CSE 4002 – MOBILE APPLICATION DEVELOPMENT

MINI PROJECT

TEAM MEMBERS :-

1.19BCN7242-K.LOKESH CHOWDARY.

2.19BCE7324-K.KAMAL.

FASTNFITNESS APPLICATION

SOURCE CODE

JAVA CODE :-

```
package com.easyfitness;

import android.Manifest;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.content.res.Configuration;
import android.os.Bundle;
import android.os.Environment;
import android.preference.PreferenceManager;
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.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
```

```
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.easyfitness.DAO.CVSManager;
import com.easyfitness.DAO.DAOCardio;
import com.easyfitness.DAO.DAOFonte;
import com.easyfitness.DAO.DAOMachine;
import com.easyfitness.DAO.DAOProfil;
import com.easyfitness.DAO.DatabaseHelper;
import com.easyfitness.DAO.Fonte;
import com.easyfitness.DAO.Machine;
import com.easyfitness.DAO.Profile;
import com.easyfitness.DAO.bodymeasures.BodyMeasure;
import com.easyfitness.DAO.bodymeasures.DAOBodyMeasure;
import com.easyfitness.DAO.cardio.DAOOldCardio;
import com.easyfitness.DAO.cardio.OldCardio;
import com.easyfitness.bodymeasures.BodyPartListFragment;
import com.easyfitness.fonte.FontesPagerFragment;
import com.easyfitness.intro.MainIntroActivity;
import com.easyfitness.machines.MachineFragment;
import com.easyfitness.utils.CustomExceptionHandler;
import com.easyfitness.utils.FileChooserDialog;
import com.easyfitness.utils.ImageUtil;
import com.easyfitness.utils.MusicController;
import com.mikhaellopez.circularimageview.CircularImageView;
import com.onurkaganaldemir.ktoastlib.KToast;
import com.theartofdev.edmodo.cropper.CropImage;
import com.theartofdev.edmodo.cropper.CropImageView;

import java.io.File;
import java.util.ArrayList;
import java.util.List;

//import com.crashlytics.android.Crashlytics;

//import io.fabric.sdk.android.Fabric;

public class MainActivity extends AppCompatActivity {

    private static final int TIME_INTERVAL = 2000; // # milliseconds, desired
time passed between two back presses.
    public static String FONTESPAGER = "FontePager";
    public static String FONTES = "Fonte";
    public static String HISTORY = "History";
    public static String GRAPHIC = "Graphics";
    public static String CARDIO = "Cardio";
    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 PREFS_NAME = "prefsfile";
    private final int REQUEST_CODE_INTRO = 111;
    private final int MY_PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE = 1001;
    CustomDrawerAdapter mDrawerAdapter;
    List<DrawerItem> dataList;
    private FontesPagerFragment mpFontesPagerFrag = null;
    private CardioFragment mpCardioFrag = 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 String currentFragmentName = "";
    private DAOProfil mDbProfils = null;
    private Profile mCurrentProfile = null;
    private long mCurrentProfilID = -1;
    private String m_importCVSchosenDir = "";
    private Toolbar top_toolbar = null;
    /* Navigation Drawer */
    private DrawerLayout mDrawerLayout = null;
    private ListView mDrawerList = null;
    private ActionBarDrawerToggle mDrawerToggle = null;
    private CharSequence mDrawerTitle;
    private CharSequence mTitle;
    private MusicController musicController = new MusicController(this);
    private CircularImageView roundProfile = null;
    private String mCurrentMachine = "";
    private boolean mIntro014Launched = false;
    private boolean mMigrationBD15done = false;
    private PopupMenu.OnMenuItemClickListener onMenuItemClick = item -> {
        switch (item.getItemId()) {
            case R.id.create_newprofil:
                getActivity().CreateNewProfil();
                return true;
            case R.id.photo_profil:
                String[] optionListArray = new String[2];
                optionListArray[0] =
getActivity().getResources().getString(R.string.camera);
                optionListArray[1] =
getActivity().getResources().getString(R.string.gallery);
                //profilListArray[2] = "Remove Image";

                //requestPermissionForWriting(pF);

                AlertDialog.Builder itemActionbuilder = new
AlertDialog.Builder(getActivity());
                itemActionbuilder.setTitle("").setItems(optionListArray, new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        ListView lv = ((AlertDialog) dialog).getListView();

                        switch (which) {
                            // Galery
                            case 1:
                                Intent photoPickerIntent = new
```

```
Intent(Intent.ACTION_PICK);
                                photoPickerIntent.setType("image/*");
                                startActivityForResult(photoPickerIntent,
ImageUtil.REQUEST_PICK_GALERY_PHOTO);
                                break;
                            // Camera with Cropping
                            case 0:
                                //dispatchTakePictureIntent(mF);
                                // start picker to get image for cropping and
then use the image in cropping activity
                                CropImage.activity()

.setGuidelines(CropImageView.Guidelines.ON)
                                        .start(getActivity());
                                break;
                            case 2: // Delete picture

                                break;
                            // Camera
                            default:
                        }
                    }
                });
                itemActionbuilder.show();
                return true;
            case R.id.change_profil:
                String[] profilListArray =
getActivity().mDbProfils.getAllProfil();

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

changeProfilbuilder.setTitle(getActivity().getResources().getText(R.string.pr
ofil_select_profil))
                    .setItems(profilListArray, (dialog, which) -> {
                        ListView lv = ((AlertDialog) dialog).getListView();
                        Object checkedItem = lv.getAdapter().getItem(which);
                        setCurrentProfil(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.getAllProfil();

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

deleteProfilbuilder.setTitle(getActivity().getResources().getText(R.string.pr
ofil_select_profil_to_delete))
                    .setItems(profildeleteListArray, (dialog, which) -> {
```

```
                        ListView lv = ((AlertDialog) dialog).getListView();
                        Object checkedItem = lv.getAdapter().getItem(which);
                        if
(getCurrentProfil().getName().equals(checkedItem.toString())) {
                            KToast.errorToast(getActivity(),
getActivity().getResources().getText(R.string.impossibleToDeleteProfile).toSt
ring(), Gravity.BOTTOM, KToast.LENGTH_LONG);
                        } else {
                            Profile profileToDelete =
mDbProfils.getProfil(checkedItem.toString());
                            mDbProfils.deleteProfil(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;
        }
    };
