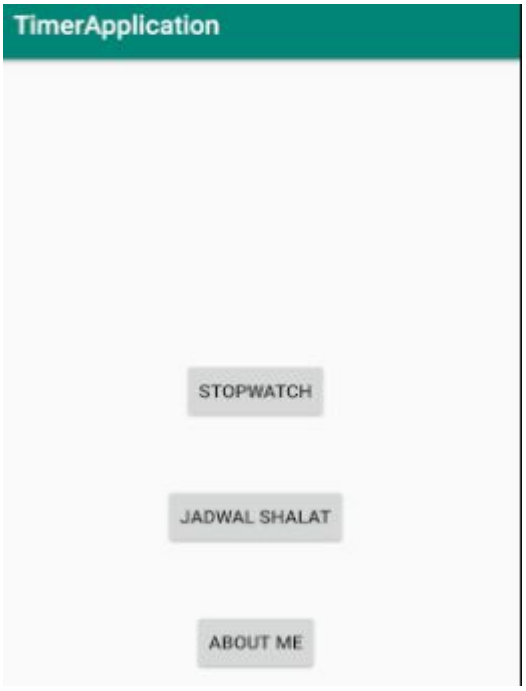
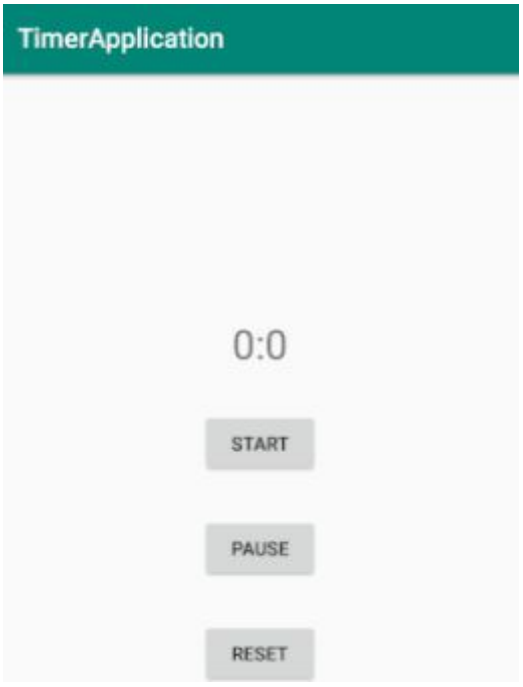
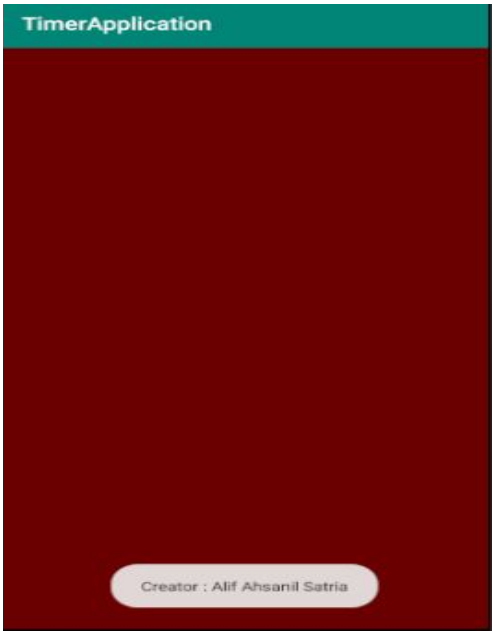
Nama : ALIF AHSANIL SATRIA
NPM : 1606882540

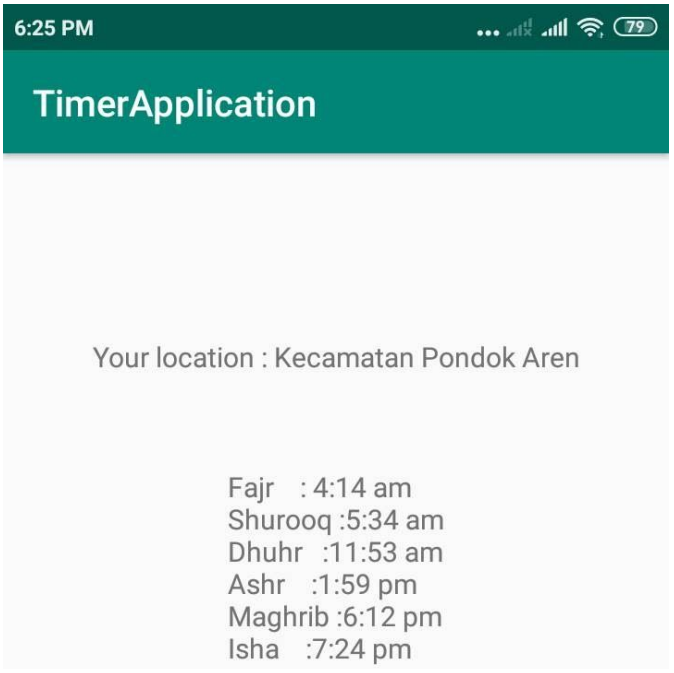
Apps ini digunakan sebagai stopwatch, menentukan jadwal shalat berdasarkan lokasi user saat ini, dan about me untuk memberikan informasi terkait creator apps ini (saya sendiri) sekaligus sebagai implementasi requirement opengl.

