Term Project
Cool Running

Dylan Duan
Eric Yoo
Andreas Rebsamen
Fall-2019-CNIT-35500-001
Software Development For Mobile Computers
Purdue Polytechnic
Professor: Byung-Cheol Min

December 4, 2019

Contents

1 Readme

The pretty formatted markdown readme is at the end of the document at chapter 6.1 beginning from page 40

```
# CoolRunning
This readme describes the Term Project for CNIT Fall-2019-CNIT-35500-001 at Purdue Polytechnic.

## License
This project is open source and available under MIT License at: https://github.com/Andi-Sail/CoolRunning

## Brief Description
Cool Running is a speed tracking mobile application for android devices that tracks the speed of the user while either walking, running, or riding on a vehicle. If the runner's speed is out of the selected range of the speed, then the program will send the alarm to the runner indicating the status.

### Running Modes
The following four running programs will be implemented:
* Interval
  * The target speed alternates between fast and slow at a constant time interval.
* Increasing speed
  * The target speed begins at a start value and always increase after a time interval. The goal for the user is to keep up as long as possible.
* Constant speed
  * The target speed is set to a constant value and stays on that value for the full run.
* Random speed
  * The target speed changes at a constant time interval to a new random speed. That way the user can practice to adapt to different speeds.

For each program the difficulty level determines the various target speed and time intervals. These numbers will be evaluated during testing to see in a real life scenario what numbers will make the most sense.

# How to compile and run
## User
This app can be installed on any android phone with the given APK: CoolRunning/app/release/app-release.apk
Simply copy the APK to your phone and install it.
Minimum Required Android version: 8.0 (Oreo, SDK-Version 26)
Required Permissions:
* fine location
* coarse location
* foreground service
* read external storage
* write external storage

## Developer
This project can be opened in Android Studio and compiled.

### Google Maps API-Key
The device for development and/or the key to sign  the APK need to be registered for the given API key. Non Team members will need to create a new key and replace the existing one.
Go to https://developers.google.com/maps/documentation/android-sdk/get-api-key for further information
```

```
41
42 ### Custom target speed
43 The target speeds are set to preferences of the developers. They can be adjusted to
       every individuals preferences in TargetSpeedUpdater.java
44
45 # Implementation
46 This App consists of four activities and two services. One of the services is run as
       a foreground service.
47
48 ## Cross Communication
49 For several threads, services and the user interface to communicate there is the "
       CoolRunningCom" class. This class provides static access to all the necessary
       data. This data can always be accessed and is used to asynchronous store and read
       data. To avoid reentrancy problems the data is encapsulated in static
       synchronized methods.
50
51 ## Source Files
52 ### Activities layouts
53 The layouts for the activities for this project are in "/CoolRunning/app/src/main/
       res/layout/".
54 The following files are used:
55 * activity main.xml
56   * The main activity is shown on startup and prompts the user to enter the settings
       for the run and then start the rung
57 * activity running.xml
58   * The running activity is shown while the user is running. It shows current and
       target speed. It also shows weather the user needs to slow down or speed up.
59 * activity result.xml
60   * After the user finishes running the result activity is shown with information
       about the time of the run and how good the user could stick to the target speed.
61 * activity maps.xml
62   * This activity displays a Google Maps fragment. On this map the tracked path is
       displayed
63
64 ### Activities Java
65 Each layout xml file has a corresponding java file in "CoolRunning/app/src/main/java
       /ch/arebsame/coolrunning/".
66 * MainActivity.java
67   * This file handles the interaction with the user until he starts running. It
       receives the initial settings for the run and makes sure they are valid before
       the user starts running. When the user starts running the ActivityRunning is
       started
68 * ActivityRunning.java
69   * From this activity the the services to monitor the speed are started. It
       periodically updates the user interface weather the user is too fast or too slow
       according to the data of the services. When the user is finished running it stops
       the services and starts the ActivityResult
70 * ActivityResult
71   This activity gets the values resulted from the run and displays them in the user
       interface. The user can choose to finish or display the map from the run.
72
73 ### Services
74 While the user is running two services are used to obtain new position and speed
       information, and to monitor the current speed and decide if the user is too fast
       or too slow. The java classes for the services are in "CoolRunning/app/src/main/
       java/ch/arebsame/coolrunning/"
75 * SpeedService.java
76   *  This service accesses Androids localization API using a LocationManager. It
       subscribes to position updates with the criteria to get a new position every 0.1
```

```
    seconds with no criteria on the distance to the last position.
77    * This service is started as a foreground service and will display an notification
        once started. This is necessary the avoid androids limitation on localization
      information when the app is in the background.
78  * SpeedMonitorService.java
79    * This service compares the current speed to the target speed and decides if the
      user is too fast or too slow. It plays a sound if the user is wrong.
80
81  ### other Java classes
82  The java classes used for this project are in "CoolRunning/app/src/main/java/ch/
      arebsame/coolrunning/"
83  The following files are used:
84  * CoolRunningCom.java
85    * This is a static class used for communication between various activities and
      services. This can be accessed to store or get current information about the apps
        status.
86  * GPXGenerator.java
87    * This is used by the SpeedService to store the position information in a *.gpx
      file if the user wishes to save the track
88  * SpeedMonitor.java
89    * This class is used by the SpeedMonitorService and compares the current to target
        speed and deices weather the user is too fast or too slow.
90  * TargetSpeedUpdater.java
91    * This class is used by the SpeedMonitorService. It starts a thread that will
      periodically update the target speed according to the mode setting of the user.
92
93  ### Enums
94  Several emuns are used as common values for several definitions saved in "
      CoolRunning/app/src/main/java/ch/arebsame/coolrunning/".
95  * RunningError.java
96    * This defines the states if the user is too fast, too slow or running correct.
97  * RunningMode.java
98    * This defines the possible running modes for the user
```

