Tutorial Letter 104/2/2016

Interactive Programming

ICT2612

Semester 2

School of Computing

IMPORTANT INFORMATION:

This tutorial letter contains important information regarding Assignment 3 (Portfolio / Application).

DUE DATE: 7 October 2016

UNIQUE NUMBER: 721266

BAR CODE

Contents

1. Introduction

This tutorial letter contains the detailed information on the applications that you may choose from to develop as well the instructions on how, when and where to submit the final application.

2. Getting started

This document contains specifications of the Android Applications. You should choose any **ONE** of these applications to design, develop and implement as your final project in this module.

Read and study all the specifications of all the applications before you make a choice. Once you've made a choice, stick to it. It is very difficult to change half-way through the semester and start with a new application.

The purpose of the application is to test your programming skills in both Java and Android that you will be learning during the semester.

Do not leave the development and implementation to the last. It is going to take a lot of your time and hence, plan it from the start and continue with the development whilst you are working through the learning units.

The learning units and examples of code provided to you should be sufficient for most of the components. Don't forget to download and view the examples of code on myUnisa. Watch the podcasts for specific topics/aspects.

3. What to do

1) Read through the specifications of all three applications.
2) Choose any one of the applications.
3) In the past we've allowed students to *uniquely name* their applications. As from this semester you have to create an application using your STUDENTNUMBER. Thus, in the end, your application will be *yourstudentnumber.apk*, e.g. 12345678.apk where 12345678 represents YOUR student number.
4) In order for the markers to know the purpose of your application, you are going to indicate to the marker as part of the comments in the code the purpose of your application.

4. The applications – in general

You will notice that the applications are in general very similar, although the functionality and application thereof differ. It is important that you first read and study all of the specifications before making a decision. You will need to demonstrate your Java and Android skills in the application. It is your choice where in the application you are going to apply the skills learned. The specifications below are the MINIMUM specifications.

5. Hints and tips

Make copies of your code (the complete folder) to a separate device on a REGULAR basis. You are welcome to use the Dropbox on myUNISA for this purpose – in fact it is encouraged that you use it. When you are satisfied that a certain portion of the application is working, copy and save it to a separate device. Don't get caught by system failures or by your own mistakes such as overwriting working code.

It is your responsibility to keep your code safe and away from other students. Plagiarism is a very serious offence and will not be tolerated. You may use and refer to code from other sources, but then you MUST indicate (reference) in your code where you sourced the code. Don't just copy and paste it – use it and modify it to be part of your code. No two people can code the same! Should it be found that you've copied from another student or from an external source without proper referencing then you will receive 0% and may be forward to the student disciplinary department.

Don't include non-trivial code in your final coding. If it is not used in the application, remove it. You will be penalised in the evaluation of the application if there is code that has no function or cannot be tested in the final application. Example: don't include an app rating widget in the application if you haven't added any code to it. Should you add images, take care that they are relevant. However tempting, don't include music or sound. It may be fun to add it to an application, but we often don't have the sound on whilst marking the portfolios.

Do include relevant comments in your final coding. Explain to the marker why you've included specific code. Use applicable variable names. "But1" doesn't really tell me anything about the button or why and where it is used in the application, same for all the other field and variable names.

TEST, TEST and again TEST while you are developing your application. That is the ONLY way that you will learn!

The last page of this tutorial letter contains the RUBRIC that will be used by the marker whilst assessing your application.

The next sections will explain to you the general requirements for all applications and then go into explaining the different applications that you may choose from.

6. General application specifications

SOUTH AFRICANISE

Unless you are not living in South Africa, then "South Africanise" your application. For instance, we don't have ZIP codes and we don't have Medical "Insurance". Our currency is Rand (not Pound or Dollars) and our date format differs from other countries, e.g. DD-MON-YYYY

UNIQUE ICON

Create or download an icon for your application. If you don't know how to create your own icon then download from sites where free icons are available example: http://findicons.com/.