    private long mBackPressed;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        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 (mpCardioFrag == null) mpCardioFrag =
CardioFragment.newInstance(CARDIO, 4);
            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, 6);
            if (mpMachineFrag == null) mpMachineFrag =
MachineFragment.newInstance(MACHINES, 7);
            if (mpBodyPartListFrag == null)
```

```
                mpBodyPartListFrag =
BodyPartListFragment.newInstance(BODYTRACKING, 9);
        } else {
            mpFontesPagerFrag = (FontesPagerFragment)
getSupportFragmentManager().getFragment(savedInstanceState, FONTESPAGER);
            mpCardioFrag = (CardioFragment)
getSupportFragmentManager().getFragment(savedInstanceState, CARDIO);
            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);
        }

        loadPreferences();

        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) {
                    String exerciseName = "";
                    Machine m = lDAOMachine.getMachine(record.getExercice());
                    exerciseName = record.getExercice();
                    if (m != null) { // if a machine exists
                        if (m.getType() == DAOMachine.TYPE_FONTE) { // if it
is not a Cardio type
                            exerciseName = exerciseName + "-Cardio"; // add a
suffix to
                        }
                    }

                    mDbCardio.addCardioRecord(record.getDate(), "00:00",
exerciseName, record.getDistance(), record.getDuration(),
record.getProfil());
                }
                mDbOldCardio.dropTable();

                DAOFonte mDbFonte = new DAOFonte(this);
                List<Fonte> mFonteList =
mDbFonte.getAllBodyBuildingRecords();
                for (Fonte record : mFonteList) {
                    mDbFonte.updateRecord(record); // Automatically update
record Type
                }
                ArrayList<Machine> machineList =
```

```
lDAOMachine.getAllMachinesArray();
                for (Machine record : machineList) {
                    lDAOMachine.updateMachine(record); // Reset all the
fields on machines.
                }
            }
            mMigrationBD15done = true;
            savePreferences();
        }

        /* creation de l'arborescence de l'application */
        File folder = new File(Environment.getExternalStorageDirectory() +
"/FastnFitness");
        boolean success = true;
        if (!folder.exists()) {
            success = folder.mkdir();
        }
        if (success) {
            folder = new File(Environment.getExternalStorageDirectory() +
"/FastnFitness/crashreport");
            success = folder.mkdir();
        }

        if (folder.exists()) {
            if (!(Thread.getDefaultUncaughtExceptionHandler() instanceof
CustomExceptionHandler)) {
                Thread.setDefaultUncaughtExceptionHandler(new
CustomExceptionHandler(
                    Environment.getExternalStorageDirectory() +
"/FastnFitness/crashreport"));
            }
        }

        if (savedInstanceState == null) {
            showFragment(FONTESPAGER, false); // 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_profile_black, true);

        dataList.add(drawerTitleItem);
        dataList.add(new
DrawerItem(this.getResources().getString(R.string.menu_Workout),
R.drawable.ic_barbell, true));
        //dataList.add(new
DrawerItem(this.getResources().getString(R.string.CardioMenuLabel),
R.drawable.ic_running, true));
        dataList.add(new
DrawerItem(this.getResources().getString(R.string.MachinesLabel),
R.drawable.ic_machine, true));
        dataList.add(new
```

```
DrawerItem(this.getResources().getString(R.string.weightMenuLabel),
R.drawable.ic_scale, true));
        dataList.add(new
DrawerItem(this.getResources().getString(R.string.bodytracking),
R.drawable.ic_measuring_tape, true));
        dataList.add(new
DrawerItem(this.getResources().getString(R.string.SettingLabel),
R.drawable.ic_params, true));
        dataList.add(new
DrawerItem(this.getResources().getString(R.string.AboutLabel),
R.drawable.ic_action_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();

        // Lance l'intro
        // Tester si l'intro a déjà été lancé
        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();
        }

        SharedPreferences SP =
PreferenceManager.getDefaultSharedPreferences(getBaseContext());
        boolean bShowMP3 = SP.getBoolean("prefShowMP3", false);
        this.showMP3Toolbar(bShowMP3);
    }
```

```
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        // Always call the superclass so it can restore the view hierarchy
        super.onRestoreInstanceState(savedInstanceState);
    }

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

        //Save the fragment's instance
        if (getFontesPagerFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, FONTESPAGER,
mpFontesPagerFrag);
        if (getCardioFragment().isAdded())
            getSupportFragmentManager().putFragment(outState, CARDIO,
mpCardioFrag);
        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);
    }

    @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) {
        final MenuItem alertMenuItem = menu.findItem(R.id.action_profil);

        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);
    }

    private void exportDatabase() {
        // Here, thisActivity is the current activity
        if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.WRITE_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED) {
            // No explanation needed; request the permission
            ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
                MY_PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE);
        } else {
            // Afficher une boite de dialogue pour confirmer
            AlertDialog.Builder exportDbBuilder = new
AlertDialog.Builder(this);


exportDbBuilder.setTitle(getActivity().getResources().getText(R.string.export
_database));

exportDbBuilder.setMessage(getActivity().getResources().getText(R.string.expo
rt_question) + " " + getCurrentProfil().getName() + "?");

            // Si oui, supprimer la base de donnee et refaire un Start.

exportDbBuilder.setPositiveButton(getActivity().getResources().getText(R.stri
ng.global_yes), (dialog, which) -> {
                CVSManager cvsMan = new
CVSManager(getActivity().getBaseContext());
                if (cvsMan.exportDatabase(getCurrentProfil())) {
                    KToast.successToast(getActivity(),
getCurrentProfil().getName() + ": " +
getActivity().getResources().getText(R.string.export_success),
Gravity.BOTTOM, KToast.LENGTH_LONG);
                } else {
                    KToast.errorToast(getActivity(),
getCurrentProfil().getName() + ": " +
getActivity().getResources().getText(R.string.export_failed), Gravity.BOTTOM,
KToast.LENGTH_LONG);
                }

                // Do nothing but close the dialog
                dialog.dismiss();
            });


exportDbBuilder.setNegativeButton(getActivity().getResources().getText(R.stri
ng.global_no), (dialog, which) -> {
                // Do nothing
                dialog.dismiss();
            });
```