../README.md

2 java

2.1 Activities

2.1.1 MainActivity.java

```
1  package ch.arebsame.coolrunning;
2
3  import androidx.annotation.NonNull;
4  import androidx.appcompat.app.AppCompatActivity;
5  import androidx.core.app.ActivityCompat;
6  import androidx.core.app.NotificationCompat;
7  import androidx.core.content.ContextCompat;
8
9  import android.Manifest;
10 import android.app.Activity;
11 import android.app.Notification;
12 import android.content.Intent;
13 import android.content.pm.PackageManager;
14 import android.media.MediaPlayer;
15 import android.os.Bundle;
16
17 import android.os.Handler;
18 import android.os.Message;
19 import android.util.Log;
20 import android.view.View;
21 import android.widget.AdapterView;
22 import android.widget.Button;
23 import android.widget.CompoundButton;
24 import android.widget.EditText;
25 import android.widget.SeekBar;
26 import android.widget.Switch;
27 import android.widget.TextView;
28
29 import android.widget.ArrayAdapter;
30 import android.widget.Spinner;
31 import android.widget.Toast;
32
33 import org.w3c.dom.Text;
34
35
36 public class MainActivity extends AppCompatActivity
37 {
38     Spinner dropdown;
39     SeekBar difficultyBar;
40
41     @Override
42     protected void onCreate(Bundle savedInstanceState)
43     {
44         super.onCreate(savedInstanceState);
45         setContentView(R.layout.activity_main);
46         setTitle("CNIT355 Running App");
47
48         // check Permissions for location
49         if (ActivityCompat.checkSelfPermission(this, Manifest.permission.
       ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
50             if (ActivityCompat.shouldShowRequestPermissionRationale((Activity) this,
        Manifest.permission.ACCESS_FINE_LOCATION)) {
```

```
51
52              } else {
53                  ActivityCompat.requestPermissions((Activity) this,
54                          new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
55                          1);
56              }
57          }
58          if (ActivityCompat.checkSelfPermission(this, Manifest.permission.
    ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
59              if (ActivityCompat.shouldShowRequestPermissionRationale((Activity) this,
     Manifest.permission.ACCESS_COARSE_LOCATION)) {
60
61              } else {
62                  ActivityCompat.requestPermissions((Activity) this,
63                          new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
64                          1);
65              }
66          }
67
68          // check permission to read, write to external storage
69          if (ContextCompat.checkSelfPermission(this, Manifest.permission.
    WRITE_EXTERNAL_STORAGE) !=
70                  PackageManager.PERMISSION_GRANTED) {
71              if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.
    permission.WRITE_EXTERNAL_STORAGE)) {
72              } else {
73                  ActivityCompat.requestPermissions(this,
74                          new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},
75                          1);
76              }
77          }
78          if (ContextCompat.checkSelfPermission(this, Manifest.permission.
    READ_EXTERNAL_STORAGE) !=
79                  PackageManager.PERMISSION_GRANTED) {
80              if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.
    permission.READ_EXTERNAL_STORAGE)) {
81              } else {
82                  ActivityCompat.requestPermissions(this,
83                          new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
84                          1);
85              }
86          }
87
88          // init difficulty bar to update starting speed
89          difficultyBar = (SeekBar) findViewById(R.id.difficultlyBar);
90          difficultyBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener
    () {
91              @Override
92              public void onProgressChanged(SeekBar seekBar, int progress, boolean
    fromUser) {
93                  if (fromUser) {
94                      updateStartingSpeed(progress);
95                  }
96              }
97
98              @Override
99              public void onStartTrackingTouch(SeekBar seekBar) {
100
101             }
```

```
102
103             @Override
104             public void onStopTrackingTouch(SeekBar seekBar) {
105
106             }
107         });
108
109         // init program spinner according to available modes RunningMode enum
110         dropdown = findViewById(R.id.programSpinner);
111         RunningMode[] modes = RunningMode.values();
112         String[] modesNames = new String[modes.length];
113         for (int i = 0; i < modes.length; i++) {
114             modesNames[i] = modes[i].name().replace('_', ' ');
115         }
116         ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.
    simple_spinner_dropdown_item, modesNames);
117         dropdown.setAdapter(adapter);
118         // set up listener for selection spinner
119         dropdown.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
    {
120             @Override
121             public void onItemSelected(AdapterView<?> adapterView, View view, int i,
     long l) {
122
123                 String selectedModeString = (String) dropdown.getSelectedItem();
124                 RunningMode selectedMode = RunningMode.valueOf(selectedModeString.
    replace(' ', '_'));
125                 CoolRunningCom.setMode(selectedMode);
126                 updateStartingSpeed(difficultyBar.getProgress());
127             }
128
129             @Override
130             public void onNothingSelected(AdapterView<?> adapterView) {
131
132             }
133         });
134         updateStartingSpeed(difficultyBar.getProgress());
135
136         // init track switch to save the track
137         Switch trackSwitch = (Switch) findViewById(R.id.trackSwitch);
138         trackSwitch.setOnCheckedChangeListener(new CompoundButton.
    OnCheckedChangeListener() {
139         public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
140             if (isChecked) {
141                 CoolRunningCom.setSaveRun(true);
142             } else {
143                 CoolRunningCom.setSaveRun(false);
144             }
145         }
146     });
147     }
148
149     /**
150      * updates the starting speed on the UI
151      * @param progress the progress of the difficult bar
152      */
153     private void updateStartingSpeed(int progress) {
154         float startingSpeed = progress+1;
155         CoolRunningCom.setTargetSpeed(startingSpeed);
```

```
156         ((TextView)findViewById(R.id.speedVariable)).setText(String.format("%.02f",
    startingSpeed));
157     }
158
159     /**
160      * the user wants to start running
161      */
162     public void onStartRunningClick(View view) {
163         if (CoolRunningCom.getSaveRun()) {
164             // make sure a name is entered if the track should be saved
165             String trackName = ((TextView)findViewById(R.id.enteredNameEdit)).
    getText().toString();
166             if (trackName.isEmpty())
167             {
168                 Toast.makeText(getBaseContext(), "Please enter a valid name to safe
    the track", Toast.LENGTH_SHORT).show();
169                 return;
170             }
171             else {
172                 CoolRunningCom.setRunName(trackName);
173             }
174         }
175         // start running activity
176         Intent runningActivity = new Intent(this, ActivityRunning.class);
177         startActivity(runningActivity);
178     }
179 }
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/MainActivity.java

2.1.2 ActivityRunning.java

```
1  package ch.arebsame.coolrunning;
2
3  import androidx.annotation.NonNull;
4  import androidx.appcompat.app.AppCompatActivity;
5
6  import android.content.Intent;
7  import android.os.Bundle;
8  import android.os.Handler;
9  import android.os.Message;
10 import android.util.Log;
11 import android.view.View;
12 import android.widget.EditText;
13 import android.widget.TextView;
14
15 public class ActivityRunning extends AppCompatActivity {
16
17     Intent speedServiceIntent;
18     Intent speedMonitorServiceIntent;
19     boolean isRunning = false;
20
21     private Handler handler = new Handler() {
22
23         @Override
24         public void handleMessage(@NonNull Message msg) {
25
```

```
26              // update and display running time
27              CoolRunningCom.updateRunningTime();
28              ((TextView) findViewById(R.id.runningValue)).setText(CoolRunningCom.
    getRunningTimeFormated());
29
30              // diplay current speed and target speed
31              ((TextView) findViewById(R.id.speedValue)).setText(String.format("%.02f"
    , CoolRunningCom.getSpeed()));
32              ((TextView) findViewById(R.id.targetValue)).setText(String.format("%.02f
    ", CoolRunningCom.getTargetSpeed()));
33
34              // display weahter to speed up or slow down with color
35              TextView result = (TextView) findViewById(R.id.resultVariable);
36              if (CoolRunningCom.getRunningError() == RunningError.tooFast) {
37                  result.setText("too fast --> slow down");
38                  result.setBackgroundColor(getResources().getColor(R.color.
    runningTooFast));
39              } else if (CoolRunningCom.getRunningError() == RunningError.tooSlow) {
40                  result.setText("too slow --> speed up");
41                  result.setBackgroundColor(getResources().getColor(R.color.
    runningTooSlow));
42              } else {
43                  result.setText("correct --> keep going");
44                  result.setBackgroundColor(getResources().getColor(R.color.
    runningGood));
45              }
46          }
47      };
48
49      @Override
50      protected void onCreate(Bundle savedInstanceState) {
51          super.onCreate(savedInstanceState);
52          setContentView(R.layout.activity_running);
53
54          setTitle("CNIT355 Running App");
55
56          // start speed service with GPS API
57          speedServiceIntent = new Intent(this, SpeedService.class);
58          startService(speedServiceIntent);
59
60          // start speed monitor service to compare current speed and target speed and
     play beep accordingly
61          speedMonitorServiceIntent = new Intent(this, SpeedMonitorService.class);
62          startService(speedMonitorServiceIntent);
63
64          // start timer
65          CoolRunningCom.setStartTimeNow();
66          isRunning = true;
67
68          Thread t = new Thread() {
69              @Override
70              public void run() {
71                  while (isRunning) {
72                      // periodically update User Interface with current data
73                      handler.sendEmptyMessage(0);
74
75                      // pass some time
76                      try {
77                          Thread.sleep(100);
```

