VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V. Puram, Bengaluru-560 004

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MOBILE APPLICATION DEVELOPMENT
MINI PROJECT (18CSMP68)

Fast N Fitness

Submitted By

1BI20CS153	**SATYAM KUMAR**

for the academic year 2022-2023

Under the guidance of

Dr. Bhanushree K J	**Prof. Manjushree N S**	**Prof. Shruthi B Gowda**
Associate Professor	**Assistant Professor**	**Assistant Professor**
Dept. of CSE, BIT	**Dept. of CSE,BIT**	**Dept. of CSE,BIT**

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
“Jnana Sangama”, Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V. Puram, Bengaluru-560 004

Department of Computer Science & Engineering

Certificate

This is to certify that the implementation of **Mobile Application Development MINI PROJECT (18CSMP68)** entitled “ **Fast N Fitness ”** has been completed by

1BI20CS153	**SATYAM KUMAR**

of VI semester B.E. for the partial fulfilment of the requirements for the Bachelor's degree in **Computer Science & Engineering** of the **Visvesvaraya Technological University** during the academic year **2022-2023**.

Lab In charges:

Dr. Bhanushree K J
Associate Professor
Dept. of CSE, BIT

Dr. Girija J.
Professor and Head
Dept. of CSE, BIT

Examiners: 1) 2)

ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned my effort with success. I would like to thank all and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my thanks to Head of Department **Dr. Girija J.** for being kind enough to provide the necessary support to carry out the mini project.

I am most humbled to mention the enthusiastic influence provided by the lab in-charges **Dr. Bhanushree K J, Prof. Manjushree N S, Prof Shruti B Gowda** on the project for their ideas, time to time suggestions for being a constant guide and co-operation showed during the venture and making this project a great success.

I would also take this opportunity to thank my friends and family for their constant support and help. I'm very much pleasured to express my sincere gratitude to the friendly co-operation showed by all the **staff members** of Computer Science Department, BIT.

SATYAM KUMAR
1BI20CS153

Table of contents

Chapter – 1

INTRODUCTION

1.1 Project Summary:

The fitness app is designed to revolutionize the way people approach their health and fitness goals. By leveraging advanced technology and data analysis, the app aims to provide users with personalized fitness experiences tailored to their specific needs. Whether it's tracking daily steps, calories burned, or heart rate monitoring, the app offers comprehensive fitness tracking capabilities to help users stay on top of their progress.

In addition to tracking, the app also offers a wide range of workout plans and exercises to choose from. These plans are carefully curated to cater to different fitness levels and goals, whether it's weight loss, muscle building, or general fitness. Users can follow guided workout routines or customize their own, ensuring a flexible and adaptable approach to fitness.

Overall, the fitness app aims to be a comprehensive fitness companion, providing users with the tools, resources, and support they need to achieve their fitness goals. With features that encompass tracking, workouts, nutrition, and community engagement, the app strives to empower individuals on their journey towards a healthier and more active lifestyle.

1.2 Project Purpose:

The purpose of the fitness app is to empower individuals to take control of their health and fitness by providing personalized tracking, workout plans, nutrition guidance, and a supportive community.

1.3 Project Scope:

The scope of this project is:

- Diverse workout Program for various fitness goals and preferences.
- Advanced Tracking for monitoring progress and analysing fitness metrics.

1.4 Technology and Literature Review:

Technologies used:

1) Android Studio
2) Java Programming language

3) Realtime Database SQL Lite.

1.5 Problem Statement:

Lack of motivation and guidance in maintaining a healthy lifestyle, necessitating a fitness app for personalized fitness tracking and support.

1.6 Objective of the Project:

- Provide personalized fitness tracking: The app will track users' activity levels, exercise routines, and progress towards their fitness goals, providing valuable insights and data-driven recommendations.
- Offer customized workout plans: The app will generate tailored workout plans based on users' fitness levels, goals, and preferences, ensuring effective and targeted exercise routines.
- Offer expert guidance and resources: The app will provide access to professional trainers, fitness experts, and educational resources such as articles, videos, and tutorials to empower users with knowledge and guidance for their fitness journey.

1.7 Organization of the Report:

Our project report consists of several chapters, each discussing different aspects of the project. In the initial chapters, we present the problem statement and provide a concise summary of the project.

Moving forward, we showcase snapshots of our application to provide a visual representation of its appearance and functionality. These snapshots give readers an overview of how our application is designed and how it operates.

By structuring our report in this manner, we ensure that all essential components are covered, including the problem statement, system requirements, architecture and design, visual representations, and concluding remarks on the project's progress and future possibilities.

Chapter – 2

SYSTEM REQUIREMENTS

2.1 User Characteristics:

The user has different characteristics:

- Exercise tracking: Allows users to record and monitor their physical activities and workouts.
- Goal setting: Enables users to set fitness goals and track their progress towards achieving them.
- Workout plans: Offers pre-designed or customizable workout routines to guide users through their fitness journey.
- Progress visualization: Displays visual representations of user progress, such as charts or graphs, to track improvements over time.
- Data analysis: Analyzes user data to provide insights and trends, helping users make informed decisions about their fitness and health.

2.2 Hardware and Software Requirements:

1) Android 5.0 and above.
2) Above 4 GB of available space.
3) Minimum of 8GB RAM. 16 GB is recommended.
4) Available Internet Connection.

2.3 Project Operation Constraints:

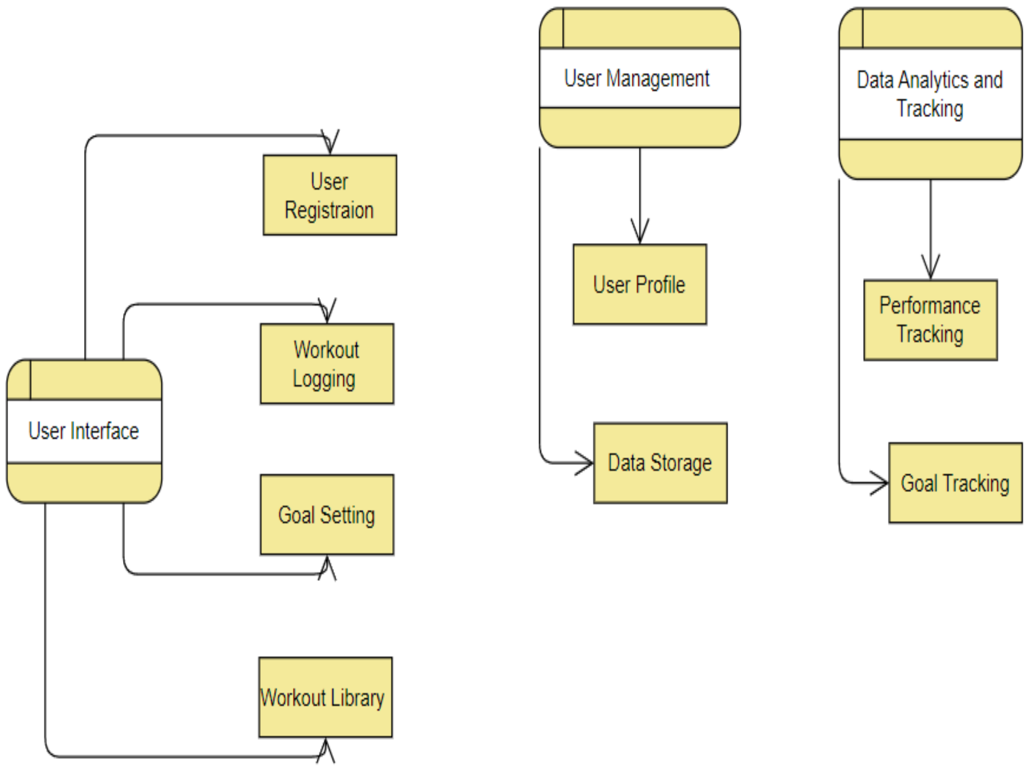
There are only two constraints to deploy this project is:

- Privacy and Data Security: Implement measures to protect user data and ensure compliance with privacy regulations.
- Availability and Reliability: Maintain a robust infrastructure and proactive monitoring to ensure uninterrupted access and reliable performance.

Chapter – 3

DESIGN

3.1 Data Flow Graph:



3.1 System Architecture

3.2 State Transition Diagram:

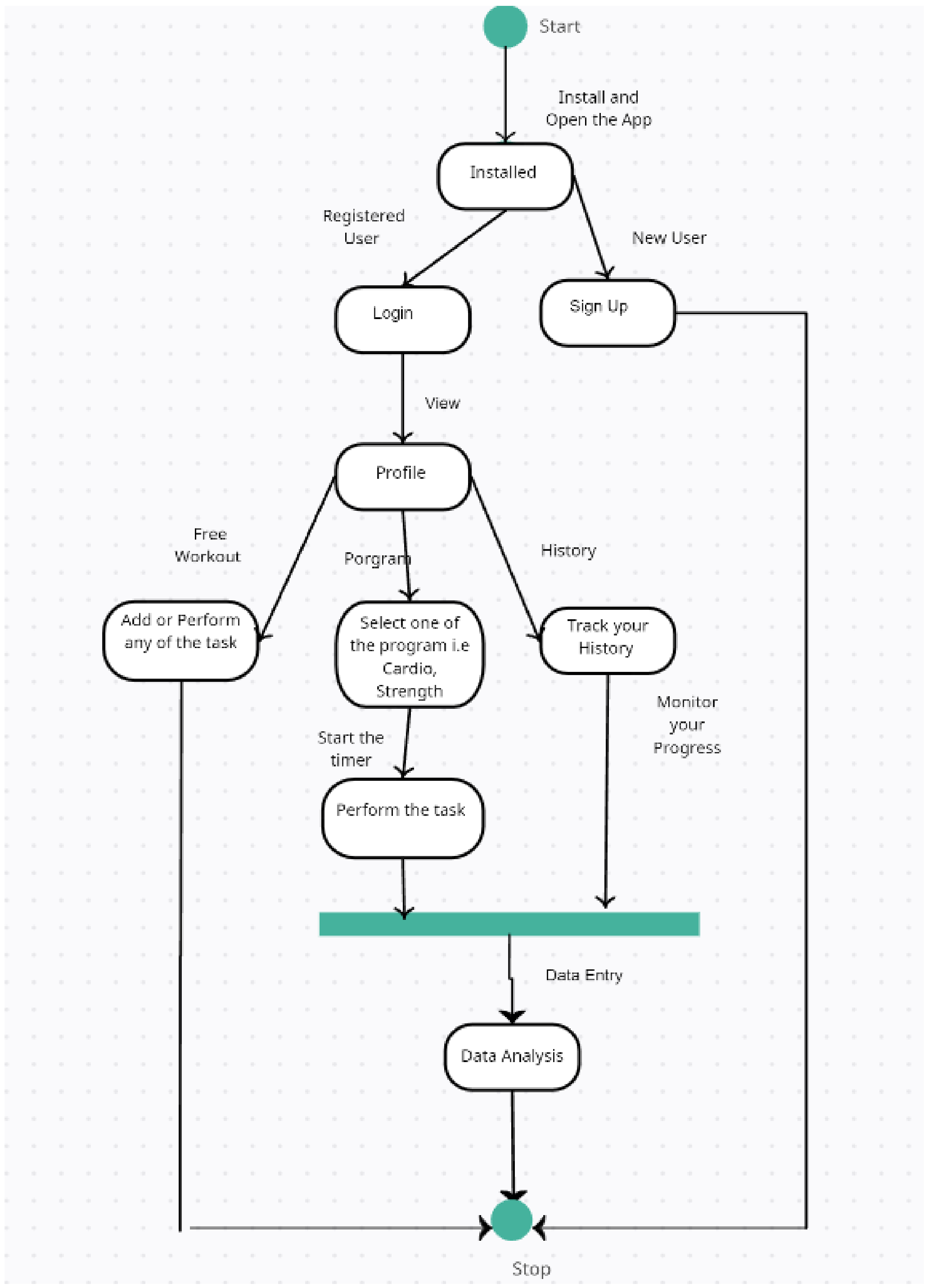


Figure 3. 2- State Transition Diagram

3.3 Module Description:

Exercise Inflater: This class is used to instantiate exercise XML files into Exercise Objects.
Exercise Item: This class represents a single exercise item in a fitness menu.
Intent: This class is used to start an activity and service related to fitness activities.
Fitness Auth: This class is used to implement user authentication for accessing fitness data and features.
Fitness User: This class is used to access all corresponding fields of the current fitness user.
Dialog Builder: This class is used to create an Alert Dialogue Box for fitness-related notifications and prompts.

