11/5/2021

# Technical Manual

## Balance Health


Name : Diarmuid Brennan
Student No : C00133947
Supervisor : Joseph Kehoe

# Abstract

The purpose of the Movesense Health application is to develop a mobile application and web application that will allow medical personnel to monitor and analyse a patient's balance performance remotely from their computer reducing some of the pressures currently place on our health system.

The application can be used by medical staff to monitor and analyse a patient's balance performance to determine if there is any deterioration in the patient's balance which may be an indicator to the patient's risk of fall or other neurological conditions.

# Table of Contents

# Introduction

The purpose of this technical document will be to demonstrate the project code used to create the mobile and web application for the Balance Health project.

The full codebase can is available on the GitHub accounts linked below.

https://github.com/Diarmuid-Brennan/MovesenseHealthApp

https://github.com/Diarmuid-Brennan/BalanceHealthWebApplication

To access much of the features of the application, a Movesense device will also be needed for the mobile application to connect with.

# Project code - Mobile Application

## Activities

### Base Activity

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Base Activity - Contains methods carried out by all inheriting
activities
 */
package com.example.movesensehealthtrackerapp.view;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

import android.app.ProgressDialog;
import android.os.Handler;
import android.view.View;
import android.widget.Toast;

import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.utils.TextViewLight;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.auth.FirebaseAuth;

abstract class BaseActivity extends AppCompatActivity {

    protected ProgressDialog progressBar;
    protected TextViewLight progress;
    protected boolean doubleClickToExit = false;

    /**
     *
     * Displays a snackbar message on the activity page
     * @param message - message to be displayed
     * @param error - if the message to be displayed is an error or not
     */
    protected void showErrorSnackBar(String message, boolean error){
        Snackbar snackbar
=Snackbar.make(findViewById(android.R.id.content), message,
Snackbar.LENGTH_LONG);
        View snackBarView = snackbar.getView();

        if(error){

snackBarView.setBackgroundColor(ContextCompat.getColor(getBaseContext(),
R.color.colorSnackBarError));
        }
        else{

snackBarView.setBackgroundColor(ContextCompat.getColor(getBaseContext(),
R.color.colorSnackBarSuccess));
        }
        snackbar.show();
    }
```

```
    /**
     * Shows a progress image when loading data from the database
     * @param display - message to be displayed
     */
    protected void showProgressDialog(String display){
        progressBar = new ProgressDialog(this);
        progressBar.setContentView(R.layout.dialog_progress);
        progressBar.setCancelable(false);
        progressBar.setCanceledOnTouchOutside(false);
        progressBar.show();
    }

    /**
     * Removes the progress image
     */
    public void hideProgressDialog(){
        progressBar.dismiss();
    }

    /**
     * Method requiring to select back button twice to exit the application
     * Logs the user out
     */
    protected void doubleBackToExit(){
        if(doubleClickToExit){
            FirebaseAuth.getInstance().signOut();
            super.onBackPressed();
            return;
        }

        this.doubleClickToExit = true;
        Toast.makeText(this, getString(R.string.click_again_to_exit),
Toast.LENGTH_SHORT);

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                doubleClickToExit = false;
            }
        },2000);
    }
}
```

*Activity_base.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".view.BaseActivity">

</androidx.constraintlayout.widget.ConstraintLayout>
```

*Splash Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Splash Activity - Displays application logo before accessing the
application
 * Enables bluetooth for the mobile device
 */
package com.example.movesensehealthtrackerapp.view;

import androidx.appcompat.app.AppCompatActivity;

import android.bluetooth.BluetoothAdapter;
import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.view.WindowInsets;
import android.view.WindowInsetsController;
import android.view.WindowManager;

import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.model.BalanceActivity;
import com.example.movesensehealthtrackerapp.utils.Constant;


public class SplashActivity extends AppCompatActivity {

    @Override
    @SuppressWarnings("DEPRECATION")
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash);

        BluetoothAdapter bAdapter = BluetoothAdapter.getDefaultAdapter();
        bAdapter.enable();

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            final WindowInsetsController insetsController =
getWindow().getInsetsController();
            if (insetsController != null) {
                insetsController.hide(WindowInsets.Type.statusBars());
            }
        } else {
            getWindow().setFlags(
                    WindowManager.LayoutParams.FLAG_FULLSCREEN,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN
            );
        }

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(getBaseContext(),
LoginActivity.class);
                startActivity(intent);
                finish();
            }
        },2500);
```

```
    }

}
```

### *Activity_splash.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/logo"
    tools:context=".view.SplashActivity">


</androidx.constraintlayout.widget.ConstraintLayout>
```

### *Login Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Login Activity - Allows user to log on to the application
 */
package com.example.movesensehealthtrackerapp.view;

import androidx.annotation.NonNull;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.view.WindowInsets;
import android.view.WindowInsetsController;
import android.view.WindowManager;

import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.controller.LoginController;
import com.example.movesensehealthtrackerapp.model.User;
import com.example.movesensehealthtrackerapp.services.FirebaseDBConnection;
import com.example.movesensehealthtrackerapp.utils.CustomButtonView;
import com.example.movesensehealthtrackerapp.utils.TextViewBold;
import com.example.movesensehealthtrackerapp.utils.TextViewLight;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class LoginActivity extends BaseActivity implements
View.OnClickListener {

    private TextInputEditText email;
    private TextInputEditText password;
    private TextViewBold tv_register;
    private CustomButtonView btn_login;
```

```
    private TextViewLight forgot_password;
    private FirebaseAuth mAuth;
    private FirebaseDBConnection firebaseDBConnection;
    private LoginController loginController;

    private static final String TAG = LoginActivity.class.getSimpleName();

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

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            final WindowInsetsController insetsController =
getWindow().getInsetsController();
            if (insetsController != null) {
                insetsController.hide(WindowInsets.Type.statusBars());
            }
        } else {
            getWindow().setFlags(
                    WindowManager.LayoutParams.FLAG_FULLSCREEN,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN
            );
        }

        mAuth = FirebaseAuth.getInstance();
        firebaseDBConnection = new FirebaseDBConnection();
        loginController = new LoginController();

        email = findViewById(R.id.et_email);
        password = findViewById(R.id.et_password);
        tv_register = findViewById(R.id.tv_register);
        tv_register.setOnClickListener(this);
        btn_login = findViewById(R.id.btn_login);
        btn_login.setOnClickListener(this);
        forgot_password = findViewById(R.id.tv_forgot_password);
        forgot_password.setOnClickListener(this);
    }

    /**
     * Authenticates the entered user details with firestore
     */
    private void loginInRegisteredUser() {
        String validatedEmail = email.getText().toString().trim();
        String validatedPassword = password.getText().toString().trim();
        if (loginController.validateLoginDetails(validatedEmail,
validatedPassword)) {

            showProgressDialog(getString(R.string.please_wait));
            mAuth.signInWithEmailAndPassword(validatedEmail,
validatedPassword)
                    .addOnCompleteListener(LoginActivity.this, new
OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult>
task) {
                            if (task.isSuccessful()) {
                                Log.d(TAG, "loginUserWithEmail:success");

firebaseDBConnection.getCurrentUserDetails(LoginActivity.this);
                            } else {
```

```
                                hideProgressDialog();
                                // If sign in fails, display a message to
the user.
                                Log.w(TAG, "login:failure",
task.getException());

showErrorSnackBar(task.getException().getMessage(), true);
                            }
                        }
                    });
        }
        else{
            showErrorSnackBar(getString(R.string.err_msg_enter_password),
true);
        }
    }

    /**
     * Method is called after successful authentication
     * Brings the user to the main activity page
     * @param user - contains the details of the logged in user
     */
    public void userLoggedIn(User user){
        hideProgressDialog();
        Log.i("User Details: ", user.getFirstname() + " " +
user.getLastName() + " " + user.getEmail());

        Intent intent = new Intent(LoginActivity.this, MainActivity.class);
        startActivity(intent);
        finish();
    }


    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.tv_forgot_password:
                Intent intent = new Intent(getBaseContext(),
ForgotPasswordActivity.class);
                startActivity(intent);
                break;
            case R.id.btn_login:
                loginInRegisteredUser();
                break;
            case R.id.tv_register:
                Intent intent2 = new Intent(getBaseContext(),
RegisterActivity.class);
                startActivity(intent2);
                break;
            default:
                break;
        }
    }
}
```

*Activity_login.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".view.LoginActivity">

    <FrameLayout
        android:id="@+id/header_image"
        android:layout_width="wrap_content"
        android:layout_height="@dimen/auth_header_image_height"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:contentDescription="@string/content_desc"
            android:scaleType="fitXY"
            android:src="@drawable/logo"/>
    </FrameLayout>

    <com.example.movesensehealthtrackerapp.utils.TextViewBold
        android:id="@+id/tv_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="35dp"
        android:text="@string/title_login"
        android:textColor="@color/colorPrimaryText"
        android:textSize="@dimen/title_textSize"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/header_image" />

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/til_email"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="35dp"
        android:layout_marginEnd="16dp"
        android:hint="@string/email_id"
        android:textColorHint="@color/colorSecondaryText"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/tv_title">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/et_email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="text"
            android:padding="@dimen/padding"
            android:textColor="@color/colorPrimaryText"
            android:textSize="@dimen/textSize"
            tools:text="abc@gmail.com" />
    </com.google.android.material.textfield.TextInputLayout>

    <!--style="@style/Widget.MaterialComponents.Button.OutlinedButton"-->
    <com.google.android.material.textfield.TextInputLayout
```

```
        android:id="@+id/til_password"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="16dp"
        android:hint="@string/password_id"
        android:textColorHint="@color/colorSecondaryText"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/til_email">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/et_password"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword"
            android:padding="@dimen/padding"
            android:textColor="@color/colorPrimaryText"
            android:textSize="@dimen/textSize"
            tools:text="123456" />
    </com.google.android.material.textfield.TextInputLayout>

    <com.example.movesensehealthtrackerapp.utils.TextViewLight
        android:id="@+id/tv_forgot_password"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="16dp"
        android:foreground="?attr/selectableItemBackground"
        android:padding="@dimen/clickable_text_view_padding"
        android:text="@string/forgot_password_id"
        android:textColor="@color/colorSecondaryText"
        android:textSize="@dimen/forgot_password_textSize"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@id/til_password" />

    <com.example.movesensehealthtrackerapp.utils.CustomButtonView
        android:id="@+id/btn_login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginStart="16dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="16dp"
        android:background="@drawable/button_background"
        android:foreground="?attr/selectableItemBackground"
        android:paddingTop="@dimen/btn_padding"
        android:paddingBottom="@dimen/btn_padding"
        android:text="@string/button_label_login"
        android:textColor="@android:color/white"
        android:textSize="@dimen/btn_textSize"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/tv_forgot_password" />

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
```

```
        android:gravity="center_vertical"
        android:orientation="horizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/btn_login">

        <com.example.movesensehealthtrackerapp.utils.TextViewLight
            android:id="@+id/tv_don_t_have_an_account"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:padding="@dimen/clickable_text_view_padding"
            android:text="@string/no_account"
            android:textColor="@color/colorSecondaryText"
            android:textSize="@dimen/lbl_text_view_textSize" />

        <com.example.movesensehealthtrackerapp.utils.TextViewBold
            android:id="@+id/tv_register"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:foreground="?attr/selectableItemBackground"
            android:padding="@dimen/clickable_text_view_padding"
            android:text="@string/register"
            android:textColor="@color/colorSecondaryText"
            android:textSize="@dimen/lbl_text_view_textSize" />
    </LinearLayout>


</androidx.constraintlayout.widget.ConstraintLayout>
```

### *Register Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Register Activity - Registers a user with the firestore database
 */
package com.example.movesensehealthtrackerapp.view;

import androidx.annotation.NonNull;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.view.WindowInsets;
import android.view.WindowInsetsController;
import android.view.WindowManager;
import android.widget.Toast;

import androidx.appcompat.widget.Toolbar;

import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.model.User;
import com.example.movesensehealthtrackerapp.services.FirebaseDBConnection;
import com.example.movesensehealthtrackerapp.utils.CustomButtonView;
import com.example.movesensehealthtrackerapp.utils.TextViewBold;
import com.google.android.gms.tasks.Task;
```

```
import com.google.android.material.textfield.TextInputEditText;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.AuthResult;



public class RegisterActivity extends BaseActivity {

    private TextViewBold tv_login;
    private Toolbar toolbar;
    private TextInputEditText firstname;
    private TextInputEditText lastname;
    private TextInputEditText email;
    private TextInputEditText password;
    private TextInputEditText confirmPassword;
    private CustomButtonView registerButton;
    private FirebaseUser user;

    private static final String TAG =
RegisterActivity.class.getSimpleName();
    private FirebaseAuth mAuth;
    private FirebaseDBConnection firebaseDBConnection;

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

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            final WindowInsetsController insetsController =
getWindow().getInsetsController();
            if (insetsController != null) {
                insetsController.hide(WindowInsets.Type.statusBars());
            }
        } else {
            getWindow().setFlags(
                    WindowManager.LayoutParams.FLAG_FULLSCREEN,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN
            );
        }

        setupBackFunction();
        firstname = findViewById(R.id.et_first_name);
        lastname = findViewById(R.id.et_last_name);
        email = findViewById(R.id.et_email);
        password = findViewById(R.id.et_password);
        confirmPassword = findViewById(R.id.et_confirm_password);
        registerButton = findViewById(R.id.btn_register);
        tv_login = findViewById(R.id.tv_login);

        mAuth = FirebaseAuth.getInstance();
        firebaseDBConnection = new FirebaseDBConnection();

        registerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                registerUser();
            }
        });
```

```
        tv_login.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getBaseContext(),
LoginActivity.class);
                startActivity(intent);
                finish();
            }
        });
    }

    /**
     * adds a button to the toolbar that returns the user to the previous
activity
     */
    private void setupBackFunction(){
        toolbar = (Toolbar) findViewById(R.id.toolbar_register_activity);
        setSupportActionBar(toolbar);

        if(getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setHomeAsUpIndicator(R.drawable.ic_baseline_keyboard_
arrow_left_24);
        }

        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onBackPressed();
            }
        });
    }

    /**
     * registers a user with the firestore database
     */
    private void registerUser() {
        if (validateRegisterDetails()) {
            showProgressDialog(getString(R.string.please_wait));

            String validatedEmail = email.getText().toString().trim();
            String validatedPassword =
password.getText().toString().trim();

            mAuth.createUserWithEmailAndPassword(validatedEmail,
validatedPassword)
                    .addOnCompleteListener(RegisterActivity.this, new
OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull Task<AuthResult>
task) {
                            if (task.isSuccessful()) {
                                Log.d(TAG, "createUserWithEmail:success");
                                user = mAuth.getCurrentUser();

                                User newUser = new User(
                                        user.getUid(),

firstname.getText().toString().trim(),
```

```
lastname.getText().toString().trim(),
                                        email.getText().toString().trim()
                                );

firebaseDBConnection.registerUser(RegisterActivity.this, newUser);
                            } else {
                                hideProgressDialog();
                                // If sign in fails, display a message to
the user.
                                Log.w(TAG, "createUserWithEmail:failure",
task.getException());

showErrorSnackBar(task.getException().getMessage(), true);
                            }
                        }
                    });
        }
    }

    /**
     * On success registration display snackbar
     */
    public void registerSuccess(){
        hideProgressDialog();
        showErrorSnackBar(getString(R.string.registered_successfully),
false);
    }

