**Assignment 1**

**Ramesh Selvam**

**8960993**

Date: 2025-02-18

Android Wear Development

PROG - 8791

**Android Wear Stopwatch App**

**Screenshots:**

**A screenshot of a device

AI-generated content may be incorrect.A screenshot of a device

AI-generated content may be incorrect.**

**A screenshot of a stopwatch

AI-generated content may be incorrect.**

***MainActivity.Java***

package com.example.rameshselvam\_assignment;  
  
import android.os.Bundle;  
import android.os.Handler;  
import android.view.View;  
import android.widget.Button;  
import android.widget.TextView;  
import androidx.activity.EdgeToEdge;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.core.graphics.Insets;  
import androidx.core.view.ViewCompat;  
import androidx.core.view.WindowInsetsCompat;  
  
*/\*\*  
 \* This class implements a Stopwatch app for Wear OS with Start, Stop, and Reset functionalities.  
 \*/*public class MainActivity extends AppCompatActivity {  
  
 private TextView timerText;  
 private Button startButton, stopButton, resetButton;  
 private Handler handler = new Handler();  
 private long startTime = 0L, elapsedTime = 0L;  
 private boolean isRunning = false;  
  
 */\*\*  
 \* Runnable task to update the timer every second.  
 \*/* @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 EdgeToEdge.*enable*(this);  
 setContentView(R.layout.*activity\_main*);  
  
 // Apply window insets for Wear OS screen optimization  
 ViewCompat.*setOnApplyWindowInsetsListener*(findViewById(R.id.*main*), (v, insets) -> {  
 Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.*systemBars*());  
 v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);  
 return insets;  
 });  
  
 // Initialize UI components  
 timerText = findViewById(R.id.*timerText*);  
 startButton = findViewById(R.id.*startButton*);  
 stopButton = findViewById(R.id.*stopButton*);  
 resetButton = findViewById(R.id.*ResetButton*);  
  
 // Initially disable Stop and Reset buttons  
 stopButton.setEnabled(false);  
 resetButton.setEnabled(false);  
  
 // Start button functionality  
 startButton.setOnClickListener(v -> {  
 if (!isRunning) {  
 startTime = System.*currentTimeMillis*();  
 handler.post(updateTimer);  
 isRunning = true;  
 startButton.setEnabled(false);  
 stopButton.setEnabled(true);  
 resetButton.setEnabled(false);  
 }  
 });  
  
 // Stop button functionality  
 stopButton.setOnClickListener(v -> {  
 if (isRunning) {  
 handler.removeCallbacks(updateTimer);  
 elapsedTime += System.*currentTimeMillis*() - startTime;  
 isRunning = false;  
 stopButton.setEnabled(false);  
 startButton.setEnabled(true);  
 resetButton.setEnabled(true);  
 }  
 });  
  
 // Reset button functionality  
 resetButton.setOnClickListener(v -> {  
 handler.removeCallbacks(updateTimer);  
 elapsedTime = 0L;  
 timerText.setText("00:00:00");  
 isRunning = false;  
 startButton.setEnabled(true);  
 stopButton.setEnabled(false);  
 resetButton.setEnabled(false);  
 });  
 }  
 private Runnable updateTimer = new Runnable() {  
 @Override  
 public void run() {  
 long timeInMillis = System.*currentTimeMillis*() - startTime + elapsedTime;  
 int seconds = (int) (timeInMillis / 1000);  
 int minutes = seconds / 60;  
 int hours = minutes / 60;  
 seconds = seconds % 60;  
 minutes = minutes % 60;  
 timerText.setText(String.*format*("%02d:%02d:%02d", hours, minutes, seconds));  
 handler.postDelayed(this, 1000);  
 }  
 };  
}