Berikut ini cuplikan tampilannya :









Sebagai informasi tambahan, somehow stopwatch nya crash ketika dicoba pertama kali pas apps nya di-install untuk pertama kalinya, tetapi ketika dicoba lagi untuk kedua, ketiga kalinya, dan seterusnya bisa digunakan. Saya sudah berusaha mencari tahu penyebabnya,

tetapi belum mendapatkan jawaban kenapa itu bisa terjadi. Lalu, user harus menyalakan gps terlebih dahulu untuk menggunakan fitur jadwal shalat karena apps nya butuh akses terhadap lokasi user saat ini.

Requirement :

1. **Menerapkan Runtime Permission**
   Mahasiswa masih perlu mendeklarasikan izin di manifest, kemudian aplikasi harus dibuat untuk meminta konfirmasi dari pengguna apakah akan memberikan izin atas resources (hardware maupun data). Ketika pengguna menyetujui akses maka aplikasi akan melanjutkan proses, namun apabila pengguna tidak menyetujui akses maka pengguna harus diarahkan ke halaman penjelasan yang berisi “mengapa aplikasi membutuhkan izin atas resource tersebut”. Cth: apabila aplikasi menggunakan Contact Number, SMS, dan GPS maka persetujuan runtime permission nya baru dimunculkan ketika resource akan diakses.

   Penjelasan :

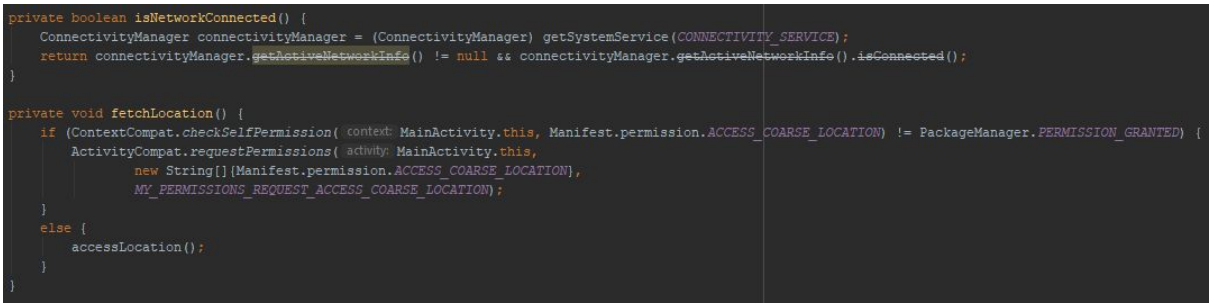
```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

   ACCESS_COARSE_LOCATION adalah deklarasi izin untuk mengakses lokasi user INTERNET untuk izin membuka network socket karena apps nya akan menggunakan internet. ACCESS_NETWORK_STATE digunakan untuk mengecek apakah kita terkoneksi dengan sebuah jaringan (dalam hal ini ingin mengecek koneksi ke internet) atau tidak. Ketiga permission ini berguna ketika kita akan mengakses fitur mencari jadwal shalat berdasarkan lokasi kita sekarang.

   Untuk implementasinya sendiri, terdapat pada **MainActivity.java**. Berikut ini adalah cuplikannya :

