Final Project Report

Final Report Course: Intelligent Mobile Development

College of Software Nankai University

Faculty: 师文轩

Student Name: MIGNON CHRISTOPHER ADRIAN WILLIAM HENRY

Student No: 2120196012

GITHUB: https://github.com/chris-mignon/BMI-Tracker-

Contents

Introduction

Many people have a problem with being overweight, this can be due to a number of factors such as dietary choices, a sedentary lifestyle or Genetic factors such as low a basic metabolic rate. A person's weight to height ratio (Body mass Index-BMI) can be an indicator of their overall health and people with high BMIs are at risk for many weight related health conditions such as heart problems and diabetes.

This app is designed to aid the users who are starting a fitness routine to calculate their Body Mass Index, keep track of if over period of time thus helping them to return to a preferred state of health.

The main features of my Android App is to calculate the Body Mass Index of the user based on their weight and height and save the results in a database so that the use can keep track of their BMI over a period of time. The users can also delete records and even add notes to the BMI calculations

Solution

The Tools and technologies used in developing this App are Android studio, Room Database.

- **Android Studio** is the official integrated development environment (IDE) for Google's Android operating system, built on Jet Brains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. In includes features such as visual Layout editor with different layouts such as constraint layout, linear layout and relative layouts. Android studio also includes an intelligent code editor that provides code completion for Kotlin, Java, and C/C++ languages. The built-in Emulator allows developers to install and run apps faster than with a physical device and simulate different configurations and features, including ARCore, Google's platform for building augmented reality experiences.
- **Room** is a persistence library, part of the Android Jetpack. Room provides an abstraction layer over SQLite to allow fluent database access while harnessing the full power of SQLite. Room is now considered as a better approach for data persistence than SQLite Database. It makes it easier to work with SQLite Database objects in your app, decreasing the amount of

boilerplate code and verifying SQL queries at compile time. Compile-time verification of SQL queries. When using Room each @Query and @Entity is checked at the compile time, that preserves your app from crash issues at runtime and not only it checks the only syntax, but also missing tables Room is also easily integrated with other Architecture components like LiveData.

Details

UI (User Interface) details

The user inter face is fairly simple which enables the user to navigate through all the controls. The app comprises of two activities the main activity and the Calculate and add interface. The start screen (Main activity) includes a Recycler-view with two buttons. The recycler-view displays the records in the database and each record can be swiped to delete the record. The CLEAR ALL button clears all the records in the database and the CALCULATE BMI buttons opens the Calculate and add interface from there the user can enter their height and weight to calculate their BMI and add the result to the database.

The Calculate and add interface comprises of a switch control which is used to change the input units between metric and imperial. There are two edit-texts for the user to enter the required inputs and these inputs are indicated by two text-views. Below these controls there is a CALCULATE BMI button which when clicked calculates the BMI result based on the input and returns the value to the text-view (BMI result) below the CALCULATE BMI button. Below the BMI result text-view there is an edit-text for the user to enter a note reminder. At the bottom of the interface there are two buttons: the ADD to RECORDS button which adds the calculated result to the database of records. The CLEAR button clears the input edit-texts.

User experience Details

Firstly when the user clicks the switch unit control the user expects to see some changes indicating that the units are switched, this is indicated in the switch control changing color and the text in the text-views changing to suit the indicated units. This ensures that the user knows which units they are using and won’t enter incorrect units thus resulting in incorrect results.

When the user enters the inputs (height and weight) the edit texts only allows the user to enter numbers thus preventing an error in the app. Additionally when the user clicks the CALCULATE BMI button without first entering the required inputs the App prevents this action and displays a toast message to remind the user to enter the height and weight. Similarly when the user clicks the add to record button without first calculating their BMI the App reminds them with a toast message to first calculate their BMI this prevents the user from entering null values into the database.

The App after calculating the BMI result not only displays the result in a text-view but also changes the background to indicate which body condition is associated with the current BMI and when the clear button is clicked the background returns back to the default background.

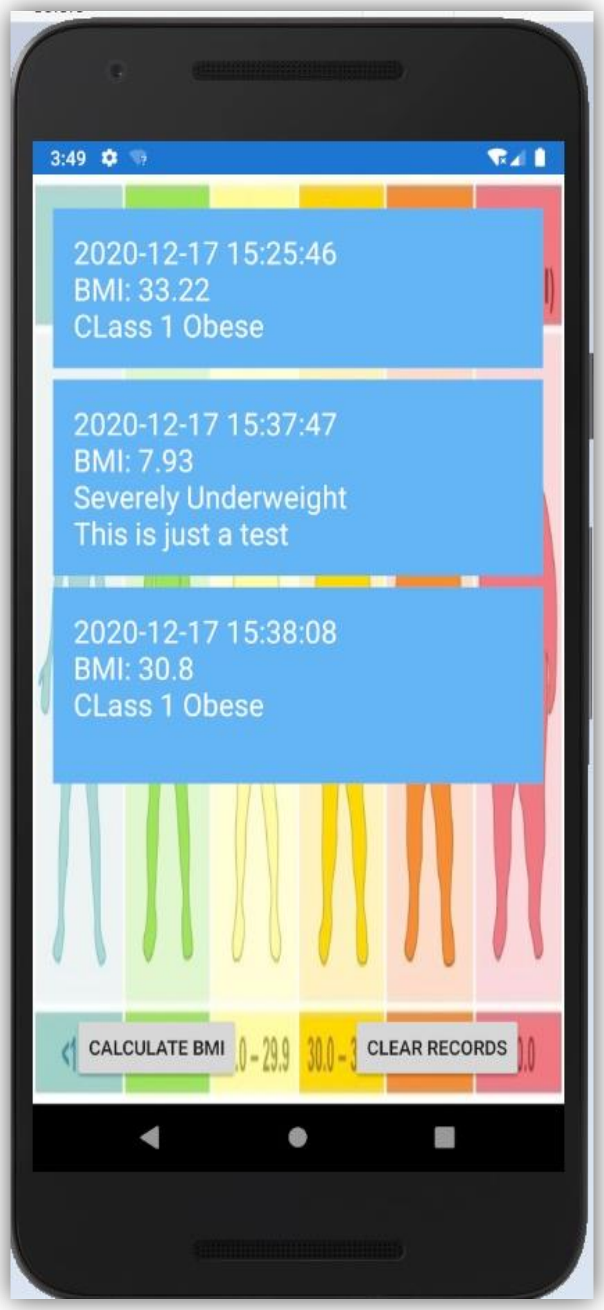
In the main interface after the user swipes a record to delete it a toast message is displayed to indicate that the record is deleted and the also when clearing all the user is also notified with a toast message.