    /**
     * Validates the user details entered to the registration form
     * A snack bar is displayed showing if the user has entered invalid
data
     * @return - boolean showing where the entered details were valid
     */
    private boolean validateRegisterDetails(){
        if(TextUtils.isEmpty(firstname.getText().toString().trim()) ||
firstname.getText().toString().trim().length() < 3){
            showErrorSnackBar(getString(R.string.err_msg_enter_first_name),
true);
            return false;
        }
        if(TextUtils.isEmpty(lastname.getText().toString().trim())){
            showErrorSnackBar(getString(R.string.err_msg_enter_last_name),
true);
            return false;
        }
        if(TextUtils.isEmpty(email.getText().toString().trim())){
            showErrorSnackBar(getString(R.string.err_msg_enter_email),
true);
            return false;
        }
        if(TextUtils.isEmpty(password.getText().toString().trim()) ||
password.getText().toString().trim().length() < 6){
            showErrorSnackBar(getString(R.string.err_msg_enter_password),
true);
            return false;
        }
        if(TextUtils.isEmpty(confirmPassword.getText().toString().trim())){

showErrorSnackBar(getString(R.string.err_msg_enter_confirm_password),
```

```
true);
            return false;
        }

if(!password.getText().toString().trim().equals(confirmPassword.getText().t
oString().trim())){

showErrorSnackBar(getString(R.string.err_msg_password_and_confirm_password_
mismatch), true);
            return false;
        }
        return true;
    }
}
```

*Actvity_register.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".view.RegisterActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar_register_activity"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <com.example.movesensehealthtrackerapp.utils.TextViewBold
            android:id="@+id/tv_title"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="center"
            android:paddingStart="@dimen/toolbar_title_paddingStart"
            android:paddingEnd="0dp"
            android:text="@string/create_an_account"
            android:textColor="@color/colorPrimaryText"
            android:textSize="@dimen/toolbar_title_text_size" />

    </androidx.appcompat.widget.Toolbar>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/toolbar_register_activity">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/til_first_name"
```

```
style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="16dp"
                android:layout_marginTop="16dp"
                android:layout_marginEnd="16dp"
                android:hint="@string/hint_first_name"
                android:textColorHint="@color/colorSecondaryText"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toTopOf="parent">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/et_first_name"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="text"
                    android:padding="@dimen/padding"
                    android:textColor="@color/colorPrimaryText"
                    android:textSize="@dimen/textSize" />
            </com.google.android.material.textfield.TextInputLayout>

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/til_last_name"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="16dp"
                android:layout_marginTop="25dp"
                android:layout_marginEnd="16dp"
                android:hint="@string/hint_last_name"
                android:textColorHint="@color/colorSecondaryText"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/til_first_name">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/et_last_name"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="text"
                    android:padding="@dimen/padding"
                    android:textColor="@color/colorPrimaryText"
                    android:textSize="@dimen/textSize" />
            </com.google.android.material.textfield.TextInputLayout>

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/til_email"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="16dp"
                android:layout_marginTop="25dp"
                android:layout_marginEnd="16dp"
                android:hint="@string/email_id"
                android:textColorHint="@color/colorSecondaryText"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
```

```
                app:layout_constraintTop_toBottomOf="@id/til_last_name">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/et_email"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="textEmailAddress"
                    android:padding="@dimen/padding"
                    android:textColor="@color/colorPrimaryText"
                    android:textSize="@dimen/textSize" />
            </com.google.android.material.textfield.TextInputLayout>

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/til_password"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="16dp"
                android:layout_marginTop="25dp"
                android:layout_marginEnd="16dp"
                android:hint="@string/password_id"
                android:textColorHint="@color/colorSecondaryText"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/til_email">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/et_password"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="textPassword"
                    android:padding="@dimen/padding"
                    android:textColor="@color/colorPrimaryText"
                    android:textSize="@dimen/textSize" />
            </com.google.android.material.textfield.TextInputLayout>

            <com.google.android.material.textfield.TextInputLayout
                android:id="@+id/til_confirm_password"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="16dp"
                android:layout_marginTop="25dp"
                android:layout_marginEnd="16dp"
                android:hint="@string/et_hint_confirm_password"
                android:textColorHint="@color/colorSecondaryText"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/til_password">

                <com.google.android.material.textfield.TextInputEditText
                    android:id="@+id/et_confirm_password"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:inputType="textPassword"
                    android:padding="@dimen/padding"
                    android:textColor="@color/colorPrimaryText"
                    android:textSize="@dimen/textSize" />
            </com.google.android.material.textfield.TextInputLayout>
```

```
            <com.example.movesensehealthtrackerapp.utils.CustomButtonView
                android:id="@+id/btn_register"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:layout_marginStart="16dp"
                android:layout_marginTop="30dp"
                android:layout_marginEnd="16dp"
                android:background="@drawable/button_background"
                android:foreground="?attr/selectableItemBackground"
                android:gravity="center"
                android:paddingTop="@dimen/btn_padding"
                android:paddingBottom="@dimen/btn_padding"
                android:text="@string/btn_lbl_register"
                android:textColor="@android:color/white"
                android:textSize="@dimen/btn_textSize"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@id/til_confirm_password" />

            <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="25dp"
                android:gravity="center_vertical"
                android:orientation="horizontal"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@id/btn_register">

                <com.example.movesensehealthtrackerapp.utils.TextViewLight
                    android:id="@+id/tv_already_have_an_account"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:padding="@dimen/clickable_text_view_padding"
                    android:text="@string/already_have_an_account"
                    android:textColor="@color/colorSecondaryText"
                    android:textSize="@dimen/lbl_text_view_textSize" />

                <com.example.movesensehealthtrackerapp.utils.TextViewBold
                    android:id="@+id/tv_login"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:foreground="?attr/selectableItemBackground"
                    android:padding="@dimen/clickable_text_view_padding"
                    android:text="@string/login"
                    android:textColor="@color/colorSecondaryText"
                    android:textSize="@dimen/lbl_text_view_textSize" />
            </LinearLayout>
        </androidx.constraintlayout.widget.ConstraintLayout>
    </ScrollView>

</androidx.constraintlayout.widget.ConstraintLayout>
```

*Forgot Password Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Forgot Password Activity - User can change password
 * Uses firestore's in-built update password functionality over email
 */
package com.example.movesensehealthtrackerapp.view;

import androidx.annotation.NonNull;
import androidx.appcompat.widget.Toolbar;

import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.WindowInsets;
import android.view.WindowInsetsController;
import android.view.WindowManager;
import android.widget.Toast;

import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.utils.CustomButtonView;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.android.material.textfield.TextInputEditText;
import com.google.firebase.auth.FirebaseAuth;

public class ForgotPasswordActivity extends BaseActivity {

    private Toolbar toolbar;
    private CustomButtonView submitButton;
    private TextInputEditText inputEmail;
    private static final String TAG =
ForgotPasswordActivity.class.getSimpleName();

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

        setupBackFunction();
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            final WindowInsetsController insetsController =
getWindow().getInsetsController();
            if (insetsController != null) {
                insetsController.hide(WindowInsets.Type.statusBars());
            }
        } else {
            getWindow().setFlags(
                    WindowManager.LayoutParams.FLAG_FULLSCREEN,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN
            );
        }

        inputEmail= findViewById(R.id.forgot_email);
        submitButton = findViewById(R.id.btn_submit);
        submitButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String email = inputEmail.getText().toString().trim();
```

```
                if (email.isEmpty()) {

showErrorSnackBar(getString(R.string.err_msg_enter_email), true);
                } else {
                    showProgressDialog(getString(R.string.please_wait));

FirebaseAuth.getInstance().sendPasswordResetEmail(email)

.addOnCompleteListener(ForgotPasswordActivity.this, new
OnCompleteListener<Void>() {
                                    @Override
                                    public void onComplete(@NonNull Task<Void>
task) {
                                        hideProgressDialog();
                                        if (task.isSuccessful()) {
                                            // Sign in success, update UI with
the signed-in user's information
                                            Log.d(TAG,
"EmailSuccessfullySent:success");


Toast.makeText(ForgotPasswordActivity.this, "Email has been successfully
sent.", Toast.LENGTH_LONG);
                                            finish();
                                        } else {
                                            // If sign in fails, display a
message to the user.
                                            Log.w(TAG,
"emailUnsuccessful:failure", task.getException());

showErrorSnackBar(task.getException().getMessage(), true);
                                        }

                                    }
                                });
                }
            }
        });
    }

    /**
     * adds a button to the toolbar that returns the user to the previous
activity
     */
    private void setupBackFunction(){
        toolbar = (Toolbar)
findViewById(R.id.toolbar_forgot_password_activity);
        setSupportActionBar(toolbar);

        if(getSupportActionBar() != null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setHomeAsUpIndicator(R.drawable.ic_baseline_keyboard_
arrow_left_24);
        }

        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onBackPressed();
            }
```

```
        });
    }
}
```

*Activity_forgot_password.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".view.ForgotPasswordActivity">

    <FrameLayout
        android:id="@+id/header_image"
        android:layout_width="wrap_content"
        android:layout_height="@dimen/auth_header_image_height"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:contentDescription="@string/content_desc"
            android:scaleType="fitXY"
            android:src="@drawable/logo"/>

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar_forgot_password_activity"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:layout_gravity="top" />
    </FrameLayout>

    <com.example.movesensehealthtrackerapp.utils.TextViewBold
        android:id="@+id/tv_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="35dp"
        android:text="@string/title_forgot_password"
        android:textColor="@color/colorPrimaryText"
        android:textSize="@dimen/forgot_password_title_textSize"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/header_image" />

    <com.example.movesensehealthtrackerapp.utils.TextViewLight
        android:id="@+id/tv_title_description"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="50dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="50dp"
        android:gravity="center"
        android:text="@string/forgot_password_title_description"
        android:textColor="@color/colorPrimaryText"

android:textSize="@dimen/forgot_password_title_description_textSize"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/tv_title" />

    <com.google.android.material.textfield.TextInputLayout
        android:id="@+id/til_email"

style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="16dp"
        android:hint="@string/email_id"
        android:textColorHint="@color/colorSecondaryText"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/tv_title_description">

        <com.google.android.material.textfield.TextInputEditText
            android:id="@+id/forgot_email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textEmailAddress"
            android:padding="@dimen/padding"
            android:textColor="@color/colorPrimaryText"
            android:textSize="@dimen/textSize" />
    </com.google.android.material.textfield.TextInputLayout>

    <com.example.movesensehealthtrackerapp.utils.CustomButtonView
        android:id="@+id/btn_submit"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginStart="16dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="16dp"
        android:background="@drawable/button_background"
        android:foreground="?attr/selectableItemBackground"
        android:gravity="center"
        android:paddingTop="@dimen/btn_padding"
        android:paddingBottom="@dimen/btn_padding"
        android:text="@string/btn_lbl_submit"
        android:textColor="@android:color/white"
        android:textSize="@dimen/btn_textSize"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/til_email" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

### *Main Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Main Activity - Allows the user to connect to the Movesense device and
access the balance activities
 */
```

```java
package com.example.movesensehealthtrackerapp.view;

import androidx.appcompat.app.AlertDialog;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.content.Intent;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.WindowInsets;
import android.view.WindowInsetsController;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.model.MyScanResult;
import com.example.movesensehealthtrackerapp.utils.Constant;
import com.example.movesensehealthtrackerapp.utils.CustomButtonView;
import com.google.firebase.auth.FirebaseAuth;
import com.movesense.mds.Mds;
import com.movesense.mds.MdsConnectionListener;
import com.movesense.mds.MdsException;
import com.polidea.rxandroidble2.RxBleClient;
import com.polidea.rxandroidble2.RxBleDevice;
import com.polidea.rxandroidble2.scan.ScanSettings;

import java.util.ArrayList;

import io.reactivex.disposables.Disposable;

public class MainActivity extends BaseActivity implements
AdapterView.OnItemClickListener  {
    private static final String LOG_TAG =
MainActivity.class.getSimpleName();
    static MyScanResult device = null;

    // MDS
    private Mds mMds;
    static private RxBleClient mBleClient;
    // UI
    private ListView mScanResultListView;
    private ArrayList<MyScanResult> mScanResArrayList = new ArrayList<>();
    private ArrayAdapter<MyScanResult> mScanResArrayAdapter;
    private CustomButtonView exerciseListButton;
    private CustomButtonView logoutButton;

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

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            final WindowInsetsController insetsController =
getWindow().getInsetsController();
            if (insetsController != null) {
```

```
                insetsController.hide(WindowInsets.Type.statusBars());
            }
        } else {
            getWindow().setFlags(
                    WindowManager.LayoutParams.FLAG_FULLSCREEN,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN
            );
        }

        mScanResultListView = (ListView)findViewById(R.id.listScanResult);
        mScanResArrayAdapter = new ArrayAdapter<>(this,
                android.R.layout.simple_list_item_1, mScanResArrayList);
        mScanResultListView.setAdapter(mScanResArrayAdapter);
        mScanResultListView.setOnItemClickListener(this);
        exerciseListButton = findViewById(R.id.balanceExListButton);
        exerciseListButton.setVisibility(View.INVISIBLE);

        requestNeededPermissions();

        logoutButton = findViewById(R.id.logout_button);
        logoutButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                FirebaseAuth.getInstance().signOut();
                Intent intent = new Intent(MainActivity.this,
LoginActivity.class);
                startActivity(intent);
                finish();
            }
        });
        initMds();
        initializeScan();
    }

    /**
     * Initializes the Movesense Mds API
     */
    private void initMds() {
        mMds = Mds.builder().build(this);
    }

    /**
     * requests the users mobile location permissions to access the
Movesense sensor device over Bluetooth
     */
    public void requestNeededPermissions() {
        if (ContextCompat.checkSelfPermission(this.getApplicationContext(),
                android.Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED){
            //continue
        } else {
            ActivityCompat.requestPermissions(this, new
String[]{android.Manifest.permission.ACCESS_FINE_LOCATION},
                    Constant.MY_PERMISSIONS_REQUEST_LOCATION);
        }
    }

    @Override
    public void onBackPressed(){
        doubleBackToExit();
```

```
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[]
permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if(requestCode == Constant.MY_PERMISSIONS_REQUEST_LOCATION){
            //continue
        }else{
            Toast.makeText(this,
getString(R.string.location_permission_denied), Toast.LENGTH_SHORT).show();
        }
    }

    /**
     * Creates a BLE client object
     * @return
     */
    private RxBleClient getBleClient() {
        if (mBleClient == null) mBleClient = RxBleClient.create(this);
        return mBleClient;
    }

    Disposable mScanSubscription;

    /**
     * Scans for nearby Movesense devices over Bluetooth
     * Display found devices to an adapter list
     */
    private void initializeScan() {
        mScanResArrayList.clear();
        mScanResArrayAdapter.notifyDataSetChanged();

        mScanSubscription = getBleClient().scanBleDevices(
                new ScanSettings.Builder()
                        .build()
        )
                .subscribe(
                        scanResult -> {
                            Log.d(LOG_TAG,"scanResult: " + scanResult);

                            if (scanResult.getBleDevice()!=null &&
                                    scanResult.getBleDevice().getName() !=
null &&

scanResult.getBleDevice().getName().startsWith(Constant.MOVESENSE)) {

                                MyScanResult msr = new
MyScanResult(scanResult);
                                if (mScanResArrayList.contains(msr))
mScanResArrayList.set(mScanResArrayList.indexOf(msr), msr);
                                else mScanResArrayList.add(0, msr);

mScanResArrayAdapter.notifyDataSetChanged();
                            }
                        },
                        throwable -> {
                            Log.e(LOG_TAG,"scan error: " + throwable);
                            onScanStopClicked(null);
                        }
```

```
                );
    }

    /**
     * Unsubscribes Bluetooth connection from connected device
     * @param view -takes in the selected adapter list item to disconnect
from
     */
    private void onScanStopClicked(View view) {
        if (mScanSubscription != null)
        {
            mScanSubscription.dispose();
            mScanSubscription = null;
        }
    }

    /**
     * Connects to device selected from adapter list
     * @param parent
     * @param view
     * @param position
     * @param id
     */
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {
        if (position < 0 || position >= mScanResArrayList.size())
            return;

        device = mScanResArrayList.get(position);
        if (!device.isConnected()) {
            onScanStopClicked(null);
            connectBLEDevice(device);
        }
    }

    /**
     * Connects to a found deVice over BLE
     * @param device - takes in the details of the found devices
     */
    private void connectBLEDevice(MyScanResult device) {
        RxBleDevice bleDevice =
getBleClient().getBleDevice(device.macAddress);

        Log.i(LOG_TAG, "Connecting to BLE device: " +
bleDevice.getMacAddress());
        mMds.connect(bleDevice.getMacAddress(), new MdsConnectionListener()
{
            @Override
            public void onConnect(String s) {
                Log.d(LOG_TAG, "onConnect:" + s);
            }

            @Override
            public void onConnectionComplete(String macAddress, String
serial) {
                for (MyScanResult sr : mScanResArrayList) {
                    if (sr.macAddress.equalsIgnoreCase(macAddress)) {
                        sr.markConnected(serial);
                        break;
                    }
```

```
                }
                onConnectionSuccessDisplayMessage();

findViewById(R.id.balanceExListButton).setVisibility(View.VISIBLE);
                mScanResArrayAdapter.notifyDataSetChanged();
            }

            @Override
            public void onError(MdsException e) {
                Log.e(LOG_TAG, "onError:" + e);
                showConnectionError(e);
            }

            @Override
            public void onDisconnect(String bleAddress) {
                Log.d(LOG_TAG, "onDisconnect: " + bleAddress);
                for (MyScanResult sr : mScanResArrayList) {
                    if (bleAddress.equals(sr.macAddress))
                        sr.markDisconnected();
                }
                mScanResArrayAdapter.notifyDataSetChanged();
            }
        });
    }

    /**
     * Displays a message when the application has successfully connected
to a selected Movesense device
     */
    private void onConnectionSuccessDisplayMessage(){
        Toast.makeText(this,
getString(R.string.connected_to_movesense_device),
Toast.LENGTH_SHORT).show();
    }

    /**
     * Displays an error when a connection error occurs with Movesense
device
     */
    private void showConnectionError(MdsException e) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this)
                .setTitle(getString(R.string.connection_error))
                .setMessage(e.getMessage());
        builder.create().show();
    }

    /**
     * Button that brings the user to th exercise list activity
     */
    public void onExerciseListClicked(View view){
        Intent balanceExListIntent = new Intent(this,
BalanceExerciseListActivity.class);
        balanceExListIntent.putExtra(Constant.SERIAL,
device.connectedSerial);
        startActivity(balanceExListIntent);
    }

}
```

*Actvity_main.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/coordinatorLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".view.MainActivity">

    <FrameLayout
        android:id="@+id/header_image"
        android:layout_width="wrap_content"
        android:layout_height="@dimen/auth_header_image_height"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:contentDescription="@string/content_desc"
            android:scaleType="fitXY"
            android:src="@drawable/logo"/>
    </FrameLayout>

        <ListView
            android:id="@+id/listScanResult"
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:layout_marginStart="0dp"
            android:layout_marginTop="60dp"
            app:layout_constraintTop_toBottomOf="@id/header_image"
            app:layout_constraintStart_toStartOf="parent" />

