MONASH University

FIT5046 – Mobile and Distributed Computing Systems

Android Practical Assignment (30%)

Due Friday 3 pm, 14th of May (Week 10 in Australia)

An Overview:

- The assignment is **an individual assignment**.
- This assignment requires **an interview**.
- **Demos/interviews** will be held in the labs or during the consultation times **in Week 11 and 12** (you should book a time with your tutor)
- This assignment requires **submitting Android project files** AND **a document including the screenshots of each screen** according to the order listed in this document.
- The marks allocated for all the tasks in this assignment will add up to 100 marks and it is **worth 30%** of the total mark for this Unit.

Due Date: The submission is due on **Friday 14th May by 3 pm.**

This assignment achieves the following learning outcomes of the unit:

- identify and describe different approaches and methods for building distributed and mobile computing systems;
- evaluate several models and approaches and select suitable mobile computing solution to a particular case;
- propose and develop a mobile distributed system that is appropriate to a problem domain;

Personalised Mobile Pain Diary

The Android practical assignment aims towards building a mobile and distributed, personalised pain diary application that enables individuals to record their pain experience and related factors such as their pain intensity level, pain location, physical activities (or number of steps taken daily) and mood. It allows users to view daily, weekly and monthly reports and monitor their health and progress. These reports can help the individual as well as their doctor to better understand their condition and the relationships between the environmental variables such as their pain intensity and weather.

The mobile app will interact with the public web APIs (e.g. weather public APIs) to retrieve relevant and useful information.

The assignment MUST be implemented **in Android Studio 11**.

MONASH University

Task 1 Sign in and Sign up (10 marks):

Basic level (5 marks)

a) The sign-in screen (one screen) enables existing users to **sign in** AND a button to navigate to the sign-up screen so a new user can **sign up (register)**.

The **sign-in screen** will allow existing users to enter their username (email) and password to login. The **sign-up (register) screen** will allow a new user to enter an email and a password and then click on a Sign up/Register button. After the user is successfully registered, they can be directed to the sign-in screen to login using their registered email and password. Data entry validation and error messages should be implemented where necessary. **(3 marks)**

b) To create the user interface for the sign-in and sign-up screens, you need to follow two password design guidelines and two form guidelines (refer to the mobile user interface lecture). (**2 marks)**

Advanced level (5 marks)

c) You need to use **Firebase authentication** for signing in existing users, and for signing up (registering) new users. You need to make sure all the functionalities of sign-in and sign-up work without any issues at run-time. (**5 marks)**

During the interview, you need to login to your Firebase account and under Authentication show the account of existing user(s).

Task 2 Home screen (15 marks):

Basic level (7 marks)

a) The home screen should display a related and meaningful title and picture[1]. **(1 mark)**

b) The main page will use the Navigation component of Jetpack and the navigation drawer for navigation to other screens (destinations). These destinations (fragments) include Pain Data Entry, Daily Record, Reports and Maps. All the functionalities must be implemented and work without any issues at run time. **(6 marks)**

Advanced level (8 marks)

c) The home screen will also show the current temperature, humidity and pressure (aka atmospheric/barometric pressure) for ~~Melbourne~~ your current location (city) that will retrieved from the public weather APIs using Retrofit. **(8 marks)**

Task 3 Pain Data Entry (20 marks):

The pain data entry screen will allow the user to enter their pain intensity level, pain location, mood level and steps every day as described below:

Basic level (10 marks)

a) The user will enter a **pain intensity level**, a value between 0 and 10 according to the Numeric Rating Scale (NRS). The NRS scale provides patients with a scale where the 'zero' means no pain and 'ten' represents the upper limit, the worst possible pain. The scale can be created using different

[1] Please make sure to use a copyright free picture, refer to this link for help https://guides.lib.monash.edu/CreativeCommons#s-lg-box-21505264

types of the UI elements e.g. a slider or a spinner (EditText not acceptable). Make sure your data entry widget is easy to use and clearly shows the scale and its options from 0 to 10. **(2 marks)**

b) The user will enter **the location of pain** including back, neck, head, knees, hips, abdomen, elbows, shoulders, shins, jaw and facial. You can use a spinner or check boxes or another appropriate UI component (EditText not acceptable) to achieve this. For simplicity, only one selection is allowed. **(2 marks)**

c) The user will enter their **mood** considering 5 levels including: very low, low, average, good, and very good. You need to use both appropriate icons (copyright free) and labels to represent each mood level that can be selected. To enter a mood level, only one selection is allowed. **(2 marks)**

d) The user will enter/set a goal (e.g. a recommended goal could be 10,000 steps per day). You present the 10,000 as the default value but allow the user to edit it if they want. The user will also need to enter the total number of their physical steps taken for the current day (because of using an emulator, we will enter the steps manually). You need to decide on the best data entry UI widget. **(2 marks)**

e) This screen must include a Save button and an Edit button. After the Save, the data entry must be disabled, and only enabled if the Edit button is clicked. You need to provide confirmation messages for a successful save or edit, and error messages if any data is missing (a Toast can be used) **(2 marks)**

The Save button will save the entries in the Room database as described in Task 4. The Edit will update the record in the Room database. The user can perform the update as many times as they like. The Save and Edit operations are covered in Task 4.

