

National College of Ireland

<BSHC Computing Final Year>

<Software Development>

<Academic Year i.e. 2021/2022>

<Bartosz Wojdas>

<x19128118>

x19128118@student.ncirl.ie

[GitHub Link to Project](#)

[Bart1999/BWFitness99 \(github.com\)](https://github.com/Bart1999/BWFitness99)

<BW Fitness>

Technical Report

Contents

1

1.0 Introduction2

1.1. Background2

1.2. Aims2

1.3. Technology2

2.0 System2

2.1.	Non Functional Requirements	2
2.1.1.1.	Use Case Diagram	3
2.1.1.2.	Functional Requirements	7
2.1.1.3.	Description & Priority	7
2.1.1.4.	Use Case Time	8
2.2.	Design & Architecture	8
2.3.	Implementation	Error! Bookmark not defined.
2.4.	Graphical User Interface (GUI)	23
2.5.	Testing	46
2.6.	Evaluation	59
3.0	Conclusions	60
4.0	Further Development or Research	61
5.0	References	61
6.0	Appendices	63
6.1.	Project Proposal	63
6.2.	Reflective Journals	66
6.3.	Invention Disclosure Form (Remove if not completed)	72

Executive Summary

In this report I will outline that I am doing a gym and fitness application on Android Studio, the technologies used will be mentioned in the report. With my invention I aim to make fitness lives of people of all kinds and ages easier. The main reason I have picked this project is because of my passion for gym and fitness as well as the growing popularity of fitness with the help of social media. This report will contain use cases to show the functionality of the different parts of my app. The use cases I will make will be for registration, login, gym slot booking, PT booking, calorie tracking, payment for membership/PT use case and a use case for settings. Each use case will have a scope and description. The design of each page will be described first in words then images (GUIs) Graphical User Interface screenshots will be put in as prototypes for my app to show how it will roughly look like for all the different pages I plan on implementing. After these steps I will conclude my work and describe what is an advantage & disadvantage and the challenges to implement all of these different features. I will also put in information for future projects and research from my experience with creating the technical report for this project. All final details will be put into the appendices section.

1.0 Introduction

1.1. Background

I am choosing to undertake this project as I am passionate about the gym, fitness and a healthy lifestyle. I want to help people of all kinds and all stages in their fitness journey by letting them track their workouts and calorie intake and output in order to see how they are comparing to the goals they set out. The fitness industry is constantly growing as more people want to get in shape with the constant influence of social media which links to IT.

1.2. Aims

With this project I am to help all kinds of people reach their fitness goals with the creation of this app. It will be the most helpful for the people starting out as they will need the most help whether it's for exercise help, workout splits and most importantly of all their calorie intake and output. Fitness can be very complicated until you get more experienced so that's why this app will make the beginners lives easier. This app will also help intermediate and advanced fitness people. They will know what they're doing so they will be open to trying different things in all aspects so with the help of tracking they will be able to find out which approach works best for them.

1.3. Technology

My Gym App will be built on android studio with the use of Java, HTML,XML, CSS, JavaScript, Bootstrap and SQL. The SQL will be used for the database to store the users details on Microsoft Management Studio . The rest of the technologies will be used to design and implement the app on Android Studio.

2.0 System

2.1. Non Functional Requirements

1. The App is reliable and consistent in terms of loading speed
2. The App is secure so the users account and payment details don't get leaked
3. The App will have large capacity requirement to store large amounts of data by the user
4. Its an android studio app so only compatible with android devices

2.1.1. Functional Requirements

Priority 1: Requirement 1: The user has to be able to register and login without this they can't use the app, so this requirement is the most important

Priority 2: Requirement 8: Without the ability to pay the user will not be able to pay for gym membership or book any PTs, only with payment this app will be able to used its full potential by the user.

Priority 3: Requirement 3: Home page is the main navigating tool as all the parts of the app can be accessed from here

Priority 4: Requirement 10: The Navigation Menu is next in terms of importance as no matter which page the user is on they can always redirect to any page within the app with the use of the navigation menu

Priority 5: Requirement 2: The main purpose of help for this app is for users to go gym, allowing the user to buy membership and book time slots allows for this to happen

Priority 6: Requirement 4: PTs will help guide users on the right path to success, this part is important especially for beginners

Priority 7: Requirement 5: Calorie tracking is next level of importance as it allows the user to put in how much food they ate and how much calories they've burned through activity. This feature will be important for progress

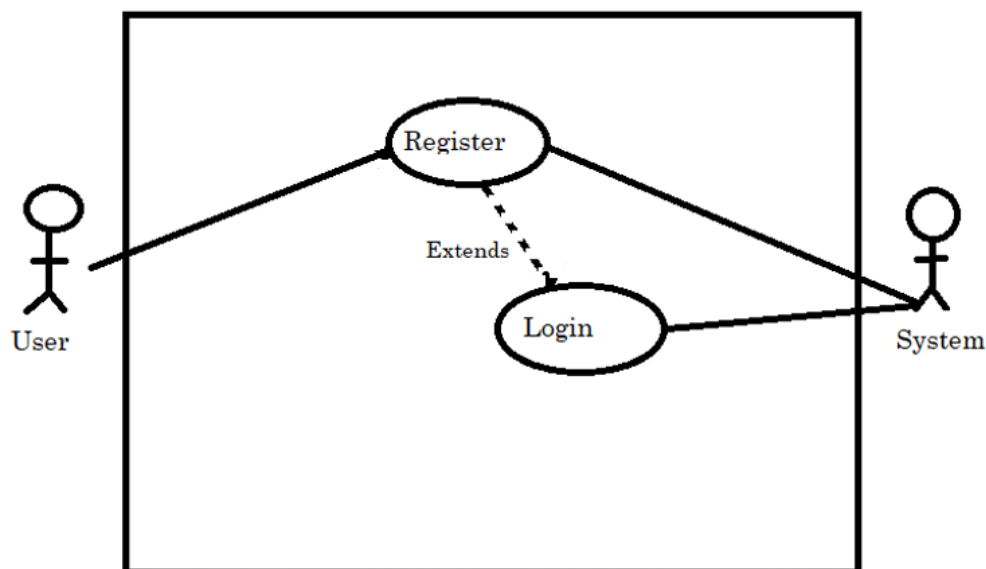
Priority 8: Requirement 6: Workout Tracking is next level of importance as it will allow users to keep tabs on their workouts and how they are progressing compared to their older workouts. This feature will also be important for progress

Priority 9: Requirement 7: It's important the user is allowed to make changes to their account in case some of their information needs changing from the time of signing up.

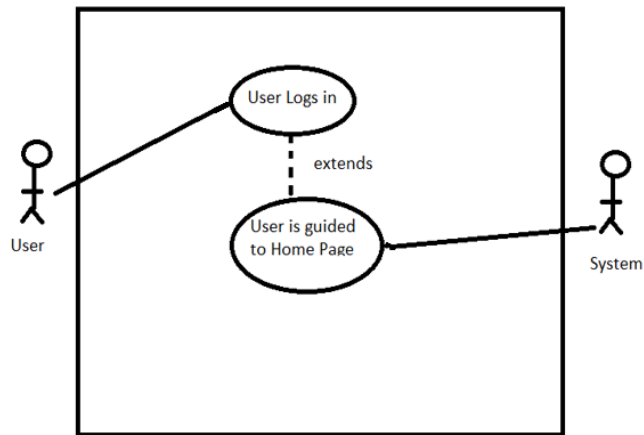
Priority 10: Requirement 9: The Search Functionality is important as it makes the users life easier by letting them search for exactly what they want but the features above are more important. account in case some of their information needs changing from the time of signing up.

2.1.1.1. Use Case Diagram

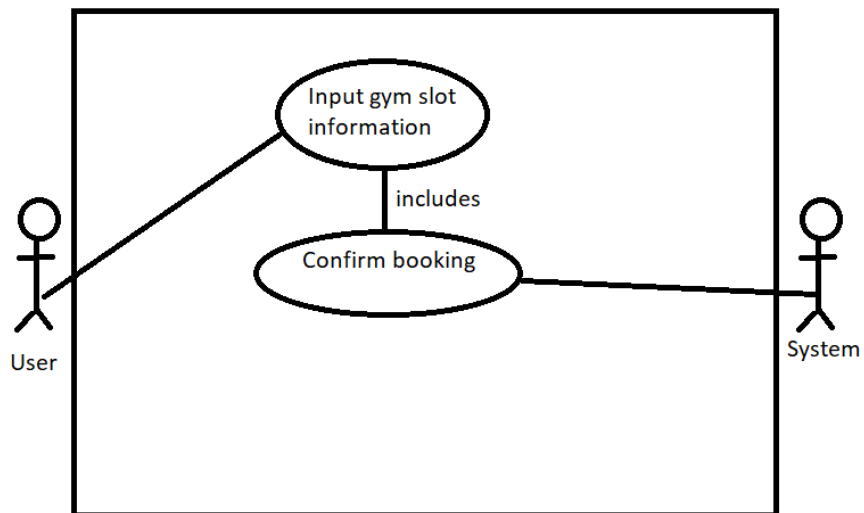
Registration and Login Use Case



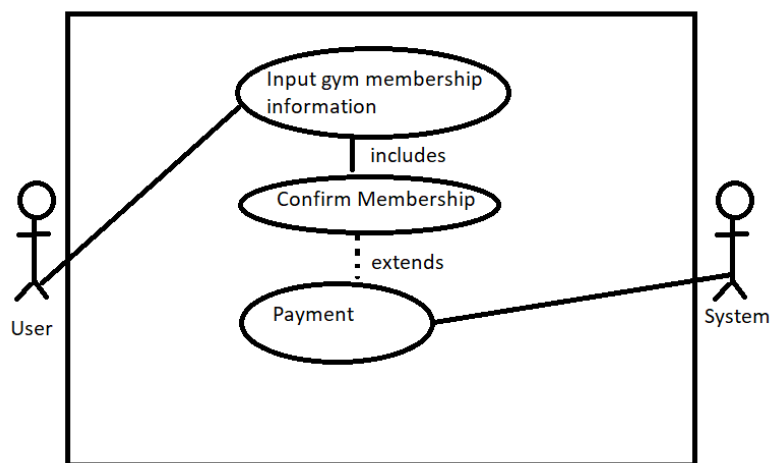
Home Page Use Case



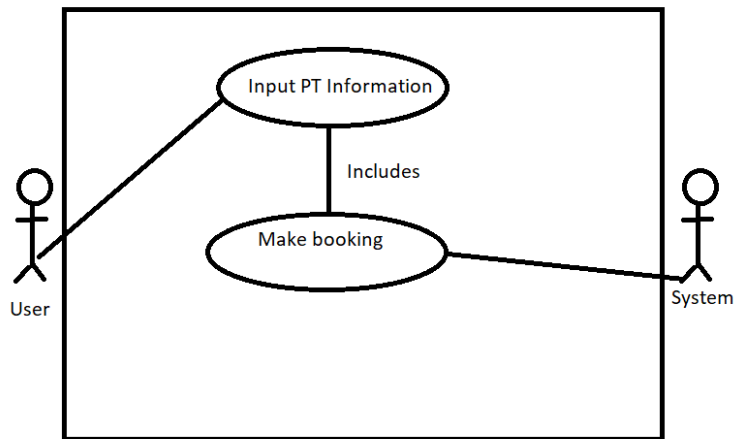
Gym Slot booking Use Case



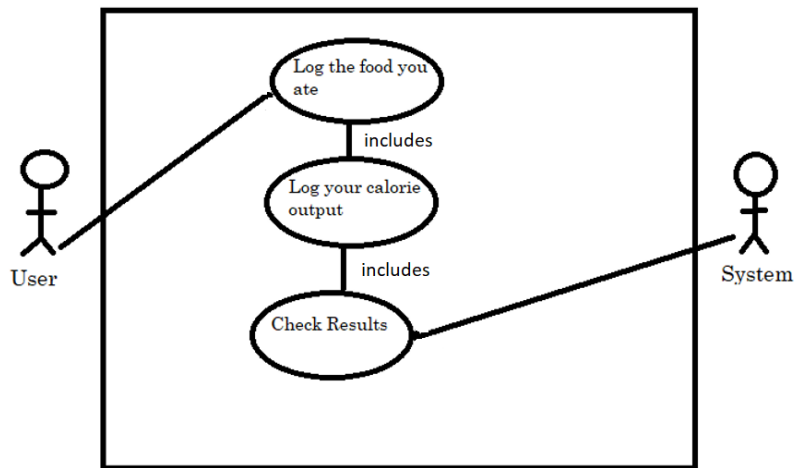
Gym Membership Booking Use Case



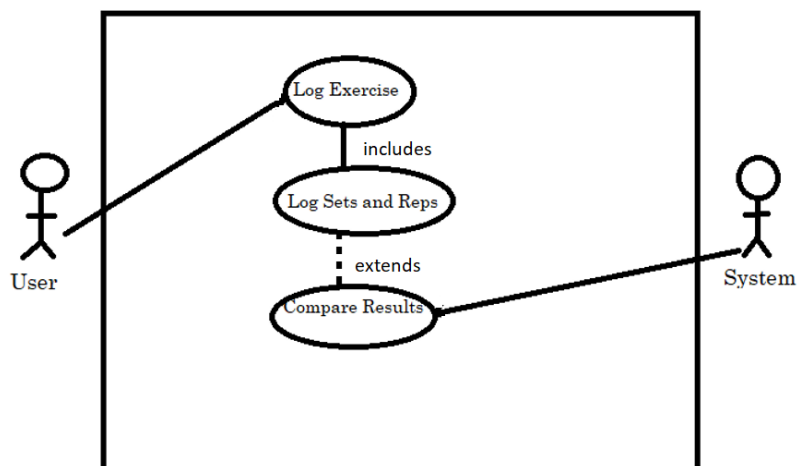
PT Booking Use Case



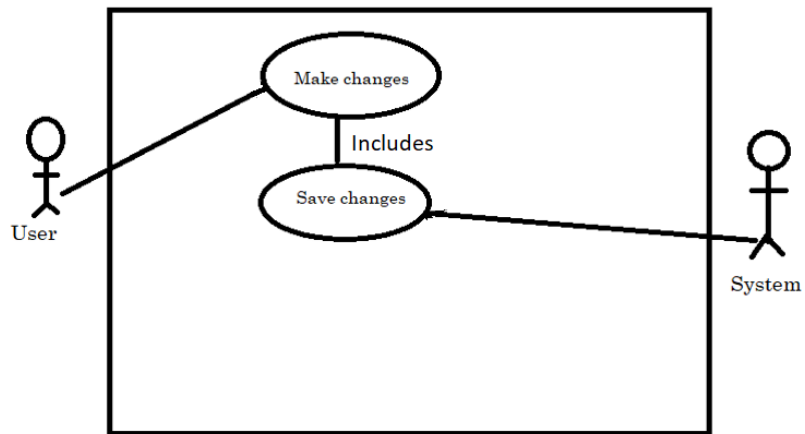
Calorie Tracking Use Case



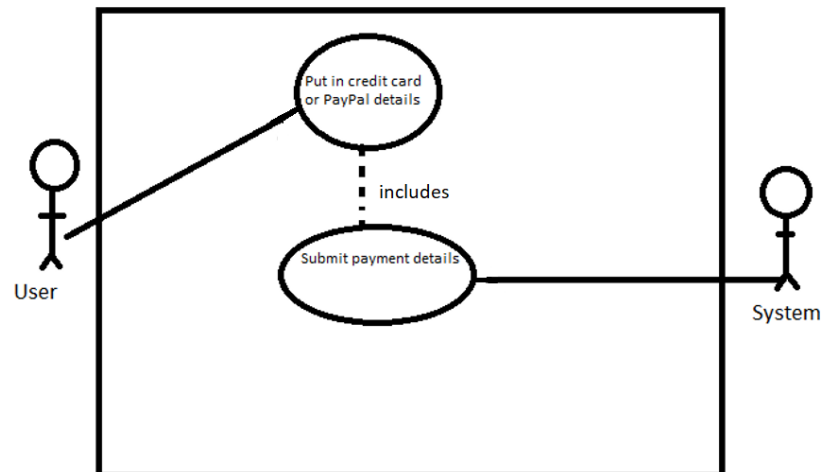
Workout Tracking Use Case



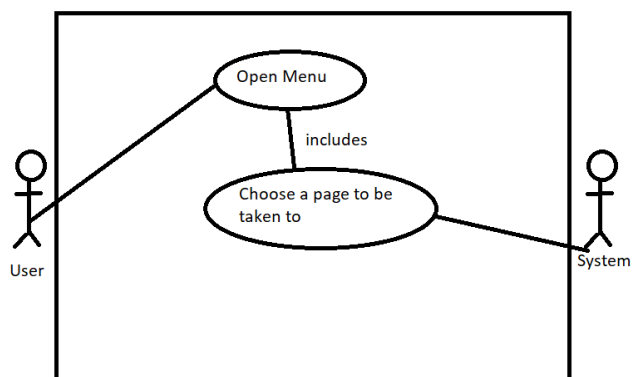
Settings Use Case



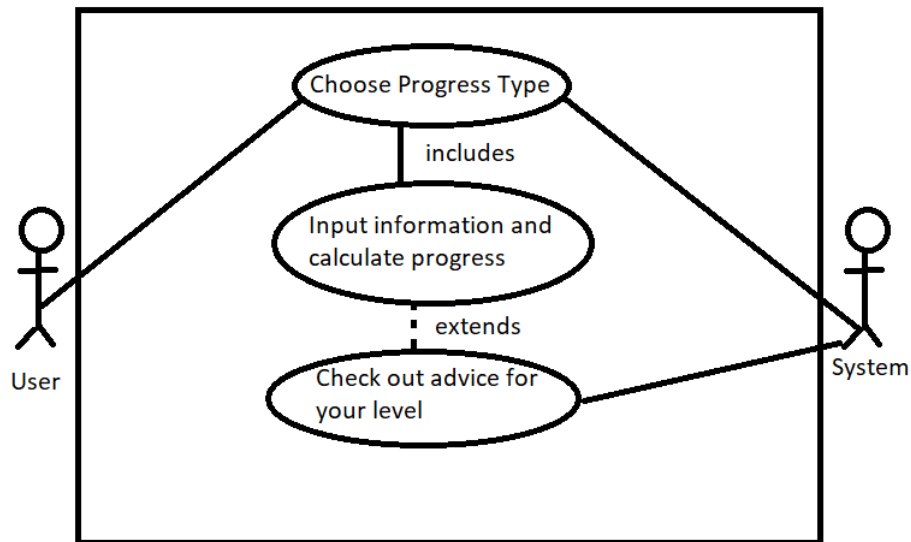
Payment Use Case



Navigation Menu Use Case



Progress Use Case



2.1.1.2. Functional Requirements

Requirement 1: Registration and login

Requirement 2: Membership and gym slot booking

Requirement 3: Home Page

Requirement 4: PT booking

Requirement 5: Calorie Tracking

Requirement 6: Workout Tracking

Requirement 7: Account Settings

Requirement 8: Payment Authentication

Requirement 9: Search Functionality

Requirement 10: Navigation Menu

2.1.1.3. Description & Priority

Priority 1: Requirement 1: The user has to be able to register and login without this they can't use the app, so this requirement is the most important

Priority 2: Requirement 8: Without the ability to pay the user will not be able to pay for gym membership or book any PTs, only with payment this app will be able to use its full potential by the user.

Priority 3: Requirement 3: Home page is the main navigating tool as all the parts of the app can be accessed from here

Priority 4: Requirement 10: The Navigation Menu is next in terms of importance as no matter which page the user is on they can always redirect to any page within the app with the use of the navigation menu

Priority 5: Requirement 2: The main purpose of help for this app is for users to go gym, allowing the user to buy membership and book time slots allows for this to happen

Priority 6: Requirement 4: PTs will help guide users on the right path to success, this part is important especially for beginners

Priority 7: Requirement 5: Calorie tracking is next level of importance as it allows the user to put in how much food they ate and how much calories they've burned through activity. This feature will be important for progress

Priority 8: Requirement 6: Workout Tracking is next level of importance as it will allow users to keep tabs on their workouts and how they are progressing compared to their older workouts. This feature will also be important for progress

Priority 9: Requirement 7: It's important the user is allowed to make changes to their account in case some of their information needs changing from the time of signing up.

Priority 10: Requirement 9: The Search Functionality is important as it makes the users life easier by letting them search for exactly what they want but the features above are more important.

2.1.1.4. Use Case Time

Scope

Use Case 1: 3/4 hours

Use Case 2: 3/4 hours

Use Case 3: 3/4 hours

Use Case 4: 4/6 hours

Use Case 5: 6/8 hours

Use Case 6: 6/8 hours

Use Case 7: 3/4 hours

Use Case 8: 4/6 hours

Use Case 9: 5/6 hours

Use Case 10: 3/4 hours

Description

Use Case 1: Register and Login

Use Case 2: Membership and Gym Slot booking

Use Case 3: Home Page

Use Case 4: PT Booking

Use Case 5: Calorie Tracking

Use Case 6: Workout Tracking

Use Case 7: Settings

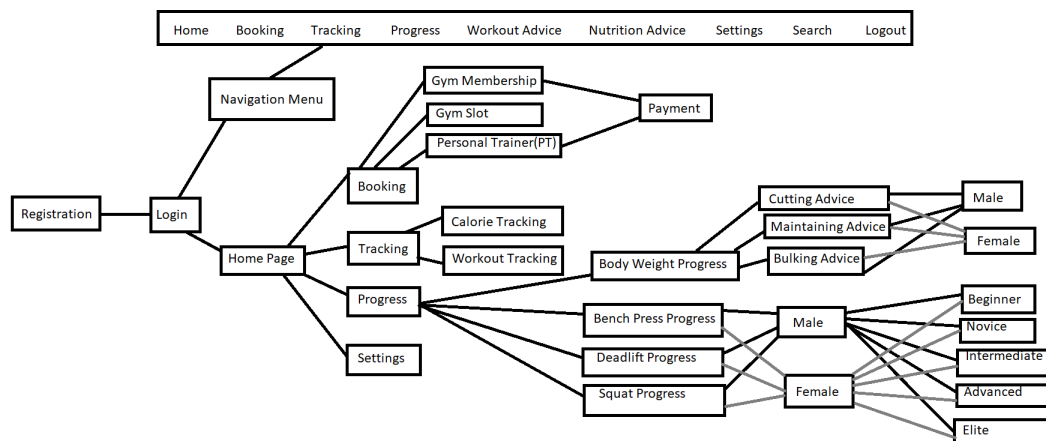
Use Case 8: Payment Authentication

Use Case 9: Search Functionality(New Feature)

Use Case 10: Navigation Menu

2.2. Design & Architecture

App run through/walkthrough diagram



This app is based on a registration & login like any other app and once you login you have a various amount of features to use which is provided by the app. Once logged in you are brought to the home screen which contains 4 of the main features of the app which you can click into and use. The home page as well as every page within this app contains a navigation menu which allows the user to access whichever feature at any time of them using the app. If what they want cannot be found in the menu (since only the main headings are in the menu) they can use the search functionality and get to where they want that way. A key characteristic of this app is when one major feature is chosen it will contain other features within it e.g. when booking is chosen you can book gym membership, gym slot and a PT. The user will be able to insert, update, view and delete information when e.g. booking a gym slot and they choose the gym slot they insert their initial training time and if they change their mind they can update the training time and view the booking to make sure they've put in the correct times or delete the booking altogether if they cannot make their session for whatever reason. The user will be able to use this functionality wherever they are required to submit information so e.g. workout & calorie tracking, registration & login details and settings to name a few examples. The levels for lifting progress are: Beginner, Novice, Intermediate, Advanced and Elite with advice given to each level in order to improve and go above their current level.

