Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

Name: **Ryan Weafer**
Student ID: **X12513017**
Supervisor: **Cristina Hava Muntean**

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature:_____*Ryan Weafer*_____ Date:___27/7/17_____

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,

What constitutes plagiarism or cheating?

The following is extracted from the college's formal statement on plagiarism as quoted in the Student Handbooks. References to "assignments" should be taken to include any piece of work submitted for assessment.

Paraphrasing refers to taking the ideas, words or work of another, putting it into your own words and crediting the source. This is acceptable academic practice provided you ensure that credit is given to the author. Plagiarism refers to copying the ideas and work of another and misrepresenting it as your own. This is completely unacceptable and is prohibited in all academic institutions. It is a serious offence and may result in a fail grade and/or disciplinary action. All sources that you use in your writing must be acknowledged and included in the reference or bibliography section. If a particular piece of writing proves difficult to paraphrase, or you want to include it in its original form, it must be enclosed in quotation marks

and credit given to the author.

When referring to the work of another author within the text of your project you must give the author's surname and the date the work was published. Full details for each source must then be given in the bibliography at the end of the project

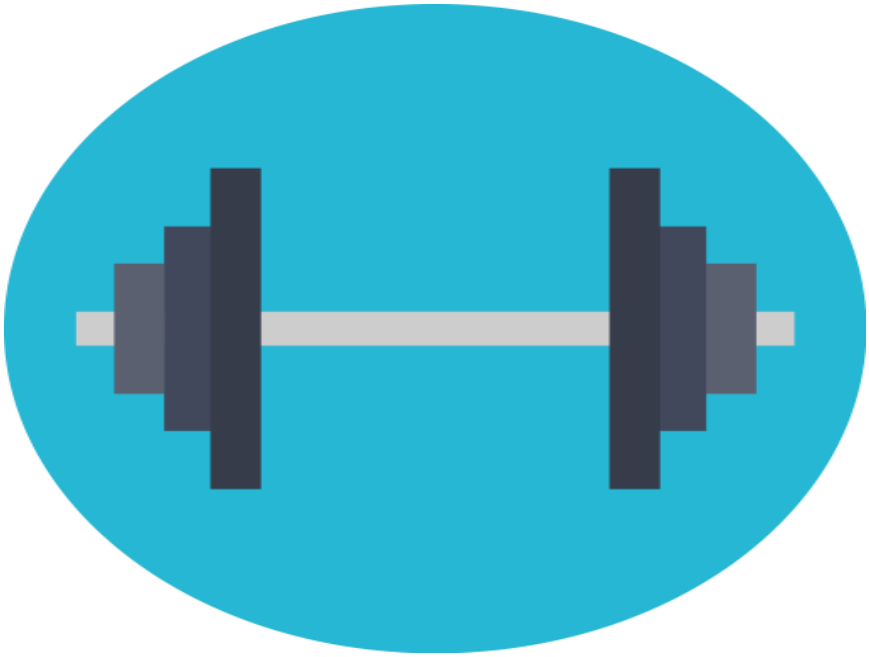
Penalties for Plagiarism

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Where the Disciplinary Committee makes a finding that there has been plagiarism, the Disciplinary Committee may recommend

- that a student's marks shall be reduced
- that the student be deemed not to have passed the assignment
- that other forms of assessment undertaken in that academic year by the same student be declared void
- that other examinations sat by the same student at the same sitting be declared void

Further penalties are also possible including

- suspending a student college for a specified time,
- expelling a student from college,
- prohibiting a student from sitting any examination or assessment.,
- the imposition of a fine and
- the requirement that a student to attend additional or other lectures or courses or undertake additional academic work.



Fitness App

Ryan Weafer

Supervisor: Cristina Hava Muntean

Final Project Report

X12513017

BSHC4MAD

Contents

Executive Summary

Over recent years the world has seen a spike in the download and usage of fitness and health apps. In 2014 fitness app usage grew at a substantial rate, being up there as the most used category of application for that year. Since then it has maintained its user base and continues to grow, with the inclusion of wearables like google fit, Fitbit and Healthkit. This is the dawn of a new era, an era where people look more to their mobiles or their fitness watches to check on their health, rather than the traditional method of going and seeing a doctor. These apps provide a great avenue for those who are interested on tracking their fitness levels runners, cyclists, and gym goers alike. Everything can be tracked nowadays, even the standard iPhone comes with a health app built in, with a range of features.