Advanced level (10 marks)

f) In real clinical studies, it is essential to collect the daily data at a fixed time to comply with data quality standards and ensure accuracy and reliability of results in statistical tests. The Pain Data Entry screen will first ask the user for entering a specific time (e.g. 4pm) using the Android **TimePicker**. Then the app will **send a notification message** (e.g. using a Toast) to remind the user to enter daily record **2 minutes before** their selected time (e.g. if the selected time is 4pm, the reminder will be displayed at 3:58pm). You need to achieve this task using Android **AlarmManager**. During the interview, you must make sure that this functionality can be tested by setting a time and getting a reminder. Here, you do not need to use a Service or WorkManager.

Task 4 Daily Record using Room Database and LiveData (10 marks):

Basic level (5 marks)

a) You will store the user data locally using **Android Room**. The database will have only one flat table called the PainRecord table. The PainRecord table will store the pain intensity level, pain location, mood level, steps taken and the date of entry (one entry allowed per day), and the temperature, humidity and pressure for the day (this data will be retrieved from a public API), plus the user email per each record (it is OK to repeat this in each record). The user email is entered during the login. The implementation should include the Room components plus the Repository and ViewModel (with LiveData). You need to consider a unique primary key for the table such as an auto-generated id. **(2 marks)**

b) In Task 3, when the user saves data, all the four pieces of daily data plus the current date, the current temperature, humidity and pressure, and the user email) will be inserted into the PainRecord table. The Edit in Task 3 is used to update the data using the current day. **(3 marks)**

Advanced level (5 marks)

c) The Daily Record screen will show all the data stored in the Room table as a list using **a RecyclerView**. The data displayed in the RecyclerView must be automatically **updated using the Room's ViewModel and LiveData** based on the observer design pattern. **(5 marks)**

Task 5 Reports (20 marks):

Basic level (10 marks)
The Reports screen will provide the user with three types of reports (pie chart, line chart, and bar chart) that will require querying the PainRecord table.

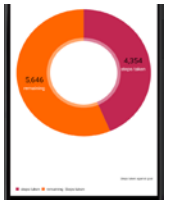
To provide sufficient data, make sure you **insert daily records for at least ten subsequent days.** (Note: You could enter any meaningful temperature, humidity and pressure data for these ten days yourself for this task but in Tasks 3 and 4 you must use the weather API to get these weather variables at run time)

Since different types of charts are used for providing reports, you need come up with an appropriate design to show each chart separately. You could either provide the user with a menu (secondary navigation, e.g. tiles using buttons) to select the type of the graph and then navigate to it, or use a container in the screen and replace the charts within the container, or any acceptable good design option.

a) **Pain location pie chart**: this screen is used to create a pie chart (any type) for the pain location data (for all the days stored). You need to use a query that will return all the pain location names and their frequencies (e.g. neck 4, shoulder 3, back 12 …). The labels and percentages should be shown on the chart. **(6 marks)**

This is just an example

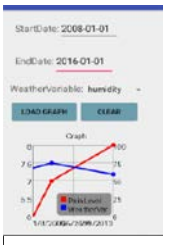
b) **Steps taken pie chart**: This chart shows the steps entered for the current date and the remaining steps (based on the set goal), using a donut pie chart. Make sure the chart has labels and legend below the chart. **(4 marks)**



This is just an example

Advanced level (10 marks)

c) **Pain and Weather Line chart**: this screen allows the user to enter a starting date and ending date using the Android's DatePickerDialog (**a popup date picker**) and also select one of weather variables (temperature, humidity or pressure) from **a spinner** to create a line graph with **two Y axes** where one **Y axis represents pain** and the **second Y axis is for the selected weather variable** (and **the date as X axis**). The labels should be shown on the graph. **(6 marks)**



This is just an example

d) When the Pain and Weather line chart is displayed, there should be a button to perform the correlation test for the pain and the selected weather variable. Appendix A provides the Android code to perform a correlation test. The correlation test will return a r value and a significance value. Display these values as well. **(6 marks)**

This is just an example

MONASH University

Task 6 Map (15 marks):

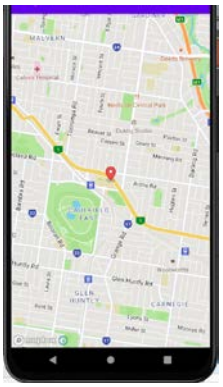
Basic level (7 marks)

a) The map screen will first ask the user to enter an address (a valid/correct and full address) to show the user's home location. You need to programmatically convert the full address of the user based on their address and postcode into latitude and longitude values (e.g. using Android Geocoder). Then use the latitude and longitude values for displaying the location on the map. **(7 marks)**

You can use any map API to complete this task, e.g. Mapbox, MapQuest or any other free APIs.

Advanced level (8 marks)

b) The location of the user is shown on the map using a marker. If you use Mapbox, you need to use Mapbox Annotation Plugin for Android, and add the appropriate dependencies.



