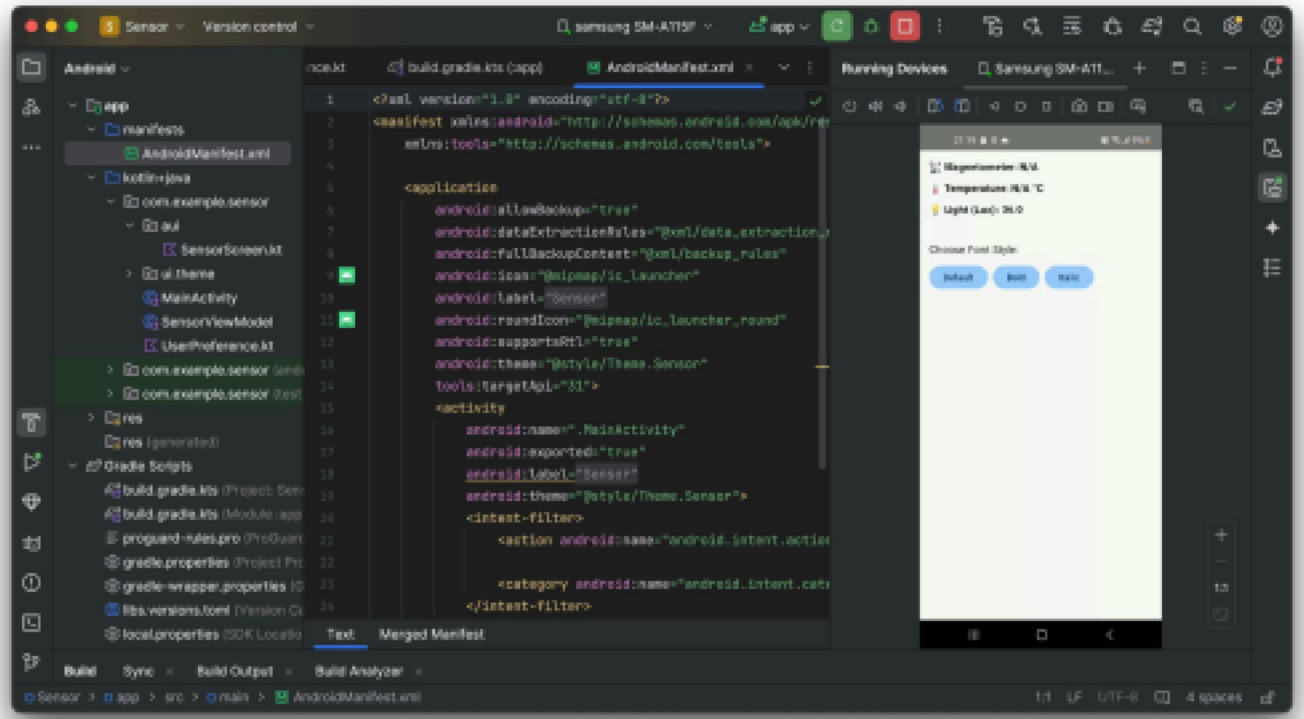
Name: **Daba**

# Mobile Technology and Programming

## Assignment L10_EX1



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/re
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="Sensor"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Sensor"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="Sensor"
            android:theme="@style/Theme.Sensor">
            <intent-filter>
                <action android:name="android.intent.actio