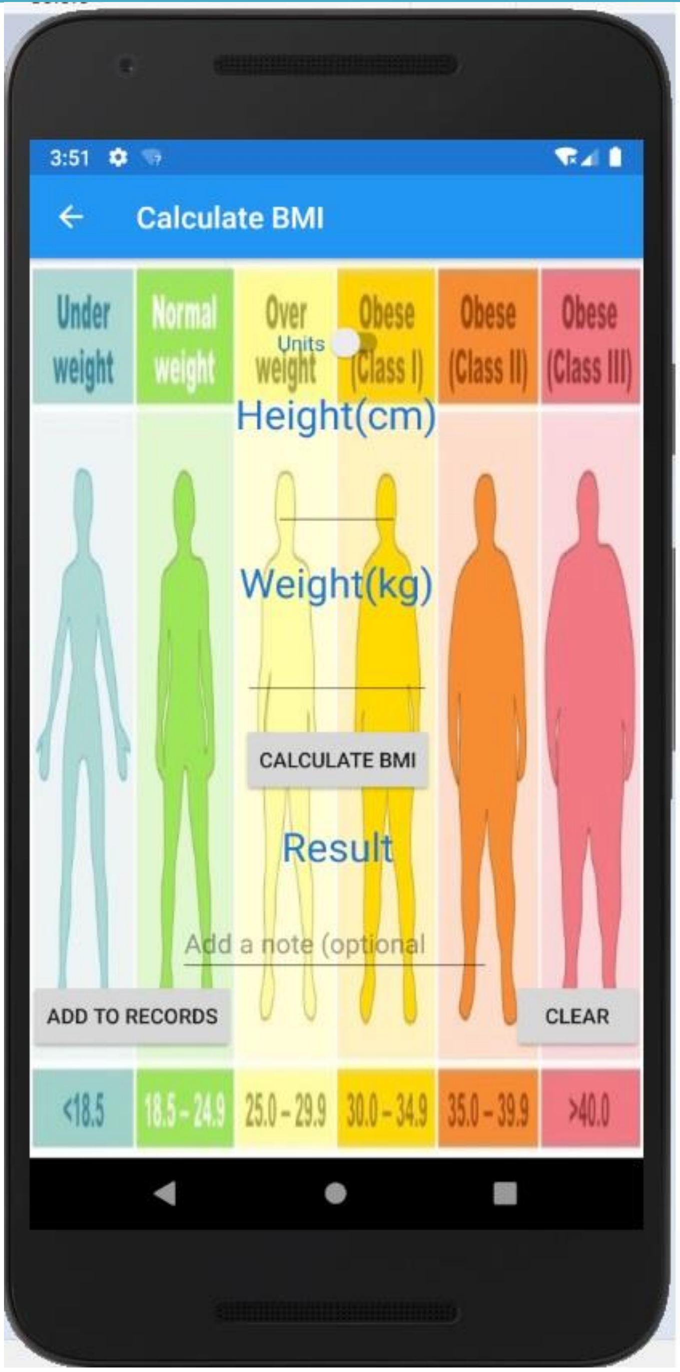
Repository Interactions

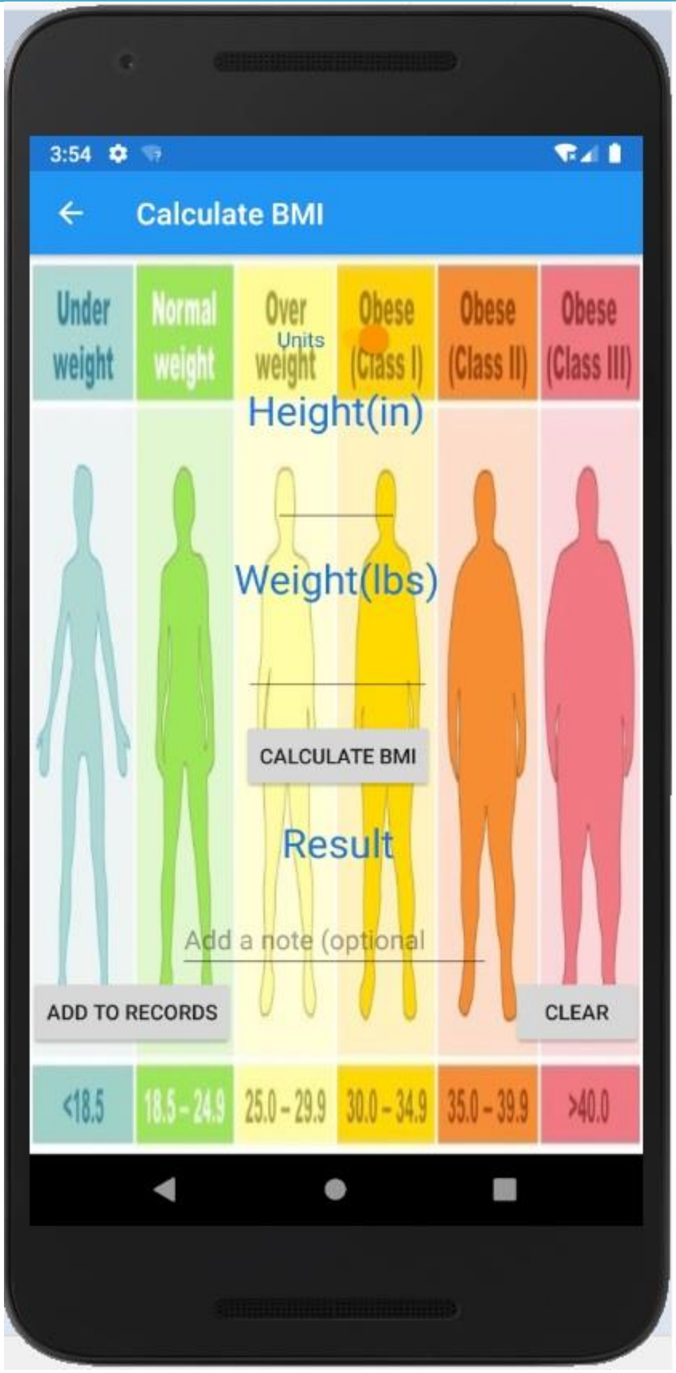
In the App all of the repository interactions are done by the view model which acts an intermediary between the user interface and the repository. There are three main methods for repository interactions which are the insert method, the delete method and the delete all method.

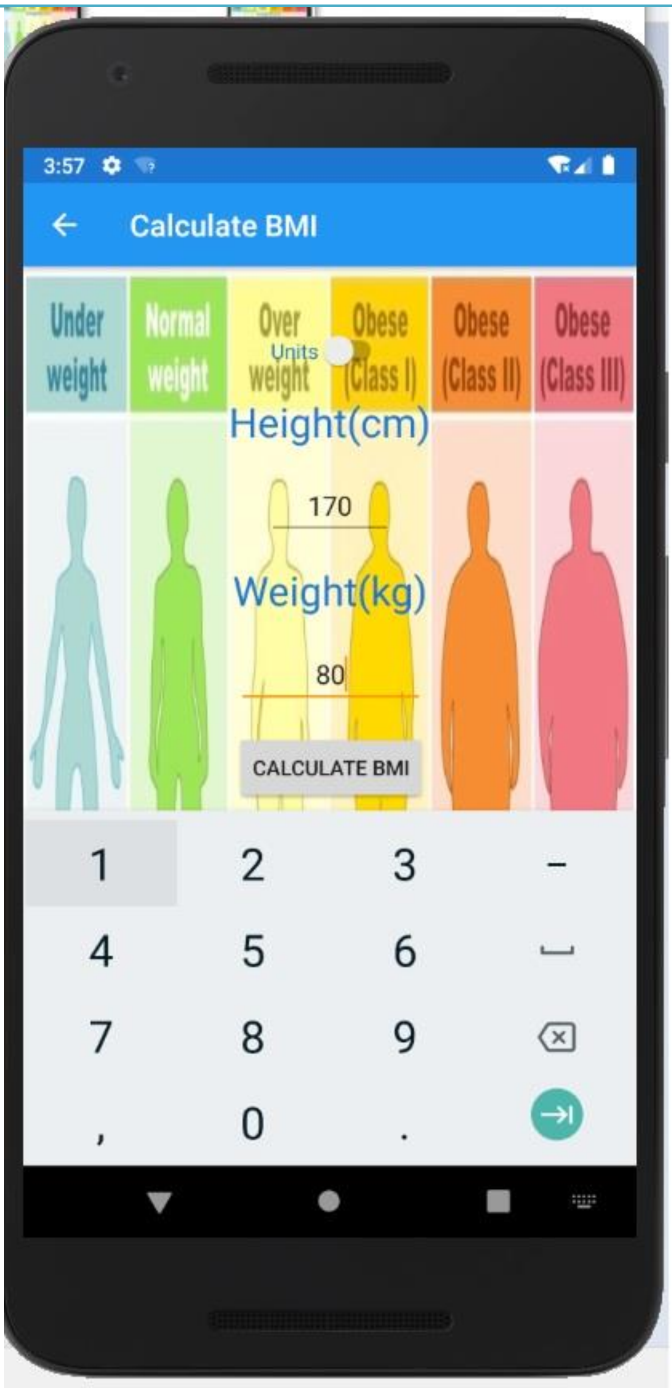
The interactions are handled by a repository object in the view model class which invokes these methods from the repository class. In the main activity class in the oncreate method a view model object is used to call the getallwords method in the repository which then populates the recycler view adapter. When the user swipes a record or clicks the CLEAR RECORDS button, the viewmodel object is used to invoke the delete and delete all methods respectively.