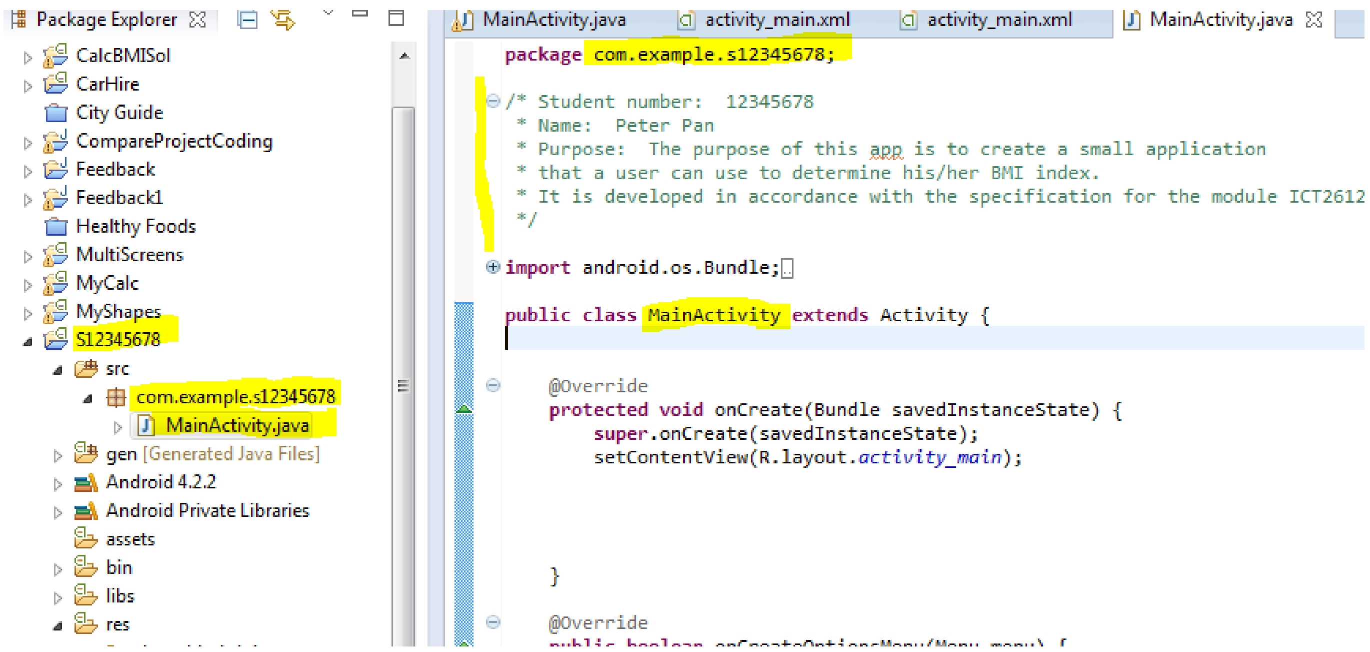
UNIQUE NAME FOR YOUR APPLICATION

You may not choose a unique name for your application (sorry!). The name for your application will be "S" + YOUR student number. Thus if your student number is 12345678, then your application's name will be **S**12345678.

VERIFICATION OF THE SYSTEM

Do not rename the `MainActivity.java` class to something else. Keep it as `MainActivity.java.`

Include YOUR student number and a short description of the purpose of the application before the import of the classes. e.g.



```
package com.example.s12345678;

/* Student number:  12345678
 * Name:  Peter Pan
 * Purpose:  The purpose of this app is to create a small application
 * that a user can use to determine his/her BMI index.
 * It is developed in accordance with the specification for the module ICT2612
 */

import android.os.Bundle;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);



    }

    @Override
```

.

However tempting, **do not** create a splash screen. The first screen of your application will be the first screen of your application.