Chapter – 4

IMPLEMENTATION

4.1 Built-In Functions:

1) setOnClickListner(): system executes the code you write in onClick(View) after the user presses the button. The system executes the code in onClick on the main thread.
2) setVisibility(): You can hide/show views using thread visibility.
3) Intent(): This class is used to start an activity and service. It is also used to redirect the user from one screen to another.
4) getText(): It is used to retrieve text from EditTextView.
5) setText(): It is used to set text dynamically to TextView.

4.2Android Manifest and Special Permissions:

```
<?xml version="1.0" encoding="utf-8"?> <manifest
xmlns:android="http://schemas.android.com/apk/res/a
ndroid" package="com.easyfitness"> <uses-ermission
android:name="android.permission.VIBRATE" />
android:name="android.permission.READ_EXTERN
AL_STORAGE" /> <uses-permission
android:name="android.permission.WRITE_EXTER
NAL_STORAGE" android:maxSdkVersion="28" />
<uses-permission
android:name="android.permission.ACTION_HEAD
SET_PLUG" /> <uses-feature
android:name="android.hardware.camera.any"
android:required="false" /> <uses-feature
android:name="android.hardware.camera.autofocus"
android:required="false" /> <application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:requestLegacyExternalStorage="true"
```

```
android:theme="@style/AppTheme"
android:preserveLegacyExternalStorage="true">
<activity android:name=".MainActivity"
    android:configChanges="orientation|screenSize"
    android:label="@string/app_name"
    android:windowSoftInputMode="stateUnchanged">
    <intent-filter> <action
    android:name="android.intent.action.MAIN" />
    <category
    android:name="android.intent.category.LAUNCHER"
    /> <action
    android:name="android.intent.action.VIEW" />
    </intent-filter> </activity> <activity
    android:name=".intro.MainIntroActivity"
    android:theme="@style/Theme.Intro" /> <activity
    android:name="com.theartofdev.edmodo.cropper.Cro
    pImageActivity"
    android:theme="@style/Base.Theme.AppCompat" />
    <meta-data
    android:name="firebase_crash_collection_enabled"
    android:value="@bool/FIREBASE_CRASH_ENABL
    ED" /> <provider
    android:name="androidx.core.content.FileProvider"
    android:authorities="${applicationId}.fileprovider"
    android:exported="false"
    android:grantUriPermissions="true"> <meta-data
    android:name="android.support.FILE_PROVIDER_P
    ATHS" android:resource="@xml/provider_paths" />
    </provider> <receiver
    android:name=".utils.AlarmReceiver"
    android:process=":remote" /> </application>
    </manifest>
```

4.3XML and JAVA Code:

XML Codes:

```
<androidx.drawerlayout.widget.DrawerLayout
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/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

   <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/background"
        android:focusableInTouchMode="true"
        android:orientation="vertical"
        android:paddingLeft="0dp"
        android:paddingTop="0dp"
        android:paddingRight="0dp"
        android:paddingBottom="0dp">
        <androidx.appcompat.widget.Toolbar
            android:id="@+id/actionToolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/icon_enable"
            android:elevation="4dp"
            android:minHeight="?attr/actionBarSize"
            android:visibility="visible">
<com.mikhaellopez.circularimageview.CircularImageView
                android:id="@+id/imageProfile"
                android:layout_width="40dp"
                android:layout_height="40dp"
                android:layout_alignParentStart="true"
                android:layout_centerVertical="true"
                android:layout_gravity="end"
                android:src="@drawable/ic_person"
                app:civ_border_color="#EEEEEE"
                app:civ_border_width="0dp"
                app:civ_shadow="false"
                app:civ_shadow_color="#8BC34A"
                app:civ_shadow_radius="0" />
        </androidx.appcompat.widget.Toolbar>
```

```
<androidx.appcompat.widget.Toolbar
        android:id="@+id/musicToolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_margin="0dp"
        android:background="@color/toolbar_background"
        android:elevation="4dp"
        android:minHeight="?attr/actionBarSize"
        android:padding="0dp"
        app:contentInsetStart="0dp"
        app:titleMargins="0dp">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="0dp"
            android:orientation="vertical"
            android:padding="0dp">
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="0dp"
                android:orientation="vertical">
                <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:orientation="horizontal">
                    <TextView
                        android:id="@+id/playerSongProgress"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_marginLeft="4dp"
                        android:layout_marginRight="4dp"
                        android:maxLines="1"
                        tools:text="00:00" />
                    <TextView
                        android:id="@+id/playerSongTitle"
                        android:layout_width="match_parent"
                        android:layout_height="wrap_content"
                        android:layout_gravity="top"
                        android:layout_marginLeft="4dp"
                        android:layout_marginRight="4dp"
                        android:ellipsize="marquee"
                        android:marqueeRepeatLimit="marquee_forever"
                        android:scrollHorizontally="true"
                        android:singleLine="true"
                        tools:text="musique_test.mp3" />

                </LinearLayout>
                <SeekBar
                    android:id="@+id/playerSeekBar"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginStart="0dp"
                    android:layout_marginLeft="0dp"
                    android:layout_marginTop="-4dp"
                    android:layout_marginEnd="0dp"
                    android:layout_marginRight="0dp"
                    android:layout_marginBottom="-4dp" />
                <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="32dp"
```

```
android:orientation="horizontal">
<ImageButton
    android:id="@+id/playerPrevious"
    android:layout_width="36dp"
    android:layout_height="36dp"
    android:layout_gravity="center_vertical"
    android:layout_weight="5"
    android:adjustViewBounds="false"
    android:background="@android:color/transparent"
    android:baselineAlignBottom="false"
    android:cropToPadding="false"
    android:scaleType="fitCenter"
    android:src="@drawable/ic_skip_previous" />
<ImageButton
    android:id="@+id/playerStop"
    android:layout_width="36dp"
    android:layout_height="36dp"
    android:layout_gravity="center_vertical"
    android:layout_weight="5"
    android:adjustViewBounds="false"
    android:background="@android:color/transparent"
    android:baselineAlignBottom="false"
    android:cropToPadding="true"
    android:scaleType="fitCenter"
    android:src="@drawable/ic_stop" />
<ImageButton
    android:id="@+id/playerPlay"
    android:layout_width="36dp"
    android:layout_height="36dp"
    android:layout_gravity="center_vertical"
    android:layout_weight="5"
    android:adjustViewBounds="true"
```

```
        android:background="@android:color/transparent"
        android:baselineAlignBottom="false"
        android:cropToPadding="false"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_play_arrow" />
    <ImageButton
        android:id="@+id/playerNext"
        android:layout_width="36dp"
        android:layout_height="36dp"
        android:layout_gravity="center_vertical"
        android:layout_weight="5"
        android:adjustViewBounds="false"
        android:background="@android:color/transparent"
        android:baselineAlignBottom="false"
        android:cropToPadding="true"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_skip_next" />
    <ImageButton
        android:id="@+id/playerLoop"
        android:layout_width="36dp"
        android:layout_height="36dp"
        android:layout_gravity="center_vertical"
        android:layout_weight="5"
        android:adjustViewBounds="false"
        android:background="@android:color/transparent"
        android:baselineAlignBottom="false"
        android:cropToPadding="false"
        android:padding="4dp"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_replay_white" />

    <ImageButton
        android:id="@+id/playerRandom"
        android:layout_width="36dp"
```

```
                android:layout_height="36dp"
                android:layout_gravity="center_vertical"
                android:layout_weight="5"
                android:adjustViewBounds="false"
                android:background="@android:color/transparent"
                android:baselineAlignBottom="false"
                android:cropToPadding="false"
                android:padding="4dp"
                android:scaleType="fitCenter"
                android:src="@drawable/ic_random_white" />

            <ImageButton
                android:id="@+id/playerList"
                android:layout_width="36dp"
                android:layout_height="36dp"
                android:layout_gravity="center_vertical"
                android:layout_weight="5"
                android:adjustViewBounds="false"
                android:background="@android:color/transparent"
                android:baselineAlignBottom="false"
                android:cropToPadding="false"
                android:padding="4dp"
                android:scaleType="fitCenter"
                android:src="@drawable/ic_library_music" />
          </LinearLayout>
        </LinearLayout>
      </LinearLayout>
    </androidx.appcompat.widget.Toolbar>

    <LinearLayout
      android:id="@+id/fragment_container"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:layout_above="@id/musicToolbar"
      android:layout_below="@id/actionToolbar"
```

```
            android:orientation="vertical" />
    </RelativeLayout>
  <ListView
        android:id="@+id/left_drawer"
        android:layout_width="240dp"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        android:background="#111"
        android:choiceMode="none"
        android:divider="@android:color/transparent"
        android:dividerHeight="0dp" />
</androidx.drawerlayout.widget.DrawerLayout>
```

