Assignment 1 – Battery Monitor Application

Objective

1. **Activity** – Create a main screen (Activity) with a simple UI that has a button to start/stop a background service and a TextView to display status.
2. **Intent** – Use Intent objects to start the service from the Activity.
3. **BroadcastReceiver** – Use a BroadcastReceiver to detect device battery-level changes (or any other system event) and send updates.
4. **Service** – Implement a Service that runs in the background, registers for battery-level broadcasts, and communicates status back to the Activity.
5. **User Interface** – Implement a simple layout in the Activity to show battery level and a service status indicator.

Features to Implement

1. **MainActivity**:
    - A layout with:
        - A TextView to show the current battery level (e.g., "Battery Level: 50%").
        - A Button to start the background monitoring service.
        - A Button to stop the background monitoring service.
        - Another TextView or any UI element showing whether the service is running or not.
    - When the user taps **Start Service**, the app starts a Service using an Intent.
    - When the user taps **Stop Service**, the app stops that Service.
2. **MonitoringService (Service)**:
    - Runs in the background.
    - Registers a BroadcastReceiver at runtime to listen for battery-level changes (or other interesting system events).
    - Whenever the battery level changes, the Service can send an update back to the Activity (for example, via a local broadcast or by storing data in a shared location, etc.).
3. **BatteryBroadcastReceiver (BroadcastReceiver)**:
    - Listens for the ACTION_BATTERY_CHANGED system broadcast (or any custom broadcast).
    - Extracts battery-level information.

- Passes that information to the Service or sends a local broadcast that the Activity can pick up.

4. **Manifest Configuration**:
    - Declare the Service in the AndroidManifest.xml.
    - If using a manifest-registered BroadcastReceiver, declare that as well
5. **Local or Global Communication**:
    - Use either a LocalBroadcastManager or a direct callback mechanism to communicate from the Service to the Activity. Alternatively, if you want to keep it simple, you can use a static variable or a singleton to store the latest battery info, and when the Activity is in the foreground, it reads from that variable.

Rubric for Evaluation

Project Setup & Structure (10 Points)

- **Criteria**:
    - Project compiles and runs without errors.
    - Proper folder/package structure.
    - AndroidManifest.xml accurately declares the Activity and Service (and BroadcastReceiver if needed).
    - Code is logically organized (classes/files are in appropriate directories).

2. Activity & UI Implementation (20 Points)

- **Criteria**:
    - Main Activity has required UI elements (e.g., TextView for battery level, Buttons to start/stop service).
    - Buttons function correctly (start/stop service).
    - Layout is user-friendly, clear, and follows standard UI guidelines.
    - Overall design is consistent and visually coherent.

3. Service Implementation (25 Points)

- **Criteria**:
    - Service correctly implemented (onCreate, onDestroy, etc.).
    - Proper registration and unregistration of BroadcastReceiver within the Service (if applicable).
    - Service runs in the background without crashing or freezing.

- Logic within the Service (e.g., handling battery changes or other events) is correct and efficient.

4. BroadcastReceiver Implementation (20 Points)

- **Criteria**:
  - BroadcastReceiver correctly detects the intended system event (e.g., ACTION_BATTERY_CHANGED).
  - Extracts relevant data from the broadcast Intent.
  - Forwards or communicates the data effectively to the Service or Activity (via LocalBroadcast, callback, etc.).
  - Registration/unregistration is handled properly (either in Service or Manifest).

5. Intents & Communication (10 Points)

- **Criteria**:
  - Appropriate use of Intents to start/stop the Service.
  - Data (if passed) is handled cleanly (use of Intent extras or local broadcasts).
  - Communication flow between Activity, Service, and BroadcastReceiver is clear and reliable.

6. Functionality & Correctness (10 Points)

- **Criteria**:
  - App performs the intended functionality without major bugs or crashes.
  - Accurate real-time updates (e.g., battery level) shown in the UI.
  - Service status reflection in the UI is correct (e.g., “Service Running” vs. “Service Stopped”).
  - Edge cases are handled (e.g., app resumes after pause, service restarts, etc.).

7. Code Quality & Documentation (5 Points)

- **Criteria**:
  - Code is clean, well-indented, and follows naming conventions.
  - Comments explain complex or critical sections of code.
  - Avoidance of unused or redundant code.
  - Overall readability and maintainability of the codebase.

Assignment Given out: 2/17/2025

Assignment Due: 03/07/2025

Submission:

1. Create a zip file for the project.
2. Upload the zip file on the Black board Assignment.