Minimum CODING specification:

Include a number of widgets on your screens, including radio buttons, drop down boxes, etc. Take care that these widgets are properly labelled. “`textView1`” is not very descriptive but “`txtDisplayTel`” tells me something more about the purpose of the text field.

Demonstrate your Java skills by applying String manipulation, using arrays, applying OO techniques such as classes and inheritance, etc. etc. See the RUBRIC for more information.

Take care that the buttons that move from the one screen to the other is coded properly.

Add an EXIT button to your application.

7. Individual application specifications

Application 1 – Visa app

General purpose for this app:

The purpose of this app is to display visa information of the owner of the mobile phone. Should the owner be in an accident or in a situation where assistance is required then somebody else must be able to open this application and retrieve visa information and other relevant information.

Minimum specification:

This application should include at least the following: (your app may have more screens and functionality).

1) Refer to the general application specifications (section 6)
2) At least two screens.
3) A main screen displaying amongst other the following information:
   a. Name of the person, medical details, such as medical aid name, medical aid number, medical aid plan, blood type, and allergies
   b. contact details of person to contact in case of accidents and emergencies
   c. a button, when pressed, that will forward an sms to a dedicated number, providing the name and details of the person in distress as well as the GPS location.
4) An update screen that the user (owner of the phone) can use to enter and update the information that is displayed on the main screen.
   a. The information is captured and saved in a TEXT file.
   b. The main screen extracts and displays the information retrieved from the file.

Application 2 – Bingo app

General purpose for this app:

Bingo is a popular game often played at parties or when people just want to have fun. It can be played on a more formal basis where money is involved or just for the fun. Read more on the nature of bingo on: https://en.wikipedia.org/wiki/Bingo_(U.S.) There are already a variety of apps available that randomly generate the numbers that can be used during a bingo game. Have a look at them to get some ideas. The app that we are going to develop will differ slightly from those apps.

The purpose of this app is to randomly generate ("call") numbers between 0 and 99 (both inclusive). There are some rules:

- During a bingo game a number may not be called twice.
- At any given time the numbers already called must be displayed in ascending order to the players.
- The game ends when somebody calls "*Bingo*". At that stage the numbers called must be displayed to the players in order to verify that Bingo is indeed reached, if not, the game must proceed and the next number called.
- If Bingo is reached then the set of numbers should be saved to a text file, e.g. Bingo 1: 1 15 26 27 30 33 41 48 51 52 55 61 63 85 86 99
  Bingo 2: 1 3 4 11 25 28 33 36 46 55 68 69 70 75 76 79 82 95 98
  etc.…

Minimum specification:

This application should include at least the following: (your app may have more screens and functionality).

