** aims to provide a reusable instrument for asynchronous image loading, caching and displaying.
4. Volley, developed by Google engineers, used for simple http requests.
5. Retrofit, by Square. Competes with volley, but volley can be used even to download images, which retrofit is not developed for.
6. Gson, by Google. To convert Java objects into Json and back.

3.7 Summary of Assessment of current and proposed android version

Item	**Current Release**	**Proposed Release**	**Benefits of proposed Release**
User Interface (UI)	include libraries such as support library package	We would use material design	• **Branding** • **User Engagement** • **BETTER UI/UX** • **Cost Effective**
Source Code	Used Java Language	We will use Kotlin	• efficient and presents a familiar development tooling that is meant to boost developers' productivity • a good compiler • a seamless integration with the existing infrastructure (Kotlin's compatible with all Java frameworks and libraries, and it's designed to integrate easily with Marven and Gradle build systems, as well) • Provides an enhanced run-time performance.

Architectural Presentation Pattern	Use MVC design pattern	We would use the same design patterns or can change it to another one.	• It plays a significant role in developing an application as best practices formalize them that are loosely combined. • Easier to test & maintain and facilitate reusable object-oriented development. • These architecture patterns are designed to do moderate the complex codes and make the UI code cleaner and manageable.
Application Program Interface (API)	OkHttpClient and JSON Object	We would to use Retrofit with Gosn	• It can make your app simple and clear to write REST code. • Retrofit is used to manipulate the http web service api communication • Gson is used to parse web service API return data in JSON format. • In Retrofit can configure which converter is used for the data serialization
Third-party Library	Use some libraries such as OneSingle for notifications, Realm for local database.	We propose use more third-party libraries such as Retrofit with Gson.	• They ease the developer's job through code re-use

3.8Summary of Assessment of current iOS version and proposed version

Item	**Current Release**	**Proposed Release**
User Interface (UI)	Use default component in XCode without any animation or customization	We would use Library to customize design and add animation to make design more dynamic and beautiful
Source Code	Use Swift 4.2	We would use last version of swift
Architectural Presentation Pattern	Use MVC design pattern	We would use the same design patterns or can change it to another one.
Application Program Interface (API)	Use Alamofire	We propose use Alamofire with SwiftyJson
Third-party Library	Use some libraries such as OneSingle for notifications, Alamofire, MBProgressHUD, Realm for local database	We will use libraries as needed

3.9 Project Details and Information

3.9.1 Project General Overview	
Project Name	Electronic Mother and Child Health(eMCH)
Project Lead	Mohammad Raef
Project development Start and end dates	
Project Status & completion percentage	completed
Supported Platform (Android, iOS, Web control panel, API, 3rd party integration)	Android & iOS + web application
Development team size	3 (1 iOS, 2 Android)
Current support team size	1 iOS, 1 Android
Operation field (HQA, GFO, WBF, JFO, LBF)	All fields
Number of users	
3.9.2 Project Documentations	-
Software requirements specification	N/A
Software design document	N/A
User interfaces (wireframes)	N/A
Description of key components (i.e. algorithms) (if exits)	
Mobile database design document	Offline DB using Realm
Backend database design document	N/A
API design document	N/A

3.10 Development Details	
3.10.1 Backend Control panel	
Web technology stack	.Net
Database technology	SQL server
Source code	TFS
3.10.2 Backend API	
Service Architecture (API, SOAP…Others?)	API
Technology used	OkHttpClient in Android, Alamofire in iOS
3.10.3 Android	Version Code: 9 Version name: 1.2
Supported devices (mobile, tablet)	Mobile
Target OS & SDK version	Android SDK Platform (API level 28)
Programming language	Java
Key libraries & frameworks	hdodenhof/CircleImageView, EasyImage, glide,OneSignal
Source code	
Project Architecture	None
3.10.4 iOS	Version:1.3 Build: 6
Supported devices (iPhone, iPad)	Yes
iOS deployment target	iOS SDK (iOS 9)
Programming language	Swift 4.2
Key libraries and frameworks	None
Source code	