```
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
        };
        t.start();
    }

    public void onStopRunningClick(View v) {
        if (isRunning) {
            // stop all services and threads
            isRunning = false;
            stopService(speedServiceIntent);
            stopService(speedMonitorServiceIntent);
            finish();

            // show result activity
            Intent resultActivity = new Intent(this.getApplicationContext(),
    ActivityResults.class);
            startActivity(resultActivity);

        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        this.onStopRunningClick(null);
    }
}
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/ActivityRunning.java

2.1.3 ActivityResults.java

```
package ch.arebsame.coolrunning;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class ActivityResults extends AppCompatActivity
{
    TextView totalValue;
    TextView scoreValue;
    Button finishButton;

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

```
22
23          setTitle("CNIT355 Running App");
24
25          totalValue = findViewById(R.id.totalValue);
26          scoreValue = findViewById(R.id.scoreValue);
27          finishButton = findViewById(R.id.finishButton);
28
29          totalValue.setText(CoolRunningCom.getRunningTimeFormated());
30          scoreValue.setText(String.valueOf(CoolRunningCom.getScore()));
31      }
32
33      public void onFinishClick(View view)
34      {
35          // reset common data
36          CoolRunningCom.reset();
37
38          // go back to start
39          finish();
40      }
41
42      public void onShowMapClick(View view)
43      {
44          // open map
45          Intent mapActivity = new Intent(this.getApplicationContext(), MapsActivity.
        class);
46          startActivity(mapActivity);
47      }
48  }
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/ActivityResults.java

2.1.4 MapsActivity.java

```
1  package ch.arebsame.coolrunning;
2
3  import androidx.fragment.app.FragmentActivity;
4
5  import android.content.Intent;
6  import android.os.Bundle;
7  import android.view.View;
8
9  import com.google.android.gms.maps.CameraUpdateFactory;
10 import com.google.android.gms.maps.GoogleMap;
11 import com.google.android.gms.maps.OnMapReadyCallback;
12 import com.google.android.gms.maps.SupportMapFragment;
13 import com.google.android.gms.maps.model.LatLng;
14 import com.google.android.gms.maps.model.LatLngBounds;
15 import com.google.android.gms.maps.model.MarkerOptions;
16 import com.google.android.gms.maps.model.PolylineOptions;
17
18 import java.util.LinkedList;
19
20 public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
21
22     private GoogleMap mMap;
23
24     @Override
```

```
25      protected void onCreate(Bundle savedInstanceState) {
26          super.onCreate(savedInstanceState);
27          setContentView(R.layout.activity_maps);
28          // Obtain the SupportMapFragment and get notified when the map is ready to
    be used.
29          SupportMapFragment mapFragment = (SupportMapFragment)
    getSupportFragmentManager()
30                  .findFragmentById(R.id.map);
31          mapFragment.getMapAsync(this);
32      }
33
34
35      /**
36       * Manipulates the map once available.
37       * This callback is triggered when the map is ready to be used.
38       * This is where we can add markers or lines, add listeners or move the camera.
    In this case,
39       * we just add a marker near Sydney, Australia.
40       * If Google Play services is not installed on the device, the user will be
    prompted to install
41       * it inside the SupportMapFragment. This method will only be triggered once the
     user has
42       * installed Google Play services and returned to the app.
43       */
44      @Override
45      public void onMapReady(GoogleMap googleMap) {
46          mMap = googleMap;
47
48          mMap.setOnMapLoadedCallback(new GoogleMap.OnMapLoadedCallback() {
49              @Override
50              public void onMapLoaded() {
51
52                  // get al logged points
53                  LinkedList<LatLng> posList = CoolRunningCom.getPositionList();
54
55                  // make sure we have at leas 2 points to make a line
56                  if (posList != null && posList.size() >= 2) {
57                      // add line between all points
58                      mMap.addPolyline(new PolylineOptions().addAll(posList));
59
60                      // set bounds for map to be zoomed in to relevant region with
    all points in those bounds
61                      LatLngBounds bounds = new LatLngBounds(posList.getFirst(),
    posList.getFirst());
62                      for (LatLng p : posList) {
63                          bounds = bounds.including(p);
64                      }
65                      mMap.moveCamera(CameraUpdateFactory.newLatLngBounds(bounds, 200)
    );
66                  }
67              }
68          });
69      }
70
71      /**
72       * close the map and go back to result
73       */
74      public void onBackClick(View view)
75      {
```

```
76              finish();
77          }
78      }
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/MapsActivity.java

2.2 Services

2.2.1 SpeedService.java

```java
package ch.arebsame.coolrunning;

import android.Manifest;
import android.app.Activity;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Criteria;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Build;
import android.os.Bundle;
import android.os.IBinder;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

import androidx.core.app.ActivityCompat;
import androidx.core.app.NotificationCompat;

/**
 * This service uses the Location API to get the speed from GPS
 * This is run as a foreground service to ensure location updates are also provided
    if the app is
 * in background
 */
public class SpeedService extends Service {

    float speed = 0;
    protected LocationManager locationManager;
    protected LocationListener listener;
    // The minimum distance to change Updates in meters
    private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 0;
    // The minimum time between updates in milliseconds
    private static final long MIN_TIME_BW_UPDATES = 100;

    private GPXGenerator gpxGenerator;

    public SpeedService() {
        if (CoolRunningCom.getSaveRun()) {
            this.gpxGenerator = new GPXGenerator(CoolRunningCom.getRunName());
        }
    }

    @Override
    public void onCreate() {

        /*
        Set this service as a foreground service and show a notification
```

```
        References on foreground service:
        https://androidwave.com/foreground-service-android-example
        https://developer.android.com/about/versions/oreo/background-location-limits
         */

        createNotificationChannel();
        Intent notificationIntent = new Intent(this, MainActivity.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(this,
                0, notificationIntent, 0);
        Notification notification = new NotificationCompat.Builder(this, "Speed
    Service")
                .setContentTitle("Speed Service")
                .setContentIntent(pendingIntent)
                .build();

        startForeground(1, notification);

        // init location manager for GPS
        locationManager = (LocationManager) this.getSystemService(LOCATION_SERVICE);
        // Retrieve a list of location providers that have fine accuracy, no
    monetary cost, etc
        Criteria criteria = new Criteria();
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        criteria.setSpeedAccuracy(Criteria.ACCURACY_HIGH);
        criteria.setSpeedRequired(true);
        criteria.setCostAllowed(false);

        String providerName = locationManager.getBestProvider(criteria, true);

        final boolean gpsEnabled = locationManager.isProviderEnabled(LocationManager
    .GPS_PROVIDER);

        if (gpsEnabled && providerName != null) {
            CoolRunningCom.initPosList();

            //create location listener and request location updates
            listener = new LocationListener() {
                @Override
                public void onLocationChanged(Location location) {
                    if (location != null && location.hasSpeed()) {
                        speed = location.getSpeed();
                        CoolRunningCom.setSpeed(speed);
                        CoolRunningCom.addPosition(location.getLatitude(), location.
    getLongitude());
                        if (CoolRunningCom.getSaveRun() && gpxGenerator != null) {
                            gpxGenerator.addPoint(location);
                        }
                    }
                    else {
                        Log.w("location" ,"no speed in location");
                    }
                }