        <TextView
            android:id="@+id/connectToTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAlignment="center"
            app:layout_constraintTop_toBottomOf="@id/listScanResult"
            android:text="@string/connectToTextView" />
    <TextView
        android:id="@+id/unsuccessfulConnection"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAlignment="center"
        app:layout_constraintTop_toBottomOf="@id/connectToTextView"
        android:text="@string/unsuccessful_connection" />


    <com.example.movesensehealthtrackerapp.utils.CustomButtonView
        android:id="@+id/balanceExListButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginStart="16dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="16dp"
        android:background="@drawable/button_background"
```

```
        android:foreground="?attr/selectableItemBackground"
        android:paddingTop="@dimen/btn_padding"
        android:paddingBottom="@dimen/btn_padding"
        android:textColor="@android:color/white"
        android:textSize="@dimen/btn_textSize"
        android:onClick="onExerciseListClicked"
        android:text="@string/balanceExListButton"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/unsuccessfulConnection"
        />

    <com.example.movesensehealthtrackerapp.utils.CustomButtonView
        android:id="@+id/logout_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginStart="16dp"
        android:layout_marginTop="30dp"
        android:layout_marginEnd="16dp"
        android:background="@drawable/button_background"
        android:foreground="?attr/selectableItemBackground"
        android:paddingTop="@dimen/btn_padding"
        android:paddingBottom="@dimen/btn_padding"
        android:textColor="@android:color/white"
        android:textSize="@dimen/btn_textSize"
        android:text="@string/logout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/balanceExListButton"
        />


</androidx.constraintlayout.widget.ConstraintLayout>
```

### *Balance Exercise List Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Balance Exercise Activity - Provides access to the balance activities,
activity description and progress report
 */
package com.example.movesensehealthtrackerapp.view;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.WindowInsets;
import android.view.WindowInsetsController;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
```

```
import android.widget.Toast;

import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.model.BalanceActivity;
import com.example.movesensehealthtrackerapp.model.MyScanResult;
import com.example.movesensehealthtrackerapp.services.FirebaseDBConnection;
import com.example.movesensehealthtrackerapp.utils.Constant;
import com.example.movesensehealthtrackerapp.utils.CustomButtonView;

import java.util.ArrayList;
import java.util.List;

public class BalanceExerciseListActivity extends BaseActivity implements
View.OnClickListener {

        private static final String LOG_TAG =
BalanceExerciseListActivity.class.getSimpleName();
        private List<BalanceActivity> activities = new ArrayList<>();
        private String connectedSerial;

        private CustomButtonView activityDescription;
        private CustomButtonView beginActivity;
        private CustomButtonView progressReport;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_balance_exercise_list);
            Bundle extras = getIntent().getExtras();
            connectedSerial = extras.getString(Constant.SERIAL);

            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
                final WindowInsetsController insetsController =
getWindow().getInsetsController();
                if (insetsController != null) {
                    insetsController.hide(WindowInsets.Type.statusBars());
                }
            } else {
                getWindow().setFlags(
                        WindowManager.LayoutParams.FLAG_FULLSCREEN,
                        WindowManager.LayoutParams.FLAG_FULLSCREEN
                );
            }

            activityDescription = (CustomButtonView)
findViewById(R.id.activity_desc);
            activityDescription.setOnClickListener(this);

            beginActivity = (CustomButtonView)
findViewById(R.id.begin_exercise);
            beginActivity.setOnClickListener(this);

            progressReport = (CustomButtonView)
findViewById(R.id.view_progress);
            progressReport.setOnClickListener(this);
        }

    /**
     * Onclick method dpr buttons provided
     * @param v - selected view option
     */
```

```
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.activity_desc:
                Intent descIntent = new Intent(this,
ActivityDescriptionActivity.class);
                startActivity(descIntent);
                break;
            case R.id.begin_exercise:
                Intent beginIntent = new Intent(this,
BeginActivitiesActivity.class);
                beginIntent.putExtra(Constant.SERIAL, connectedSerial);
                startActivity(beginIntent);
                break;
            case R.id.view_progress:
                Intent progressIntent = new Intent(this,
ProgressReportActivity.class);
                startActivity(progressIntent);
                break;
            default:
                break;
        }
    }
```

*Activity_balance_exercise_list.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".view.BalanceExerciseListActivity">

    <FrameLayout
        android:id="@+id/header_image"
        android:layout_width="wrap_content"
        android:layout_height="@dimen/auth_header_image_height"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:contentDescription="@string/content_desc"
            android:scaleType="fitXY"
            android:src="@drawable/logo"/>
    </FrameLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="vertical"
        android:layout_marginTop="40dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@id/header_image"
        app:layout_constraintStart_toStartOf="@id/header_image">

    <com.example.movesensehealthtrackerapp.utils.CustomButtonView
```

```
        android:id="@+id/activity_desc"
        android:layout_marginTop="40dp"
        android:text="@string/activity_description"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="center"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:background="@drawable/button_background"
        android:foreground="?attr/selectableItemBackground"
        android:paddingTop="@dimen/btn_padding"
        android:paddingBottom="@dimen/btn_padding"
        android:textColor="@android:color/white"
        android:textSize="@dimen/btn_textSize"/>

    <com.example.movesensehealthtrackerapp.utils.CustomButtonView
        android:id="@+id/begin_exercise"
        android:layout_marginTop="40dp"
        android:text="@string/begin_activity"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_gravity="center"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:background="@drawable/button_background"
        android:foreground="?attr/selectableItemBackground"
        android:paddingTop="@dimen/btn_padding"
        android:paddingBottom="@dimen/btn_padding"
        android:textColor="@android:color/white"
        android:textSize="@dimen/btn_textSize"/>

        <com.example.movesensehealthtrackerapp.utils.CustomButtonView
            android:id="@+id/view_progress"
            android:text="@string/view_progress"
            android:layout_marginTop="40dp"
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:layout_gravity="center"
            android:layout_marginStart="16dp"
            android:layout_marginEnd="16dp"
            android:background="@drawable/button_background"
            android:foreground="?attr/selectableItemBackground"
            android:paddingTop="@dimen/btn_padding"
            android:paddingBottom="@dimen/btn_padding"
            android:textColor="@android:color/white"
            android:textSize="@dimen/btn_textSize"/>

    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

### *Activity Description Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Activity Description Activity - Provides a description for the balance
activities to be carried out
 */
package com.example.movesensehealthtrackerapp.view;
```

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

import android.os.Build;
import android.os.Bundle;
import android.view.WindowInsets;
import android.view.WindowInsetsController;
import android.view.WindowManager;

import com.example.movesensehealthtrackerapp.R;

public class ActivityDescriptionActivity extends AppCompatActivity {

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

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            final WindowInsetsController insetsController =
getWindow().getInsetsController();
            if (insetsController != null) {
                insetsController.hide(WindowInsets.Type.statusBars());
            }
        } else {
            getWindow().setFlags(
                    WindowManager.LayoutParams.FLAG_FULLSCREEN,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN
            );
        }
    }
}
```

*Activity_balance_description.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".view.ActivityDescriptionActivity">

    <FrameLayout
        android:id="@+id/header_image"
        android:layout_width="wrap_content"
        android:layout_height="@dimen/auth_header_image_height"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:contentDescription="@string/content_desc"
            android:scaleType="fitXY"
            android:src="@drawable/logo"/>
    </FrameLayout>

    <TextView
        android:id="@+id/tv_description"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="50px"
        android:text="@string/describe_activities"
        android:typeface="serif"
        android:textStyle="italic"
        android:lines="50"
        android:maxLines="50"
        android:layout_marginTop="35dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/header_image"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

### *Begin Exercise Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Begin Activities Activity - User can begin carrying out the balance
activities described
 */
package com.example.movesensehealthtrackerapp.view;

import androidx.activity.result.ActivityResult;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.media.AudioManager;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.media.ToneGenerator;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.WindowInsets;
import android.view.WindowInsetsController;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.model.BalanceActivity;
import com.example.movesensehealthtrackerapp.model.BalanceData;
import com.example.movesensehealthtrackerapp.model.LinearAcceleration;
import com.example.movesensehealthtrackerapp.services.FirebaseDBConnection;
import com.example.movesensehealthtrackerapp.utils.Constant;
import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
import com.github.mikephil.charting.data.Entry;
```

```
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.formatter.IndexAxisValueFormatter;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;
import com.google.firebase.Timestamp;
import com.google.gson.Gson;
import com.movesense.mds.Mds;
import com.movesense.mds.MdsException;
import com.movesense.mds.MdsNotificationListener;
import com.movesense.mds.MdsSubscription;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Date;
import java.util.List;
import java.util.Locale;

public class BeginActivitiesActivity extends BaseActivity implements
View.OnClickListener{

    // Sensor subscription
    private MdsSubscription mdsSubscription;
    public static Context context;
    private int time_limit;
    private String activityName;
    private Mds mMds;
    private boolean activityCancelled = false;
    private boolean activityCompleted = false;

    //view
    private LineChart mChart;
    private TextView xAxisTextView;
    private TextView yAxisTextView;
    private TextView zAxisTextView;
    private TextView displayActName;
    private Button startActivity;

    private List<Double> feetTogetherList = new ArrayList<>();
    private List<Double> instepList = new ArrayList<>();
    private List<Double> tandemList = new ArrayList<>();
    private List<Double> oneFootList = new ArrayList<>();

    private double[] previousValue = {0, 0, 0};
    private long timestamp = 0;
    private long SetExerciseTimeLength;
    private ToneGenerator tg;
    private int activityListPosition = 0;

    private FirebaseDBConnection firebaseDBConnection;
    private static final String LOG_TAG =
BeginActivitiesActivity.class.getSimpleName();

    private List<BalanceActivity> activities = new ArrayList<>();
    private List<BalanceData> balanceResults = new ArrayList<>();
    private String connectedSerial;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_begin_activities);
```

```
        Bundle extras = getIntent().getExtras();
        connectedSerial = extras.getString(Constant.SERIAL);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            final WindowInsetsController insetsController =
getWindow().getInsetsController();
            if (insetsController != null) {
                insetsController.hide(WindowInsets.Type.statusBars());
            }
        } else {
            getWindow().setFlags(
                    WindowManager.LayoutParams.FLAG_FULLSCREEN,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN
            );
        }
        firebaseDBConnection = new FirebaseDBConnection();
        SetExerciseTimeLength = 10;

        mChart = (LineChart) findViewById(R.id.linearAcc_lineChart);
        xAxisTextView = (TextView) findViewById(R.id.x_axis_textView);
        yAxisTextView = (TextView) findViewById(R.id.y_axis_textView);
        zAxisTextView = (TextView) findViewById(R.id.z_axis_textView);
        displayActName = (TextView) findViewById(R.id.activity_name);


xAxisTextView.setTextColor(getResources().getColor(android.R.color.holo_red
_dark));

yAxisTextView.setTextColor(getResources().getColor(android.R.color.holo_gre
en_dark));

zAxisTextView.setTextColor(getResources().getColor(android.R.color.holo_blu
e_dark));

        startActivity = (Button) findViewById(R.id.startActivity);
        startActivity.setOnClickListener(this);
        startActivity.setEnabled(false);

        context = getApplicationContext();
        checkIfActivityAlreadyCompleted();
    }

    /**
     * Method that checks the database if the user has already completed
the activities for todays date
     */
    private void checkIfActivityAlreadyCompleted(){
        showProgressDialog(getString(R.string.please_wait));
        firebaseDBConnection.checkActivities(this);
    }

    /**
     * Return method from database check confirming if the activities had
already been completed for todays date
     * @param exists - boolean confirming
     */
    public void doesDocumentExist(boolean exists){
        hideProgressDialog();
        if(exists){
            Toast.makeText(this, getString(R.string.activities_completed),
Toast.LENGTH_SHORT).show();
```

```
        }
        else{
            retrieveActivitiesFromDatabase();
        }
    }

    /**
     * Initializes the line chart to display the data gathered from the
Movesense sensor
     */
    private void initialiseChart() {
        mChart.setData(new LineData());

mChart.getDescription().setText(activities.get(activityListPosition).getAct
ivityName());
        mChart.setTouchEnabled(false);
        mChart.setAutoScaleMinMaxEnabled(true);
        mChart.invalidate();
    }

    /**
     * Method that subscribes to the Movesense sensor for th duration of an
activity
     * Displays the dat to the line graph
     * The gathered data is collected and store to the database upon
completion of the activity
     * @param accMovementList - List containing the movement data gathered
from a an activity
     */
    private void subscribeToSensors(List<Double> accMovementList ) {
        if (mdsSubscription != null) {
            unsubscribe();
        }
        final LineData mLineData = mChart.getData();
        ILineDataSet xSet = mLineData.getDataSetByIndex(0);

        if (xSet == null) {
            xSet = createSet(getString(R.string.data_x),
getResources().getColor(android.R.color.holo_red_dark));
            mLineData.addDataSet(xSet);
        }

        StringBuilder sb = new StringBuilder();
        String strContract =
sb.append(Constant.URI).append(connectedSerial).append(Constant.URI_MEAS_AC
C_13).append(Constant.URI_CLOSING_BACKET).toString();
        Log.d(LOG_TAG, strContract);
        mdsSubscription =
mMds.builder().build(this).subscribe(Constant.URI_EVENTLISTENER,
                strContract, new MdsNotificationListener() {
                    @Override
                    public void onNotification(String data) {
                        Log.d(LOG_TAG, "onNotification(): " + data);

                        LinearAcceleration accResponse = new
Gson().fromJson(data, LinearAcceleration.class);
                        if (accResponse != null &&
accResponse.body.array.length > 0) {
                            if(timestamp == 0) {
                                timestamp = accResponse.body.timestamp;
                            }else if(accResponse.body.timestamp >
```

```
timestamp+time_limit){
                                timestamp = 0;
                                tg = new
ToneGenerator(AudioManager.STREAM_MUSIC, 50000);

tg.startTone(ToneGenerator.TONE_PROP_BEEP,2000);
                                activityCompleted = true;
                                unsubscribe();
                                addScoreToDatabase(accMovementList);

                            }

                            LinearAcceleration.Array arrayData =
accResponse.body.array[0];

xAxisTextView.setText(String.format(Locale.getDefault(),
                                    "x: %.6f", arrayData.x));

yAxisTextView.setText(String.format(Locale.getDefault(),
                                    "y: %.6f", arrayData.y));

zAxisTextView.setText(String.format(Locale.getDefault(),
                                    "z: %.6f", arrayData.z));

                            double currentValue[] = {arrayData.x,
arrayData.y, arrayData.z};
                            double movement = Math.sqrt((currentValue[0] -
previousValue[0]) * (currentValue[0] - previousValue[0])
                                    + (currentValue[1] - previousValue[1])
* (currentValue[1] - previousValue[1])
                                    + (currentValue[2] - previousValue[2])
* (currentValue[2] - previousValue[2])
                            );
                            previousValue = Arrays.copyOf(currentValue,
currentValue.length);
                            mLineData.addEntry(new
Entry(accResponse.body.timestamp / 100, (float) movement), 0);
                            accMovementList.add(movement);
                            if(movement> 30){
                                tg = new
ToneGenerator(AudioManager.STREAM_MUSIC, 50000);

tg.startTone(ToneGenerator.TONE_PROP_BEEP,2000);
                                unsubscribe();
                                activityCancelled = true;
                                addScoreToDatabase(accMovementList);
                            }

                            mChart.notifyDataSetChanged();

mChart.setVisibleXRangeMaximum(Constant.DISPLAY_LIMIT);
                            mChart.moveViewToX(accResponse.body.timestamp /
Constant.ONE_HUNDRED);
                        }
                    }

                    @Override
                    public void onError(MdsException e) {
                        Log.e(LOG_TAG, "subscription onError(): ", e);
                        unsubscribe();
                    }
```

```
                });
    }

    /**
     * Calculates the average movement from the list of movement data
     * @param accMovementList - List containing all movements gathered
during activity
     * @return - the calculated average
     */
    private double calcAverage(List<Double> accMovementList)
    {
        double sum = 0;
        for (double i : accMovementList) {
            sum+=i;
        }
        return sum/(double) accMovementList.size();
    }

    /**
     * Unsubscribes from gathering data from the Movesense sensor
     */
    private void unsubscribe() {
        if (mdsSubscription != null) {
            mdsSubscription.unsubscribe();
            mdsSubscription = null;
        }
    }

    /**
     * Creates the Line to be displayed on the graph chart
     * @param name - Gives a name to the line data
     * @param color - Gives a color to the line data
     * @return - returns a LineDataSet object
     */
    private LineDataSet createSet(String name, int color) {
        LineDataSet set = new LineDataSet(null, name);
        set.setLineWidth(2.5f);
        set.setColor(color);
        set.setDrawCircleHole(false);
        set.setDrawCircles(false);
        set.setMode(LineDataSet.Mode.LINEAR);
        set.setHighLightColor(Color.rgb(190, 190, 190));
        set.setAxisDependency(YAxis.AxisDependency.LEFT);
        set.setValueTextSize(0f);
        return set;
    }

    /**
     * Updates the activity score to the database
     * @param accMovementList - list of activity movements gathered during
activity
     */
    private void addScoreToDatabase(List<Double> accMovementList) {
        showProgressDialog(getString(R.string.please_wait));
        String date = new SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault()).format(new Date());

        BalanceData balanceData = new
BalanceData(Collections.max(accMovementList),
Collections.min(accMovementList),
```

```
                calcAverage(accMovementList), date, accMovementList,
activityCompleted, activityName) ;
        balanceResults.add(balanceData);
        firebaseDBConnection.addBalanceScoreListToDB(balanceResults,this);

