|  |  |  |  |
| --- | --- | --- | --- |
| Student Information | | | |
| **Student ID:** |  | **Student name:** |  |
| **Unit code** | ICTPRG601  ICTPRG532 | **Unit title** | Apply advanced object-oriented language skills  Develop advanced mobile multi-touch applications |

**Part 1. Task Instructions**

In this task, you will be presented with a graphic design of an Android mobile app, and detailed software design specification document (Part 2 in this document). Your job is to develop and publish the Android mobile app that meet all the design requirements. Also, you are required to answer all the questions listed in Part 3 in this document.

The whole software development process can be break down into the following phases

1. Plan & Preparation
2. Client-side Android app user interface design
3. Client-side Local database design and connect to the UI
4. Server-side database design
5. Server-side Web API design
6. Connect front-end Android app to back-end web APIs
7. Software test and performance analysis
8. Documentation and quality control
9. Publish mobile app and handover project

P1

P2

P4

P3

P5

P6

P7

P9

BEGIN

END

P8

You are required to follow the same order as the above 9 phases to complete developing the software. Part 3 questions are also designed following the same logic. So, for each phase of the development, you should walkthrough and answer the corresponding questions as you work on programming. After you finished one phase of programming and questions, you are required to present your work to the assessor; you assessor will also confirm if you can move on to the next phase by signing off the section.

Your assessor also plays the role of the client. Speak with your assessor for requirement clarification if needed.

For final submission, you are required to submit the following:

|  |  |
| --- | --- |
| 1. A signed assessment document |  |
| 1. This instruction document with all questions answered and signed off by assessor for each phase. |  |
| 1. Android app project (source code) |  |
| 1. Server-side project (source code) |  |
| 1. Client-side database snapshots |  |
| 1. Server-side database snapshots |  |
| 1. Web API design document |  |
| 1. User document (html file) |  |
| 1. All test plan documents |  |
| 1. All test report documents |  |
| 1. Performance analysis report |  |
| 1. Application files for deployment (e.g. Android APK) |  |
| 1. Inter device communication program project (if programmed separately) |  |

**Part 2. Software Design Specification**

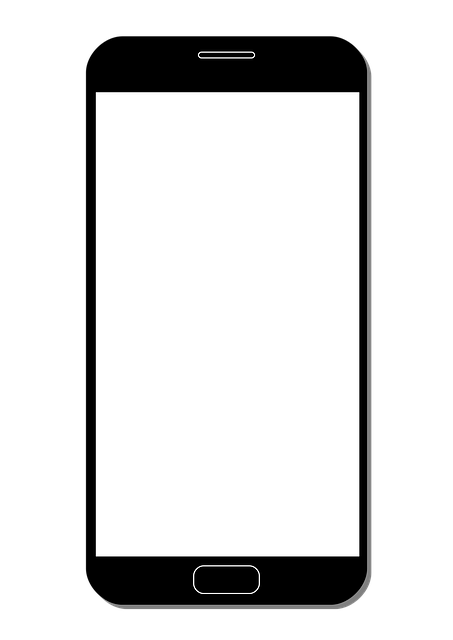
You are required to design an Android mobile app that meet the requirements given by graphic design and this specification. Graphic design resources are provided separately; requirements from the software perspective are explained in this document. Consult the client for further clarification as required.

1. **Overview of the software**

The application provides an interface to perform CRUD functionalities for data stored in databases, on both client-side and server-side. The client-side application is an Android mobile App with local DB (e.g. SQLite), the server-side service is HTTP-based RESTful API with remote DB (e.g. MySql). The overall design is illustrated below.

**Server-side**

**Client-side**



Android App

SQLite DB

Server-side API

MySql DB

DB Sync API

CRUD

CRUD

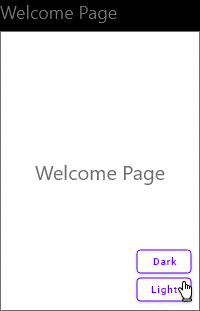
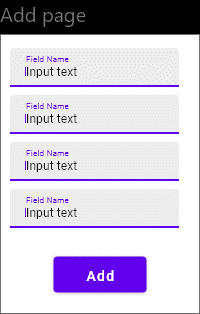
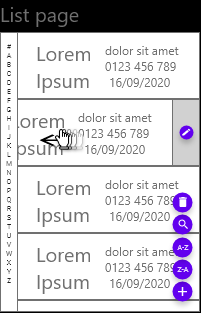
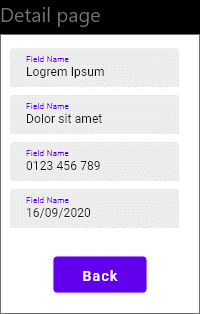
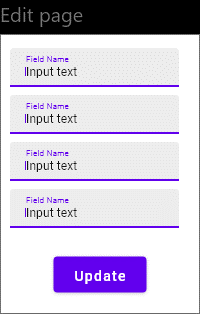
Each component and its functionalities are explained in the following sections.

The database, which is an RDS DB, shall have at least one table with **at least 4 columns besides the ID column, with at least one column of DataTime type.**

1. **Sub-systems requirements**

**Android application**

1. Android App page navigation

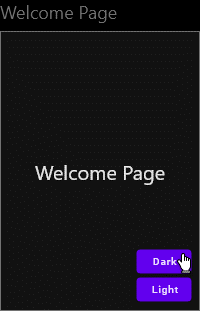
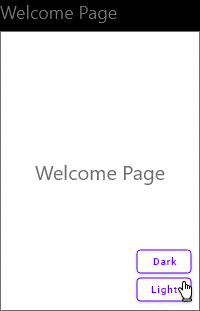


The mobile application consists of 4 pages. The navigation relationships of the four pages are as follow.

* “Welcome Page” displays welcome information. A right swipe gesture on this page navigate user to the “List Page”
* “List Page” displays a list of items and provide following navigations to other pages
  + Floating Action Button  navigates user to “Add Page” to add an entry to the table in database.
  + Clicking on an individual item navigates user to the “Detail Page”.
  + Swiping right an individual item makes appear the edit button  , clicking which navigates user to “Edit Page”
  + Drag-drop an individual item to the bin icon delete the corresponding row in database table.
* In “Add Page”, clicking “Add” button navigates user back to “List Page”.
* In “Edit Page”, clicking “Update” button navigates user back to “List Page”.
* In “Detail Page”, clicking “Back” button navigates user back to “List Page”.

