

Tim Daiber – 3rd Year Project – Specs

Tim Daiber

0877880921

G00260494

67 Gleann Rua,

Galway,

Co. Galway

Contents

Tim Daiber – 3 rd Year Project – Specs	1
Introduction	3
Technologies	3
Implementation	4
Android Studio	4
Palette	5
XML	5
Spinner	6
Check Boxes	7
TextFields	7
Toast.....	7
Intent's	8
Calculation	8
SQLite	9
Creating the database.....	9
Creating insert statements that will populate the database.	9
a. Get all data:.....	10
b. Get average age query:.....	12
SQLite Manager.....	13
Development Ideas	15
Expected Learning outcomes.....	16
Actual learning outcomes – Conclusion.....	16
Reference	17

Introduction

My Project will be an app that will give information / calculate the life expectancy of a person.

The user will provide information about themselves (e.g. smoker, weight etc.)

This information will be stored on the database and kept.

The information will be calculated by an equation and the user will get an estimated age they will live to.

I wanted to make this app since I have never developed for android before. I am familiar with java but have never created an android app.

This project has given me the perfect opportunity to learn the development in android studio and to learn about the technologies that are commonly used in the android development environment.

For example, SQLite.

Data mining has become a very big market and I wanted to know how apps store data from an app on a database. The database that I am using is only locally stored and does not send the data to a server but I find it interesting how the data is stored in the first place. In previous projects the only database I have worked with was MySQL so I want to try out other technologies that can be used to store data.

Technologies

- Android Studio
- Java
- GitHub
- SQLite
- SQLiteManager (Firefox add on)

Implementation

Android Studio

Android Studio is a very good IDE to develop for an Android platform.

Upon Creation of a new Project the main_activity (Main Page) is the page that will be initially loaded upon start-up of the app.

In this activity I have decided to Place all the main content of the app.

The user enters all the properties via Checkboxes Spinners etc.

Most information is Displayed through Text Fields.

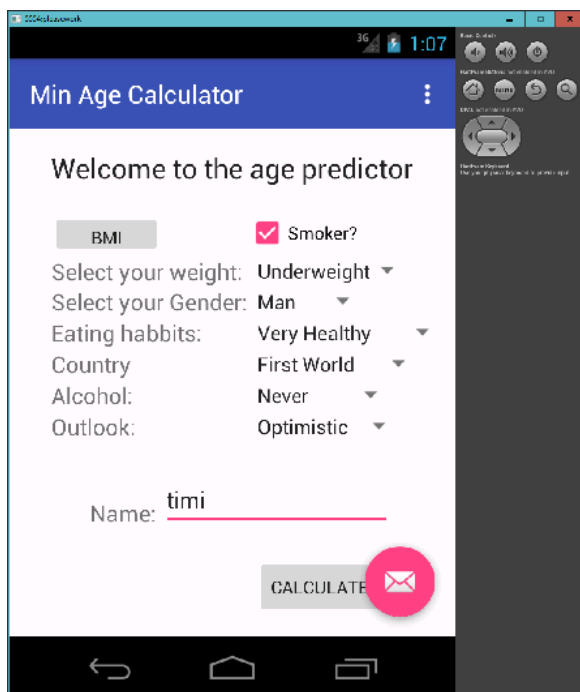


Fig 1 : Shows The main activity running in an emulator.

Activities can be manipulated in 2 ways:

Palette

Activities can be manipulated in 2 ways:

The Palette is a good and handy way to add functionality to an Activity.

It is just a drag and drop principle.

Dragging a button onto the activity will automatically create the button in the XML file for the activity.

