```kotlin
package com.example.stopwatch

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.width
import androidx.compose.material3.Button
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.rememberCoroutineScope
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import com.example.stopwatch.ui.theme.StopWatchTheme
import kotlinx.coroutines.Job
import kotlinx.coroutines.delay
import kotlinx.coroutines.launch

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            StopWatchTheme {
                StopwatchApp()
            }
        }
    }
}

@Composable
fun StopwatchApp() {
    var time by remember { mutableStateOf(0) }
    var isRunning by remember { mutableStateOf(false) }
    val scope = rememberCoroutineScope()
```

```
49      var job: Job? by remember { mutableStateOf(null) } // Store
    the Job
50
51      Column(
52          modifier = Modifier
53              .fillMaxSize()
54              .padding(16.dp),
55          horizontalAlignment = Alignment.CenterHorizontally,
56          verticalArrangement = Arrangement.Center
57      ) {
58          Text(
59              text = formatTime(time),
60              style = MaterialTheme.typography.bodyLarge
61          )
62          Spacer(modifier = Modifier.height(16.dp))
63          Row {
64              Button(onClick = {
65                  isRunning = !isRunning
66                  if (isRunning) {
67                      job = scope.launch {
68                          while (isRunning) {
69                              delay(1000)
70                              time++
71                          }
72                      }
73                  } else {
74                      job?.cancel()
75                      job = null
76                  }
77              }) {
78                  Text(if (isRunning) "Stop" else "Start")
79              }
80              Spacer(modifier = Modifier.width(8.dp))
81              Button(onClick = {
82                  time = 0
83                  isRunning = false // Ensure isRunning is also
   reset!
84                  job?.cancel()    // Cancel any existing job
85                  job = null        // Clear the job reference
86              }) {
87                  Text("Reset")
88              }
89          }
90      }
91  }
92
93
94  fun formatTime(time: Int): String {
```

```
 95      val seconds = time % 60
 96      val minutes = (time / 60) % 60
 97      val hours = time / 3600
 98      return String.format("%02d:%02d:%02d", hours, minutes,
    seconds)
 99  }
100
```