                @Override
                public void onStatusChanged(String provider, int status, Bundle
    extras) {

                }

```

```
109                 @Override
110                 public void onProviderEnabled(String provider) {
111
112                 }
113
114                 @Override
115                 public void onProviderDisabled(String provider) {
116
117                 }
118             };
119
120             locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
    MIN_TIME_BW_UPDATES, MIN_DISTANCE_CHANGE_FOR_UPDATES, listener);
121         }
122         else {
123             Toast.makeText(getBaseContext() ,"GPS is not enabled, please enable and
    try again", Toast.LENGTH_LONG).show();
124         }
125     }
126
127     private void createNotificationChannel() {
128         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
129             NotificationChannel serviceChannel = new NotificationChannel(
130                     "Speed Service",
131                     "Speed Service",
132                     NotificationManager.IMPORTANCE_DEFAULT
133             );
134
135             NotificationManager manager = getSystemService(NotificationManager.class
    );
136             manager.createNotificationChannel(serviceChannel);
137         }
138     }
139
140     @Override
141     public void onDestroy() {
142         super.onDestroy();
143         locationManager.removeUpdates(listener);
144         if (CoolRunningCom.getSaveRun() && gpxGenerator != null) {
145             gpxGenerator.writeToFile();
146         }
147     }
148
149     @Override
150     public IBinder onBind(Intent intent) {
151         // TODO: Return the communication channel to the service.
152         throw new UnsupportedOperationException("Not yet implemented");
153     }
154 }
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/SpeedService.java

2.2.2 SpeedMonitorService.java

```
1 package ch.arebsame.coolrunning;
2
3 import android.app.Service;
```

```
4  import android.content.Intent;
5  import android.media.MediaPlayer;
6  import android.os.IBinder;
7
8  /**
9   * This service keeps track of weather the user is wrong or correct
10  * It will play a beep if the user is wrong and set the Error in CoolRunningCom
        accordingly
11  */
12 public class SpeedMonitorService extends Service {
13      private TargetSpeedUpdater targetSpeedUpdater;
14      private SpeedMonitor monitor;
15      private Thread speedMonitorThread;
16
17      private MediaPlayer slowDownMusic;
18      private MediaPlayer speedUpMusic;
19
20      private boolean isRunning = false;
21
22      public SpeedMonitorService() {
23      }
24
25      @Override
26      public void onCreate() {
27          super.onCreate();
28          this.targetSpeedUpdater = new TargetSpeedUpdater();
29          this.targetSpeedUpdater.Start();
30          this.monitor = new SpeedMonitor();
31
32          this.slowDownMusic = MediaPlayer.create(this.getBaseContext(), R.raw.
        slow_down);
33          this.slowDownMusic.setLooping(false);
34          this.speedUpMusic = MediaPlayer.create(this.getBaseContext(), R.raw.speed_up
        );
35          this.speedUpMusic.setLooping(false);
36
37          this.isRunning = true;
38
39          this.speedMonitorThread = new Thread() {
40              @Override
41              public void run() {
42
43                  while (isRunning) {
44                      while (speedUpMusic.isPlaying() || slowDownMusic.isPlaying()) {
45                          // wait to make sure nothing is playing
46                      }
47
48                      //monitor.compareSpeed(CoolRunningCom.getAverageSpeed(),
        CoolRunningCom.getTargetSpeed());
49                      monitor.compareSpeed(CoolRunningCom.getSpeed(), CoolRunningCom.
        getTargetSpeed());
50
51                      if (monitor.getRunningError() == RunningError.tooSlow) {
52                          speedUpMusic.start();
53                          CoolRunningCom.setRunningError(RunningError.tooSlow);
54                      } else if (monitor.getRunningError() == RunningError.tooFast) {
55                          slowDownMusic.start();
56                          CoolRunningCom.setRunningError(RunningError.tooFast);
57                      } else {
```

```
                            CoolRunningCom.setRunningError(RunningError.correct);
                        }

                        // pass some time
                        try {
                            Thread.sleep(3000);
                        } catch (InterruptedException e) {
                            e.printStackTrace();
                        }
                    }
                }
            };
            speedMonitorThread.start();
        }

        private void monitorSpeed() {

        }

        @Override
        public void onDestroy() {
            super.onDestroy();
            this.isRunning = false;
            this.targetSpeedUpdater.Stop();
            CoolRunningCom.setScore(monitor.getCurrentScore());
        }

        @Override
        public IBinder onBind(Intent intent) {
            // TODO: Return the communication channel to the service.
            throw new UnsupportedOperationException("Not yet implemented");
        }
}
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/SpeedMonitorService.java

2.3 other classes

2.3.1 CoolRunningCom.java

```java
package ch.arebsame.coolrunning;

import com.google.android.gms.maps.model.LatLng;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.Duration;
import java.time.Instant;
import java.util.Date;
import java.util.LinkedList;
import java.util.List;

/**
 * shared data across the application
 * to communicate
 */
public class CoolRunningCom {
    private static float speed;
    private static float averageSpeed;
    private static final int speedDelayLineLength = 30;
    private static int speedDelayLineIndex = 0;
    private static float speedDelayLine[] = new float[speedDelayLineLength];
    private static float targetSpeed;
    private static RunningMode mode;
    private static State state;
    private static RunningError runningError;
    private static float score;
    private static Instant startTime;
    private static Duration runningTime;
    final static DateFormat timeFormat = new SimpleDateFormat("mm:ss");
    private static LinkedList<LatLng> positionList;
    private static String runName = "CoolRunningTrack";
    private static Boolean saveRun = false;

    /**
     * resets CoolRunningCom to initial state
     */
    public static void reset() {
        setSaveRun(false);
        if (positionList != null) {
            positionList.clear();
        }
        setRunName("CoolRunningTrack");
        setScore(0);
    }

    public static Boolean getSaveRun() {
        return saveRun;
    }

    public static void setSaveRun(Boolean saveRun) {
        CoolRunningCom.saveRun = saveRun;
    }

    public static String getRunName() {
```

```
56          return runName;
57      }
58
59      public static void setRunName(String runName) {
60          CoolRunningCom.runName = runName;
61      }
62
63      public static RunningError getRunningError() {
64          return runningError;
65      }
66
67      public synchronized static void setRunningError(RunningError runningError) {
68          CoolRunningCom.runningError = runningError;
69      }
70
71      public static float getAverageSpeed() {
72          return averageSpeed;
73      }
74
75      public static float getSpeed() {
76          return speed;
77      }
78
79      public synchronized static void setSpeed(float speed) {
80          CoolRunningCom.speed = speed;
81          speedDelayLine[speedDelayLineIndex] = speed;
82          speedDelayLineIndex++;
83          if (speedDelayLineIndex >= speedDelayLineLength) {
84              speedDelayLineIndex = 0;
85          }
86          float sum = 0;
87          for (float s : speedDelayLine) {
88              sum += s;
89          }
90          averageSpeed = sum / speedDelayLineLength;
91      }
92
93      public static float getTargetSpeed() {
94          return targetSpeed;
95      }
96
97      public synchronized static void setTargetSpeed(float targetSpeed) {
98          CoolRunningCom.targetSpeed = targetSpeed;
99      }
100
101     public static RunningMode getMode() {
102         return mode;
103     }
104
105     public synchronized static void setMode(RunningMode mode) {
106         CoolRunningCom.mode = mode;
107     }
108
109     public static State getState() {
110         return state;
111     }
112
113     public synchronized static void setState(State state) {
114         CoolRunningCom.state = state;
```