JAVA Codes:

```
package com.easyfitness;

import android.Manifest;
import android.app.AlertDialog;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.PopupMenu;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.app.AppCompatDelegate;
import androidx.appcompat.content.res.AppCompatResources;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;
import androidx.lifecycle.ViewModelProvider;
import androidx.preference.PreferenceManager;

import com.easyfitness.DAO.DAOMachine;
import com.easyfitness.DAO.DAOProfile;
import com.easyfitness.DAO.DatabaseHelper;
import com.easyfitness.DAO.Machine;
import com.easyfitness.DAO.Profile;
import com.easyfitness.DAO.cardio.DAOOldCardio;
import com.easyfitness.DAO.cardio.OldCardio;
import com.easyfitness.DAO.export.CVSManager;
import com.easyfitness.DAO.program.DAOProgram;
import com.easyfitness.DAO.program.Program;
import com.easyfitness.DAO.record.DAOCardio;
import com.easyfitness.DAO.record.DAOFonte;
import com.easyfitness.DAO.record.DAORecord;
import com.easyfitness.DAO.record.DAOStatic;
import com.easyfitness.DAO.record.Record;
import com.easyfitness.bodymeasures.BodyPartListFragment;
import com.easyfitness.enums.DistanceUnit;
import com.easyfitness.enums.ExerciseType;
import com.easyfitness.enums.WeightUnit;
import com.easyfitness.fonte.FontesPagerFragment;
import com.easyfitness.intro.MainIntroActivity;
import com.easyfitness.machines.MachineFragment;
import com.easyfitness.programs.ProgramListFragment;
import com.easyfitness.utils.DateConverter;
import com.easyfitness.utils.ImageUtil;
import com.easyfitness.utils.MusicController;
import com.easyfitness.utils.UnitConverter;
import com.mikhaellopez.circularimageview.CircularImageView;
import com.onurkaganaldemir.ktoastlib.KToast;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.Locale;

import cn.pedant.SweetAlert.SweetAlertDialog;

public class MainActivity extends AppCompatActivity {

    private static final int TIME_INTERVAL = 2000;
    public static String FONTESPAGER = "FontePager";
    public static String WEIGHT = "Weight";
    public static String PROFILE = "Profile";
    public static String BODYTRACKING = "BodyTracking";
    public static String BODYTRACKINGDETAILS = "BodyTrackingDetail";
    public static String ABOUT = "About";
    public static String SETTINGS = "Settings";
    public static String MACHINES = "Machines";
    public static String MACHINESDETAILS = "MachinesDetails";
    public static String WORKOUTS = "Workouts";
    public static String WORKOUTPAGER = "WorkoutPager";
    public static String PREFS_NAME = "prefsfile";
    private final int REQUEST_CODE_INTRO = 111;
    private final int
PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE_FOR_EXPORT = 1001;
    private final int
PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE_FOR_IMPORT = 1002;
    private final int MY_PERMISSIONS_REQUEST_READ_EXTERNAL_STORAGE = 103;

    private static final int IMPORT_DATABASE = 2;
    public static final int OPEN_MUSIC_FILE = 3;

    private final MusicController musicController = new MusicController(this);
    CustomDrawerAdapter mDrawerAdapter;
    List<DrawerItem> dataList;
    /* Fragments */
    private FontesPagerFragment mpFontesPagerFrag = null;
    private WeightFragment mpWeightFrag = null;
    private ProfileFragment mpProfileFrag = null;
    private MachineFragment mpMachineFrag = null;
    private SettingsFragment mpSettingFrag = null;
    private AboutFragment mpAboutFrag = null;
    private BodyPartListFragment mpBodyPartListFrag = null;
    private ProgramListFragment mpWorkoutListFrag;
    private String currentFragmentName = "";
    private DAOProfile mDbProfils = null;
    private Profile mCurrentProfile = null;
    private long mCurrentProfilID = -1;
    private Toolbar top_toolbar = null;
    /* Navigation Drawer */
    private DrawerLayout mDrawerLayout = null;
    private ListView mDrawerList = null;
    private ActionBarDrawerToggle mDrawerToggle = null;
    private CircularImageView roundProfile = null;
    private String mCurrentMachine = "";
    private boolean mIntro014Launched = false;
    private boolean mMigrationBD15done = false;
    private boolean mMigrationToScopedStoragedone = false;
    private long mBackPressed;
    private AppViMo appViMo;

    private final PopupMenu.OnMenuItemClickListener onMenuItemClick = item -> {
        switch (item.getItemId()) {
            case R.id.create_newprofil:
                getActivity().CreateNewProfile();
                return true;
            case R.id.change_profil:
                String[] profilListArray = getActivity().mDbProfils.getAllProfile();

                AlertDialog.Builder changeProfilbuilder = new AlertDialog.Builder(getActivity());

changeProfilbuilder.setTitle(getActivity().getResources().getText(R.string.profil_select_profil))
                        .setItems(profilListArray, (dialog, which) -> {
                            ListView lv = ((AlertDialog) dialog).getListView();
                            Object checkedItem = lv.getAdapter().getItem(which);
                            setCurrentProfile(checkedItem.toString());
                            KToast.infoToast(getActivity(),
getActivity().getResources().getText(R.string.profileSelected) + " : " + checkedItem.toString(),
Gravity.BOTTOM, KToast.LENGTH_LONG);
                            //Toast.makeText(getApplicationContext(),
getActivity().getResources().getText(R.string.profileSelected) + " : " + checkedItem.toString(),
Toast.LENGTH_LONG).show();
                        });
                changeProfilbuilder.show();
                return true;
            case R.id.delete_profil:
                String[] profildeleteListArray = getActivity().mDbProfils.getAllProfile();

                AlertDialog.Builder deleteProfilbuilder = new AlertDialog.Builder(getActivity());

deleteProfilbuilder.setTitle(getActivity().getResources().getText(R.string.profil_select_profil_to
_delete))
                        .setItems(profildeleteListArray, (dialog, which) -> {
                            ListView lv = ((AlertDialog) dialog).getListView();
                            Object checkedItem = lv.getAdapter().getItem(which);
                            if (getCurrentProfile().getName().equals(checkedItem.toString())) {
                                KToast.errorToast(getActivity(),
getActivity().getResources().getText(R.string.impossibleToDeleteProfile).toString(),
Gravity.BOTTOM, KToast.LENGTH_LONG);
                            } else {
                                Profile profileToDelete = mDbProfils.getProfile(checkedItem.toString());
                                mDbProfils.deleteProfile(profileToDelete);
                                KToast.infoToast(getActivity(), getString(R.string.profileDeleted) + ":" +
checkedItem.toString(), Gravity.BOTTOM, KToast.LENGTH_LONG);
                            }
                        });
                deleteProfilbuilder.show();
                return true;
            case R.id.rename_profil:
                getActivity().renameProfil();
                return true;
            case R.id.param_profil:
                showFragment(PROFILE);
                return true;
            default:
                return false;
        }
    };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        SharedPreferences SP =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        String dayNightAuto = SP.getString("dayNightAuto", "2");
        int dayNightAutoValue;
        try {
          dayNightAutoValue = Integer.parseInt(dayNightAuto);
        } catch (NumberFormatException e) {
          dayNightAutoValue = 2;
        }
        if (dayNightAutoValue == getResources().getInteger(R.integer.dark_mode_value)) {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
          SweetAlertDialog.DARK_STYLE = true;
        } else if (dayNightAutoValue == getResources().getInteger(R.integer.light_mode_value)) {
          AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
          SweetAlertDialog.DARK_STYLE = false;
        } else {

AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_FOLLOW_S
YSTEM);
          int currentNightMode = getResources().getConfiguration().uiMode
              & Configuration.UI_MODE_NIGHT_MASK;
          switch (currentNightMode) {
            case Configuration.UI_MODE_NIGHT_YES:
              SweetAlertDialog.DARK_STYLE = true;
              break;
            case Configuration.UI_MODE_NIGHT_NO:
            default:
              SweetAlertDialog.DARK_STYLE = false;
          }
        }

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        loadPreferences();

        top_toolbar = this.findViewById(R.id.actionToolbar);
        setSupportActionBar(top_toolbar);
        top_toolbar.setTitle(getResources().getText(R.string.app_name));

        if (savedInstanceState == null) {
          if (mpFontesPagerFrag == null)
            mpFontesPagerFrag = FontesPagerFragment.newInstance(FONTESPAGER, 6);
          if (mpWeightFrag == null) mpWeightFrag = WeightFragment.newInstance(WEIGHT,
5);
```

```
            if (mpProfileFrag == null) mpProfileFrag = ProfileFragment.newInstance(PROFILE,
10);
            if (mpSettingFrag == null) mpSettingFrag = SettingsFragment.newInstance(SETTINGS,
8);
            if (mpAboutFrag == null) mpAboutFrag = AboutFragment.newInstance(ABOUT, 4);
            if (mpMachineFrag == null) mpMachineFrag =
MachineFragment.newInstance(MACHINES, 7);
            if (mpBodyPartListFrag == null)
                mpBodyPartListFrag = BodyPartListFragment.newInstance(BODYTRACKING, 9);
            if (mpWorkoutListFrag == null)
                mpWorkoutListFrag = ProgramListFragment.newInstance(WORKOUTS, 11);
        } else {
            mpFontesPagerFrag = (FontesPagerFragment)
getSupportFragmentManager().getFragment(savedInstanceState, FONTESPAGER);
            mpWeightFrag = (WeightFragment)
getSupportFragmentManager().getFragment(savedInstanceState, WEIGHT);
            mpProfileFrag = (ProfileFragment)
getSupportFragmentManager().getFragment(savedInstanceState, PROFILE);
            mpSettingFrag = (SettingsFragment)
getSupportFragmentManager().getFragment(savedInstanceState, SETTINGS);
            mpAboutFrag = (AboutFragment)
getSupportFragmentManager().getFragment(savedInstanceState, ABOUT);
            mpMachineFrag = (MachineFragment)
getSupportFragmentManager().getFragment(savedInstanceState, MACHINES);
            mpBodyPartListFrag = (BodyPartListFragment)
getSupportFragmentManager().getFragment(savedInstanceState, BODYTRACKING);
            mpWorkoutListFrag = (ProgramListFragment)
getSupportFragmentManager().getFragment(savedInstanceState, WORKOUTS);
        }

        appViMo = new ViewModelProvider(this).get(AppViMo.class);
        appViMo.getProfile().observe(this, profile -> {
            // Update UI
            setDrawerTitle(profile.getName());
            setPhotoProfile(profile.getPhoto());
            mCurrentProfilID = profile.getId();
            savePreferences();
        });

        DatabaseHelper.renameOldDatabase(this);

        if (DatabaseHelper.DATABASE_VERSION >= 15 && !mMigrationBD15done) {
            DAOOldCardio mDbOldCardio = new DAOOldCardio(this);
            DAOMachine lDAOMachine = new DAOMachine(this);
```

```
      if (mDbOldCardio.tableExists()) {
        DAOCardio mDbCardio = new DAOCardio(this);
        List<OldCardio> mList = mDbOldCardio.getAllRecords();
        for (OldCardio record : mList) {
          Machine m = lDAOMachine.getMachine(record.getExercice());
          String exerciseName = record.getExercice();
          if (m != null) { // if a machine exists
            if (m.getType() == ExerciseType.STRENGTH) { // if it is not a Cardio type
              exerciseName = exerciseName + "-Cardio"; // add a suffix to
            }
          }
          mDbCardio.addCardioRecordToFreeWorkout(record.getDate(), exerciseName,
record.getDistance(), record.getDuration(), record.getProfil().getId(), DistanceUnit.KM);
        }
        mDbOldCardio.dropTable();

        DAORecord daoRecord = new DAORecord(this);
        List<Record> mFonteList = daoRecord.getAllRecords();
        for (Record record : mFonteList) {