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

    @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;
        }

        // Handle presses on the action bar items
        switch (item.getItemId()) {
            case R.id.export_database:
                exportDatabase();
                return true;
            case R.id.import_database:
                // Create DirectoryChooserDialog and register a callback
                FileChooserDialog fileChooserDialog =
                    new FileChooserDialog(this, chosenDir -> {
                        m_importCVSchosenDir = chosenDir;
                        //Toast.makeText(getActivity().getBaseContext(),
"Chosen directory: " +
                        //    chosenDir, Toast.LENGTH_LONG).show();
                        CVSManager cvsMan = new
CVSManager(getActivity().getBaseContext());
                        cvsMan.importDatabase(m_importCVSchosenDir,
getCurrentProfil());
                    });

                fileChooserDialog.setFileFilter("csv");

fileChooserDialog.chooseDirectory(Environment.getExternalStorageDirectory() +
"/FastnFitness/export");
                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.dele
teDB_warning));

                // Si oui, supprimer la base de donnee et refaire un Start.

deleteDbBuilder.setPositiveButton(getActivity().getResources().getText(R.stri
ng.global_yes), (dialog, which) -> {
                    // recupere le premier ID de la liste.
                    List<Profile> lList = mDbProfils.getAllProfils();
                    for (int i = 0; i < lList.size(); i++) {
```

```
                        Profile mTempProfile = lList.get(i);
                        mDbProfils.deleteProfil(mTempProfile.getId());
                    }
                    DAOMachine mDbMachines = new DAOMachine(getActivity());
                    // recupere le premier ID de la liste.
                    List<Machine> lList2 = mDbMachines.getAllMachinesArray();
                    for (int i = 0; i < lList2.size(); i++) {
                        Machine mTemp = lList2.get(i);
                        mDbMachines.delete(mTemp.getId());
                    }

                    // redisplay the intro
                    mIntro014Launched=false;

                    // Do nothing but close the dialog
                    dialog.dismiss();

                    finish();
                });


deleteDbBuilder.setNegativeButton(getActivity().getResources().getText(R.stri
ng.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,
                                           String[] permissions, int[]
grantResults) {
        // If request is cancelled, the result arrays are empty.
        if (requestCode == MY_PERMISSIONS_REQUEST_WRITE_EXTERNAL_STORAGE) {
            if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                KToast.infoToast(this, getString(R.string.access_granted),
```

```
Gravity.BOTTOM, KToast.LENGTH_SHORT);
                exportDatabase();
            } else {
                KToast.infoToast(this,
getString(R.string.another_time_maybe), Gravity.BOTTOM, KToast.LENGTH_SHORT);

            }
        }
    }

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


newProfilBuilder.setTitle(getActivity().getResources().getText(R.string.creat
eProfilTitle));

newProfilBuilder.setMessage(getActivity().getResources().getText(R.string.cre
ateProfilQuestion));

        // Set an EditText view to get user input
        final EditText input = new EditText(this);
        newProfilBuilder.setView(input);


newProfilBuilder.setPositiveButton(getActivity().getResources().getText(R.str
ing.global_ok), (dialog, whichButton) -> {
            String value = input.getText().toString();

            if (value.isEmpty()) {
                CreateNewProfil();
            } else {
                // Create the new profil
                mDbProfils.addProfil(value);
                // Make it the current.
                setCurrentProfil(value);
            }
        });


newProfilBuilder.setNegativeButton(getActivity().getResources().getText(R.str
ing.global_cancel), (dialog, whichButton) -> {
            if (getCurrentProfil() == null) {
                CreateNewProfil();
            }
        });

        newProfilBuilder.show();

        return true;
    }

    public boolean renameProfil() {
        AlertDialog.Builder newBuilder = new AlertDialog.Builder(this);


newBuilder.setTitle(getActivity().getResources().getText(R.string.renameProfi
```

```
lTitle));

newBuilder.setMessage(getActivity().getResources().getText(R.string.renamePro
filQuestion));

        // Set an EditText view to get user input
        final EditText input = new EditText(this);
        input.setText(getCurrentProfil().getName());
        newBuilder.setView(input);


newBuilder.setPositiveButton(getActivity().getResources().getText(R.string.gl
obal_ok), (dialog, whichButton) -> {
            String value = input.getText().toString();

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


newBuilder.setNegativeButton(getActivity().getResources().getText(R.string.gl
obal_cancel), (dialog, whichButton) -> {
        });

        newBuilder.show();

        return true;
    }

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

    /**
     * Swaps fragments in the main content view
     */
    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(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        mDrawerToggle.onConfigurationChanged(newConfig);
    }

    private void showFragment(String pFragmentName) {
        showFragment(pFragmentName, true);
    }

    private void showFragment(String pFragmentName, boolean addToBackStack) {

        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(CARDIO)) {
            ft.replace(R.id.fragment_container, getCardioFragment(), CARDIO);
        } 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(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;
        //if (addToBackStack) ft.addToBackStack(null);
        ft.commit();