This is just an example

Task 7 Advanced Features (10 marks) HD+:

a) You need to create a Firebase database to store the PainRecord data on a daily basis. Then, in the Android app, your code logic should make sure at the end of every day around 10:00 pm, the daily record is written (added) to the firebase database. You need to use Android WorkManager to achieve periodic daily jobs at around a specified time.

Using WorkManager, short delays are expected. Therefore, we will not be able to test this during the demo. You must use the Logcat for displaying messages and variable values within the WorkManager code where possible and then provide screenshots of Logcat for one day to prove the WorkManager works during the interview (and where the Logcat code is used). You need to include these screenshots in the document that is part of this assignment submission. **(10 marks)**

Additional Marking Criteria

Marks will be deducted if the following five criteria are not achieved.

Mark deduction could be significant (even more than 50%) without full functionality or not being able to answer interview questions

Additional marking criteria are described below. Not meeting any of these criteria can result in mark deduction:

1. **Full functionality of all the operations (during the interview)**
2. **During answering interview questions, showing a deep understanding of their code and the program logic, knowing exactly where the code for each functionality is (in which class and in which method), and answering questions correctly without hesitation during the interview (during the interview)**
3. **Handling all the exceptions and user data entry validation (and preventing the program from crashing)**

4. **Following coding standards, proper and meaningful naming of variables and methods, and providing brief but useful code comments where appropriate, avoiding code repetition**

5. **Following the basic user interface design guidelines such as consistency, alignment, balance, right colour contrast, right and consistent font type and size, avoiding clutter, visibility of objects and text.**

Submission Guideline:

A ZIP file will be uploaded to Moodle by the deadline including the following files **(any missing file in the zip file will result in mark deduction from your total marks):**

1. You need to provide **screenshots of all your screens in ONE word document** in a proper order with a title (NOT each as a separate file).
2. The **Andriod project** including **all the packages and classes and files**.
3. The zip file should have this name: **FIT5046AssignAndriod-[studentsurnname]-[studentid]-[tutor name].zip**

Late Submission:

There will be **10% penalty per day**.
The **request for any special consideration must be submitted through the special consideration website:** https://www.monash.edu/exams/changes/special-consideration Please provide them with supporting documentation such as a medical certificate prior to the submission deadline (NOT after).

Plagiarism and Collusion

Your assignment will be submitted to a Similarity Detection System that has been trained to check the programming code.

Before submitting your assignment, **please make sure that you have not breached the University plagiarism and cheating policy**. It is the student's responsibility to make themselves familiar with the contents of these documents.

Please also note the following from the Plagiarism Procedures of Monash, available at http://www.policy.monash.edu/policy-bank/academic/education/conduct/plagiarism-procedures.html

Collusion occurs when a student works with others to produce work, which is then presented as the student's own work, or the work of the other person(s). Collusion includes when a student without the authorisation of staff involved in the teaching of a unit:

- **works with one or more people to prepare and produce work;**
- **allows others to copy their work or shares their answer to an assessment task;**
- allows someone else to write (with the exception of instances where the use of a scribe is approved by the Disability Liaison Unit) or edit their work (noting that proofreading is acceptable, provided it is compliant with the University's definition contained in the Academic Integrity Policy);
- writes or edits work for another student; or
- offers to complete work or seek payment for completing academic work for other students.

Appendix A

```
import androidx.appcompat.app.AppCompatActivity;

    import android.os.Bundle;
    import android.widget.TextView;

    import org.apache.commons.math3.linear.MatrixUtils;
    import org.apache.commons.math3.linear.RealMatrix;
    import org.apache.commons.math3.stat.correlation.PearsonsCorrelation;

    public class MainActivity extends AppCompatActivity {

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

            TextView textView=findViewById(R.id.textView);
            textView.setText(testCorrelation());
        }

        public String testCorrelation(){
            // two column array: 1st column=first array, 1st column=second array
            double data[][]  = {
                    {1,1},
                    {-1,0},
                    {11,87},
                    {-6,5},
                    {-6,3},
            };

            // create a realmatrix
            RealMatrix m = MatrixUtils.createRealMatrix(data);

            // measure all correlation test: x-x, x-y, y-x, y-x
            for (int i = 0; i < m.getColumnDimension(); i++)
                for (int j = 0; j < m.getColumnDimension(); j++) {
                    PearsonsCorrelation pc = new PearsonsCorrelation();
                    double cor = pc.correlation(m.getColumn(i), m.getColumn(j));
                    System.out.println(i + "," + j + "=[" + String.format(".%2f",
    cor) + "," + "]");
                }

            // correlation test (another method): x-y
            PearsonsCorrelation pc = new PearsonsCorrelation(m);
            RealMatrix corM = pc.getCorrelationMatrix();

            // significant test of the correlation coefficient (p-value)
            RealMatrix pM = pc.getCorrelationPValues();
            return("p value:" + pM.getEntry(0, 1)+ "\n" + " correlation: " +
    corM.getEntry(0, 1));

        }
    }
```


MONASH University