```
public void jadwalShalatActivity(View view) {
    if (isNetworkConnected()) {
        fetchLocation();
    }
    else {
        new AlertDialog.Builder( context: this)
                .setTitle("Required Internet Connection")
                .setMessage("You must connected to internet to access this feature")
                .setPositiveButton( text: "OK", (dialogInterface, i) -> {
                        dialogInterface.dismiss();
                })
                .create()
                .show();
    }
}
```

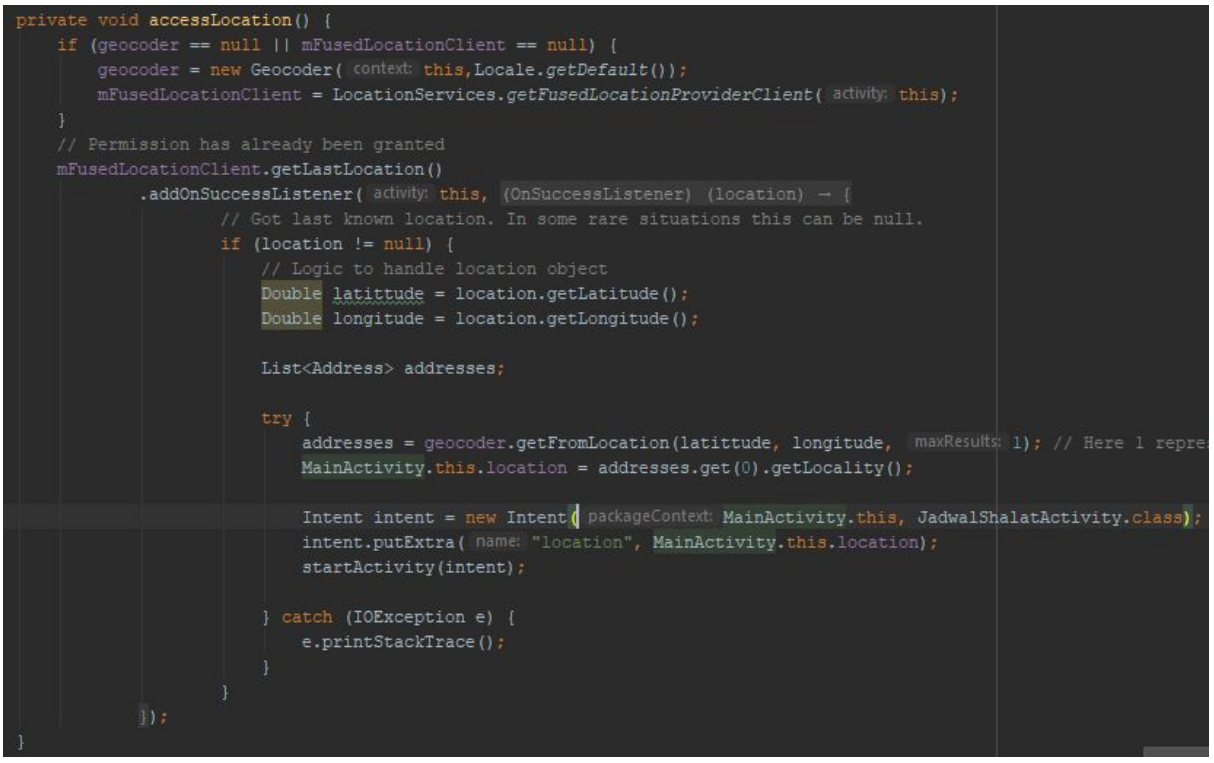
Pertama, yang dilakukan adalah mengecek dulu apakah kita terkoneksi ke internet atau tidak dgn method **isNetworkConnected()**. Jika iya, call **fetchLocation()**. Jika tidak, munculkan dialog yang menyatakan kita harus terkoneksi internet terlebih dahulu.



```
private boolean isNetworkConnected() {
    ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(CONNECTIVITY_SERVICE);
    return connectivityManager.getActiveNetworkInfo() != null && connectivityManager.getActiveNetworkInfo().isConnected();
}

private void fetchLocation() {
    if (ContextCompat.checkSelfPermission( context: MainActivity.this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions( activity: MainActivity.this,
                new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
                MY_PERMISSIONS_REQUEST_ACCESS_COARSE_LOCATION);
    }
    else {
        accessLocation();
    }
}
```

Ketika **fetchLocation()** dipanggil, kita mengecek apakah sudah diberikan akses terhadap data lokasi user atau tidak. Jika iya, call **accessLocation()**. Jika tidak, apps akan melakukan request permission terlebih dahulu apakah user mengizinkan location nya diakses atau tidak.