1. Support two colour themes

In the “Welcome Page”, user can select colour theme from two pre-defined colour themes, Light and Dark. Once selected, the colour preference shall be stored in device file system and to be used when next time app is started. Light theme is used if no preference stored.



1. CRUD functionalities

The mobile application provides CRUD functionalities to both client-side Database and server-side database.

* **Create**: “Add Page” is used for creation. By clicking “Add” button, a new entry will be added to server-side DB and / or *\** client-side DB.
* **Read**: “List Page” requires all entries to be read. “Detail Page” requires one entry to be read.
* **Update**: “Edit Page” is used for updating. By clicking “Update” button, the corresponding entry will be updated on server-side DB and / or *\** client-side DB.
* **Delete**: In “List Page”, drag-drop action allows deleting one entry on server-side DB and / or *\** client-side DB.

*\* Depending on network availability, server-side DB may not be reachable, sync up may be required when network is back*

1. Hash, sort, and search

In the list page the following functionalities shall be implemented to improve usability

* **Hash**: A hash table shall be implemented to grouping the data read from DB table. The hash table has 27 indexes from 0~26 representing “#” (meaning non-alphabetic character) and A(a)~Z(z). The hash function will take into account the first character of a given “String” and determine Hash index. For example, string “Adam” maps to hash index 1 (meaning A or a), “1Joe” maps to hash index 0 (meaning non-alphabetic character).

Use double-linked list to resolve hash collision, and items shall be sorted alphabetically for each list.

For UI, in “List Page” a left-side navigation bar with 27 buttons shall be created for quick navigation. That is to say, for example, when user click on button “K”, the list will auto scroll to the items that start with “K” or “k”.

* **Sort**: When user click on button  or , all items in the main list shall be order correctly in ascending or descending alphabetical order.
* **Search**: After search is performed, a list of the items that meet search criterial shall be presented in the “List Page”.

1. Databases synchronization

All data in both server-side DB and client-side DB need to be synchronized any time when the application is running. However, when network is not available, CRUD functions can be performed in client-side DB only, and data will not be in synchronized status before network is back. The mobile application shall be able to synchronize the two databases when network is restored, and **user shall be informed when synchronization status changed**.

1. Advanced features

You are required study and apply Android API to implement at least **one** of the following features for the app.

* Click on a phone to make a call
* Send SMS
* Using camera

**Server-side API**

Server-side HTTP-based API shall provide CRUD functions for accessing server-side database. The API shall be designed in RESTful manner.

1. **Technologies and Algorithms**

The following technologies and algorithms should be adopted for each part of the system.

|  |  |  |
| --- | --- | --- |
| **Android** | **Server-side** | **Others** |
| Java or Kotlin | PHP or C# with or without framework like Laravel and ASP.net | MVC design pattern |
| Activity (Fragment if necessary) | RESTful API structure | Git |
| ModelView | MySQL or SQL Server | Search, Sort, Hash algorithm |
| Room + LiveData |
| SQLite |
| Animation |
| Touch, gesture, drag-drop |

1. **Software testing and performance analysis**

A test plan shall be created to test the functionalities of the software.

Software shall be tested following the test plan, and results to be reported.

Performance analysis shall be performed, and result shall be documented and reported.

1. **User document**

A user level of document shall be created and published to a website with public access.

1. **Publish mobile app**

Android Package (APK) file shall be generated and published to a website with public access.

1. **Documentation requirements**
2. Code following recommended code conversions.
   * Java: <https://google.github.io/styleguide/javaguide.html>
   * Kotlin: <https://kotlinlang.org/docs/reference/coding-conventions.html>
3. Use source code version control system to manage project. You are free to choose any one of the online system like [Github](https://github.com/), [Bitbucket](https://bitbucket.org/), or [Gitlab](https://gitlab.com/)
4. Source code must be fully documented, according to code conversion guides provided in “A” above.

**Part 3. Questions**

You are required to finish all the questions in this part for final submission. The questions in this part are grouped into 8 phases which map to the same 8 phases of the implementation.

For each phase, it is recommended to read the questions and Part 2 specification first before starting programming.

**Phase 1 – Plan & Preparation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In this phase, you are required to review the requirements provided in Part 2 and then finish the following tasks:   * Identify the development environment – Android Studio * Setup and verify development environment * Complete an implementation plan and reach agreement with client. | | | | | | |
| **Q1.1** Identify development environment and complete the following table. | | | | | | |
| |  |  | | --- | --- | | (A) OOP language chosen for Android app | JAVA | | (B) IDE for Android app development | Android Studio | | (C) Test environment for Android app | Mobile Device or Virtual Mobile Device | | (D) Another OOP language for server-side system (must be different from the one chosen in (A)) | C# | | (E) IDE for server-side system development | Visual Studio | | (F) Test environment for server-side system | Microsoft SQL Management Studio (SSMS), Swagger | | (G) Source code version control system | GitHub | | | | | | | |
| **Q1.2** Setup the chosen development environment and provide screenshot for each | | | | | | |
| |  |  | | --- | --- | | IDE for Android app development | */\* create a test project and screenshot the IDE to show the project structure \*/* | | Test environment for Android app | */\* run the test project and screenshot the result \*/*  Test environment is testing the application on the Virtual Mobile Device or on an Android Phone | | Run debugger in the Android IDE with test code.  *\*You are required to create test code with loop and “if” conditions, setup breakpoints, navigate code in debug mode, watch variable values* | */\* either screenshot the debug process \*/*  https://media.discordapp.net/attachments/690034978431828090/695040187440562316/unknown.png?width=1384&height=672 | | IDE for server-side system development | */\* create a test project and screenshot the IDE to show the project structure \*/*  Visual Studio – C#, ASP .NET Core Web API solution | | Test environment for server-side system | */\* run the test project and screenshot the result \*/*  *The API can be tested using the Swagger interface* | | Source code version control system | */\* create a repository on selected git server and screenshot the empty repository \*/*  https://github.com/hatysfriend/Android-Project | | | | | | | |
| **Q1.3** Review the requirement and write down how you plan to implement the required system; then present to your client for approval. | | | | | | |
| */\* use chart, diagram, or table to facilitate the planning \*/*   |  |  | | --- | --- | | Features | Implementation | | Multi-screen application | Use ‘intents’ for multiple pages | | Phonebook List | Use Recycler View to generate the List | | Local Database | SQLite & Room database | | Server-side Database | Implemented using C# | | API | Implemented using Retrofit & GSON dependencies | | Additional Feature (Camera) | Implemented using the ‘MediaStore’ library |   Client signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | |
| **Checklist (To be completed by the learner’s facilitator)** | | | | | Yes | No |
| 1. Learner has correctly identified the development environment. | | | | |  |  |
| 1. Learner has successfully setup the development environment. | | | | |  |  |
| 1. Learner has written down an implementation plan. | | | | |  |  |
| 1. Learner has presented the plan to client and agreement has been reached. | | | | |  |  |
| **Assessor Name** |  | **Assessor Signature** |  | **Date** |  | |

