NAME: García Ramírez Kerry Steven

GROUP: 4-B

Native, Non-Native, and Cross-Platform Applications

1. Native Applications

- **Definition**: Apps developed specifically for a particular operating system (OS), such as iOS or Android.
- **Languages**: Swift/Objective-C for iOS; Java/Kotlin for Android.
- **Advantages**:
    - High performance.
    - Full access to device features (camera, GPS, etc.).
    - Better user experience tailored to the OS.
- **Disadvantages**:
    - Development requires expertise in multiple programming languages.
    - Higher costs and longer development time if targeting multiple platforms.

2. Non-Native Applications

- **Definition**: Apps not specifically designed for an OS, often running in web browsers (e.g., web apps).
- **Languages**: HTML, CSS, JavaScript.
- **Advantages**:
    - Single codebase for all platforms.
    - Reduced development time and cost.
    - Easy updates without requiring app store submissions.
- **Disadvantages**:
    - Limited access to device features.
    - Performance may be slower compared to native apps.
    - Heavily dependent on internet connectivity.

3. Cross-Platform Applications

- **Definition**: Apps that run on multiple platforms but use a single codebase.

- **Frameworks**: Flutter, React Native, Xamarin.
- **Advantages**:
    - Efficient development with shared code.
    - Access to some native features via plugins or APIs.
    - Faster time to market.
- **Disadvantages**:
    - Performance may not match native apps.
    - Framework limitations can restrict advanced features.
    - Possible dependency on third-party tools.

Key Considerations:

- **Use Native**: When performance and user experience are top priorities.
- **Use Non-Native**: For simpler applications or those requiring broad compatibility.
- **Use Cross-Platform**: When balancing development efficiency and acceptable performance.

Sources:

- https://www.netguru.com/blog/cross-platform-vs-native-app-development
- https://www.youtube.com/watch?v=yux5kD-sVeA