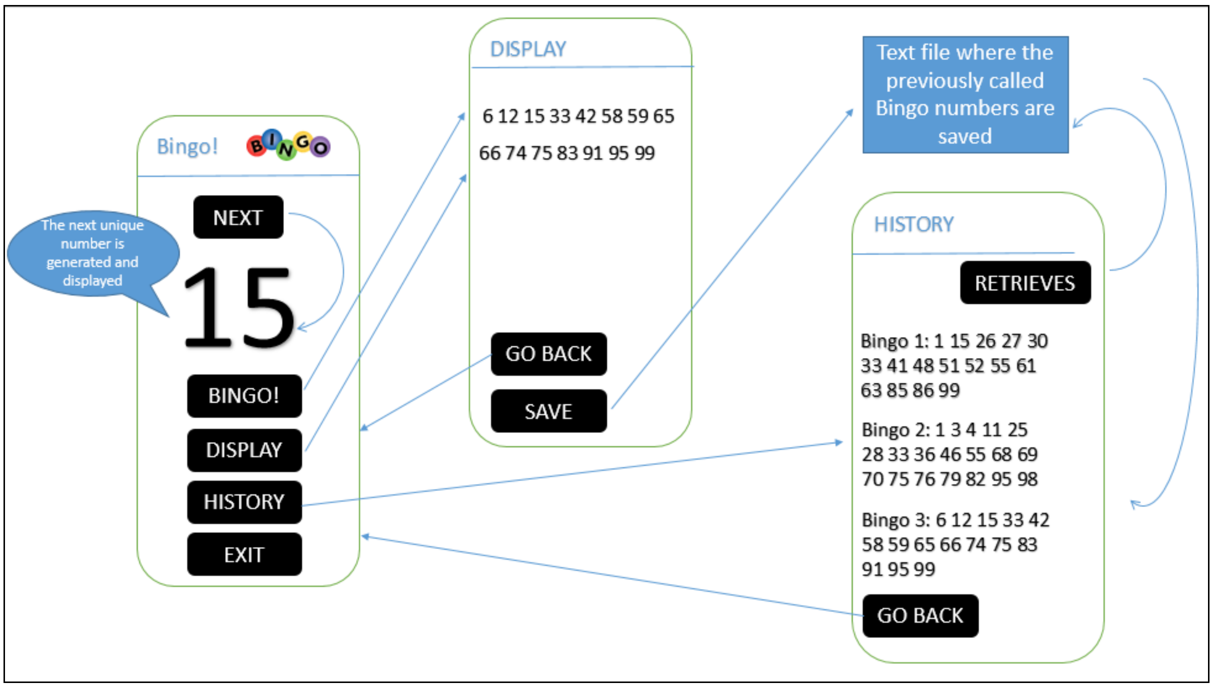
1) Refer to the general application specifications (section 6)
2) At least three screens.
   Screen 1 (Bingo!) A main screen displaying the randomly generated number. The displayed number must be large enough in order to show it to display it to the players.
   Five buttons:
   Button 1: NEXT (this button generates and displays the next number)
   Button 2: BINGO (this button will be pressed when somebody calls Bingo!. It opens Screen 2 and displays the numbers already called.)

Button 3: DISPLAY (this button opens Screen 2 and displays the numbers already called in *ascending* order.)
Button 4: HISTORY (this button opens Screen 3 and displays the contents of the text file).
Button 5: Exit and close the app.

Screen 2 (DISPLAY): A second screen displays the numbers already called during the game in *ascending* order.
Two buttons:
Button 1: GO BACK (this button is pressed when Bingo is not yet reached or when players are just viewing the already called numbers).
Button 2: SAVE (this button will be pressed when Bingo is indeed reached and the called numbers are save to file)

Screen 3 (HISTORY): The purpose of this screen is to display numbers called during the last three games.
Two buttons:
Button 1: RETRIEVE (this button retrieves and displays the information from the text file on this screen)
Button 2: GO BACK (this button takes you back to Screen 1)
For an extra challenge – instead of just displaying the contents of the text file, read the contents of the file, manipulate it and display the statistics of the called number, e.g. "1" was called 2 times. "2" was called 5 times. "3" was called 7 times". etc.

Below are some ideas on the layout of the screens to assist you.


Bingo!
BINGO
The next unique number is generated and displayed
NEXT
15
BINGO!
DISPLAY
HISTORY
EXIT
DISPLAY
6 12 15 33 42 58 59 65
66 74 75 83 91 95 99
GO BACK
SAVE
Text file where the previously called Bingo numbers are saved
HISTORY
RETRIEVES
Bingo 1: 1 15 26 27 30 33 41 48 51 52 55 61 63 85 86 99
Bingo 2: 1 3 4 11 25 28 33 36 46 55 68 69 70 75 76 79 82 95 98
Bingo 3: 6 12 15 33 42 58 59 65 66 74 75 83 91 95 99
GO BACK

Application 3 – Electricity consumption app

General purpose for this app:

The purpose of this app is to allow the user thereof to keep track of his monthly electricity expenses and always give an indication of the average electricity consumption for the house. Keep in mind that the electricity price can be adjusted in South Africa during the winter months May to August.