**Phase 2 – Client-side Android app user interface design**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In this phase, you are required to analysis the UI requirements and complete building the “view” part of the Android app. The following Android components should be used to build up the app in this phase:   * [Activity](https://developer.android.com/guide/components/activities/intro-activities?hl=en) with layout resource * [Fragment](https://developer.android.com/guide/components/fragments?hl=en) with layout resource (optional) * [ViewModel](https://developer.android.com/topic/libraries/architecture/viewmodel?hl=en)   You are also free to choose any other Android components (e.g. [RecyclerView](https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=en) to create a list of something) and technologies (e.g. [support different screen size](https://developer.android.com/training/multiscreen/screensizes)) for the app where needed to meet the UI requirements.  In addition, you are also required to practice the following:   * Develop according to code conversion document * Follow the MVC design pattern to layer source code properly. * Document source code as you program * Manage source code with version control software   Database is not required and should NOT be involved in this phase, but interface to database should be considered. In order to feed data to the UI, static variables can be used to hold the data and play a role of database. After database is implemented in next stage, you can simply switch the data source from static variables to actual database interfaces. | | | | | | |
| **Q2.1** Design test plan for UI functionalities. Paste the test plan file name below.  Note that CRUD function with static variables need to be implemented and verified in this phase. | | | | | | |
| */\* Test plan file name \*/*  <Test_Plan_UI_Functionalities.docx> | | | | | | |
| **Q2.2** Review “Android application / Android App page navigation” requirements. Briefly explain how you use Android “Activity” and / or “Fragment” components to implement multiple view pages and navigations between them? | | | | | | |
| */\* How to use Activity / Fragment to create multiple pages \*/*  An “Activity” or “Fragment” represents a page. When creating a new Activity, an **Activity class** file and a **Layout resource** file is created. These two files together are required to create an Android page. The Activity class contains the coding (i.e. variables and functionality) and the Layout file is the UI design of the page.  */\* How to implement navigation between pages \*/*  To navigate between pages, an Intent object needs to be created with the current page’s activity and the next page’s activity as parameters. (e.g. Intent i = new Intent( currentpage.this, nextpage.class); ) and then, basically, the “**startActivity( i )**” method is called to start or open the intent/next page. The ‘startActivity()’ method can be placed inside an on click button function or gestures function to create navigation between pages.  Another function which can be used to navigate between pages is the “**onBackPressed()**” function which can be called to return back to the previous page. | | | | | | |
| **Q2.3** Build all pages and correctly implement page navigations as required.   * Screenshot all pages created in layout editor * Screenshot the code sample that for page navigation | | | | | | |
| */\* Screenshot of all Pages \*/*    */\* Screenshot page navigation code \*/* | | | | | | |
| **Q2.4** What is Activity lifecycle? In terms of lifecycle, explain what happens to activity when auto-rotating and auto-resizing the device? | | | | | | |
| */\* Explain Activity lifecycle* *\*/*  Activity lifecycle is the set of functions or ‘stages’ an activity goes through, from when the activity is created, to when the activity is closed. The stages of an Activity’s lifecycle consists of:  onCreate() – is called when the activity is created  onStart() – is called when the activity becomes visible to the user  onResume() – is called when the activity is ready to interact with user  onPause() – is called when the activity is moved to the device’s background  onStop() – is called when the activity is no longer visible to the user  onDestroy() – is called before the activity is destroyed    */\* What happens to activity when auto-rotating and auto-resizing the device \*/*  When an Android device changes its orientation, all existing activities are destroyed and are ‘recreated’ to adapt to the required orientation view (Vertical or Landscape). This is done so that the application can reload all the require resource files and configurations related to the orientation. However, by destroying the activity when rotating, variable data is lost, so the android developer must code a way to save the data or the “application’s current state” so that the data is saved when changing orientation. | | | | | | |
| **Q2.5** Explain what lifecycle events are triggered when state changed? Screenshot your code where life cycle events are responded. | | | | | | |
| */\* Explain lifecycle events* *\*/*  Life cycle events are functions which are called during certain events during the life of an Activity e.g. onCreate(), onStart(), onPause(), onDestroy(). When a state is changed, such as changing the orientation of the screen, the following lifecycle functions are called:  onPause()  onStop()  onDestroy()  onCreate()  onStart()  onResume()  https://user-images.githubusercontent.com/7859175/81483183-79f4c980-923c-11ea-92e6-0b3763655861.png  */\* Screenshot your code* *\*/*  *When an Activity is created, the onCreate() method is automatically called.* | | | | | | |
| **Q2.6** Discuss the options in Android to save app data in file system, for example save user configuration. Discuss at least two options | | | | | | |
| */\* Option No.1* *\*/*  Shared Preferences – Used to save primitive data, stored using a ‘key-value pair’ structure. The data is stored in an xml file in a private directory. The **key** must be a String and the corresponding **value** can be either a Boolean, float, int, long or string. Shared preferences are generally used to store application settings.  */\* Option No.2* *\*/*  SQLite – SQLite is a relational database which is built in into Android, used to store, manipulate and retrieve data. The SQLite database is built and created for the app only, meaning the database cannot be used by other apps. | | | | | | |
| **Q2.7** How do you save colour theme configuration in file system when user makes a selection in “Welcome Page”? Screenshot code sample. | | | | | | |
| */\* Explain how do you save configuration in file system* *\*/*  The Colour theme configuration is saved using “Shared preferences”. Shared Preferences is a built-in library in Android which allows users to store basic data in “Key – Value pairs”. Users can store key-value pairs in the Shared Preferences xml file by using the following code structure seen in the screenshot below. In this example, the editor.putBoolean(**KEY\_ISNIGHTMODE**, **nightmode**) represents the Key – Value pair, where the “Key” is a string stored in the ‘KEY\_ISNIGHTMODE’ variable and the “Value” is a Boolean value passed via the ‘nightmode’ variable.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.8** In Android what technologies are used to auto adjust position of UI elements when screen size changed? Discuss at least two technologies and their use case | | | | | | |
| */\* Technology No.1* *\*/*  Constraint Layout is used to constrain the elements within the screen. Elements can be linked to the “parent” (i.e. edges of the screen) or to other elements to control the position of the elements. When screen size changes or the screen changes between Portrait or landscape, the elements will auto-adjust their positions relative to their defined constraints.  e.g. **app:layout\_constraintEnd\_toEndOf="parent"**  */\* Technology No.2* *\*/*  A ‘Landscape Variation’ of a page can be generated (e.g. have two separate layouts for a page, a Portrait AND a Landscape version) which can be used to control positioning of elements. When changing the screen between Portrait and Landscape, the application will automatically change between the two layouts. | | | | | | |
| **Q2.9** In your app, how do you auto adjust position of UI elements in response to the change of screen size? Screen code sample as well. | | | | | | |
| */\* Explain how you auto adjust UI position* *\*/*  Elements auto adjust their position based on the constraints set for them and are based, relatively to the screen edge or position of other elements. In the example below, the “Light” button is vertically and horizontally constrained with:  **app:layout\_constraintBottom\_toBottomOf="parent" app:layout\_constraintEnd\_toEndOf="parent"**  These constraints will keep the “Light” button in a position relative to the ‘Bottom’ and ‘End’ of the parent (i.e. Bottom and Right-edge of screen). These constraints allow the element to adjust its position whenever screen size or orientation changes.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.10** Data saved in Activity class may be lost when Activity is rebuilt, i.e. during auto-rotation. In Android, what technology is used to prevent losing data? Explain how does it work? | | | | | | |
| */\* Name the technology* *\*/*  Use “View Model” technology from the Jetpack Library  */\*Explain how it works \*/*  The ViewModel is a class which will store and manage UI data for the duration of an Activity’s lifecycle. ViewModel technology allows data to survive between lifecycle events such as OnCreate, OnDestroy e.g. Screen rotation. How it does this is, the ViewModel retains data in memory until it is scoped to go away permanently. | | | | | | |
| **Q2.11** Explain and screenshot the code to show that you have used proper technology (separating the model and view) to prevent losing data when Activity is rebuilt (e.g. rotate device). | | | | | | |
| */\* Explain your implementation* *\*/*  A ViewModel class was created which has a property to store a ‘Hash Table’.  When working with the Contact List on the List Page, the Contact List data is stored in the ViewModel object “hash.myHash”. This ViewModel object represents the HashTable of the app and is used when displaying the contact list on the screen.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.12** Explain how to build a list view in Android app? | | | | | | |
| To Create a List View:   1. Create a layout with a ‘Recycler View’ element 2. Create an Item layout which will be repeated in the recycler view. 3. Create the ItemViewHolder class which extends ‘RecyclerView.ViewHolder’, this class will represent each ‘Item’ in the RecyclerView List. 4. Create RecyclerViewAdapter class which extends ‘RecyclerView.Adapter<*ItemViewHolder class*>’ with the *itemViewHolder* class created in step 3. Implement required methods, and link the ‘Item Layout’ (step 2) to the ViewHolder. | | | | | | |
| **Q2.13** Android components, for example RecyclerView, that used for creating a list view often requires a Data Adapter. Use the list view component you selected as example and explain how Dataset (as model), Adapter and List View component work together. You may explain by using code samples from you project. | | | | | | |
| The dataset (the contact List) is converted in a viewModel object (hash.myHash.toList()).  The adapter is an instance of the MainListRecyclerViewAdapter which contains a Contact List property where the viewModel is stored. The adapter represents the contact List and is set or stored in the recyclerView. | | | | | | |
| **Q2.14** It is required that left swipe gesture navigate user from “Welcome Page” to “List Page”. Explain and screenshot your implementation in source code. | | | | | | |
| */\* Explain how you implement swipe gesture* *\*/*  Import the “GestureDetectorCompat” library which provides the gesture listener functionality. Gesture Listener enables the application to implement functions such as “onFling()”. So when an onFling motion event occurs, some code can be executed, e.g. navigate to another page.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.15** In “List Page”, left drag an individual item will move the top layer to left and bottom layer with edit button will appear. When the top layer changes position, animation is used to smooth the transition. Explain and screenshot your implementation in source code. | | | | | | |
| */\* Explain how you move UI position in response to figure touch and movement* *\*/*  An onTouch Listener is attached to a viewHolder (a Contact in the List). During an OnTouch event, co-ordinates of the viewHolder are used to determine if the position of the viewHolder reaches are certain point. Once the viewHolder is swiped left, pass the certain position, the “edit button” is revealed underneath the viewHolder.  */\* Screenshot your code* *\*/*    */\* Explain how you implement changing UI position with animation* *\*/*  When the user swipes a contact to the left, the “edit Button” is drawn.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.16** In “List Page”, drag-drop an item to “bin” icon will delete the item. Explain and screenshot your implementation in source code. | | | | | | |
| */\* Explain how you implement drag-drop* *\*/*  In the MainListViewHolder class (which represents an object of each item in the recycler view), the ‘OnLongClickListener’ is implemented on this item, so when a longClick event occurs on a contact item, code is executed to create a drag-able shadow and also to get the position/index of the contact.  The floatingActionDeleteButton has the “setOnDragListener()” applied to it, which means the delete button can ‘listen’ to drag events (such as ACTION\_DROP) and execute code when a drop event occurs.  So, when a OnLongClick event occurs, a drag-able shadow is created and the position is retrieved.  With the position, the contact to be deleted can be retrieved.  Then when the shadow is dropped on the floating action delete button, the Drag.event.ACTION\_DROP occurs which deletes the retrieved contact.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.17** In “List Page”, click or tap an item will navigate user to detail page. Explain and screenshot your implementation in source code. | | | | | | |
| */\* Explain how do you response a tap on the item view \*/*  First make each item of the recyclerView “Clickable” by implementing the OnClickListener on the MainListItemViewHolder class and set OnClickListener on the itemView. The position of the contact is also retrievable and passed via the extension RecyclerView.ViewHolder, getAdaptorPosition() method.  The List page implements another interface called ContactRecordListner which contains interface method, onClickOfAContact() which is called when a Contact/itemView is clicked.  When a Contact is clicked, the position is retrieved and used to retrieve the correct Contact object and the object is passed to the Detail Page using intent.PutExtra(“key”, value) which passes a key-value pair with the intent. The passed object is then displayed on the Detail Page.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.18** There are multiple instances that functions are triggered by clicking button. Use an example in the app to explain how do you implement button click. Provide screenshot for that as well. | | | | | | |
| */\* Explain how do you implement button click \*/*  To implement a button click, need to register an element (e.g. Button or itemView) with a setOnClickListener().  In the example below, the ‘btnAdd’ (a floating action button element) has a setOnClickListener() method attached and within this method, contains a onClick() method which is executed when the ‘add floating action button’ is clicked.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.19** A left navigation system is required in “List Page”. And you required to use Hash technology to implement the feature. Explain how do you implement this using Hash technology and provide code screenshot | | | | | | |
| */\* Explain how the navigation with Hash technology \*/*  The navigation buttons work by calculating an ‘offset’ based on the navigation button clicked. Depending on the button clicked, a key value is passed (a value between 0 - 26 e.g. #=0, a=1, b=2, c=3, etc.)  The offset is calculated using the hash table, where the ‘key’ passed is the ‘list character’ e.g. (key = 3/c, hashtable[3]) in the hash table and the offset value is equal to the sum of all the contacts within the lists, up to the key list.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.20** You are required to use double-linked list to resolve collision for the Hash table in the app. Explain how do you implement it and screenshot the code. | | | | | | |
| By having a List for each index in the hash table, elements will not collide or override when adding an element into the same index.  e.g. adding “Bella” into PhonebookDB[2] and then “Ben”, collision occurs and “Ben” will overwrite “Bella”,  however by having a List inside PhonebookDB[2], both elements, “Ben” and “Bella” can be stored, thus avoiding collision.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.21** In “List Page” sort functions are required. Explain the sorting algorithm you implemented and provide screenshot for that. | | | | | | |
| */\* Explain how the sorting algorithm your select \*/*  When the hashTable is built, it is built in an alphabetical manner where contacts are sorted in lists where each list represents a letter of the alphabet (and also # for non-alphabetical values). Each contact that is added to the hashTable is passed through a hash function “hash()” to sort where the contact will go, based on the Contact’s name.  So when the toList() function is called an alphabetically sorted listed is returned because the list is built in an alphabetical manner.  When the toList(true) is called with value”true”, the sort function use an inbuilt function from the *collections* library, called *reverse*(<list>). The reverse function is able to sort at ArrayList in ‘reverse’ order i.e. reverse the alphabetical list.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.22** In “List Page” search functions are required. Explain the search algorithm you implemented for Hash table and provide screenshot for that. | | | | | | |
| */\* Explain how the search algorithm your select \*/*  Search algorithm used was a simple foreach loop to cycle through the AllContacts, then compare contact name with search input using .contains(searchInput). If the search input contains the character typed in the search box, the matched contacts will be added to the ‘searchList’.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.23** Binary search tree can be an alternative approach for storing and searching. Create a program with selected OOP language to implement “binary search tree” with sample data and perform search. Explain how do you do it and screenshot the code. | | | | | | |
| */\* Explain how do you implement binary search tree for data storing and searching \*/*  Binary Tree is a type of data structure similar to double-linked lists, where data is stored in “nodes” which contain 3 properties, ‘the data’, ‘left-node’ and ‘right-node’. The Binary Tree structure size is dynamic, i.e. data can be continually add to the binary tree (e.g. same as Lists). In Binary Tree Structure, data is stored in way, where the newly added node compares its value to the ‘root node’ and subsequent nodes until it reaches to the end of a ‘branch’. A recursive function is used to traverse through the binary tree, where if the newly added node is less than the compared node, it will go to the ‘left’, if greater, it will go to the right and thus eventually find a spot at the end of a branch.  Searching through the binary tree is performed in the same fashion, starting from the root, compare the search value with a node’s value and if the search value is less than the compared node’s value, go to the left node, else go right. The search function will recursively execute until a value is found or if it reaches the end of a branch.  */\* Screenshot your code* *\*/*  *// Binary tree algorithm, Adding Nodes //Example of creating a Binary Tree structure*    *//Binary Search Example* | | | | | | |
| **Q2.24** Explain what is nested class in chosen programming language for Android app? Screenshot example in your code where a nested class is implemented. | | | | | | |
| */\* Explain nested class \*/*  Nested class is a class within a class, for example the MainListItemViewHolder class in the MainListRecyclerViewAdaptor Class. Nested classes are used to group classes that belong together. Nested classes are used when it makes logical sense to do so and for code readability. E.g. in example below, the ItemViewHolder class is the object(s) displayed within the RecyclerView so nesting these classes can be beneficial.  */\* Screenshot your code* *\*/* | | | | | | |
| **Q2.25** Test the UI according to test plan using emulator or android mobile device, report test result using template. | | | | | | |
| */\* Test report file name* *\*/* | | | | | | |
| **Q2.26** Address any issue(s) found in the test report. Use a software related issue as an example, explain how do you use IDE embedded debugger for troubleshooting and fix the bug. | | | | | | |
| */\* Problem Description* *\*/*  */\* Troubleshooting process* *\*/*  */\* Cause of the issue* *\*/*  */\* Solution Description* *\*/* | | | | | | |
| **Checklist (To be completed by the learner’s facilitator)** | | | | | Yes | No |
| 1. Test plan for phase 2 has been created | | | | |  |  |
| 1. Colour theme configuration can be saved and loaded from file system | | | | |  |  |
| 1. Page navigation works as expected | | | | |  |  |
| 1. CURD function works as expected with static variable as data source | | | | |  |  |
| 1. Tap, gesture, touch, drag-drop functions works as expected | | | | |  |  |
| 1. Animation function works as expected | | | | |  |  |
| 1. App correctly responds to device auto-rotating and auto-resizing | | | | |  |  |
| 1. Hash navigation function works as expected | | | | |  |  |
| 1. Sorting function works as expected | | | | |  |  |
| 1. Searching function works as expected | | | | |  |  |
| 1. Test is finished according to plan and results have been reported | | | | |  |  |
| 1. Issues identified during test have been addressed | | | | |  |  |
| 1. Software debugging process has been demonstrated and recorded | | | | |  |  |
| **Assessor Name** |  | **Assessor Signature** |  | **Date** |  | |