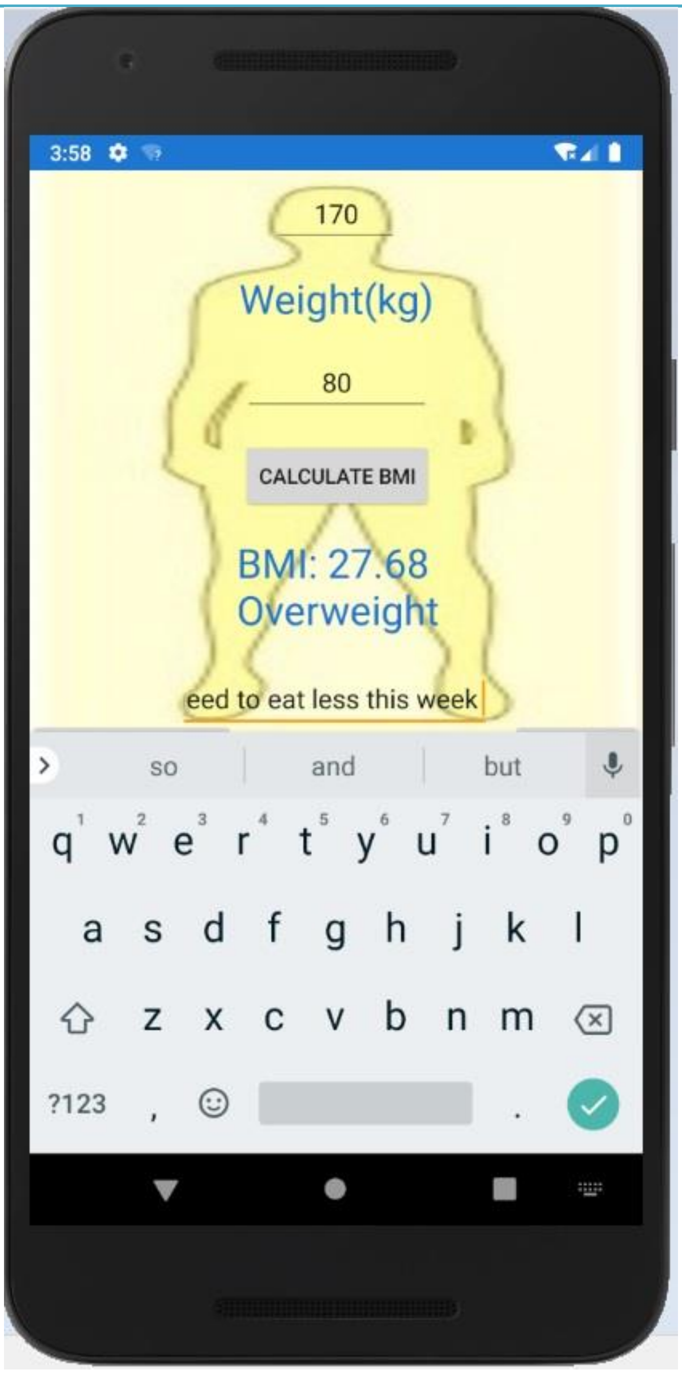
Screenshots and Descriptions

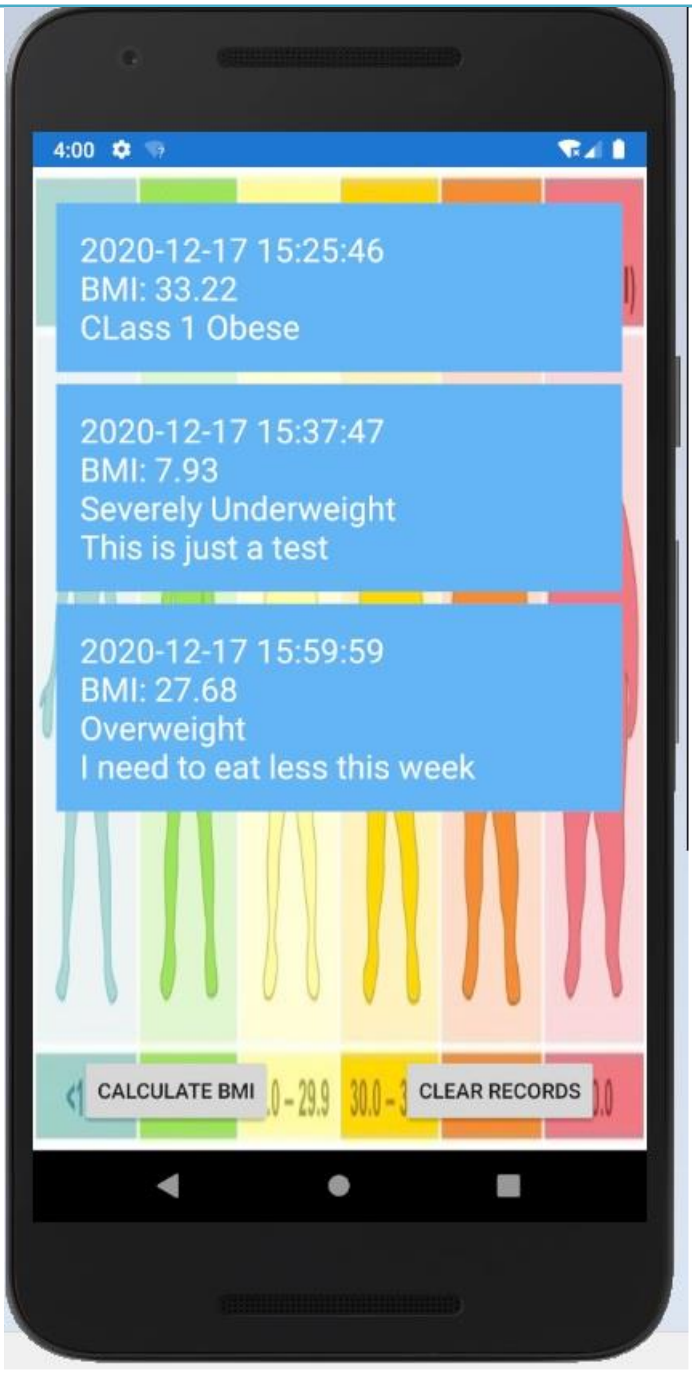
Number	Screenshots	Descriptions
1	 3:49 2020-12-17 15:25:46 BMI: 33.22 CLass 1 Obese 2020-12-17 15:37:47 BMI: 7.93 Severely Underweight This is just a test 2020-12-17 15:38:08 BMI: 30.8 CLass 1 Obese	This is the main interface of the App. It starts by displaying the previous records of the user's BMI calculation. From there the user can choose to calculate their current BMI and add the result to the list of records. The records include the time of the calculation, the BMI result and an optional note. From there the user can click the CALCULATE BMI button to calculate their BMI and add the result to the records. They can also click the CLEAR RECORDS button to clear the database and delete all the records or swipe a record to delete that specific record.