Minimum specification:

This application should include at least the following: (your app may have more screens and functionality).

1) Refer to the general application specifications (section 6)
2) At least two screens.

   A main screen displaying a basic input screen with five fields (some accessible by the user and others not)

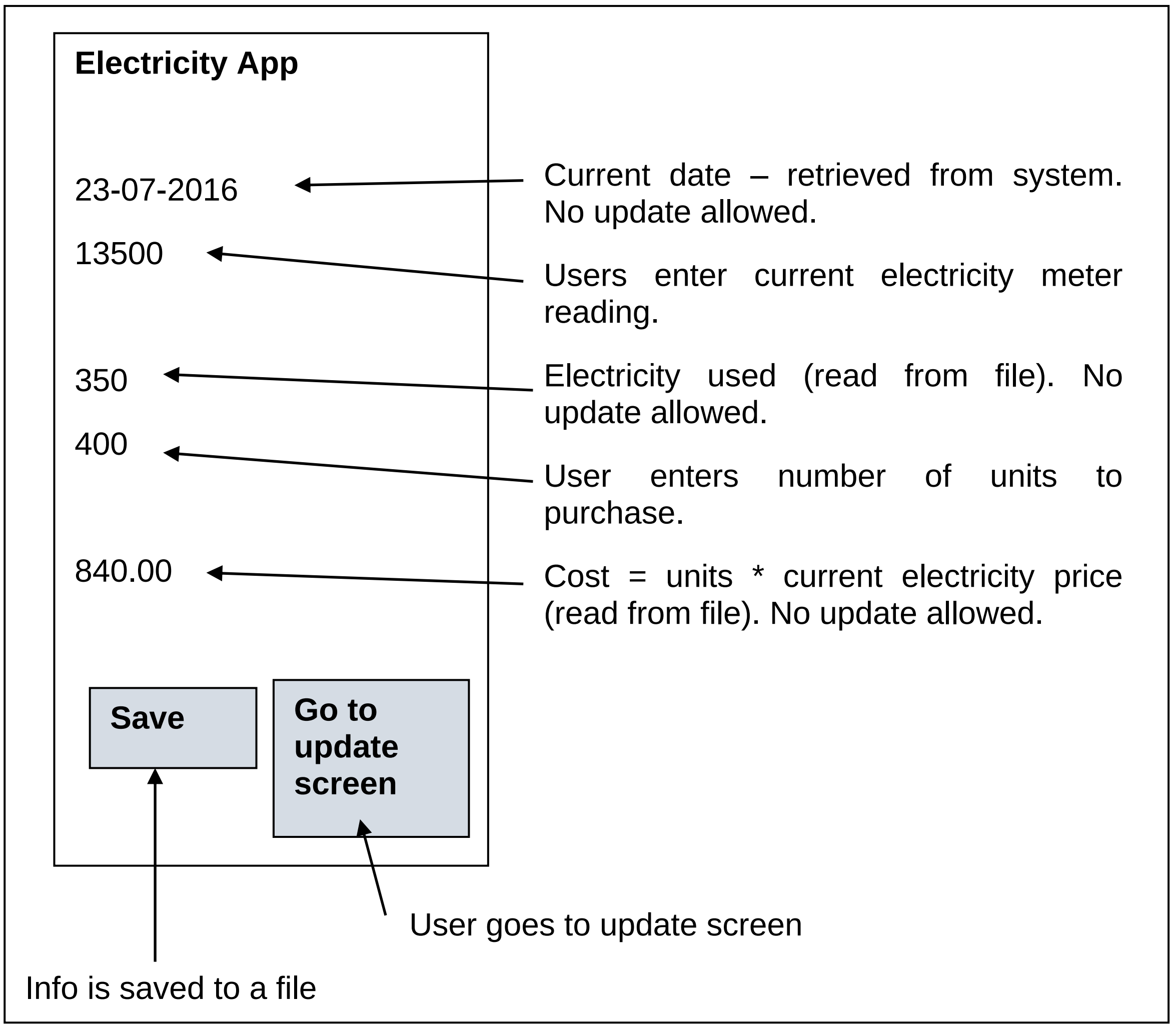
   - One *input* field for the current date (this can be done automatically as well).
   - One *input* field for the current electricity reading.
   - A *display* field (that is not accessible to the user) that subtracts the last electricity reading from the current electricity reading. The previous electricity reading is read from a text file.
   - One *input* field for the electricity units required for the house for the next month.