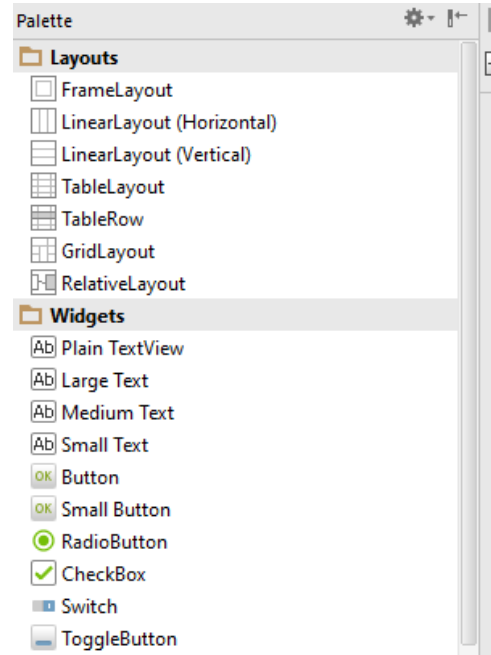


Fig 2: Shows part of the Android Studio Palette.

XML

The other way to Design the activity can be done through the XML directly.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    app:layout_behavior="android.support.design.widget.AppBarLayout$ScrollingView..."
    tools:context="com.example.tim.myfinalproject.MainActivity"
    tools:showIn="@layout/activity_main">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Welcome to the age predictor"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <CheckBox
```

Fig 3: Shows Part of the content_main.xml xml code

Of course both of these option can be used together as well.

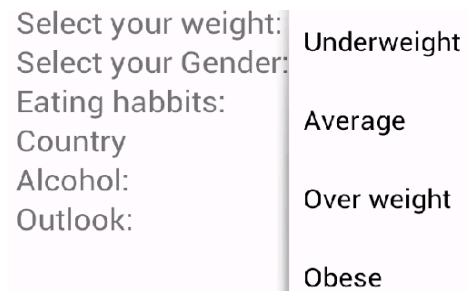
Spinner

A Spinner is basically a dropdown list in Android Studio.

To Create a Spinner you could drag it into the activity or by adding it through the XML directly

```
<Spinner
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/spinner"
    android:layout_marginTop="34dp"
    android:layout_alignTop="@+id/smoker"
    android:layout_toRightOf="@+id/textView4"
    android:layout_toEndOf="@+id/textView4" />
```

Fig 4: Shows the XML code for the weight spinner object.



Select your weight: Underweight
Select your Gender: Average
Eating habits: Over weight
Country: Obese
Alcohol:
Outlook:

Fig 5: Shows the weight spinner populated in the emulator

To add values to the Spinner the string.xml class has to be edited.

```
<string-array name="Weight">
    <item>Underweight</item>
    <item>Average</item>
    <item>Over weight</item>
    <item>Obese</item>
</string-array>
```

This is achieved by creating an Array of Strings

Where each item in the array represents one value that will be displayed in the dropdown list.

Fig 6: Shows the String array for the weight spinner in the string.xml file

To display the items in the string array the activity must be edited through some Java code.

The spinner and the array must be linked together on creation of the app for the items to appear in the spinner.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    spinner = (Spinner) findViewById(R.id.spinner);
    adapter = ArrayAdapter.createFromResource(this, R.array.Weight, android.R.layout.simple_spinner_item);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spinner.setAdapter(adapter);
    spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            Toast.makeText(getApplicationContext(), parent.getItemAtPosition(position) + " selected", Toast.LENGTH_LONG).show();
        }
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
        }
    });
}
```

Fig 7: Shows the code for populating the spinner with the string array and adds select functionality

The code snippet above shows the linking of the String array “weight” to the spinner (spinner)

It also gives the functionality of selecting an item.

Check Boxes

Checkboxes are another easy user friendly way to get information of the user.

Checkboxes can easily be accessed by searching for their assigned id and a simple if statement

Can be used to check if the checkbox is checked or not.

```
Button button = (Button) v;  
CheckBox smoker = (CheckBox) findViewById(R.id.smoker);  
if(smoker.isChecked()){  
    age = age -13;  
    issmoker = "Yes";  
    setAge1(age);  
}  
else{  
    issmoker = "No";  
}
```

Fig 8: Shows the code code to get a specific checkbox by ID and checking if it is checked.