    }
```

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

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    public Profile getCurrentProfil() {
        return mCurrentProfile;
    }

    //@SuppressLint("RestrictedApi")
    public void setCurrentProfil(String newProfilName) {
        Profile newProfil = this.mDbProfils.getProfil(newProfilName);
        setCurrentProfil(newProfil);
    }

    public void setCurrentProfil(Profile newProfil) {
        if (newProfil != null)
            if ( mCurrentProfile == null || mCurrentProfilID !=
newProfil.getId() || !mCurrentProfile.equals(newProfil)) {

                mCurrentProfile = newProfil;
                mCurrentProfilID = mCurrentProfile.getId();

                // rafraichit le fragment courant
                FragmentManager fragmentManager =
getSupportFragmentManager();
                //FragmentTransaction ft=fragmentManager.beginTransaction();
                //showFragment(WEIGHT);

                // Moyen de rafraichir tous les fragments. Attention, les
View des fragments peuvent avoir ete detruit.
                // Il faut donc que cela soit pris en compte dans le refresh
des fragments.
                for (int i = 0; i < fragmentManager.getFragments().size();
i++) {
                    if (fragmentManager.getFragments().get(i) != null)

fragmentManager.getFragments().get(i).onHiddenChanged(false);
                }

                setDrawerTitle(mCurrentProfile.getName());
                setPhotoProfile(mCurrentProfile.getPhoto());

                savePreferences();
            }
    }

    public long getCurrentProfilID() {
        return mCurrentProfile.getId();
    }
```

```
    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);
        if (thumbPath != null) {
            ImageUtil.setPic(roundProfile, thumbPath);
            mDrawerAdapter.getItem(0).setImg(thumbPath);
            mDrawerAdapter.notifyDataSetChanged();
            mDrawerLayout.invalidate();
        } else {

roundProfile.setImageDrawable(getActivity().getResources().getDrawable(R.draw
able.ic_profile_black));

mDrawerAdapter.getItem(0).setImgResID(R.drawable.ic_profile_black);
            mDrawerAdapter.getItem(0).setImg(null); // Img has priority over
Resource
            mDrawerAdapter.notifyDataSetChanged();
            mDrawerLayout.invalidate();
        }
    }

    private void savePhotoProfile(String path) {
        mCurrentProfile.setPhoto(path);// Enregistrer sur le profile le path
de la photo.
        mDbProfils.updateProfile(mCurrentProfile);
    }

    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);
    }

    private void savePreferences() {
        // Restore preferences
        SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
        SharedPreferences.Editor editor = settings.edit();
        if (mCurrentProfile != null) editor.putLong("currentProfil",
mCurrentProfile.getId());
        editor.putBoolean("intro014Launched", mIntro014Launched);
```

```
        editor.putBoolean("migrationBD15done", mMigrationBD15done);
        editor.apply();
    }

    private FontesPagerFragment getFontesPagerFragment() {
        if (mpFontesPagerFrag == null)
            mpFontesPagerFrag = (FontesPagerFragment)
getSupportFragmentManager().findFragmentByTag(FONTESPAGER);
        if (mpFontesPagerFrag == null)
            mpFontesPagerFrag = FontesPagerFragment.newInstance(FONTESPAGER,
6);

        return mpFontesPagerFrag;
    }

    private CardioFragment getCardioFragment() {
        if (mpCardioFrag == null)
            mpCardioFrag = (CardioFragment)
getSupportFragmentManager().findFragmentByTag(CARDIO);
        if (mpCardioFrag == null) mpCardioFrag =
CardioFragment.newInstance(CARDIO, 2);

        return mpCardioFrag;
    }

    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 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 restoreToolbar() {
        if (top_toolbar != null) setSupportActionBar(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;
                this.savePreferences();
            } else {
                // Cancelled the intro. You can then e.g. finish this
activity too.
```

```
                finish();
            }
        }
    }

    @Override
    public void onBackPressed() {
        int index =
getActivity().getSupportFragmentManager().getBackStackEntryCount() - 1;
        if (index >= 0) { // Si on est dans une sous activité
            FragmentManager.BackStackEntry backEntry =
getSupportFragmentManager().getBackStackEntryAt(index);
            String tag = backEntry.getName();
            Fragment fragment =
getSupportFragmentManager().findFragmentByTag(tag);
            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 DAOProfil(this.getApplicationContext());

        // Pour la base de donnee profil, il faut toujours qu'il y ai au
moins un profil
        /*if (mDbProfils.getCount() == 0 || mCurrentProfilID == -1) {
            // Ouvre la fenetre de creation de profil
            this.CreateNewProfil();
        } else {*/
        mCurrentProfile = mDbProfils.getProfil(mCurrentProfilID);
        if (mCurrentProfile == null) { // au cas ou il y aurait un probleme
de synchro
            try {
                List<Profile> lList = mDbProfils.getAllProfils();
                mCurrentProfile = lList.get(0);
            } catch (IndexOutOfBoundsException e) {
                this.CreateNewProfil();
            }
        }

        if (mCurrentProfile != null)
setCurrentProfil(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(WEIGHT);

setTitle(getResources().getText(R.string.weightMenuLabel));
                    break;
                case 4:
                    showFragment(BODYTRACKING);
                    setTitle(getResources().getText(R.string.bodytracking));
                    break;
                case 5:
                    showFragment(SETTINGS);
                    setTitle(getResources().getText(R.string.SettingLabel));
                    break;
                case 6:
                    showFragment(ABOUT);
                    setTitle(getResources().getText(R.string.AboutLabel));
                    break;
                default:
                    showFragment(FONTESPAGER);
                    setTitle(getResources().getText(R.string.FonteLabel));
            }
        }
    }
}
```