- One *display* field that calculates the electricity units with the current electricity price per unit. The current electricity price is read from a text file.

A button on this screen that do the required calculations and write the information to a text file.

3) The second screen is a screen that the user can use to update the new electricity price (the first Monday of the month). This value is stored to a text file.
4) This second screen also displays the average units of electricity used per month thus far as well as the total cost to date. To do this the previous information is read from a text file.

Below are some screen dumps to assist you.


Electricity App
23-07-2016
13500
350
400
840.00
Save
Go to update screen
Current date – retrieved from system. No update allowed.
Users enter current electricity meter reading.
Electricity used (read from file). No update allowed.
User enters number of units to purchase.
Cost = units * current electricity price (read from file). No update allowed.
User goes to update screen
Info is saved to a file

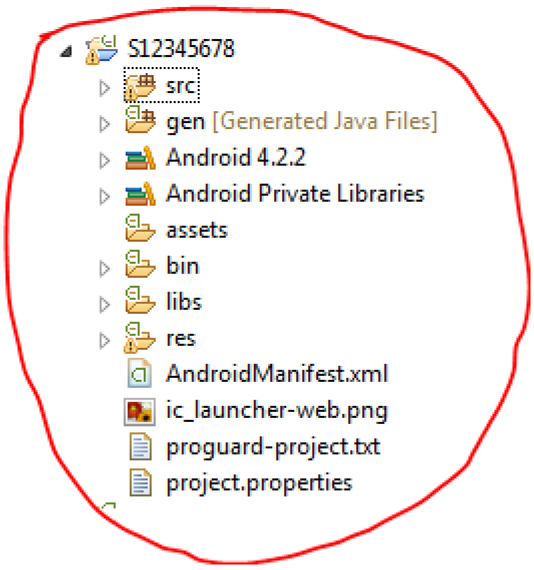
8. Submit the portfolio (assignment 3)

When you are done with the development of your application and have tested it in BlueStacks, the Virtual Environment and/or and Android device then you are ready to submit the Android application (assignment 3).

Follow the instructions below.

You are going to submit the application in two places, namely on **myUNISA** as well as in the **Dropbox**.