There are several highly successful fitness apps on the market, some of these include mapmyrun, mapmyride, and the FitBit app. Both mapmyrun and mapmyride are made by the same group and are simple route trackers for running and cycling. Once your journey is finished the app then gives detailed statistics of the route and calories burned. The FitBit app is one that I use, it works in tandem with a wearable FitBit watch, which tracks your steps and monitors your heart rate.

But all of these apps although state of the art, have one thing in common, they are jam packed with irrelevant features which take away from the initial idea and distract the user. I for one when working out want a hassle-free interaction with an application, I want to get in there and get it done.

Background

After using a variety of fitness apps on a day to day basis I began to become frustrated with all of the clutter and irrelevant features included with them. Some of these apps are too complex and can take away from the apps main purpose. Simplicity could go a long way in a health/fitness application. This is one of the reasons I believed building a simpler fitness app would be a good idea.

I am an active individual who takes part in sports as well as attending the gym regularly. There were things I noticed myself doing on a day to day basis which I felt could be included in an app. For example, before going to the gym every day, I would sit in the car park and jot down what workout I was going to do in the notes app on my iPhone. Of course, there are many complex and extravagant apps out there which cater for this but come with all that baggage. Hence why I opted to use the most basic app there is.

There are so many complex and extravagant apps out there for almost every aspect of health/fitness and to be honest there really isn't much that hasn't already been done in this market. But I had the idea that I wanted to create an app that would implement each of these aspects simply, so as to be hassle free and not take away from the goal which the user set out in the first place.

Aims

The purpose of my application is to develop an application that is valuable to gym goers and people who exercise in general who would like to track their workouts and accomplish their fitness goals.

The graphical user interface of the app should look appealing to the user so as to entice them. The app should provide a pleasant experience and a provide a feeling of accomplishment after being used to encourage recurrent usage. It should be highly accessible regardless of the user's familiarity with applications. Whether the user is a novice or is experienced, the app will be good for both.

The key to this app is simplicity and this app will provide a few features popular in this market, through a simple and straight to the point application. The app should also provide the user with a fun experience. The popular features I have aimed to include in this application are;

A step counter -which has been made highly popular through the FitBit watches. A step counter can give an idea of how far someone has travelled and how many calories they have burned. This would be associated more frequently with running.

A stopwatch – A simple stopwatch for interval training when working out.

A workout planner -A simple workout planner which allows you to add and delete workout routines.

A BMI calculator – A BMI calculator to give the user an idea of their current health/fitness level.

A tracker – A geolocation tracker build using google maps to track a user's run.

These are the five core features I set out to create in order to make a no-nonsense fitness app.

Technologies

In order to complete this project, I have utilised android studio as it is an application that I have become familiar with during my coursework. During my coursework, I have used android studio in the creation of numerous projects, and thus, have used this knowledge to create the classes and GUI elements of this one. I also used two different android devices with different operating systems, one being a Samsung Galaxy s4 on Android 4.2.2 Jelly Bean and the other being an Acer iconia one 8 on Android 5.0 Lollipop. These would prove useful during the testing of the application.

Google Firebase was used for user authentication in the app. This was used to implement the login and registration sections. Choosing Firebase was an easy decision as it is easy to implement and provides a variety of authentication options through the Firebase console. The console makes it easy to manage users and includes an option for users to have a password reset sent to their email address.

SQLite database was used for the storage of workout plans. Through the class DbHelper which extends SQLiteOpenHelper, I was able to store the user's workouts in a table called 'Task'. This class also contains a method for the user to delete workouts.

Google maps API was used in the app for user geolocation and tracking. By creating a google map called mMap and through the use of LocationManager, 'ACCESS_FINE_LOCATION', and the onLocationChanged method, I was able to track the user's phone.

After implementing the hardware sensor stepcounter and hardware sensor stepdetector features in the AndroidManifest file, I was able to code a step counting activity into the app. This was done using a sensor event listener.

System

This section will concentrate on a number of areas. I will provide a detailed description of the various types of requirements for the app, the design and architecture of the system, the technologies used and implementation of the system, the graphical user interface and finally the testing methods. I will try to go into as much detail as possible whilst describing each of these topics.

Requirements

The requirements specification part of any project is a very important one. It allows us to gather information on what an audience actually wants from the app itself. This can give us an insight into the perspective of what would be required from the app in order for it to be successful.

In order to gather my requirements for this project, I asked three friends, one of whom is a personal trainer, whilst the other two frequent the gym what they would like to get from a fitness app. Between these three individuals they use a variety of different apps for fitness so they are familiar with what they like and don't like. Also between them, they use a mixture of both iPhones and android devices. This gave me a solid mixture of requirements that users might come to expect.