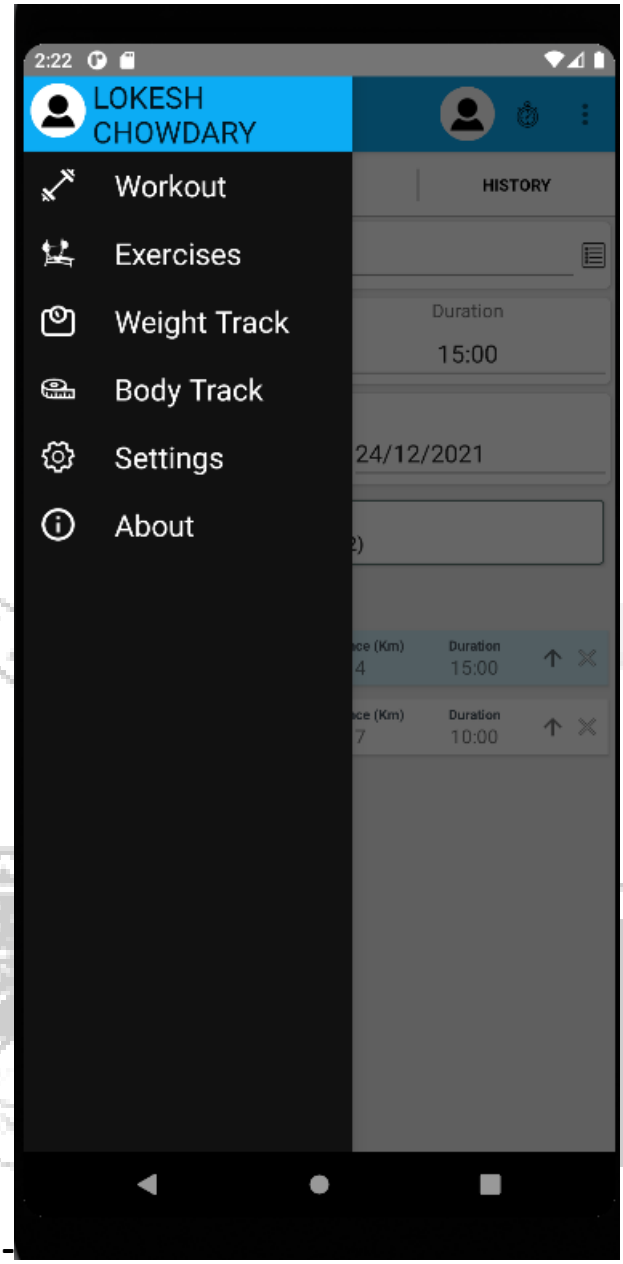
XML CODE :-

```
<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">

    <!-- The main content view -->

    <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/toolbar_background"
            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_profile_black"
                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"
```

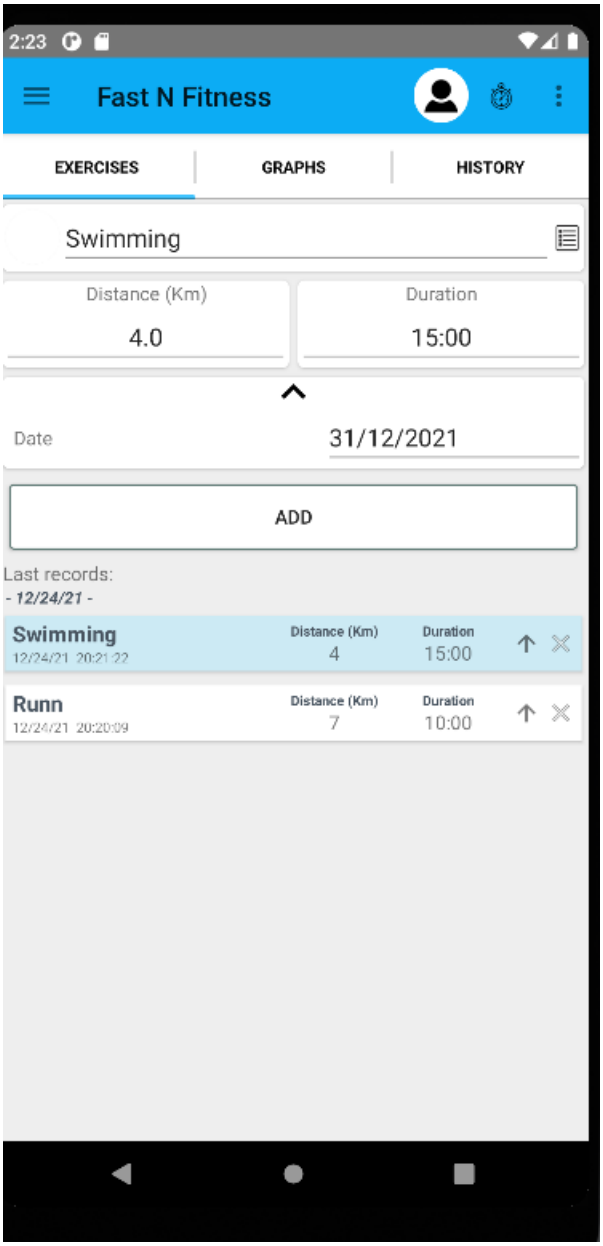
```
            android:padding="0dp"
            app:contentInsetStart="0dp"
            app:titleMargins="0dp">

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="0dp"
                android:orientation="vertical"
                android:padding="0dp">

                <LinearLayout
                    android:id="@+id/playerTopLayout"
                    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" />
```