The user will be able to track their progress in terms of nutrition & body weight progress and workout & lifting progress and they will be put in certain categories based on their results and will be given advice based on their result in order to be able to improve since my aim with the app is to help every user progress. The advice section are split into Nutrition and workout advice and within each there is summarised advice beside each level with detailed advice for that level with the click of a button beside it. The levels are for body weight progress: Underweight, Normal weight and Overweight and the advice given is for bulking(underweight), maintaining(normal weight), cutting(overweight). The user will also have to pay for the different services provided by the app such as gym memberships and booking Personal Trainers. The user will be asked for their credit card details and they will submit the information but before the payment can be confirmed the payment information the user submits will come up as a message to make sure the user put in the correct payment information and then once the user is sure the payment

can be confirmed and processed. The progress for workouts will be split into female and male categories since females and males have different strength standards.

The user will be able to change their account information in the settings page. If the user wants to change their current login credentials they can, in the settings page they can view their current login credentials before updating it to new login credentials. If the user has moved house they can change their home address in the settings page by updating it and the same goes for the users phone number if that has been changed they can update it in the settings page. It is important for the user to always have their latest information in the app. The users full name and email address can also be changed if there is a need for that in the settings page.

The algorithm used in the making of the registration and login page is very important for security reasons as the user can only register and login once they provide their correct credentials. A series of counter if and else statements are included to only let the user register or login once the correct credentials are provided to prevent any unwanted users using a certain users account and potentially stealing data. Security is one of the most important factors in creating software as the user has to feel secure and safe when using a software and not feeling threatened that their personal details may be stolen by someone that breaks into their account so that's why the registration and login process is so secure.

The insert, view, update and delete methods were crucial in this app each one for their own reasons. The insert method was crucial because it allowed the user to input data in and store it in a safe place which is a SQLite database. The user was able to safely input data throughout the app in the various features it provided so the registration, login, workout & calorie tracking, the various booking pages, the various progress pages and settings. The update feature was important in a scenario where something needed updating for the user e.g. an email address or a home address, it is important to always have the latest information possible provided into the app in case that piece of information has to be used in order to help the customer in a case of confusion.

The view method is important as it shows the user what they have input and the user can use this to make sure they have put in the right information and if they haven't this is where the delete method comes in as the user can delete data if they have put in wrong information.

The K-nearest algorithm implementation in the body weight progress and the workout progress(bench, squat and deadlift) is one of the most important algorithm in this project as it is the most complex and it provides the user with very useful information followed up with advice based on their current result in order to improve either a certain lift or their BMI Score/body weight. The algorithm is implemented for both males and females as both sexes need to benefit from this help and advice. For the body weight progress the users height and weight will be taken to calculate their BMI score based on the BMI formula which is $\text{weight}/\text{height}^2$. When the BMI is calculated a

message will pop up letting the user know if they're underweight, normal weight or overweight and based on that result they will be taken to a new page with advice for their certain group. Overweight will be brought to cutting advice, normal weight to maintaining advice and underweight to bulking advice. From whichever page they're taken to they will be able to view their current nutrition plan, will be able to create a new plan and view advice for other weight groups.

The Array List in the Search functionality was a key to making it work as it allowed the possibility to search for whichever feature within the app which was handy for the user since they didn't have to manually navigate to the feature they wanted to access since they could just search for it. When an item is being searched it will have an image, title and a description to help the user know if what they've searched for is what they intended to search for. The way this Array List works is that the user searches a key word or even types in some letters and whichever feature contains that word or letters they will come up as a suggestion for the user to pick from.

All the buttons used in every page in this application are key since they link different pages of this app together without the use of buttons transitions would not be possible e.g. in the progress feature the user would not be able to be brought to whichever advice page without the use of Buttons. Without the use of buttons the user would not be able to insert, update, view or delete data. The intent function with the use of buttons is also crucial in the menu functionality since without it the user would not be able to access any feature which is included in the navigation menu. So in summary buttons are key in this application to link the app together.

The switch function with cases is important in the navigation menu since it provides the user access to all key features within the app with the use of intent. Once the feature is clicked on the menu the user will be brought to that feature since that activity will be launched when clicked on the navigation menu. Another important part of the navigation menu is the extension of the NavDrawerBase class (The java class for the menu) to every activity and the activity binding which allows the menu to extend to whichever activity in the app so obviously the whole app since the menu has to be linked in the whole application.

2.3. Implementation

Registration and Login screenshots and code explanation

Below we see all the different scenarios when using the Registration and Login. In the first screenshot shows us what happens when we try register without filling in any information. So obviously we can't log in because we haven't supplied any information so the app tells us we have to "Fill in all the fields". In the second scenario we see what happens when we fill in the information but our passwords don't match, the app requires you to fill in your password then under confirm the password again. So in this

scenario they don't match which causes the app to tell you "Passwords not Matching" so you have to repeat the password correctly then you will successfully register.

In the third Scenario we see what happens when you put in the information correctly so the username and the password correctly twice, you then register successfully and the app lets you know "Registration Successful". In the fourth scenario we see what happens when you try register as an account that already exists, since the account already exists the app tells you "User Already exists. Please Sign in" so then you log in.

The fifth scenario shows us what happens when you try login but you put in the wrong credentials, the app tells you "Invalid Credentials, Please try again" so we have to put in the correct credentials in order to be able to login. The sixth scenario shows us what happens when you successfully login, you are taken to the home page and you can use the app freely from there.



Coding screenshots responsible for each scenario

Here in the code we tell the app if the username,password and repeat password fields are blank to print out a message telling the user to fill in all the fields

```
btnSignUp.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        String user = username.getText().toString();  
        String pass = password.getText().toString();  
        String repass = repassword.getText().toString();  
  
        if(user.equals("") || pass.equals("") || repass.equals(""))  
        {  
            Toast.makeText(context: MainActivity.this, text: "Fill all the fields.",Toast.LENGTH_SHORT).show  
        }  
        else  
        {  

```

In the below screenshot we see the code responsible for each of the scenario, I've written the number of which piece of code is responsible for which scenario. They're all a series of if and else statements so if one thing doesn't happen then this will happen and if that doesn't happen then another thing will happen and so on. So here if the registration isn't successful a message comes up on the screen for the registration either failing, to fill in all fields, user already exists or passwords are not matching

```
if(pass.equals(repass))  
{  
    Boolean usercheckResult = myDB.checkusername(user);  
    if(usercheckResult == false)  
    {  
        Boolean regResult = myDB.insertData(user,pass);  
        3 if(regResult == true){  
            Toast.makeText(context: MainActivity.this, text: "Registration Successful",Toast.LENG  
            Intent intent = new Intent(getApplicationContext(),SignUpActivity.class);  
            startActivity(intent);  
        }  
        else  
        {  
            Toast.makeText(context: MainActivity.this, text: "Registration Failed",Toast.LENGTH_S  
        }  
    }  
    else 4  
    {  
        Toast.makeText(context: MainActivity.this, text: "User Already Exists.\n Please Sign In",  
    }  
} else 2  
{  
    Toast.makeText(context: MainActivity.this, text: "Password not Matching",Toast.LENGTH_SHORT)]  
}
```

Below we see what the code does in the login page, so either you don't enter any credentials and the app then tells you to enter your credentials then more if and else statements. So next is you input the credentials correctly the app will use intent to bring you to the Home Page and lastly if you input the wrong credentials the app will tell you your credentials are wrong and you have to try login again with correct credentials

```

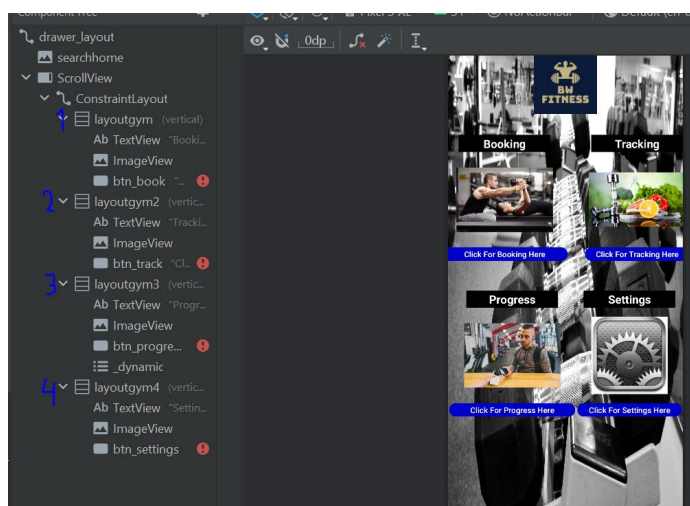
btnLogin.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String user = username.getText().toString();
        String pass = password.getText().toString();

        if(user.equals("") || pass.equals("")){
            Toast.makeText(context: SignUpActivity.this, text: "Please Enter Your credentials.", Toast.LENGTH_SHORT);
        }
        else
        {
            Boolean result = myDB.checkusernamePassword(user,pass);
            if(result==true){
                Intent intent = new Intent(getApplicationContext(),HomePageActivity.class);
                startActivity(intent);
            }
            else
            {
                Toast.makeText(context: SignUpActivity.this, text: "Invalid Credentials, Please Try Again.", Toast.LENGTH_SHORT);
            }
        }
    }
});

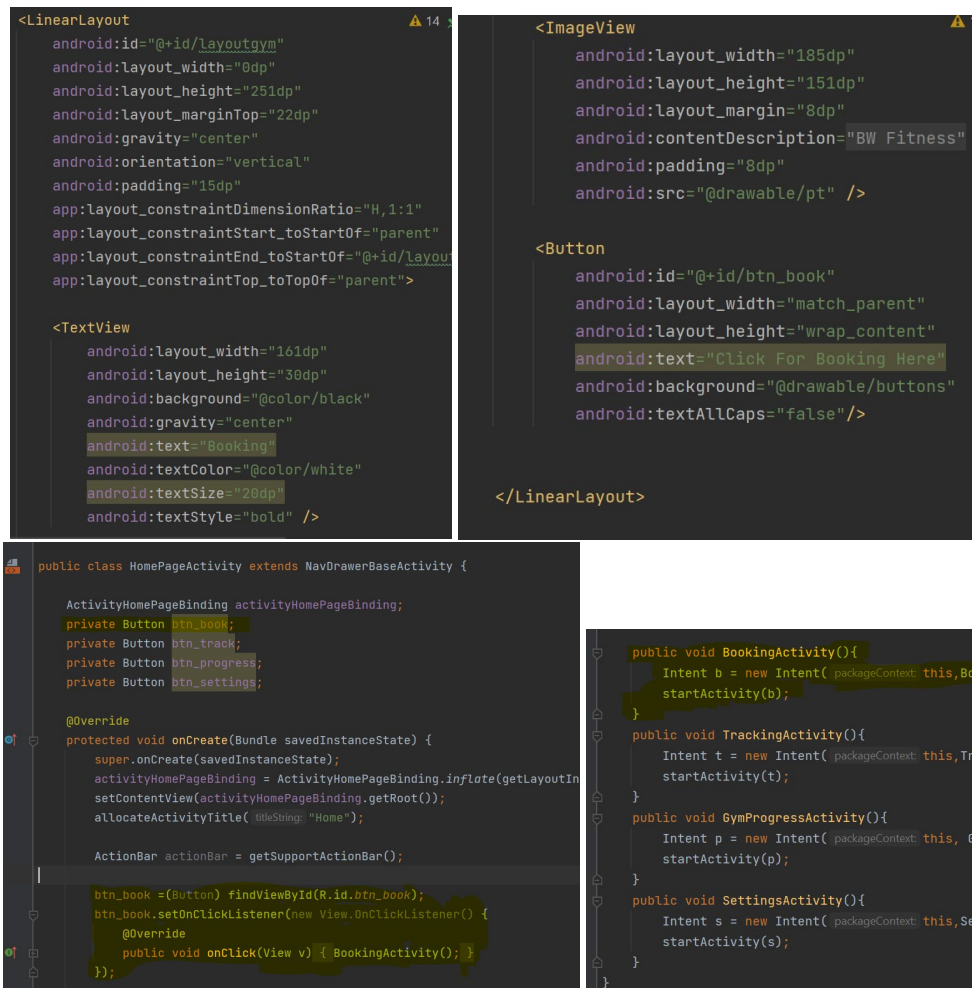
```

Home page screenshots and code explanation

Below we see the structure of the home page it is a series of 4 Linear Layouts that include a title an image and a button which when clicked brings you to that section of the app. The Sections being Booking,Tracking,Progress and Settings more features can be found in the navigation menu and in the search functionality which I will cover later. The home page also includes a background image just like any page on this app and also the app logo in the middle top of the screen.

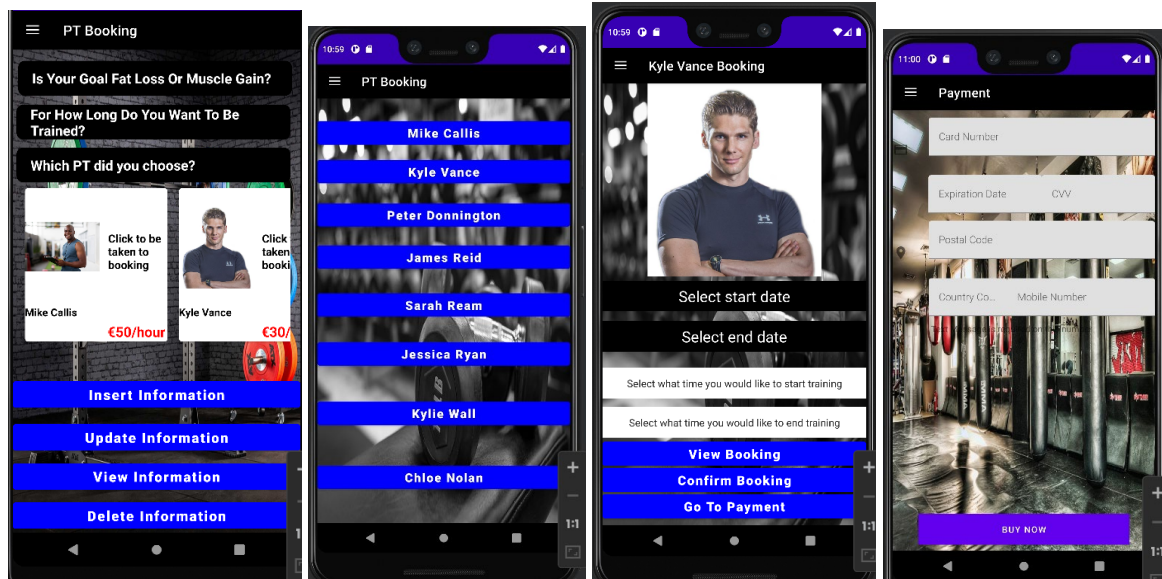


Below I've included screenshots of the xml file which shows the design fundamentals of one of the linear layouts, all 4 of them have the same fundamentals so showing just one shows how they all work. Then I've shown the java code of the home page which are buttons to take you to different pages/activities. The highlighted code shows initiating the booking button then finding it by its ID then launching the activity for booking using Intent.

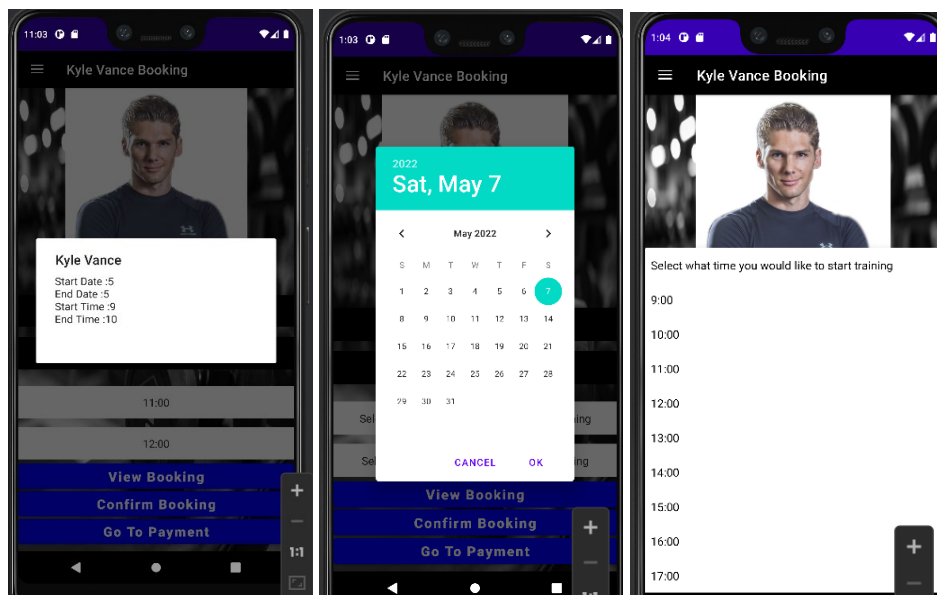


PT Booking page with code explanation

In the below screenshots you see what is included in the PT booking page the user puts in their training goal, training length and which PT they choose. The user chooses the PT from a recycler view menu, the app supplies the user with 8 PTs 4 being male 4 being female. The user can then insert, view, update and delete the information they put into the PT booking page. When the recycler view is clicked it brings us to another page in which we choose which PT we want to book and from there we book whichever PT and we are brought to Payment. When making the booking the user has to select a start and end date they will be booking the PT for and the times they will like to train with them. A calendar feature is used for the dates and a spinner with an array of times is used for the booking times.



When the booking is confirmed successfully it can be viewed using the View Booking button, here is also a screenshot of the calendar and spinner features. The spinner acts as a dropdown menu.



Below we see the code from this feature. We initiate the text in order to input the information, the buttons to submit the information, the DBHelper in order to store the information on the database and the RecyclerView to show the PT list menu.

```
public class PTActivity extends NavDrawerBaseActivity implements
    ActivityPtactivityBinding activityPtactivityBinding;
    EditText target_performance, pt_length, pt_payment;

    Button insert_pt, update_pt, view_pt, delete_pt;

    DBHelper myDB;

    RecyclerView trainer_recycler;
```

Below here I will show the code for the insert,view,update and delete functions.

Insert: Here the text would be received from the EditText fields and the method from the DB Helper class would insert the data into the pt table into the database

```
insert_pt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String target_performanceTXT = target_performance.getText().toString();
        String pt_lengthTXT = pt_length.getText().toString();
        String pt_paymentTXT = pt_payment.getText().toString();

        Boolean checktrainerdata = myDB.inserttrainerdata( target_performanceTXT , pt_lengthTXT , pt_paymentTXT );
        if(checktrainerdata==true) {
            Toast.makeText( context: PTAactivity.this, text: "New Entry Inserted",Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: PTAactivity.this, text: "Entry Not Inserted",Toast.LENGTH_SHORT).show();
    }
});

public Boolean inserttrainerdata(String target_performance , String pt_length , String pt_payment )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues content = new ContentValues();
    content.put(" target_performance " , target_performance );
    content.put(" pt_length " , pt_length );
    content.put(" pt_payment " , pt_payment);

    long result = myDB.insert( table: " pt_table " , nullColumnHack: null, content);
    if(result==-1){
        return false;
    }
    else
    {
        return true;
    }
}
```

Update:

Here the text would be received from the EditText fields and the method from the DB Helper class would update the data into the pt table into the database

```
update_pt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String targetperformanceTXT = target_performance.getText().toString();
        String pt_lengthTXT = pt_length.getText().toString();
        String pt_paymentTXT = pt_payment.getText().toString();

        Boolean checkupdatetrainerdata = myDB.updatetrainerdata( targetperformanceTXT , pt_lengthTXT , pt_paymentTXT );
        if(checkupdatetrainerdata==true) {
            Toast.makeText( context: PTAactivity.this, text: "Entry Has Been Updated",Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: PTAactivity.this, text: "Entry Did Not Update",Toast.LENGTH_SHORT).show();
    }
});
```

```

public Boolean updatetrainerdata( String target_performance , String pt_length , String pt_payment )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues content = new ContentValues();
    content.put(" target_performance ", target_performance );
    content.put(" pt_length ", pt_length );
    content.put(" pt_payment ", pt_payment );

    Cursor cursor = myDB.rawQuery( sql: "Select * from pt_table where target_performance=?",new String[]{});
    if(cursor.getCount()>0)
    {
        long result = myDB.update( table: " pt_table ", content, whereClause: " target_performance=? ", new String[]{});
        if (result == -1) {
            return false;
        } else {
            return true;
        }
    }else{
        return false;
    }
}

```

Delete: Here only the primary key passed in and the method from the DB Helper class would delete the data from the pt table in the database by only using the primary key from that table.

```

delete_pt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String targetperformanceTXT = target_performance.getText().toString();

        Boolean checkdeletetrainerdata = myDB.deletetrainerdata( targetperformanceTXT );
        if(checkdeletetrainerdata==true) {
            Toast.makeText( context: PTAActivity.this, text: "Entry Has Been Deleted",Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: PTAActivity.this, text: "Entry Did Not Delete",Toast.LENGTH_SHORT).show();
    }
});

```

```

public Boolean deletetrainerdata( String target_performance )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from PersonalTrainer where target_performance=?",new String[]{});
    if(cursor.getCount()>0)
    {
        long result = myDB.delete( table: " pt_table ", whereClause: " target_performance=? ", new String[]{});
        if (result == -1) {
            return false;
        } else {
            return true;
        }
    }else{
        return false;
    }
}

```

View : Here the text that has been input is received and is then viewed along with a title beside it and the method from the DB Helper class will view the data from the pt table in the database. If no information has been inserted then no information can be viewed so then the “No Entry Found” message will come up

```

view_pt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Cursor res = myDB.viewtrainerdata();
        if(res.getCount()==0){
            Toast.makeText( context: PTAactivity.this, text: "No Entry Found",Toast.LENGTH_SHORT).show();
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext()){
            buffer.append(" target_performance :"+res.getString( columnIndex: 0)+"\n");
            buffer.append(" pt_length :"+res.getString( columnIndex: 1)+"\n");
            buffer.append(" pt_payment :"+res.getString( columnIndex: 2)+"\n\n");
        }

        AlertDialog.Builder builder = new AlertDialog.Builder( context: PTAactivity.this);
        builder.setCancelable(true);
        builder.setTitle(" PersonalTrainer ");
        builder.setMessage(buffer.toString());
        builder.show();
    }
});
}

public Cursor viewtrainerdata()
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: " Select * from pt_table ", selectionArgs: null);
    return cursor;
}
}

```

Below I will show the ArrayList responsible for showing the PTs within the RecyclerView Menu. The model class initiated variables for name, price and an image to be viewed by the RecyclerView. All the methods to setup the adapter for the recycler view were made in the personalTrainerAdapter class

```

List<PersonalTrainer> personalTrainerList = new ArrayList<>();

personalTrainerList.add(new PersonalTrainer( name: "Mike Callis", price: "€50/hour", R.drawable.pt_p1));
personalTrainerList.add(new PersonalTrainer( name: "Kyle Vance", price: "€30/hour", R.drawable.pt_pt2));
personalTrainerList.add(new PersonalTrainer( name: "Peter Donnington", price: "€40/hour", R.drawable.pt_pt3));
personalTrainerList.add(new PersonalTrainer( name: "James Reid", price: "€35/hour", R.drawable.pt_p4));
personalTrainerList.add(new PersonalTrainer( name: "Sarah Ream", price: "€55/hour", R.drawable.pt_p5));
personalTrainerList.add(new PersonalTrainer( name: "Jessica Ryan", price: "€40/hour", R.drawable.pt_p6));
personalTrainerList.add(new PersonalTrainer( name: "Kylie Wall", price: "€40/hour", R.drawable.pt_p7));
personalTrainerList.add(new PersonalTrainer( name: "Chloe Nolan", price: "€45/hour", R.drawable.pt_p8));

setTrainerRecycler(personalTrainerList);
}

private void setTrainerRecycler(List<PersonalTrainer> personalTrainerList){
    trainer_recycler=findViewById(R.id.trainer_recycler);
    RecyclerView.LayoutManager layoutManager = new LinearLayoutManager( context: this,RecyclerView.HORIZONTAL);
    trainer_recycler.setLayoutManager(layoutManager);
    personalTrainerAdapter=new PersonalTrainerAdapter( context: this,personalTrainerList, recyclerViewInterface);
    trainer_recycler.setAdapter(personalTrainerAdapter);
}
}