TextFields

Textfields are an easy way to display messages to the user.

I have used quiet a few text fields to display text to the user.

Also a textField is used to display the calculated age to the user.

Toast

Toast is a class that can be used to display popup messages to the user.

```
if(isInserted ==true){  
    Toast.makeText(MainActivity.this,"Data Inserted",Toast.LENGTH_LONG).show();  
}else{  
    Toast.makeText(MainActivity.this,"Data not Inserted",Toast.LENGTH_LONG).show();  
}
```

Fig 9: Shows the code for the display Toast message.

In this toast message the user gets a little popup message to determine if the insertion into the SQLite database was successful.

Intent's

“An Android **Intent** is an abstract description of an operation to be performed. It can be used with **startActivity** to launch an Activity, **broadcastIntent** to send it to any interested BroadcastReceiver components, and **startService(Intent)** or **bindService(Intent, ServiceConnection, int)** to communicate with a background Service.”

See Reference 1.

I have used Intents to change from one activity to another.

```
Intent intent = new Intent(getApplicationContext(), Main2Activity.class);
intent.putExtra("parameter name", newage);
startActivity(intent);
```

Fig 10: Shows the code of creating new intent and passing in a value/variable.

This intent is in a Button on click. When the button is clicked the button listener will recognise that the button is clicked.

A new intent is created loading the Main2Activity class (Main2Activity Page).

A great thing about intents are that you can pass information from one activity to another through the intent. In the code snippet above I am using the intent to pass the variable “newage” into the new activity that is loaded by the intent.

Calculation

The calculation for the age is very simply done by the adding or subtracting from the base age of 85

The user's choices in the app determine what calculation is appropriate

e.g.: if smoker checkbox is ticked expected age goes down by 13 years.

SQLite

I am using SQLite to create a locally stored database on the phone.

The DatabaseHelper Class is responsible for:

Creating the database.

```
public class DatabaseHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "prediction6.db";
    public static final String TABLE_NAME = "prediction_table";

    public static final String COL_1 = "id";
    public static final String COL_2 = "name";
    public static final String COL_3 = "smoker";
    public static final String COL_4 = "weight";
    public static final String COL_5 = "gender";
    public static final String COL_6 = "eating";
    public static final String COL_7 = "country";
    public static final String COL_8 = "alcohol";
    public static final String COL_9 = "outlook";
    public static final String COL_10 = "age";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, 1);
        SQLiteDatabase db = this.getWritableDatabase();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create table " + TABLE_NAME + " (id integer primary key autoincrement,name text,smoker text,weight text,gender text, eating text,country text,alcohol text,outlook text,age integer)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }
}
```

Fig 11: Shows DatabaseHelper class and the create query for the database and a table.

An instance of the DatabaseHelper class is created upon creation of the main activity with the Columns as seen above.

Creating insert statements that will populate the database.

```
public boolean insertData(String name,String smoker, String weight, String gender, String eating,String country, String alcohol,String outlook, int age){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues contentValues = new ContentValues();

    contentValues.put(COL_2, name);
    contentValues.put(COL_3, smoker);
    contentValues.put(COL_4, weight);
    contentValues.put(COL_5, gender);
    contentValues.put(COL_6, eating);
    contentValues.put(COL_7, country);
    contentValues.put(COL_8, alcohol);
    contentValues.put(COL_9, outlook);
    contentValues.put(COL_10, age);

    long result = db.insert(TABLE_NAME,null ,contentValues);
    if(result == -1){
        return false;
    }else {
        return true;
    }
}
```