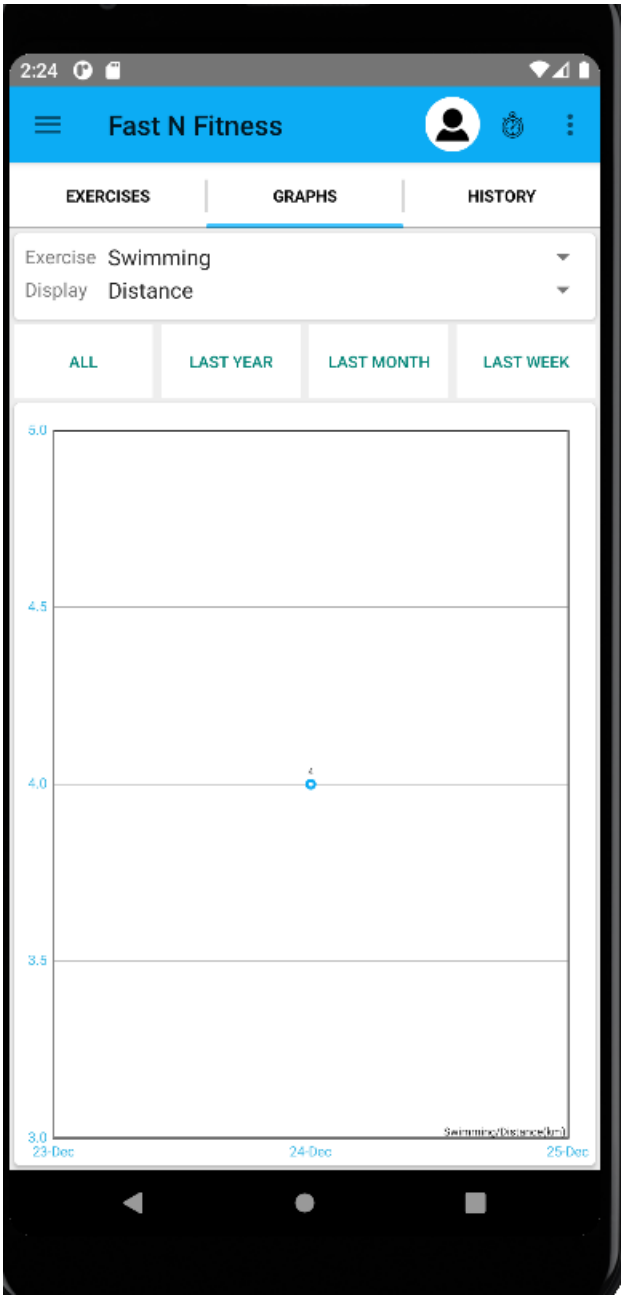
```
                </LinearLayout>

                <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="32dp"
                    android:orientation="horizontal">

                    <ImageButton
                        android:id="@+id/playerPrevious"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center_vertical"
                        android:layout_weight="5"
                        android:adjustViewBounds="false"
                        android:background="@android:color/transparent"
                        android:baselineAlignBottom="false"
                        android:cropToPadding="true"
                        android:padding="5dp"
                        android:src="@drawable/ic_skip_previous_black_36dp"
/>

                    <ImageButton
                        android:id="@+id/playerStop"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center_vertical"
                        android:layout_weight="5"
                        android:adjustViewBounds="false"
                        android:background="@android:color/transparent"
                        android:baselineAlignBottom="false"
                        android:cropToPadding="true"
                        android:src="@drawable/ic_stop_black_36dp" />

                    <ImageButton
                        android:id="@+id/playerPlay"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center_vertical"
                        android:layout_weight="5"
                        android:adjustViewBounds="false"
                        android:background="@android:color/transparent"
                        android:baselineAlignBottom="false"
                        android:cropToPadding="true"
                        android:src="@drawable/ic_play_arrow_black_36dp" />

                    <ImageButton
                        android:id="@+id/playerNext"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center_vertical"
                        android:layout_weight="5"
                        android:adjustViewBounds="false"
                        android:background="@android:color/transparent"
                        android:baselineAlignBottom="false"
                        android:cropToPadding="true"
                        android:src="@drawable/ic_skip_next_black_36dp" />
```

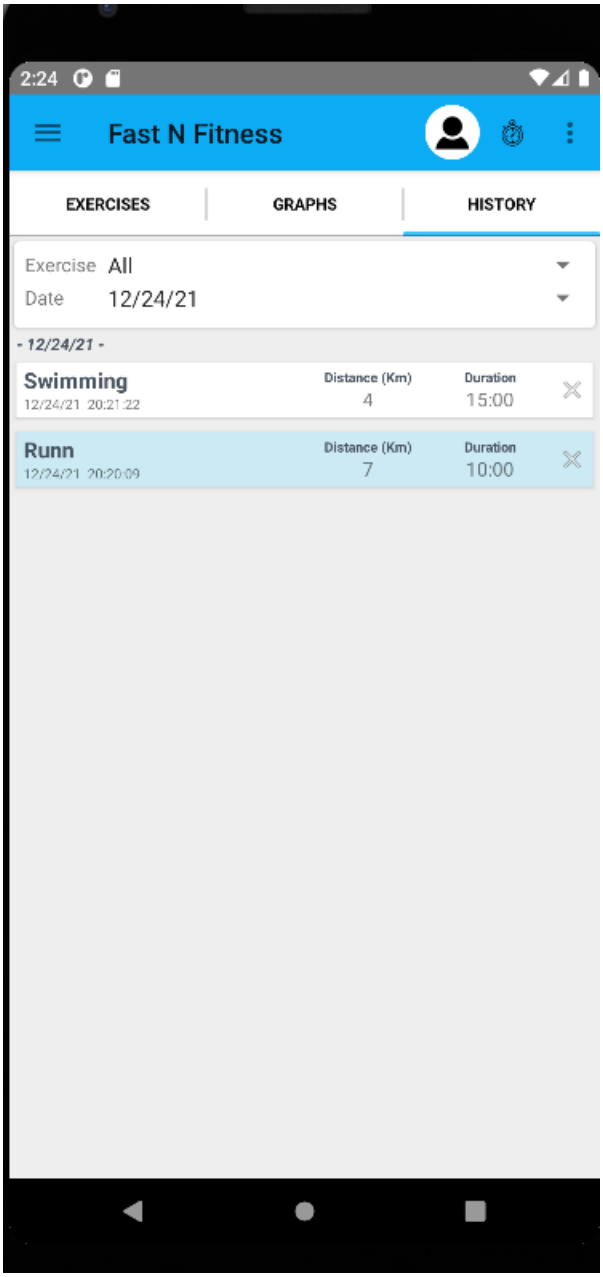
```
                    <ImageButton
                        android:id="@+id/playerLoop"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center_vertical"
                        android:layout_weight="5"
                        android:adjustViewBounds="false"
                        android:background="@android:color/transparent"
                        android:baselineAlignBottom="false"
                        android:cropToPadding="true"
                        android:src="@drawable/ic_replay_blue_36dp" />

                    <ImageButton
                        android:id="@+id/playerList"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_gravity="center_vertical"
                        android:layout_weight="5"
                        android:adjustViewBounds="false"
                        android:background="@android:color/transparent"
                        android:baselineAlignBottom="false"
                        android:cropToPadding="true"
                        android:src="@drawable/ic_library_music_black_36dp"
/>

                </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>

    <!-- The navigation drawer -->

    <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>
```