    }

    /**
     * return method upon uploading activity results to database
     * Determines what step to take next
     * - cancel activity
     * - perform next activity
     * - activities completed fro today
     */
    public void resultsUploadedSuccess(){
        activityCompleted = false;
        hideProgressDialog();
        if(activityCancelled){
            Intent displayMessageIntent = new Intent(this,
DisplayMessageActivity.class);
            displayMessageIntent.putExtra("message",
getString(R.string.you_failed_an_activity));
            displayMessageIntent.putExtra("heading",
getString(R.string.Hard_Luck));
            startActivity(displayMessageIntent);
            startActivity.setEnabled(false);
        }
        else{
            if(activityListPosition < 3){
                Intent displayMessageIntent = new Intent(this,
DisplayMessageActivity.class);
                displayMessageIntent.putExtra("message",
getString(R.string.you_completed_an_activity));
                displayMessageIntent.putExtra("heading",
getString(R.string.Congratulations));
                startActivity(displayMessageIntent);
                activityListPosition++;
                //accMovementList.clear();
                startActivity.setEnabled(true);
                startActivity();
            }
            else{
                Intent displayMessageIntent = new Intent(this,
DisplayMessageActivity.class);
                displayMessageIntent.putExtra("message",
getString(R.string.you_completed_all_activities));
                displayMessageIntent.putExtra("heading",
getString(R.string.Congratulations));
                startActivity(displayMessageIntent);
                startActivity.setEnabled(false);
            }
        }

    }

    /**
     * Retrieves the activities to be carried out from the database
     */
    private void retrieveActivitiesFromDatabase(){
        showProgressDialog(getString(R.string.please_wait));
        firebaseDBConnection.getBalanceActivities(this);
```

```
    }

    /**
     * Return method from retrieving activities from database
     * @param activity - List of activities
     */
    public void progressRetrievedSuccess(List<BalanceActivity> activity){
        hideProgressDialog();
        activities = activity;
        startActivity();
        startActivity.setEnabled(true);
    }
    /**
     * Return method from retrieving activities from database failed
     */
    public void progressRetrievedFailed(){
        hideProgressDialog();
        Toast.makeText(this, getString(R.string.no_activities_set),
Toast.LENGTH_SHORT).show();
    }

    /**
     * Begins each activity and initializes the line chart
     */
    private void startActivity(){
        activityName =
activities.get(activityListPosition).getActivityName();
        time_limit =
activities.get(activityListPosition).getTime_limit()*1000;

displayActName.setText(activities.get(activityListPosition).getActivityName
());
        initialiseChart();
    }

    /**
     * Displays a 3,2,1 countdown before an activity is begun when start
button is selected
     * @param v - Takes in the view selected
     */
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.startActivity:
                startActivity.setEnabled(false);
                Intent displayCountdown = new Intent(this,
CountdownActivity.class);
                startActivityForResult(displayCountdown, 1);
                break;
            default:
                break;
        }
    }

    /**
     * Once the countdown view has completed returns to Begin Activities
activity
     * Displays next activity to be carried out
     * @param requestCode
     * @param resultCode
     * @param data
```

```
     */
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch (requestCode) {
            case 1:
                switch (activityName) {
                    case "Stand with your feet side-by-side":
                        subscribeToSensors(feetTogetherList);
                        break;
                    case "Instep Stance":
                        subscribeToSensors(instepList);
                        break;
                    case "Stand on one foot":
                        subscribeToSensors(oneFootList);
                        break;
                    case "Tandem Stance":
                        subscribeToSensors(tandemList);
                        break;
                    default:
                        break;
                }
                break;
            default:
                break;
        }
    }

}
```

*Activity_begin_activities.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp">

    <LinearLayout
        android:id="@+id/linearLayout0"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/activity_name"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginTop="20dp"
            android:layout_weight="2"
            android:textAlignment="center"
            android:textSize="24sp"
            android:textStyle="bold"
            android:textColor="@color/wine"/>

    </LinearLayout>
```

```
    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:orientation="horizontal"
        android:weightSum="3">

        <TextView
            android:id="@+id/x_axis_textView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/x"
            android:textSize="16sp" />

        <TextView
            android:id="@+id/y_axis_textView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/y"
            android:textSize="16sp" />

        <TextView
            android:id="@+id/z_axis_textView"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@string/z"
            android:textSize="16sp" />


    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="vertical">


        <com.github.mikephil.charting.charts.LineChart
            android:id="@+id/linearAcc_lineChart"
            android:layout_width="match_parent"
            android:layout_height="450dp" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_marginBottom="10dp"
        android:orientation="vertical"
        >

        <com.example.movesensehealthtrackerapp.utils.CustomButtonView
            android:id="@+id/startActivity"
            android:layout_width="match_parent"
```

```
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginStart="16dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="16dp"
            android:background="@drawable/button_background"
            android:foreground="?attr/selectableItemBackground"
            android:paddingTop="@dimen/btn_padding"
            android:paddingBottom="@dimen/btn_padding"
            android:textColor="@android:color/white"
            android:textSize="@dimen/btn_textSize"
            android:text="@string/start_activity"
            android:textAlignment="center"
            />
    </LinearLayout>
</LinearLayout>
```

### *Countdown Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Countdown Activity - Displays 3,2,1, countdown before activity is
carried out
 * Returns to the calling activity once the countdown has been completed
 */
package com.example.movesensehealthtrackerapp.view;

import androidx.appcompat.app.AppCompatActivity;

import android.app.Activity;
import android.content.Intent;
import android.media.AudioManager;
import android.media.ToneGenerator;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.util.DisplayMetrics;
import android.widget.TextView;

import com.example.movesensehealthtrackerapp.R;

public class CountdownActivity extends AppCompatActivity {
    public int counter = 3;
    private ToneGenerator tg;
    private BeginActivitiesActivity activity;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_countdown);
        activity = new BeginActivitiesActivity();

        DisplayMetrics dm = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics((dm));

        int width = dm.widthPixels;
        int height = dm.heightPixels;

        getWindow().setLayout((int)(width*0.4), (int)(height* 0.4));

        final TextView counttime=findViewById(R.id.counttime);
        new CountDownTimer(3000,1000) {
```

```
            @Override
            public void onTick(long millisUntilFinished) {
                tg = new ToneGenerator(AudioManager.STREAM_MUSIC, 50000);
                tg.startTone(ToneGenerator.TONE_PROP_BEEP,500);
                counttime.setText(String.valueOf(counter));
                counter--;

            }
            @Override
            public void onFinish() {
                counttime.setText("Begin");
                Intent resultIntent = new Intent();
                setResult(Activity.RESULT_OK, resultIntent);

                finish();
            }
        }.start();
    }
}
```

### *Activity_countdowm.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:id="@+id/layout"
    android:gravity="center"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:id="@+id/counttime"
        android:layout_width="match_parent"
        android:gravity="center"
        android:textSize="30sp"
        android:layout_height="wrap_content" />
</LinearLayout>
```

### *Display Message Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Display Message Activity - Displays a message after the user has carried
out a an activity
 */
package com.example.movesensehealthtrackerapp.view;

import androidx.appcompat.app.AppCompatActivity;

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

import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.utils.Constant;

public class DisplayMessageActivity extends AppCompatActivity {
    private String message;
    private String heading;
    private TextView displayMessage;
    private TextView displayHeading;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_message);
        Bundle extras = getIntent().getExtras();
        message = extras.getString("message");
        heading = extras.getString("heading");


        DisplayMetrics dm = new DisplayMetrics();
        getWindowManager().getDefaultDisplay().getMetrics((dm));

        int width = dm.widthPixels;
        int height = dm.heightPixels;

        getWindow().setLayout((int)(width*0.6), (int)(height* 0.6));

        displayHeading = (TextView) findViewById(R.id.tv_heading);
        displayHeading.setText(heading);

        displayMessage = (TextView) findViewById(R.id.displayMessage);
        displayMessage.setText(message);
    }
}
```

*Activity_display_message.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/linearLayout3"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#A0ACAE"
    tools:context=".view.DisplayMessageActivity">



    <TextView
        android:id="@+id/tv_heading"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="3dp"
        android:layout_marginLeft="3dp"
        android:layout_marginTop="114dp"
        android:layout_marginEnd="3dp"
        android:layout_marginRight="3dp"
        android:text="Text"
        android:textAlignment="center"
        android:textSize="80px"
        android:textStyle="italic"
        android:typeface="serif"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/displayMessage"
        android:layout_width="match_parent"
```

```
        android:layout_height="441dp"
        android:layout_marginTop="88dp"
        android:lines="30"
        android:maxLines="30"
        android:text="Text"
        android:textAlignment="center"
        android:textSize="50px"
        android:textStyle="italic"
        android:typeface="serif"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tv_heading" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

### *Progress Report Activity*

```
/**
 * Diarmuid Brennan
 * 10/03/22
 * Progress Report Activity - Displays a report of the results from the
activities carried out
 */
package com.example.movesensehealthtrackerapp.view;

import android.content.Context;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.view.WindowInsets;
import android.view.WindowInsetsController;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
import com.example.movesensehealthtrackerapp.R;
import com.example.movesensehealthtrackerapp.model.BalanceData;
import com.example.movesensehealthtrackerapp.services.FirebaseDBConnection;
import com.example.movesensehealthtrackerapp.utils.Constant;
import com.github.mikephil.charting.charts.LineChart;
import com.github.mikephil.charting.components.XAxis;
import com.github.mikephil.charting.components.YAxis;
//import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.Entry;
import com.github.mikephil.charting.data.LineData;
import com.github.mikephil.charting.data.LineDataSet;
import com.github.mikephil.charting.formatter.IndexAxisValueFormatter;
import com.github.mikephil.charting.interfaces.datasets.ILineDataSet;
import com.google.firebase.firestore.FirebaseFirestore;
import android.text.format.DateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;
```

```
public class ProgressReportActivity extends BaseActivity implements
AdapterView.OnItemSelectedListener{

    private LineChart mChart;
    private FirebaseDBConnection firebaseDBConnection;
    public static Context context;
    private List<String> dates = new ArrayList<>();
    private List<BalanceData> balanceProgress = new ArrayList<>();
    private List<BalanceData> feetTogether = new ArrayList<>();
    private List<BalanceData> instepStance = new ArrayList<>();
    private List<BalanceData> tandemStance = new ArrayList<>();
    private List<BalanceData> oneFoot = new ArrayList<>();
    private FirebaseFirestore fd = FirebaseFirestore.getInstance();
    private List<Map<String,Object>> retrieveResults;

    private TextView lastActivityTaken;
    private TextView together;
    private TextView instep;
    private TextView tandem;
    private TextView onefoot;

    private Spinner spinner;
    private static final String[] paths = {"Stand with your feet side-by-
side", "Instep Stance", "Tandem Stance", "Stand on one foot"};

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

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            final WindowInsetsController insetsController =
getWindow().getInsetsController();
            if (insetsController != null) {
                insetsController.hide(WindowInsets.Type.statusBars());
            }
        } else {
            getWindow().setFlags(
                    WindowManager.LayoutParams.FLAG_FULLSCREEN,
                    WindowManager.LayoutParams.FLAG_FULLSCREEN
            );
        }

        lastActivityTaken = (TextView) findViewById(R.id.lastActivity);
        together = (TextView) findViewById(R.id.feetTogether);
        instep = (TextView) findViewById(R.id.instepStance);
        tandem = (TextView) findViewById(R.id.tandemStance);
        onefoot = (TextView) findViewById(R.id.oneFoot);

        context = getApplicationContext();
        firebaseDBConnection = new FirebaseDBConnection();
        mChart = (LineChart) findViewById(R.id.progress_lineChart);
        spinner = (Spinner)findViewById(R.id.spinner);
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(ProgressReportActivity.this,
                android.R.layout.simple_spinner_item,paths);


adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_it
em);
```

```
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(this);

        retrieveProgressFromDatabase();
    }

    /**
     * retrieves the users previous results from the database
     */
    private void retrieveProgressFromDatabase(){
        showProgressDialog(getString(R.string.please_wait));
        firebaseDBConnection.getBalanceProgress(this);
    }

    /**
     * return method containing th previous results gathered from the
database
     * @param results - List of results from previous activities carried
out
     */
    public void progressRetrievedSuccess(List<Map<String,Object>> results){
        retrieveResults = results;
        hideProgressDialog();
        separateActivityResults();
    }

    /**
     * return method containing th previous results gathered from the
database failed
     */
    public void progressRetrievedFailed(){
        hideProgressDialog();
        Toast.makeText(this, getString(R.string.no_data_found),
Toast.LENGTH_SHORT).show();
    }

    /**
     * Separates the results gathered from the database into their separate
activities
     */
    private void separateActivityResults(){
        Iterator<Map<String, Object>> iterator =
retrieveResults.iterator();
        while (iterator.hasNext()) {
            Map<String, Object> activity = iterator.next();
            Set<Map.Entry<String, Object>> entrySet =activity.entrySet();

            // for-each loop
            for(Map.Entry<String, Object> entry : entrySet) {
                if(entry.getKey().equals("Stand with your feet side-by-
side")){
                    Object act = entry.getValue();
                    BalanceData data = convertToObject(act);
                    feetTogether.add(data);
                }
                else if(entry.getKey().equals("Instep Stance")){
                    Object act = entry.getValue();
                    BalanceData data = convertToObject(act);
                    instepStance.add(data);
                }
                else if(entry.getKey().equals("Tandem Stance")){
```

```
                    Object act = entry.getValue();
                    BalanceData data = convertToObject(act);
                    tandemStance.add(data);
                }
                else{
                    Object act = entry.getValue();
                    BalanceData data = convertToObject(act);
                    oneFoot.add(data);
                }
            }
        }
        displayData();
    }

    /**
     * displays the number of completed activities
     */
    private void displayData(){
        String lastDate = feetTogether.get(0).getDate_set();
        lastActivityTaken.setText("Last Activity taken : " + lastDate);
        together.setText(getString(R.string.feet_together) +
getCompletedValues(feetTogether));
        instep.setText(getString(R.string.instep) +
getCompletedValues(instepStance));
        tandem.setText(getString(R.string.tandem) +
getCompletedValues(tandemStance));
        onefoot.setText(getString(R.string.one_foot) +
getCompletedValues(oneFoot));
    }

    /**
     * Calculates the number of successfully completed activities
     * @param activity - takes in the activity
     * @return -returns the calculated result
     */
    private String getCompletedValues( List<BalanceData> activity){
        int completed = 0;
        for (BalanceData data : activity){
            if(data.getCompleted()){
                completed++;
            }
        }
        String total = String.valueOf(completed) + "/"
+String.valueOf(activity.size());
        return total;
    }

    /**
     * Converts a Firestore object to a Balance Data class object
     * @param obj - Firestore object
     * @return - Balance Data object
     */
    private BalanceData convertToObject(Object obj){
        HashMap hashMap = (HashMap) obj;
        BalanceData data = new BalanceData(
                (double)hashMap.get("max_value"),
                (double)hashMap.get("min_value"),
                (double)hashMap.get("avg_value"),
                hashMap.get("date_set").toString(),
                (boolean)hashMap.get("completed"),
                hashMap.get("activityName").toString()
```

```
        );
        return data;
    }

    /**
     * Spinner view - allows user to select from a dropdown list
     * @param parent
     * @param view
     * @param position
     * @param id
     */
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
        switch (position) {
            case 0:
                initialiseChart(feetTogether);
                break;
            case 1:
                initialiseChart(instepStance);
                break;
            case 2:
                initialiseChart(tandemStance);
                break;
            case 3:
                initialiseChart(oneFoot);
            break;
        }
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
        //initialiseChart(feetTogether);
    }

    /**
     * Creates a chart displaying the progress results for a selected
activity
     * @param data - Takes in a list of Balance Data objects
     */
    private void initialiseChart(List<BalanceData> data) {
        mChart.setData(new LineData());
        if(!data.isEmpty())
mChart.getDescription().setText(data.get(0).getActivityName());
        mChart.setTouchEnabled(false);
        mChart.setAutoScaleMinMaxEnabled(true);
        mChart.invalidate();

        XAxis xAxis = mChart.getXAxis();
        xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
        xAxis.setDrawAxisLine(true);
        xAxis.setLabelCount(data.size(), true);
        xAxis.setGranularity(1.0f);
        xAxis.setDrawLabels(true);

        if(!data.isEmpty()){
            formatResultsDates(data);
            xAxis.setValueFormatter(new IndexAxisValueFormatter(dates));
        }

        YAxis yAxisLeft = mChart.getAxisLeft();
```

```
        yAxisLeft.setEnabled(true);

        mChart.getAxisRight().setEnabled(false);
        displayProgress(data);
    }

    /**
     * formats the dates an activity was carried out to be displayed on the
graph
     * @param data - Takes in the list of Balance Data objects for selected
activities
     */
    private void formatResultsDates(List<BalanceData> data){
        for (int currentVal = data.size()-1; currentVal >= 0; currentVal--)
{
            String date = data.get(currentVal).getDate_set();
            dates.add(date.substring(5));
        }
    }