Functional Requirements

- Register: The app will have a registration feature where the user will be required to enter a valid email address and password. These details will then be saved with google firebase.
- Login: Users will be required to log in to use the application. They must enter their email address and

password. These details are then authenticated against the database and the user is directed to the homepage.

- Step Counter: The Step counter will allow the user to track how many steps they have taken and get an estimate of how many calories they have burned.
- BMI calculator: The BMI calculator allows users to calculate their body mass index by simply entering their height and weight.
- Interval timer: The interval timer allows users to time themselves and perform interval training in the gym or at home.
- Workout Planner: The workout planner allows users to create a workout plan so that they are can remember what they intend to do.
- Tracker: The tracker allows users to track their location and see how far they have travelled.

Data Requirements

In this section, I will give a detailed account of the data requirements essential in implementing the above features.

- SQLite: The app will make use of SQLiteOpenHelper feature in order to store user's workout plans. These plans will be saved to a db named "workout". The workout plan class is done in the form similar to notes

where users will also be able to delete from the database.

- Google Firebase: The app will also make use of google firebase to host user credentials. The firebase authentication functionality is used for the login and registration methods.

User Requirements

This section will describe the user requirements needed so that they will be able to use the application efficiently.

- Android phone/tablet: The user must possess an android phone or tablet as the app is not compatible with other devices.
- API level: The minimum sdk level for the app is level 15 and the target sdk is 25.
- Hardware: For the step counter function to work the stepcounter hardware sensor must be available on the target device.
- Internet access: The user will also need internet access to sign in/register and to use the google maps tracking feature.

Environmental Requirements

- Android phone/tablet: Required to run the application during development and to test.
- Internet: Used to access various resources for the project.
- Computer/laptop: Required to work with android studio to develop the app.
- Firebase Console: used to monitor activity of user login and crash reporting.
- Google maps API: API key used in project to develop geolocation and tracking.

Usability Requirements

- Simple: The app will be simple and easy to navigate, yet so simple as to cause users to lose interest. Whilst the app will be easy to use, it will also have a good balance for novice and experienced users alike, so as to appeal to levels.
- Engaging GUI: The layout should be simple and make use of a concurrent theme with a good colour scheme, in order to grab the attention of users.
- Response times/operability: Error messages should explain issues as they occur. The app won't leave users on a loading screen wondering when, if ever something is going to happen, as this can lead to them not returning to use the app. Toast messages make up the

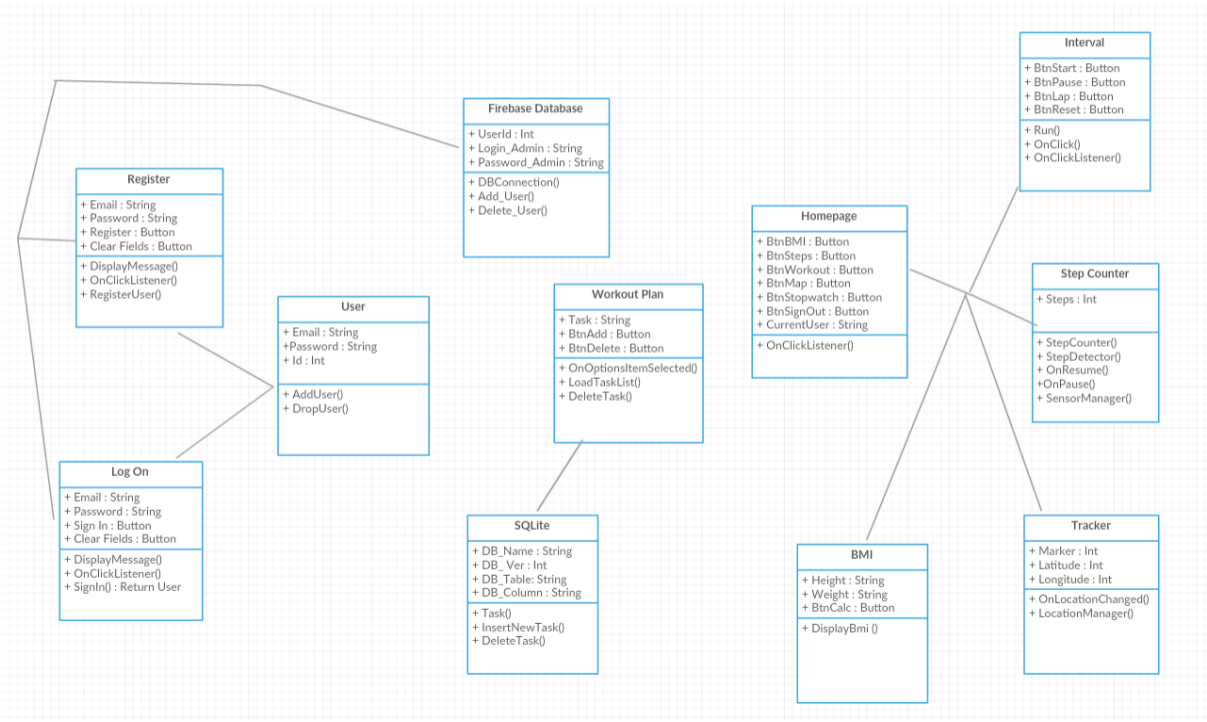
basis of this. The apps response time should be instantaneous so that user does not lose interest.