**Phase 3 – Client-side Local database design and connect to the UI**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Before starting Phase 3, you are required to have **Phase 2 CRUD** functionality programming completed, and verified. In this phase you are required to introduce Android local SQL database facility, the SQLite, into the application to replace the phase 2 data source, which is implemented with static variables as temporary solution.  For better practice, [Architecture Components](https://developer.android.com/topic/libraries/architecture/viewmodel?hl=en) should be leveraged for more robust SQLite database access. You should consider using the following Architecture Components from the [Jatpack](https://developer.android.com/jetpack) library:   * [LiveData](https://developer.android.com/topic/libraries/architecture/livedata) * [Room](https://developer.android.com/topic/libraries/architecture/room) ([DAO](https://developer.android.com/training/data-storage/room/accessing-data), [Room Entity](https://developer.android.com/training/data-storage/room/defining-data))   SQLite management tool is also required to manage local database. [*DB Browser for SQLite*](https://sqlitebrowser.org/) is recommended.  In addition, you are also required to practice the following:   * Develop according to code conversion document * Follow the MVC design pattern to layer source code properly. * Document source code as you program * Manage source code with version control software | | | | | | |
| **Q3.1** Design test plan for UI + SQLite functionalities. Paste the test plan file name below.  Note that a database management tool is required to verify the data in database. You should design at least one test case for using DB tool to verify data. | | | | | | |
| */\* Test plan file name \*/* | | | | | | |
| **Q3.2** Explain how to use “[Room Entity](https://developer.android.com/training/data-storage/room/defining-data)” to define an entity (or table) for SQLite? | | | | | | |
| To use the Room database Structure, the Room Persistence Library needs to be implemented into the project by adding the “Room” dependencies to the app’s build gradle. To use Room database structure, the app needs a ‘database’ class, ‘Entity’ class, and a DAO interface.  When using the Room database structure, to create a table or entity, it is defined in the project’s Model classes. As seen in the example below, the Contact “Entity” or table is created by using the ‘Room Entity’ annotations e.g. setting the Table name, column names with @Entity(), @ColumnInfo(). | | | | | | |
| **Q3.3** Screenshot in your code that you have convert Data Model built in phase 2 to Room Entity. | | | | | | |
|  | | | | | | |
| **Q3.4** Explain how to use “[Room DAO](https://developer.android.com/training/data-storage/room/accessing-data)” to design database access interface for operations like CRUD. | | | | | | |
| Room DAO stands for ‘Room Data Access Objects’ which are used to perform queries to the local database, e.g. CRUD functions. An interface class is created to store all the DAO functions. Inside this interface class, an “annotation” is used to define the interface as a ‘DAO interface’ using @dao, and within the interface are methods which are assigned with query annotations such as @insert, @update, @delete or @query to define the methods as DAO methods. | | | | | | |
| **Q3.5** Screenshot in your code that you have coded all DAO interfaces for the app | | | | | | |
|  | | | | | | |
| **Q3.6** Explain the synchronisation technique adopted to synchronise data between the Android app and SQLite data source? That means if data is modified in database, corresponding data in UI elements is updated automatically and instantly. | | | | | | |
| In this project, the contactList that is displayed in the List page is defined in the MainListRecyclerViewAdaptor class. To ensure the ‘contactList’ in the app and database’s contact list is synchronised, the loadList() function is called whenever a change to the database has occurred, e.g. data is changed in the database after a add, update or delete, so after a CRUD function is completed, the loadList() function is called which:   * grabs the latest data from the database with contactDao.getAllContacts(), * creates a new hashTable with the latest data * reload the recyclerView/adaptor with the latest contactList * refresh the List Page with notifyDataSetChanged(); | | | | | | |
| **Q3.7** Screenshot the synchronisation technique you implemented from your code. | | | | | | |
|  | | | | | | |
| **Q3.8** Explain what need to be done in order to connect the UI built in Phase 2 to the Room interfaces you have done in Phase 3. Screenshot code samples. | | | | | | |
| */\* Explain how to connect UI to Room interfaces \*/*   1. Implement “Room Persistence Library” by adding the “Room” dependencies to the app’s build gradle. 2. Define a ‘Model class’ as a Table in the database by defining table name and columns with Room Entity annotations. 3. Create DAO Interface class which contain the methods that perform CRUD functionality or SQL queries. The interface class needs to be annotated with @dao and interface methods need to annotated with query attributes e.g. @Insert, @Update, @Delete or @Query 4. Create a Database class which will represent the database in SQLite. This class needs to have the @Database annotation with the tables defined i.e. which model classes are in this database. This class also needs to be an Abstract class which extends ‘RoomDatabase’. It also needs to contain abstract methods with 0 parameters which link to the DAO interface classes that are annotated with @dao.   */\* Screenshot your code* *\*/*    *Example of interacting with database* | | | | | | |
| **Q3.9** Explain how do you use a SQLite management tool to manage the database for this app | | | | | | |
| SQLiteDatabaseBrowser was the SQLite management tool used to manage and verify database data for the app. This tool can open the application’s .db files which can be accessed via Android Studio’s ‘Device File Explorer’ window. The SQliteDatabaseBrower is like any other SQL management tool, and was used to view Table data and perform SQL queries. | | | | | | |
| **Q3.10** Test the UI according to test plan using emulator or android mobile device, report test result using template. | | | | | | |
| */\* Test report file name* *\*/* | | | | | | |
| **Q3.11** Address any issue found in the test report. Use a software related issue as an example, explain how do you use IDE embedded debugger for troubleshooting and fix the bug. | | | | | | |
| */\* Problem Description* *\*/*  */\* Troubleshooting process* *\*/*  */\* Cause of the issue* *\*/*  */\* Solution Description* *\*/* | | | | | | |
| **Checklist (To be completed by the learner’s facilitator)** | | | | | Yes | No |
| 1. Test plan for Phase 3 has been created | | | | |  |  |
| 1. Android development features for accessing local database have been identified. | | | | |  |  |
| 1. Database interfaces for the mobile app have been implemented | | | | |  |  |
| 1. Database operations including CRUD functionalities have been verified | | | | |  |  |
| 1. Using DBMS management tool for local database has been demonstrated | | | | |  |  |
| 1. Test has been finished according to plan and results are reported | | | | |  |  |
| 1. Issues identified during test have been addressed | | | | |  |  |
| 1. Software debugging process has been demonstrated and recorded | | | | |  |  |
| **Assessor Name** |  | **Assessor Signature** |  | **Date** |  | |