          record.setExerciseType(ExerciseType.STRENGTH);
          daoRecord.updateRecord(record); // Automatically update record Type
        }
        List<Machine> machineList = lDAOMachine.getAllMachinesArray();
        for (Machine record : machineList) {
          lDAOMachine.updateMachine(record); // Reset all the fields on machines.
        }
      }
      mMigrationBD15done = true;
      savePreferences();
    }
    if (savedInstanceState == null) {
      showFragment(FONTESPAGER); // Create fragment, do not add to backstack
      currentFragmentName = FONTESPAGER;
    }
    dataList = new ArrayList<>();
    mDrawerLayout = findViewById(R.id.drawer_layout);
    mDrawerList = findViewById(R.id.left_drawer);
    DrawerItem drawerTitleItem = new DrawerItem("TITLE", R.drawable.ic_person, true);

    dataList.add(drawerTitleItem);
    dataList.add(new DrawerItem(this.getResources().getString(R.string.menu_Workout),
R.drawable.ic_fitness_center, true));
```

```
        dataList.add(new DrawerItem(this.getResources().getString(R.string.MachinesLabel),
R.drawable.ic_exercises, true));
        dataList.add(new DrawerItem("Programs List", R.drawable.ic_exam, true));
        dataList.add(new DrawerItem(this.getResources().getString(R.string.weightMenuLabel),
R.drawable.ic_bathroom_scale, true));
        dataList.add(new DrawerItem(this.getResources().getString(R.string.bodytracking),
R.drawable.ic_ruler, true));
        dataList.add(new DrawerItem(this.getResources().getString(R.string.SettingLabel),
R.drawable.ic_settings, true));
        dataList.add(new DrawerItem(this.getResources().getString(R.string.AboutLabel),
R.drawable.ic_info_outline, true));
        mDrawerAdapter = new CustomDrawerAdapter(this, R.layout.custom_drawer_item,
                dataList);
        mDrawerList.setAdapter(mDrawerAdapter);

        roundProfile = top_toolbar.findViewById(R.id.imageProfile);

        mDrawerToggle = new ActionBarDrawerToggle(
                this,                  /* host Activity */
                mDrawerLayout,         /* DrawerLayout object */
                top_toolbar,  /* nav drawer icon to replace 'Up' caret */
                R.string.drawer_open, R.string.drawer_close
        );

        // Set the list's click listener
        mDrawerList.setOnItemClickListener(new DrawerItemClickListener());

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setHomeButtonEnabled(true);

        musicController.initView();

        // Test if intro has been launched already
        if (!mIntro014Launched) {
            Intent intent = new Intent(this, MainIntroActivity.class);
            startActivityForResult(intent, REQUEST_CODE_INTRO);
        }
    }

    @Override
    protected void onStart() {
        super.onStart();  // Always call the superclass method first
```

```
 if (mIntro014Launched) {
            initActivity();
            initDEBUGdata();
        }

        if ( !mMigrationToScopedStoragedone) { //do the migration only once.
            migrateDatabase();
        }

        SharedPreferences SP =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        boolean bShowMP3 = SP.getBoolean("prefShowMP3", false);
        this.showMP3Toolbar(bShowMP3);
        this.checkLastBackup(SP);
    }
    private void checkLastBackup(SharedPreferences SP) {
        int milliSecondsOfDay = 86400000; // 24 * 60 * 60 * 1000
        int milliSecondsOfWeek = 604800000; // 7 * 24 * 60 * 60 * 1000
        long milliSecondsOfMonth = 2419200000L; // 4 * 7 * 24 * 60 * 60 * 1000

        long lastBackupUTCTime = SP.getLong("prefLastTimeBackupUTCTime", -1);
        int prefBackupSetting = Integer.parseInt(SP.getString("defaultBackupSetting", "0"));
        if (lastBackupUTCTime == -1 && prefBackupSetting > 0) {
            //KToast.warningToast(getActivity(),
getActivity().getResources().getText(R.string.backup_warning_never).toString(),
Gravity.BOTTOM, KToast.LENGTH_LONG);

exportDatabase(getActivity().getResources().getText(R.string.backup_warning_never).toString()
);
        } else {
            if (prefBackupSetting == 1 && System.currentTimeMillis() - lastBackupUTCTime >
milliSecondsOfDay) {
                //KToast.warningToast(getActivity(),
getActivity().getResources().getText(R.string.backup_warning_day).toString(),
Gravity.BOTTOM, KToast.LENGTH_LONG);

exportDatabase(getActivity().getResources().getText(R.string.backup_warning_day).toString());
            } else if (prefBackupSetting == 2 && System.currentTimeMillis() -
lastBackupUTCTime > milliSecondsOfWeek) {
                //KToast.warningToast(getActivity(),
getActivity().getResources().getText(R.string.backup_warning_week).toString(),
Gravity.BOTTOM, KToast.LENGTH_LONG);
exportDatabase(getActivity().getResources().getText(R.string.backup_warning_week).toString()
);
            } else if (prefBackupSetting == 3 && System.currentTimeMillis() -
lastBackupUTCTime > milliSecondsOfMonth) {
                //KToast.warningToast(getActivity(),
getActivity().getResources().getText(R.string.backup_warning_month).toString(),
Gravity.BOTTOM, KToast.LENGTH_LONG);

exportDatabase(getActivity().getResources().getText(R.string.backup_warning_month).toString(
));             }         }
    }
    private void initDEBUGdata() {
        if (BuildConfig.DEBUG_MODE) {
            // do something for a debug build
            DAOFonte lDbFonte = new DAOFonte(this);
            if (lDbFonte.getCount() == 0) {
                lDbFonte.addStrengthRecordToFreeWorkout(DateConverter.dateToDate(2019, 7, 1,
12, 34, 56), "Example 1", 1, 10, 40, WeightUnit.KG, "", this.getCurrentProfile().getId());
                lDbFonte.addStrengthRecordToFreeWorkout(DateConverter.dateToDate(2019, 6, 30,
12, 34, 56), "Example 2", 1, 10, UnitConverter.LbstoKg(60), WeightUnit.LBS, "",
this.getCurrentProfile().getId());
            }
            DAOCardio lDbCardio = new DAOCardio(this);
            if (lDbCardio.getCount() == 0) {
                lDbCardio.addCardioRecordToFreeWorkout(DateConverter.dateToDate(2019, 7, 1),
"Running Example", 1, 10000, this.getCurrentProfile().getId(), DistanceUnit.KM);
                lDbCardio.addCardioRecordToFreeWorkout(DateConverter.dateToDate(2019, 7, 31),
"Cardio Example", UnitConverter.MilesToKm(2), 20000, this.getCurrentProfile().getId(),
DistanceUnit.MILES);
            }
            DAOStatic lDbStatic = new DAOStatic(this);
            if (lDbStatic.getCount() == 0) {
                lDbStatic.addStaticRecordToFreeWorkout(DateConverter.dateToDate(2019, 7, 1, 12,
34, 56), "Exercise ISO 1", 1, 50, 40, this.getCurrentProfile().getId(), WeightUnit.KG, "");
                lDbStatic.addStaticRecordToFreeWorkout(DateConverter.dateToDate(2019, 7, 31, 12,
34, 56), "Exercise ISO 2", 1, 60, UnitConverter.LbstoKg(40), this.getCurrentProfile().getId(),
WeightUnit.LBS, "");
            }
            DAOProgram lDbWorkout = new DAOProgram(this);
            if (lDbWorkout.getCount() == 0) {
                lDbWorkout.populate();
            }
        }
    }
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        // Always call the superclass so it can restore the view hierarchy
        super.onRestoreInstanceState(savedInstanceState);
    }

    @Override
    protected void onSaveInstanceState(@NonNull Bundle outState) {
        super.onSaveInstanceState(outState);

        //Save the fragment's instance
        if (getFontesPagerFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, FONTESPAGER,
mpFontesPagerFrag);
        if (getWeightFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, WEIGHT, mpWeightFrag);
        if (getProfileFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, PROFILE, mpProfileFrag);
        if (getMachineFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, MACHINES, mpMachineFrag);
        if (getAboutFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, ABOUT, mpAboutFrag);
        if (getSettingsFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, SETTINGS, mpSettingFrag);
        if (getBodyPartFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, BODYTRACKING,
mpBodyPartListFrag);
        if (getWorkoutListFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, WORKOUTS,
mpWorkoutListFrag);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu items for use in the action bar
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_activity_actions, menu);

        // restore the profile picture in case it was overwritten during the menu inflate
        if (mCurrentProfile != null) setPhotoProfile(mCurrentProfile.getPhoto());

        return super.onCreateOptionsMenu(menu);
    }
```

```
  @Override
  public boolean onPrepareOptionsMenu(Menu menu) {
    roundProfile.setOnClickListener(v -> {
      PopupMenu popup = new PopupMenu(getActivity(), v);
      MenuInflater inflater = popup.getMenuInflater();
      inflater.inflate(R.menu.profile_actions, popup.getMenu());
      popup.setOnMenuItemClickListener(onMenuItemClick);
      popup.show();
    });

    return super.onPrepareOptionsMenu(menu);
  }

  @SuppressWarnings("deprecation")
  private boolean migrateToScopedStorage() {
    boolean success = true;
    File folder = new File(Environment.getExternalStorageDirectory() + "/FastnFitness");

    if (folder.exists()) { //migrate all pictures
      //Migrate profile pictures
      DAOProfile daoProfile = new DAOProfile(getBaseContext());
      List<Profile> profileList = daoProfile.getAllProfiles(daoProfile.getWritableDatabase());
      for (Profile profile:profileList) {
        if(profile.getPhoto()!=null && !profile.getPhoto().isEmpty()) {
          File sourceFile = new File(profile.getPhoto());
          File storageDir =
getActivity().getExternalFilesDir(Environment.DIRECTORY_PICTURES);
          if (sourceFile.exists()) {
            try {
              File destFile = ImageUtil.copyFile(sourceFile, storageDir);
              profile.setPhoto(destFile.getAbsolutePath());
              ImageUtil.saveThumb(destFile.getAbsolutePath());
              daoProfile.updateProfile(profile);
            } catch (IOException e) {
              e.printStackTrace();
              profile.setPhoto("");
              daoProfile.updateProfile(profile);
              success = false;
            }
          } else {
            profile.setPhoto("");
            daoProfile.updateProfile(profile);
          }
        }
      }
```

```
            //Migrate exercises pictures
            DAOMachine daoMachine = new DAOMachine(getBaseContext());
            List<Machine> machineList = daoMachine.getAllMachinesArray();
            for (Machine machine:machineList) {
                if(machine.getPicture()!=null && !machine.getPicture().isEmpty()) {
                    File sourceFile = new File(machine.getPicture());
                    File storageDir =
getActivity().getExternalFilesDir(Environment.DIRECTORY_PICTURES);
                    if (sourceFile.exists()) {
                        machine.setPicture("");
                        try {
                            File destFile = ImageUtil.copyFile(sourceFile, storageDir);
                            ImageUtil.saveThumb(destFile.getAbsolutePath());
                            machine.setPicture(destFile.getAbsolutePath());
                            daoMachine.updateMachine(machine);
                        } catch (Exception e) {
                            e.printStackTrace();
                            success = false;
                        }
                    }
                    daoMachine.updateMachine(machine);
                }
            }
        }
        return success;
    }

    @SuppressWarnings("deprecation")
    private void migrateDatabase() {
        File folder = new File(Environment.getExternalStorageDirectory() + "/FastnFitness");
        if (!folder.exists()) {
            mMigrationToScopedStoragedone = true;
            savePreferences();
            return;
        }

        AlertDialog.Builder exportDbBuilder = new AlertDialog.Builder(this);

exportDbBuilder.setTitle(getActivity().getResources().getText(R.string.database_migration));
        exportDbBuilder.setMessage(R.string.disclaimer_scoped_storage);