Project Architecture	None
3.11 Testing and Maintenance	
Test cases and test results	
Number of open incidents	
Number of pending requirements	
3.12 Integration with other systems	
What systems is the application integrated to	
Integration architecture design	
3.13 Deployment	
Android deployment location (play store, our server, by email, other)	Intranet
iOS deployment location (apple store)	N/A
Backend deployment location (server location)	
Deployment plan	
Deployment keys, token?	
3.14 Other Comments	

3.15 Business Function propose

- Edit Change password steps and flow.
- Modify data that can user edit when editing his profile

- **Enhanced Authentication Mechanisms in eMCH Application**

  Most mobile apps implement some kind of user authentication. Even though part of the authentication and state management logic is performed by the backend service, authentication is such an integral part of most mobile app architectures that understanding its common implementations is important.

  The basic concepts are identical on iOS and android.

<u>Steps of User Login:</u>

1. User entering username (Individual registration No. or Medical File No. for UNRWA users or email for NON- UNRWA users) and password.
2. Ensure that all data have been entered correctly
3. Send request to backend service
4. After request success, the return response include data such as user token, expire in, token type, these data stored locally (in shared preference in Android and in user default in iOS) in application.
5. Then the user entering the home screen of application
6. After logging to the application, the user password saved locally because it's using when the user wants to change the password.

4 Non-communicable diseases (eNCD)

4.1 Introduction

Non-communicable diseases (NCDs) is a mobile application works on two platforms (Android and iOS) that will be developed with UNRWA. The application designed to provide the target users with all the data they need about the Non-communicable diseases, which has been designed and illustrated based on the guidelines that have been referenced by UNRWA. The Goal of this application is to allow the target users from accessing their health-related data in an effective way. This document assesses the current release of application on two platforms (Android and iOS) and proposes enhancements if needed

4.1 User Interface

The user interface (UI) for an Android app is built as a hierarchy of layouts and widgets. The layouts are ViewGroup objects, containers that control how their child views are positioned on the screen. Widgets are View objects, UI components such as buttons and text boxes.

4.1.1 <u>Android Version</u>

In our Android Application that include libraries such as support library package that contains the largest set of APIs compared to the other libraries, including support for application components, user interface features, accessibility, data handling, network connectivity, and programming utilities. We would add some enhancement to the design, transition between screens, and components of current release to make the app has a consistent layout, look, and feel using material design or other libraries.

4.1.2 <u>iOS Version</u>

The Interface Builder editor within XCode makes it simple to design a full user interface without writing any code. Simply drag and drop windows, buttons, text fields, and other objects onto the design canvas to create a functioning user interface. Because Cocoa and Cocoa Touch are built using the Model-View-Controller pattern, it is easy to independently design your interfaces, separate from their implementations. User interfaces are actually archived Cocoa or Cocoa Touch objects (saved as .nib files), and macOS and iOS will dynamically create the connection between UI and code when the app is run

A complete iOS app is composed of multiple views through which the user navigates. The relationships between these views are defined by storyboards, which show a complete view of your app’s flow. Interface Builder’s storyboard designer makes it easy to create and design new views, and chain them together to create a complete user interface that’s ready for your custom code.
eMCH iOS application has a complete and separate storyboard for each type of user in the application or special section and clean design but if we add some customization on some component such as a cell used in table of user profile or bottom bar in the main screen used in the design.

4.2 Source Code

Java is the language used in the current android version, but today most programmers are directed towards the Kotlin language because of its many advantages.

Some of the main goals of Kotlin's development team are rapidly becoming some of the biggest advantages of using Kotlin over Java. Kotlin is:

- Efficient and presents a familiar development tooling that is meant to boost developers' productivity.
- A good compiler.
- A seamless integration with the existing infrastructure (Kotlin's compatible with all Java frameworks and libraries, and it's designed to integrate easily with Marven and Gradle build systems, as well).
- It provides an enhanced run-time performance.

In **iOS Platform** used swift language version 4.2 and the last version from apple is swift 5.1

Swift 5.1 now makes it easier to create and share binary frameworks with others. It also includes features that make it easier to design better APIs and reduce the amount of common boilerplate code.