    /**
     * Displays the results to the line graph chart
     * @param data - Takes in the list of Balance Data objects for selected
activities
     */
    private void displayProgress(List<BalanceData> data){
        final LineData mLineData = mChart.getData();
        ILineDataSet balanceSet = mLineData.getDataSetByIndex(0);

        if (balanceSet == null) {
            balanceSet = createSet(getString(R.string.averageScores),
getResources().getColor(android.R.color.holo_red_dark));

            mLineData.addDataSet(balanceSet);

            if (!data.isEmpty()){
                int num = 0;
                for(int currentVal =data.size()-1; currentVal >=0;
currentVal--)
                {
                    mLineData.addEntry(new Entry((float)num, (float)
data.get(currentVal).getAvg_Value()), 0);
                    mLineData.notifyDataChanged();
                    mChart.notifyDataSetChanged();
                    num++;
                }
            }
        }
    }

    /**
     * Creates the Line to be displayed on the graph chart
     * @param name - Gives a name to the line data
     * @param color - Gives a color to the line data
     * @return - returns a LineDataSet object
     */
    private LineDataSet createSet(String name, int color) {
        LineDataSet set = new LineDataSet(null, name);
        set.setLineWidth(2.5f);
        set.setColor(color);
        set.setMode(LineDataSet.Mode.LINEAR);
```

```
        set.setHighLightColor(Color.rgb(190, 190, 190));
        set.setAxisDependency(YAxis.AxisDependency.LEFT);
        set.setValueTextSize(10f);
        return set;
    }

}
```

*Activity_progress_report.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="6dp"
    tools:context=".view.ProgressReportActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/progress_report"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAlignment="center"
            android:text="@string/progress_report"
            android:textSize="22sp"
            android:textStyle="bold"
            android:textColor="@color/wine"/>

        <TextView
            android:id="@+id/lastActivity"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:text="@string/last_activity_completed"
            android:textSize="16sp" />

        <TextView
            android:id="@+id/activities_completed"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textAlignment="center"

android:text="@string/number_of_activities_successfully_completed"
            android:layout_marginTop="20dp"
            android:textSize="22sp"
            android:textStyle="bold"
            android:textColor="@color/wine"/>

        <TextView
            android:id="@+id/feetTogether"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
```

```
            android:text="@string/feet_together"
            android:textSize="16sp" />
        <TextView
            android:id="@+id/instepStance"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:text="@string/instep"
            android:textSize="16sp" />
        <TextView
            android:id="@+id/tandemStance"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:text="@string/tandem"
            android:textSize="16sp" />
        <TextView
            android:id="@+id/oneFoot"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:text="@string/one_foot"
            android:textSize="16sp" />



    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:orientation="vertical">

        <Spinner
            android:id="@+id/spinner"
            android:layout_width="301dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp" />

        <com.github.mikephil.charting.charts.LineChart
            android:id="@+id/progress_lineChart"
            android:layout_width="match_parent"
            android:layout_height="372dp" />

    </LinearLayout>

</LinearLayout>
```

## Models

### Balance Activity Model

```
/**
 * Diarmuid Brennan
 * 13/03/22
 * Balance Activity - Contains the data for each activity to be carried out
 */

package com.example.movesensehealthtrackerapp.model;

import android.os.Parcel;
import android.os.Parcelable;

import com.google.firebase.Timestamp;

import java.util.List;

public class BalanceActivity implements Parcelable {
    private String name;
    private String description;
    private int time_limit;
    private boolean isCompleted;

    public BalanceActivity(){

    }

    /**
     * Constructor- initializes calss
     * @param name -name of activity
     * @param description - description of activity
     * @param time_limit - time limit of activity
     */
    public BalanceActivity(String name, String description, int time_limit)
{
        this.name = name;
        this.description = description;
        this.time_limit = time_limit;
        isCompleted = false;
    }

    /**
     * Creates a parcelable object of the class allowing it to be passed
between activities
     * @param in - input object
     */
    protected BalanceActivity(Parcel in) {
        name = in.readString();
        description = in.readString();
        time_limit = in.readInt();
    }

    /**
     * Creates Balance Activity from parcelable object
     */
    public static final Creator<BalanceActivity> CREATOR = new
Creator<BalanceActivity>() {
        @Override
```

```
        public BalanceActivity createFromParcel(Parcel in) {
            return new BalanceActivity(in);
        }

        @Override
        public BalanceActivity[] newArray(int size) {
            return new BalanceActivity[size];
        }
    };
    public String getDescription() {
        return description;
    }
    public int getTime_limit() {
        return time_limit;
    }
    public String getActivityName() {
        return name;
    }

    /**
     * toString method
     * @return name of Balance Activity
     */
    public String toString() {
        return ( name );
    }

    @Override
    public int describeContents() {
        return 0;
    }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(name);
        dest.writeString(description);
        dest.writeInt(time_limit);
    }
}
```

### *Balance Data Model*

```
/**
 * Diarmuid Brennan
 * 13/03/22
 * Balance Data - Contains the movement data gathered from a carried out
activity
 */
package com.example.movesensehealthtrackerapp.model;

import com.google.firebase.Timestamp;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BalanceData {
    private double Max_Value;
```

```
    private double Min_Value;
    private String Date_set;
    private double Avg_Value;
    private List<Double> accData;
    private boolean completed;
    private String activityName;

    public BalanceData(){

    }

    /**
     * Constructor
     * @param max_value - Activity max movement value
     * @param min_value - Activity min movement value
     * @param avg_value - Activity average movement value
     * @param date_set - Date activity set
     * @param accData - List of movement data from activity
     * @param completed - boolean if activity was successfully completed
     * @param activityName - Activities name
     */
    public BalanceData(double max_value, double min_value, double
avg_value, String date_set, List<Double> accData, boolean completed, String
activityName) {
        Max_Value = max_value;
        Min_Value = min_value;
        Date_set = date_set;
        Avg_Value = avg_value;
        this.accData = accData;
        this.completed = completed;
        this.activityName = activityName;
    }

    /**
     * Constructor
     * @param max_value - Activity max movement value
     * @param min_value - Activity min movement value
     * @param avg_value - Activity average movement value
     * @param date_set - Date activity set
     * @param completed - boolean if activity was successfully completed
     * @param activityName - Activities name
     */
    public BalanceData(double max_value, double min_value, double
avg_value, String date_set,  boolean completed, String activityName) {
        Max_Value = max_value;
        Min_Value = min_value;
        Date_set = date_set;
        Avg_Value = avg_value;
        this.accData = accData;
        this.completed = completed;
        this.activityName = activityName;
    }

    public double getMax_value() {
        return Max_Value;
    }

    public void setMax_value(float max_value) {
        Max_Value = max_value;
    }
```

```
    public double getMin_value() {
        return Min_Value;
    }

    public void setMin_value(float min_value) {
        Min_Value = min_value;
    }

    public String getDate_set() {
        return Date_set;
    }

    public void setDate_set(String date_set) {
        Date_set = date_set;
    }

    public double getAvg_Value() {
        return Avg_Value;
    }

    public void setAvg_Value(float avg_Value) {
        Avg_Value = avg_Value;
    }

    public List<Double> getAccData() {
        return accData;
    }

    public void setAccData(List<Double> accData) {
        this.accData = accData;
    }

    public boolean getCompleted() {
        return completed;
    }

    public void setCompleted(boolean completed) {
        this.completed = completed;
    }

    public String getActivityName() {
        return activityName;
    }

    public void setActivityName(String activityName) {
        this.activityName = activityName;
    }


}
```

### *Linear Acceleration Model*

```
/**
 * Diarmuid Brennan
 * 13/03/22
 * Linear Acceleration - Serializes the linear acceleration data gathered
from the Movesense sensor device
 */
package com.example.movesensehealthtrackerapp.model;
import com.google.gson.annotations.SerializedName;
```

```java
public class LinearAcceleration {

    @SerializedName("Body")
    public final Body body;

    /**
     * Constructor - Takes in the data passed from the sensor
     * @param body - sensor data passed
     */
    public LinearAcceleration(Body body) {
        this.body = body;
    }

    public static class Body {
        @SerializedName("Timestamp")
        public final long timestamp;

        @SerializedName("ArrayAcc")
        public final Array[] array;

        @SerializedName("Headers")
        public final Headers header;

        public Body(long timestamp, Array[] array, Headers header) {
            this.timestamp = timestamp;
            this.array = array;
            this.header = header;
        }
    }

    public static class Array {
        @SerializedName("x")
        public final double x;
        @SerializedName("y")

        public final double y;
        @SerializedName("z")
        public final double z;

        public Array(double x, double y, double z) {
            this.x = x;
            this.y = y;
            this.z = z;
        }
    }

    public static class Headers {
        @SerializedName("Param0")
        public final int param0;

        public Headers(int param0) {
            this.param0 = param0;
        }
    }
}
```

*My Scan Result Model*

```
/**
 * Diarmuid Brennan
 * 13/03/22
 * My Scan result class - Contains the results of the results of the
Bluetooth Low Energy scan for nearby devices
 */
package com.example.movesensehealthtrackerapp.model;

import com.polidea.rxandroidble2.RxBleDevice;
import com.polidea.rxandroidble2.scan.ScanResult;

public class MyScanResult {
    public int rssi;
    public String macAddress;
    public String name;
    public String connectedSerial;

    /**
     * Constructor - Takes in the BLE scan results
     * @param scanResult - mac-address, rssi and name of found devices
     */
    public MyScanResult(ScanResult scanResult) {
        this.macAddress = scanResult.getBleDevice().getMacAddress();
        this.rssi = scanResult.getRssi();
        this.name = scanResult.getBleDevice().getName();
    }

    /**
     * Check if the device is connected
     * @return true if connected
     */
    public boolean isConnected() {return connectedSerial != null;}

    /**
     * Marks the scanned item as connected to the application
     * @param serial - returns the serial number of the device
     */
    public void markConnected(String serial) {connectedSerial = serial;}
    /**
     * Marks the scan item as disconnected from the application
     */
    public void markDisconnected() {connectedSerial = null;}

    /**
     * Checks if the selected device is connected to the correct Movesense
device
     * @param object- passes in a scanned object
     * @return - returns true if the application has connected to this
device
     */
    public boolean equals(Object object) {
        if(object instanceof MyScanResult &&
((MyScanResult)object).macAddress.equals(this.macAddress)) {
            return true;
        }
        else if(object instanceof RxBleDevice &&
((RxBleDevice)object).getMacAddress().equals(this.macAddress)) {
            return true;
        } else {
            return false;
```

```
        }
    }

    /**
     * toString method
     * @return - name, rssi and mac-address of the scanned device
     */
    public String toString() {
        return ( name + "\n" + macAddress + "\n" + " [" + rssi + "]");
    }
}
```

### User Model

```
/**
 * Diarmuid Brennan
 * 13/03/22
 * User class - Contains the details of an logged in User
 */
package com.example.movesensehealthtrackerapp.model;

public class User {
    private String userUID;
    private String firstname;
    private String lastName;
    private String email;

    /**
     * Constructor
     * @param userUID - Logged in users firestores UID
     * @param firstname - Users first name
     * @param lastName - Users last name
     * @param email - Users email
     */
    public User(String userUID, String firstname, String lastName, String
email) {
        this.userUID = userUID;
        this.firstname = firstname;
        this.lastName = lastName;
        this.email = email;
    }

    public User(){

    }

    public String getUserUID() {
        return userUID;
    }

    public void setUserUID(String userUID) {
        this.userUID = userUID;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }
```

```
    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

## Services

### Firebase DB Collection

```
/**
 * Diarmuid Brennan
 * 13/03/22
 * Firebase Connection class - Contains methods for accessing the data
stored on the Firestore database
 */
package com.example.movesensehealthtrackerapp.services;

import android.util.Log;

import androidx.annotation.NonNull;

import com.example.movesensehealthtrackerapp.model.BalanceActivity;
import com.example.movesensehealthtrackerapp.view.BeginActivitiesActivity;
import com.example.movesensehealthtrackerapp.view.LoginActivity;
import com.example.movesensehealthtrackerapp.view.ProgressReportActivity;
import com.example.movesensehealthtrackerapp.view.RegisterActivity;
import com.example.movesensehealthtrackerapp.model.BalanceData;
import com.example.movesensehealthtrackerapp.model.User;
import com.example.movesensehealthtrackerapp.utils.Constant;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.QuerySnapshot;

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
```

```
import java.util.Locale;
import java.util.Map;

public class FirebaseDBConnection{

    private  FirebaseFirestore firestore;
    private FirebaseAuth mAuth;
    private static final String TAG =
FirebaseDBConnection.class.getSimpleName();

    /**
     * Constructor - initializes the firestore authorization and connection
     */
    public FirebaseDBConnection() {
        firestore = FirebaseFirestore.getInstance();
        mAuth = FirebaseAuth.getInstance();
    }

    /**
     * Registers a ne user on the database
     * @param activity
     * @param user - Users entered details
     */
    public void registerUser(RegisterActivity activity, User user){
        firestore.collection(Constant.USER)
                .document(user.getUserUID())
                .set(user)
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(@NonNull Void unused) {
                        activity.registerSuccess();
                        Log.d(TAG, "User registered successfully");
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        activity.hideProgressDialog();
                        Log.w(TAG, "Error registering user", e);
                    }
                });
    }

    /**
     * Retrieves the UID for the current logged in User
     * @return - Users UID
     */
    public String getCurrentUserID(){
        FirebaseUser currentUser =
FirebaseAuth.getInstance().getCurrentUser();
        String currentUserID = "";
        if(currentUser != null) currentUserID = currentUser.getUid();
        return currentUserID;
    }

    /**
     * Retrieves the email for the current logged in User
     * @return - Users Email
     */
    public String getCurrentUserEmail(){
        FirebaseUser currentUser =
```

```
FirebaseAuth.getInstance().getCurrentUser();
        String currentUserEmail = "";
        if(currentUser != null) currentUserEmail = currentUser.getEmail();
        return currentUserEmail;
    }

    /**
     * Retrieves the balance activities to be carried out from the database
     * @param activity
     */
    public void getBalanceActivities(BeginActivitiesActivity activity){
        List<BalanceActivity> activities = new ArrayList<>();
        firestore.collection(Constant.ACTIVITIES)
                .get()
                .addOnSuccessListener(new
OnSuccessListener<QuerySnapshot>() {
                    @Override
                    public void onSuccess(QuerySnapshot
queryDocumentSnapshots) {
                        if (!queryDocumentSnapshots.isEmpty()) {
                            List<DocumentSnapshot> list =
queryDocumentSnapshots.getDocuments();
                            for (DocumentSnapshot document : list) {
                                Log.d(TAG, document.getId() + " => " +
document.getData());
                                BalanceActivity activityData = new
BalanceActivity((String)document.get("name"),
(String)document.get("description"),
((Long)document.get("time_limit")).intValue());
                                activities.add(activityData);
                            }
                            activity.progressRetrievedSuccess(activities);
                        } else {
                            activity.progressRetrievedFailed();
                            Log.d(TAG, "No data currently stored in
database");
                        }
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                activity.hideProgressDialog();
                Log.w(TAG, "Error retrieving data", e);
            }
        });
    }

    /**
     * Check to see if the user has already completed the activities to be
carried out for todays date
     * @param activity
     */
    public void checkActivities(BeginActivitiesActivity activity){
        String date = new SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault()).format(new Date());
        DocumentReference docRef =
firestore.collection(Constant.PATIENT_SCORES)
                .document(getCurrentUserEmail())
                .collection(Constant.SCORES)
                .document(date);
        docRef.get().addOnCompleteListener(new
```

```
OnCompleteListener<DocumentSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                if (task.isSuccessful()) {
                    DocumentSnapshot document = task.getResult();
                    if (document.exists()) {
                        Log.d(TAG, "DocumentSnapshot data: " +
document.getData());
                        activity.doesDocumentExist(true);
                    } else {
                        activity.doesDocumentExist(false);
                        Log.d(TAG, "No such document");
                    }
                } else {
                    activity.hideProgressDialog();
                    Log.d(TAG, "get failed with ", task.getException());
                }
            }
        });
    }

    /**
     * Retrieves the current users details from the database
     * @param activity
     */
    public void getCurrentUserDetails(LoginActivity activity){
        firestore.collection(Constant.USER)
                .document(getCurrentUserID())
                .get()
                .addOnSuccessListener(new
OnSuccessListener<DocumentSnapshot>() {
                    @Override
                    public void onSuccess(DocumentSnapshot
documentReference) {
                        Log.d(TAG, "Retrieved user details: " +
documentReference.getId());
                        User user = documentReference.toObject(User.class);
                        activity.userLoggedIn(user);
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        activity.hideProgressDialog();
                        Log.w(TAG, "Error retrieving user details", e);
                    }
                });
    }

    /**
     * Uploads the activity results for the activities carried out to the
database
     * @param balanceScores - List of the balance scores achieved
     * @param activity
     */
    public void addBalanceScoreListToDB(List<BalanceData> balanceScores,
BeginActivitiesActivity activity) {
        String date = new SimpleDateFormat("yyyy-MM-dd",
Locale.getDefault()).format(new Date());
        Map<String,Object> map = new HashMap<String,Object>();
        map = ConvertObjectToMap(balanceScores);
```

```
        firestore.collection(Constant.PATIENT_SCORES)
                .document(getCurrentUserEmail())
                .collection(Constant.SCORES)
                .document(date)
                .set(map)
                .addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(@NonNull Void unused) {
                        activity.resultsUploadedSuccess();
                        Log.d(TAG, "Document uploaded to database");
                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        activity.hideProgressDialog();
                        Log.w(TAG, "Error adding document", e);
                    }
                });
    }