Fig 12: Shows insert method for inserting data into TABLE_NAME

The insert query is called when the calculate button is clicked.

```
boolean isInserted = myDB.insertData(name,issmoker,myweight,gender,eating,country,alcohol,outlook,newage);

if(isInserted ==true){
    Toast.makeText(MainActivity.this,"Data Inserted",Toast.LENGTH_LONG).show();
}else{
    Toast.makeText(MainActivity.this,"Data not Inserted",Toast.LENGTH_LONG).show();
}
```

Fig 13: Shows code where insertData method is called

It is responsible for all other queries:

a. **Get all data:**

```
public Cursor getAllData() {  
    SQLiteDatabase db = this.getWritableDatabase();  
    Cursor res = db.rawQuery("select * from "+ TABLE_NAME ,null);  
    return res;  
}
```

Fig 14: Shows getAllData method in DatabaseHelper class

Creates a SQL query to show all the data in the TABLE_NAME table.

This Query is linked with an AlertDialog which is linked to an onClick event of the View Details Button.

```
public void viewData(View v) {  
    //Button button = (Button) v;  
    Cursor res = myDB.getAllData();  
    if(res.getCount()==0) {  
        // show message  
        showMessage("Error", "No Data Fount");  
        return;  
    }  
    else  
    {  
        StringBuffer buffer = new StringBuffer();  
  
        while(res.moveToNext())  
        {  
            buffer.append("ID :" + res.getString(0)+"\n");  
            buffer.append("Name :" + res.getString(1)+"\n");  
            buffer.append("Smoker :" + res.getString(2)+"\n");  
            buffer.append("Weight :" + res.getString(3)+"\n");  
            buffer.append("Gender :" + res.getString(4)+"\n");  
            buffer.append("Eating :" + res.getString(5)+"\n");  
            buffer.append("Country :" + res.getString(6)+"\n");  
            buffer.append("Alcohol :" + res.getString(7)+"\n");  
            buffer.append("Outlook :" + res.getString(8)+"\n");  
            buffer.append("Age :" + res.getInt(9)+"\n\n");  
  
            // SHow it  
            showMessage("Data",buffer.toString());  
        }  
    }  
}
```

Fig 15: Shows viewData method in DatabseHelper class

Upon clicking the button, a AlertDialog is launched that will display the data gathered by the query.

```
public void showMessage(String title,String Message){  
    AlertDialog.Builder builder = new AlertDialog.Builder(this);  
    builder.setCancelable(true);  
    builder.setTitle(title);  
    builder.setMessage(Message);  
    builder.setNegativeButton("Ok",null);  
    builder.show();  
}
```

Fig 16: Shows AlertDialog method.

The OK button will allow the closing of the ALertDialog window.

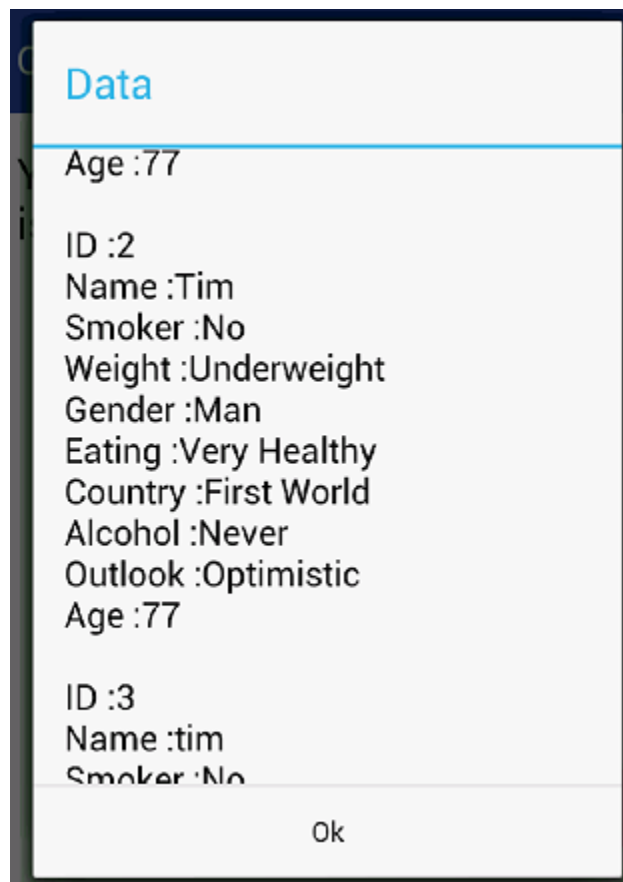


Fig 17: Shows AlertDialog Displaying data in the emulator.