Key Features

- Module stability defines a new text-based module interface file that describes the API of a binary framework.
- Property wrappers introduce a general-purpose syntax for defining custom access patterns for property.
- Opaque result types help hide implementation details in APIs.
- 'Self' can now be used for classes and value types.
- Support for handling and updating diffs on collections of appropriate types.
- Improvements to SIMD and String types.

We propose update programing language to last version of swift and upgrade pods file that application included.

4.3 Architectural Presentation Pattern

In the current release of the application (**Android** and **iOS**) using MVC architectural pattern so the software components grouped and separated with a useful way that allow modification and extended as easy.
There are three types of most commonly used architectural UI design patterns such as MVC, MVP, and MVVM. MVP is an abbreviation of Model-View-Presenter. MVC is an abbreviation of Model-View-Controller. Whereas MVVM stands for Model-View-View Model.
All of these design patterns

- They play a significant role in developing an application as best practices formalize them that are loosely combined.
- Easier to test & maintain and facilitate reusable object-oriented development.
- These architecture patterns are designed to do moderate the complex codes and make the UI code cleaner and manageable.

4.4 Application Program Interface (API)

4.4.1 Android Version

Application uses square OkHttp3 library to establish connection network. HTTP is the way modern applications network. It's how we exchange data & media. Doing HTTP efficiently makes your stuff load faster and saves bandwidth.

OkHttp is an HTTP client that's efficient by default:

- HTTP/2 support allows all requests to the same host to share a socket.
- Connection pooling reduces request latency (if HTTP/2 isn't available).
- Transparent GZIP shrinks download sizes.
- Response caching avoids the network completely for repeat requests.

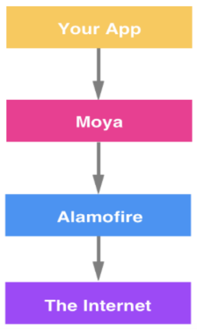
OkHttp perseveres when the network is troublesome: it will silently recover from common connection problems. OkHttp supports modern TLS features (TLS 1.3, ALPN, certificate pinning). It can be configured to fall back for broad connectivity. Using OkHttp is easy. Its request/response API is designed with fluent builders and immutability. It supports both synchronous blocking calls and async calls with callbacks.

- We propose use the third-party library such as Retrofit with Gson because it can make your app simple and clear to write REST code. Retrofit is used to manipulate the http web service api communication, Gson is used to parse web service API return data in JSON format.
- In Retrofit can configure which converter is used for the data serialization
- Retrofit uses the OkHttp library for HTTP requests.

4.4.2 iOS Version

Most developers probably use Alamofire to abstract away access to URLSession and lots of smart developers write ad hoc network abstraction layers. They are probably called "APIManager" or "NetworkModel". Moya is used in the current version; the basic idea of Moya is that we want some network abstraction layer that sufficiently encapsulates actually calling Alamofire directly. It should be simple enough that common things are easy but comprehensive enough that complicated things are also easy.

Some awesome features of Moya:



- Compile-time checking for correct API endpoint accesses.
- Let's you define a clear usage of different endpoints with associated enum values.
- Treats test stubs as first-class citizens so unit testing is super-easy.

4.5 Third-party Library

In the current **Android** version, we use some third-party libraries so we propose use it because it provides many benefits.

<u>Top third-party libraries can be used in Android development are:</u>

- **Font Awesome -** Font Awesome gives you scalable vector icons that can instantly be customized - size, color, drop shadow, and anything that can be done with the power of CSS.
- **Material Design icons -** material-design-icons are the official open-source icons featured in the Google Material Design specification.
- **Universal Image Loader for Android -** aims to provide a reusable instrument for asynchronous image loading, caching and displaying.
- Volley, developed by Google engineers, used for simple http requests.
- Retrofit, by Square. Competes with volley, but volley can be used even to download images, which retrofit is not developed for.
- Gson, by Google. To convert Java objects into Json and back.