```

Below I will show the code responsible for booking a PT. Firstly I've shown what happens when the confirm booking button is clicked, we get the input from the calendar and the spinners for the booking dates and times and we use a method from the DB Helper class to insert the data into the database. The Method for the calendar is then initiated where the day, month and year are setup, the update label method showcases the calendar when the fields for the calendar dates are clicked on the PT Booking page.


```

btnBookKyle.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        String start_pt2TXT = start_pt2.getText().toString();
        String end_pt2TXT = end_pt2.getText().toString();
        String KyleSpinner1TXT = KyleSpinner1.getSelectedItem().toString();
        String KyleSpinner2TXT = KyleSpinner2.getSelectedItem().toString();

        Boolean checkkyldata = myDB.insertkyldata( start_pt2TXT , end_pt2TXT , KyleSpinner1TXT , KyleSp
        if(checkkyldata==true) {
            Toast.makeText( context: KyleVanceActivity.this, text: "Booking Successful", Toast.LENGTH_SHORT).show
        } else
            Toast.makeText( context: KyleVanceActivity.this, text: "Booking Unsuccessful", Toast.LENGTH_SHORT).ST
    }
});
}

//***** Kyle Vance PT Booking *****
public Boolean insertkyldata(String start_pt2 , String end_pt2 , String KyleSpinner1, String Kyle
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contents = new ContentValues();
    contents.put(" start_pt2 " , start_pt2);
    contents.put(" end_pt2 " , end_pt2);
    contents.put(" KyleSpinner1 " , KyleSpinner1);
    contents.put(" KyleSpinner2 " , KyleSpinner2);

    long result = myDB.insert( table: " kyldata ", nullColumnHack: null, contents);
    if(result>=1){
        return false;
    }
    else
    {
        return true;
    }
}

DatePickerDialog.OnDateSetListener date = new DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
        myCalendar2.set(Calendar.YEAR, year);
        myCalendar2.set(Calendar.MONTH, month);
        myCalendar2.set(Calendar.DAY_OF_MONTH, dayOfMonth);
        updateLabel2(myCalendar2, start_pt2);
    }
};

start_pt2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        new DatePickerDialog( context: KyleVanceActivity.this, date, myCalendar2
            .get(Calendar.YEAR), myCalendar2.get(Calendar.MONTH),
            myCalendar2.get(Calendar.DAY_OF_MONTH)).show();
    }
});

private void updateLabel2(Calendar myCalendar2, EditText start_pt2) {
    String myFormat = "MM/dd/yy";
    SimpleDateFormat sdf = new SimpleDateFormat(myFormat, Locale.UK);

    start_pt2.setText(sdf.format(myCalendar2.getTime()));
}

```

The user can also view their booking. Here the values that have been input is received and is then viewed along with a title beside it and the method from the DB Helper class will view the data from the pt table in the database. So In this case the titles will be start date, end date, start time and end time. If no information has been inserted then no information can be viewed so then the "No Entry Found" message will come up. An sql query is written in the DB Helper class to select all information from the table but in the java class the titles are used to only pick that certain information which I have already named.

```

btnViewKyle.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Cursor res = myDB.viewkyldata();
        if(res.getCount()==0){
            Toast.makeText( context: KyleVanceActivity.this, text: "No Entry Found", Toast.LENGTH_SHORT).show();
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext()){
            buffer.append(" Start Date : "+res.getInt( columnIndex: 0)+"\n");
            buffer.append(" End Date : "+res.getInt( columnIndex: 1)+"\n");
            buffer.append(" Start Time : "+res.getInt( columnIndex: 2)+"\n");
            buffer.append(" End Time : "+res.getInt( columnIndex: 3)+"\n\n");
        }

        AlertDialog.Builder builder = new AlertDialog.Builder( context: KyleVanceActivity.this);
        builder.setCancelable(true);
        builder.setTitle(" Kyle Vance ");
        builder.setMessage(buffer.toString());
        builder.show();
    }
});
}

```

```

public Cursor viewkyldata()
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from kytable ", selectionArgs: null);
    return cursor;
}

```

After the booking is made the user will be able to pay by clicking the Payment button and they will be taken to the payment page.

```

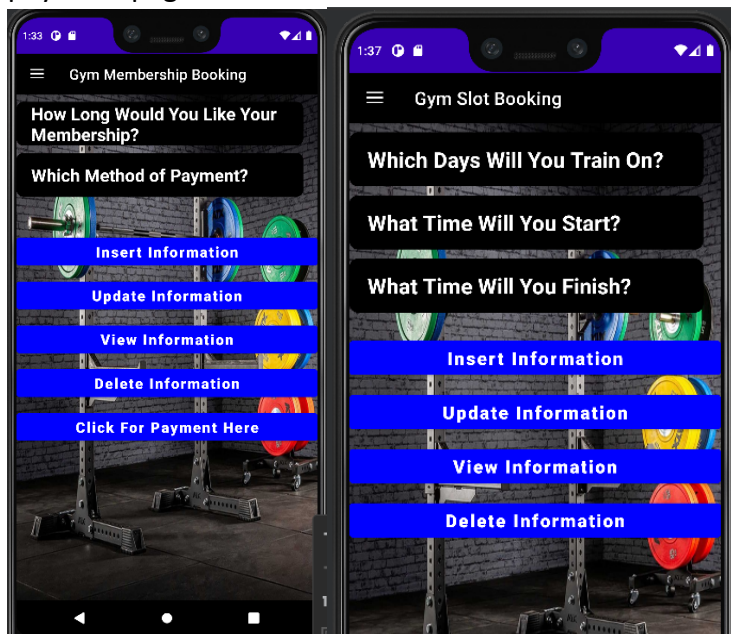
btnPayKyle =(Button) findViewById(R.id.btnPayKyle);
btnPayKyle.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { Payment_Activity(); }
});

public void Payment_Activity(){
    Intent p = new Intent( packageContext: this,Payment_Activity.class);
    startActivity(p);
}

```

Gym Slot and Gym Membership booking screenshots and code explanation

Both of these pages are similar in terms of how they are setup and how the code works so I will explain them together. They both have a series of EditText fields which information is written in and the submitted by a button and then the information is put into the database. In both pages the user can insert,update,view and delete their information. The only major difference being that in the gym membership booking page the user is required to bay hence the payment button at the bottom which will take the user to the payment page when clicked.



All the methods I've listed below work the same way on both pages so I will just show screenshots from the gym membership class to show how they both work

Insert: Here the text would be received from the EditText fields and the method from the DB Helper class would insert the data into the pt table into the database. If successful a new entry is inserted but if not noothing is inserted into the database

```

insert_membership.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String membership_lengthTXT = membership_length.getText().toString();
        String payment_methodTXT = payment_method.getText().toString();

        Boolean checkmembershipdata = myDB.insertmembershipdata( membership_lengthTXT , payment_methodTXT );
        if(checkmembershipdata==true) {
            Toast.makeText( context: MembershipActivity.this, text: "New Entry Inserted",Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: MembershipActivity.this, text: "Entry Not Inserted",Toast.LENGTH_SHORT).show();
    }
});
}

//-----Gym Membership-----
public Boolean insertmembershipdata(String membership_length , String payment_method )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues content = new ContentValues();
    content.put(" membership_length ", membership_length );
    content.put(" payment_method ", payment_method );

    long result = myDB.insert( table: " Membership ", nullColumnHack: null, content);
    if(result== -1){
        return false;
    }
    else
    {
        return true;
    }
}
}

```

Update: Here the text would be received from the EditText fields and the method from the DB Helper class would insert the data into the pt table into the database.If successful data is updated if not its not updated.

```

update_membership.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String membership_lengthTXT = membership_length.getText().toString();
        String payment_methodTXT = payment_method.getText().toString();

        Boolean checkupdatemembershipdata = myDB.updatemembershipdata( membership_lengthTXT , payment_methodTXT );
        if(checkupdatemembershipdata==true) {
            Toast.makeText( context: MembershipActivity.this, text: "Entry Has Been Updated",Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: MembershipActivity.this, text: "Entry Did Not Update",Toast.LENGTH_SHORT).show();
    }
});
}

public Boolean updatemembershipdata( String membership_length , String payment_method )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues content = new ContentValues();
    content.put(" membership_length ", membership_length);
    content.put(" payment_method ", payment_method);
    Cursor cursor = myDB.rawQuery( sql: "Select * from Membership where membership_length=?",new String[]{});
    if(cursor.getCount()>0)
    {
        long result = myDB.update( table: "Membership", content, whereClause: "membership_length=?", new String[]{});
        if (result == -1) {
            return false;
        } else {
            return true;
        }
    }else{
        return false;
    }
}
}

```

View: Here the text that has been input is received and is then viewed along with a title beside it and the method from the DB Helper class will view the data from the table in the database. If no information has been inserted then no information can be viewed so then the “No Entry Found” message will come up

```

view_membership.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Cursor res = myDB.viewmembershipdata();
        if(res.getCount()==0){
            Toast.makeText( context: MembershipActivity.this, text: "No Entry Found", Toast.LENGTH_SHORT).show();
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext()){
            buffer.append(" membership_length :"+res.getString( columnIndex: 0)+"\n");
            buffer.append(" payment_method :"+res.getString( columnIndex: 1)+"\n\n");
        }

        AlertDialog.Builder builder = new AlertDialog.Builder( context: MembershipActivity.this);
        builder.setCancelable(true);
        builder.setTitle(" Membership ");
        builder.setMessage(buffer.toString());
        builder.show();
    }
});

```

```

public Cursor viewmembershipdata()
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: " Select * from Membership ", selectionArgs: null);
    return cursor;
}

```

Delete: Here the text is only received from the Primary Key and the method from the DB Helper class will delete the data from the table just by identifying the primary key. If successful data is deleted if not its not deleted.

```

delete_membership.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String membership_lengthTXT = membership_length.getText().toString();

        Boolean checkdeletemembershipdata = myDB.deletemembershipdata( membership_lengthTXT );
        if(checkdeletemembershipdata==true) {
            Toast.makeText( context: MembershipActivity.this, text: "Entry Has Been Deleted", Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: MembershipActivity.this, text: "Entry Did Not Delete", Toast.LENGTH_SHORT).show();
    }
});

```

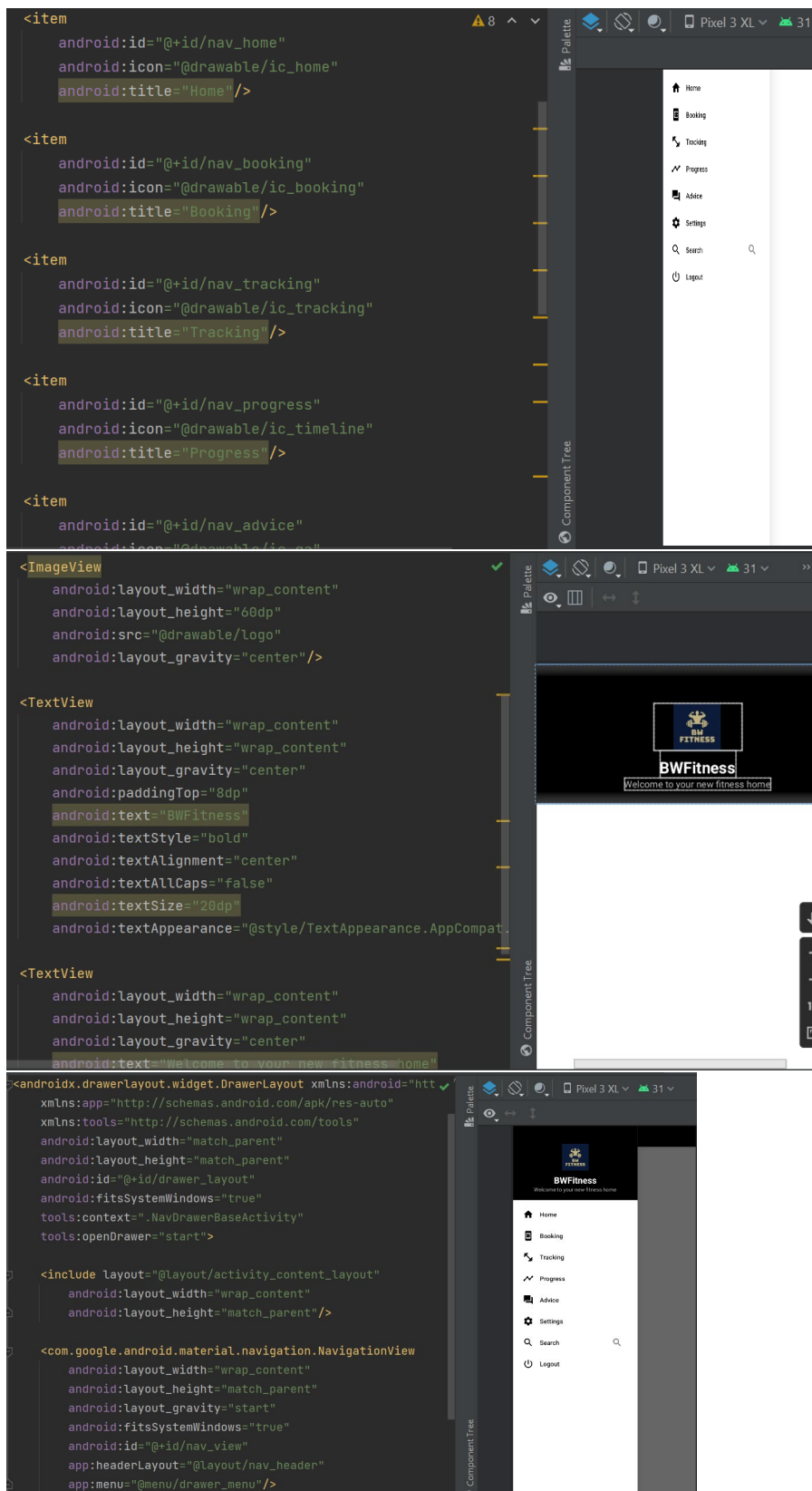
```

public Boolean deletemembershipdata(String membership_length)
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: " Select * from Membership where membership_length=?", new String[] {membership_length} );
    if(cursor.getCount()>0)
    {
        long result = myDB.delete( table: " Membership ", whereClause: " membership_length=?", new String[] {membership_length} );
        if (result == -1) {
            return false;
        } else {
            return true;
        }
    } else {
        return false;
    }
}

```

Navigation Menu screenshots and code explanation

Below we see how the menu is designed. An item is added for each feature within the menu and is given an ID in order to be found by the java file and also a title and icon to be shown on the menu beside each feature. I've also shown how the header on the menu is made which includes an image which is the app logo and 2 pieces of text below it.



Here below we see the code responsible for the menu. We call in the drawer layout which is the first class screenshoted above and set it as a view on whichever page it is viewed. The toolbar is also pulled in which will be visible at the top of the page, the nav view is then

pulled in which is the id of the whole menu put together so the drawer layout and the header put together. The toggle is then programmed to either open or close the menu.

```
public class NavDrawerBaseActivity extends AppCompatActivity implements NavigationView.OnNavigationItemSelectedListener {

    DrawerLayout drawerLayout;

    @Override
    public void setContentView(View view) {
        drawerLayout = (DrawerLayout) getLayoutInflater().inflate(R.layout.activity_nav_drawer_base, root: null);
        FrameLayout container = drawerLayout.findViewById(R.id.activityContainer);
        container.addView(view);
        super.setContentView(drawerLayout);

        Toolbar toolbar = drawerLayout.findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        NavigationView navigationView = drawerLayout.findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle( activity: this, drawerLayout, toolbar, "Drawer Open", "Drawer Close");
        drawerLayout.addDrawerListener(toggle);
        toggle.syncState();
    }
}
```

Here are the cases made for each feature within the menu so whenever a feature is clicked the will be brought to that page with the use of Intent.

```
@Override
public boolean onNavigationItemSelectedListener(@NonNull MenuItem item) {
    drawerLayout.closeDrawer(GravityCompat.START);

    switch (item.getItemId()){
        case R.id.nav_home:
            startActivity(new Intent( packageContext: this, HomePageActivity.class));
            overridePendingTransition( enterAnim: 0, exitAnim: 0);
            break;
        case R.id.nav_booking:
            startActivity(new Intent( packageContext: this, BookingActivity.class));
            overridePendingTransition( enterAnim: 0, exitAnim: 0);
            break;
        case R.id.nav_tracking:
            startActivity(new Intent( packageContext: this, TrackingActivity.class));
            overridePendingTransition( enterAnim: 0, exitAnim: 0);
            break;
        case R.id.nav_progress:
            startActivity(new Intent( packageContext: this, GymProgressActivity.class));
            overridePendingTransition( enterAnim: 0, exitAnim: 0);
            break;
    }
}
```

The title within the toolbar will be shown because of the method below

```
protected void allocateActivityTitle(String titleString) {
    if(getSupportActionBar() !=null){
        getSupportActionBar().setTitle(titleString);
    }
}
```

Below is what the menu looks like in the app, the toggle on the left, title of the page in the middle and the toolbar. The screenshot on the right is the menu when the toggle is clicked to open the menu.



In order for the menu to be included on whichever page/activity the `NavDrawerBase` class has to be extended which is responsible for making the menu and then it has to be initiated and binded within the class which is seen below. The title is then given of what you want the page title to be.

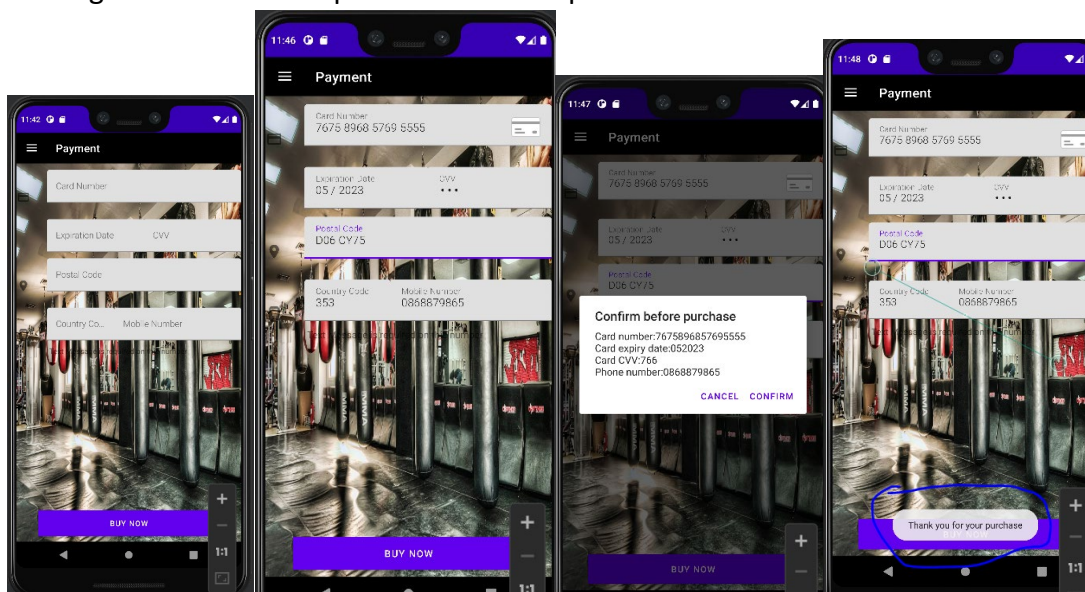
```
public class HomePageActivity extends NavDrawerBaseActivity {

    ActivityHomePageBinding activityHomePageBinding;
    private Button btn_book;
    private Button btn_track;
    private Button btn_progress;
    private Button btn_settings;

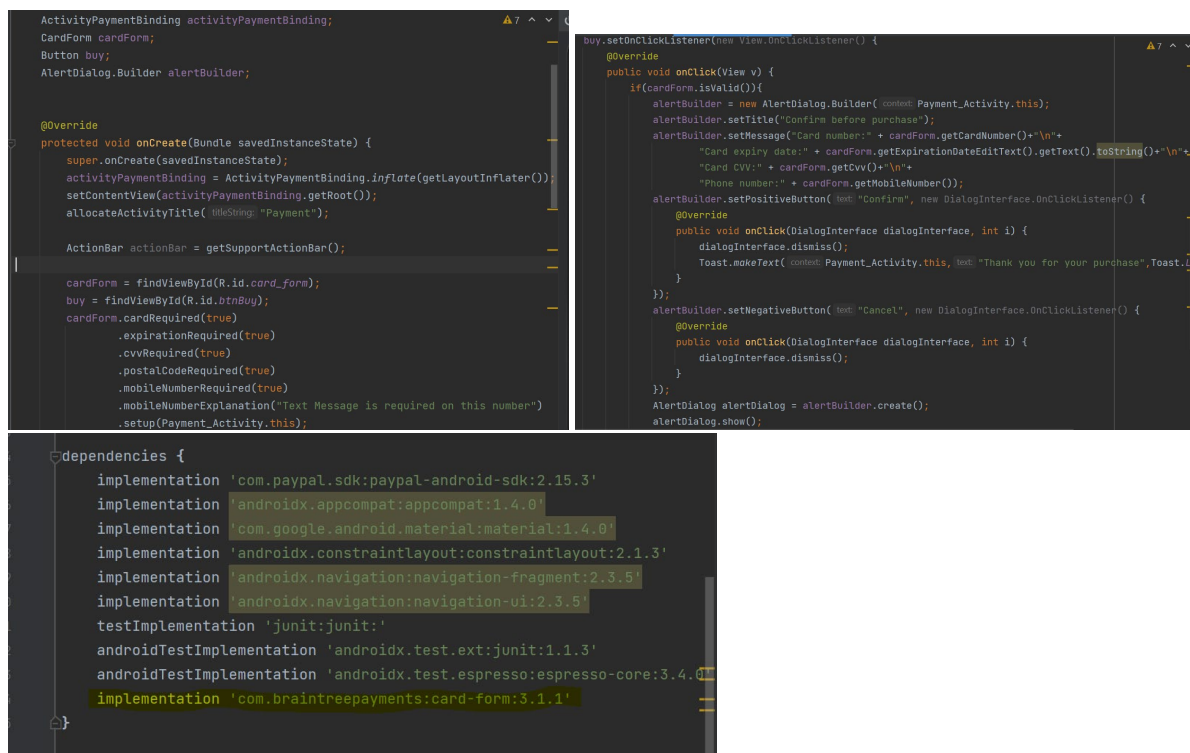
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        activityHomePageBinding = ActivityHomePageBinding.inflate(getLayoutInflater());
        setContentView(activityHomePageBinding.getRoot());
        allocateActivityTitle("Home");
    }
}
```

Payment page screenshots and code explanation

Below is a screenshot of what the payment page looks like it's a card form with a buy button underneath to confirm the purchase. I have then demonstrated the process I've filled out the card form the app then asks to confirm the purchase and when it's confirmed the message for a successful purchase comes up



In order for the payment page to work a gradle dependency had to be imported in as seen highlighted in the screenshot with dependencies. In the java class above we have to initiate the cardForm which will contain all the different fields within the payment which include card number, cvv number , expiration date , postal code and phone number. Then lastly a button to buy is initiated to complete the purchase once the card information is filled out. The alert builder is also brought in to print out messages depending on the action to the user. When the button to buy is clicked a message to confirm before purchasing will come up to confirm to make sure the user is sure of their details. Then when the purchase is successful the message thank you for your purchase will come up.