    /**
     * Maps the Balance Data class object to a List of maps to be stored on
the database
     * @param balanceData -List of Balance Data objects
     * @return - List of maps
     */
    private  Map<String,Object> ConvertObjectToMap(List<BalanceData>
balanceData) {
        Map<String,Object> list = new HashMap<String,Object>();

        for(int i =0; i< balanceData.size(); i++){
            Map<String, Object> scoreMap = new HashMap<String,Object>();
            scoreMap.put("activityName",
balanceData.get(i).getActivityName());
            scoreMap.put("completed", balanceData.get(i).getCompleted());
            scoreMap.put("date_set", balanceData.get(i).getDate_set());
            scoreMap.put("avg_value", balanceData.get(i).getAvg_Value());
            scoreMap.put("acc_data", balanceData.get(i).getAccData());
            scoreMap.put("max_value", balanceData.get(i).getMax_value());
            scoreMap.put("min_value", balanceData.get(i).getMin_value());
            list.put(balanceData.get(i).getActivityName(),scoreMap);

        }
        return list;
    }

    /**
     * Retrieves the Users previous Balance activity results from the
database
     * @param activity
     */
    public void getBalanceProgress(ProgressReportActivity activity){
        List<Map<String, Object>> allDocuments = new ArrayList<>();
        firestore.collection(Constant.PATIENT_SCORES)
                .document(getCurrentUserEmail())
                .collection(Constant.SCORES)
                .get()
                .addOnSuccessListener(new
OnSuccessListener<QuerySnapshot>() {
                    @Override
```

```
                    public void onSuccess(QuerySnapshot
queryDocumentSnapshots) {
                        if (!queryDocumentSnapshots.isEmpty()) {
                            List<DocumentSnapshot> list =
queryDocumentSnapshots.getDocuments();
                            for (DocumentSnapshot document : list) {
                                Map<String, Object> data =
document.getData();
                                allDocuments.add(data);
                            }
                            int size = allDocuments.size();
                            if(size > 7){
                                List<Map<String,Object>> lastSeven = new
ArrayList<>();
                                for (int i = size-1; i >= size-7; i--){
                                    lastSeven.add(allDocuments.get(i));
                                }

activity.progressRetrievedSuccess(lastSeven);
                            }
                            else
activity.progressRetrievedSuccess(allDocuments);
                        } else {
                            activity.progressRetrievedFailed();
                            Log.d(TAG, "No data currently stored in
database");
                        }
                    }
                }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                activity.hideProgressDialog();
                Log.w(TAG, "Error retrieving data", e);
            }
        });
    }

}
```

## Utils

### Constant

```
package com.example.movesensehealthtrackerapp.utils;

public class Constant {
    public static final String USER = "users";
    public static final String PATIENT_LIST = "patient_list";
    public static final String PATIENT_ID = "patient_id";
    public static final int MY_PERMISSIONS_REQUEST_LOCATION = 1;
    public static final String URI_CONNECTEDDEVICES =
"suunto://MDS/ConnectedDevices";
    public static final String URI_EVENTLISTENER =
"suunto://MDS/EventListener";
    public static final String SCHEME_PREFIX = "suunto://";
    public static final String MOVESENSE = "Movesense";
    public static final String SERIAL = "serial";
    public static final String URI_MEAS_ACC_13 = "/Meas/Acc/13";
    public static final String HEART_RATE_PATH = "/Meas/hr";
```

```
    public static final String ECG_VELOCITY_PATH = "/Meas/ECG/128";
    public static final String URI ="{\"Uri\": \"";
    public static final String URI_CLOSING_BACKET ="\"}";
    public static final int DISPLAY_LIMIT = 50;
    public static final int ONE_HUNDRED = 100;
    public static final String BALANCE_EXERCISE1_SCORE =
"exercise1_balance_score";
    public static final String BALANCE_DATA = "balance_activity_data";
    public static final String DATE_SET = "date_set";
    public static final String PROGRESS_REPORT = "Progress Report";
    public static final String BALANCE = "Balance";
    public static final String HEART_RATE = "Heart Rate";
    public static final String PATIENT_ACTIVITIES= "patient_activities";
    public static final String ACTIVITIES= "activities";
    public static final String SCORES = "scores";
    public static final String NAME = "name";
    public static final String TIME_LIMIT = "time_limit";
    public static final String DESCRIPTION = "description";
    public static final String PARSED = "parsed";
    public static final String PATIENT_SCORES= "patient_scores";
}
```

### *Custom Button View*

```
package com.example.movesensehealthtrackerapp.utils;

import android.content.Context;
import android.graphics.Typeface;
import android.util.AttributeSet;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.widget.AppCompatButton;

public class CustomButtonView extends AppCompatButton {

    private Context context;
    private AttributeSet attrs;

    public CustomButtonView(@NonNull Context context, @Nullable
AttributeSet attrs) {
        super(context, attrs);
        this.context = context;
        this.attrs = attrs;
        applyFont();
    }

    private void applyFont(){
        Typeface t = Typeface.createFromAsset(context.getAssets(),
"Montserrat-Bold.ttf");
        setTypeface(t);
    }
}
```

### *Edit Text View Light*

```
package com.example.movesensehealthtrackerapp.utils;

import android.content.Context;
import android.graphics.Typeface;
import android.util.AttributeSet;
```

```
import androidx.appcompat.widget.AppCompatTextView;

public class EditTextViewLight extends AppCompatTextView {
    private Context context;
    private AttributeSet attrs;

    public EditTextViewLight(Context context, AttributeSet attrs) {
        super(context, attrs);
        this.context = context;
        this.attrs = attrs;
        applyFont();
    }

    private void applyFont(){
        Typeface t = Typeface.createFromAsset(context.getAssets(),
"Montserrat-Bold.ttf");
        setTypeface(t);
    }
}
```

### *Text View Bold*

```
package com.example.movesensehealthtrackerapp.utils;

import android.content.Context;
import android.graphics.Typeface;
import android.util.AttributeSet;
import android.widget.TextView;

import androidx.appcompat.widget.AppCompatTextView;

public class TextViewBold extends AppCompatTextView {
    private Context context;
    private AttributeSet attrs;

    public TextViewBold(Context context, AttributeSet attrs) {
        super(context, attrs);
        this.context = context;
        this.attrs = attrs;
        applyFont();
    }

    private void applyFont(){
        Typeface t = Typeface.createFromAsset(context.getAssets(),
"Montserrat-Bold.ttf");
        setTypeface(t);
    }
}
```

### *Text View Light*

```
package com.example.movesensehealthtrackerapp.utils;

import android.content.Context;
import android.graphics.Typeface;
import android.util.AttributeSet;

import androidx.appcompat.widget.AppCompatTextView;

public class TextViewLight extends AppCompatTextView {
```

```
    private Context context;
    private AttributeSet attrs;

    public TextViewLight(Context context, AttributeSet attrs) {
        super(context, attrs);
        this.context = context;
        this.attrs = attrs;
        applyFont();
    }

    private void applyFont(){
        Typeface t = Typeface.createFromAsset(context.getAssets(),
"Montserrat-Regular.ttf");
        setTypeface(t);
    }
}
```

# Project code - Web Application

## App.py

```
"""
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
app.py
contains all methods for mapping urls to specific functions and webpages
contains methods for GET and POST HTTP method calls to urls
"""
from flask import (
    Flask,
    render_template,
    url_for,
    redirect,
    request,
    session,
    flash,
)
import os

from data_utils import (
    register_user,
    login_user,
    add_patient,
    get_patients,
    add_activity,
    get_activities,
    get_patient,
    get_patient_activities,
    get_patient_scores,
    add_comment,
    retrieve_comments,
)
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
import io
import base64
import plotly.express as px
from datetime import date, timedelta, datetime
import numpy as np

user = {"is_logged_in": False}

app = Flask(__name__)

app.config["SECRET_KEY"] = os.urandom(24)


@app.route("/")
@app.route("/login", methods=["GET", "POST"])
def login():
    """
```

```
    login function
    GET -displays login webpage for user
    POST - validates entered details
       if successful passes user to welcome page
       if unsuccessful returns user to login page
    """
    if request.method == "POST":
        userDetails = request.form
        userlogin = login_user(userDetails)
        if userlogin == None:
            return render_template("login.html")
        global user
        user["is_logged_in"] = True
        session["userId"] = userlogin["localId"]
        return render_template("welcome.html")
    return render_template("login.html")


@app.route("/logout")
def logout():
    """
    logout function
    clears session variables
    returns user to login page
    """
    global user
    user["is_logged_in"] = False
    session.clear()
    return redirect(url_for("login"))


@app.route("/register", methods=["GET", "POST"])
def register():
    """
    register function
    GET -displays register webpage
    POST - validates entered details
       if successful registers new user and displays login page
       if unsuccessful returns user to register page displaying an error
message
    """
    if request.method == "POST":
        userDetails = request.form
        if validate_register_details(userDetails):
            userregister = register_user(userDetails)
            if userregister == None:
                return render_template("register.html")
            return redirect(url_for("login"))
    return render_template("register.html")


@app.route("/welcome")
def welcome():
    """
    welcome function
    GET -displays welcome webpage
    """
    if user["is_logged_in"] == True:
        return render_template("welcome.html")
    else:
        flash("You must be logged in to access webpage.", "error")
```

```
        return redirect(url_for("login"))


@app.route("/create_patient", methods=["GET", "POST"])
def create_patient():
    """
   create patient function
   GET - displays create patient webpage
   POST - validates entered details
      if successful creates new patient
      if unsuccessful returns user to create patient page displaying an
error message
   """
    if user["is_logged_in"] == True:
        if request.method == "POST":
            userDetails = request.form
            add_patient(userDetails)
            return render_template("create_patient.html")
        return render_template("create_patient.html")

    else:
        flash("You must be logged in to access webpage.", "error")
        return redirect(url_for("login"))


@app.route("/edit_patient", methods=["GET", "POST"])
def edit_patient():
    """
    edit patient function
    GET - displays edit patient webpage
    POST - validates entered details
       if successful edits a patients details
       if unsuccessful returns user to edit patient page displaying an
error message
    """
    if user["is_logged_in"] == True:
        data = get_patients()
        patient_details = None
        if request.method == "POST":
            userDetails = request.form
            email = userDetails["results"]
            if email == "":
                add_patient(userDetails)
            else:
                email = userDetails["results"]
                patient_details = get_patient(email)

            return render_template(
                "edit_patient.html", data=data,
patient_details=patient_details
            )
        return render_template(
            "edit_patient.html", data=data, patient_details=patient_details
        )
    else:
        flash("You must be logged in to access webpage.", "error")
        return redirect(url_for("login"))


@app.route("/view_patients", methods=["GET", "POST"])
def view_patients():
```

```
    """
    view patients function
    GET - displays view patients webpage
    POST - dispalys the patient details webpage of the selected patient
from patient list
    """
    if user["is_logged_in"] == True:
        if request.method == "POST":
            session["user_email"] = request.form["user_email"]
            return redirect(url_for("patient_details"))
        data = get_patients()
        return render_template("view_patients.html", data=data)
    else:
        flash("You must be logged in to access webpage.", "error")
        return redirect(url_for("login"))


@app.route("/patient_details", methods=["GET", "POST"])
def patient_details():
    """
    patient details function
    GET - displays patient details webpage
       displays the patients personal details and actvities
       displays any comments left by the medical staff on each of the
actvities carried out
    POST - retrieves selected activities comments and displays in table
format
    """
    if user["is_logged_in"] == True:
        if "user_email" in session:
            user_email = session["user_email"]

        patient_detail = get_patient(user_email)
        if patient_detail == None:
            return redirect(url_for("view_patients"))
        session["patient_detail"] = patient_detail

        patient_activities = get_activities()
        session["patient_activities"] = patient_activities

        if request.method == "POST":
            details = request.form
            if details["activity"] == "feetTogether":
                patient_comments = retrieve_comments(
                    "Stand with your feet side-by-side", user_email
                )
            elif details["activity"] == "instep":
                patient_comments = retrieve_comments("Instep Stance",
user_email)
            elif details["activity"] == "tandem":
                patient_comments = retrieve_comments("Tandem Stance",
user_email)
            elif details["activity"] == "general":
                patient_comments = retrieve_comments("General comments",
user_email)
            else:
                patient_comments = retrieve_comments("Stand on one foot",
user_email)
        else:
            patient_comments = retrieve_comments("General comments",
user_email)
```

```
        return render_template(
            "patient_details.html",
            data=patient_detail,
            patient_comments=patient_comments,
            patient_activities=patient_activities,
        )
    else:
        flash("You must be logged in to access webpage.", "error")
        return redirect(url_for("login"))


@app.route("/create_activity", methods=["GET", "POST"])
def create_activity():
    """
    create activity function
    GET - displays create activity webpage
    POST - validates entered details
       if successful adds an activity
       if unsuccessful returns user to create activity page displaying an
error message
    """
    if user["is_logged_in"] == True:
        if request.method == "POST":
            activityDetails = request.form
            add_activity(activityDetails)
            return render_template("create_activity.html")
        return render_template("create_activity.html")
    else:
        flash("You must be logged in to access webpage.", "error")
        return redirect(url_for("login"))


@app.route("/view_activities")
def view_activities():
    """
    view activities function
    GET - displays view activities webpage
    POST - dispalys the created activities details in a table
    """
    if user["is_logged_in"] == True:
        data = get_activities()
        return render_template("view_activities.html", data=data)
    else:
        flash("You must be logged in to access webpage.", "error")
        return redirect(url_for("login"))


@app.route("/view_activity_progress", methods=["GET", "POST"])
def view_activity_progress():
    """
   view activity progress function
   GET - displays view activity progress webpage for selected user
   retirves the selected users overall balance performance and dispalys
   the result in graph and table format
   POST - dispalys the users activities results for the selected amount of
time
      from the dropdown provided
   """
    if user["is_logged_in"] == True:
```

```
        user_email = session["user_email"]
        activities = get_activities()
        patient_scores = get_patient_scores(user_email)
        rows = create_activity_rows(patient_scores)
        df = pd.DataFrame(rows)
        df["date_set"] = pd.to_datetime(
            df["date_set"], format="%Y-%m-%d", utc=True
        ).dt.date

        activitiesTaken_df = df[
            df["activityName"] == "Stand with your feet side-by-side"
        ]
        firstDate = activitiesTaken_df["date_set"].min()
        today = date.today()

        idx = pd.date_range(firstDate, today)
        idx = idx[-7:]
        s = activitiesTaken_df.groupby(["date_set"]).size()
        s = s.reindex(idx, fill_value=0)
        fig, ax = plt.subplots()
        plt.xticks(rotation=90)
        plt.yticks([0, 1])

        ax.bar(idx.to_pydatetime(), s, color="red")
        ax.set_title("Dates activities taken last week", fontsize=18,
color="#8C55AA")

        fig.savefig("static/images/fig.png")

        fig = px.bar(
            df,
            x="activityName",
            color="completed",
            barmode="group",
            text="date_set",
            title="Activities completed",
            labels=dict(count="Activities carried out"),
        )
        fig.write_image("static/images/fig1.png")

        percentages = calculate_percentages(df)

        sunburst = px.sunburst(df, path=["activityName", "date_set",
"completed"])
        sunburst.write_image("static/images/fig4.png")

        df = df.sort_values(by="date_set", ascending=False)
        df = df[
            [
                "activityName",
                "date_set",
                "max_value",
                "min_value",
                "avg_value",
                "completed",
            ]
        ]
        last_date = df["date_set"].max()
        lastActivity = df[df["date_set"] == last_date]
        last_week = df[df["date_set"] > (today - timedelta(7))]
        last_month = df[df["date_set"] > (today - timedelta(28))]
```

```
        if request.method == "POST":
            details = request.form
            comment = details["comment_made"]
            if comment == "":
                if details["results"] == "row_data_lastweek":
                    row_data = list(last_week.values.tolist())
                elif details["results"] == "row_data":
                    row_data = list(lastActivity.values.tolist())
                elif details["results"] == "row_data_lastmonth":
                    row_data = list(last_month.values.tolist())
                else:
                    row_data = list(df.values.tolist())
            else:
                if "user_email" in session:
                    user_email = session["user_email"]
                add_comment(details, user_email, "General comments")
                row_data = list(lastActivity.values.tolist())
        else:
            row_data = list(lastActivity.values.tolist())

        return render_template(
            "view_activity_progress.html",
            activities=activities,
            patient_scores=patient_scores,
            column_names=df.columns.values,
            row_data=row_data,
            percentages=percentages,
            lastActivity=last_date,
        )

    else:
        flash("You must be logged in to access webpage.", "error")
        return redirect(url_for("login"))


def create_activity_rows(patient_scores):
    """
    create activity rows function

    Parameters
    -------------
    patient scores : List of users balance scores retrieved from database

    Returns
    ------------
    List containing each of the activities taken separated into maps
    """
    rows = []
    for data in patient_scores:
        if "Stand with your feet side-by-side" in data.keys():
            data_row = data["Stand with your feet side-by-side"]
            rows.append(data_row)
        if "Tandem Stance" in data.keys():
            data_row1 = data["Tandem Stance"]
            rows.append(data_row1)
        if "Instep Stance" in data.keys():
            data_row2 = data["Instep Stance"]
            rows.append(data_row2)
        if "Stand on one foot" in data.keys():
            data_row3 = data["Stand on one foot"]
```

```
            rows.append(data_row3)
    return rows


def calculate_percentages(df):
    """
    calculte percentages function

    Parameters
    -------------
    df : dataframe containing users performance scores