exportDbBuilder.setPositiveButton(getActivity().getResources().getText(R.string.global_yes),
(dialog, which) -> {
            if (!migrateToScopedStorage()) {
                AlertDialog.Builder errorDialogBuilder = new AlertDialog.Builder(this);
                errorDialogBuilder.setTitle(R.string.database_migration);
                errorDialogBuilder.setMessage(R.string.something_went_wrong);
                AlertDialog errorDialog = errorDialogBuilder.create();
                errorDialog.show();
            } else {
                KToast.infoToast(this, getString(R.string.database_migration_success),
Gravity.BOTTOM, KToast.LENGTH_SHORT);
            }
            mMigrationToScopedStoragedone = true;
            savePreferences();
        });

        AlertDialog exportDbDialog = exportDbBuilder.create();
        exportDbDialog.show();
    }

    private void openExportDatabaseDialog(String autoExportMessage) {
        AlertDialog.Builder exportDbBuilder = new AlertDialog.Builder(this);

        exportDbBuilder.setTitle(getActivity().getResources().getText(R.string.export_database));
        exportDbBuilder.setMessage(autoExportMessage + " " +
getActivity().getResources().getText(R.string.export_question) + " " +
getCurrentProfile().getName() + "?");

exportDbBuilder.setPositiveButton(getActivity().getResources().getText(R.string.global_yes),
(dialog, which) -> {
            CVSManager cvsMan = new CVSManager(getActivity().getBaseContext());
            SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy_MM_dd_H_m_s",
Locale.getDefault());
            Date date = new Date();
            String folderName = Environment.DIRECTORY_DOWNLOADS +
"/FastnFitness/export/" +  dateFormat.format(date);
            if (cvsMan.exportDatabase(getCurrentProfile(),folderName)) {
                SharedPreferences SP =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
                long currentTime = System.currentTimeMillis();
                SP.edit().putLong("prefLastTimeBackupUTCTime", currentTime).apply();
                if (mpSettingFrag.getContext() != null) {
                    mpSettingFrag.updateLastBackupSummary(SP, currentTime);
                }
```

```
 KToast.successToast(getActivity(), getCurrentProfile().getName() + ": " +
getActivity().getResources().getText(R.string.export_success) + " - " + folderName,
Gravity.BOTTOM, KToast.LENGTH_LONG);
            } else {
                KToast.errorToast(getActivity(), getCurrentProfile().getName() + ": " +
getActivity().getResources().getText(R.string.export_failed), Gravity.BOTTOM,
KToast.LENGTH_LONG);
            }
            dialog.dismiss();
        });

exportDbBuilder.setNegativeButton(getActivity().getResources().getText(R.string.global_no),
(dialog, which) -> dialog.dismiss());

        AlertDialog exportDbDialog = exportDbBuilder.create();
        exportDbDialog.show();
    }
    private void exportDatabase(String autoExportMessage) {
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.Q) {
            if (ContextCompat.checkSelfPermission(this,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE)
                    != PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},

PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE_FOR_EXPORT);
            } else {
                openExportDatabaseDialog(autoExportMessage);
            }
        } else {
            openExportDatabaseDialog(autoExportMessage);
        }
    }

    private void importDatabase() {
        Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
        intent.addCategory(Intent.CATEGORY_OPENABLE);
        intent.setType("text/*");
        startActivityForResult(intent, IMPORT_DATABASE);
    }
```

```
  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    // The action bar home/up action should open or close the drawer.
    // ActionBarDrawerToggle will take care of this.
    if (mDrawerToggle.onOptionsItemSelected(item)) {
      return true;
    }
    switch (item.getItemId()) {
      case R.id.export_database:
        exportDatabase("");
        return true;
      case R.id.import_database:
        importDatabase();
        return true;
      case R.id.action_deleteDB:
        // Afficher une boite de dialogue pour confirmer
        AlertDialog.Builder deleteDbBuilder = new AlertDialog.Builder(this);

deleteDbBuilder.setTitle(getActivity().getResources().getText(R.string.global_confirm));

deleteDbBuilder.setMessage(getActivity().getResources().getText(R.string.deleteDB_warning));

deleteDbBuilder.setPositiveButton(getActivity().getResources().getText(R.string.global_yes),
(dialog, which) -> {
          List<Profile> lList =
mDbProfils.getAllProfiles(mDbProfils.getReadableDatabase());
          for (int i = 0; i < lList.size(); i++) {
            Profile mTempProfile = lList.get(i);
            mDbProfils.deleteProfile(mTempProfile.getId());
          }

          DAOMachine mDbMachines = new DAOMachine(getActivity());
          List<Machine> lList2 = mDbMachines.getAllMachinesArray();
          for (int i = 0; i < lList2.size(); i++) {
            Machine mTemp = lList2.get(i);
            mDbMachines.delete(mTemp.getId());
          }

          DAOProgram mDbPrograms = new DAOProgram(getActivity());
          List<Program> programList = mDbPrograms.getAll();
          for (int i = 0; i < programList.size(); i++) {
            Program mTemp = programList.get(i);
            mDbPrograms.delete(mTemp.getId());
```

```
                        DAORecord mDbRecords = new DAORecord(getActivity());
                        List<Record> recordList =
mDbRecords.getAllTemplateRecordByProgramArray(mTemp.getId());
                        for (int j = 0; j < recordList.size(); j++) {
                            Record mTempRecord = recordList.get(j);
                            mDbRecords.deleteRecord(mTempRecord.getId());
                        }
                    }
                    mIntro014Launched = false; // redisplay the intro
                    dialog.dismiss(); // Close the dialog
                    finish(); // Close app
                });

deleteDbBuilder.setNegativeButton(getActivity().getResources().getText(R.string.global_no),
(dialog, which) -> {
                    // Do nothing
                    dialog.dismiss();
                });
                AlertDialog deleteDbDialog = deleteDbBuilder.create();
                deleteDbDialog.show();
                return true;
            case R.id.action_apropos:
                // Display the fragment as the main content.
                showFragment(ABOUT);
                //getAboutFragment().setHasOptionsMenu(true);
                return true;
            //case android.R.id.home:
            //onBackPressed();
            //          return true;
            case R.id.action_chrono:
                ChronoDialogbox cdd = new ChronoDialogbox(MainActivity.this);
                cdd.show();
                return true;
            default:
                return super.onOptionsItemSelected(item);
        }
    }
    @Override
    public void onRequestPermissionsResult(int requestCode,
                                           @NonNull String[] permissions, @NonNull int[] grantResults) {
        // If request is cancelled, the result arrays are empty.
        if (requestCode ==
PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE_FOR_EXPORT) {
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                KToast.infoToast(this, getString(R.string.access_granted), Gravity.BOTTOM,
KToast.LENGTH_SHORT);
                openExportDatabaseDialog("");
            } else {
                KToast.infoToast(this, getString(R.string.another_time_maybe), Gravity.BOTTOM,
KToast.LENGTH_SHORT);
            }
        } else if (requestCode ==
PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE_FOR_IMPORT) {
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                KToast.infoToast(this, getString(R.string.access_granted), Gravity.BOTTOM,
KToast.LENGTH_SHORT);
                importDatabase();
            } else {
                KToast.infoToast(this, getString(R.string.another_time_maybe), Gravity.BOTTOM,
KToast.LENGTH_SHORT);
            }
        } else if (requestCode ==
MY_PERMISSIONS_REQUEST_READ_EXTERNAL_STORAGE) {
            if (grantResults.length > 0
                    && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                KToast.infoToast(this, getString(R.string.access_granted), Gravity.BOTTOM,
KToast.LENGTH_SHORT);
                musicController.chooseDirectory();
            } else {
                KToast.infoToast(this, getString(R.string.another_time_maybe), Gravity.BOTTOM,
KToast.LENGTH_SHORT);
            }      }    }

        public boolean CreateNewProfile() {
        AlertDialog.Builder newProfilBuilder = new AlertDialog.Builder(this);

        newProfilBuilder.setTitle(getActivity().getResources().getText(R.string.createProfilTitle));

newProfilBuilder.setMessage(getActivity().getResources().getText(R.string.createProfilQuestion
));
        final EditText input = new EditText(this);
        newProfilBuilder.setView(input);
newProfilBuilder.setPositiveButton(getActivity().getResources().getText(android.R.string.ok),
(dialog, whichButton) -> {
            String value = input.getText().toString();

            if (value.isEmpty()) {
                CreateNewProfile();
            } else {
                // Create the new profil
                mDbProfils.addProfile(value);
                // Make it the current.
                setCurrentProfile(value);
            }
        });

newProfilBuilder.setNegativeButton(getActivity().getResources().getText(android.R.string.canc
el), (dialog, whichButton) -> {
            if (getCurrentProfile() == null) {
                CreateNewProfile();
            }
        });
        newProfilBuilder.show();
        return true;
    }
    public boolean renameProfil() {
        AlertDialog.Builder newBuilder = new AlertDialog.Builder(this);
        newBuilder.setTitle(getActivity().getResources().getText(R.string.renameProfilTitle));
newBuilder.setMessage(getActivity().getResources().getText(R.string.renameProfilQuestion));

        // Set an EditText view to get user input
        final EditText input = new EditText(this);
        input.setText(getCurrentProfile().getName());
        newBuilder.setView(input);
        newBuilder.setPositiveButton(getActivity().getResources().getText(android.R.string.ok),
(dialog, whichButton) -> {
            String value = input.getText().toString();

            if (!value.isEmpty()) {
                // Get current profile
                Profile temp = getCurrentProfile();
                // Rename it
                temp.setName(value);
                // Commit it
                mDbProfils.updateProfile(temp);
                // Make it the current.
                setCurrentProfile(value);
            }       });

newBuilder.setNegativeButton(getActivity().getResources().getText(android.R.string.cancel),
(dialog, whichButton) -> {
        });
        newBuilder.show();
        return true;
    }

    private void setDrawerTitle(String pProfilName) {
        mDrawerAdapter.getItem(0).setTitle(pProfilName);
        mDrawerAdapter.notifyDataSetChanged();
        mDrawerLayout.invalidate();
    }

   private void selectItem(int position) {
        // Highlight the selected item, update the title, and close the drawer
        mDrawerList.setItemChecked(position, true);
        //setTitle(mPlanetTitles[position]);
        mDrawerLayout.closeDrawer(mDrawerList);
    }

    @Override
    public void setTitle(CharSequence title) {
        getSupportActionBar().setTitle(title);
    }

    @Override
    protected void onPostCreate(Bundle savedInstanceState) {
        super.onPostCreate(savedInstanceState);
        // Sync the toggle state after onRestoreInstanceState has occurred.
        mDrawerToggle.syncState();
    }

    @Override
    public void onConfigurationChanged(@NonNull Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        mDrawerToggle.onConfigurationChanged(newConfig);
    }
    private void showFragment(String pFragmentName) {
        if (currentFragmentName.equals(pFragmentName))
