Assignment Module 7 – Android Development With Media

1. Explain the concept of Kotlin Coroutines. How do coroutines improve performance over traditional threading mechanisms?

Kotlin Coroutines are a lightweight way to handle asynchronous tasks without blocking the main thread. They simplify code by allowing functions to be **paused and resumed** without creating multiple threads.

✪ How Coroutines Improve Performance:

1. **Lightweight** – Unlike traditional threads, coroutines don't create a new OS thread for each task, reducing memory and CPU usage.
2. **Non-blocking** – They suspend execution instead of blocking, allowing the main thread (UI) to remain responsive.
3. **Structured Concurrency** – Coroutines provide built-in scopes to manage tasks easily, reducing memory leaks.
4. **Easy Switching** – They switch between background and main threads seamlessly using Dispatchers.

2. Discuss the importance of unit testing in Android development. What is the difference between unit testing and UI testing?

○ Importance of Unit Testing in Android Development

Unit testing ensures that individual components (like functions or classes) work correctly and independently. It helps in:

- Catching bugs early
- Improving code quality and reliability
- Making refactoring safer
- Speeding up development with fewer regressions

○ Difference Between Unit Testing and UI Testing

Feature	Unit Testing	UI Testing
Focus	Tests individual functions or classes	Tests user interface and interactions
Speed	Fast (runs in milliseconds)	Slower (involves UI rendering)
Dependencies	Mocks external dependencies (e.g., API calls)	Uses real UI components
Tools	JUnit, Mockito	Espresso, UI Automator

3. Explain the steps involved in preparing an Android app for publishing. Discuss the significance of ProGuard and app signing.

○ Steps to Prepare an Android App for Publishing:

1. Code Optimization & Testing – Fix bugs, test thoroughly (unit & UI tests).
2. Versioning – Update versionCode and versionName in build.gradle.
3. Optimize & Shrink App – Use ProGuard to remove unused code and obfuscate it.
4. Sign the App – Generate a signed APK/AAB using a Keystore (required for publishing).
5. Generate App Bundle (AAB) – Recommended for Google Play to optimize delivery.
6. Prepare Store Listing – Write descriptions, upload screenshots, set pricing & policies.
7. Upload to Google Play Console – Submit for review and release!

○ Significance of ProGuard & App Signing

- ProGuard – Shrinks app size, obfuscates code (harder to reverse-engineer), and improves security.
- App Signing – Ensures app authenticity and prevents tampering; required for Play Store uploads.