```
private void accessLocation() {
    if (geocoder == null || mFusedLocationClient == null) {
        geocoder = new Geocoder( context: this,Locale.getDefault());
        mFusedLocationClient = LocationServices.getFusedLocationProviderClient( activity: this);
    }
    // Permission has already been granted
    mFusedLocationClient.getLastLocation()
            .addOnSuccessListener( activity: this, (OnSuccessListener) (location) → {
                    // Got last known location. In some rare situations this can be null.
                    if (location != null) {
                        // Logic to handle location object
                        Double latittude = location.getLatitude();
                        Double longitude = location.getLongitude();

                        List<Address> addresses;

                        try {
                            addresses = geocoder.getFromLocation(latittude, longitude,  maxResults: 1); // Here 1 repre
                            MainActivity.this.location = addresses.get(0).getLocality();

                            Intent intent = new Intent( packageContext: MainActivity.this, JadwalShalatActivity.class);
                            intent.putExtra( name: "location", MainActivity.this.location);
                            startActivity(intent);

                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
            });
}
```

Method **accessLocation()** mencoba mengakses data lokasi user (dalam hal ini kita mengambil latitude dan longitude). Data latitude dan longitude digunakan untuk mengakses locality dari user yang mengakses apps pada saat itu. Setelah locality didapat, data ini akan di-pass sebagai extras sebelum akhirnya pindah ke **JadwalShalatActivity.java**.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    if(requestCode == MY_PERMISSIONS_REQUEST_ACCESS_COARSE_LOCATION){
        // Permission is not granted
        // Should we show an explanation?
        if(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
            accessLocation();
        }
        else if (ActivityCompat.shouldShowRequestPermissionRationale( activity: MainActivity.this,
                Manifest.permission.ACCESS_COARSE_LOCATION)){
            new AlertDialog.Builder( context: this)
                    .setTitle("Required Location Permission")
                    .setMessage("You have to give this permission to access this feature. " +
                            "We need to know your location in order to " +
                            "know the corresponding pray schedule")
                    .setPositiveButton( text: "OK", (dialogInterface, i) -> {
                            ActivityCompat.requestPermissions( activity: MainActivity.this,
                                    new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
                                    MY_PERMISSIONS_REQUEST_ACCESS_COARSE_LOCATION);
                    })
                    .setNegativeButton( text: "Cancel", (dialogInterface, i) -> {
                            dialogInterface.dismiss();
                    })
                    .create()
                    .show();
        }
    }
}
```

Method **onRequestPermissionResult()** akan dipanggil setelah user memberikan response terhadap **ActivityCompat.requestPermissions** . Jika user memberikan izin, call **accessLocation()**. Jika tidak, munculkan dialog yang menyatakan mengapa user harus memberikan izin ini.

2. **Memanfaatkan JNI (Java Native Interface)**

Perhatikan cuplikan kode berikut :

```
/**
 * A native method that is implemented by the 'native-lib' native library,
 * which is packaged with this application.
 */
public native int[] convertToMinutesAndSeconds(int count);
```

```
#include <jni.h>
#include <string>

extern "C" JNIEXPORT jintArray JNICALL
Java_com_example_timerapplication_StopwatchActivity_convertToMinutesAndSeconds(
        JNIEnv *env,
        jobject, jint count) {

    jintArray result;
    result = env->NewIntArray(2);
    if (result == NULL) {
        return NULL; /* out of memory error thrown */
    }

    jint minutes = count/60;
    jint seconds = count%60;

    jint minutesAndSeconds[2];
    minutesAndSeconds[0] = minutes;
    minutesAndSeconds[1] = seconds;

    env->SetIntArrayRegion(result, 0, 2, minutesAndSeconds);

    return result;
}
```

Apps ini menggunakan JNI untuk melakukan konversi dari **detik** ke **menit:detik**.

3. **Menampilkan animasi dengan OpenGL**

Perhatikan cuplikan kode berikut :

```
public class OGLView extends GLSurfaceView {
    public OGLView(Context context) {
        super(context);
        init();
    }

    public OGLView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    private void init() {
        // use opengl es 2.0
        setEGLContextClientVersion(2);

        // store opengl context
        setPreserveEGLContextOnPause(true);

        // set renderer
        setRenderer(new OGLRenderer());
    }
}
```

```
public class OGLRenderer implements GLSurfaceView.Renderer {

    private double redValue = 1.0f;
    private static final double DURATION_OF_FLASH = 1000.0; // 1 second

    @Override
    public void onSurfaceCreated(GL10 gl10, EGLConfig eglConfig) {
        GLES20.glClearColor((float)redValue, v1: 0.0f, v2: 0.0f, v3: 1.0f);
    }

    @Override
    public void onSurfaceChanged(GL10 gl10, int i, int i1) {

    }

    @Override
    public void onDrawFrame(GL10 gl10) {
        GLES20.glClearColor((float)redValue, v1: 0.0f, v2: 0.0f, v3: 1.0f);
        GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT);

        redValue = ((Math.sin(System.currentTimeMillis() * 2 * Math.PI / DURATION_OF_FLASH) * 0.5) + 0.5);
    }
}
```