            return; // If this is already the current fragment, do no replace.

        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction ft = fragmentManager.beginTransaction();

        // Then show the fragments
        if (pFragmentName.equals(FONTESPAGER)) {
            ft.replace(R.id.fragment_container, getFontesPagerFragment(), FONTESPAGER);
        } else if (pFragmentName.equals(WEIGHT)) {
            ft.replace(R.id.fragment_container, getWeightFragment(), WEIGHT);
        } else if (pFragmentName.equals(SETTINGS)) {
            ft.replace(R.id.fragment_container, getSettingsFragment(), SETTINGS);
        } else if (pFragmentName.equals(MACHINES)) {
            ft.replace(R.id.fragment_container, getMachineFragment(), MACHINES);
        } else if (pFragmentName.equals(WORKOUTS)) {
            ft.replace(R.id.fragment_container, getWorkoutListFragment(), WORKOUTS);
        } else if (pFragmentName.equals(ABOUT)) {
            ft.replace(R.id.fragment_container, getAboutFragment(), ABOUT);
        } else if (pFragmentName.equals(BODYTRACKING)) {
            ft.replace(R.id.fragment_container, getBodyPartFragment(), BODYTRACKING);
        } else if (pFragmentName.equals(PROFILE)) {
            ft.replace(R.id.fragment_container, getProfileFragment(), PROFILE);
        }
        currentFragmentName = pFragmentName;
        ft.commit();
    }

    @Override
    protected void onStop() {
        super.onStop();
        savePreferences();
    }

    public Profile getCurrentProfile() {
        return appViMo.getProfile().getValue();
    }

    public void setCurrentProfile(String newProfileName) {
        Profile newProfile = this.mDbProfils.getProfile(newProfileName);
        setCurrentProfile(newProfile);
    }
```

```
   public void setCurrentProfile(Profile newProfil) {
      appViMo.setProfile(newProfil);
   }

   private void setPhotoProfile(String path) {
      ImageUtil imgUtil = new ImageUtil();

      // Check if path is pointing to a thumb else create it and use it.
      String thumbPath = imgUtil.getThumbPath(path);
      boolean success = false;
      if (thumbPath != null) {
         success = ImageUtil.setPic(roundProfile, thumbPath);
         mDrawerAdapter.getItem(0).setImg(thumbPath);
      }
      if (!success) {
         roundProfile.setImageDrawable(AppCompatResources.getDrawable(getActivity(),
R.drawable.ic_person));
         mDrawerAdapter.getItem(0).setImg(null); // Img has priority over Resource so it needs to
be reset
         mDrawerAdapter.getItem(0).setImgResID(R.drawable.ic_person);
      }
      mDrawerAdapter.notifyDataSetChanged();
      mDrawerLayout.invalidate();
   }
   public String getCurrentMachine() {
      return mCurrentMachine;
   }
   public void setCurrentMachine(String newMachine) {
      mCurrentMachine = newMachine;
   }
   public MainActivity getActivity() {
      return this;
   }
   private void loadPreferences() {
      // Restore preferences
      SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
      mCurrentProfilID = settings.getLong("currentProfil", -1); // return -1 if it doesn't exist
      mIntro014Launched = settings.getBoolean("intro014Launched", false);
      mMigrationBD15done = settings.getBoolean("migrationBD15done", false);
      mMigrationToScopedStoragedone =
settings.getBoolean("migrationToScopedStoragedone", false);
   }
```

```
  private void savePreferences() {
    // Restore preferences
    SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
    SharedPreferences.Editor editor = settings.edit();
    editor.putLong("currentProfil", mCurrentProfilID);
    editor.putBoolean("intro014Launched", mIntro014Launched);
    editor.putBoolean("migrationBD15done", mMigrationBD15done);
    editor.putBoolean("migrationToScopedStoragedone", mMigrationToScopedStoragedone);
    editor.apply();
  }
  private FontesPagerFragment getFontesPagerFragment() {
    if (mpFontesPagerFrag == null)
      mpFontesPagerFrag = (FontesPagerFragment)
getSupportFragmentManager().findFragmentByTag(FONTESPAGER);
    if (mpFontesPagerFrag == null)
      mpFontesPagerFrag = FontesPagerFragment.newInstance(FONTESPAGER, 6);
    return mpFontesPagerFrag;
  }
  private WeightFragment getWeightFragment() {
    if (mpWeightFrag == null)
      mpWeightFrag = (WeightFragment)
getSupportFragmentManager().findFragmentByTag(WEIGHT);
    if (mpWeightFrag == null) mpWeightFrag = WeightFragment.newInstance(WEIGHT, 5);
    return mpWeightFrag;
  }

  private ProfileFragment getProfileFragment() {
    if (mpProfileFrag == null)
      mpProfileFrag = (ProfileFragment)
getSupportFragmentManager().findFragmentByTag(PROFILE);
    if (mpProfileFrag == null) mpProfileFrag = ProfileFragment.newInstance(PROFILE, 10);
    return mpProfileFrag;
  }
  private MachineFragment getMachineFragment() {
    if (mpMachineFrag == null)
      mpMachineFrag = (MachineFragment)
getSupportFragmentManager().findFragmentByTag(MACHINES);
    if (mpMachineFrag == null) mpMachineFrag =
MachineFragment.newInstance(MACHINES, 7);
    return mpMachineFrag;
  }
  private AboutFragment getAboutFragment() {
    if (mpAboutFrag == null)
      mpAboutFrag = (AboutFragment)
```

```
getSupportFragmentManager().findFragmentByTag(ABOUT);
        if (mpAboutFrag == null) mpAboutFrag = AboutFragment.newInstance(ABOUT, 6);

        return mpAboutFrag;
    }

    private BodyPartListFragment getBodyPartFragment() {
        if (mpBodyPartListFrag == null)
            mpBodyPartListFrag = (BodyPartListFragment)
getSupportFragmentManager().findFragmentByTag(BODYTRACKING);
        if (mpBodyPartListFrag == null)
            mpBodyPartListFrag = BodyPartListFragment.newInstance(BODYTRACKING, 9);

        return mpBodyPartListFrag;
    }
    private ProgramListFragment getWorkoutListFragment() {
        if (mpWorkoutListFrag == null)
            mpWorkoutListFrag = (ProgramListFragment)
getSupportFragmentManager().findFragmentByTag(WORKOUTS);
        if (mpWorkoutListFrag == null)
            mpWorkoutListFrag = ProgramListFragment.newInstance(WORKOUTS, 10);
        return mpWorkoutListFrag;
    }
    private SettingsFragment getSettingsFragment() {
        if (mpSettingFrag == null)
            mpSettingFrag = (SettingsFragment)
getSupportFragmentManager().findFragmentByTag(SETTINGS);
        if (mpSettingFrag == null) mpSettingFrag = SettingsFragment.newInstance(SETTINGS, 8);
        return mpSettingFrag;
    }
    public Toolbar getActivityToolbar() {
        return top_toolbar;
    }
    public void showMP3Toolbar(boolean show) {
        Toolbar mp3toolbar = this.findViewById(R.id.musicToolbar);
        if (!show) {
            mp3toolbar.setVisibility(View.GONE);
        } else {
            mp3toolbar.setVisibility(View.VISIBLE);
        }
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CODE_INTRO) {
            if (resultCode == RESULT_OK) {
                initActivity();
                mIntro014Launched = true;
                initDEBUGdata();
                this.savePreferences();
            } else {
                // Cancelled the intro. You can then e.g. finish this activity too.
                finish();
            }
        } else if (resultCode == RESULT_OK && requestCode == IMPORT_DATABASE) {
            Uri file;
            if (data != null) {
                file = data.getData();
                CVSManager cvsMan = new CVSManager(getActivity().getBaseContext());
                InputStream inputStream;
                try {
                    inputStream = getContentResolver().openInputStream(file);
                } catch (FileNotFoundException e) {
                    e.printStackTrace();
                    inputStream = null;
                }

                if (cvsMan.importDatabase(inputStream, appViMo.getProfile().getValue())) {
                    KToast.successToast(getActivity(), getCurrentProfile().getName() + ": " +
getActivity().getResources().getText(R.string.imported_successfully), Gravity.BOTTOM,
KToast.LENGTH_LONG);
                } else {
                    KToast.errorToast(getActivity(), getCurrentProfile().getName() + ": " +
getActivity().getResources().getText(R.string.import_failed), Gravity.BOTTOM,
KToast.LENGTH_LONG);
                }
            }
        } else if (resultCode == RESULT_OK && requestCode == OPEN_MUSIC_FILE) {
            Uri uri;
            if (data != null) {
                uri = data.getData();
                // Return for MusicController Choose file
                musicController.OpenMusicFileIntentResult(uri);
                int takeFlags = data.getFlags();
                takeFlags &= (Intent.FLAG_GRANT_READ_URI_PERMISSION
                        | Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
                // Check for the freshest data.
```

```
                getContentResolver().takePersistableUriPermission(uri, takeFlags);
            }
        }
    }
    @Override
    public void onBackPressed() {
        int index = getActivity().getSupportFragmentManager().getBackStackEntryCount() - 1;
        if (index >= 0) { // Si on est dans une sous activité
            super.onBackPressed();
            getActivity().getSupportActionBar().show();
        } else { // Si on est la racine, avec il faut cliquer deux fois
            if (mBackPressed + TIME_INTERVAL > System.currentTimeMillis()) {
                super.onBackPressed();
                return;
            } else {
                Toast.makeText(getBaseContext(), R.string.pressBackAgain,
Toast.LENGTH_SHORT).show();
            }
            mBackPressed = System.currentTimeMillis();
        }
    }
    public void initActivity() {
        // Initialisation des objets DB
        mDbProfils = new DAOProfile(this.getApplicationContext());

        // Pour la base de donnee profil, il faut toujours qu'il y ai au moins un profil
        mCurrentProfile = mDbProfils.getProfile(mCurrentProfilID);
        if (mCurrentProfile == null) { // au cas ou il y aurait un probleme de synchro
            try {
                List<Profile> lList = mDbProfils.getAllProfiles(mDbProfils.getReadableDatabase());
                mCurrentProfile = lList.get(0);
            } catch (IndexOutOfBoundsException e) {
                this.CreateNewProfile();
            }
        }

        if (mCurrentProfile != null) setCurrentProfile(mCurrentProfile.getName());
    }

    private class DrawerItemClickListener implements ListView.OnItemClickListener {
        @Override
        public void onItemClick(AdapterView parent, View view, int position, long id) {
            selectItem(position);
            // Insert the fragment by replacing any existing fragment
            switch (position) {
                case 0:
                    showFragment(PROFILE);
                    setTitle(getString(R.string.ProfileLabel));
                    break;
                case 1:
                    showFragment(FONTESPAGER);
                    setTitle(getResources().getText(R.string.menu_Workout));
                    break;
                case 2:
                    showFragment(MACHINES);
                    setTitle(getResources().getText(R.string.MachinesLabel));
                    break;
                case 3:
                    showFragment(WORKOUTS);
                    setTitle(getString(R.string.workout_list_menu_item));
                    break;
                case 4:
                    showFragment(WEIGHT);
                    setTitle(getResources().getText(R.string.weightMenuLabel));
                    break;
                case 5:
                    showFragment(BODYTRACKING);
                    setTitle(getResources().getText(R.string.bodytracking));
                    break;
                case 6:
                    showFragment(SETTINGS);
                    setTitle(getResources().getText(R.string.SettingLabel));
                    break;
                case 7:
                    showFragment(ABOUT);
                    setTitle(getResources().getText(R.string.AboutLabel));
                    break;
                default:
                    showFragment(FONTESPAGER);
                    setTitle(getResources().getText(R.string.FonteLabel));
            }
        }
    }
```