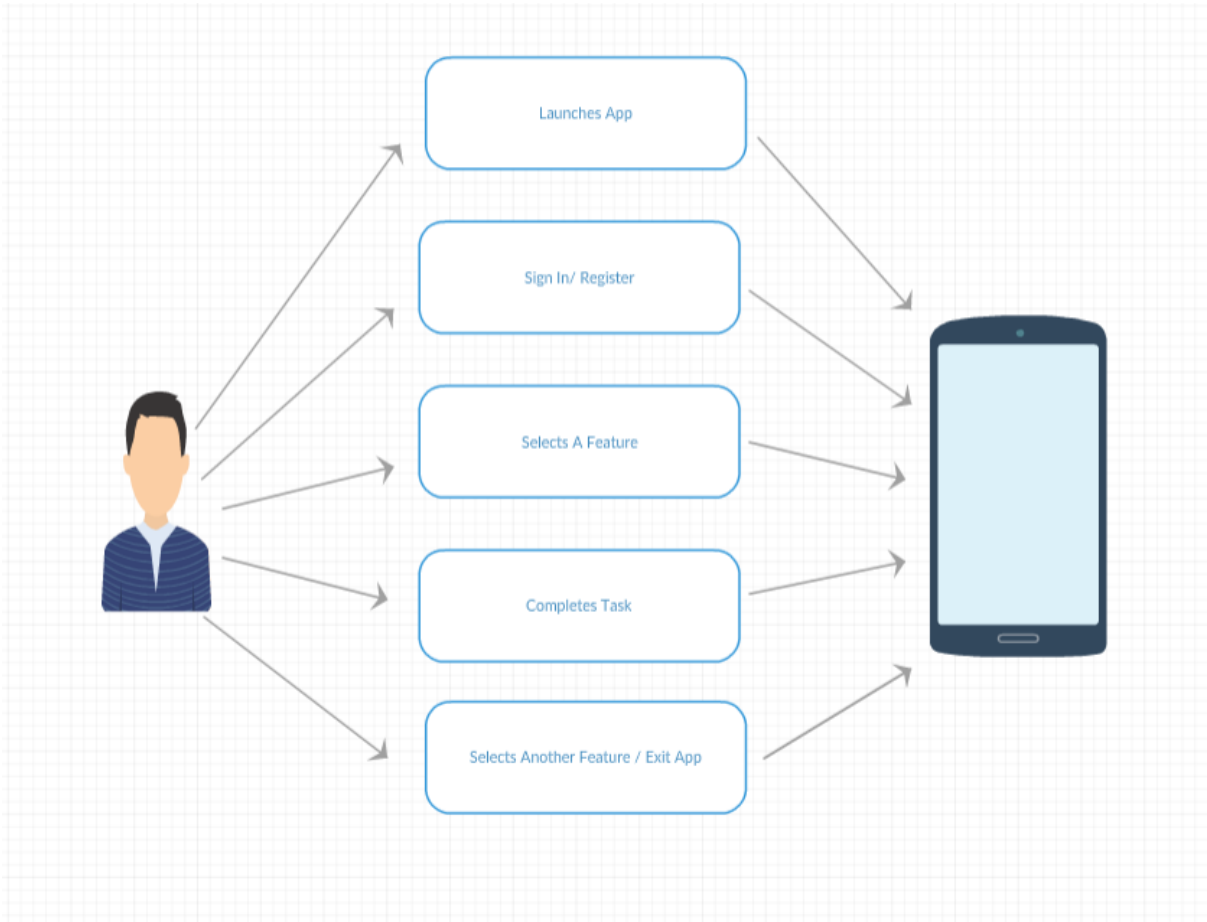
- Well-structured GUI: Sometimes apps can pack too much information into our phones tiny screens. This is definitely something the app will attempt to avoid by being well structured and only having a small portion of information visible at any time.