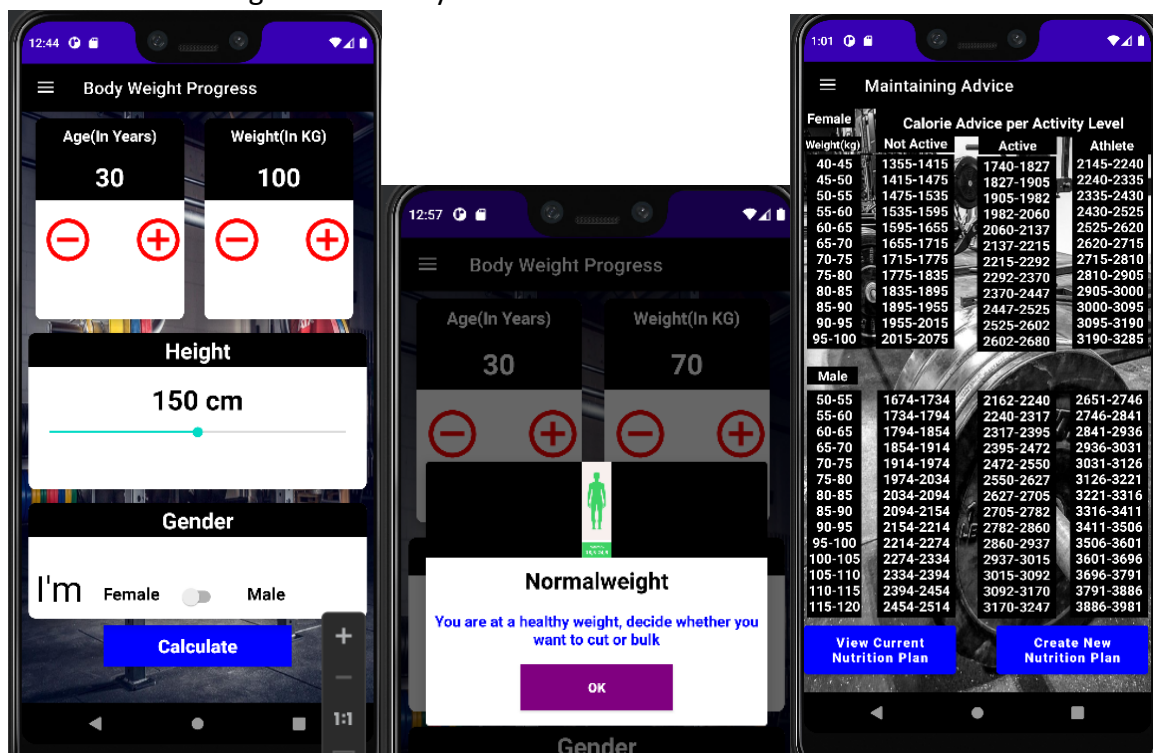


Body Weight progress page with screenshots and code explanation

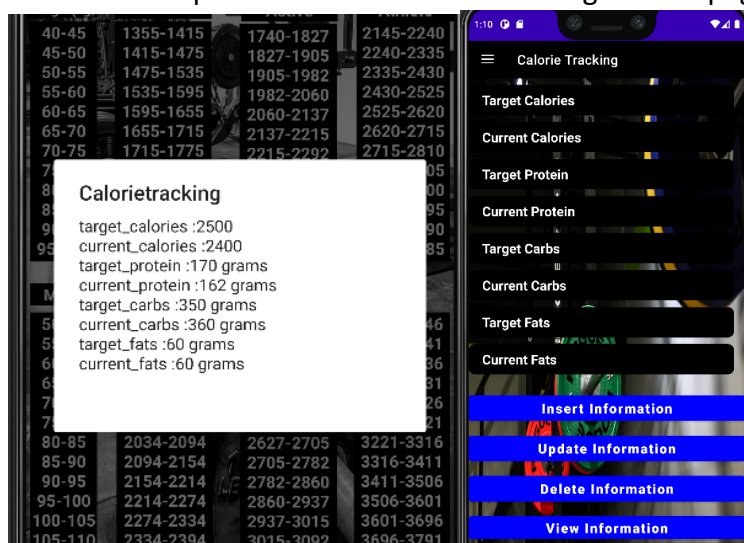
The design of this page is based on a linear layout at the top with 2 relative layouts which both contain card views for the age and weight which also include 2 text fields for the age and the age number for the first card view, For the second card view there's also 2 text fields for the weight and then the weight number. Both linear layouts contain 2 buttons aswell to increase and decrease the age and weight value. In the first relative layout we have an adjustable seekbar which determines the height value in cm by sliding it left to decrease and right to increase the value. We also have a textfield to outline that this relative layout is for Height.

In the second relative layout we have a gender switch for the user to pick their gender. Lastly we have a calculate button which the user clicks once all the information is submitted. The methods to increase & decrease Age and Weight are created in the java class. Based on the user input the user can either be underweight, normalweight or overweight which is all determined by the users BMI (Body Mass Index) score which is calculated by a method. Then another method is made to classify the user into whichever category that I've named based

on their BMI score. Once the information is submitted the user is informed with a popup which weight class they are and then with the use of a timer and intent after 2 seconds of that message popup they are brought to another page which is filled with advice for that particular weight class. Advice for other weight classes can be viewed from that page too and also the user's current nutrition plan and the option to make a new nutrition plan based on the new advice the user has received from the page they have been brought to. If the user is overweight they'll be brought to the cutting advice page, if normal weight then brought to maintaining weight advice and if underweight they'll be brought to the bulking advice page. Each of these pages will have tables of recommended calorie consumption based on each weight and activity level.

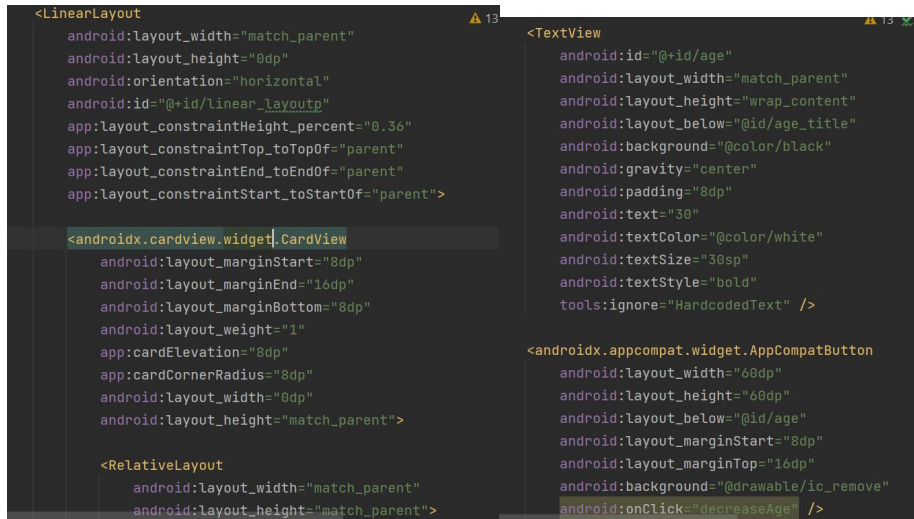


Below I will show what happens when both buttons are clicked. When the view current nutrition plan button is clicked the user's current nutrition plan comes up. When the create new nutrition plan is clicked the user is brought to the page to create a new nutrition plan.

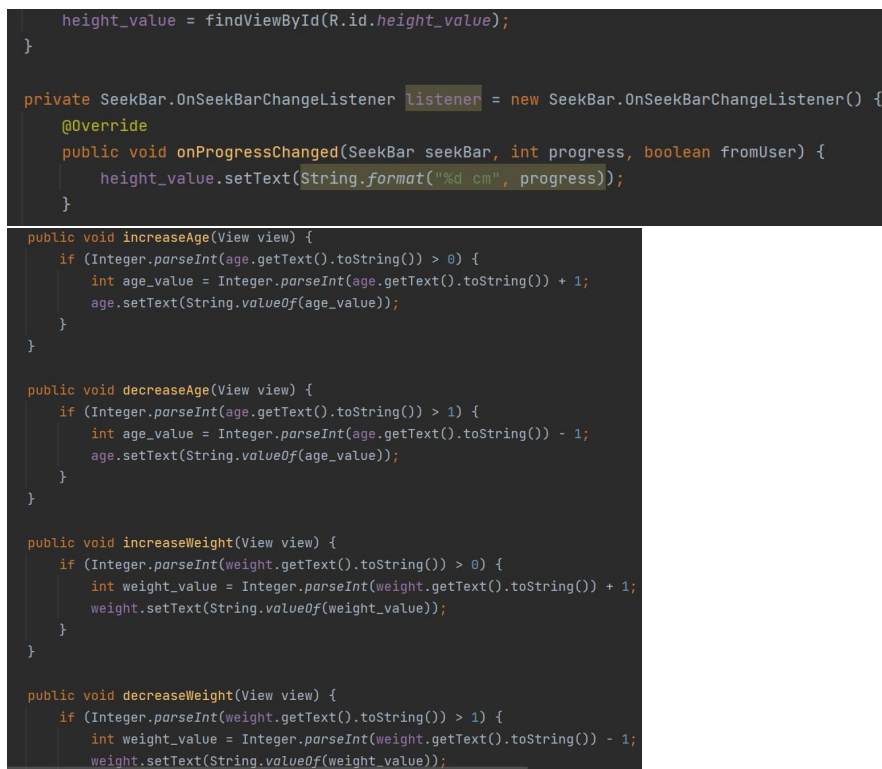


Code explanation

Firstly I have shown the design of the page in the xml file both card views within the linear layout have the same structure so I've shown just the one. As I have already said it's 2 relative layouts within a linearlayout that holds them. The relative layouts contain a cardview each with 2 text fields for the title and it's number value below and 2 buttons to decrease and increase the number value.



Now the java code, we initiate the seekBar in order for the height value to be able to be decreased and increased according to the users height input. Then below are the methods to increase & decrease both Weight and Age. For increaseing the value is increased by 1 if the input is greater than 0 and for decrease it decreases by 1 if the input is greater than 1.



Next is the method to set the BMI which we use a formula which we take in the weight and divide it by the height squared.

```
public void showResult(View view){
    int get_age = Integer.parseInt(age.getText().toString());
    int weight_value = Integer.parseInt(weight.getText().toString());
    double get_height = (double)seekBar.getProgress() / 100;

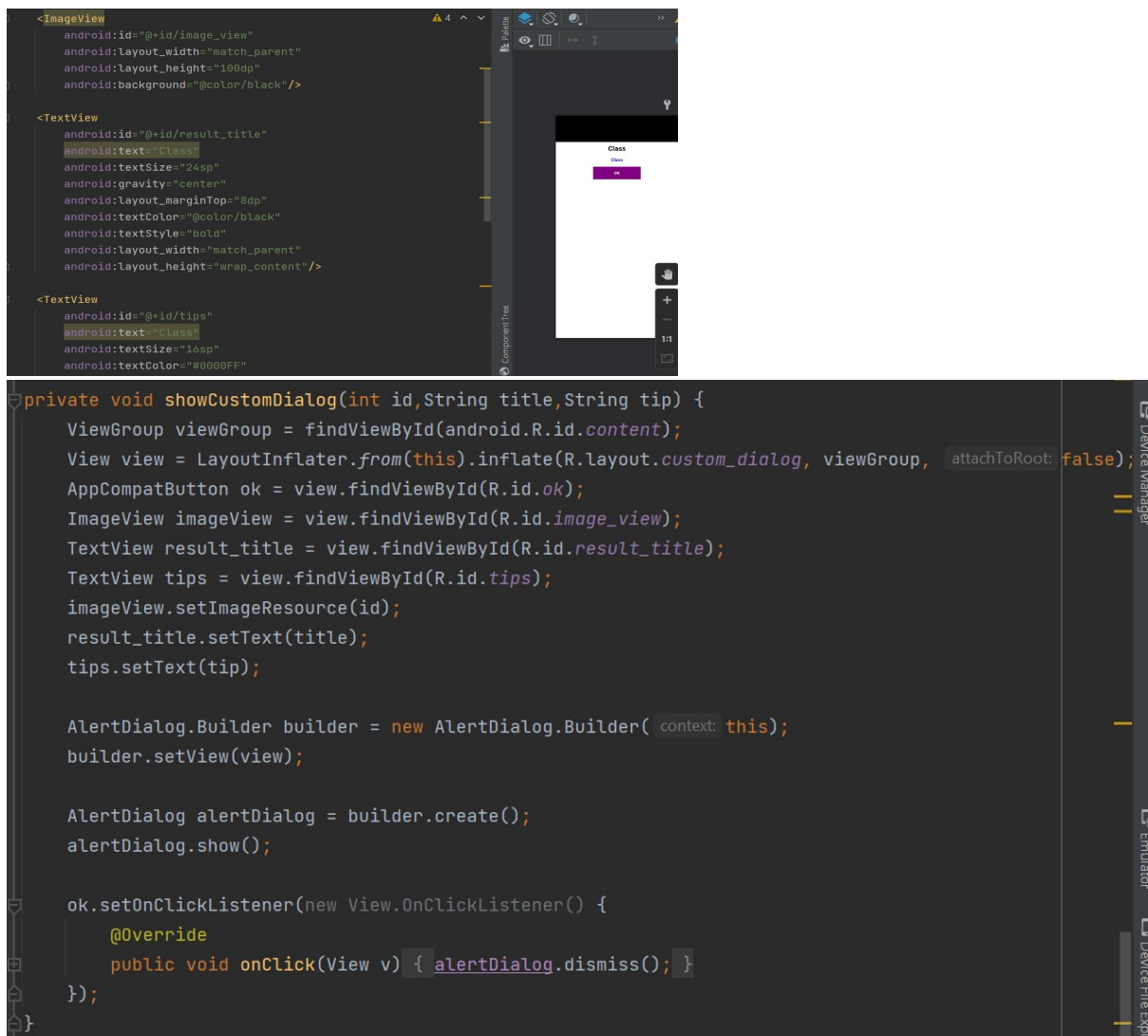
    int bmi = weight_value / (int)(get_height * get_height) ;

    showBMI(bmi);
}
```

Next is the method to classify the user within of the 3 weight categories depending on their BMI score. This is my K-Nearest Algorithm implementation I have mentioned in my proposal in order to make my project more complex. If below 18.5 they're underweight, If between 18.5 and 24.9 they're normal weight and if over 24.9 they're overweight and depending on the result they will be brought to an advice page for either cutting(overweight),maintaining(normal weight) or bulking(underweight). A timer method has been used to open the advice page using Intent, so a new activity is launched after a 2 second delay once a message pops up with the users weight result.

```
private void showBMI(int bmi) {
    if (bmi < 18.5) {
        showCustomDialog(R.drawable.underweight, title: "Underweight", tip: "Get into a calorie surplus and",
        timer = new Timer();
        timer.schedule(() -> {
            Intent i = new Intent( packageContext: WeightProgressActivity.this, BulkingAdviceActivity.class);
            startActivity(i);
            finish();
        }, delay: 2000);
    } else if (bmi > 18.5 && bmi < 24.9){
        showCustomDialog(R.drawable.normalweight, title: "Normalweight", tip: "You are at a healthy weight,
        timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                Intent i = new Intent( packageContext: WeightProgressActivity.this, MaintainAdviceActivity.class);
                startActivity(i);
                finish();
            }
        }, delay: 2000);
    } else {
        showCustomDialog(R.drawable.overweight, title: "Overweight", tip: "Get into a calorie deficit and l
        timer = new Timer();
        timer.schedule(() -> {
```

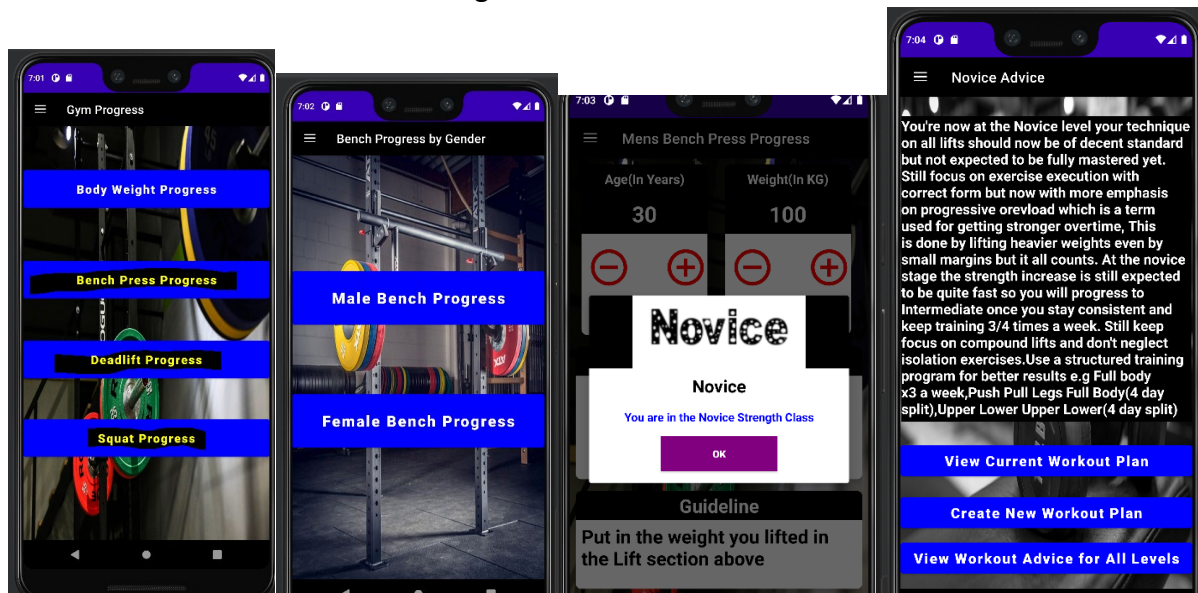
The code below is responsible for the message popup of what weight class the user is based on their BMI score. A custom dialog was made with a title,tip and an image to let the user know what weight class they are.



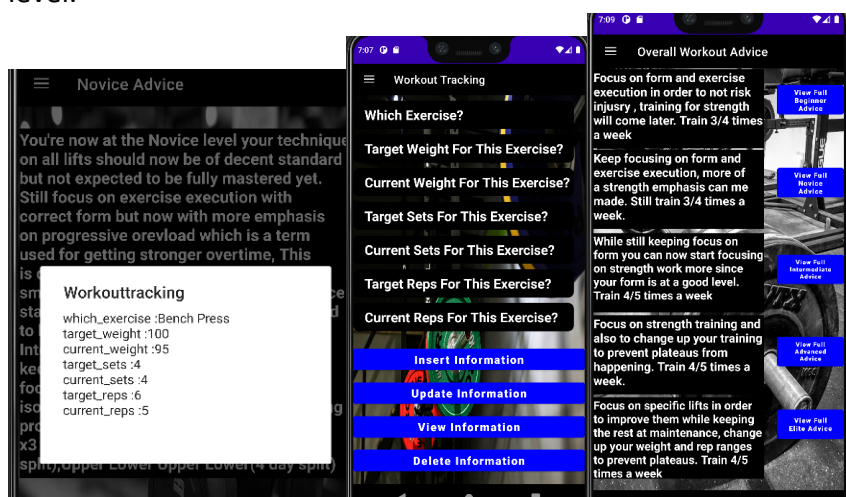
Workout Progress(Squat ,Bench and Deadlift) screenshots and code explanation

This feature is majorly similar to the body weight progress with a few differences which I will capture with screenshots and code explanation but the parts that are the same I won't cover as I have already covered them in the body weight progress. The structure of the page is the same as it's a linear layout at the top that contain 2 relative layouts with 2 card views each that contain text fields for the title and number value of e.g age below and 2 buttons to either increase or decrease the value. The major difference in the design that the first relative layout below asks for the lift(how much kg the user has lifted) instead of their height and then below A guideline is written instead of the user choosing their gender because they choose that on the page previous to this one. A calculate button is at the bottom of the page for the user to submit their information and then based on their lift they will be put into one of five strength categories which are either beginner, novice, intermediate, advanced or elite. Before accessing the page that calculates the progress the user has to choose for which lift they're choosing and then once the lift is chosen they've to choose if to pick for male or female strength standards as they are different since males are stronger. Once the progress is submitted and calculated a popup message using the custom dialog once again comes up telling the user their strength level and from there with the use

of a timer method and intent the user is brought to a new activity which is the advice for their strength level in order to improve and go a level above. From that page the user can also view their current workout, submit a new workout plan based on the advice and also view workout advice for other strength levels.



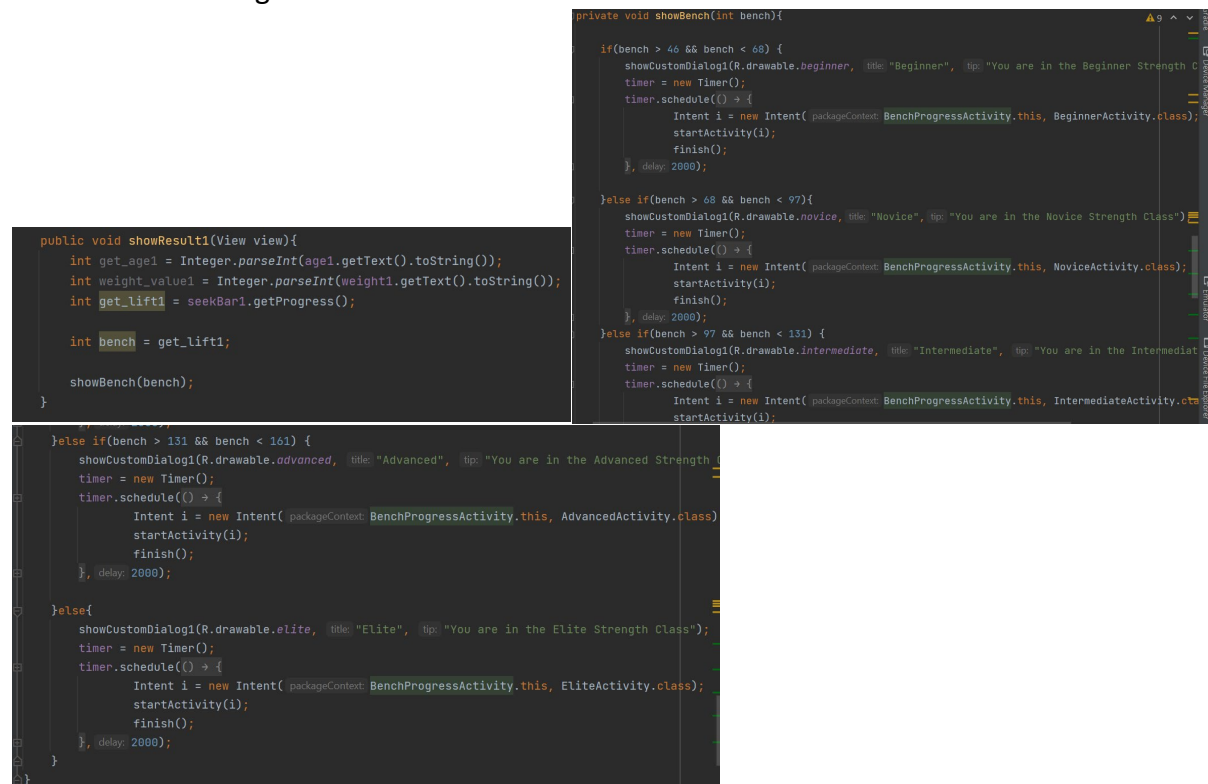
When the view current workout button is clicked you will be able to view your current workout as seen below. When you press the button to create a new workout you will be brought to the page to submit new workout information(workout tracking). When the button for workout advice for all levels is clicked you will be brought to the pagw with summarised advice for each strength level and beside the summarised advice you will be able to click a button to bring you to a page with more detailed advice for that strength level.



Code explanation

The xml page has the same structure as the body weight progress so that ahs already been explained. In the java code the methods for the seekBar, increase & decrease Age and Weight are the same as the methods in the body weight progress. The custom dialog method works the same way for the message popup to show the user their strength level.

