Native Android Sensor Data Collector using NDK (C++)

Project Overview:

This project involves building a native Android application that collects sensor data (e.g., accelerometer, gyroscope)
from an Android smartphone using C++ with the Android Native Development Kit (NDK). The data is processed or streamed
to an external system, such as a laptop, over a network connection for further analysis.

The project is designed to minimize Java/Kotlin dependencies, with the majority of the functionality (sensor access,
data processing, and network communication) implemented natively in C++.

Project Highlights:

- Native C++ Development: Core functionality, including sensor data collection, networking, and file I/O, is built using C++ with the Android NDK.
- Cross-Platform C++ Skills: Leveraging native C++ allows for more efficient use of system resources, with potential for cross-platform adaptation.
- Real-Time Sensor Data Handling: Accelerometer and gyroscope data are captured in real-time and streamed to an external system for further analysis.
- Network Communication: Uses TCP sockets in C++ to send data to a remote server (e.g., laptop) for further processing.
- Minimal Java/Kotlin Usage: Java is used only for initializing the user interface and binding the C++ library, emphasizing native development.

Key Technologies & Tools:

- Android NDK (Native Development Kit): Enables C++ code to run natively on Android.

- C++ (Core Language): Used to handle sensor data acquisition, processing, networking, and file management.

- Android Studio: Development environment for building Android applications with C++ support.

- JNI (Java Native Interface): Used to interface between Java (Android UI) and the native C++ layer.

- Socket Programming (C++): For network communication between the Android device and the external system (e.g., laptop).

Project Structure Overview:

1. Java Interface (Minimal Java/Kotlin Use):

- MainActivity.java: The primary Java class serves as the bridge between the Android operating system and the native C++ code.

- JNI Integration: Java calls to C++ functions using JNI for sensor data collection and communication.

2. Native Code (C++ via NDK):

- Sensor Management (C++): Handles direct sensor data acquisition using the Android NDK's ASensorManager API.

- Network Communication (C++): Manages socket communication to send sensor data to a remote system over TCP/UDP using native sockets.

- Data Processing (C++): Performs any real-time processing on the data, like filtering or formatting, before sending it to the remote system.

3. External System (Laptop/Server - C++):

- A companion program running on the laptop (or server) receives the sensor data from the Android device over a TCP connection.

Key Features:

1. Real-Time Sensor Data Collection: Use the accelerometer and gyroscope sensors on the phone to capture real-time data.

2. Native Data Processing: Data is processed natively using C++ to minimize latency and overhead.

3. Networking: The app sends the sensor data over a TCP or UDP connection using C++.

4. Efficient Resource Usage: Using C++ allows for more efficient memory and CPU usage on Android devices compared to Java/Kotlin.

Sample Workflow:

1. Initialize the NDK Environment: Set up an Android Studio project with C++ support enabled.

2. Access Sensor Data Using NDK: Use the Android NDK's ASensorManager to directly access the accelerometer, gyroscope, or other sensors in C++.

3. Process and Transmit Data: Collect sensor data (x, y, z values) in real-time and transmit it over a TCP or UDP connection.

4. External System (Laptop): Set up a C++ program that listens for incoming connections and displays/ processes the sensor data in real-time.

What You Will Learn:

1. Native Mobile Development: Understand how to work with the Android NDK to build efficient C++ applications on mobile devices.

2. Sensor Data Collection and Processing: Learn how to handle and process raw sensor data for various applications.

3. Networking with C++: Learn the fundamentals of network programming using sockets in C++.

4. Project Structure and Workflow: Understand how to structure an Android project that primarily uses C++ code and integrate it with Java for minimal interaction.

Resume Entry Example:

Native Android Sensor Data Collector using NDK (C++)

- Developed an Android application using C++ and the Android Native Development Kit (NDK) to collect and process real-time accelerometer and gyroscope sensor data.
- Implemented socket programming in C++ to transmit sensor data to a remote system for analysis, minimizing Java/Kotlin dependencies.
- Leveraged Android's ASensorManager for native sensor access, optimizing for real-time data acquisition and efficient resource usage.
- Built a companion C++ program on a laptop to receive and display the sensor data over a TCP network connection.
- Gained experience with JNI, Android Studio, and cross-platform development using C++.