b. Get average age query:

```
public Cursor getAverageAge(){
    SQLiteDatabase db = this.getWritableDatabase();
    Cursor res = db.rawQuery("select AVG(age) from "+ TABLE_NAME ,null);
    return res;
}
```

Fig 18: Shows getAverageMethod in the DatabaseHelper Class.

Gets the average age of all the data in the table TABLE_NAME.

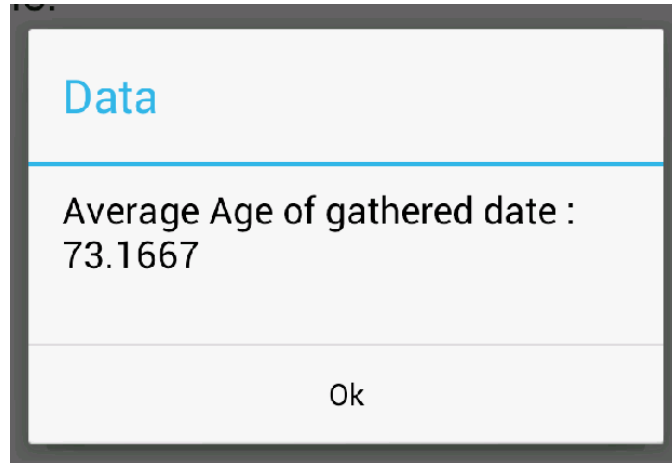


Fig 19: Shows the AlertDialog message Running in the emulator.

SQLite Manager

SQLite Manager is a FireFox add-on that lets the user open up a SQLite database and manipulate it.

I have used this Technology to Test if the database has successfully been created, If the data has successfully been inserted into the database and have used it to test the query's that I want to use in the App.

When the App runs the first time the Database is created on start-up and can be extracted out of the emulator by going into the "Android Device Monitor" -> Data -> data -> find the app in the list -> database.