```
115     }
116
117     public static float getScore() {
118         return score;
119     }
120
121     public synchronized static void setScore(float score) {
122         if (score < 0) {
123             CoolRunningCom.score = 0;
124         }
125         else if (score > 100) {
126             CoolRunningCom.score = 100;
127         }
128         else {
129             CoolRunningCom.score = score;
130         }
131     }
132
133     public synchronized static void setStartTimeNow() {
134         CoolRunningCom.startTime = Instant.now();
135     }
136
137     public synchronized static Duration updateRunningTime() {
138         if (CoolRunningCom.startTime == null) {
139             CoolRunningCom.setStartTimeNow();
140         }
141         CoolRunningCom.runningTime = Duration.between(CoolRunningCom.startTime,
    Instant.now());
142         return CoolRunningCom.runningTime;
143     }
144
145     public static Duration getRunningTime() {
146         if (CoolRunningCom.runningTime != null) {
147             return CoolRunningCom.runningTime;
148         }
149         return Duration.ZERO;
150     }
151
152     public static long getRunningTimeMillis() {
153         if (CoolRunningCom.runningTime != null) {
154             return CoolRunningCom.runningTime.toMillis();
155         }
156         return 0;
157     }
158
159     public static String getRunningTimeFormated() {
160         long currentMillis = CoolRunningCom.getRunningTimeMillis();
161         Date d = new Date(currentMillis);
162         String timeString = timeFormat.format(d);
163         return timeString;
164     }
165
166     public synchronized static void initPosList() {
167         CoolRunningCom.positionList = new LinkedList<LatLng>();
168     }
169
170     public synchronized static void addPosition(LatLng pos) {
171         if (CoolRunningCom.positionList != null && pos != null) {
172             CoolRunningCom.positionList.add(pos);
```

```
173         }
174     }
175
176     public synchronized static void addPosition(double lat, double lng) {
177         CoolRunningCom.addPosition(new LatLng(lat, lng));
178     }
179
180     public static LinkedList<LatLng> getPositionList() {
181         return CoolRunningCom.positionList;
182     }
183 }
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/CoolRunningCom.java

2.3.2 GPXGenerator.java

```
1  package ch.arebsame.coolrunning;
2
3  import android.location.Location;
4  import android.os.Environment;
5  import android.provider.ContactsContract;
6  import android.util.Log;
7
8  import java.io.File;
9  import java.io.FileWriter;
10 import java.io.IOException;
11 import java.text.DateFormat;
12 import java.text.SimpleDateFormat;
13 import java.util.Date;
14
15 /**
16  * This class receives poins as location objects and stores them to a file
17  * in GPX format.
18  * Reference: https://stackoverflow.com/questions/46490192/how-to-export-a-gpx-file-
       from-a-latlng-arraylist
19  */
20 public class GPXGenerator {
21
22     String fileName;
23     String header;
24     String nameTag;
25     String footer;
26     DateFormat df;
27     String segments;
28     String path;
29
30     private boolean dir_exists(String dir_path)
31     {
32         boolean ret = false;
33         File dir = new File(dir_path);
34         if(dir.exists() && dir.isDirectory())
35             ret = true;
36         return ret;
37     }
38
39     /**
40      * Initialized the GPX generator
```

```
41      * GPX file will be saved in "external storage directory"/CoolRunning/"name".gpx
42      * if the directory /CoolRunning does not exist it will be created
43      * @param name name for the file to save the gpx file
44      */
45     public GPXGenerator(String name) {
46         this.fileName = name + ".gpx";
47         this.path = Environment.getExternalStorageDirectory().getAbsolutePath() + "/
    CoolRunning";
48 
49         if (!dir_exists(this.path)){
50             File directory = new File(this.path);
51             directory.mkdirs();
52         }
53 
54         // initialize xml tags for GPX
55         this.header = "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"
    ?><gpx xmlns=\"http://www.topografix.com/GPX/1/1\" creator=\"MapSource 6.15.5\"
    version=\"1.1\" xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\"  xsi:
    schemaLocation=\"http://www.topografix.com/GPX/1/1 http://www.topografix.com/GPX
    /1/1/gpx.xsd\"><trk>\n";
56         this.nameTag = "<name>" + name + "</name><trkseg>\n";
57 
58         this.segments = "";
59         this.df = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssZ");
60 
61         this.footer = "</trkseg></trk></gpx>";
62 
63     }
64 
65     /**
66      * adds a new point to the gpx segments
67      * @param location the new point to add (latitude, longitude and time will be
    saved)
68      */
69     public void addPoint(Location location) {
70         // add new gpx xml segment for this point
71         segments += "<trkpt lat=\"" + location.getLatitude() + "\" lon=\"" +
    location.getLongitude() + "\"><time>" + df.format(new Date(location.getTime())) +
     "</time></trkpt>\n";
72     }
73 
74     /**
75      * saves all previously given points to a file with the name given in the
    constructor
76      */
77     public void writeToFile() {
78         try {
79             File file = new File(path + "/" + fileName);
80 
81             FileWriter writer = new FileWriter(file, false);
82             writer.append(header);
83             writer.append(nameTag);
84             writer.append(segments);
85             writer.append(footer);
86             writer.flush();
87             writer.close();
88 
89         } catch (IOException e) {
90             Log.e("generateGPX", "Error Writing GPX file",e);
```

```
        }
    }

}
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/GPXGenerator.java

2.3.3 SpeedMonitor.java

```
package ch.arebsame.coolrunning;

/**
 * this class handles the comparisons of the current speed and the target speed
 * It keeps count of how often the user was not correct and calculates a score at
    the end
 */
public class SpeedMonitor
{
    // currents speed needs to be in targetSpeed +- toleranceTreashold
    private float toleranceTreashold = 1;

    float error;
    private RunningError runningError = RunningError.correct;

    private long totalComparisons = 0;
    private long badComparisons = 0;


    public SpeedMonitor()
    {
        error = 0.0f;
    }

    /**
     * Compares the currentSpeed and target Speed
     * @param currentSpeed the current GPS speed
     * @param targetSpeed the current target speed
     */
    public void compareSpeed(float currentSpeed, float targetSpeed)
    {
        this.totalComparisons++;
        error = currentSpeed - targetSpeed;
        if (error <= -toleranceTreashold) {
            this.runningError = RunningError.tooSlow;
            this.badComparisons++;
        }
        else if (error >= toleranceTreashold) {
            this.runningError = RunningError.tooFast;
            this.badComparisons++;
        }
        else {
            this.runningError = RunningError.correct;
        }
    }

    /**
     * @return how much the user is off of the target speed
```

```
48          */
49         public float getError()
50         {
51             return error;
52         }
53 
54         /**
55          * @return the error if too fast, too slow or correct
56          */
57         public RunningError getRunningError() {
58             return this.runningError;
59         }
60 
61         /**
62          * @return the current score | 0 = all bad, 100 = perfect run
63          */
64         public float getCurrentScore() {
65             if (totalComparisons > 0) {
66                 return 100* (totalComparisons - badComparisons) / totalComparisons;
67             }
68 
69             return 0;
70         }
71 }
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/SpeedMonitor.java