    Returns
    ------------
    List contaning the perrcantages of completed exercises for each actvity
    contained in the dataframe calculated
    """
    numInstepSuccess = df[
        (df["completed"] == True) & (df["activityName"] == "Instep Stance")
    ].shape[0]
    numInstep = df[(df["activityName"] == "Instep Stance")].shape[0]

    numTademSuccess = df[
        (df["completed"] == True) & (df["activityName"] == "Tandem Stance")
    ].shape[0]
    numTandem = df[(df["activityName"] == "Tandem Stance")].shape[0]

    numFeetTogetherSuccess = df[
        (df["completed"] == True)
        & (df["activityName"] == "Stand with your feet side-by-side")
    ].shape[0]
    numFeetTogether = df[
        (df["activityName"] == "Stand with your feet side-by-side")
    ].shape[0]

    numOneFootSuccess = df[
        (df["completed"] == True) & (df["activityName"] == "Stand on one
foot")
    ].shape[0]
    numOneFoot = df[(df["activityName"] == "Stand on one foot")].shape[0]

    percentages = [
        (round(((numFeetTogetherSuccess / numFeetTogether) * 100), 2)),
        (round(((numInstepSuccess / numInstep) * 100), 2)),
        (round(((numTademSuccess / numTandem) * 100), 2)),
        (round(((numOneFootSuccess / numOneFoot) * 100), 2)),
    ]
    return percentages


@app.route("/view_selected_activity/<activity>", methods=["GET", "POST"])
def view_selected_activity(activity):
    """
   view selected activity function

   Parameters
   -------------
   activty : name of selected activity

   GET - displays view selected activity webpage for selected activity
      retirves the selected users balance performance for selected activity
```

```
and dispalys
      the results in graph and table format
   POST - adds any cooments made by the medical staff to the database
      refreshes page
   """
    if user["is_logged_in"] == True:
        if request.method == "POST":
            if "user_email" in session:
                user_email = session["user_email"]
            comments = request.form
            add_comment(comments, user_email, activity)

        activities = get_activities()
        activities = [i for i in activities if not (i["name"] == activity)]
        dict1 = {"name": "Overall"}
        activities.append(dict1)
        if "user_email" in session:
            user_email = session["user_email"]
        patient_scores = get_patient_scores(user_email)
        rows = create_activity_rows(patient_scores)
        df = pd.DataFrame(rows)
        df = df[df["activityName"] == activity]

        fig2 = px.sunburst(
            df.head(7),
            path=["date_set", "completed"],
            hover_name="activityName",
            color="completed",
        )
        fig2.write_image("static/images/fig3.png")

        fig = px.line(
            df,
            x="date_set",
            y=["avg_value", "max_value"],
            title="Overall Average Score",
        )
        fig.write_image("static/images/fig2.png")

        i = 0
        fig = Figure()
        font1 = {"family": "serif", "color": "blue", "size": 10}
        fig = plt.figure(figsize=(18, 16))
        for index, row in df.head(7).iterrows():
            i = i + 1

            y = np.array(row["acc_data"])
            plt.subplot(3, 3, i)
            if row["completed"] is False:
                plt.plot(y, color="r")
            else:
                plt.plot(y)  # Plot the chart

            plt.title(activity + row["date_set"], fontdict=font1)
            plt.xlabel("Time")
            plt.ylabel("Movement")

        pngImage = io.BytesIO()
        FigureCanvas(fig).print_png(pngImage)
        pngImageB64String = ""
        pngImageB64String +=
```

```
base64.b64encode(pngImage.getvalue()).decode("utf8")

        return render_template(
            "view_selected_activity.html",
            activities=activities,
            image=pngImageB64String,
        )

    else:
        flash("You must be logged in to access webpage.", "error")
        return redirect(url_for("login"))


def validate_register_details(data):
    """
    validate register details

    Parameters
        -------------
        data : entered user details when registering

    validates that the enteerd password and confirmed password match
    """
    if data["confirm_password"] != data["password"]:
        flash("Passwords do not match.", "error")
        return False
    return True


if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

## *Data_utils.py*

```
"""
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
data_utils.py
contains methods for connecting to and communicating with firestore
database
"""
import firebase_admin
from firebase_admin import credentials
from firebase_admin import firestore
import pyrebase
from flask import flash
import json
from flask import session
from models.patient import Patient
import config as cfg
from datetime import date, timedelta, datetime

cred = credentials.Certificate("firebase_sdk.json")
firebase_admin.initialize_app(cred)
db = firestore.client()
```

```
firebaseConfig = cfg.firebaseConfig
firebase = pyrebase.initialize_app(firebaseConfig)
storage = firebase.storage()
# storage = getStorage(firebase_admin);
auth = firebase.auth()


def register_user(user_details):
    """
    register a user using firbase authentication

    Parameters
    -------------
    user_details : Entered user details

    Displays a message if the user was registered successfully or not
    """
    try:
        user = auth.create_user_with_email_and_password(
            user_details["email"], user_details["password"]
        )
        register_medical_staff(user["localId"], user_details)
        flash("SUCCESSFULLY REGISTERED USER.", "success")
        return user
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")
        return None


def register_medical_staff(userId, user_details):
    """
    adss a newly register user details to the firestore database

    Parameters
    -------------
    user_id : created user firebase UID
    user_details : Entered user details

    Displays a message if the user details were added successfully or not
    """
    try:
        doc_ref = db.collection(u"medical_staff").document(userId)
        doc_ref.set(
            {
                u"firstname": user_details["first_name"],
                u"lastname": user_details["last_name"],
                u"email": user_details["email"],
                u"userUid": userId,
            }
        )
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def login_user(user_details):
    """
    logs in a user using firbase authentication

    Parameters
    -------------
```

```
    user_details : Entered user login details

    Displays an error message if the user details do not match the database
entry
    """
    try:
        login = auth.sign_in_with_email_and_password(
            user_details["email"], user_details["password"]
        )
        doc_ref =
db.collection(u"medical_staff").document(login["localId"])
        doc = doc_ref.get()
        if doc.exists:
            return login
        else:
            flash("Could not authenticate user", "error")
            return None

    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")
        return None


def add_activity(activity_details):
    """
    adds a balance activity details to the database

    Parameters
    -------------
    activity_details : Entered balance activity details

    Displays a message if the activity details were added successfully or
not
    """
    try:
        doc_ref = db.collection(u"activities").add(
            {
                u"name": activity_details["activity_name"],
                u"description": activity_details["description"],
                u"time_limit": int(activity_details["time_limit"]),
            }
        )
        flash("SUCCESSFULLY Added activity.", "success")
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def add_comment(comment, email, activity):
    """
    adds a comment for a patinets balance activity performance to the
database

    Parameters
    -------------
    comment : comment to be stored
    email : patients email address
    activity : name of activity to be commented on

    Displays a message if the activity comment added was unsuccessful
    """
    try:
```

```
        today = date.today().strftime("%Y-%m-%d")
        doc_ref = (
            db.collection(u"comments")
            .document(email)
            .collection(activity)
            .add(
                {u"comment": comment["comment"], u"date": today,
u"activity": activity,}
            )
        )
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def retrieve_comments(activity, email):
    """
    retrieves comments made for a patinets balance activity performance
from the database

    Parameters
    -------------
    email : patients email address
    activity : name of activity to retrieve comments from

    Displays a message if retrieving activity comments was unsuccessful
    """
    try:
        comments = []
        docs = (
            db.collection(u"comments")
            .document(email)
            .collection(activity)
            .order_by(u"date", direction=firestore.Query.DESCENDING)
            .stream()
        )
        for doc in docs:
            comment = doc.to_dict()
            comments.append(comment)
        return comments
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def get_activities():
    """
    retrieves activities from the database

    Displays a message if retrieving activity comments was unsuccessful
    """
    try:
        docs = db.collection(u"activities").stream()
        activities = []
        for doc in docs:
            activity = doc.to_dict()
            activities.append(activity)
        return activities
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def add_patient(user_details):
```

```
    """
    adds a new patient details to the database

    Parameters
    -------------
    user_details : Entered patient details

    Displays a message if the patient details were added successfully or
not
    """
    try:
        userid = session["userId"]
        doc_ref = (
            db.collection(u"patients")
            .document(userid)
            .collection(u"patient_details")
            .document(user_details["email"])
            .set(
                {
                    u"firstname": user_details["first_name"],
                    u"lastname": user_details["last_name"],
                    u"email": user_details["email"],
                    u"D.O.B": user_details["age"],
                    u"condition": user_details["condition"],
                }
            )
        )
        flash("SUCCESSFULLY Added/Updated patient.", "success")
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def add_activities(email):
    """
    adds activities for a patient to the database

    Parameters
    -------------
    email : email address of patient

    Displays a message if the patient activity was added unsuccessfully
    """
    activities = get_activities()
    for a in activities:
        try:
            doc_ref = (
                db.collection(u"patient_activities")
                .document(email)
                .collection(u"activities")
                .document(a["name"])
                .set(
                    {
                        u"name": a["name"],
                        u"description": a["description"],
                        u"time_limit": int(a["time_limit"]),
                    }
                )
            )
        except Exception as e:
            flash(json.loads(e.args[1])["error"]["message"], "error")
```

```
def get_patients():
    """
    retrieves patient lists related to the logged in medical personnel from
the database

    Displays a message if retrieving patients was unsuccessful
    """
    try:
        userid = session["userId"]
        docs = (
            db.collection(u"patients")
            .document(userid)
            .collection(u"patient_details")
            .stream()
        )
        patients = []
        for doc in docs:
            patient = doc.to_dict()
            patients.append(patient)
        return patients
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def get_patient(email):
    """
    retrieves a selected patients details from the database

    Parameters
    -------------
    email : email address of patient

    Displays a message if request was successful or not
    """
    try:
        userid = session["userId"]
        doc_ref = (
            db.collection(u"patients")
            .document(userid)
            .collection(u"patient_details")
            .document(email)
        )

        doc = doc_ref.get()
        if doc.exists:
            return doc.to_dict()
        else:
            flash("No record found for patient.", "error")
            return None
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")
        return None


def get_patient_activities(email):
    """
    retrieves activities set for a patient from the database

    Parameters
    -------------
```

```
    email : email address of patient

    Displays a message if the request was unsuccessful
    """
    try:
        docs = (
            db.collection(u"patient_activities")
            .document(email)
            .collection(u"activities")
            .stream()
        )
        patient_activities = []
        for d in docs:
            activity = d.to_dict()
            patient_activities.append(activity)
        return patient_activities
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def get_patient_scores(email):
    """
    retrieves a selected patients activity scores from the database

    Parameters
    -------------
    email : email address of patient

    Displays a message if request was unsuccessful
    """
    try:
        docs = (
            db.collection(u"patient_scores")
            .document(email)
            .collection(u"scores")
            .stream()
        )
        patient_scores = []
        for d in docs:
            score = d.to_dict()
            patient_scores.append(score)
        return patient_scores
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def add_patient_activity(activity_details, email):
    """
    adds activities for a patient to the database

    Parameters
    -------------
    activity_details : entered details for activity
    email : email address of patient

    Displays a message if the request was unsuccessful
    """
    try:
        doc_ref = (
            db.collection(u"patient_activities")
            .document(email)
```

```
            .collection(u"activities")
            .document(activity_details["name"])
            .set(
                {
                    u"name": activity_details["name"],
                    u"description": activity_details["description"],
                    u"time_limit": int(activity_details["time_limit"]),
                }
            )
        )
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")


def get_activity_results(activity, email):
    """
    retrieves a selected patients activity results from the database

    Parameters
    -------------
    actvity : the name of the actvity
    email : email address of patient

    Displays a message if request was unsuccessful
    """
    try:
        docs = (
            db.collection(u"patient_activities")
            .document(email)
            .collection(u"activities")
            .document(activity)
            .collection(u"scores")
            .stream()
        )
        scores = []
        for doc in docs:
            activity = doc.to_dict()
            scores.append(activity)
        return scores
    except Exception as e:
        flash(json.loads(e.args[1])["error"]["message"], "error")
```

## Templates

### Base.html

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
Base.html this file contains the base structure that will be inherited by
all other pages
-->"""
<!DOCTYPE html>
<html>
    <head>
    <title>Balance Health</title>
```

```
    <link rel="stylesheet" type="text/css" href="/static/css/nav_bar.css"/>
        <link rel="stylesheet" type="text/css"
href="/static/css/style.css">

        <script type="text/javascript"
src="/static/js/add_details.js"></script>
    </head>

    <body>
    <div>
   {% block navbar %}
      {% include '_navbar.html' %}
   {% endblock %}
   </div>

   <div>
        {% block content %}
      {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
         <ul class ="error_messages">
         {% for category, message in messages %}
         <li class="{{category}}" >{{ message }}</li>
         {% endfor %}
         </ul>
      {% endif %}
      {% endwith %}

      {% block sub_content %}
      {% endblock %}
   {% endblock %}
   </div>

    </body>
</html>
```

*_navbar.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
_navbar.html - html page for adding navigation bar for each page in the
application
-->"""
<ul class = "nbar">
  <li class="nbarlist"><a class="navbar-brand" href="#">Balance
Health</a></li>

  <li class="dropdown">
    <a href="javascript:void(0)" class="dropbtn">Patient</a>
    <div class="dropdown-content">

      <a href="/view_patients">View Patients</a>
      <a href="/create_patient">Create Patient</a>
      <a href="/edit_patient">Edit Patient</a>
    </div>
  </li>

   <li class="dropdown">
    <a href="javascript:void(0)" class="dropbtn">Activity</a>
    <div class="dropdown-content">
```

```
      <a href="/create_activity">Create Activity</a>
      <a href="/view_activities">View Activities</a>
    </div>
  </li>
  <li class="nbarlist" style="float:right"><a href="/logout">Logout</a>
</ul>
```

*Register.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
register.html - html page for registering a medical personnel on the
database
-->"""
<!DOCTYPE html>
<html>
    <head>
    <title>Balance Health</title>
    <link rel="stylesheet" type="text/css" href="/static/css/nav_bar.css"/>
        <link rel="stylesheet" type="text/css"
href="/static/css/style.css">

        <script type="text/javascript"
src="/static/js/add_details.js"></script>
    </head>

    <body>

   <div>
        {% block content %}
      {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
         <ul class ="error_messages">
         {% for category, message in messages %}
         <li class="{{category}}" >{{ message }}</li>
         {% endfor %}
         </ul>
      {% endif %}
      {% endwith %}

      {% block sub_content %}

   <center>
     <h1> Register Account </h1>


   <div style="text-align: center;">

   <div class="form">
   <form class="addform" method="POST">

      <label class="inputbox" for="fname">First Name</label>
      <input class="input_add" type="text" id="first_name"
name="first_name" placeholder="Your name.." required>

      <label class="inputbox" for="lname">Last Name</label>
      <input class="input_add" type="text" id="last_name" name="last_name"
placeholder="Your last name.." required>
```

```
      <label class="inputbox" for="email">Email</label>
      <input class="input_add" type="email" id="email" name="email"
placeholder="Email.." required>

      <label class="inputbox" for="pass">Password</label>
      <input class="input_add" type="password" minlength="6" id="pass"
name="password" required>

      <label class="inputbox" for="confirm_pass">Confrim Password</label>
      <input class="input_add" type="password" minlength="6"
id="confirm_pass" name="confirm_password" required>

      <div >
      <input class="submit_button" type="submit" value="Submit">
      </div>
      <div>

      <a href="{{ url_for('login') }}">Login</a>
         </div><br><br>
   </form>
   </div>

   </div>

   </center>
   {% endblock %}
   {% endblock %}
   </div>

    </body>
</html>
```

*Login.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
login.html - html page for logging in a user to the application
-->"""
<!DOCTYPE html>
<html>
    <head>
    <title>Balance Health</title>
    <link rel="stylesheet" type="text/css" href="/static/css/nav_bar.css"/>
        <link rel="stylesheet" type="text/css"
href="/static/css/style.css">

        <script type="text/javascript"
src="/static/js/add_details.js"></script>
    </head>

    <body>

   <div>
        {% block content %}
      {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
         <ul class ="error_messages">
         {% for category, message in messages %}
```

```
        <li class="{{category}}" >{{ message }}</li>
        {% endfor %}
        </ul>
      {% endif %}
      {% endwith %}

      {% block sub_content %}
   <center>
     <h1> Sign In </h1>


   <div style="text-align: center;">

   <div class="form">
   <form method="POST" class="addform" >


      <label class="inputbox" for="email">Email</label>
      <input class="input_add" type="email" id="email" name="email"
placeholder="Email.." required>

      <label class="inputbox" for="pass">Password</label>
      <input class="input_add" type="password" minlength="6" id="pass"
name="password" required>

       <div>

      <a href="{{ url_for('register') }}">No Account? Register</a>
        </div><br><br>

      <div >
      <input class="submit_button" type="submit" value="Submit">
      </div>
   </form>
   </div>

   </div>

   </center>
   {% endblock %}
   {% endblock %}
   </div>

    </body>
</html>
```

*Welcome.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
welcome.html - html page for displaying a welcome page
-->"""
{%extends "base.html" %}