Method **onSurfaceCreated** dipanggil ketika awal proses rendering, **onDrawFrame** dipanggil untuk menggambar frame, **onSurfaceChanged** dipanggil ketika size surface nya berubah, misalnya ketika layar hp kita berubah orientasinya. **onDrawFrame** diimplementasikan untuk menciptakan efek warna merah yang berkelap-kelip (untuk lebih lanjut silahkan dicoba apps nya).

```java
public class AboutMeActivity extends AppCompatActivity {

    private OGLView oglView;

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

        oglView = findViewById(R.id.oglView);

        Context context = getApplicationContext();
        CharSequence text = "Creator : Alif Ahsanil Satria";
        int duration = Toast.LENGTH_LONG;
        Toast toast = Toast.makeText(context, text, duration);
        toast.show();
    }

    @Override
    protected void onPause() {
        super.onPause();
        oglView.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
        oglView.onResume();
    }
}
```

AboutMeActivity digunakan sebagai medium untuk menempel view **OGLView**.

4. **Memanfaatkan ConnectivityManager untuk mengetahui status konektifitas yang dihubungkan dengan fitur tertentu**

```java
private boolean isNetworkConnected() {
    ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(CONNECTIVITY_SERVICE);
    return connectivityManager.getActiveNetworkInfo() != null && connectivityManager.getActiveNetworkInfo().isConnected()
}
```

Implementasi diatas digunakan untuk mengecek apakah HP kita sudah terkoneksi ke internet atau belum. Status konektivitas ini digunakan untuk fitur mengambil data jadwal adzan berdasarkan lokasi user saat ini.

5. **Menerapkan service background yang berhubungan dengan tampilan / aplikasi utama. Cth: Service pemutar lagu tetap berjalan walaupun tampilan/aplikasi utama sedang lost focus.**

Saya menggunakan JobScheduler untuk mengimplementasikan service background. Hal tersebut terdapat pada file **StopwatchActivity.java** dan **StopwatchJobService.java** . Selain itu, saya juga menggunakan SharedPreferences untuk menyimpan count detik pada stopwatch.

SharedPreferences dimanfaatkan untuk melakukan update detik pada apps dan notifikasi tepat ketika detiknya di-increment.

Perhatikan cuplikan kode berikut :

StopwatchJobService.java

```
@RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
public class StopwatchJobService extends JobService {
    private static final String TAG = "StopwatchJobService";
    private boolean jobCancelled = false;

    @Override
    public boolean onStartJob(JobParameters params) {
        Log.d(TAG,  msg: "Job started");
        doBackgroundWork(params);

        return true;
    }
```

```
private void doBackgroundWork(final JobParameters params) {
    new Thread((Runnable) () → {
            SharedPreferences countPref = getApplicationContext().getSharedPreferences( s: "countPref",  i: 0);
            SharedPreferences.Editor editor = countPref.edit();
            while (true) {
                if (jobCancelled) {
                    return;
                }

                int currentCount = countPref.getInt( s: "count",  i: 9999);
                editor.putInt( s: "count",  i: currentCount+1);
                editor.apply();

                try {
                    Thread.sleep( millis: 1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
    }).start();
}

@Override
public boolean onStopJob(JobParameters params) {
    Log.d(TAG,  msg: "Job cancelled before completion");
    jobCancelled = true;
    return false;
}
```