2.3.4 TargetSpeedUpdater.java

```
1  package ch.arebsame.coolrunning;
2 
3  /**
4   * This class updates the target speed periodically according to the running Mode
5   */
6  public class TargetSpeedUpdater {
7 
8      private boolean isRunning = false;
9      private float startSpeed;
10     private RunningMode mode;
11 
12     private Boolean intervalIsFast = true;
13 
14     private Thread updaterThread;
15 
16     public TargetSpeedUpdater()  {
17 
18         this.updaterThread = new Thread() {
19             @Override
20             public void run() {
21                 while (isRunning) {
22                     float nextTargetSpeed = calcNextTargetSpeed();
23                     long nextInterval = calcNextInterval();
24                     CoolRunningCom.setTargetSpeed(nextTargetSpeed);
25                     // pass some time
26                     try {
27                         Thread.sleep(nextInterval*1000);
28                     } catch (InterruptedException e) {
```

```
29                          e.printStackTrace();
30                      }
31                  }
32              }
33          };
34      }
35
36      /**
37       * calculates the next target speed according to the running mode
38       * @return the next target speed
39       */
40      private float calcNextTargetSpeed() {
41
42          float nextSpeed = 0; // next Target speed in m/s
43
44          switch (CoolRunningCom.getMode()) {
45              case Interval:
46                  float intervalFast = 6;
47                  float intervalSlow = 3;
48                  if (intervalIsFast) {
49                      nextSpeed = intervalSlow;
50                      intervalIsFast = false;
51                  }
52                  else {
53                      nextSpeed = intervalFast;
54                      intervalIsFast = true;
55                  }
56                  break;
57              case Random_Speed:
58                  nextSpeed = (float) (Math.random() * 5) + 3;
59                  break;
60              case Constant_Speed:
61                  nextSpeed = CoolRunningCom.getTargetSpeed();
62                  break;
63              case Increasing_Speed:
64                  nextSpeed = CoolRunningCom.getTargetSpeed() + (float) 0.1;
65                  break;
66          }
67
68          return nextSpeed;
69      }
70
71      /**
72       * Calculates the time until the next update of the target speed
73       * @return a time in seconds to wait until the next update
74       */
75      private long calcNextInterval() {
76          long nextInterval = 1; // time to next target speed change in sec
77
78          switch (CoolRunningCom.getMode()) {
79              case Interval:
80                  if (intervalIsFast) nextInterval = 45;
81                  else nextInterval = 15;
82                  break;
83              case Random_Speed:
84                  nextInterval = Math.round((Math.random() * 10) + 7);
85                  break;
86              case Constant_Speed:
87                  nextInterval = 10;
```

```
88                  break;
89              case Increasing_Speed:
90                  nextInterval = 10;
91                  break;
92          }
93
94          return nextInterval;
95      }
96
97      /**
98       * starts a thread to update the target speed
99       */
100     public void Start() {
101         this.isRunning = true;
102         this.updaterThread.start();
103     }
104
105     /**
106      * stops updating the target speed
107      */
108     public void Stop() {
109         this.isRunning = false;
110     }
111 }
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/TargetSpeedUpdater.java

2.4 Enums

2.4.1 RunningError.java

```java
package ch.arebsame.coolrunning;

/**
 * the possible errors while the user is running
 */
public enum RunningError {
    correct,
    tooSlow,
    tooFast,
}
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/RunningError.java

2.4.2 RunningMode.java

```java
package ch.arebsame.coolrunning;

/**
 * Current Running mode of the app
 */
public enum RunningMode {
    Interval,
    Increasing_Speed,
    Constant_Speed,
    Random_Speed
}
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/RunningMode.java

2.4.3 State.java

```java
package ch.arebsame.coolrunning;

/**
 * The state of the app
 * each state has an individual User Interface
 */
public enum State {
    Start,
    Running,
    Finished,
}
```

../CoolRunning/app/src/main/java/ch/arebsame/coolrunning/State.java

3 AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ch.arebsame.coolrunning">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <!--
             The API key for Google Maps-based APIs is defined as a string resource.
             (See the file "res/values/google_maps_api.xml").
             Note that the API key is linked to the encryption key used to sign the
    APK.
             You need a different API key for each encryption key, including the
    release key that is used to
             sign the APK for publishing.
             You can define the keys for the debug and release targets in src/debug/
     and src/release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps" />

        <service
            android:name=".SpeedMonitorService"
            android:enabled="true"
            android:exported="true" />

        <activity android:name=".ActivityResults" />
        <activity android:name=".ActivityRunning" />

        <service
            android:name=".SpeedService"
            android:enabled="true"
            android:exported="true" />

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
```

```
        </activity>
    </application>

</manifest>
```

../CoolRunning/app/src/main/AndroidManifest.xml

4 Gradle

4.1 build.gradle CoolRunning

```
// Top-level build file where you can add configuration options common to all sub-
    projects/modules.

buildscript {
    repositories {
        google()
        jcenter()

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.2'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()

    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

../CoolRunning/build.gradle

4.2 build.gradle app

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "ch.arebsame.coolrunning"
        minSdkVersion 26
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
```

```
12          testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
13      }
14      buildTypes {
15          release {
16              minifyEnabled false
17              proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), '
    proguard-rules.pro'
18          }
19      }
20  }
21
22  dependencies {
23      implementation fileTree(dir: 'libs', include: ['*.jar'])
24      implementation 'androidx.appcompat:appcompat:1.0.2'
25      implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
26      implementation 'com.google.android.gms:play-services-maps:16.1.0'
27      testImplementation 'junit:junit:4.12'
28      androidTestImplementation 'androidx.test.ext:junit:1.1.0'
29      androidTestImplementation 'androidx.test.espresso:espresso-core:3.1.1'
30  }
```

../CoolRunning/app/build.gradle

5 res

5.1 layout

5.1.1 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.
    android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:screenOrientation="portrait"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="15sp"
        android:layout_marginRight="15sp"
        android:layout_marginTop="25sp"
        android:layout_marginBottom="25sp"
        android:orientation="vertical">

        <TextView
            android:id="@+id/programText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="15sp"
            android:text="@string/program_title"
            android:textColor="@color/colorPrimary"
            android:textSize="22sp" />

        <Spinner
            android:id="@+id/programSpinner"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="40sp"
            android:spinnerMode="dropdown"
            android:textSize="20sp" />

        <TextView
            android:id="@+id/difficultyText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="40sp"
            android:text="@string/difficulty_title"
            android:textColor="@color/colorPrimary"
            android:textSize="22sp" />

        <SeekBar
            android:id="@+id/difficultlyBar"
            style="@style/Widget.AppCompat.SeekBar.Discrete"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="50sp"
            android:max="10"
            android:progress="3"
```