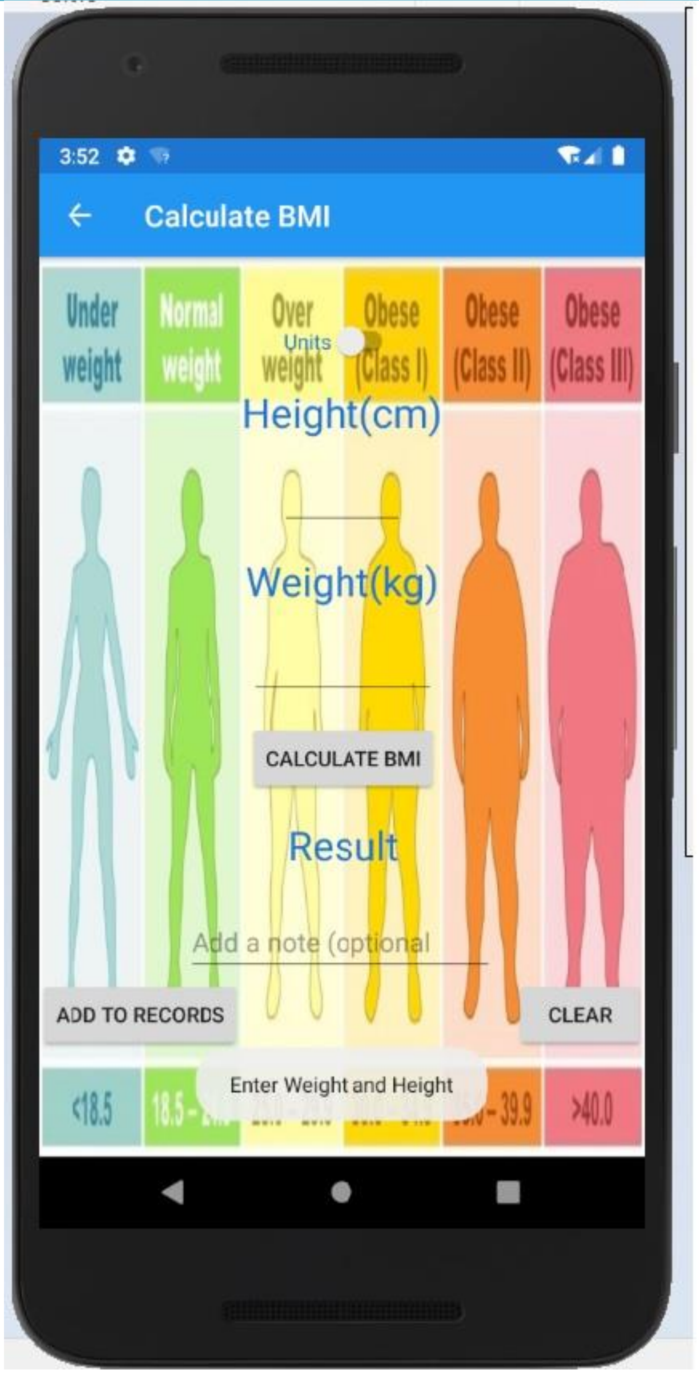
2	 3:51 Calculate BMI Under weight Normal weight Over weight Obese (Class I) Obese (Class II) Obese (Class III) Units Height(cm) Weight(kg) CALCULATE BMI Result Add a note (optional ADD TO RECORDS CLEAR <18.5 18.5 - 24.9 25.0 - 29.9 30.0 - 34.9 35.0 - 39.9 >40.0	This is the calculate BMI interface, there the user is presented with an attractive interface that allows them to choose the units to calculate their BMI and also two edit-text controls to enter their weight and height. In addition to the two inputs needed to calculate the BMI the user is also provided with an edit-text control to input a note such as a reminder. In this interface there are three buttons: one to calculate the BMI, one to clear the edit-texts and one to add to records. Therefore the user has the option to add the current BMI result to the records if they wish.