Dropbox: The Dropbox is only for backup purposes and will NOT be used or referred to by the marker whilst assessing the application. Create a folder in the Dropbox on myUNISA. Call it – “**Final Application”.** Copy into this folder the COMPLETE set of your code. Everything. This is for backup purposes only. The markers do not have access to your myUNISA Dropbox and will not refer to it during the assessment. However, you MUST upload the complete set of code - everything to the Dropbox. You may ZIP it and upload as one file.



```
S12345678
  src
  gen [Generated Java Files]
  Android 4.2.2
  Android Private Libraries
  assets
  bin
  libs
  res
  AndroidManifest.xml
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

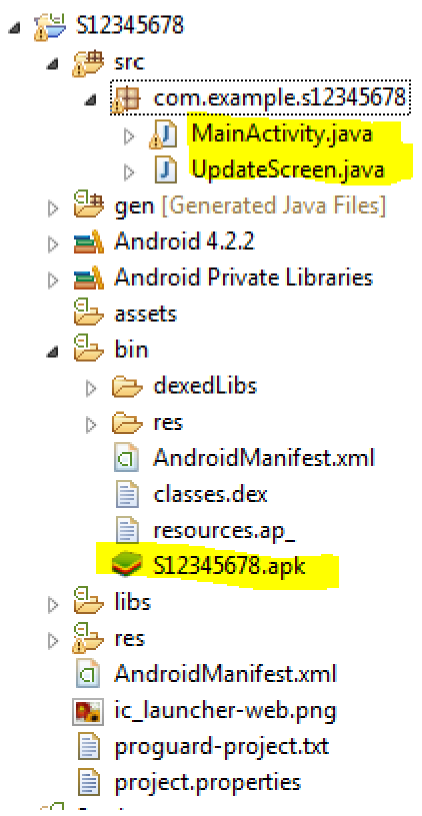
myUNISA online: You MUST submit the application (see below exactly how) online via myUNISA. Only the .apk and ,java files are required, nothing else. These are the only files that the marker will have access to and mark. Nothing else. The marker does not have access to the Dropbox. Follow the instructions described below very carefully to avoid receiving a zero or low mark for your application. Only the ZIP folder received via myUNISA will be used to assess the application.

ZIP the application:

1) Follow the instructions explained in Assignment 1 on how to create a ZIP folder for this assignment.

You must submit the following as one ZIP file

The application (.apk) file as well as the class (.java) files. In the example below it will be the MainActivity.java, UpdateScreen.java and S12345678.apk. Depending your application you may have more .java files.



```
S12345678
  src
    com.example.s12345678
      MainActivity.java
      UpdateScreen.java
  gen [Generated Java Files]
  Android 4.2.2
  Android Private Libraries
  assets
  bin
    dexedLibs
    res
    AndroidManifest.xml
    classes.dex
    resources.ap_
    S12345678.apk
  libs
  res
  AndroidManifest.xml
  ic_launcher-web.png
  proguard-project.txt
  project.properties
```

Save it as YourStudentNumber_as3.ZIP, example 12345678_as3.ZIP where 12345678 represents YOUR student number.

As explained to you in Assignment 1. Take care that your ZIP file is not corrupted, that all the files in the ZIP file can be accessed.

A day after the due date the portfolios are downloaded and forward to the markers. If the ZIP file is corrupt, or if all the files (the .apk and the .java's) are not included the marker cannot assess it and you will get 0%. The markers do not have access to the dropbox on myUNISA.

Logon to myUNISA, select ICT2612, select Portfolio (**Assignment 3**).

Follow the instructions and upload Assignment 3.

Please note that we will NOT be able to give you feedback on the application or warn you that there are missing or corrupted files. Although the portfolio is seen as an assignment is also seen as part of your examination mark, thus you are not going to receive feedback or have another opportunity to submit. It is your responsibility to ensure that everything is uploaded in time and correct.

We are looking forward to your application!

9. RUBRIC

Student			**0%**
Files included	☐ java FALSE	☐ apk file FALSE	
Criteria	**Total marks**	**Student mark**	**Comments (marker)**
General (max 10 marks)			
Launch icon	1		
Student number used for the app	1		
Comments (purpose of the app)	2		
Other screens created	2		
General layout of screens	2		
Move between screen	2		
Widgets used in the application (max 10 marks)			
Various Buttons	2		
Radio boxes	2		
Tick boxes	2		
Drop down lists	2		
Lists for arrays	2		
Input fields	2		
Coding (.java files) (max 15 marks)			
Variable and field names	2		
Applicable comments added by the student	2		
Applicable methods used	2		
String manipulation demonstrated	2		
Loops: For	2		
Loops: While / Do-while	2		
If statements	2		
Switch statements	2		
Arrays used for data manipulation	2		
Classes and objects to simplify coding	4		
Array lists used to display information	2		
Database: Either text or SQL lite files (max 10 marks)			
Save info to TEXT file	4		
Retrieving info from TEXT file	4		
Exception handling	2		
Extras (max 5 marks)			
Applicable images	1		
Application a solution for the purpose	5		
TOTAL	50	0	