```
52              android:progressTint="#0012E3EB" />
53
54          <LinearLayout
55              android:layout_width="match_parent"
56              android:layout_height="wrap_content"
57              android:orientation="horizontal"
58              android:layout_marginBottom="40sp">
59
60              <TextView
61                  android:id="@+id/speedText"
62                  android:layout_width="wrap_content"
63                  android:layout_height="wrap_content"
64                  android:layout_weight="1"
65                  android:text="@string/starting_speed_title"
66                  android:textColor="@color/colorPrimary"
67                  android:textSize="22sp" />
68
69              <TextView
70                  android:id="@+id/speedVariable"
71                  android:layout_width="wrap_content"
72                  android:layout_height="wrap_content"
73                  android:layout_weight="1"
74                  android:text="@string/speed_hint"
75                  android:textSize="22sp" />
76
77          </LinearLayout>
78
79          <Switch
80              android:id="@+id/trackSwitch"
81              android:layout_width="wrap_content"
82              android:layout_height="wrap_content"
83              android:layout_marginBottom="40sp"
84              android:text="@string/save_track_title"
85              android:textSize="22sp" />
86
87          <LinearLayout
88              android:layout_width="match_parent"
89              android:layout_height="wrap_content"
90              android:orientation="horizontal"
91              android:layout_marginBottom="40sp">
92
93              <TextView
94                  android:id="@+id/trackNameText"
95                  android:layout_width="wrap_content"
96                  android:layout_height="wrap_content"
97                  android:layout_weight="1"
98                  android:text="@string/track_name_title"
99                  android:textColor="@color/colorPrimary"
100                 android:textSize="18sp" />
101
102             <EditText
103                 android:id="@+id/enteredNameEdit"
104                 android:layout_width="wrap_content"
105                 android:layout_height="wrap_content"
106                 android:layout_weight="1"
107                 android:ems="10"
108                 android:hint="@string/track_name_hint"
109                 android:inputType="textPersonName"
110                 android:textSize="18sp"
```

```
                android:importantForAutofill="no" />

        </LinearLayout>

        <Button
            android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:onClick="onStartRunningClick"
            android:text="@string/start_running_btn"
            android:textSize="22sp" />

    </LinearLayout>

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

../CoolRunning/app/src/main/res/layout/activity_main.xml

5.1.2 activity_maps.xml

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

    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="10"
        android:screenOrientation="portrait" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:onClick="onBackClick"
        android:text="@string/back_btn" />

</LinearLayout>
```

../CoolRunning/app/src/main/res/layout/activity_maps.xml

5.1.3 activity_running.xml

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

```
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:screenOrientation="portrait"
8       android:layout_marginLeft="15sp"
9       android:layout_marginRight="15sp"
10      android:layout_marginTop="25sp"
11      android:layout_marginBottom="25sp"
12      tools:context=".ActivityRunning">
13
14      <TextView
15          android:id="@+id/runningText"
16          android:layout_width="wrap_content"
17          android:layout_height="wrap_content"
18          android:text="@string/running_time_title"
19          android:textSize="18sp"
20          app:layout_constraintStart_toStartOf="parent"
21          app:layout_constraintTop_toTopOf="parent" />
22
23      <Button
24          android:id="@+id/stopButton"
25          android:layout_width="wrap_content"
26          android:layout_height="wrap_content"
27          android:layout_gravity="center"
28          android:layout_marginTop="44dp"
29          android:textSize="22sp"
30          android:onClick="onStopRunningClick"
31          android:text="@string/stop_running_btn"
32          app:layout_constraintEnd_toEndOf="parent"
33          app:layout_constraintHorizontal_bias="0.50"
34          app:layout_constraintStart_toStartOf="parent"
35          app:layout_constraintTop_toBottomOf="@+id/resultVariable" />
36
37      <TextView
38          android:id="@+id/targetText"
39          android:layout_width="match_parent"
40          android:layout_height="wrap_content"
41          android:layout_marginTop="32dp"
42          android:text="@string/target_speed_title"
43          android:textSize="18sp"
44          app:layout_constraintTop_toBottomOf="@+id/speedValue"
45          tools:layout_editor_absoluteX="0dp" />
46
47      <TextView
48          android:id="@+id/targetValue"
49          android:layout_width="match_parent"
50          android:layout_height="wrap_content"
51          android:background="@drawable/my_border"
52          android:padding="10sp"
53          android:text="target speed value"
54          android:textColor="@color/colorPrimary"
55          android:textSize="32sp"
56          app:layout_constraintTop_toBottomOf="@+id/targetText"
57          tools:layout_editor_absoluteX="-16dp" />
58
59      <TextView
60          android:id="@+id/runningValue"
61          android:layout_width="match_parent"
62          android:layout_height="wrap_content"
63          android:background="@drawable/my_border"
```

```
64          android:padding="10sp"
65          android:text="run time value"
66          android:textColor="@color/colorPrimary"
67          android:textSize="32sp"
68          app:layout_constraintTop_toBottomOf="@+id/runningText"
69          tools:layout_editor_absoluteX="15dp" />
70
71      <TextView
72          android:id="@+id/speedValue"
73          android:layout_width="match_parent"
74          android:layout_height="wrap_content"
75          android:background="@drawable/my_border"
76          android:padding="10sp"
77          android:text="current speed value"
78          android:textColor="@color/colorPrimary"
79          android:textSize="32sp"
80          app:layout_constraintStart_toStartOf="parent"
81          app:layout_constraintTop_toBottomOf="@+id/speedText" />
82
83      <TextView
84          android:id="@+id/speedText"
85          android:layout_width="match_parent"
86          android:layout_height="wrap_content"
87          android:layout_marginTop="32dp"
88          android:text="@string/current_speed_title"
89          android:textSize="18sp"
90          app:layout_constraintStart_toStartOf="parent"
91          app:layout_constraintTop_toBottomOf="@+id/runningValue" />
92
93      <TextView
94          android:id="@+id/resultVariable"
95          android:layout_width="match_parent"
96          android:layout_height="wrap_content"
97          android:layout_marginTop="64dp"
98          android:gravity="center"
99          android:padding="10sp"
100         android:text="Result"
101         android:textColor="@color/colorPrimary"
102         android:textSize="24sp"
103         app:layout_constraintTop_toBottomOf="@+id/targetValue"
104         tools:layout_editor_absoluteX="0dp" />
105
106 </androidx.constraintlayout.widget.ConstraintLayout>
```

../CoolRunning/app/src/main/res/layout/activity_running.xml

5.1.4 activity_results.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:layout_weight="2"
7     android:screenOrientation="portrait"
8     android:orientation="vertical"
9     tools:context=".ActivityResults">
```

```
10
11      <TableLayout
12          android:id="@+id/resultTable"
13          android:layout_width="match_parent"
14          android:layout_height="match_parent"
15          android:layout_weight="1"
16          android:orientation="vertical"
17          tools:layout_editor_absoluteX="1dp"
18          tools:layout_editor_absoluteY="1dp">
19
20          <TableRow
21              android:layout_width="fill_parent"
22              android:layout_height="wrap_content"
23              android:layout_weight="1">
24
25              <TextView
26                  android:id="@+id/totalRunTime"
27                  android:layout_width="match_parent"
28                  android:layout_height="match_parent"
29                  android:layout_weight="1"
30                  android:background="@drawable/my_border"
31                  android:gravity="center"
32                  android:text="@string/running_time_title"
33                  android:textAlignment="center"
34                  android:textSize="20sp"
35                  android:textStyle="bold" />
36
37              <TextView
38                  android:id="@+id/scoreText"
39                  android:layout_width="match_parent"
40                  android:layout_height="match_parent"
41                  android:layout_weight="1"
42                  android:text="@string/speed_score_title"
43                  android:textAlignment="center"
44                  android:textSize="20sp"
45                  android:textStyle="bold"
46                  android:gravity="center"
47                  android:background="@drawable/my_border"/>
48          </TableRow>
49
50          <TableRow
51              android:layout_width="fill_parent"
52              android:layout_height="wrap_content"
53              android:layout_weight="1">
54
55              <TextView
56                  android:id="@+id/totalValue"
57                  android:layout_width="match_parent"
58                  android:layout_height="match_parent"
59                  android:layout_weight="1"
60                  android:background="@drawable/my_border"
61                  android:gravity="center"
62                  android:textAlignment="center"
63                  android:textSize="32sp"
64                  android:textStyle="bold" />
65
66              <TextView
67                  android:id="@+id/scoreValue"
68                  android:layout_width="match_parent"
```

```
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:background="@drawable/my_border"
                android:gravity="center"
                android:textAlignment="center"
                android:textSize="32sp"
                android:textStyle="bold" />

        </TableRow>

    </TableLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:gravity="center"
        android:orientation="horizontal">

        <Button
            android:id="@+id/finishButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_gravity="center"
            android:onClick="onFinishClick"
            android:text="@string/finish_btn"
            android:textSize="18sp"
            android:textAlignment="center"/>

        <Button
            android:id="@+id/mapButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:layout_gravity="center"
            android:onClick="onShowMapClick"
            android:text="@string/show_map_btn"
            android:textSize="18sp"
            android:textAlignment="center"/>
    </LinearLayout>

</LinearLayout>
```