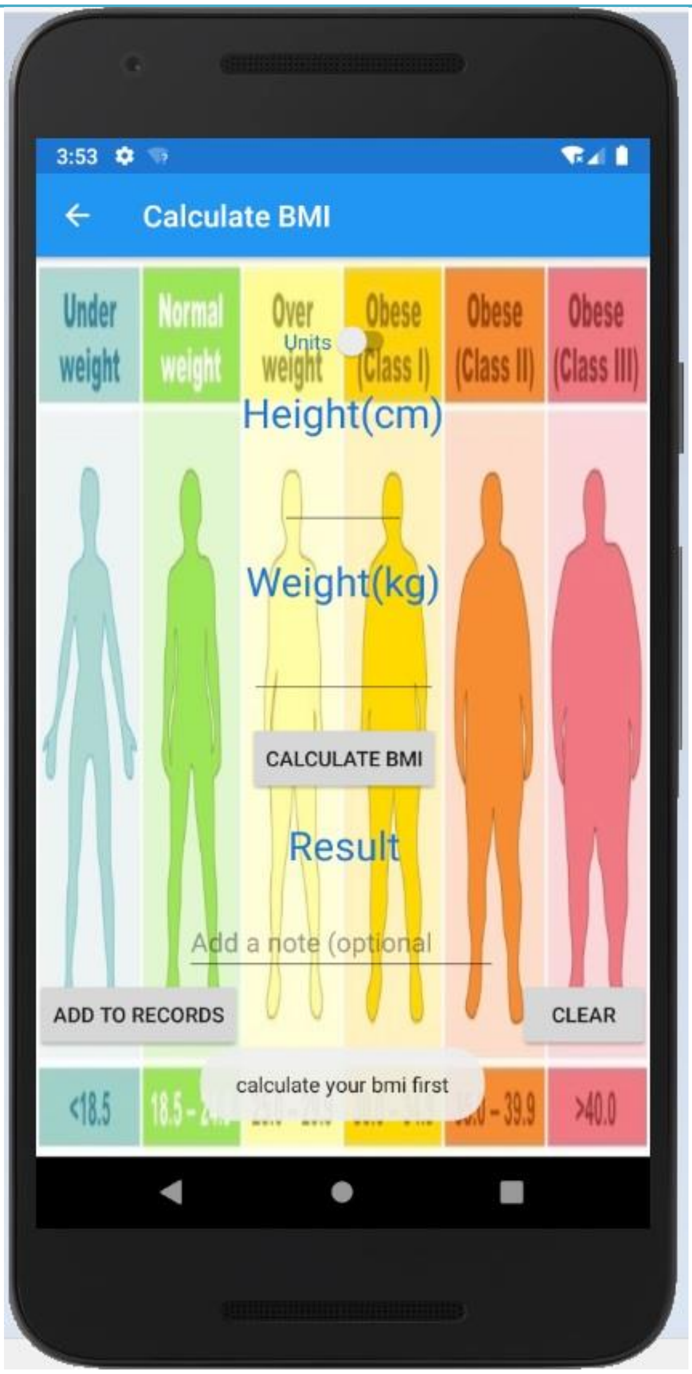
3	 3:54 Calculate BMI Under weight Normal weight Over weight Obese (Class I) Obese (Class II) Obese (Class III) Units Height(in) Weight(lbs) CALCULATE BMI Result Add a note (optional ADD TO RECORDS CLEAR <18.5 18.5 - 24.9 25.0 - 29.9 30.0 - 34.9 35.0 - 39.9 >40.0	This is the calculate BMI activity. In this interface the user can enter the required inputs which are his or her height and weight. Not the user has options to choose either metric units or imperial units using the switch widget at the top of the interface. The method source code for the switch unit function is explained in Appendix 1.