The database name is prediction8.db

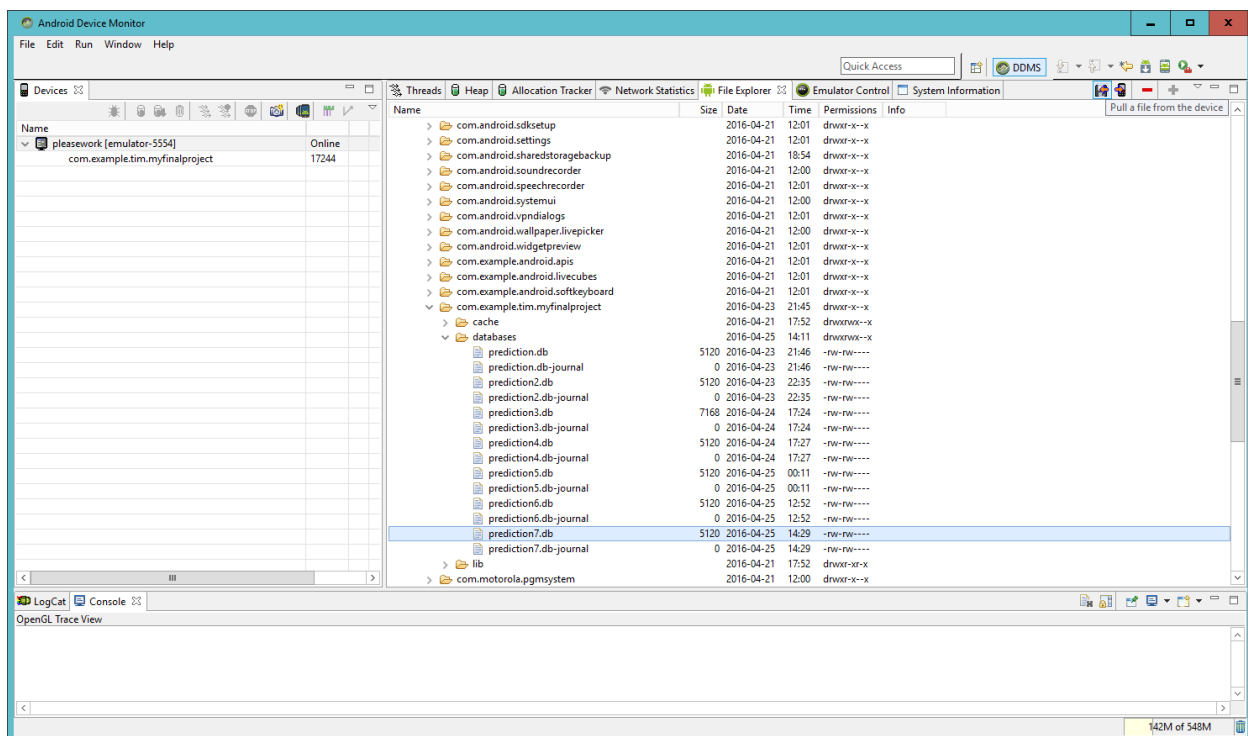


Fig 20: Shows where the database can be found on the emulator.

The database can be extracted from here.

I have saved a copy of the database in the folder.

Using the SQLite Manager add on you can open the database.

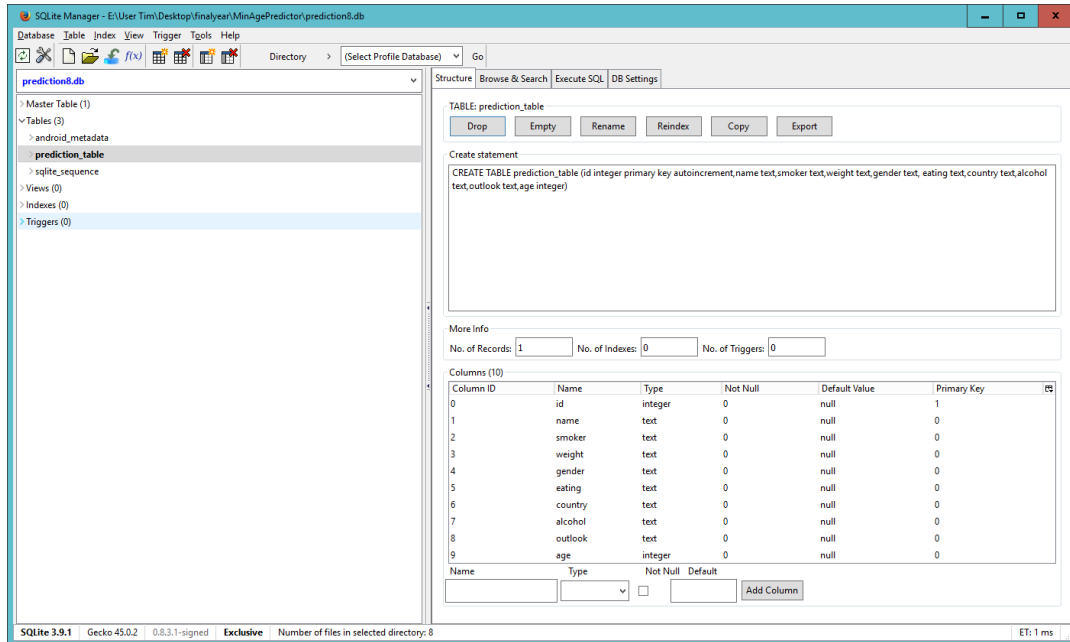


Fig 21: Shows the prediction_table in SQLiteManager add-on.