**Phase 4 – Server-side database design**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Although it is recommended, but you don’t need to finish Phase 2 & 3 before starting Phase 4 & 5. Client-side applications depend on server-side services, but server-side services are designed to work independently in a “passive” mode.  In phase 4 you are required to design the server-side database for the system to store the data.  You also need to confirm the server-side technology stacks for Phase 4 and 5.  You are free to select the server-side technology for the system. But you must select a different language used in Android app. That means, if you used Java for Android app, server-side Java technology is not an option. Common server-side stacks are:   * [XAMPP](https://www.apachefriends.org/index.html) or [Uniform Server](https://www.uniformserver.com/) * [ASP.Net](https://dotnet.microsoft.com/apps/aspnet) * [Node JS](https://nodejs.org/en/) * [Java](https://www.java.com/)   In addition, you are also required to practice the following:   * Develop according to code conversion document * Follow the MVC design pattern to layer source code properly. * Document source code as you program * Manage source code with version control software | | | | | | |
| **Q4.1** Confirm server-side stack for Phase 4 & 5. | | | | | | |
| |  |  | | --- | --- | | **Name of server-side stack** | Retrofit & GSON | | **OOP language** | C# | | **Database** | “PhonebookDB” | | **DBMS management tool** | SSMS (Microsoft SQL Server Management Studio) | | **HTTP(s) server** | **http://192.168.0.91:5000/** | | **IDE for development** | Android Studio & Visual Studio | | | | | | | |
| **Q4.2** Setup and verify the target sever-side environment. Screenshot that the server-side setup is ready. | | | | | | |
| */\* E.g. screenshot that the related services are running, and test page is working* *\*/* | | | | | | |
| **Q4.3** Design server-side database structure and screenshot the design below. | | | | | | |
| */\* E.g. screenshot Entity Relationship Diagram (ERD), or DB/table creation SQL command* *\*/* | | | | | | |
| **Q4.4** Create the server-side database and all entities (or tables) and add some sample data. Provide screenshot when it’s done. | | | | | | |
| */\* Screenshot that db and tables have been created and sample data have been injected* *\*/* | | | | | | |
| **Q4.5** Using the selected server-side OOP language to create a simple webpage that retrieve data from the database created and verify the result using web browser. Screenshot the test code and web browser result | | | | | | |
| */\* Screenshot test code* *\*/*  Code used to test the API was the ‘Swagger’ tool which implemented from Swashbuckle.Asp.NetCore NuGet package. Swagger is an API testing interface which was used to verify the API’s functionality.    */\* Screenshot web browser result \*/* | | | | | | |
| **Checklist (To be completed by the learner’s facilitator)** | | | | | Yes | No |
| 1. Server-side stack has been identified and confirmed. | | | | |  |  |
| 1. Server-side database has been designed and created with sample data. | | | | |  |  |
| 1. Server-side database is accessible by the server-side OOP language, and has been tested with web browser. | | | | |  |  |
| **Assessor Name** |  | **Assessor Signature** |  | **Date** |  | |