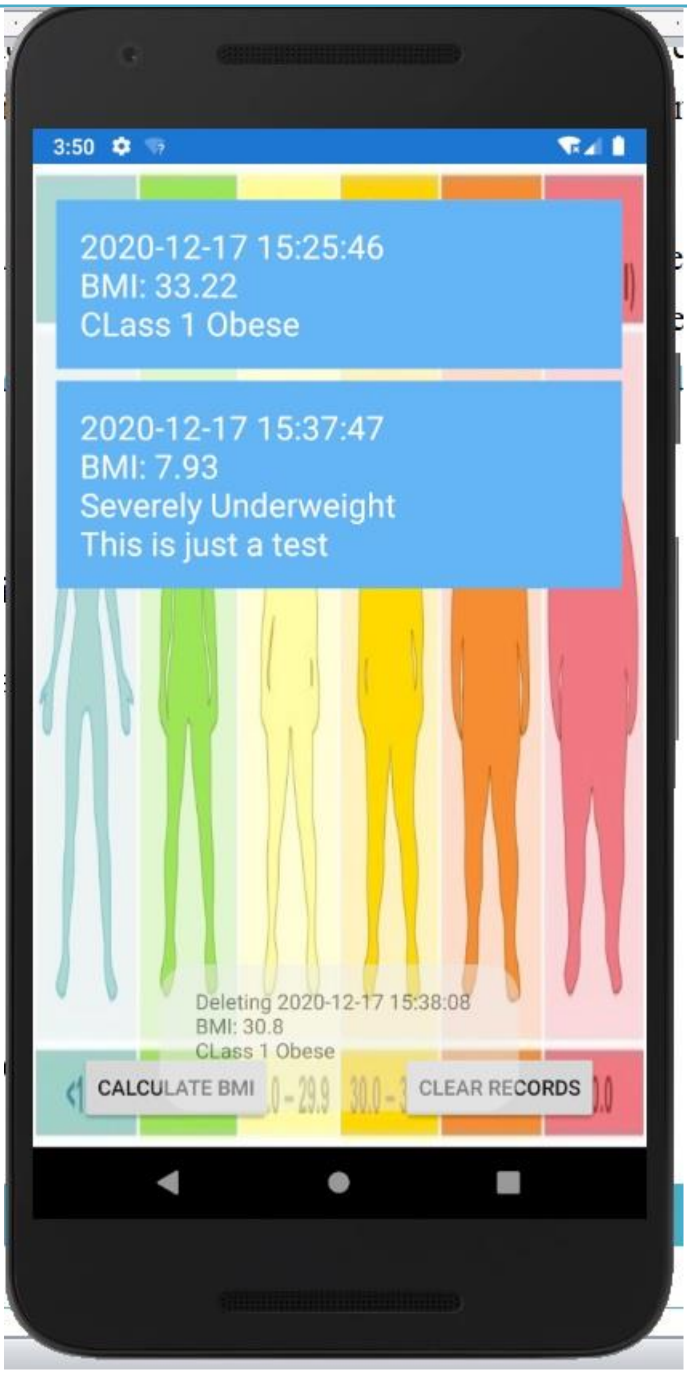
4	 3:57 Calculate BMI Under weight Normal weight Over weight Obese (Class I) Obese (Class II) Obese (Class III) Units Height(cm) 170 Weight(kg) 80 CALCULATE BMI	This is an example of the user clicking the switch control to change the input units. As previously mentioned the user can choose between metric units and imperial (US) units to calculate their BMI. After entering the required inputs (i.e. weight and height) the user clicks the CALCULATE BMI button to calculate the BMI result which is then displayed in the RESULT text-view. Note that input control was used in order to prevent the user from entering incorrect inputs. The user can only enter numbers in these two edit-texts.

5	 3:58 170 Weight(kg) 80 CALCULATE BMI BMI: 27.68 Overweight eed to eat less this week	This is the Calculate BMI interface after the user has clicked the CALCULATE BMI button. We can observe that the background image has changed to match the BMI result showed. From there the user can add an optional Message and then click the ADD To RECORDS button to save the result in the database. The method source code for the calculate BMI function is explained in Appendix 2.

5	 2020-12-17 15:25:46 BMI: 33.22 CLass 1 Obese 2020-12-17 15:37:47 BMI: 7.93 Severely Underweight This is just a test 2020-12-17 15:59:59 BMI: 27.68 Overweight I need to eat less this week 	This is the main interface with a new record added with the optional message reminder. Note the record includes the time, BMI result and the body condition associated with the BMI result.

6	 3:52 Calculate BMI Under weight Normal weight Over weight Obese (Class I) Obese (Class II) Obese (Class III) Units Height(cm) Weight(kg) CALCULATE BMI Result Add a note (optional ADD TO RECORDS CLEAR <18.5 >40.0 Enter Weight and Height	When the user clicks the CALCULATE BMI button without first entering the required inputs the App prevents this action and displays a toast message to remind the user to enter the height and weight.

	 3:53 Calculate BMI Under weight Normal weight Over weight Obese (Class I) Obese (Class II) Obese (Class III) Units Height(cm) Weight(kg) CALCULATE BMI Result Add a note (optional ADD TO RECORDS CLEAR <18.5 >40.0 calculate your bmi first	When the user clicks the ADD TO RECORD button without first calculating a record the app notifies the user with a toast message.

7	 2020-12-17 15:25:46 BMI: 33.22 CLass 1 Obese 2020-12-17 15:37:47 BMI: 7.93 Severely Underweight This is just a test Deleting 2020-12-17 15:38:08 BMI: 30.8 CLass 1 Obese 	Deleting a record: the user just has to swipe a record to delete it, when a record is deleted a toast message is displayed.

Conclusion

After completing this project I found an interest in mobile application development since this was my first time developing a mobile application solution. Android studio is a powerful development tool with many useful features which application development convenient.

This application is basic in terms of functionality and there are many more features that can be added to make it more useful and user friendly such as a fitness tracker to keep track of the user's daily activity in the form of steps taken. The App includes a lot of the material that we learned during the course such as: input selection, layout design, Intents and working with the different control widgets in android studio. Despite the many shortcomings on my final project which is mainly due to lack of expertise in the java programming language and the android development platform, I Learned a lot about android and mobile application development.

Additional features that can be added include: a user profile and options to place advertisements in the app as a means of earning an income. In the future I would like to further develop the app and also learn more about android application development.