Design and architecture

The app was developed to be based solely around the user's health and fitness, so all of the features in it are based around this. The app was built in android studio and each page of it was designed following the same theme. Each feature has an individual class which can be accessed from the homepage. Through customizing my colors.xml file I was able to create a colour scheme which is present throughout every page of the app. Using different layouts and layout components I was able to design each page as I saw fit.

I developed an ad hoc model to study interactions and different concepts during the design process. I looked at all of the different components and the relationships between them which ultimately laid the foundation for the overall design of my application. The apps design allows the user to traverse through the various pages easily.


Interval
+ BtnStart : Button
+ BtnPause : Button
+ BtnLap : Button
+ BtnReset : Button
+ Run()
+ OnClick()
+ OnClickListener()
Firebase Database
+ UserId : Int
+ Login_Admin : String
+ Password_Admin : String
+ DBConnection()
+ Add_User()
+ Delete_User()
Register
+ Email : String
+ Password : String
+ Register : Button
+ Clear Fields : Button
+ DisplayMessage()
+ OnClickListener()
+ RegisterUser()
Homepage
+ BtnBMI : Button
+ BtnSteps : Button
+ BtnWorkout : Button
+ BtnMap : Button
+ BtnStopwatch : Button
+ BtnSignOut : Button
+ CurrentUser : String
+ OnClickListener()
Step Counter
+ Steps : Int
+ StepCounter()
+ StepDetector()
+ OnResume()
+OnPause()
+ SensorManager()
User
+ Email : String
+Password : String
+ Id : Int
+ AddUser()
+ DropUser()
Workout Plan
+ Task : String
+ BtnAdd : Button
+ BtnDelete : Button
+ OnOptionsItemSelected()
+ LoadTaskList()
+ DeleteTask()
Log On
+ Email : String
+ Password : String
+ Sign In : Button
+ Clear Fields : Button
+ DisplayMessage()
+ OnClickListener()
+ SignIn() : Return User
SQLite
+ DB_Name : String
+ DB_Ver : Int
+ DB_Table: String
+ DB_Column : String
+ Task()
+ InsertNewTask()
+ DeleteTask()
BMI
+ Height : String
+ Weight : String
+ BtnCalc : Button
+ DisplayBmi ()
Tracker
+ Marker : Int
+ Lattitude : Int
+ Longitude : Int
+ OnLocationChanged()
+ LocationManager()


Launches App
Sign In/ Register
Selects A Feature
Completes Task
Selects Another Feature / Exit App

Implementation

The purpose of this section is to cover the technologies that were used in the development of this application.

Android Studio

The tool I used to create this application from start to finish was Android Studio. The version of I used in making this was 2.2.2, compileSdkVersion “25”, buildToolsVersion “25.0.2”.

Step Counter

The step counter class of the app requires the target device to have hardware sensors for both step counter and step detector present, without these the feature would cease to work. This class implements SensorEventListener so that the device recognizes when the user is moving. When running = true the onSensorChanged method updates the tv_steps TextView so that it changes its output with each step.

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

    tv_steps = (TextView) findViewById(R.id.tv_steps);

    sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);


}

@Override
protected void onResume(){
    super.onResume();
    running = true;
    Sensor countSensor = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER);
    if(countSensor != null){
        sensorManager.registerListener(this, countSensor, SensorManager.SENSOR_DELAY_UI);
    } else {
        Toast.makeText(this, "Sensor not found!!", Toast.LENGTH_SHORT).show();
    }
}

@Override
protected void onPause() {
    super.onPause();
    running = false;
    sensorManager.unregisterListener(this);
}

@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    if(running){
        tv_steps.setText(String.valueOf(sensorEvent.values[0]));
    }

}
```

Stopwatch

The stopwatch activity allows users to do interval training. It is done in milliseconds and has a start, pause, lap, and reset button. The calculations for the timer in milliseconds was implemented in the run method. A handler, customHandler was included to handle communication with runnable updateTimerThread. Row.xml inflated in “container”, is where laps are made visible. This is a LinearLayout nested within a ScrollView.