In the SQLite manager you can see the details of the Table and the create statement.

In the Brows and Search tab of the add on you can observe what entries have been made to the database.

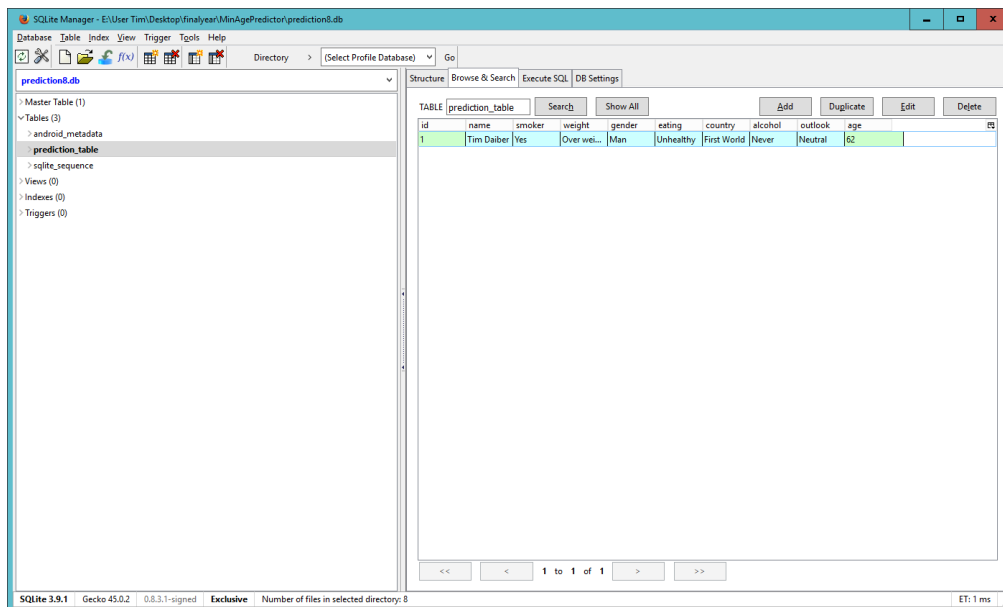


Fig 22: Shows the entries made in the SQLite manager add-on.

In Execute SQL tab queries can be tested before implementation into the app.