References

1. https://developer.android.com/codelabs/android-training-livedata-viewmodel#0
2. https://developer.android.com/codelabs/android-training-room-delete-data#0
3. https://stackabuse.com/how-to-format-a-string-in-java-with-examples/
4. https://www.pluralsight.com/guides/making-a-notes-app-using-room-database
5. https://uniqueandrocode.com/android-room-database-with-recyclerview/#Android_Room_Database_With_Recyclerview
6. https://developer.android.com/studio
7. https://medium.com/mindorks/using-room-database-android-jetpack-675a89a0e942

Appendix 1

When the user clicks the Switch control the `Setdisp(View view)` on-click method is called. This method then changed the united displayed in the interface.

NOTE: both a combination of string values and String constants are used in this method.

```
public void Setdisp(View view) {
    if (switch1.isChecked())
    {
        txtWeight.setText(R.string.dispweight);
        txtHeight.setText(R.string.dispheight);
    }
    else {
        txtWeight.setText("Weight(kg)");
        txtHeight.setText("Height(cm)");

    }
}
```

Appendix 2

When the user clicks the CALCULATE BMI button the on -click method `CalculateBMI(View view)` is invoked.

```
public void CalculateBMI(View view) {

    if (switch1.isChecked())
    {
        impcal();
    }
    else {
        metcal();
    }
}
```

Based on the checked state of switch 1 one of the two methods are called.

The `impcal()` method calculated the BI result in imperial(US) units.

```
public void impcal()// the impcal method used to calculate the BMI in
english units and display the results.
{try {
    String S1 = weight.getText().toString();
    String S2 = height.getText().toString();
    // String S2 = height.getText().toString(); //declare string variables
    float wightvalue, heightvalue; // declare float variables used in the
arithmetic calculations
    wightvalue = Float.parseFloat(S1);
    heightvalue = Float.parseFloat(S2);
    float bmi = wightvalue / (heightvalue * heightvalue) * 703; // the BMI
formula used to calculate the BMi

    if (bmi < 16) {
        BMIresult = "Severely Underweight";
        layout.setBackgroundResource(R.drawable.underweight);
    } else if (bmi < 18.5) {
        BMIresult = "Underweight";
        layout.setBackgroundResource(R.drawable.underweight);

    } else if (bmi >= 18.5 && bmi <= 24.9) {
        BMIresult = "Normal weight";
        layout.setBackgroundResource(R.drawable.normalweight);
    } else if (bmi >= 25 && bmi <= 29.9) {
        BMIresult = "Overweight";
        layout.setBackgroundResource(R.drawable.overweight);
    } else if (bmi >= 30 && bmi <= 34.9) {
        BMIresult = "CLass 1 Obese";
        layout.setBackgroundResource(R.drawable.obese1);
    } else if (bmi >= 35 && bmi <= 39.9) {
        BMIresult = "Class 2 Obese";
        layout.setBackgroundResource(R.drawable.obese2);
    } else {
        BMIresult = "Class 3 Obese";
        layout.setBackgroundResource(R.drawable.obese3);
    }

    bmi = Math.round(bmi * 100.0f) / 100.0f;// round off the BMI value to
2 decimal places
    calculation = "BMI: " + bmi + "\n" + BMIresult;
    resulted.setText(calculation);
}
catch (NumberFormatException e)
{
    showmsg();
}

}
```

The `metcal()` method calculated the BI result in metric (SI) units.

```
public void metcal()// the metcal method used to calculate the BMI in SI
units and display the results.
{ try {
    String S1 = weight.getText().toString();
    String S2 = height.getText().toString();
    // String S2 = height.getText().toString(); //declare string variables
    float wightvalue, heightvalue; // declare float variables used in the
arithmetic calculations
    wightvalue = Float.parseFloat(S1);
    heightvalue = Float.parseFloat(S2) / 100;
    float bmi = wightvalue / (heightvalue * heightvalue); // the BMI
formula used to calculate the BMi

    if (bmi < 16) {
        BMIresult = "Severely Underweight";
        layout.setBackgroundResource(R.drawable.underweight);
    } else if (bmi < 18.5) {
        BMIresult = "Underweight";
        layout.setBackgroundResource(R.drawable.underweight);

    } else if (bmi >= 18.5 && bmi <= 24.9) {
        BMIresult = "Normal weight";
        layout.setBackgroundResource(R.drawable.normalweight);
    } else if (bmi >= 25 && bmi <= 29.9) {
        BMIresult = "Overweight";
        layout.setBackgroundResource(R.drawable.overweight);
    } else if (bmi >= 30 && bmi <= 34.9) {
        BMIresult = "CLass 1 Obese";
        layout.setBackgroundResource(R.drawable.obese1);
    } else if (bmi >= 35 && bmi <= 39.9) {
        BMIresult = "Class 2 Obese";
        layout.setBackgroundResource(R.drawable.obese2);
    } else {
        BMIresult = "Class 3 Obese";
        layout.setBackgroundResource(R.drawable.obese3);
    }
    bmi = Math.round(bmi * 100.0f) / 100.0f;// round off the BMI value to
2 decimal places
    calculation = "BMI: " + bmi + "\n" + BMIresult;
    resulted.setText(calculation);
}
catch (NumberFormatException e)
{
    showmsg();
}
}
```

Both these methods are responsible for calculating the BMI result and changing the interface background to match the BMI result.

Additionally exception handling was implemented using a try catch structure which ensures that the user enters the required inputs.