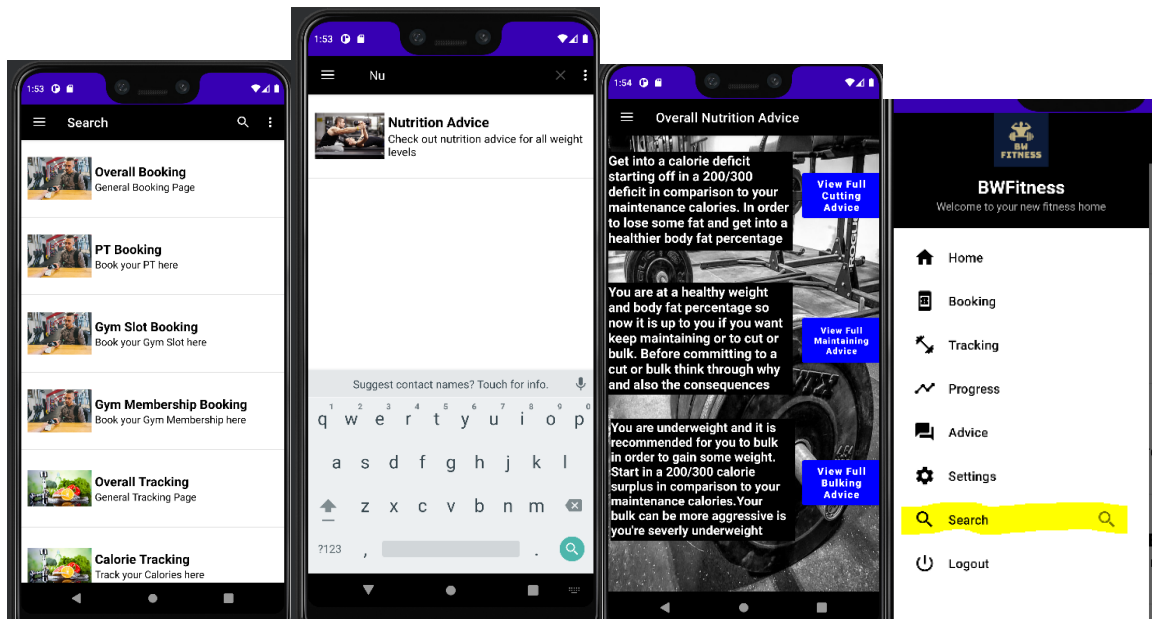
The only major differences is the method to calculate the strength level and then the method to classify the user within one of 5 strength levels which is also my implementation of the K-Nearest algorithm.



The K nearest algorithm implementations are also involved in the deadlift and squat progress for both males and lifts with the only difference being the weight ranges since the weight lifted differs for both lifts but the structure is the same the input is took from the user they're put in a strength class a message comes up letting the user know what class they're in and then they're brought to the advice page for that strength class.

Search Functionality screenshots and code explanation

The search functionality was added as a late addition to make the app more creative and add to the complexity of creating this application, it wasn't initially listed in the project proposal. The search functionality is made to make the life of the user easier when using the app. It allows the user to search for any feature within the app which are all displayed in the first screenshot on left below, all the user has to do is search letters or a word associated with the page they want to access and it will come up as a suggestion as seen in the middle screenshot below. When the user clicks on the page they will be brought to the activity which displays that page. The search function can be accessed by the user at anytime of using the app since it is a part of the navigation menu which is linked to the whole application.



Code explanation

A model and an adapter class have to be made in order for the objects from the model to be accessed by an array list in the adapter class which will allow the possibility of searching for any feature within the app once all the if & else statements are made to include every feature in its method. A viewholder is created within the adapter to supply the view for the search functionality as well as a layout inflater and context. The row which displays the information is made as an xml file and it displays an image, title and description which are all written out for each feature in the java class for the search functionality.

```
public class ListViewAdapter extends BaseAdapter {
    Context mContext;
    LayoutInflater inflater;
    List<Model> modelList;
    ArrayList<Model> arrayList;

    public ListViewAdapter(Context context, List<Model> modelList){
        this.mContext = context;
        this.modelList = modelList;
        inflater = LayoutInflater.from(mContext);
        this.arrayList = new ArrayList<Model>();
        this.arrayList.addAll(modelList);
    }

    public class ViewHolder{
        TextView mTitleTv, mDescTv;
        ImageView mIconIV;
    }

    @Override
    public View getView(int position, View view, ViewGroup parent) {
        ViewHolder holder;
        if (view==null){
            holder = new ViewHolder();
            view = inflater.inflate(R.layout.row, root null);

            holder.mTitleTv = view.findViewById(R.id.mainTitle);
            holder.mDescTv = view.findViewById(R.id.mainDesc);
            holder.mIconIV = view.findViewById(R.id.mainIcon);

            view.setTag(holder);
        }
        else{
            holder = (ViewHolder) view.getTag();
        }
        holder.mTitleTv.setText(modelList.get(position).getTitle());
        holder.mDescTv.setText(modelList.get(position).getDesc());
        holder.mIconIV.setImageResource(modelList.get(position).getIcon());

        view.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(modelList.get(position).getTitle().equals("Overall Booking")){
                    Intent intent = new Intent(mContext, BookingActivity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("PT Booking")) {
                    Intent intent = new Intent(mContext, PTActivity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("Home Page")){
                    Intent intent = new Intent(mContext, HomePageActivity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("Gym Membership Booking")){
                    Intent intent = new Intent(mContext, MembershipActivity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("Gym Slot Booking")){
                    Intent intent = new Intent(mContext, GymSlotActivity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("Overall Tracking")){
                    Intent intent = new Intent(mContext, TrackingActivity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("Calorie Tracking")){
                    Intent intent = new Intent(mContext, CalorieActivity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("Workout Tracking")){
                    Intent intent = new Intent(mContext, WorkoutActivity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("Payment")){
                    Intent intent = new Intent(mContext, Payment_Activity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("Settings")){
                    Intent intent = new Intent(mContext, SettingsActivity.class);
                    mContext.startActivity(intent);
                }
                else if (modelList.get(position).getTitle().equals("Progress")){
                    Intent intent = new Intent(mContext, GymProgressActivity.class);
                    mContext.startActivity(intent);
                }
            }
        });
    }
}
```

```

}else if
(modellist.get(position).getTitle().equals("Workout Advice")){
    Intent intent = new Intent(mContext, OverallActivity.class);
    mContext.startActivity(intent);
}else if
(modellist.get(position).getTitle().equals("Nutrition Advice")){
    Intent intent = new Intent(mContext, OverallNutritionActivity.class);
    mContext.startActivity(intent);
}else if
(modellist.get(position).getTitle().equals("Bench Press Progress")){
    Intent intent = new Intent(mContext, BenchProgressActivity.class);
    mContext.startActivity(intent);
}else if
(modellist.get(position).getTitle().equals("DeadLift Progress")){
    Intent intent = new Intent(mContext, DeadLiftProgressActivity.class);
    mContext.startActivity(intent);
}else if
(modellist.get(position).getTitle().equals("Squat Progress")){
    Intent intent = new Intent(mContext, SquatProgressActivity.class);
    mContext.startActivity(intent);
}else if
(modellist.get(position).getTitle().equals("Body Weight Progress")) {
    Intent intent = new Intent(mContext, WeightProgressActivity.class);
    mContext.startActivity(intent);
}
}

```

Below is the java class which sets the image,title and description for each row that will display a feature/activity

```

title = new String[]{"Overall Booking","Home Page", "PT Booking", "Gym Slot Booking", "Gym Memb
"Overall Tracking", "Calorie Tracking", "Workout Tracking", "Progress",
"Settings", "Payment", "Workout Advice", "Nutrition Advice",
"Bench Press Progress", "DeadLift Progress", "Squat Progress", "Body Weight Progress"};
description = new String[]{"General Booking Page", "Access the home page here", "Book your PT he
, "General Tracking Page", "Track your Calories here", "Track your Workouts here", "Tra
, "Settings page", "Payment page", "Check out Workout Advice for all experience levels"
"Check your Bench Press Progress", "Check your DeadLift Progress", "Check your Squat Progress", "C
icon = new int[]{R.drawable.gym_pay, R.drawable.hhhh, R.drawable.gym_pay, R.drawable.gym_pay, R.
R.drawable.gym_track, R.drawable.gym_track, R.drawable.pt,
R.drawable.gym_settings, R.drawable.gym_pay, R.drawable.pt, R.drawable.pt,
R.drawable.bench, R.drawable.dl, R.drawable.squat, R.drawable.bmi};

listView = findViewById(R.id.listView);

for (int i = 0; i < title.length; i++) {
    Model model = new Model(title[i], description[i], icon[i]);
    arrayList.add(model);
}

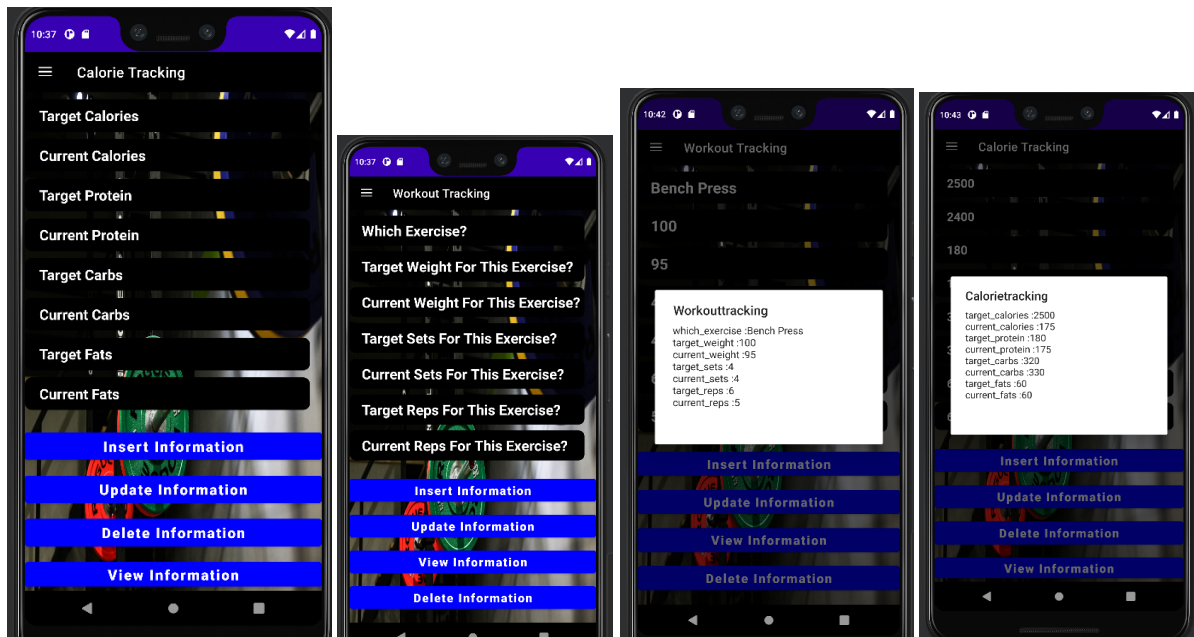
adapter = new ListViewAdapter( context: this, arrayList);

listView.setAdapter(adapter);
}

```

Calorie and Workout tracking screenshots and code explanation

With the calorie tracking the user will be able to track their nutrition so their target calories vs their current calories, target protein vs current protein, target carbs vs current carbs and target fats vs current fats. The workout tracking works in the same way but except with workout information so the user will be asked to input their exercise, target weight vs current weight, target sets vs current sets and target reps vs current reps. Both pages are a series of Edit Text fields which the user puts information into and submits by clicking one of the 4 buttons underneath which are the user will be able to insert , view , update and delete information for both pages. Below on the right I've shown an example of information being inserted into both pages and viewing the information that was input.



Code Explanation

Both pages have the same structure and fundamentals so I will only cover the workout tracking to show how both of them work. As I've said the user can insert, view, update and delete information and this is made possible by methods in the Java classes and SQL in the DBHelper class

Text fields and Buttons

```
ActionBar actionBar = getSupportActionBar();

which_exercise = findViewById(R.id.which_exercise);
target_weight = findViewById(R.id.target_weight);
current_weight = findViewById(R.id.current_weight);
target_sets = findViewById(R.id.target_sets);
current_sets = findViewById(R.id.current_sets);
target_reps = findViewById(R.id.target_reps);
current_reps = findViewById(R.id.current_reps);

insert_workout = findViewById(R.id.insert_workout);
update_workout = findViewById(R.id.update_workout);
view_workout = findViewById(R.id.view_workout);
delete_workout = findViewById(R.id.delete_workout);

myDB = new DBHelper( context: this);
```

Insert: Here the text would be received from the EditText fields and the method from the DB Helper class would insert the data into the pt table into the database. If successful a new entry is inserted but if not nothing is inserted into the database

```

insert_workout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String whichexerciseTXT = which_exercise.getText().toString();
        String targetweightTXT = target_weight.getText().toString();
        String currentweightTXT = current_weight.getText().toString();
        String targetsetsTXT = target_sets.getText().toString();
        String currentsetsTXT = current_sets.getText().toString();
        String targetrepsTXT = target_reps.getText().toString();
        String currentrepsTXT = current_reps.getText().toString();

        Boolean checkworkoutdata = myDB.insertworkoutdata( whichexerciseTXT , targetweightTXT , currentweightTXT , targetsetsTXT , currentsetsTXT , targetrepsTXT , currentrepsTXT );
        if(checkworkoutdata==true) {
            Toast.makeText( context: WorkoutActivity.this, text: "New Entry Inserted", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText( context: WorkoutActivity.this, text: "Entry Not Inserted", Toast.LENGTH_SHORT).show();
        }
    }
});

public Boolean insertworkoutdata(String which_exercise , String target_weight , String current_weight , String target_sets , String current_sets , String target_reps , String current_reps )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contents = new ContentValues();
    contents.put(" which_exercise " , which_exercise );
    contents.put(" target_weight " , target_weight );
    contents.put(" current_weight " , current_weight );
    contents.put(" target_sets " , target_sets );
    contents.put(" current_sets " , current_sets );
    contents.put(" target_reps " , target_reps );
    contents.put(" current_reps " , current_reps );

    long result = myDB.insert( table: " Workouttracking " , nullColumnHack: null, contents);
    if(result==-1){
        return false;
    }
    else {
        return true;
    }
}

```

Update: Here the text would be received from the EditText fields and the method from the DB Helper class would update the data into the pt table into the database

```

update_workout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String whichexerciseTXT = which_exercise.getText().toString();
        String targetweightTXT = target_weight.getText().toString();
        String currentweightTXT = current_weight.getText().toString();
        String targetsetsTXT = target_sets.getText().toString();
        String currentsetsTXT = current_sets.getText().toString();
        String targetrepsTXT = target_reps.getText().toString();
        String currentrepsTXT = current_reps.getText().toString();

        Boolean checkupdateworkout = myDB.updateworkoutdata( whichexerciseTXT , targetweightTXT , currentweightTXT , targetsetsTXT , currentsetsTXT , targetrepsTXT , currentrepsTXT );
        if(checkupdateworkout==true) {
            Toast.makeText( context: WorkoutActivity.this, text: "Workout Updated", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText( context: WorkoutActivity.this, text: "Workout Did Not Update", Toast.LENGTH_SHORT).show();
        }
    }
});

```

```

public Boolean updateworkoutdata( String which_exercise , String target_weight , String current_weight )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues content = new ContentValues();
    content.put(" which_exercise ", which_exercise );
    content.put(" target_weight ", target_weight );
    content.put(" current_weight ", current_weight );
    content.put(" target_sets ", target_sets);
    content.put(" current_sets ", current_sets);
    content.put(" target_reps ", target_reps);
    content.put(" current_reps ", current_reps);
    Cursor cursor = myDB.rawQuery( sql: "Select * from Workouttracking where which_exercise=?", new String[] { which_exercise } );
    if(cursor.getCount()>0)
    {
        long result = myDB.update( table: " Workouttracking ", content, whereClause: " which_exercise=?", cursor );
        if (result == -1) {
            return false;
        } else {
            return true;
        }
    }else{
        return false;
    }
}

```

Delete: Here only the primary key passed in and the method from the DB Helper class would delete the data from the pt table in the database by only using the primary key from that table.

```

public Boolean deleteworkoutdata( String which_exercise )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from Workouttracking where which_exercise=?", new String[] { which_exercise } );
    if(cursor.getCount()>0)
    {
        long result = myDB.delete( table: " Workouttracking ", whereClause: " which_exercise=?", cursor );
        if (result == -1) {
            return false;
        } else {
            return true;
        }
    }else{
        return false;
    }
}

delete_workout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String whichexercisetxt = which_exercise.getText().toString();

        Boolean checkdeleteworkoutdata = myDB.deleteworkoutdata( whichexercisetxt );
        if(checkdeleteworkoutdata==true) {
            Toast.makeText( context: WorkoutActivity.this, text: "Entry Deleted", Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: WorkoutActivity.this, text: "Entry Did Not Delete", Toast.LENGTH_SHORT).show();
    }
});

```

View: Here the text that has been input is received and is then viewed along with a title beside it and the method from the DB Helper class will view the data from the table in the database. If no information has been inserted then no information can be viewed so then the “No Entry Found” message will come up. Titles beside the text input have been written

to print out the value e.g which exercise: bench press is then printed out in the app when the information is viewed

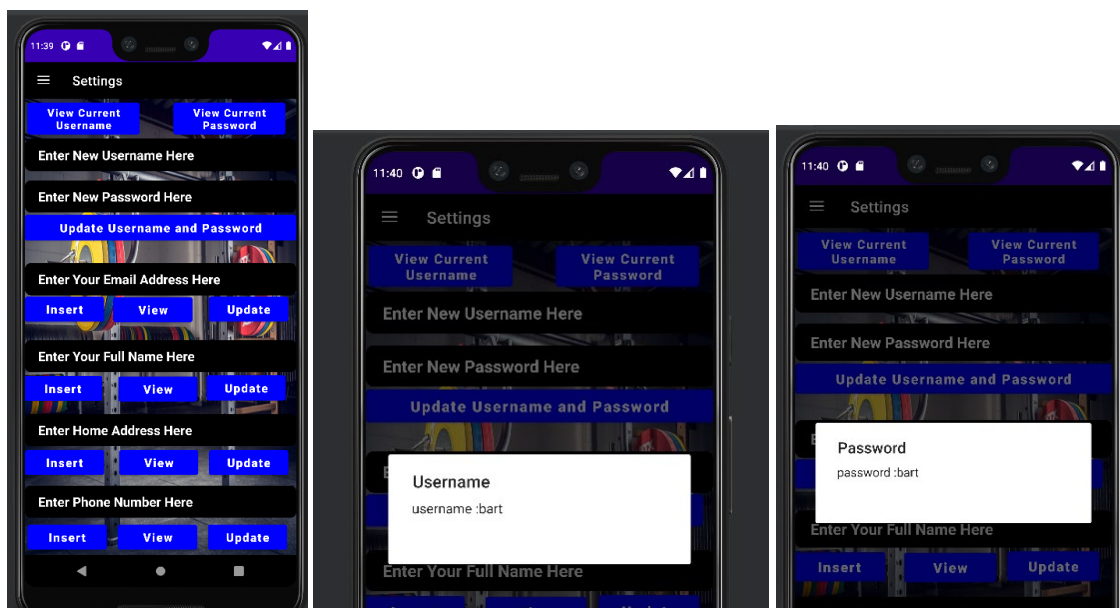
```
view_workout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Cursor res = myDB.viewworkoutdata();
        if(res.getCount()==0){
            Toast.makeText( context: WorkoutActivity.this, text: "No Entry Found",Toast.LENGTH_SHORT).show();
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext()){
            buffer.append(" which_exercise :"+res.getString( columnIndex: 0)+"\n");
            buffer.append(" target_weight :"+res.getString( columnIndex: 1)+"\n");
            buffer.append(" current_weight :"+res.getString( columnIndex: 2)+"\n");
            buffer.append(" target_sets :"+res.getString( columnIndex: 3)+"\n");
            buffer.append(" current_sets :"+res.getString( columnIndex: 4)+"\n");
            buffer.append(" target_reps :"+res.getString( columnIndex: 5)+"\n");
            buffer.append(" current_reps :"+res.getString( columnIndex: 6)+"\n\n");
        }

        AlertDialog.Builder builder = new AlertDialog.Builder( context: WorkoutActivity.this);
        builder.setCancelable(true);
        builder.setTitle(" Workouttracking ");
        builder.setMessage(buffer.toString());
        builder.show();
    }
})

public Cursor viewworkoutdata()
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: " Select * from Workouttracking ", selectionArgs: null);
    return cursor;
}
```

Settings page screenshots and code explanation

In this page the user is able to update existing login credentials if they want to change them, they are able to view their current credentials as seen below before they decide to update them. Then below the user will insert their email, full name, home address and phone number. They will also be able to view these pieces of information and update them if they need to.



View Username and Password:

Both username and password use the same view data method from DB helper. In order to show the username we choose the columnIndex 0 which is the row that contains the username information and we print it out to be shown when the view username button is clicked. To show the password we choose columnIndex 1 which is the row for the password information and will also be printed out when the user clicks to view their current password.

```
viewusername.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Cursor res = myDB.viewdata();
        if(res.getCount()==0){
            Toast.makeText(context, SettingsActivity.this, text: "No Entry Found", Toast.LENGTH_SHORT);
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext()){
            buffer.append(" Username :"+res.getString( columnIndex 0)+"\n\n");
        }

        AlertDialog.Builder builder = new AlertDialog.Builder( context, SettingsActivity.this);
        builder.setCancelable(true);
        builder.setTitle(" Username ");
        builder.setMessage(buffer.toString());
        builder.show();
    }
});

viewpassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Cursor res = myDB.viewdata();
        if(res.getCount()==0){
            Toast.makeText(context, SettingsActivity.this, text: "No Entry Found", Toast.LENGTH_SHORT);
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext()){
            buffer.append(" Password :"+res.getString( columnIndex 1)+"\n\n");
        }

        AlertDialog.Builder builder = new AlertDialog.Builder( context, SettingsActivity.this);
        builder.setCancelable(true);
        builder.setTitle(" Password ");
        builder.setMessage(buffer.toString());
        builder.show();
    }
});
```

Method that views both Username and Password in DB Helper

```
public Cursor viewdata()
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from users", selectionArgs: null);
    return cursor;
}
```

Update User:

To update we use the same method we used to insert the user credentials when they were initially registering at the start of this whole process of using the app but now their credentials can be changed here in the settings page by filling in the edit text fields for username and password and submitting them. The user can then view the information by clicking the view username and password button to make sure they put in the correct credentials.


```

updateuser.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String usernameTXT = username.getText().toString();
        String passwordTXT = password.getText().toString();

        Boolean checkupdateusername = myDB.insertData( usernameTXT, passwordTXT );
        if(checkupdateusername==true) {
            Toast.makeText( context: SettingsActivity.this, text: "Username and Password Updated", Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: SettingsActivity.this, text: "Username and Password Not Updated", Toast.LENGTH_SHORT).show();
    }
});

```

Insert and Update Email

The methods of insert, view and update work the same way for email, full name, home address and phone number so I will only use the Email address example to show how they all work. We get the text input from an Edit Text field its submitted by a button and inserted into the database by the insert method created in the DB Helper class which is also used for updating the email.

```

insertemail.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String emailTXT = email.getText().toString();

        Boolean checkemaildata = myDB.insertemaildata( emailTXT );
        if(checkemaildata==true) {
            Toast.makeText( context: SettingsActivity.this, text: "Email Inserted", Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: SettingsActivity.this, text: "Email Not Inserted", Toast.LENGTH_SHORT).show();
    }
});

update_email.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String emailTXT = email.getText().toString();

        Boolean checkupdateemaildata = myDB.insertemaildata( emailTXT );
        if(checkupdateemaildata==true) {
            Toast.makeText( context: SettingsActivity.this, text: "Email Has Been Updated", Toast.LENGTH_SHORT).show();
        } else
            Toast.makeText( context: SettingsActivity.this, text: "Email Did Not Update", Toast.LENGTH_SHORT).show();
    }
});

```

```

public Boolean insertemaildata( String email )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues contentslot = new ContentValues();
    contentslot.put(" email " , email );

    long result = myDB.insert( table: " settings ", nullColumnHack: null, contentslot);
    if(result==1){
        return false;
    }
    else
    {
        return true;
    }
}

```

View Email:

The method we use to view the email information is a general view method made for the settings page and we get the email information to be printed out by choosing the first line within the table as seen below the columnIndex chosen is 0 which is the first row of data which is the email and its printed out to the view email button when it's clicked

```

viewemail.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Cursor res = myDB.viewsettingsdata();
        if(res.getCount()==0){
            Toast.makeText( context: SettingsActivity.this, text: "No Entry Found", Toast.LENGTH_SHORT).show();
            return;
        }
        StringBuffer buffer = new StringBuffer();
        while(res.moveToNext()){
            buffer.append(" email :"+res.getString( columnIndex: 0)+"\n\n");
        }

        AlertDialog.Builder builder = new AlertDialog.Builder( context: SettingsActivity.this);
        builder.setCancelable(true);
        builder.setTitle(" Email ");
        builder.setMessage(buffer.toString());
        builder.show();
    }
});

```

Method responsible for viewing email, phone number, home address and full name information

```

public Cursor viewsettingsdata()
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: " Select * from settings ", selectionArgs: null);
    return cursor;
}

```

2.4. Graphical User Interface (GUI)

Registration Prototype

Registration Prototype

A registration form prototype with a black border. It contains five elements: three text fields and two buttons. The text fields are labeled 'User Email Address', 'Password', and 'Repeat/Confirm Password'. The buttons are labeled 'Register' and 'Log into existing account'. To the right of each element is a label in parentheses: '(Text field)' for the first three and '(Button)' for the last two.