Chapter – 5

SNAPSHOTS

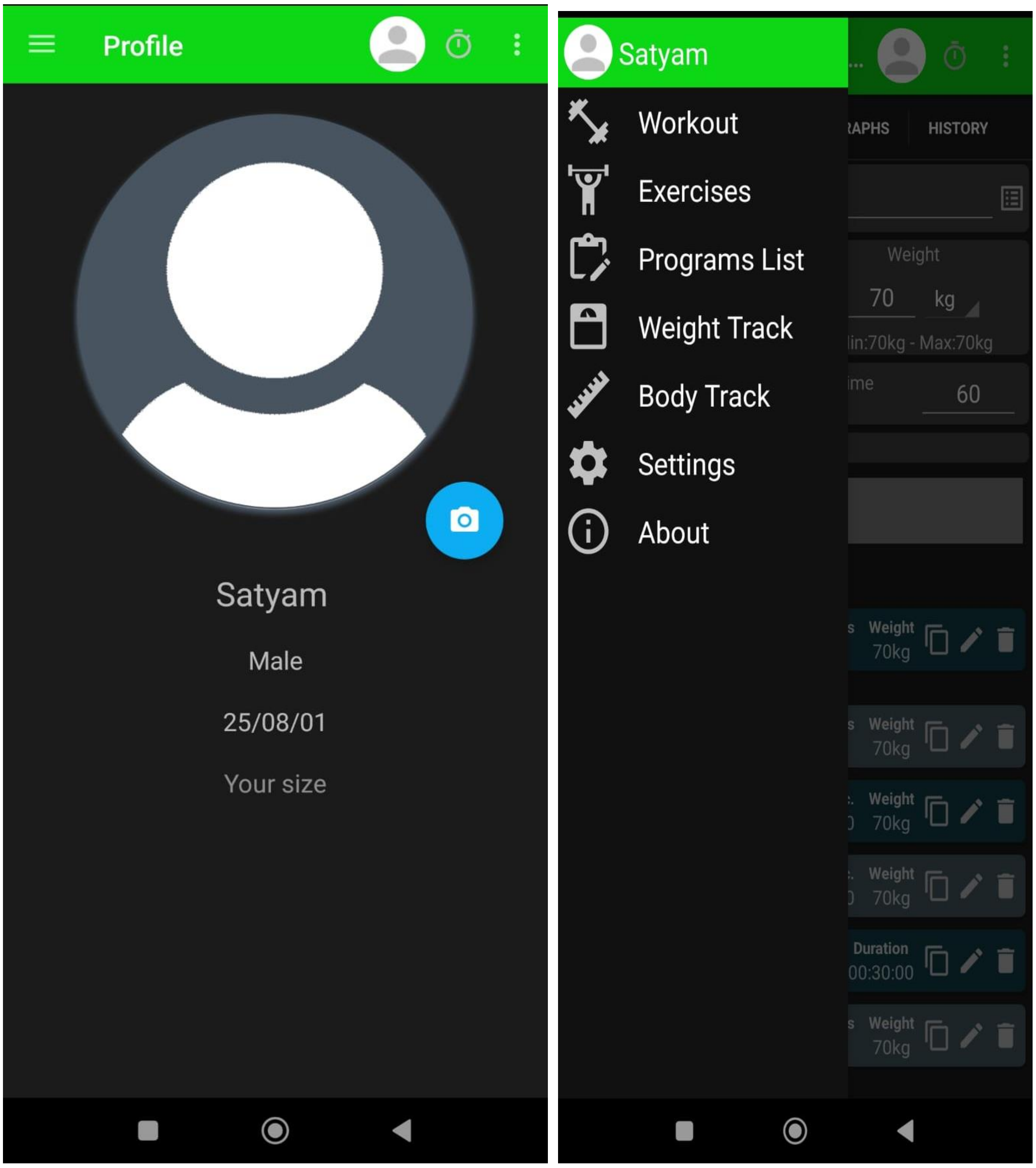
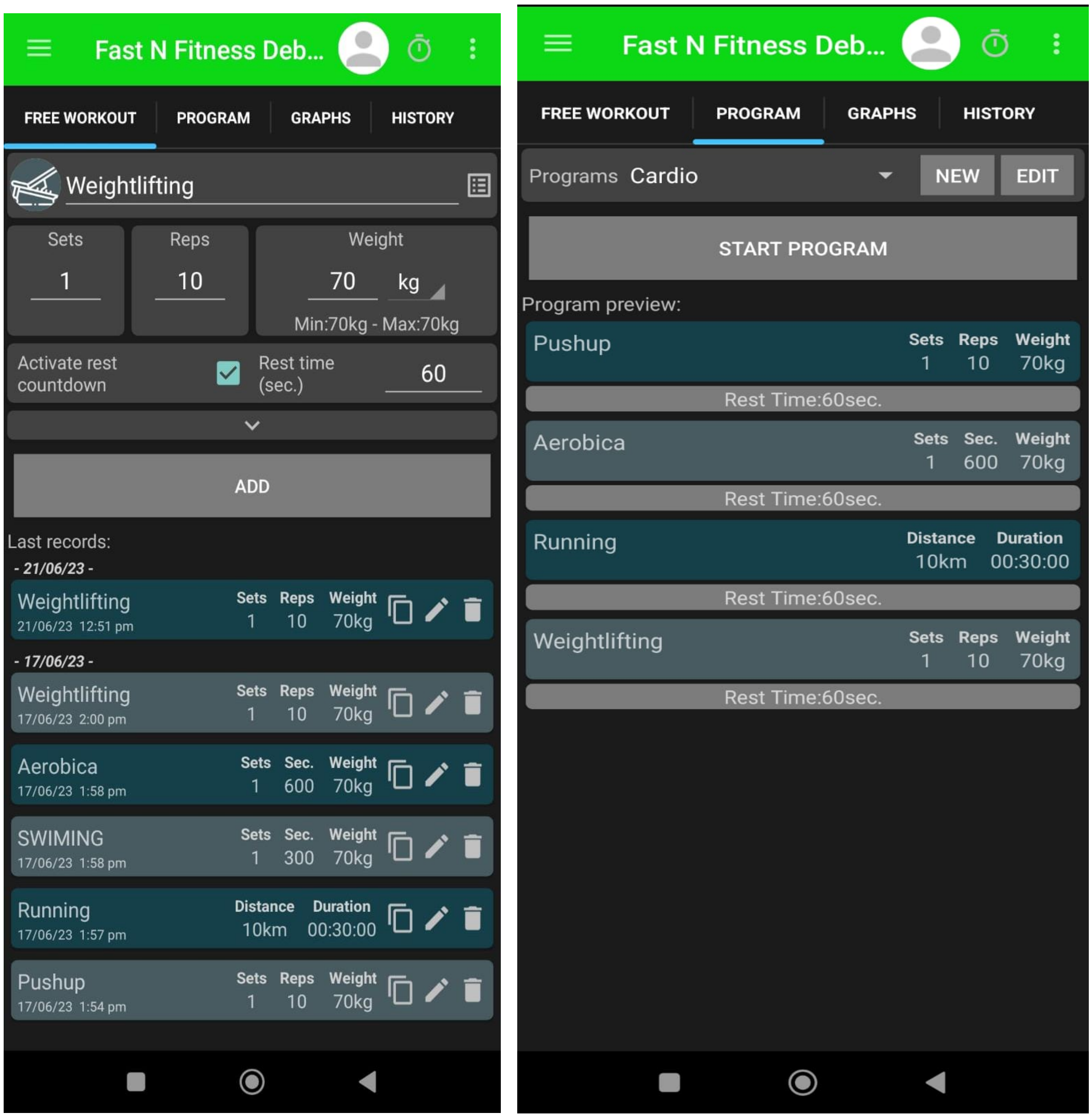


Figure 5.1: Profile View and Menu View



Fast N Fitness Deb...

FREE WORKOUT | PROGRAM | GRAPHS | HISTORY

Weightlifting

Sets	Reps	Weight
1	10	70 kg

Min:70kg - Max:70kg

Activate rest countdown ✓ Rest time (sec.) 60

ADD

Last records:

- 21/06/23 -

	Sets	Reps	Weight
Weightlifting 21/06/23 12:51 pm	1	10	70kg

- 17/06/23 -

Weightlifting 17/06/23 2:00 pm	Sets 1	Reps 10	Weight 70kg
Aerobica 17/06/23 1:58 pm	Sets 1	Sec. 600	Weight 70kg
SWIMING 17/06/23 1:58 pm	Sets 1	Sec. 300	Weight 70kg
Running 17/06/23 1:57 pm	Distance 10km	Duration 00:30:00	
Pushup 17/06/23 1:54 pm	Sets 1	Reps 10	Weight 70kg

Fast N Fitness Deb...

FREE WORKOUT | PROGRAM | GRAPHS | HISTORY

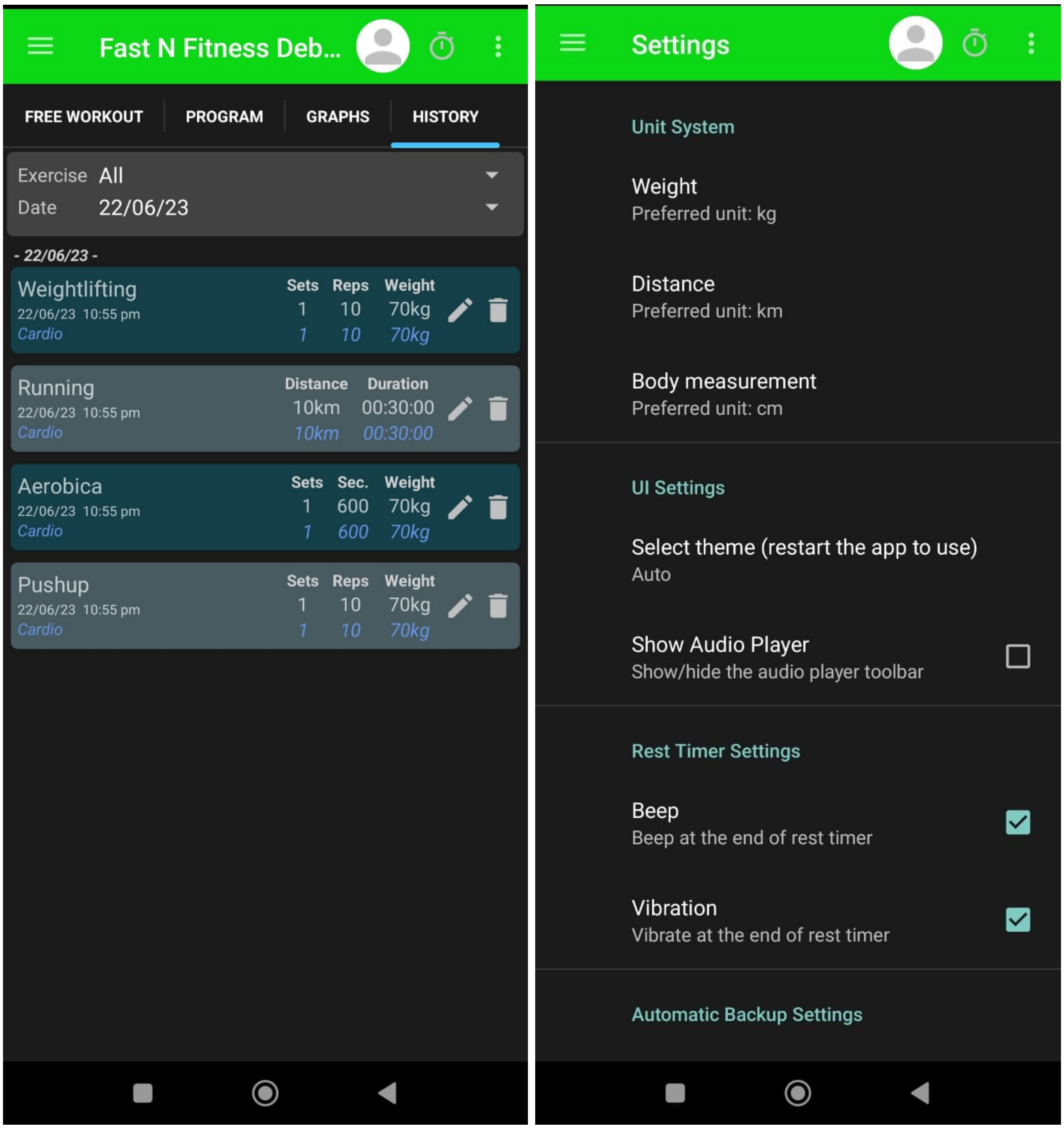
Programs Cardio NEW EDIT

START PROGRAM

Program preview:

Pushup	Sets 1	Reps 10	Weight 70kg
Rest Time:60sec.			
Aerobica	Sets 1	Sec. 600	Weight 70kg
Rest Time:60sec.			
Running	Distance 10km	Duration 00:30:00	
Rest Time:60sec.			
Weightlifting	Sets 1	Reps 10	Weight 70kg
Rest Time:60sec.			

Figure 5.2: Dashboard View (Free Workout and Programs)


Fast N Fitness Deb...