**Phase 5 – Server-side Web API design**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Before starting Phase 5, you are required to have Phase 4 completed so that you have a database and some sample data to work on.  For Phase 5, you are required to develop a set of web APIs for client application to consume, the following are required   * HTTP(S) based * RESTful manner * Data exchange with JSON * CRUD functionalities   You are also required to verify the web APIs with API testing tool, like [PostMan](https://www.getpostman.com/).  In addition, you are also required to practice the following:   * Develop according to code conversion document * Follow the MVC design pattern to layer source code properly. * Document source code as you program * Manage source code with version control software | | | | | | |
| **Q5.1** Explain what does “RESTful” mean in web API design. | | | | | | |
| */\* Your answer here* *\*/*  “RESTful” refers to the methodology or architectural style used for data transfer, usually used for communication operations for web services. | | | | | | |
| **Q5.2** Design “RESTful” web API structure and create an API document (a Word document or HTML file). Screenshot API document below | | | | | | |
| */\* File name of API document* *\*/*  PhonebookDB\_WEB\_API.sln  */\* Screenshot API document* *\*/* | | | | | | |
| **Q5.3** Design test plan for Web APIs. Paste the test plan file name below.  Note that you are required to use API testing tool, like PostMan for API test in the phase. Not until full tested with API testing tool, should the web APIs be consumed by a client application. | | | | | | |
| */\* Test plan file name* *\*/* | | | | | | |
| **Q5.4** Test the web APIs according to test plan using API testing tool, report test result using template.  Note all issues found during the test must be properly addressed. | | | | | | |
| */\* Test report file name* *\*/* | | | | | | |
| **Q5.5** For at least one web API, screenshot the step-by-step process that you use API testing tool to test the API. | | | | | | |
| */\* Screenshot how do you do it* *\*/* | | | | | | |
| **Checklist (To be completed by the learner’s facilitator)** | | | | | Yes | No |
| 1. HTTP(s) based RESTful web APIs with CRUD functionalities have been implemented | | | | |  |  |
| 1. Web APIs have been tested with API testing tool. | | | | |  |  |
| **Assessor Name** |  | **Assessor Signature** |  | **Date** |  | |