User Email Address	(Text field)
Password	(Text field)
Repeat/Confirm Password	(Text field)
Register	(Button)
Log into existing account	(Button)

Login Prototype

Login prototype

A login form prototype with a black border. It contains three elements: two text fields and one button. The text fields are labeled 'User email address' and 'Password'. The button is labeled 'Sign In'. To the right of each element is a label in parentheses: '(Text Field)' for the first two and '(Button)' for the last one.

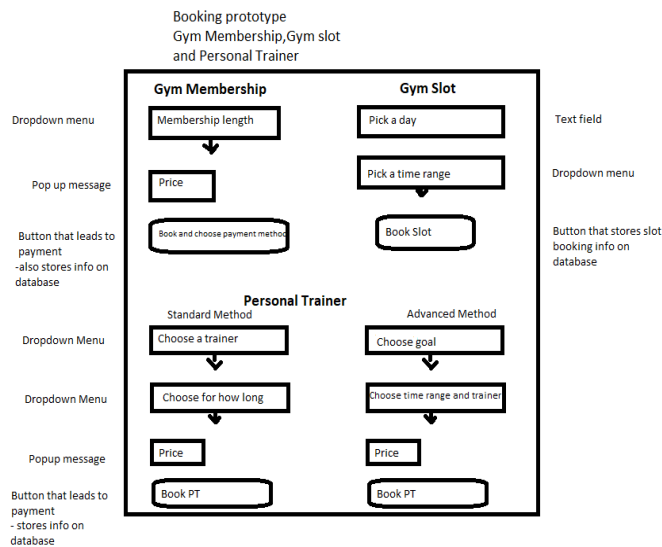
User email address	(Text Field)
Password	(Text field)
Sign In	(Button)

Home Page Prototype

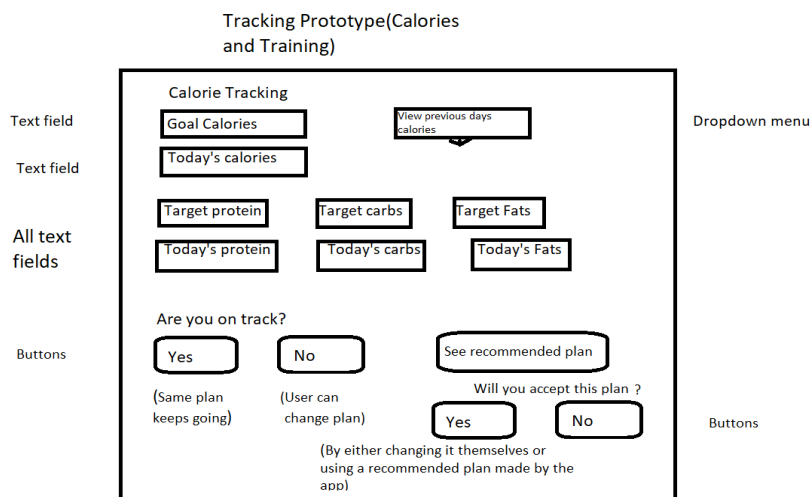
A home page prototype with a black border. It features a menu bar at the top with five items: 'Booking', 'Tracking', 'Gym Starter guide', 'Settings', and 'Payment'. Below the menu bar are three columns of content. The first column is titled 'Need to buy a gym membership?' and contains a placeholder for a picture and a button labeled 'Buy membership here'. The second column is titled 'Need to book a gym slot?' and contains a placeholder for a picture and a button labeled 'Book gym slot here'. The third column is titled 'Are you new and need gym guidance?' and contains a placeholder for a picture and a button labeled 'Access gym guide here'. Below these columns is a section titled 'Below you can keep up to date with your tracking' which contains three placeholders for pictures and buttons labeled 'Calorie Tracking', 'Workout Tracking', and 'Access here'. To the left of the main content area are labels for the menu items and the buttons: 'Menu' for the menu bar, 'Buttons' for the buttons in the first section, and 'Buttons' for the buttons in the second section.

Home Page Prototype (Dropdown Menus)					
Menu	Booking	Tracking	Gym Starter guide	Settings	Payment
	Need to buy a gym membership?				
	(Picture)				
Buttons	Buy membership here				
	Need to book a gym slot?				
	(Picture)				
Buttons	Book gym slot here				
	Are you new and need gym guidance?				
	(Picture)				
Buttons	Access gym guide here				
	Below you can keep up to date with your tracking				
	(Picture)				
Buttons	Calorie Tracking				
	(Picture)				
Buttons	Workout Tracking				
	Settings and payment				
	(Picture)				
Buttons	Access here				

Booking Prototype



Calorie Tracking Prototype



Form for new calorie plan

Time period for new calories

Target Calories

Target Protein

Target Carbs

Target Fats

Target Weight

Submit new plan

(All text fields bar submit new plan that's a button)

Workout Tracking Prototype

Workout Tracking Prototype

Dropdown Menu: Which body part did you train?

Fill in your exercise selection for today's workout

Exercise 1:

Exercise 2:

Exercise 3:

Exercise 4:

Exercise 5:

Exercise 6:

Dropdown Menu: Choose exercise from above

Text Fields: Target Weight, Target Sets, Target Repetitions

Text Fields: Today's Weight, Today's Sets, Today's Repetitions

Buttons: Are you on track? Yes, No, See recommended plan

Buttons: Will you accept this plan? Yes, No

(Same plan keeps going) (User can change plan)

(By either changing it themselves or using a recommended plan made by the app)

Workout Plan Form

All text fields: Body Part, Exercise, Target Weight, Target Sets, Target Repetitions

Button: Submit Plan

Settings Prototype

Settings Prototype

Text fields: Email Address, Password, Gym Membership Length, PT Booked and length, Gym slots booked

Buttons: Change email, Change password, Change membership, Change PT, Change time slots

Button: Submit New Settings

Buttons (when clicked user is brought to a form to input new details)

Payment prototype

Payment prototype

Text Fields

Button that stores credit card details on a database

Text Fields

Button that stores PayPal details on a database

Credit or Debit Card

Credit/Debit Card Number

Expiry date

CVC/CVV (3 digits)

Name on card

Save credit/debit card details

PayPal

Email Address

Password

Login and pay

Testing

I used Test Cases for all my testing for all testing types including Unit and Integration Testing. Unit Testing: whole feature, Integration: Part of a feature

Unit Testing Test Cases

Name of Test Case: Successful Registration

Description of Test Case: Filling in all the credentials (Username, Password and confirm Password) in the registration page in order to make an account in order to be able to use the application

Expected Result of Test Case: Successful Registration and being transferred to the login page after

Actual Result of Test Case: Successful Registration and being transferred to the login page after

```
else
{
    if(pass.equals(repass))
    {
        Boolean usercheckResult = myDB.checkusername(user);
        if(usercheckResult == false)
        {
            Boolean regResult = myDB.insertData(user,pass);
            if(regResult == true){
                Toast.makeText(context, MainActivity.this, text: "Registration Successful", Toast.LENGTH_SHORT);
                Intent intent = new Intent(getApplicationContext(), SignUpActivity.class);
                startActivity(intent);
            }
        }
    }
}
```

Name of Test Case: Successful Login

Description of Test Case: Filling in all the credentials in the Login page(Username and password) from an already registered account in order to login to use the app

Expected Result of Test Case: Successful login and user is transferred to the home page

Actual Result of Test Case: Successful login and user is transferred to the home page

```

else
{
    Boolean result = myDB.checkusernamePassword(user,pass);
    if(result==true){
        Intent intent = new Intent(getApplicationContext(),HomePageActivity.class);
        startActivity(intent);
    }
}

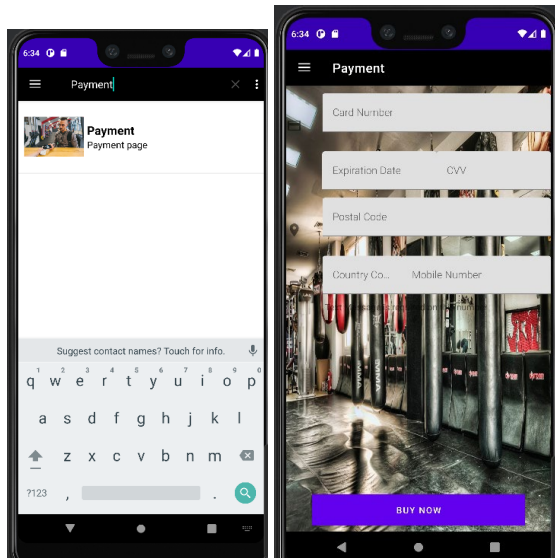
```

Name of Test Case: Successful use of the search functionality

Description of Test Case: Typing in a feature which is in this app in order to see if it comes up as a suggestion and seeing when its clicked does it bring you to that feature.

Expected Result of Test Case: Successful suggestion of the feature once typed in an when clicked user is taken to that feature

Actual Result of Test Case: Success as I typed in Payment and the Payment page came up and when clicked I was taken to the payment page.

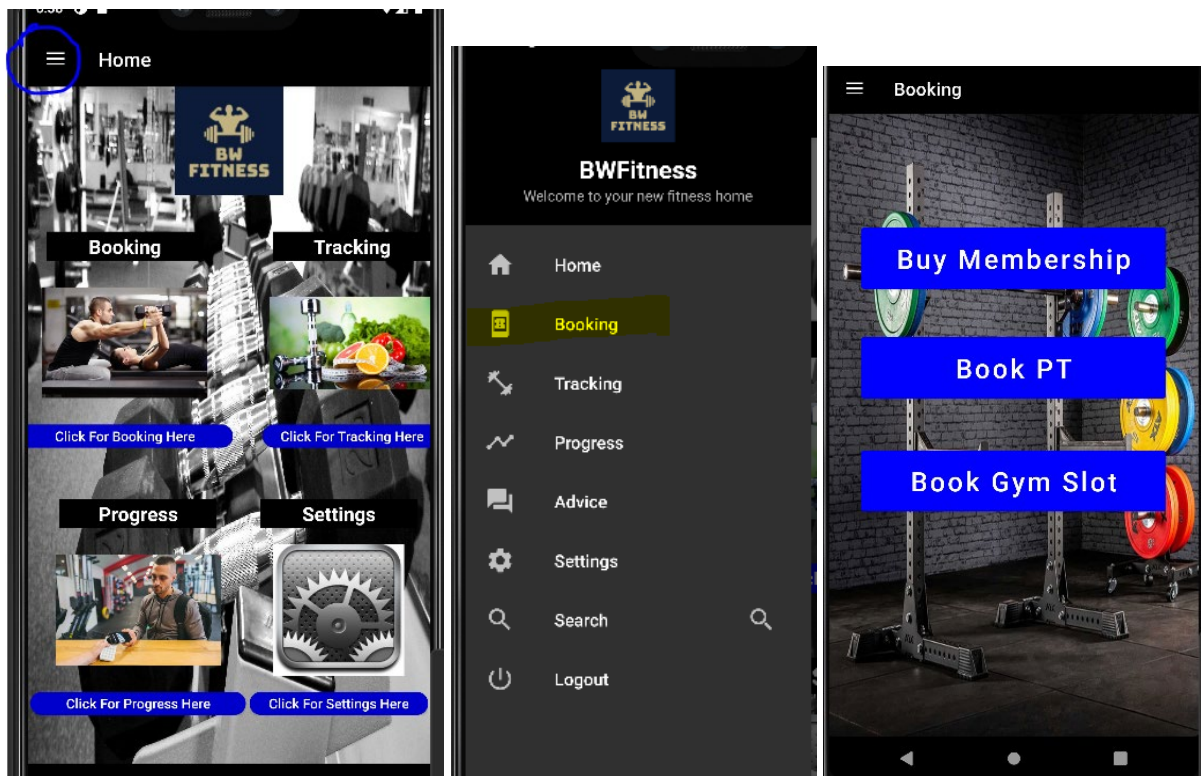


Name of Test Case: Successful use of the navigation menu

Description of Test Case: Clicking the toggle button in the top left corner of any page in order to open the navigation menu and then clicking on a feature and when clicked user is brought to the page of that feature

Expected Result of Test Case: Successful use of the menu, clicking the toggle button opens the menu and when a feature is clicked the user is brought to that feature

Actual Result of Test Case: Success as I opened the menu using the toggle button and I chose to click on the Booking feature and I was taken there when I clicked on it.

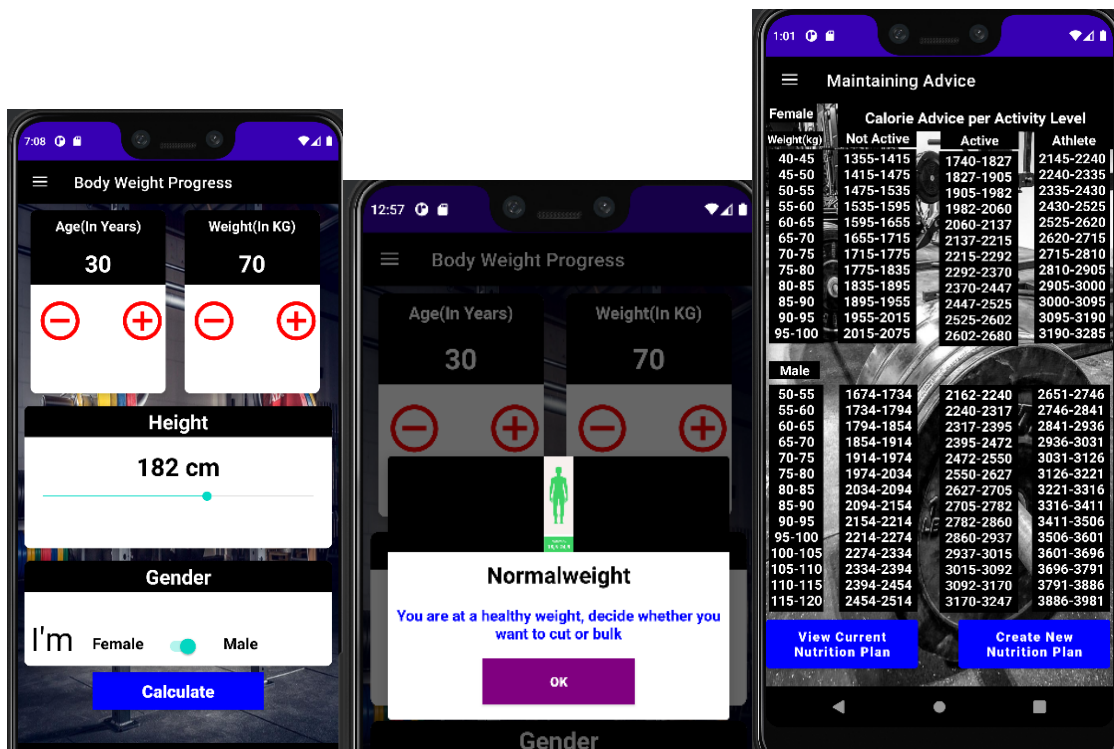


Name of Test Case: Successful use of the body weight progress page

Description of Test Case: Filling in the information for age, weight and height and pressing the calculate button in order for the users BMI to be calculated

Expected Result of Test Case: Successful use of the body weight progress page, information is filled in the calculate button is clicked and a message pops up showing the user whether they're underweight, normal weight or overweight. User is then taken to the advice page to whichever level they are

Actual Result of Test Case: Success as I filled in the required information clicked the calculate button and was shown the normal weight message and was taken to the weight maintaining advice page



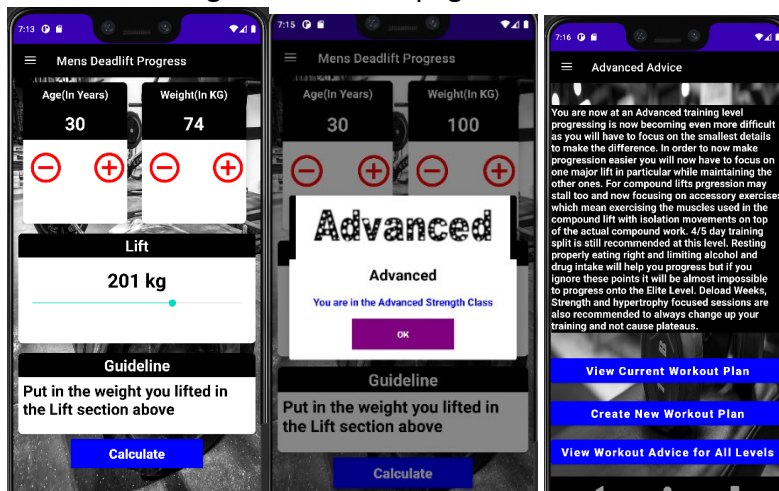
This feature contains Bench,Squat and Deadlift but I will only demonstrate the use case for deadlift as they are all programmed and tested the same way

Name of Test Case: Successful use of the weight lifting progress page, example show on male deadlift progress

Description of Test Case: Filling in the information for age, weight and lift and pressing the calculate button in order for the users strength level

Expected Result of Test Case: Successful use of the weight lifting/ workout progress page, information is filled in the calculate button is clicked and a message pops up showing the user whether they're a Beginner, Novice, Intermediate, Advanced or Elite. User is then taken to the advice page to whichever level they are

Actual Result of Test Case: Success as I filled in the required information clicked the calculate button and was shown the advanced strength class message and was taken to the advanced strength class advice page



Integration Testing Test Cases

Name of Test Case: Failed Registration by not filling in credentials

Description of Test Case: Trying to register without entering any credentials in order to see if the error handling works properly

Expected Result of Test Case: Failed Registration, app asks user to enter all fields

Actual Result of Test Case: Failed Registration, app asks user to enter all fields

```
if(user.equals("") || pass.equals("") || repass.equals(""))
{
    Toast.makeText( context: MainActivity.this, text: "Fill all the fields.",Toast.LENGTH_SHORT).show();
}
```

Name of Test Case: Failed Registration by repeating the wrong password

Description of Test Case: Trying to register by entering the wrong password in the confirm password field in order to see if the error handling works properly

Expected Result of Test Case: Failed Registration, app asks user to enter the correct password in the confirm password field

Actual Result of Test Case: Failed Registration, app asks user to enter the correct password in the confirm password field

```
else
{
    Toast.makeText( context: MainActivity.this, text: "Password not Matching",Toast.LENGTH_SHORT).show();
}
```

Name of Test Case: Failed Registration by trying to register an already registered account

Description of Test Case: Trying to register by entering credentials the user has used to previously register in order to see if the error handling works properly

Expected Result of Test Case: Failed Registration, app tells user account already exists so they can now login

Actual Result of Test Case: Failed Registration, app tells user account already exists so they can now login

```
else
{
    Toast.makeText( context: MainActivity.this, text: "User Already Exists.\n Please Sign In",Toast.LENGTH_SHORT).show();
}
```

Name of Test Case: Failed Login by not filling in any credentials

Description of Test Case: Trying to login by not entering any credentials in order to see if the error handling works properly

Expected Result of Test Case: Failed Login, app tells user to fill in all credentials

Actual Result of Test Case: Failed Login, app tells user to fill in all credentials

```
if(user.equals("") || pass.equals("")){
    Toast.makeText( context: SignUpActivity.this, text: "Please Enter Your credentials.",Toast.LENGTH_SHORT).show();
}
else
```

Name of Test Case: Failed Login by putting in credentials that don't match with the registration

Description of Test Case: Trying to login by entering wrong credentials compared to the ones the user used for registration in order to see if the error handling works properly

Expected Result of Test Case: Failed Login, app tells user wrong credentials and asks for the correct credentials to be put in for a successful login

Actual Result of Test Case: Failed Login, app tells user wrong credentials and asks for the correct credentials to be put in for a successful login

```
}  
else  
{  
    Toast.makeText( context: SignUpActivity.this, text: "Invalid Credentials, Please Try Again. ", Toast.  
}
```

This test case covers all of the insert data methods throughout the app since they are all programmed and tested the exact same way

Name of Test Case: Successful insertion of information into the database

Description of Test Case: Filling in information into text fields and submitting the information by pressing the insert information button

Expected Result of Test Case: Successful insertion of information into the database, information can be viewed with the view information functionality

Actual Result of Test Case: At first failed because of a mistake by not putting in one of the text fields that contain information but a second time around it worked. Successful insertion of information into the database, information can be viewed with the view information functionality

Example of an insert information method

```
public Boolean inserttrainerdata(String target_performance , String pt_length , String pt_payment )  
{  
    SQLiteDatabase myDB = this.getWritableDatabase();  
    ContentValues content = new ContentValues();  
    content.put(" target_performance " , target_performance );  
    content.put(" pt_length " , pt_length );  
    content.put(" pt_payment " , pt_payment);  
  
    long result = myDB.insert( table: " pt_table " , nullColumnHack: null, content);  
    if(result==-1){  
        return false;  
    }  
    else  
    {  
        return true;  
    }  
}
```

This test case covers all of the update data methods throughout the app since they are all programmed and tested the exact same way

Name of Test Case: Successful update of information into the database

Description of Test Case: Filling in information into text fields that have already been filled in and inserted into the database but the user has changed their mind and they want to update the previously submitted information. User fills in all the text fields required with information and submits the information by pressing the update information button

Expected Result of Test Case: Successful insertion of information into the database, information can be viewed with the view information functionality

Actual Result of Test Case: At first failed because of a bug with Android studio so second time around I renamed the variable for the contents value in the DB Helper class in the update method wiped the data from the virtual phone ran it again and it worked. The information has been successfully updated in the database.