```
public class Watch extends AppCompatActivity {

    Button btnStart, btnPause, btnLap, btnReset;
    TextView txtTimer;
    Handler customHandler = new Handler();
    LinearLayout container;

    long startTime=0L, timeInMilliseconds=0L, timeSwapBuff=0L, updateTime=0L;

    Runnable updateTimerThread = new Runnable() {
        @Override
        public void run() {
            timeInMilliseconds = SystemClock.uptimeMillis()-startTime;
            updateTime = timeSwapBuff+timeInMilliseconds;
            int secs=(int)(updateTime/1000);
            int mins=secs/60;
            secs%=60;
            int milliseconds=(int)(updateTime%1000);
            txtTimer.setText(""+mins+":"+String.format("%2d", secs)+":"
                                    +String.format("%3d", milliseconds));
            customHandler.postDelayed(this, 0);

        }
    };

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

        btnStart =(Button)findViewById(R.id.btnStart);
        btnPause =(Button)findViewById(R.id.btnPause);
        btnLap =(Button)findViewById(R.id.btnLap);
```

Workout

The workout class works in a manner similar to that of taking notes. It allows the user to create a workout plan simply. A plan can be added by clicking the add menu item in the top right. There is also a delete button if the user is not satisfied with the plan or does not wish to keep it any longer. The loadTaskList method is used here to communicate with the DbHelper class in order to populate information. The information is then passed to task_title in the workout_row.xml file which in turn fills the activity_workout

listView. Once the add function is triggered in the menu a new dialog interface opens where the user will enter their workout. Once the user hits save dbHelper.insertNewTask(task) pushes this data to the table. Then the data is called again using loadTaskList(). The same goes for the deleteTask method, dbHelper.deleteTask(task) removes this workout from the table and then the listView is updated.

```
private void loadTaskList() {
    ArrayList<String> taskList = dbHelper.getTaskList();
    if(mAdapter==null){
        mAdapter = new ArrayAdapter<String>(this, R.layout.workout_row, R.id.task_title, taskList);
        lstTask.setAdapter(mAdapter);
    } else {
        mAdapter.clear();
        mAdapter.addAll(taskList);
        mAdapter.notifyDataSetChanged();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu){...}

public boolean onOptionsItemSelected(MenuItem item){

    switch (item.getItemId()){
        case R.id.action_add_task:
            final EditText taskEditText = new EditText(this);
            AlertDialog dialog = new AlertDialog.Builder(this)
                    .setTitle("Add New Excercise")
                    .setMessage("What excercise would you like to do?")
                    .setView(taskEditText)
                    .setPositiveButton("Add", new DialogInterface.OnClickListener(){
                        @Override
                        public void onClick(DialogInterface dialog, int which){
                            String task = String.valueOf(taskEditText.getText());
                            dbHelper.insertNewTask(task);
                            loadTaskList();
                        }
                    })
                    .setNegativeButton("Cancel", null)
                    .create();
```

```
private static final int DB_VER= 1;
public static final String DB_TABLE="Task";
public static final String DB_COLUMN="TaskName";

public DbHelper(Context context) { super(context, DB_NAME, null, DB_VER); }

@Override
public void onCreate(SQLiteDatabase db) {...}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {...}

public void insertNewTask(String task){
    SQLiteDatabase db = this.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(DB_COLUMN, task);
    db.insertWithOnConflict(DB_TABLE, null, values, SQLiteDatabase.CONFLICT_REPLACE);
    db.close();
}

public void deleteTask(String task){
    SQLiteDatabase db = this.getWritableDatabase();
    db.delete(DB_TABLE, DB_COLUMN + " = ?", new String[]{task});
    db.close();
}

public ArrayList<String> getTaskList(){
    ArrayList<String> taskList = new ArrayList<>();
    SQLiteDatabase db = this.getReadableDatabase();
    Cursor cursor = db.query(DB_TABLE, new String[]{DB_COLUMN}, null, null, null, null, null);
    while(cursor.moveToNext()){
        int index = cursor.getColumnIndex(DB_COLUMN);
        taskList.add(cursor.getString(index));
    }
    cursor.close();
    db.close();
    return taskList;
}
```

Body Mass Calculator

The BMI class allows users to simply enter both their height in cm's and their weight in kg's and be given their body mass index value. There is also a comment along with that value to let the user know what it means. The calculation is done as follows, bmi = weightValue / (heightValue * heightValue). The displayBMI method will display a message depending on what value is calculated, so for example between 18.5 and 25 the user can be said to have a normal BMI.