**Phase 6 – Connect front-end Android app to back-end web APIs**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Now the Android app and web API are already. It’s time to establish web connection for the Android app and consume web APIs.  In Phase 6, you are required to utilize a third-party web API library for Android to consume the web APIs developed in previous phases.  [Volley](https://developer.android.com/training/volley) is an HTTP library that can be used for Android app. You can also choose other libraries.  Ideally, the system requires a well-designed strategy / algorithm to synchronize the local database and remote database. However, for this assessment, it is important for you to demonstrate the skill to call the API CRUD operations from Android app and get proper results. **You will NOT fail this assessment if the two databases are not synchronized.**  In addition, you are also required to practice the following:   * Develop according to code conversion document * Follow the MVC design pattern to layer source code properly. * Document source code as you program * Manage source code with version control software | | | | | | |
| **Q6.1** Identify the third-party HTTP library used for the project; and have the official documentation ready. | | | | | | |
| */\* Which HTTP library to use?* *\*/*  */\* URL to its official documentation* *\*/* | | | | | | |
| **Q6.2** What is the planned strategy to work with both local and remote databases? Consider the availability of the mobile network. | | | | | | |
| */\* Describe the strategy here \*/*  The Android app will mainly work with the local database so an internet connection is not always required for the app. Synchronisation with the remote database will occur once an internet connection is available. The database with the most recent data | | | | | | |
| **Q6.3** Design test plan for calling web API using the selected third-party HTTP library in Android app. Paste the test plan file name below. | | | | | | |
| */\* Test plan file name* *\*/* | | | | | | |
| **Q6.4** Implement web API call in mobile app. Screenshot the Android app code where web API is called. | | | | | | |
| */\* Screenshot the code* *\*/* | | | | | | |
| **Q6.5** Test the Android app calling web APIs according to test plan, report test result using template.  Note all issues found during the test must be properly addressed. | | | | | | |
| */\* Test report file name* *\*/* | | | | | | |
| **Q6.6** Implement any one of the following advanced features   * Click on a phone to make a call * Send SMS * Using camera   Provide screenshot of the source code and test result. | | | | | | |
| */\* Which feature did you implemented?* *\*/*  **Camera**  */\* Screenshot the source code* *\*/*  */\* Screenshot the test result* *\*/* | | | | | | |
| **Q6.7** Implement communicates with another device, using a communication protocol (e.g. socket)  Note this program does not need to be part of the mobile app. | | | | | | |
| */\* Which feature did you implemented?* *\*/*  */\* Screenshot the test result* *\*/*  *You must submit the project for this question* | | | | | | |
| **Checklist (To be completed by the learner’s facilitator)** | | | | | Yes | No |
| 1. Web API connectivity and data manipulation has been implemented | | | | |  |  |
| 1. Test plan has been created | | | | |  |  |
| 1. Web API connectivity and data manipulation has been tested and test results have been recorded. | | | | |  |  |
| 1. One advanced feature has been implemented | | | | |  |  |
| 1. Inter device communication has been programmed | | | | |  |  |
| **Assessor Name** |  | **Assessor Signature** |  | **Date** |  | |