Example of an update information method

```
public Boolean updatetrainerdata( String target_performance , String pt_length , String pt_payment )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    ContentValues content = new ContentValues();
    content.put(" target_performance ", target_performance );
    content.put(" pt_length ", pt_length );
    content.put(" pt_payment ", pt_payment );

    Cursor cursor = myDB.rawQuery( sql: "Select * from pt_table where target_performance=?",new String[]{});
    if(cursor.getCount()>0)
    {
        long result = myDB.update( table: " pt_table ", content, whereClause: " target_performance=?", new ContentValues());
        if (result == -1) {
            return false;
        } else {
            return true;
        }
    }
    else{
        return false;
    }
}
```

This test case covers all of the view data methods throughout the app since they are all programmed and tested the exact same way

Name of Test Case: Successful viewing of information from the database

Description of Test Case: Pressing the view button in order to view information that has already been inserted into the database

Expected Result of Test Case: Successful viewing of information from the database, information that has previously been inserted or updated by the user can be seen once the view information button is clicked

Actual Result of Test Case: Successful viewing of information from the database, information that has previously been inserted or updated by the user can be seen once the view information button is clicked

Example of a view information method

```
public Cursor viewtrainerdata()
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: " Select * from pt_table ", selectionArgs: null);
    return cursor;
}
```

This test case covers all of the delete data methods throughout the app since they are all programmed and tested the exact same way

Name of Test Case: Successfully deleting information from the database

Description of Test Case: Pressing the delete information button in order to delete information that has already been inserted into the database

Expected Result of Test Case: Successfully deleting information from the database, information that has previously been inserted or updated by the user is now deleted once the delete button is clicked

Actual Result of Test Case: Successfully deleting information from the database, information that has previously been inserted or updated by the user is now deleted once the delete button is clicked

Example of a delete information method

```
public Boolean deletetrainerdata( String target_performance )
{
    SQLiteDatabase myDB = this.getWritableDatabase();
    Cursor cursor = myDB.rawQuery( sql: "Select * from PersonalTrainer where target_performance=?", new
    if(cursor.getCount()>0)
    {
        long result = myDB.delete( table: " pt_table ", whereClause: " target_performance=? ", new String
        if (result == -1) {
            return false;
        } else {
            return true;
        }
    }else{
        return false;
    }
}
```

Name of Test Case: Successful use of Buttons with intent

Description of Test Case: Pressing a button and being brought a different activity

Expected Result of Test Case: Successfully being transferred to a new activity when a button is clicked

Actual Result of Test Case: Successfully being transferred to a new activity when a button is clicked

```
male_bench =(Button) findViewById(R.id.male_bench);
male_bench.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { BenchProgressActivity(); }
});

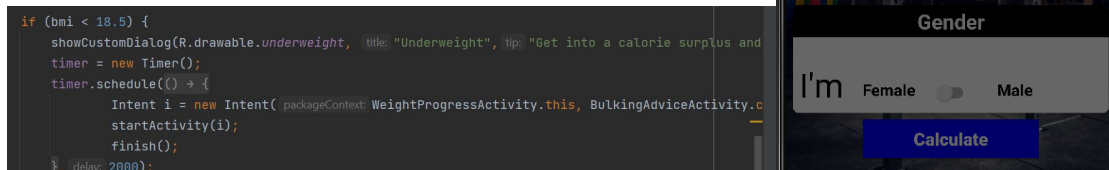
public void BenchProgressActivity(){
    Intent b = new Intent( packageContext: this,BenchProgressActivity.class);
    startActivity(b);
}
```

Name of Test Case: Successful use of the body weight progress page in order to get an underweight user

Description of Test Case: Filling in the information for age, weight and height and pressing the calculate button in order for the user to be underweight, the BMI has to equal less than 18.5

Expected Result of Test Case: If BMI is under 18.5 the user is underweight and is brought to the bulking advice page

Actual Result of Test Case: Success as BMI is under 18.5 and user is brought to the bulking advice page

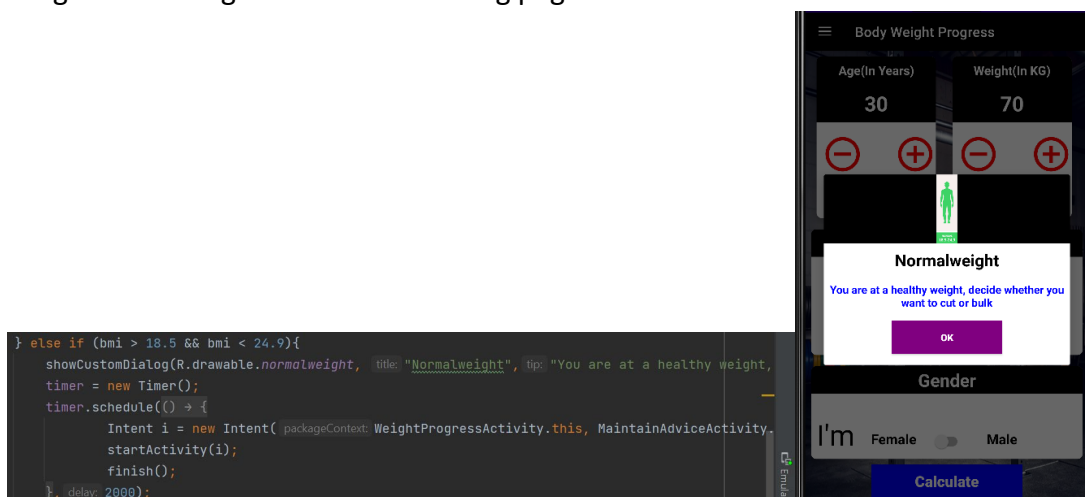


Name of Test Case: Successful use of the body weight progress page in order to get a normal weight user

Description of Test Case: Filling in the information for age, weight and height and pressing the calculate button in order for the user to be normal weight, the BMI has to be between 18.5 and 24.9

Expected Result of Test Case: If BMI is between 18.5 and 24.9 the user is normal weight and is brought to the maintaining advice page

Actual Result of Test Case: Success as BMI is between 18.5 and 24.9 the user is normal weight and brought to the maintaining page



Name of Test Case: Successful use of the body weight progress page in order to get an overweight user

Description of Test Case: Filling in the information for age, weight and height and pressing the calculate button in order for the user to be overweight, the BMI has to be over 24.9

Expected Result of Test Case: If BMI is over 24.9 the user is overweight and is brought to the cutting advice page

Actual Result of Test Case: Success as BMI is over 24.9 the user is overweight and is brought to the cutting advice page



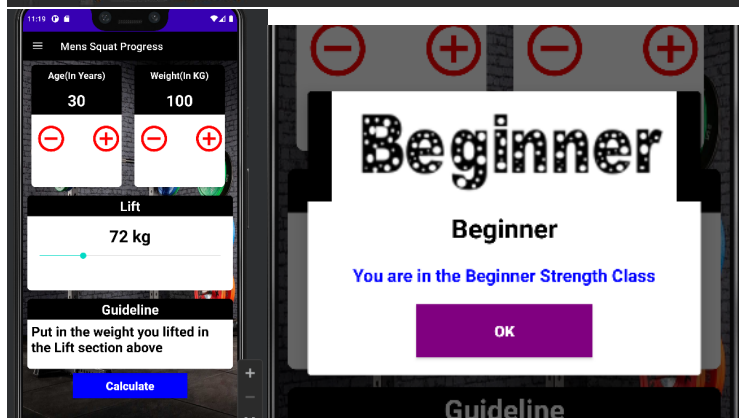
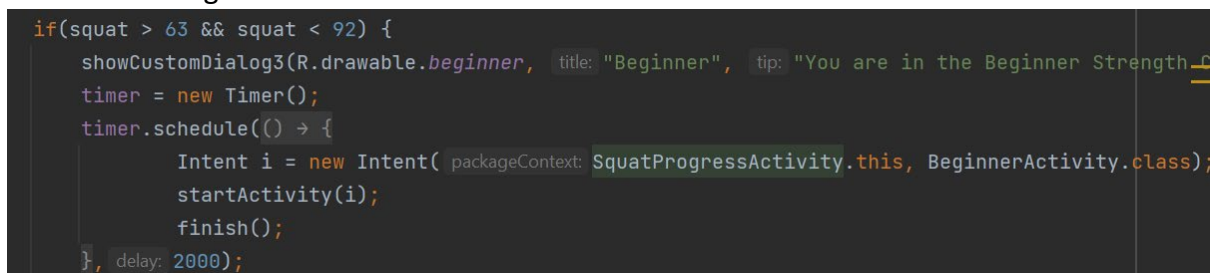
Bench, Squat and Deadlift strength classes all work and are tested the same for both males and females so I will demonstrate the testing only on Squat for Males

Name of Test Case: Successful use of the workout progress page in order to get a user in the beginner strength class

Description of Test Case: Filling in the information for age, weight and lift and pressing the calculate button in order for the user to be a beginner

Expected Result of Test Case: If the squat is between 63kg and 92kg the user is a beginner

Actual Result of Test Case: Success as the squat was 71kg and is between 63kg and 92kg and the user is a beginner

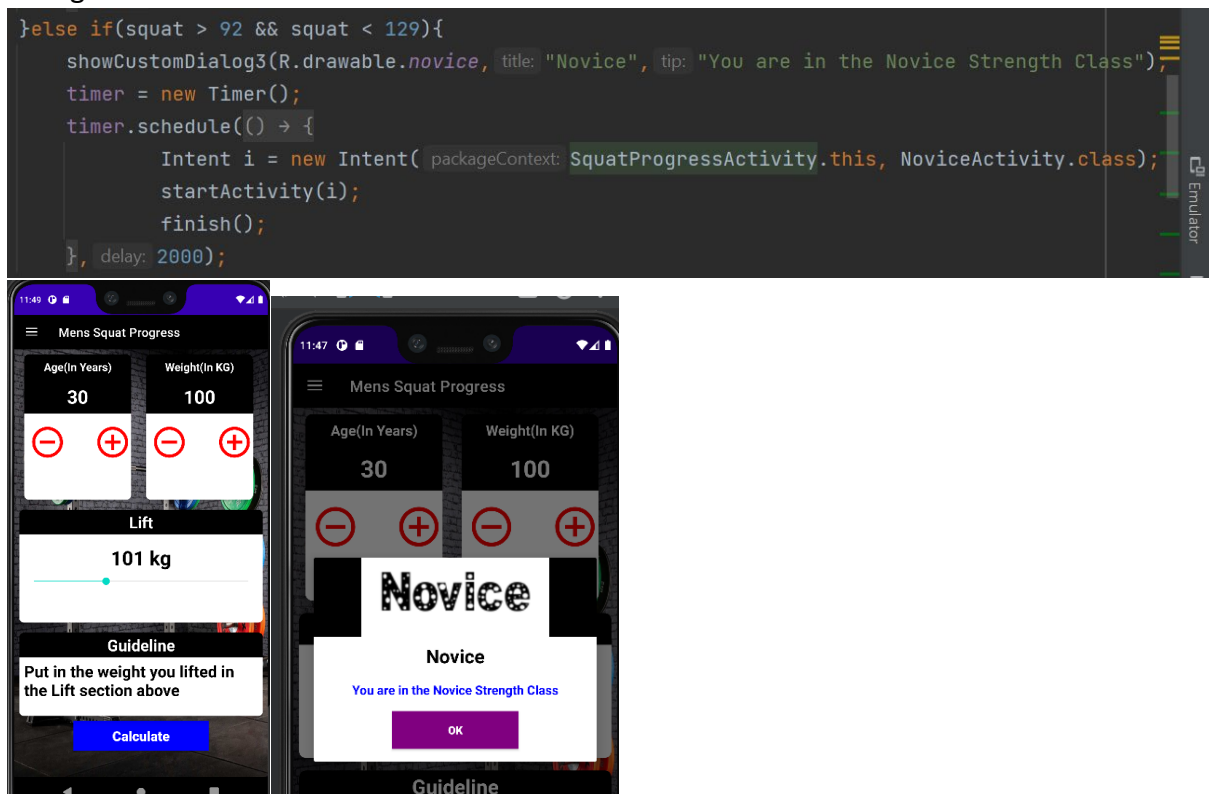


Name of Test Case: Successful use of the workout progress page in order to get a user in the novice strength class

Description of Test Case: Filling in the information for age, weight and lift and pressing the calculate button in order for the user to be a novice

Expected Result of Test Case: If squat is between 92kg and 129kg the user is a novice

Actual Result of Test Case: Success as the value chosen was 101kg so between 92kg and 129kg and the user is a novice

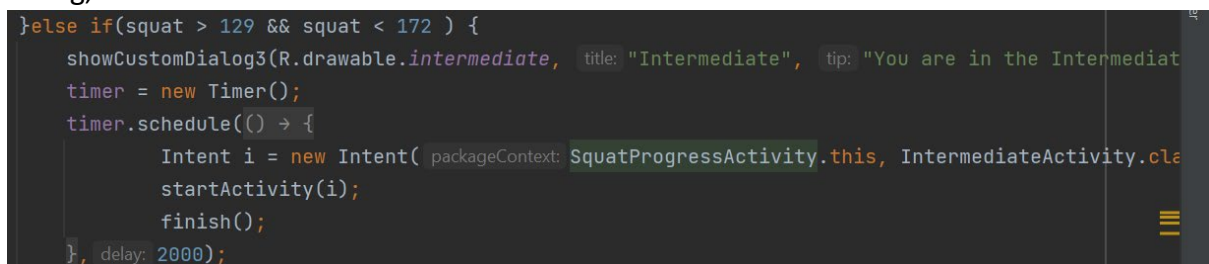


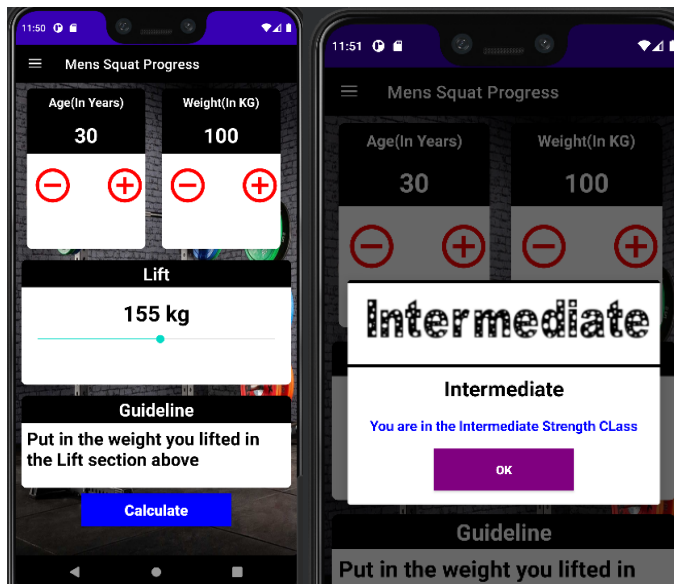
Name of Test Case: Successful use of the workout progress page in order to get a user in the intermediate strength class

Description of Test Case: Filling in the information for age, weight and lift and pressing the calculate button in order for the user to be an intermediate

Expected Result of Test Case: If squat is between 129kg and 172kg the user is an intermediate

Actual Result of Test Case: Success as squat chosen is 155kg and is between 129kg and 172kg, the user is an intermediate





Name of Test Case: Successful use of the workout progress page in order to get a user in the advanced strength class

Description of Test Case: Filling in the information for age, weight and lift and pressing the calculate button in order for the user to be an advanced lifter

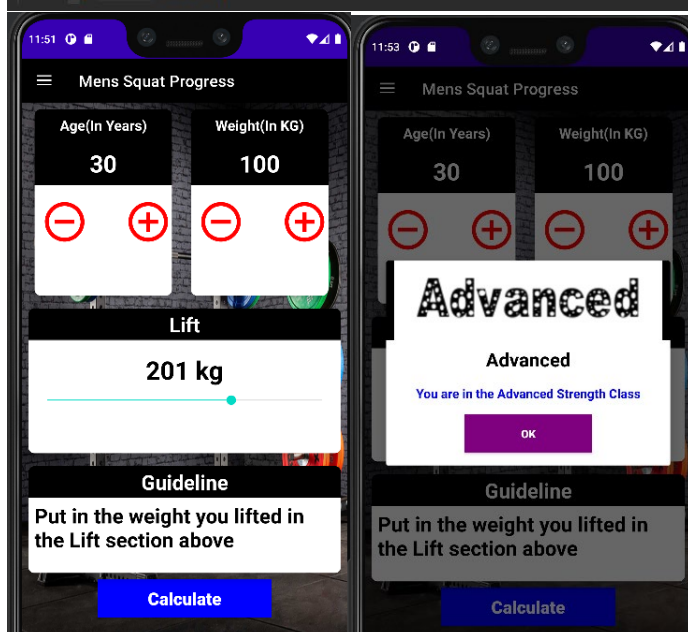
Expected Result of Test Case: If squat is between 172kg and 218kg the user is advanced

Actual Result of Test Case: Success as squat is 201kg and the user is an advanced lifter

```

}else if(squat > 172 && squat < 218) {
    showCustomDialog3(R.drawable.advanced, title: "Advanced", tip: "You are in the Advanced Strength Class");
    timer = new Timer();
    timer.schedule(() -> {
        Intent i = new Intent( packageContext: SquatProgressActivity.this, AdvancedActivity.class);
        startActivity(i);
        finish();
    }, delay: 2000);
}

```



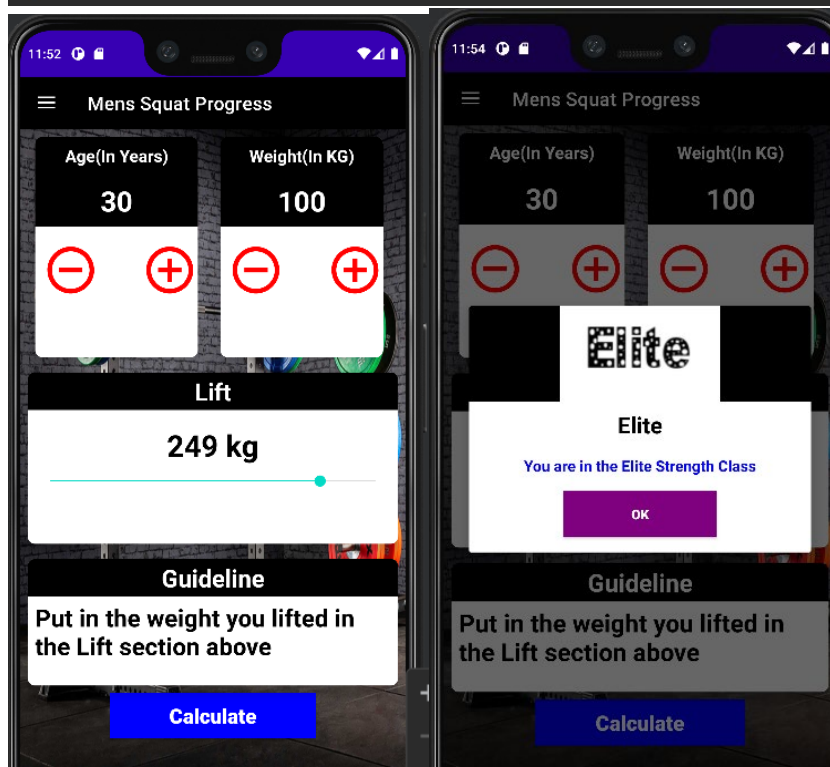
Name of Test Case: Successful use of the workout progress page in order to get a user in the elite strength class

Description of Test Case: Filling in the information for age, weight and lift and pressing the calculate button in order for the user to be an elite lifter

Expected Result of Test Case: If squat is over 218kg the user in an elite lifter

Actual Result of Test Case: Success as squat is 249kg, the user in an elite lifter

```
}else{
    showCustomDialog3(R.drawable.elite, title: "Elite", tip: "You are in the Elite Strength Class");
    timer = new Timer();
    timer.schedule(() -> {
        Intent i = new Intent( packageContext: SquatProgressActivity.this, EliteActivity.class);
        startActivity(i);
        finish();
    }, delay: 2000);
}
```



Deployment

Due to financial constraints I was not able to deploy the app but I will show below the steps that need to be done in order to host the app on the Google Play Store.

1.A Google developer account has to be created: A google developer account can be created by using an existing Google account and the registration fee which is €25 has to be paid in order to host the app

2.A merchant account has to be added to the main account: A merchant account has to be added if you want to publish paid applications and in application purchases like subscriptions, when a merchant account is created its linked to the main Google developer account

- 3.The documents that are going to be uploaded have to be prepared: Documentation is necessary to be done for legal reasons because without it rule breaches may occur when the application is published
- 4.Google Developer Policies have to be studied: Before the app is published you must know the policies of Google
- 5.Technical Requirements have to be taken into consideration: The user has to make sure the app is unique and hasn't been used before
- 6.The app is created on the Google console: Before publishing the app its details are outlined and specified here
- 7.Store Listing: The potential customer information is given as well as the description of the app you are publishing
- 8.Content Rating: The passing of a rating questionnaire is a requirement and if not passed the app is classed as unrated and is at risk of being taken down.
- 9.The Application Pricing: The specification of the app cost whether it's free a free application or you have to pay for it, in this case you would have to supply the price of the application
- 10.APK is uploaded and sent for review: At last all the information is sent out and requested for review if the review is passed the app will be successfully published and available to download on the Google Play Store

2.5. Evaluation

The app I created does everything I planned it to do. A user is required to register in order to make an account which in order lets the user use the app once they login after they've registered. Once registered and logged in the user is taken to the home page and from here they are free to use the app in whatever way they want to. One of the key requirements was for the user to track their calories and their workouts and they are able to do both since pages with text fields were made and information submission was made possible with backend functionality behind the insert, view, update and delete buttons. Another key requirement was for the user to freely navigate through the app when using it and this was made possible with the navigation menu which allows the user to switch to any key feature within the app by clicking on whichever row on the menu, the navigation is also made easy by the use of buttons which transfer the user to different activities by just clicking the button. Yet another requirement was for the user to be able to search and use whatever feature they wanted within the app and this is made possible by me implementing the search functionality which allow the user to do that, the user searches a keyword or types in some letters and suggestions associated with the user input into the search bar come up and once a row is clicked the user will be taken to that feature. The K-Nearest algorithm is a complexity requirement in this app and this was implemented in all the progress features which track the users progress in terms of body weight and workout progress(Bench, Squat and Deadlift) and based on the user input they will be put into a level and will be then taken to the advice page for that level in order to get help and to improve to go a level above in the future,

all the levels in these progress pages are adjusted to all experience levels of gym users in order to help everybody since that is the aim of my gym app.

The user is also required to be able to make bookings which is made possible with my booking page which allows the user to book gym memberships, gym slots and personal trainers, The users have to fill in required information and submit it which is structured the same for gym membership and gym slot but for PTs its different as the user has to pick calendar dates and start and end times of their training with their desired PT. This leads onto the next key requirements which is fulfilled in my app since the user has to be able to pay for gym memberships and the PTs they are using which is made with my payment page which allows the user to pay with their credit card. Lastly the app is required to let the user change information if needed which is made possible in the settings page where they can change their login credentials, full name, home address, email and phone number.

So in summary everything works like it should.

3.0 Conclusions

The advantages of this project include it being made for fitness users of all ages ,experience levels and goals as my app promotes all kinds of people to improve fitness wise in order to reach their goals and live a healthy life. This can be evidently seen with my tracking, booking and progress features. Users are able to book gym memberships, gym slots and PTs which will give them the opportunity to train and with the help of PTs they can take their training to the next level. In tracking users will be able to track their nutrition and workout which is a great strength since the user will always have a goal to beat in terms of beating their last workout performance and with the calorie tracking they will always have their target nutrition figures to hit in order to improve their body.

The progress pages will show the user how they're progressing which is also a great strength as it lets the user know if they're improving or not no matter their progress result they will always be given advice based on their results in order to improve and when the user checks again and they improve and go a level above it will be a satisfying experience for the user. Another advantage of the app is the ease of use as the UI design with the colour selection, the use of buttons, well organised navigation menu and search functionality make the app. Everything is clear to see for the user with the colour selection with plenty of colour contrast and use of evident colours like blue, white and black. Every page has a background image to make the app aesthetically pleasing for the user. The menu and search functionality allow the user to go to whichever feature of the app at any time which is a major strength.