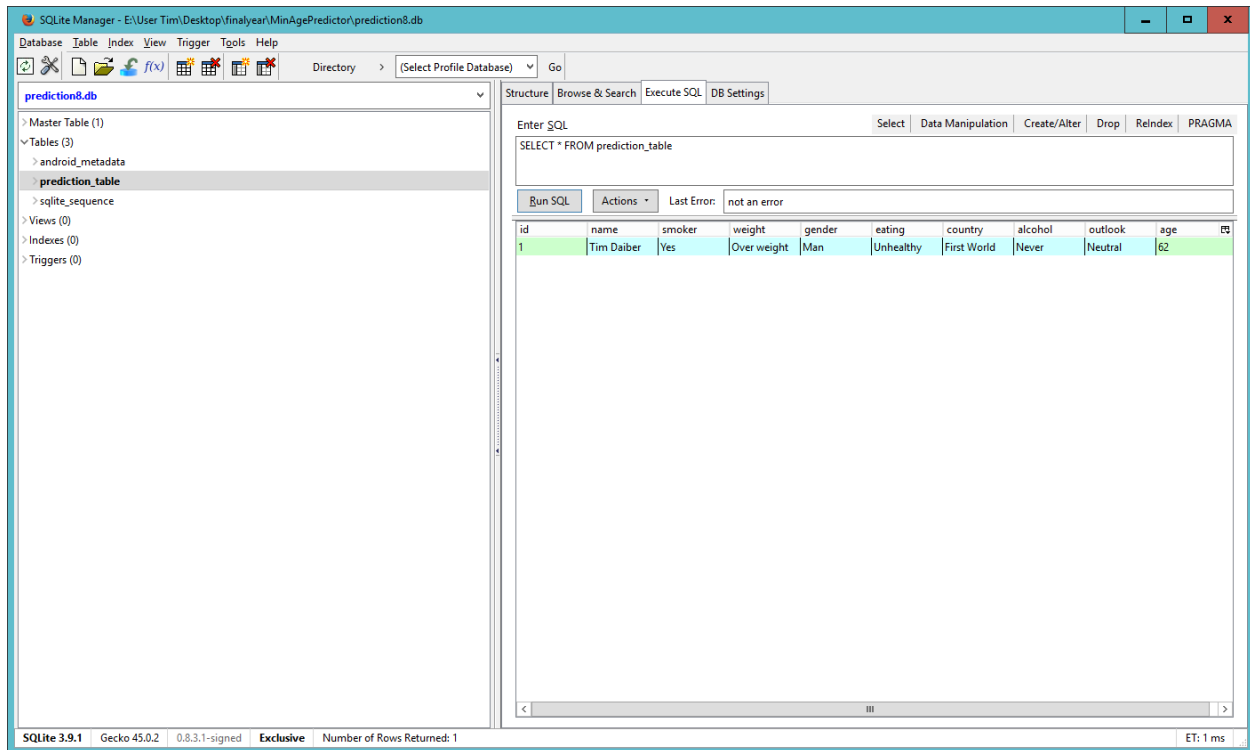


Fig 23: Shows the execute query tab in the SQLite manager add-on executing a test query.

I have included a screen cast of the extraction of the database and the use of SQLite Manager add-on in the Project folder.

Development Ideas

The Development of the app will be done on Android Studio (Java).

Android Studio will be linked with my GitHub account to store my project and make it easily accessible from anywhere.

The Database will be designed in SQLite and will hold data that the user will input / receive.

Expected Learning outcomes

Android app expertise

Database creation and management

Designing and Developing an app

Actual learning outcomes – Conclusion

In this project I have learned quite a lot.

Since I have never touched android studio before everything I have done in this project is new to me.

I also never worked with SQLite.

1. I have learned how to create an app in Android studio
2. How to add content to an activity content page through either XML or the designer.
 - a. TextFields
 - b. Spinners
 - c. EditFields
 - d. Buttons
 - e. Manipulating the content of the above named objects.
3. Create multiple activities
4. Navigate through activities using intents
5. Passing data from one activity to another using intent.
6. The use of buttons
 - a. Creating on click event methods
 - b. Linking button methods to buttons
7. Creating SQLite database
8. Entering data into the SQLite database
9. Querying the data stored in the SQLite database
10. Creating String arrays in the String.xml file
 - a. Adding the String arrays to Spinners onCreate
 - b. Using Adapters with a Spinner object.
11. Creating a Toast Message
12. Creating AlertDialog popup windows to display data
 - a. Using the Builder class
 - b. Adding a Button to a AlertDialog Message
 - c. That returns the user to the app.

I have enjoyed working on this project and will definitely create more app's in future.

I have not even scratched the surface of what is possible in the development of an app

And am now intrigued to learn more about things that can be done with these Technologies.

Reference

http://www.tutorialspoint.com/android/android_intents_filters.htm

<https://www.youtube.com/watch?v=Glu0EeMTVHY>

https://www.youtube.com/watch?v=KhSM_CRCLRo

<https://www.youtube.com/watch?v=PA4A9lesyCg>

<https://www.youtube.com/watch?v=pzf-XGqVcjM>

<http://www.death-clock.org/>

https://en.wikipedia.org/wiki/Life_expectancy

<http://never-ending-of-art.blogspot.ie/2011/05/how-to-calculate-your-bmi.html>