4.6 Summary of Assessment of current and proposed android version

Item	Current Release	Proposed Release	Benefits of proposed Release
User Interface (UI)	include libraries such as support library package	We would use libraries to enhance design	• **Branding** • **User Engagement** • **BETTER UI/UX** • **Cost Effective**

Source Code	Used Java Language	We propose use Kotlin	• efficient and presents a familiar development tooling that is meant to boost developers' productivity • a good compiler • a seamless integration with the existing infrastructure (Kotlin's compatible with all Java frameworks and libraries, and it's designed to integrate easily with Marven and Gradle build systems, as well) • It provides an enhanced run-time performance.
Architectural Presentation Pattern	Use MVC design pattern	We would use the same design patterns or can change it to another one.	• IT plays a significant role in developing an application as best practices formalize them that are loosely combined • Easier to test & maintain and facilitate reusable object-oriented development. • These architecture patterns are designed to do moderate the complex codes and make the UI code cleaner and manageable.
Application Program Interface (API)	OkHttpClient and Gson	We would to use Retrofit with Gson	• It can make your app simple and clear to write REST code. • Retrofit is used to manipulate the http web service API communication. • Gson is used to parse web service API return data in JSON format. • In Retrofit can configure which converter is used for the data serialization.
Third-party Library	Use some libraries such as OneSingle for notifications, Realm for local database.	We will use libraries as needed	• They ease the developer's job through code re-use.

4.7 Summary of Assessment of current and proposed iOS version

Item	Current Release	Proposed Release
User Interface (UI)	Use default component in XCode without any animation or customization	We would use Library for custom design and animation to make design more dynamic and beautiful if we needed
Source Code	Use Swift 4.2	We would use last version of swift
Architectural Presentation Pattern	Use MVC design pattern	We would use the same design patterns or can change it to another one.
Application Program Interface (API)	Use Moya	We don't need to make any changes
Third-party Library	Use some libraries such as OneSingle for notifications, Moya, SVProgressHUD, Realm for local database	We will use libraries as needed

4.8 Project Details and Information

### 4.8.1 Project General Overview	
Project Name	Non-communicable diseases (eNCD)
Project Lead	Mohammad Raef
Project development Start and end dates	
Project Status & completion percentage	completed
Supported Platform (Android, iOS, Web control panel, API, 3rd party integration)	Android & iOS + web application
Development team size	3 (1 iOS, 2 Android)
Current support team size	1 iOS, 1 Android
Operation field (HQA, GFO, WBF, JFO, LBF)	All fields
Number of users	

4.8.2 Project Documentations	1. The Document is missing important details about Text to speech technology that are used in application 2. Need more description for the flow of two types of notification sent by CMS. 3. Need more details about notification rules.
Software requirements specification	N/A
Software design document	N/A
User interfaces (wireframes)	N/A
Description of key components (i.e. algorithms) (if exits)	
Mobile database design document	Offline DB using Realm
Backend database design document	N/A
API design document	N/A
4.9 Development Details	
4.9.1 Backend Control panel	
Web technology stack	.Net
Database technology	Sql server
Source code	TFS
4.9.2 Backend API	
Service Architecture (API, SOAP…Others?)	API
Technology used	OkHttpClient in Android, Moya in iOS

4.9.3 Android	Version Code: 3 Version name: 1.1
Supported devices (mobile, tablet)	Mobile
Target OS & SDK version	Android SDK Platform (API level 28)
Programming language	Java
Key libraries & frameworks	hdodenhof/CircleImageView, Firebase-core, jsoup
Source code	
Project Architecture	None
4.9.4 iOS	
Supported devices (iPhone, iPad)	Yes
iOS deployment target	iOS SDK (9)
Programming language	Swift 4.2
Key libraries and frameworks	None
Source code	
Project Architecture	None

4.10 Testing and Maintenance

Test cases and test results	
Number of open incidents	
Number of pending requirements	

4.11 Integration with other systems

What systems is the application integrated to	
Integration architecture design	

4.12 Deployment

Android deployment location (play store, our server, by email, other)	

iOS deployment location (apple store)	N/A
Backend deployment location (server location)	
Deployment plan	
Deployment keys, token?	

4.13 Other Comments