Another advantage is that the user will always be safe when using the app in terms of the storage of their information as it will be safely stored on a SQLite database and they will only be able to use the app once they provide the correct login credentials once they have made an account by registering, login credentials can be changed anytime when using the app by accessing the settings page and changing them there. Some disadvantages may

include the loading time of the app to be longer than usual sometimes since it includes a lot of features and information. Other disadvantages/limitations include with more time more features would have been possible to develop which would have widened the functionality horizon of the app and made it even better for the user to use but we will take that as a consideration for any other major project in the future.

4.0 Further Development or Research

With additional time I would have added more features. I would have added features that would require in app subscription which would give the user more to use within the app. Another feature I would like to have added with more time and resources is a feature which lets the user track the food they have eaten and the nutritional value in terms of protein, carbs and fats like what is possible on MyFitnessPal. I would have also liked to include video tutorials of exercise execution which would help beginners starting out in the gym. In my app I included calorie and workout(weight lifting) features with more time I would have liked to include a section for cardio work for the users that prefer doing cardio to weights. In this feature I would have let the user input the type of cardio, the target length and the actual length of the cardio they did and would give them advice based off their result. Lastly with more time I would have liked to integrate an API which would supply a map service that would show the user local gyms around the area they are based in, by doing this I would make the users life easier as it will show them where their local gyms are if they did not have any idea of them.

5.0 References

Bibliography

Android Dashboard Screen Layout Design | UI Design | Android Studio, 2020. *Android Dashboard Screen Layout Design | UI Design | Android Studio*. [Online]

Available at: <https://www.youtube.com/watch?v=ejbX9MO2ems&t=952s>

[Accessed 28 January 2022].

Android Studio - Automatically Change Activity after Few Seconds | Tutorial, 2018. *Android Studio - Automatically Change Activity after Few Seconds | Tutorial*. [Online]

Available at: <https://www.youtube.com/watch?v=2JGm67leD7g>

[Accessed 28 April 2022].

Braintree SDK for Android, 2021. *Braintree SDK for Android*. [Online]

Available at:

https://github.com/braintree/braintree_android?msclkid=1dc107c9d12411ec81757ec11806f050

[Accessed 03 April 2022].

Custom ListView with SearchView on ActionBar/Toolbar & onItemClick listener - Android Studio, 2018. *Custom ListView with SearchView on ActionBar/Toolbar & onItemClick listener - Android Studio*. [Online]

Available at: <https://www.youtube.com/watch?v=EIYM-wwObI>

[Accessed 1 May 2022].

Female Bench Press Standards (kg), n.d. *Female Bench Press Standards (kg)*. [Online]
Available at: <https://strengthlevel.com/strength-standards/bench-press>
[Accessed 23 April 2022].

Female Deadlift Standards (kg), n.d. *Female Deadlift Standards (kg)*. [Online]
Available at: <https://strengthlevel.com/strength-standards/deadlift>
[Accessed 23 April 2022].

Female Squat Standards (kg), n.d. *Female Squat Standards (kg)*. [Online]
Available at: <https://strengthlevel.com/strength-standards/squat>
[Accessed 23 April 2022].

How Strong Should You Be? (Noob To Freak), 2021. *How Strong Should You Be? (Noob To Freak)*. [Online]
Available at: https://www.youtube.com/watch?v=LrDJXIQ_-eg&t=576s
[Accessed 19 April 2022].

How to Make a Button Open a New Activity - Android Studio Tutorial, 2019. *How to Make a Button Open a New Activity - Android Studio Tutorial*. [Online]
Available at: <https://www.youtube.com/watch?v=bglUdb-7Rqo>
[Accessed 4 December 2021].

Login and register form using SQLite database in android studio, 2021. *Login and register form using SQLite database in android studio*. [Online]
Available at: https://www.youtube.com/watch?v=q_fGZv3TxTU
[Accessed 07 December 2021].

Male Bench Press Standards (kg), n.d. *Male Bench Press Standards (kg)*. [Online]
Available at: <https://strengthlevel.com/strength-standards/bench-press>
[Accessed 23 April 2022].

Male Deadlift Standards (kg), n.d. *Male Deadlift Standards (kg)*. [Online]
Available at: <https://strengthlevel.com/strength-standards/deadlift>
[Accessed 23 April 2022].

Male Squat Standards (kg), n.d. *Male Squat Standards (kg)*. [Online]
Available at: <https://strengthlevel.com/strength-standards/squat>
[Accessed 23 April 2022].

Navigation Drawer on Multiple Activities Using Base Activity - 08 - Water Delivery Android App, 2021. *Navigation Drawer on Multiple Activities Using Base Activity - 08 - Water Delivery Android App*. [Online]
Available at: <https://www.youtube.com/watch?v=pRieCkF1Yts&t=545s>
[Accessed 16 February 2022].

Normal Weight Ranges: Body Mass Index (BMI), n.d. *Normal Weight Ranges: Body Mass Index (BMI)*. [Online]
Available at: <https://www.cancer.org/cancer/cancer-causes/diet-physical-activity/body-weight-and-cancer-risk/adult-bmi.html?msclkid=ba244d18d12311ec848b6836e721c3b2>
[Accessed 12 April 2022].

RecyclerView | Everything You Need to Know, 2021. *RecyclerView | Everything You Need to Know*. [Online]

Available at: <https://www.youtube.com/watch?v=Mc0XT58A1Z4&t=84s>

[Accessed 30 March 2022].

SQLite Database Tutorial Android Studio | Insert, Delete, Update and View Data in SQLite Database, 2021. *SQLite Database Tutorial Android Studio | Insert, Delete, Update and View Data in SQLite Database*. [Online]

Available at: <https://www.youtube.com/watch?v=9t8VVWebRFM&t=1202s>

[Accessed 03 March 2022].

TDEE Calculator, 2022. *TDEE Calculator*. [Online]

Available at: <https://tdeecalculator.net/?msclkid=cea297d1d12411ec9577e89ee892cb7d>

[Accessed 25 April 2022].

6.0 Appendices

6.1. Project Proposal

<BW Fitness>

Updated Project Proposal

1.0 Objectives

With my project I will build a gym fitness app on android studio.

This gym fitness app will be built with the use of Java, HTML, XML, CSS, JavaScript, Bootstrap and SQL. In my app the user will have to register to create an account, once registered user can login and use the app. Returning users that already have an account will have their details remembered and each time they access they will only have to login.

The users details will be stored a database. Once logged in the user will select the length of gym membership they want to get depending on what they want.

When the membership is chosen the user will be able to book timeslots for what times they want to go and which days. Different people of different ages, weights and goals will have different training styles some people may want to train more days a week than others.

Once in the app the user will be able to choose a PT to help them train if they are new or if they want a trainer to push them towards a certain goal if they need some help

regarding training so the physical aspect or equally as important the mental aspect by motivating their clients. There will be an information page with videos and descriptions of exercises, this will be targeted to new users that need a dummy guide on how to get started in terms of strength training.

Using the “K Nearest Programming Algorithm” the user will be recommended a certain PT for certain goals e.g. fat loss depending on how their progress is going. This algorithm will take in information on how the user is progressing and depending on their progress and their goals this algorithm will recommend certain Pts for the user to train with.

To track progress the user will put in a starting weight, a goal weight and they will track their calories with their food intake and with the number of calories they’re burning through exercise

The user will be able track their workouts by being able to put in what they’re training, and the exercises reps and sets done. Once again by using the “K Nearest Programming Algorithm” the app will recommend new calorie intake depending on the progress and the goals of the customer. For certain training days e.g. if a user chooses to train chest the app will recommend exercises and rep ranges to train in to make the users life easier.

The app will have Credit card and PayPal authentication so users can pay for PTs and gym membership.

2.0 Background

I chose this project as I am passionate about the gym and fitness, it is well advised to choose An area you are passionate about for this project. With the influence of social media always growing in our era people see other people with their dream body which can either motivate or put down someone, people often compare themselves to others. With my app I want to put an end to the negative aspect of fitness and I want people to enjoy the gym and fitness in order to reach their goals and keep a healthy balanced lifestyle as a result.

I will meet the objectives I’ve set out by doing extensive research at various existing gym & fitness apps and websites to make sure which ones are the best fit for this project in order to make it the most successful with customers.

I will look into similar existing apps in order to get ideas for potential designs and colour schemes I will use for my app.

Most importantly with my app I want to make sure it has all the requirements I need to make it work in order to achieve maximum customer satisfaction.

3.0 State of the Art

The similar apps that exist already include Gymshark, THRST App and MyFitnessPal to name a few examples. The main difference between my app to the ones I have names is that it will be a combination of what a standard gym app would have (like booking membership, time slots and general gym information), I am also going to add other parts to my app to make it different hence why I’m adding the calorie tracking ,training tracking, special PT training for goals and recommendations for future goals in terms of calorie consumption and exercise selection in the training part. I hope to make my application stand out and be different with my recommendation

concept through using the Dynamic Programming Algorithm in which customers will be recommended as I have already said new calorie ranges, new Pts and new exercise selection based on their progress and goals. This certainly has the potential to make customers satisfied as customers are sometimes stuck and out of ideas when it comes to exercise selection and nutrition so with this recommendation algorithm it will give customers nutrition and training ideas without them having to do any extensive work all they have to do is track what they're doing in regard to their training and nutrition and the app will make suggestions and recommendations based on the user interaction.

4.0 Technical Approach

I will take up the Agile Methodology for my project as I feel it is the best suited to the type of project we have been given. The project starts in October 2021 and ends in May 2022 which gives it a large scope. For this project to be successful risk vs reward factor will be a major asset since we want to build the best project to give us the best chance of landing the graduate job that we really want.

Since the scope of this project is large with a lot of workloads Agile is best suited since as I said risks will be taken so mistakes could happen but with agile that's possible and in comparison, with Waterfall Methodology it's not. The Agile Methodology is flexible, and the team involved can always adapt to the changing needs of the project if needs be. Since for example the requirements of this project could change overtime if the project is either going ahead or behind the plan.

The Requirements will be carefully planned out with the scope of the project in mind. The project work will be carried out in Sprints and the work done in the sprints will be reviewed in the Sprint Retrospective meetings. The requirements will also be planned on the customer needs in order to supply the best product possible

5.0 Technical Details

The project will be done using Java, HTML, XML and SQL.

The software used will be Android Studio. For the database SQLite will be used which comes with Android Studio.

The most important algorithm is the K-Nearest algorithm used for 2 separate parts of the app which is the Bodyweight progress and Workout Progress.

Making the K-Nearest algorithms work will be the most challenging part of the app because it takes in the user interaction into account. The Payment authentication will also be a difficult part as I have never dealt with this type of coding problem before so from all these challenges it will only improve my coding skills.

The menus will be important to make to let the user navigate through the app with no problems and a search bar if they do not feel like using the menu.

Apart from all the advanced programming put into this project it's important that I get the fundamentals of Android Studio correct as I have not used this software before.

It will also be crucial to make the app safe and secure for users to use since their personal details will be put in through registration/login as well as their payment details which is all very private information.

6.0 Special Resources Required

The only resources required are Android studio a SQL Database and a place to host the app which will be the Android Store. The app will be deployed to the Google Play Store and will

only be available to Android users so e.g. To use the app you will need an android phone or tablet.

7.0 Project Plan

I will start off with a high-level design which will be the base idea of the project but then I will move into low level design. By doing low level design I will investigate each part of the project in more detail to make it the best it can be.

Since I am using the Agile Methodology the first step will be to scope out the project and split the project into a series of sprints. This structure will let me know if I am on track with the project after each sprint and if the project is proving too many requirements can be removed. Up until the midpoint I am to have the base structure of the app finished with the registration and login page done with the rest of the pages started with a rough guide on how they will look come the project end. To get a better idea at the midpoint stage of what the app will look like prototypes will be made of how each page is planned to look like. After the midpoint stage the membership and gym slot booking will be made followed by: PT Booking, calorie tracking, training tracking, calorie /nutrition recommendation, training recommendation, PT recommendation, information page, settings and payment authentication.

Secondly the requirements analysis for the sprints will be made with information written down for what is expected to be done at the end of the sprint which will include UML use case diagrams. Once the requirements process is finished the practical work starts putting the requirement work into practice as its implemented into code. At the end of each sprint review meetings will take place to review the work done during the sprints to check what went to plan, what failed and what can be done better for future sprints.

Next of all the work will be released into production where it will be tested here we will find out if what was done was all correct or not. Any mistakes or failures will be addressed here and if the work is good enough to pass then it will be released.

8.0 Testing

When the app is built I will test it all myself to make sure everything runs how I want it to run. This will be done by me creating a series of test cases to see if they app runs exactly how I want it to run, if no I will make the necessary changes .

6.2. Reflective Journals

October

Supervision & Reflection Template

Student Name	Bartosz Wojdas
Student Number	X19128118
Course	BSHC Computing

Month: October

What?

Reflect on what has happened in your project this month?

I chose my project topic I created the project pitch which included a video and a word document. I researched why it's the right decision for me to pick what I picked. I have started working on the project proposal and it is currently a work in progress as it is due on November 7th. I have also started work on Project ethics since in my project I will use APIs from existing companies.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

The successes were that I now fully know which topic I have chosen and I am fully committed to completing that project. I completed the project pitch and doing that gave me more reasoning for choosing my project topic. The outstanding challenge left right now is to complete the project proposal as I have to do more research on requirements analysis, technologies and project testing.

I still need to gain access to APIs from companies that will populate my app with data

Now What?

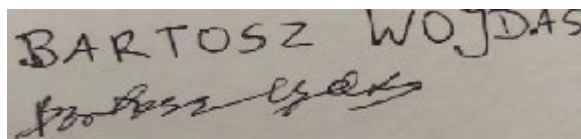
What can you do to address outstanding challenges?

I have to do more research on requirements analysis, technologies and project testing.

Once I do the research and get a good understanding of how these things apply to my project I will be able to finish the project proposal.

I still have to email the companies whom I want to use APIs from to populate my app with their data.

Student Signature



November

Supervision & Reflection Template

Student Name	Bartosz Wojdas
Student Number	X19128118
Course	BSHC Computing

Month: October

What?

Reflect on what has happened in your project this month?

I chose my project topic I created the project pitch which included a video and a word document. I researched why it's the right decision for me to pick what I picked. I have started working on the project proposal and it is currently a work in progress as it is due on November 7th. I have also started work on Project ethics since in my project I will use APIs from existing companies.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

The successes were that I now fully know which topic I have chosen and I am fully committed to completing that project. I completed the project pitch and doing that gave me more reasoning for choosing my project topic. The outstanding challenge left right now is to complete the project proposal as I have to do more research on requirements analysis, technologies and project testing.

I still need to gain access to APIs from companies that will populate my app with data

Now What?

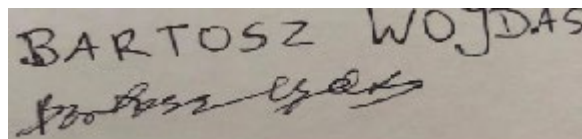
What can you do to address outstanding challenges?

I have to do more research on requirements analysis, technologies and project testing.

Once I do the research and get a good understanding of how these things apply to my project I will be able to finish the project proposal.

I still have to email the companies whom I want to use APIs from to populate my app with their data.

Student Signature


December**Supervision & Reflection Template**

Student Name	Bartosz Wojdas
Student Number	X19128118
Course	BSHC Computing Final Year

Month: December

What?

Reflect on what has happened in your project this month?

I have completed the Login and Registration pages in my project. I have completed the amount of the Technical Report which was demanded for the midpoint submission. I have submitted the midpoint submission in this month.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

I have an even greater understanding of the project after completing the majority of the Technical Report which was a success. Completing the Registration and Login pages were a big step forward.

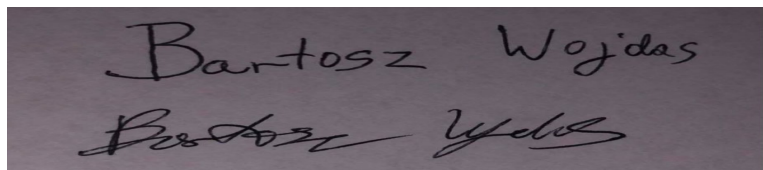
Creating the prototypes for the rest of the pages in the app means I know what has to be done exactly it's all about implementing it now.

Now What?

What can you do to address outstanding challenges?

Implementing the rest of the code for the project as well as the rest of the technical report which can be completed once the code is finished.

Student Signature


January**Supervision & Reflection Template**

Student Name	Bartosz Wojdas
Student Number	X19128118
Course	BSHC Computing Final Year

Month: January

What?

Reflect on what has happened in your project this month?

I have readjusted and changed around the design of the Registration and Login to look better, I also made sure the functionality works and the data gets saved on a database. I have started coding up the Home page and the menu that will be used across the whole app.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

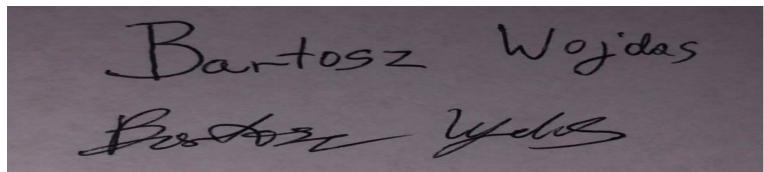
The successes are a fully working registration and login page that now looks better because I have changed the design colours and background. The challenges that remain is the rest of the app has to be done the next challenge is completing the home page and app menu.

Now What?

What can you do to address outstanding challenges?

I can plan out the coding structure and the design for the rest of the pages which will make it easier for me to complete the rest of the app

Student Signature



February and March

Supervision & Reflection Template

Student Name	Bartosz Wojdas
Student Number	X19128118
Course	BSHC Computing Final Year

Month: February and March

What?

Reflect on what has happened in your project this month?

During this month I have setup the navigation of my app so this consists of the menu that will be used across the app and also buttons on each page which will link to the other designated pages depending on which one you click. I have also edited the background of each page aswell as completing the home page. I have populated each page in terms of design and I have coded up action of uploading information to the database

in the various booking and tracking classes. The user can now insert, view, update and delete information from those pages.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

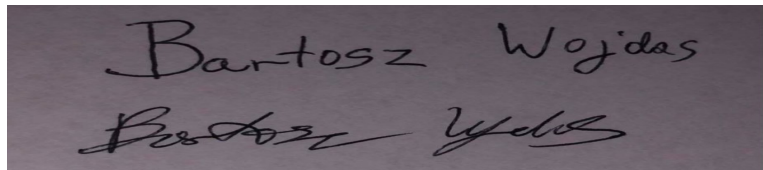
The successes were completing the home page and completing the navigation part of the app which is the menu and buttons. More of the successes include populating each of the booking and tracking pages design wise and coding up the action of being able to upload information to the database. The challenge after that is implementing the k-nearest algorithm for the calorie and workout tracking pages as well as setting up the payment authentication.

Now What?

What can you do to address outstanding challenges?

I plan to complete the following challenges and if I get stuck I will get tips from my supervisor and use the internet for help.

Student Signature

The image shows a handwritten signature in black ink on a dark background. The signature is written in a cursive style and appears to read 'Bartosz Wojdas'. There is a second, slightly lighter and less distinct signature below it.

April

Supervision & Reflection Template

Student Name	Bartosz Wojdas
Student Number	X19128118
Course	BSHC Computing Final Year

Month: April

What?

Reflect on what has happened in your project this month?

I've implemented the K Nearest algorithm implementation for the gym progress in terms of workout and body weight for the users & as well as the advice pages connected with them. I have setup payment authentication for the user as well as the settings page

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

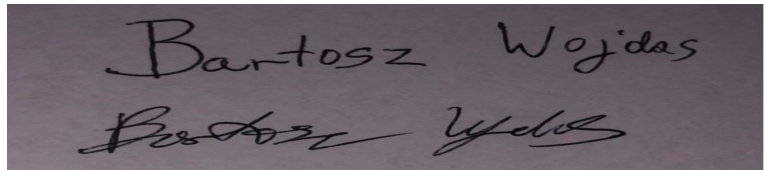
The successes were completing the K nearest algorithm implementation, payment and settings.

The challenge that still remains is completing the search functionality and finishing the documentation

Now What?

What can you do to address outstanding challenges?

I plan to complete the following challenges and if I get stuck I will get tips from my supervisor and use the internet for help.

Student Signature


6.3. Invention Disclosure Form (Remove if not completed)

Please fill in the following sections, if you think your idea is innovative:

1. Title of Invention

BWFitness

2. Inventors

Name	School/Research Institute	Affiliation with Institute (i.e. department, student, staff, visitor)	Address, contact phone no., e-mail	% Contribution to the Invention
Bartosz Wojdas	National College of Ireland	Student	51 Mount Olive Grove, Kilbarrack, Dublin 5 0871185505 X19128118@student.ncirl.ie	100%

3. Contribution to the Invention

Each contributor/potential inventor should write a paragraph relating to his/her contribution and include a signature and date at the end of the paragraph.

This invention was created fully by myself Bartosz Wojdas

4. Description of Invention

(Please highlight the novelty/patentable aspect. Attach extra sheets if necessary including diagrams where appropriate). What is novel, the 'inventive step'? For more information on patents, please look at <http://www.patentsoffice.ie/en/patents.aspx>

A gym application built on Android studio with the use of Java, HTML, XML and SQLite database. Built for gym users of all ages and experience levels with the aim to help the user improve in their fitness journey.

5. Why is this invention more advantageous than present technology?

What is its novel or unusual features? What problems does it solve? What are the problems associated with these technologies, products or processes? Explain how this invention overcomes these problems (*i.e.* what are its advantages).

I have combined features involved in multiple gym apps and brought them into one.

I have made sure it's suited to trainers of all experiences, they can track their workouts and nutrition. They can book gym membership, gym slots and Personal Trainers.

The user can check their progress and based on their level they will be given advice in order to improve to go above their current level.

It's advantageous since it suits all ages and training experience for both males and females.

6. What is the current stage of development / testing of the invention?

Prototype

7. List the names of companies which you think would be interested in using, developing or marketing this invention

Gymshark, THRST, MyFitnessPal, Nike training club

8. Funding Partner(s)

Government Agency & Department	N/A
% Support	N/A
Contract/Grant No.	N/A
Contact Name	N/A
Phone No.	N/A
Address	N/A

Industry or other Sponsor	N/A
% Support	N/A
Contract/Grant No.	N/A
Contact Name	N/A
Phone No.	N/A
Address	N/A

9. Where was the research carried out?

National College of Ireland

10. What is the potential commercial application of this invention?

Gymshark, THRST, MyFitnessPal, Nike training club

11. Was there transfer of any materials/information to or from other institutions regarding this invention?

If so please give details and provide signed agreements where relevant.

No

12. Have any third parties any rights to this invention?

If yes, give names and addresses and a brief explanation of involvement.

No

13. Are there any existing or planned disclosures regarding this invention?

Please give details.

No

14. Has any patent application been made? Yes/No **Answer:No**

If yes, give date: _____ Application No.: _____

Name of patent agent: _____

Please supply copy of specification.

15. Is a model or prototype available? Has the invention been demonstrated practically?

Yes

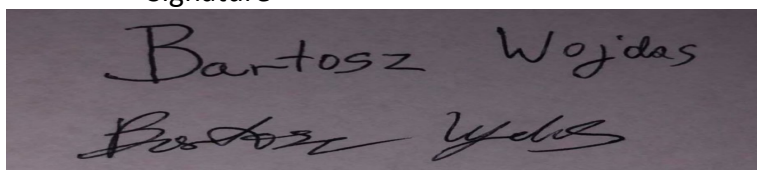
I/we acknowledge that I/we have read, understood and agree with this form and the Institute's *Intellectual Property and Procedures* and that all the information provided in this disclosure is complete and correct.

I/we shall take all reasonable precautions to protect the integrity and confidentiality of the IP in question.

Inventor: _____Bartosz Wojdas_____

Signature

Date:11/05/2022

A photograph of a handwritten signature in black ink on a light-colored surface. The signature is written in a cursive style and reads 'Bartosz Wojdas'. Below the main signature, there is a second, slightly less legible signature that also appears to read 'Bartosz Wojdas'.