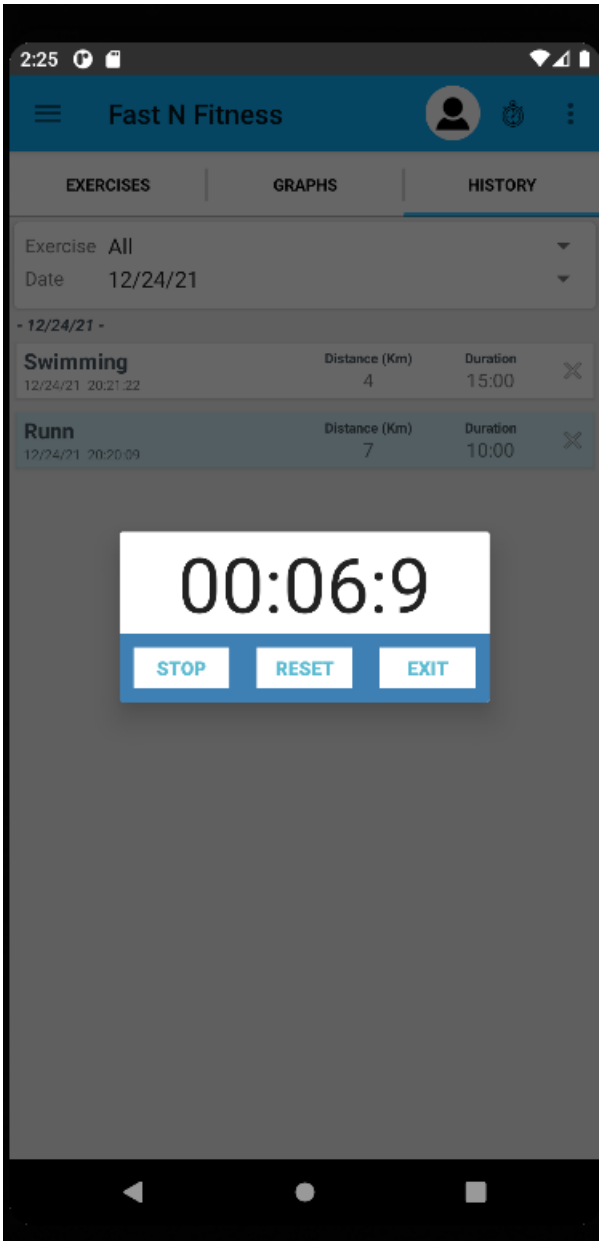


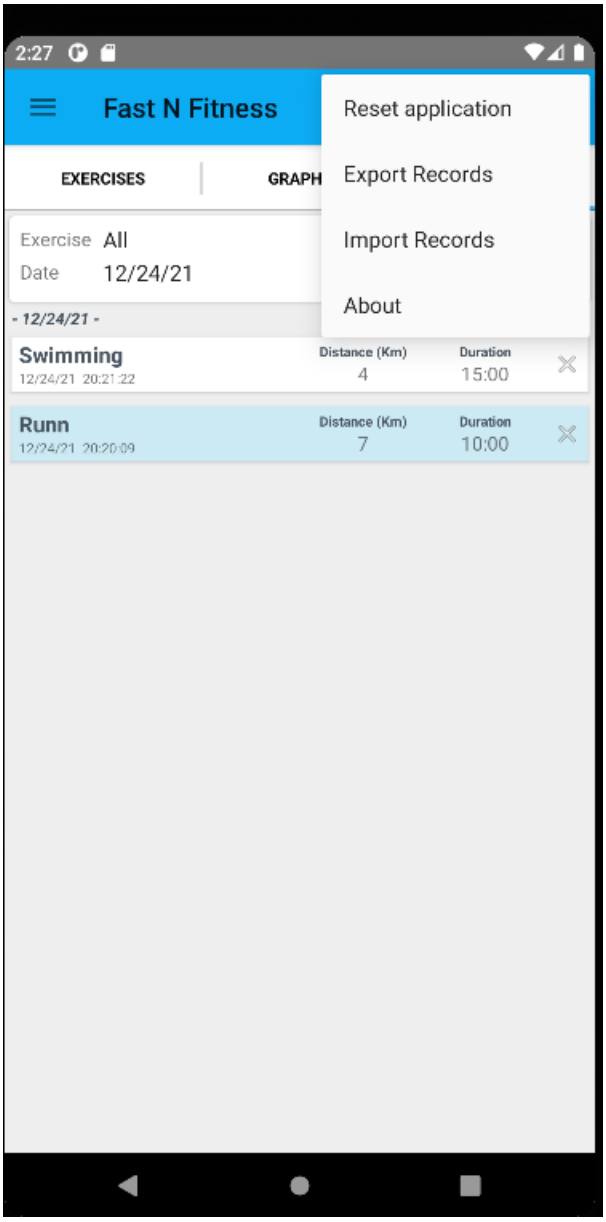
APPLICATION OUTPUT :-

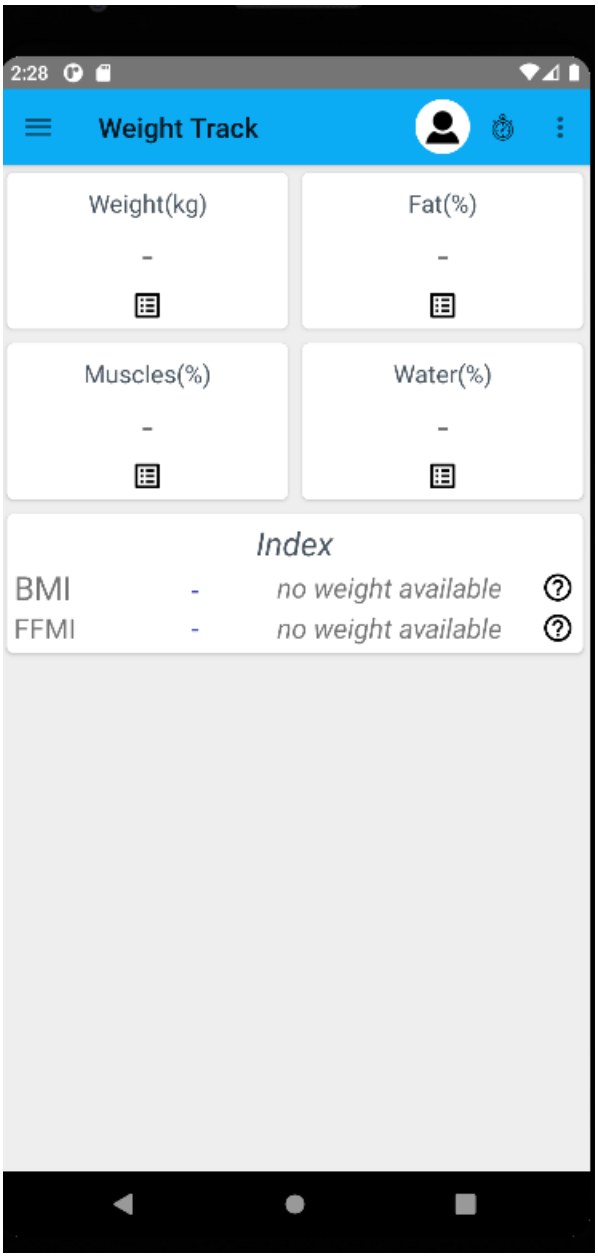

2:23
Fast N Fitness
EXERCISES
GRAPHS
HISTORY
Swimming
Distance (Km)
4.0
Duration
15:00
Date
31/12/2021
ADD
Last records:
- 12/24/21 -
Swimming
12/24/21 20:21:22
Distance (Km)
4
Duration
15:00
Runn
12/24/21 20:20:09
Distance (Km)
7
Duration
10:00

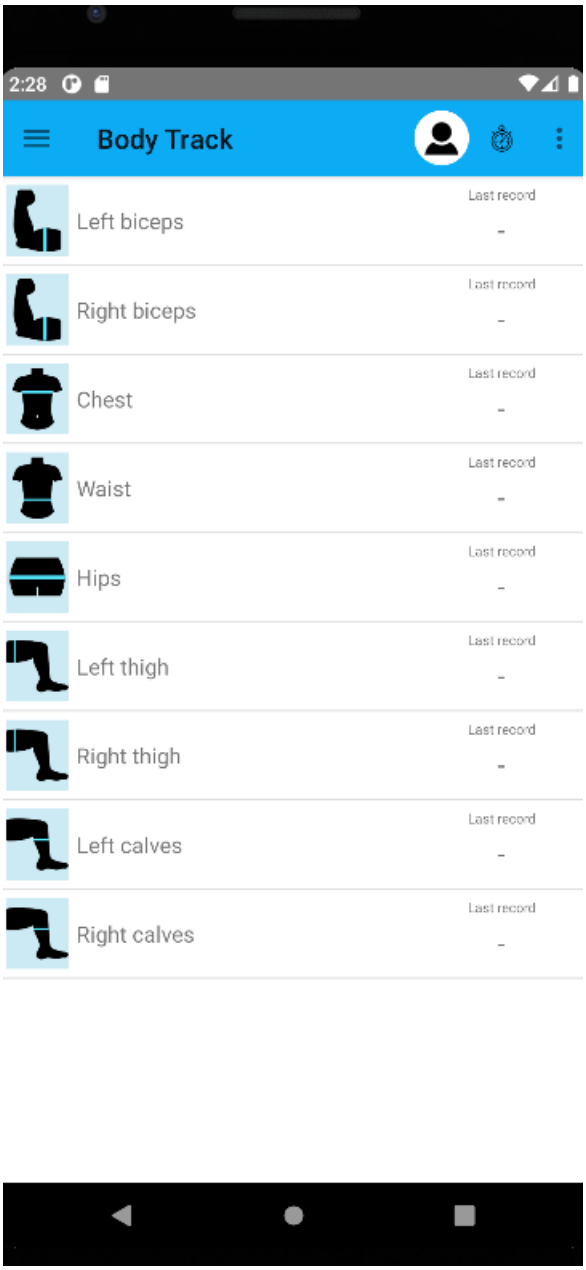

2:24
Fast N Fitness
EXERCISES
GRAPHS
HISTORY
Exercise Swimming
Display Distance
ALL
LAST YEAR
LAST MONTH
LAST WEEK
5.0
4.5
4.0
3.5
3.0
23-Dec
24-Dec
25-Dec


2:24
Fast N Fitness
EXERCISES
GRAPHS
HISTORY
Exercise All
Date 12/24/21
- 12/24/21 -
Swimming
12/24/21 20:21:22
Distance (Km)
4
Duration
15:00
Runn
12/24/21 20:20:09
Distance (Km)
7
Duration
10:00


2:25
Fast N Fitness
EXERCISES
GRAPHS
HISTORY
Exercise All
Date 12/24/21
- 12/24/21 -
Swimming
12/24/21 20:21:22
Distance (Km)
4
Duration
15:00
Runn
12/24/21 20:20:09
Distance (Km)
7
Duration
10:00
00:06:9
STOP
RESET
EXIT


2:27
Fast N Fitness
EXERCISES
GRAPH
Exercise All
Date 12/24/21
- 12/24/21 -
Swimming
12/24/21 20:21:22
Distance (Km)
4
Duration
15:00
Runn
12/24/21 20:20:09
Distance (Km)
7
Duration
10:00
Reset application
Export Records
Import Records
About


2:28
Weight Track
Weight(kg)
-
Fat(%)
-
Muscles(%)
-
Water(%)
-
Index
BMI - no weight available
FFMI - no weight available



----------THE END----------