```
    String heightStr = height.getText().toString();
    String weightStr = weight.getText().toString();

    if (heightStr != null && !"".equals(heightStr)
            && weightStr != null && !"".equals(weightStr)){
        float heightValue = Float.parseFloat(heightStr) / 100;
        float weightValue = Float.parseFloat(weightStr);

        float bmi = weightValue / (heightValue * heightValue);

        displayBMI(bmi);
    }
}

private void displayBMI(float bmi){
    String bmiLabel = "";

    if(Float.compare(bmi, 15f) <= 0){
        bmiLabel = "Very severely underweight";
    } else if (Float.compare(bmi, 15f) > 0 && Float.compare(bmi, 16f) <= 0){
        bmiLabel = "Severely underweight";
    } else if (Float.compare(bmi, 16f) > 0 && Float.compare(bmi, 18.5f) <= 0) {
        bmiLabel = "Underweight";
    } else if (Float.compare(bmi, 18.5f) > 0 && Float.compare(bmi, 25f) <= 0) {
        bmiLabel = "Normal";
    } else if (Float.compare(bmi, 25f) > 0 && Float.compare(bmi, 30f) <= 0) {
        bmiLabel = "Overweight";
    } else if (Float.compare(bmi, 30f) > 0 && Float.compare(bmi, 35f) <= 0) {
        bmiLabel = "Obese Class I (Moderately obese)";
    } else if (Float.compare(bmi, 35f) > 0 && Float.compare(bmi, 40f) <= 0) {
        bmiLabel = "Obese Class II (Severely obese)";
    } else {
        bmiLabel = "Obese Class III (Morbidly obese)";
    }

    bmiLabel = bmi + "\n\n" + bmiLabel;
    result.setText(bmiLabel);
```

Google maps geolocation

The MapsActivity class allows us to track the user. Firstly, permission must be requested to access user location. LocationManager.requestLocationUpdates allows us to keep track of the user through their network provider. If the network provider is not available a separate method has been written to request the location through gps provider. These methods get both the latitude and longitude of the user. Then the geocoder can add a marker to the map and

the camera is automatically moved to look down at this point.

```
if (ActivityCompat.checkSelfPermission(this, android.Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRAN
    // TODO: Consider calling
    //    ActivityCompat#requestPermissions
    // here to request the missing permissions, and then overriding
    //   public void onRequestPermissionsResult(int requestCode, String[] permissions,
    //                                          int[] grantResults)
    // to handle the case where the user grants the permission. See the documentation
    // for ActivityCompat#requestPermissions for more details.
    return;
}

//check whether the network provider is enabled
if(locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER)){
    locationManager.requestLocationUpdates(locationManager.NETWORK_PROVIDER, 0, 0, new LocationListener() {
        @Override
        public void onLocationChanged(Location location) {
            //get latitude
            double latitude = location.getLatitude();
            //get longitude
            double longitude = location.getLongitude();
            //instantiate class latlng
            LatLng latLng = new LatLng(latitude, longitude);

            points.add(latLng);

            //instantiate the class for geocoder
            Geocoder geocoder = new Geocoder(getApplicationContext());
            try {
                List<Address> addressList = geocoder.getFromLocation(latitude, longitude, 1);
                String str = addressList.get(0).getLocality()+",";
                str += addressList.get(0).getCountryName();
                mMap.addMarker(new MarkerOptions().position(latLng).title(str));
                mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
            } catch (IOException e) {
                e.printStackTrace();
            }
```

Firebase

For both my login and registration classes Firebase auth allows me authenticate users. For my login page Firebase saves input from both the email editText and password editText and runs it against the database. If successful the app starts an intent to the homepage of the app. The register class works in the same way saving the details from both editTexts to the database and sending a toast to say registration was successful, the user is then brought to the homepage.

```
        buttonSignIn.setOnClickListener(this);
        textViewRegistration.setOnClickListener(this);
    }

    private void userLogin(){
        String email = editTextEmail.getText().toString().trim();
        String password = editTextPassword.getText().toString().trim();

        if(TextUtils.isEmpty(email)){
            Toast.makeText(this, "Please enter email", Toast.LENGTH_SHORT).show();
            return;
        }

        if(TextUtils.isEmpty(password)){
            Toast.makeText(this, "Please enter password", Toast.LENGTH_SHORT).show();
            return;
        }

        progressDialog.setMessage("Logging In, Please wait...");
        progressDialog.show();

        firebaseAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, (task) → {
                        progressDialog.dismiss();

                        if(task.isSuccessful()){
                            finish();
                            startActivity(new Intent(getApplicationContext(), HomePage.class));