../CoolRunning/app/src/main/res/layout/activity_results.xml

5.2 values

5.2.1 colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#000000</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#D81B60</color>
    <color name="runningGood">#216D06</color>
    <color name="runningTooFast">#4596DD</color>
    <color name="runningTooSlow">#DF0000</color>
</resources>
```

../CoolRunning/app/src/main/res/values/colors.xml

5.2.2 strings.xml

```
<resources>
    <string name="app_name">CoolRunning</string>
    <string name="title_activity_maps">Your track</string>
    <string name="program_title">Select Program</string>
    <string name="difficulty_title">Difficulty</string>
    <string name="starting_speed_title">Starting Speed:</string>
    <string name="track_name_title">Track Name:</string>
    <string name="start_running_btn">Start Running!</string>
    <string name="save_track_title">Save Track</string>
    <string name="track_name_hint">Enter Name</string>
    <string name="speed_hint">speed m/s</string>
    <string name="running_time_title">Total Running Time</string>
    <string name="speed_score_title">Speed Score</string>
    <string name="finish_btn">Finish</string>
    <string name="show_map_btn">Show Map</string>
    <string name="stop_running_btn">Stop Running</string>
    <string name="target_speed_title">Target Speed</string>
    <string name="current_speed_title">Current Speed</string>
    <string name="back_btn">Back</string>
</resources>
```

../CoolRunning/app/src/main/res/values/strings.xml

5.2.3 styles.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

../CoolRunning/app/src/main/res/values/styles.xml

6 Appendix

6.1 Readme formatted

CoolRunning

This readme describes the Term Project for CNIT Fall-2019-CNIT-35500-001 at Purdue Polytechnic.

License

This project is open source and available under MIT License at: https://github.com/Andi-Sail/CoolRunning

Brief Description

Cool Running is a speed tracking mobile application for android devices that tracks the speed of the user while either walking, running, or riding on a vehicle. If the runner's speed is out of the selected range of the speed, then the program will send the alarm to the runner indicating the status.

Running Modes

The following four running programs will be implemented:

- Interval
  - The target speed alternates between fast and slow at a constant time interval.
- Increasing speed
  - The target speed begins at a start value and always increase after a time interval. The goal for the user is to keep up as long as possible.
- Constant speed
  - The target speed is set to a constant value and stays on that value for the full run.
- Random speed
  - The target speed changes at a constant time interval to a new random speed. That way the user can practice to adapt to different speeds.

For each program the difficulty level determines the various target speed and time intervals. These numbers will be evaluated during testing to see in a real life scenario what numbers will make the most sense.

How to compile and run

User

This app can be installed on any android phone with the given APK: CoolRunning/app/release/app-release.apk Simply copy the APK to your phone and install it. Minimum Required Android version: 8.0 (Oreo, SDK-Version 26) Required Permissions:

- fine location
- coarse location
- foreground service
- read external storage
- write external storage

Developer

This project can be opened in Android Studio and compiled.

Google Maps API-Key

The device for development and/or the key to sign the APK need to be registered for the given API key. Non Team members will need to create a new key and replace the existing one. Go to https://developers.google.com/maps/documentation/android-sdk/get-api-key for further information

Custom target speed

The target speeds are set to preferences of the developers. They can be adjusted to every individuals preferences in TargetSpeedUpdater.java

Implementation

This App consists of four activities and two services. One of the services is run as a foreground service.

Cross Communication

For several threads, services and the user interface to communicate there is the "CoolRunningCom" class. This class provides static access to all the necessary data. This data can always be accessed and is used to asynchronous store and read data. To avoid reentrancy problems the data is encapsulated in static synchronized methods.

Source Files

Activities layouts

The layouts for the activities for this project are in "/CoolRunning/app/src/main/res/layout/". The following files are used:

- activity main.xml
    - The main activity is shown on startup and prompts the user to enter the settings for the run and then start the rung
- activity running.xml
    - The running activity is shown while the user is running. It shows current and target speed. It also shows weather the user needs to slow down or speed up.
- activity result.xml
    - After the user finishes running the result activity is shown with information about the time of the run and how good the user could stick to the target speed.
- activity maps.xml
    - This activity displays a Google Maps fragment. On this map the tracked path is displayed

Activities Java

Each layout xml file has a corresponding java file in "CoolRunning/app/src/main/java/ch/arebsame/coolrunning/".

- MainActivity.java

    - This file handles the interaction with the user until he starts running. It receives the initial settings for the run and makes sure they are valid before the user starts running. When the user starts running the ActivityRunning is started
- ActivityRunning.java
    - From this activity the the services to monitor the speed are started. It periodically updates the user interface weather the user is too fast or too slow according to the data of the services. When the user is finished running it stops the services and starts the ActivityResult
- ActivityResult This activity gets the values resulted from the run and displays them in the user interface. The user can choose to finish or display the map from the run.

Services

While the user is running two services are used to obtain new position and speed information, and to monitor the current speed and decide if the user is too fast or too slow. The java classes for the services are in "CoolRunning/app/src/main/java/ch/arebsame/coolrunning/"

- SpeedService.java
    - This service accesses Androids localization API using a LocationManager. It subscribes to position updates with the criteria to get a new position every 0.1 seconds with no criteria on the distance to the last position.
    - This service is started as a foreground service and will display an notification once started. This is necessary the avoid androids limitation on localization information when the app is in the background.
- SpeedMonitorService.java
    - This service compares the current speed to the target speed and decides if the user is too fast or too slow. It plays a sound if the user is wrong.

other Java classes

The java classes used for this project are in "CoolRunning/app/src/main/java/ch/arebsame/coolrunning/" The following files are used:

- CoolRunningCom.java
    - This is a static class used for communication between various activities and services. This can be accessed to store or get current information about the apps status.
- GPXGenerator.java
    - This is used by the SpeedService to store the position information in a *.gpx file if the user wishes to save the track
- SpeedMonitor.java
    - This class is used by the SpeedMonitorService and compares the current to target speed and deices weather the user is too fast or too slow.
- TargetSpeedUpdater.java
    - This class is used by the SpeedMonitorService. It starts a thread that will periodically update the target speed according to the mode setting of the user.

Enums

Several emuns are used as common values for several definitions saved in "CoolRunning/app/src/main/java/ch/arebsame/coolrunning/".

- RunningError.java
    - This defines the states if the user is too fast, too slow or running correct.
- RunningMode.java
    - This defines the possible running modes for the user