StopwatchActivity.java

```
public void scheduleJob(View v) {
    ComponentName componentName = new ComponentName( pkg: this, StopwatchJobService.class);
    JobInfo info = null;
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.LOLLIPOP) {
        info = new JobInfo.Builder( jobId: 123, componentName)
                .setOverrideDeadline(0)
                .build();
    }

    JobScheduler scheduler = null;
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.LOLLIPOP) {
        scheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);
    }
    int resultCode = 0;
    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.LOLLIPOP) {
        resultCode = scheduler.schedule(info);

    }
    if (resultCode == JobScheduler.RESULT_SUCCESS) {
        NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder( context: this, CHANNEL_1_ID)
                .setSmallIcon(R.drawable.ic_launcher_background)
                .setContentTitle("Stopwatch Notification")
                .setContentText("0:0")
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setCategory(NotificationCompat.CATEGORY_MESSAGE);

        notificationManager.notify( id: 1, notificationBuilder.build());

        Log.d(TAG,  msg: "Job scheduled");
```

```
    } else {
        Log.d(TAG,  msg: "Job scheduling failed");
    }
}
```

```
public void cancelJob(View v) {
    JobScheduler scheduler = null;
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
        scheduler = (JobScheduler) getSystemService(JOB_SCHEDULER_SERVICE);
    }
    switch (v.getId()) {
        case R.id.pauseButton:
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                scheduler.cancel( i: 123);
                Log.d(TAG,  msg: "Job cancelled");
            }
            break;
        case R.id.resetButton:
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                scheduler.cancel( i: 123);
                Log.d(TAG,  msg: "Job cancelled");
                SharedPreferences countPref = getApplicationContext().getSharedPreferences( s: "countPref",  i: 0);
                SharedPreferences.Editor editor = countPref.edit();
                editor.putInt( s: "count",  i: 0);
                editor.apply();
            }
            break;
    }
}
```

```
@Override
public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String s) {
    int updatedCount = sharedPreferences.getInt(s, i: 9999);
    TextView countTV = findViewById(R.id.count);

    int[] minutesAndSeconds = convertToMinutesAndSeconds(updatedCount);
    String formattedCount = Integer.toString(minutesAndSeconds[0]) + ":" + Integer.toString(minutesAndSeconds[1]);

    Log.d( tag: "Output", formattedCount);

    countTV.setText(formattedCount);
    Log.d(TAG,  msg: "Count Updated on TextView");

     NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder( context: this, CHANNEL_1_ID)
            .setSmallIcon(R.drawable.ic_launcher_background)
            .setContentTitle("Stopwatch Notification")
            .setContentText(formattedCount)
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .setCategory(NotificationCompat.CATEGORY_MESSAGE);

    notificationManager.notify( id: 1, notificationBuilder.build());
}
```

Ketika tombol **START** pada stopwatch ditekan, **scheduleJob** akan dieksekusi dan selanjutnya akan mengeksekusi **onStartJob**, dilanjutkan mengeksekusi **doBackgroundWork** yang akan increment detik demi detik secara asynchronous. Ketika detiknya di-increment, terjadi perubahan pada sharedpreferences, dan kemudian **onSharedPreferenceChanged** dipanggil untuk mengupdate count detik terbaru pada apps dan notifikasi.

Ketika tombol **PAUSE** atau **RESET** ditekan, **cancelJob** akan dieksekusi dan dilanjutkan dengan **onStopJob** akan dieksekusi juga. **PAUSE** untuk memberhentikan count detik sementara, **RESET** untuk mengulang hitungan detik dari nol.

6. **Menerapkan Notifikasi atas event-event penting yang terjadi saat aplikasi dalam keadaan lost focus. Cth: aplikasi *countdown* akan menampilkan notifikasi ketika *countdown* berakhir, aplikasi jadwal sholat yang menampilkan notifikasi kapan adzan dan kapan qomat, aplikasi menampilkan notifikasi judul lagu baru ketika ada pergantian lagu.**

   Penjelasan terkait notifikasi sudah sedikit disinggung pada point 5. Penjelasan tambahannya adalah notifikasi juga akan update hasil increment detik ketika aplikasinya sedang lost focus.

   Selain itu, terdapat cuplikan kode untuk membuat notification channel yang akan digunakan seperti pada point 5 :

```java
public class App extends Application {
    public static final String CHANNEL_1_ID = "channel1";

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

        createNotificationChannels();
    }

    private void createNotificationChannels() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel1 = new NotificationChannel(
                    CHANNEL_1_ID,
                    name: "Channel 1",
                    NotificationManager.IMPORTANCE_HIGH
            );
            channel1.setDescription("This is Channel 1");


            NotificationManager manager = getSystemService(NotificationManager.class);
            manager.createNotificationChannel(channel1);
        }
    }
}
```

Untuk implementasi apps ini, kita hanya menggunakan 1 channel notifikasi saja karena hanya membutuhkan notifikasi sebagai informasi detik terbaru hasil increment di apps kita.