                        }
```

Testing

Unit Testing

Throughout the course of developing this application, after each new feature was added or changes were made I performed unit testing. Whenever I added something, I ran the app through an emulator on my laptop as well as on both of the android devices I have. I made use of both the android monitor and logcat when testing the app and this was key in

helping me identify and eradicate errors. The gradle console also helped with error detection. In regard to Firebase, any issues I encountered were logged via crash reporting and I was able to view the log via Firebase console.

Backwards Compatibility Testing

The app was developed on android sdk version 25 with the minimum sdk version set at 15. The app has been tested on a number of different versions and multiple devices. Most of the features can be used from low versions and upwards. The only concern I would have is if the user's device does not have the hardware required for the step counter to work.

Persona Testing

In order to test the application, I made two personas for people to test the app.

Bank Worker: Daniel is a twenty-four-year-old bank worker who holds a degree in culinary arts. He is a gym goer who attends the gym from 4-5 days a week. He is the owner of a Samsung galaxy s8. He thought the app was quite simple. In particular he liked the workout planner and interval

stopwatch. He didn't like the layout of the homepage saying, "he couldn't put his finger on it, maybe it's a bit cluttered".

Personal Trainer: Adam is a twenty-four-year-old personal trainer with a degree in sports science. He is the owner of an iPhone 6. He thought the app was quite good and was particularly fond of the interval stopwatch and BMI calculator.

Scenario Testing

Scenario 1

- The user was tasked with opening the app.
- They will then have to login with pre-made log in details to use it.
- Once registered they are tasked with opening the tracker.
- Then navigate to the workout planner and create a new workout.
- Once finished they must navigate to the interval timer and start the stopwatch creating laps and then resetting the timer.
- When finished the user will sign out of the app.

User

Daniel, novice user

Task analysis

The user was able to log into the app easily. He got a bad impression when he reached the homepage. From their it was rather straightforward for him to navigate to the tracker section. When on the tracker page he was not quite sure what its purpose was. He moved onto the workout planner which he seemed to like. He set about his task and left positive feedback about how handy it was. Moving to the Interval timer he set about his task which he completed easily. He then proceeded to log out. The most positive feedback he left was “this is handy I would have had to go into two separate apps to do both of these”, referring to the workout planner and interval timer. The negative feedback her left, “he couldn’t put his finger on it, maybe it’s a bit cluttered”, referring to the app’s homepage.

Scenario 2

- The user was tasked with opening the app.
- The user will then log in with pre-made details.
- Once logged in they are tasked with navigating to the BMI page where they will enter height and weight and calculate their BMI.

- Once they have received their result they must go to the step counter where they will proceed to run on the spot for 20 seconds.
- Then they will go to the interval page where they will start the stopwatch, make laps, pause, and reset it.
- Then they will proceed to log out.

User

Adam, experienced user

Task Analysis

The user was able to log in easily and was impressed with the login screen. Once he reached the BMI page he seemed happy with the result of his calculation and he left positive feedback. He then proceeded to the step counter and fulfilled his task and looked a bit surprised that the device had in fact recorded his steps. Upon completing that he moved to the interval timer and completed his task easily. He said interval training was something he did regularly and seemed happy with its inclusion. He proceeded to sign out. Adam left some positive feedback saying, “I like the fact that there was a BMI calculator and an interval timer”. He also suggested I include a something in regard to daily calorie intake as this would be useful. He pointed me towards a chart.

Evaluation

Upon completing the testing of this app, I really had my eyes opened at how important testing is in terms of delivering a product. We have our own biased ideas towards a project and testing allows us to view it through other people's eyes highlighting problems you probably wouldn't have noticed which ultimately could have a massive impact on whether someone will come back and use the app again. I took both Adam and Daniels advice on board and was made changes where necessary. After Daniels comments on homepage I decided to remove some text and re arrange the buttons so that it looked less full. After Adams suggestion regarding daily calorie intake I was able to make a minor change an include a page with some information on the average intake by age, gender, and activity level. Perhaps other people might have felt the same as they did and lost interest.

Conclusion

This report gives an insight into the background as to why I chose to develop this application. I believed that there was a problem with apps of this type and that if done a certain way this could be solved. The idea and my motivations have been stated clearly. The requirements are described in detail and the technologies used have also been outlined in this

document. I have touched on the Systems design and architecture and provided a class diagram to explain it further.

I believe that I achieved most of what I have set out to do when this idea was conceived. However, there are multiple different areas which could be improved and lots of room for expansion.

Future Prospects

In the future, this application could be greatly improved and expanded to include new features. The tracker and step counter can be improved. A calorie counter could be added to allow users keep track of their daily intake and pursue their weight loss goals. This would mean populating a database with vast amounts of food and nutritional data and allowing the user to enter food eaten after every meal. Once entered the app takes the number of calories from their daily allowance. An exercise instruction manual which advises users of exercises and how to do them.