**Phase 7 – Software test and performance analysis**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In the previous 6 phases, all the subsystems have been created. It is the time to create a “end-to-end” test plan to verify the all required functionalities according to the overall design specification.  Also in this Phase, you are required to analysis the performance of the Android app, and report the result. | | | | | | |
| **Q7.1** Design end-to-end test plan overall system according to system specification. Paste the test plan file name below. | | | | | | |
| */\* Test plan file name* *\*/* | | | | | | |
| **Q7.2** Test the system according to test plan, report test result using template.  Note all issues found during the test must be properly addressed. | | | | | | |
| */\* Test report file name* *\*/* | | | | | | |
| **Q7.3** Address any issues found in the test report. Explain how do you troubleshoot and resolve the issue. | | | | | | |
| */\* Problem Description* *\*/*  */\* Troubleshooting process* *\*/*  */\* Cause of the issue* *\*/*  */\* Solution Description* *\*/* | | | | | | |
| **Q7.4** Establish at least 3 performance objectives and explain how did you apply them to the Android app? | | | | | | |
| */\* 3 objectives* *\*/*  */\* How are they applied* *\*/* | | | | | | |
| **Q7.5** Conduct performance analysis and report the result using provided template. | | | | | | |
| */\* Analysis report file name* *\*/* | | | | | | |
| **Q7.6** Usingproper tools to identify useless and garbage application on a target device, then delete them. Explain how did you do it? | | | | | | |
| */\* Explain how you did it* *\*/* | | | | | | |
| **Q7.7** Review application against system specifications and address variances if there are any. Please make notes below. | | | | | | |
| */\* Review notes* *\*/*  */\* The following variances have been address and explain how* *\*/* | | | | | | |
| **Q7.8** Present the system to client and get approval from the client | | | | | | |
| Client signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | |
| **Checklist (To be completed by the learner’s facilitator)** | | | | | Yes | No |
| 1. End-to-end test plan has been created | | | | |  |  |
| 1. End-to-end test has been executed and results have been reported | | | | |  |  |
| 1. All issues found during the test have been addressed. | | | | |  |  |
| 1. Learner has presented the system to client and client has approved the result. | | | | |  |  |
| **Assessor Name** |  | **Assessor Signature** |  | **Date** |  | |

**Phase 8 – Documentation and quality control**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In this phase, you are required to create a user level document.  Also you need to confirm the following requirements are met for quality control purpose   * Develop according to code conversion document * Document source code as you program * Manage source code with version control software | | | | | | |
| **Q8.1** Develop user help document for the system in HTML format. | | | | | | |
| */\* Screenshot the document* *\*/* | | | | | | |
| **Q8.2** Explain and confirm that your codes follow provided conversion document | | | | | | |
| */\* Explain and screenshot code samples* *\*/* | | | | | | |
| **Q8.3** Confirm that your codes are fully documented | | | | | | |
| */\* Confirm and screenshot code samples* *\*/* | | | | | | |
| **Q8.4** Confirm that your codes are managed by a version control system | | | | | | |
| */\* Confirm and screenshot the version control system,* ***URL must be included*** *in the screenshot \*/* | | | | | | |
| **Checklist (To be completed by the learner’s facilitator)** | | | | | Yes | No |
| 1. User level document has been created | | | | |  |  |
| 1. Development followed provided conversion document | | | | |  |  |
| 1. All codes are documented. | | | | |  |  |
| 1. Source codes are managed by version control system. | | | | |  |  |
| **Assessor Name** |  | **Assessor Signature** |  | **Date** |  | |

**Phase 9 - Publish mobile app and handover project**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| The mobile, user document (html page) need to be published for public download. You are required to   * Research and document how to publish Android app to [Google Play Store](https://play.google.com/) * Publish Android app to either 1) Google Play Store; or 2) a website with public access * Publish user document (html page) to a website with public access | | | | | | |
| **Q9.1** How to publish Android app to [Google Play Store](https://play.google.com/)? | | | | | | |
|  | | | | | | |
| **Q9.2** Publish Android app to a site with public access, and screenshot the result. | | | | | | |
| */\* Screenshot result.* ***URL must be included*** *in the screenshot* *\*/* | | | | | | |
| **Q9.3** Publish user document to a site with public access, and screenshot the result. | | | | | | |
| */\* Screenshot result.* ***URL must be included*** *in the screenshot* *\*/* | | | | | | |
| **Checklist (To be completed by the learner’s facilitator)** | | | | | Yes | No |
| 1. Learner understand how to publish Android app to Google Play Store | | | | |  |  |
| 1. Android app has been published with public access. | | | | |  |  |
| 1. User document has been published with public access. | | | | |  |  |
| **Assessor Name** |  | **Assessor Signature** |  | **Date** |  | |

*End of document*