**LAB # 10**

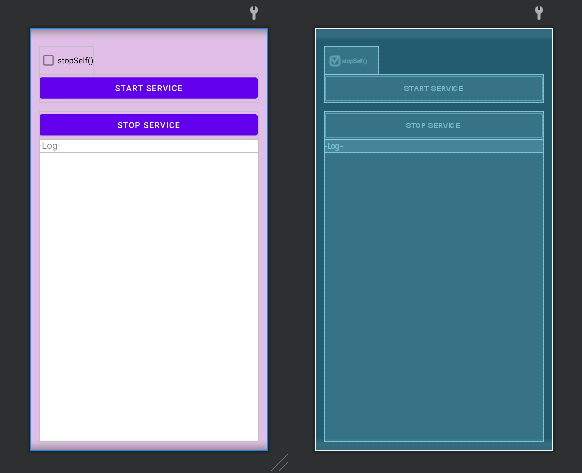
**Task #1: You have to create an Android application by which a started-service can be started or stopped by the user through start-button and stop-button respectively. There must be a provision in the UI to include stopSelf() method for execution if user want it.**

**Suppose all the service lifecycle callback methods are displaying their names in Logcat screen. These methods include only the follows:**

* **onCreate()**
* **onStartCommand()**
* **onDestroy()**

**Solution:**

**Design:**



**Code:**

**MyService Class:**

public class MyService extends Service {

private static final String TAG = "MyService";

private StringBuilder logCat = new StringBuilder();

private boolean stopSelf;

@Override

public void onCreate() { super.onCreate(); Log.d(TAG, "onCreate()"); appendToLog("onCreate()"); }

@Override

public int onStartCommand(Intent intent, int flags, int startId) {

Log.d(TAG, "onStart()"); appendToLog("onStart()"); stopSelf = intent.getBooleanExtra("stopSelf", false);

if (stopSelf) { appendToLog("stopSelf() requested"); }

broadcastLog(); return START\_STICKY; }

@Override

public void onDestroy() {

super.onDestroy(); Log.d(TAG, "onDestroy()"); appendToLog("onDestroy()"); broadcastLog(); }

@Nullable

@Override

public IBinder onBind(Intent intent) { return null; }

private void appendToLog(String message) { logCat.append(message).append("\n"); }

private void broadcastLog() { Intent logIntent = new Intent("LOG\_UPDATE"); logIntent.putExtra("logCat", logCat.toString()); LocalBroadcastManager.getInstance(this).sendBroadcast(logIntent); logCat.setLength(0);

if (stopSelf) { stopSelf(); } }}

**MainActivity Class:**

public class MainActivity extends AppCompatActivity {

private Button startButton;

private Button stopButton;

private CheckBox selfStopCheckBox;

private TextView textView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

startButton = findViewById(R.id.startButton);

stopButton = findViewById(R.id.stopButton);

selfStopCheckBox = findViewById(R.id.selfStopCheckBox);

textView = findViewById(R.id.logTextView);

textView.append("\n");

startButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

startService(); } });

stopButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

stopService(); } }); }

private BroadcastReceiver logReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

String logCat = intent.getStringExtra("logCat");

if (logCat != null) {

textView.append(logCat); } } };

@Override

protected void onStart() {

super.onStart();

LocalBroadcastManager.getInstance(this).registerReceiver(logReceiver, new IntentFilter("LOG\_UPDATE"));

}

@Override

protected void onStop() {

super.onStop();

// Unregister the log receiver

LocalBroadcastManager.getInstance(this).unregisterReceiver(logReceiver);

}

private void startService() {

Intent serviceIntent = new Intent(this, MyService.class);

boolean stopSelf = selfStopCheckBox.isChecked();

System.out.println("StopSelf value:"+stopSelf);

if (stopSelf) {

serviceIntent.putExtra("stopSelf()", true);

}

startService(serviceIntent);

}

private void stopService() {

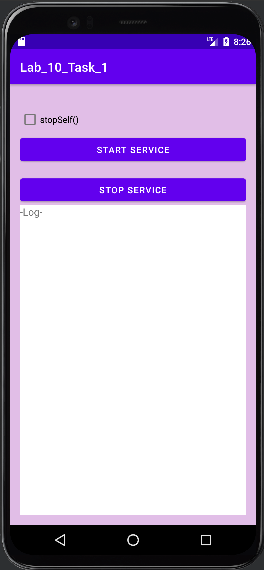
Intent serviceIntent = new Intent(this, MyService.class);

stopService(serviceIntent);

}

}

**Output:**



|  |  |
| --- | --- |
| **Without StopSelf():** | **With StopSelf():** |

**Task #2: Observe the behavior of your app, developed in Task-1, and fill out the table. User is assumed to restart the app every time before any of the following options is answered.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **stopSelf() execution option during service call** | |
| **stopSelf() is enabled** | **stopSelf() is disabled** |
| **Button pressing sequence** | **start** |  |  |
|
|
| **start, stop** |  |  |
|
|
| **start, stop, start** |  |  |
|
|