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

## MainActivity.kt

```kotlin
package com.example.sensor
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.viewModels
import androidx.compose.runtime.*
import androidx.compose.ui.text.font.FontStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.lifecycle.lifecycleScope
import com.example.sensor.aui.SensorScreen
import com.example.sensor.ui.theme.SensorTheme
import kotlinx.coroutines.flow.first
import kotlinx.coroutines.launch
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.runtime.LaunchedEffect
class MainActivity : ComponentActivity() {
    private val viewModel: SensorViewModel by viewModels() private lateinit
    var userPreferences: UserPreferences override fun
onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  userPreferences = UserPreferences(applicationContext)
  setContent {
  SensorTheme {
  val magnetometer by viewModel.magnetometer.collectAsState() val temperature by
  viewModel.temperature.collectAsState() val lux by
  viewModel.lux.collectAsState()

  var fontStyle by remember { mutableStateOf(FontStyle.Normal) } var fontWeight
  by remember { mutableStateOf(FontWeight.Normal) }

  LaunchedEffect(Unit) {
  val savedFont = userPreferences.getFontStyle().first() updateFont(savedFont) {
  fontStyle = it.first
  fontWeight = it.second
  }
  }
  // Main UI
  SensorScreen(
  fontStyle = fontStyle,
  fontWeight = fontWeight,
  magnetometerData = magnetometer,
  temperatureData = temperature,
  luxData = lux,
  onFontChange = { selected ->
  updateFont(selected) {
  fontStyle = it.first
  fontWeight = it.second
  }

  lifecycleScope.launch {
  userPreferences.saveFontStyle(selected)
  }
  }
  )
  }
  }
  }
  private fun updateFont(
  selected: String,
  apply: (Pair<FontStyle, FontWeight>) -> Unit
  ) {
  when (selected) {
  "bold" -> apply(Pair(FontStyle.Normal, FontWeight.Bold)) "italic" ->
  apply(Pair(FontStyle.Italic, FontWeight.Normal)) else ->
  apply(Pair(FontStyle.Normal, FontWeight.Normal)) }
  }
  }
```

## SensorViewModel.kt

```
package com.example.sensor
import android.app.Application
```

```
        import android.content.Context
        import android.hardware.Sensor
        import android.hardware.SensorEvent
        import android.hardware.SensorEventListener
        import android.hardware.SensorManager
        import androidx.lifecycle.AndroidViewModel
        import kotlinx.coroutines.flow.MutableStateFlow
        import kotlinx.coroutines.flow.StateFlow
        class SensorViewModel(application: Application) :
        AndroidViewModel(application),SensorEventListener {
        private val sensorManager =
        application.getSystemService(Context.SENSOR_SERVICE)asSensorManager

        private val _magnetometer = MutableStateFlow("N/A")
        val magnetometer: StateFlow<String> = _magnetometer
        private val _temperature = MutableStateFlow("N/A")
        val temperature: StateFlow<String> = _temperature
        private val _lux = MutableStateFlow("N/A")
        val lux: StateFlow<String> = _lux
        init {

        sensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD)?.let {
        sensorManager.registerListener(this, it, SensorManager.SENSOR_DELAY_NORMAL)}
        sensorManager.getDefaultSensor(Sensor.TYPE_AMBIENT_TEMPERATURE)?.let {
        sensorManager.registerListener(this, it, SensorManager.SENSOR_DELAY_NORMAL)}
        sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT)?.let {
        sensorManager.registerListener(this, it, SensorManager.SENSOR_DELAY_NORMAL)}
        }
        override fun onSensorChanged(event: SensorEvent?) {
        event?.let {
        when (it.sensor.type) {
        Sensor.TYPE_MAGNETIC_FIELD -> {
        val values = it.values
        _magnetometer.value = "X: %.2f, Y: %.2f, Z: %.2f".format(values[0],
        values[1], values[2])
        }
        Sensor.TYPE_AMBIENT_TEMPERATURE -> {
        _temperature.value = "%.1f".format(it.values.firstOrNull() ?:0f)}
        Sensor.TYPE_LIGHT -> {
        _lux.value = "%.1f".format(it.values.firstOrNull() ?: 0f) }
        }
        }
        }
        override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {
    }
}
```

## UserPreferences.kt

```
package com.example.sensor
import android.content.Context
import androidx.datastore.preferences.core.Preferences import
```

```
androidx.datastore.preferences.core.edit
import androidx.datastore.preferences.core.stringPreferencesKey import
androidx.datastore.preferences.preferencesDataStore import
kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.map
private val Context.dataStore by preferencesDataStore(name = "settings") class
UserPreferences(private val context: Context) { companion object {
        val FONT_STYLE_KEY = stringPreferencesKey("font_style") }
    suspend fun saveFontStyle(style: String) {
        context.dataStore.edit { settings ->
            settings[FONT_STYLE_KEY] = style
        }
    }
    fun getFontStyle(): Flow<String> {
              return context.dataStore.data.map { preferences ->
                   preferences[FONT_STYLE_KEY] ?: "default"
        }
    }
}
```