{% block sub_content %}
   <center>
   <h1> Balance Health  </h1>

   <div>
```

```
      <img
src="static/images/balance.jpeg"style="width:60%;height:30%"><br><br>
   </div>
   </center>
{% endblock %}
```

*Create_activity.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
crete_actvity.html - html page for adding an activity to database
-->"""
{%extends "base.html" %}

{% block sub_content %}

<script  type="text/javascript">
   function confirmCheck(){
      var response ;
      var name = document.getElementById("activity_name").value;
      response = confirm('Confirm: Add activity ' + name) ;

      if(response){//if user selects yes pass details to php file
         return true ;
               }
      else
         {//if user selects no return to screen
          return false;
         }
      }
</script>

<center>
  <h1> Add Activity </h1>


   <div style="text-align: center;">

   <div class="form">
   <form class="addform" onsubmit = "return confirmCheck();" method="POST"
>

      <label class="inputbox" for="activity_name">Name</label>
      <input class="input_add" type="text" id="activity_name"
name="activity_name" placeholder="activty name" required>

      <label class="inputbox" for="description">Description</label>
      <div>
      <textarea id="description" name="description" rows="10" cols="60"
placeholder="Activity description" required>
      </textarea>
      </div>

      <label class="inputbox" for="time_limit">Time Limit</label>
      <input class="input_add" type="text" id="time_limit"
name="time_limit" placeholder="time in seconds" required>

      <div >
      <input class="submit_button" type="submit" value="Submit">
```

```
      </div>
      <div>

   </form>
   </div>

   </div>

   </center>
{% endblock %}
```

*View_activities.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
view_activities.html - html page for displaying the activities stored on
the database
-->"""
{%extends "base.html" %}

{% block sub_content %}

<center>

<h1>Activity List</h1>


{% if data %}
<div>
<table class = "patient_list" style="width: 60%;" >
   <tr>
      <th>Name</th><th>Description</th><th>Time Limit</th>
   </tr>


{% for row in data %}
   <tr>

      <td>{{ row['name'] }}</td>
      <td>{{ row['description'] }}</td>
      <td>{{ row['time_limit'] }}</td>

   </tr>
{% endfor %}
</table>
</div>
{% else %}
<br><br>
<p>No actviities currently exist!</p>

{% endif %}

</center>

{% endblock %}
```

*Create_patient.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
crete_patient.html - html page for adding a patient to the database
-->"""
{%extends "base.html" %}

{% block sub_content %}

<script  type="text/javascript">
   function confirmCheck(){
      var response ;
      var fname = document.getElementById("first_name").value;
      var lname = document.getElementById("last_name").value;
      response = confirm('Confirm: Add patient ' + fname + " " + lname) ;

      if(response){//if user selects yes pass details to php file
         return true ;
               }
      else
         {//if user selects no return to screen
          return false;
         }
      }
</script>

<center>
   <h1> Add Patient</h1>


   <div style="text-align: center;">

   <div class="form">
   <form class="addform" onsubmit = "return confirmCheck();" method="POST">

      <label class="inputbox" for="fname">First Name</label>
      <input class="input_add" type="text" id="first_name"
name="first_name" placeholder="patient name.." required>

      <label class="inputbox" for="lname">Last Name</label>
      <input class="input_add" type="text" id="last_name" name="last_name"
placeholder="patient last name.." required>

      <label class="inputbox" for="age">Date of Birth</label>
      <input class="input_add" type="date" id="age" name="age"  required>

      <label class="inputbox" for="email">Email</label>
      <input class="input_add" type="email" id="email" name="email"
placeholder="contact email.." required>

      <label class="inputbox" for="condition">Condition</label>
      <input class="input_add" type="text" id="condition" name="condition"
placeholder="patients condition..." required><br>

      <!--<label class="inputbox" for="image">Image Upload</label>
      <input class="input_add" type="file" name="myimage">
      <div >-->
      <input class="submit_button" type="submit" value="Submit">
      </div><br><br>
```

```
   </form>
   </div>

   </div>

</center>
{% endblock %}
```

*Delete_patient.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
delete_patient.html - html page for removing a patient from the database
-->"""
{%extends "base.html" %}

{% block sub_content %}

<center>

<h1>Delete Patient</h1>


{% if data %}
<div>
<table class = "patient_list">
   <tr>
      <th>Firstname</th><th>Lastname</th><th>Date of Birth</th>
      <th>Email</th><th>Condition</th>
   </tr>


{% for row in data %}
   <tr>
      <td>{{ row['firstname'] }}</td>
      <td>{{ row['lastname'] }}</td>
      <td>{{ row['D.O.B'] }}</td>
      <td>{{ row['email'] }}</td>
      <td>{{ row['condition'] }}</td>
      <td>
         <form name="patient_details" method="POST">
         <input type="hidden" name="user_email" value="{{ row['email'] }}"
/>
         <input class = "table_button" id="select" type="submit"
name="select" value="Delete">

         </form>
      </td>
   </tr>
{% endfor %}
</table>
</div>
{% else %}
<br><br>
<p>No patients currently exist!</p>

{% endif %}
```

```
</center>

{% endblock %}
```

*Edit_patient.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
edit_patient.html - html page for editing a patient details in the database
-->"""
{%extends "base.html" %}

{% block sub_content %}

<script  type="text/javascript">
   function confirmCheck(){
      var response ;
      var fname = document.getElementById("first_name").value;
      var lname = document.getElementById("last_name").value;
      response = confirm('Confirm: Edit patient ' + fname + " " + lname) ;

      if(response){//if user selects yes pass details to php file
         return true ;
               }
      else
         {//if user selects no return to screen
          return false;
         }
      }
</script>

<center>
   <h1> Edit Patient</h1>


   <div style="text-align: center;">
   <form method="POST" >
   <select name="results" id="results">
   {% for row in data %}
      <option value="{{ row['email']  }}">{{ row['firstname']  }} {{
row['lastname']  }}</option>
      {% endfor %}
     </select>
     <input type="submit" value="Submit">
   </form>

   <div class="form">
   <form class="addform" onsubmit = "return confirmCheck();" method="POST">
   {% if patient_details is none %}
      <label class="inputbox" for="fname">First Name</label>
      <input class="input_add" type="text" id="first_name"
name="first_name" placeholder="patient name.." disabled>

      <label class="inputbox" for="lname">Last Name</label>
      <input class="input_add" type="text" id="last_name" name="last_name"
placeholder="patient last name.." disabled>

      <label class="inputbox" for="age">Date of Birth</label>
      <input class="input_add" type="date" id="age" name="age" disabled>
```

```
      <label class="inputbox" for="email">Email</label>
      <input class="input_add" type="email" id="email" name="email"
placeholder="contact email.." disabled>

      <label class="inputbox" for="condition">Condition</label>
      <input class="input_add" type="text" id="condition" name="condition"
placeholder="patients condition..." disabled>



      {% else %}
      <label class="inputbox" for="fname">First Name</label>
      <input class="input_add" type="text" id="first_name"
name="first_name" value="{{ patient_details['firstname'] }}" >

      <label class="inputbox" for="lname">Last Name</label>
      <input class="input_add" type="text" id="last_name" name="last_name"
value="{{ patient_details['lastname'] }}" >

      <label class="inputbox" for="age">Date of Birth</label>
      <input class="input_add" type="date" id="age" name="age"  value="{{
patient_details['D.O.B'] }}">

      <label class="inputbox" for="email">Email</label>
      <input class="input_add" type="email" id="email" name="email"
value="{{ patient_details['email'] }}" >

      <label class="inputbox" for="condition">Condition</label>
      <input class="input_add" type="text" id="condition" name="condition"
value="{{ patient_details['condition'] }}" >
      <input type="hidden" name="results" value= "{{ row }}" />
      {% endif %}
      <div >
      <input class="submit_button" type="submit" value="Submit">
      </div><br><br>

   </form>
   </div>

   </div>

</center>
{% endblock %}
```

*View_patients.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
view_patients.html - html page for dispalying a medical personnels
patients list retrieved from the database
-->"""
{%extends "base.html" %}

{% block sub_content %}

<center>

<h1>Patient List</h1>
```

```
{% if data %}
<div>
<table class = "patient_list">
   <tr>
      <th>Firstname</th><th>Lastname</th><th>Date of Birth</th>
      <th>Email</th><th>Condition</th>
   </tr>


{% for row in data %}
   <tr>
      <td>{{ row['firstname'] }}</td>
      <td>{{ row['lastname'] }}</td>
      <td>{{ row['D.O.B'] }}</td>
      <td>{{ row['email'] }}</td>
      <td>{{ row['condition'] }}</td>
      <td>
         <form name="patient_details" method="POST">
         <input type="hidden" name="user_email" value="{{ row['email'] }}"
/>
         <input class = "table_button" id="select" type="submit"
name="select" value="View">

         </form>
      </td>
   </tr>
{% endfor %}
</table>
</div>
{% else %}
<br><br>
<p>No patients currently exist!</p>

{% endif %}

</center>

{% endblock %}
```

## *Patient_details.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
patient_details.html - html page for retrieving and displaying a patients
details
-->"""
{%extends "base.html" %}

{% block sub_content %}

<script  type="text/javascript">
   function confirmCheck(){
      var response ;
      response = confirm('Confirm: Add activity for patient') ;
      if(response){//if user selects yes pass details to php file
         return true ;
               }
```

```
      else
         {//if user selects no return to screen
          return false;
         }
      }
</script>

<center>
  <h1> {{ data['firstname'] }}  {{ data['lastname'] }} </h1>
</center>

<div style="width: 100%;overflow:auto;">

<center>
<div style="float:left; width: 40%;">

   <form>
   <div class="text-center">
      <h1> Patient Details </h1>
   </div>

   <div class="col-25">
   <label class="inputbox2" for="fname">First Name :</label>
   </div>
   <div class="col-75">
   <input class="input_view" type="text" id="first_name" name="first_name"
value="{{ data['firstname'] }}" disabled ><br>
   </div>

   <div class="col-25">
   <label class="inputbox2" for="lname">Last Name :</label>
   </div>
   <div class="col-75">
   <input class="input_view" type="text" id="last_name" name="last_name"
value="{{ data['lastname'] }}" disabled ><br>
   </div>

   <div class="col-25">
   <label class="inputbox2" for="age">Age :</label>
   </div>
   <div class="col-75">
   <input class="input_view" type="text" id="age" name="age" value="{{
data['D.O.B'] }}" disabled><br>
   </div>

   <div class="col-25">
   <label class="inputbox2" for="email">Email :</label>
   </div>
   <div class="col-75">
   <input class="input_view" type="email" id="email" name="email" value="{{
data['email'] }}" disabled ><br>
   </div>

   <div class="col-25">
   <label class="inputbox2" for="condition">Condition :</label>
   </div>
   <div class="col-75">
   <input class="input_view" type="text" id="condition" name="condition"
value="{{ data['condition'] }}" disabled ><br>
   </div>
```

```
</form>

</div>
</center>


<center>
<div style="float: right;width: 60%;">
       <div class="text-center">
      <h1> Activities </h1>
   </div>



   <div>
   {% if patient_activities %}
   <center>
   <table class = "patient_list" style="width: 80%;">
   <tr>
      <th>Name</th><th>Description</th><th>Time Limit</th>
   </tr>
   {% for row in patient_activities %}
   <tr>
      <td>{{ row['name'] }}</td>
      <td>{{ row['description'] }}</td>
      <td>{{ row['time_limit'] }}</td>
   </tr>
   {% endfor %}

   </table>
   {% else %}
   <br><br>
   <p>No activities currently exist!</p>

   {% endif %}
   </center>

   </div>

</div>
</div>
<center>
<div class="text-center">
      <h1> Activities Comments </h1>
   </div>

   <form method="POST" >
<select name="activity" id="activity">
   <option value="general">General Comments</option>
    <option value="feetTogether">Stand with your feet side-by-side</option>
    <option value="instep">Instep Stance</option>
    <option value="tandem">Tandem Stance</option>
    <option value="oneFoot">Stand on one foot</option>
  </select>
  <input type="submit" value="Submit">
</form>
  <br><br>
  </div>
</center>

<center>
```

```
<div>
{% if patient_comments %}
<div class="scrollWrapper">
<table class = "patient_list">
<tbody style = "height=600px">
    <tr>

       <th>Activity Name</th><th>Comment</th><th>Date set</th>
    </tr>

     {% for row in patient_comments %}
     <tr>
          <td>{{ row['activity'] }}</td>
      <td>{{ row['comment'] }}</td>
      <td>{{ row['date'] }}</td>

   </tr>
   {% endfor %}
</table><br><br><br><br>
{% else %}
   <br><br>
   <p>No comments left for this activity!</p>

   {% endif %}
   </div>
<div>
   <a href="{{ url_for('view_activity_progress') }}"><button
class="submit_button">Balance Progress</button></a>
</div>
</center>
{% endblock %}
```

*View_activity_progress.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
view_activity_progress.html - html page for dispalying a patients balance
performance results retrieved from the database
-->"""
{%extends "base.html" %}

{% block sub_content %}
<div style="width: 100%;overflow:auto;">


<div class="sidebar" style="float:left; width:15%">
   {% if activities %}
      {% for row in activities %}
         <a href="{{ url_for('view_selected_activity',
activity=row['name']) }}" style="color: #8C55AA">{{ row['name']  }}</a>

      {% endfor %}
   {% endif %}
</div>

<center>
<div style="float: right;width: 85%;">

   {% if not activities %}
```

```
      <br><br><br><br>
      <p>No activities have been set for patient</p>
   {% endif %}

   <h1> Overall Performance</h1>

   <div>
      <img src="static/images/fig.png"style="width:40%;height:30%"><br><br>
      <label style="color: #8C55AA;" for="lastActivity">Date Last Activity
Taken       :     </label> <label>{{ lastActivity }}</label><br><br>
   </div><br><br>

   <h1> Previous Activity Success Rate</h1>
   <div>

      <img
src="static/images/fig1.png"style="width:60%;height:50%"><br><br>
      <table class = "success_table">
         <tr>
                <th>Activity</th>
                <th>Success Rate</th>
         </tr>
         <tr>
                 <td>Stand with your feet side-by-side</td>
                 <td>{{ percentages[0] }}%</td>
         </tr>
         <tr>
                 <td>Instep Stance</td>
                 <td>{{ percentages[1] }}%</td>
         </tr>
         <tr>
                 <td>Tandem Stance</td>
                 <td>{{ percentages[2] }}%</td>
         </tr>
         <tr>
                 <td>Stand on one foot</td>
                 <td>{{ percentages[3] }}%</td>
         </tr>
      </table>
   </div><br><br>

   <div>
      <h1> Previous Results</h1>

      <img src="static/images/fig4.png"style="width:60%;height:40%">

      <form method="POST" >
         <select name="results" id="results">
            <option value="row_data">Last Activity</option>
            <option value="row_data_lastweek">Last Weeks</option>
            <option value="row_data_lastmonth">Last Month</option>
            <option value="row_data_all">All</option>
         </select>
         <input type="hidden" name="comment_made" value= "" />
         <input type="submit" value="Submit">
      </form><br><br>
   </div>
   <div class="scrollWrapper">
      <table class = "success_table">
         <tbody style = "height=400px">
             <tr>
```

```
            <th>Activity Name</th><th>Date set</th><th>Max Value</th>
            <th>MIn Value</th><th>Avg Value</th><th>Completed</th>
         </tr>

             {% for row in row_data %}
            {% if False in row %}
               <tr style ="background-color: red">
            {% else %}
               <tr>
            {% endif %}
                {% for item in row %}
                   <td> {{ item }} </td>
            {% endfor %}
         {% endfor %}
         </tr>

      </table><br><br><br><br>
   </div>

   <div style="text-align: center;">
      <center>
      <form method="POST">
         <label class="inputbox" for="comment" style=" vertical-align:
top;">Comments
         <div>
            <textarea id="comment" name="comment" rows="10" cols="60"
placeholder="Activity comments" required>
            </textarea></label>
            <input type="hidden" name="comment_made" value= "comment" />
            <input type="submit" value="Submit">
         </div>

      </form>
   </div>

</div>
</div>
{% endblock %}
```

## *View_selected_activity.html*

```
"""<!--
Name : Diarmuid Brennan
Project : Balance Health Web Application
Date : 05/04/2022
view_selected_activity.html - html page for displaying the results of a
selected activity retrieved from the database
-->"""
{%extends "base.html" %}

{% block sub_content %}
<div class="sidebar" style="width:15%">
   {% if activities %}
      {% for row in activities %}
         {% if row['name'] == "Overall" %}
            <a href="{{ url_for('view_activity_progress') }}" style="color:
#8C55AA">{{ row['name'] }}</a>
         {% else %}
            <a href="{{ url_for('view_selected_activity',
activity=row['name']) }}" style="color: #8C55AA">{{ row['name'] }}</a>
         {% endif %}
```

```
      {% endfor %}
   {% endif %}
</div>

<center>
<div>
   <h1>Overall Average Scores</h1>
   <img src="/static/images/fig2.png"style="width:50%;height:30%"><br><br>
</div>

<div>
   <h1>Previous Completed Actviities</h1>
   <img src="/static/images/fig3.png"style="width:50%;height:30%"
loading="lazy"><br><br>
</div>

<div>
   <h1>Previous Activities Movements</h1>
   <img src="{{ image }}" style="width:60%;height:50%"><br><br>
</div>

<div style="text-align: center;">
   <center>
   <form method="POST">
      <label class="inputbox" for="comment" style=" vertical-align:
top;">Comments
      <div>
         <textarea id="comment" name="comment" rows="10" cols="60"
placeholder="Activity comments" required>
         </textarea></label>
         <input type="submit" value="Submit">
      </div>

   </form>
   </center>
</div>
</center>

{% endblock %}
```