FREE WORKOUT | PROGRAM | GRAPHS | HISTORY

Exercise All
Date 22/06/23

- 22/06/23 -

Weightlifting
22/06/23 10:55 pm
Cardio

Sets	Reps	Weight
1	10	70kg
1	10	70kg

Running
22/06/23 10:55 pm
Cardio

Distance	Duration
10km	00:30:00
10km	00:30:00

Aerobica
22/06/23 10:55 pm
Cardio

Sets	Sec.	Weight
1	600	70kg
1	600	70kg

Pushup
22/06/23 10:55 pm
Cardio

Sets	Reps	Weight
1	10	70kg
1	10	70kg

Settings

Unit System

Weight
Preferred unit: kg

Distance
Preferred unit: km

Body measurement
Preferred unit: cm

UI Settings

Select theme (restart the app to use)
Auto

Show Audio Player ☐
Show/hide the audio player toolbar

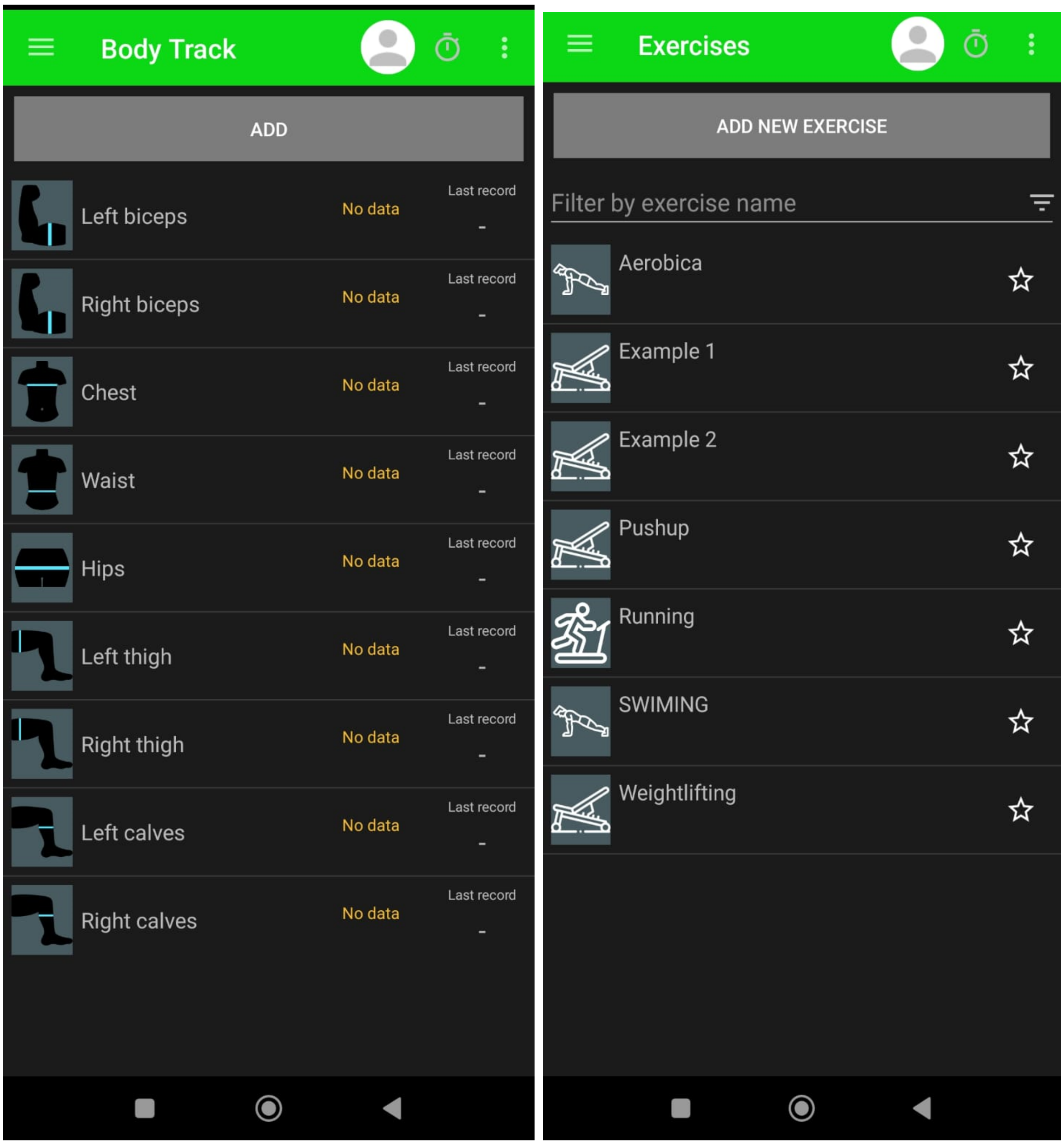
Rest Timer Settings

Beep ✓
Beep at the end of rest timer

Vibration ✓
Vibrate at the end of rest timer

Automatic Backup Settings

Figure 5.3: History and Settings



Body Track

ADD

Left biceps	No data	Last record -
Right biceps	No data	Last record -
Chest	No data	Last record -
Waist	No data	Last record -
Hips	No data	Last record -
Left thigh	No data	Last record -
Right thigh	No data	Last record -
Left calves	No data	Last record -
Right calves	No data	Last record -

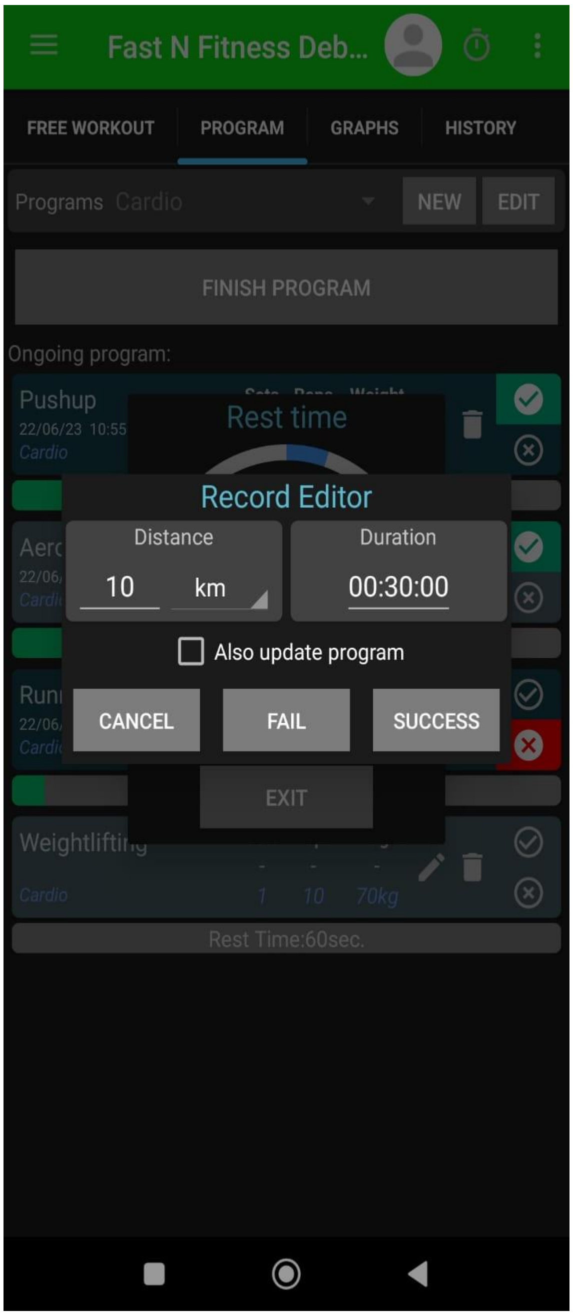
Exercises

ADD NEW EXERCISE

Filter by exercise name

- Aerobica ☆
- Example 1 ☆
- Example 2 ☆
- Pushup ☆
- Running ☆
- SWIMING ☆
- Weightlifting ☆

Figure 5.4: Body Track and List of Exercises



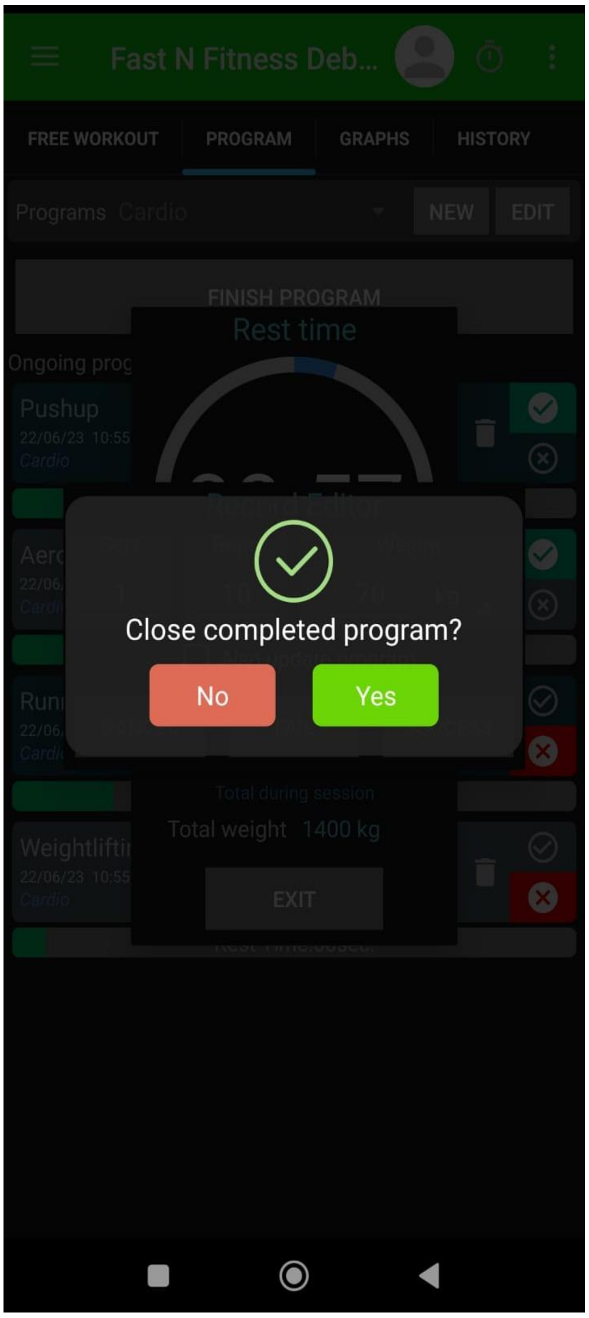


Figure 5.5: Failure of Program and Successful Completion of Program

Chapter – 6

CONCLUSION

“Fast N Fitness” project was done in a perspective of understanding the Android Studio software. Through this project, we have acquired a much deeper knowledge of the Mobile Application Development and understood the process and details of developing a Mobile Application.

We thus would like to emphasize the importance of this project to many other perspectives of Prototyping, Designing, Technical and software concepts which we were unaware of.

6.1 Future Enhancements

- In future, the same project can be enhanced in such a way that we can interact more with the project.
- A vast amount of future work can be possible by following investigations and strategies.
- Wearable integration: Integrate fitness app with wearable devices to track and sync data for a seamless user experience.
- Gamification features: Add game-like elements to the app to motivate users and make fitness activities more enjoyable.

BIBLIOGRAPHY

Reference Books

1) Google Developer Training, "Android Developer Fundamentals Course – Concept Reference”, Google Developer Training Team, 2017.
2) Erik Hellman, “Android Programming – Pushing the Limits”, 1st Edition, Wiley India Pvt Ltd, 2014.

Websites

1) www.google.com
2) www.wikipedia.org
3) www.geeksforgeeks.org