# INTRODUCTION

Mobile applications have become an indispensable part of our lives. From communication and entertainment to shopping and banking, there's an app for practically everything. But before diving into the exciting world of mobile app development, understanding the different types of apps, programming languages, frameworks, and development processes is crucial. This report explores these key aspects to equip you with the knowledge needed to navigate the mobile app development landscape.

# REVIEW OF TASK 1

1. **Review and comparison of the major types of mobile apps and their differences (native, progressive web apps, hybrid apps)**

**Native Apps: The Gold Standard**

Native apps reign supreme in terms of user experience (UX) and performance. Developed with platform-specific languages (e.g., Java/Kotlin for Android, Swift/Objective-C for iOS), they seamlessly integrate with device features like GPS, camera, and sensors. This translates to smooth animations, intuitive interactions, and optimal resource utilization, leading to a superior user experience. Additionally, native apps have unrestricted access to the full spectrum of device functionalities, enabling developers to create feature-rich applications that leverage the device's full potential.

However, this advantage comes at a cost. Native app development requires separate codebases for each targeted platform (iOS, Android), significantly increasing development time and resource allocation. This "write-twice" approach can be a major drawback for projects with tight budgets or short timeframes.

**Progressive Web Apps (PWAs): Bridging the Gap**

PWAs offer a compelling alternative, blurring the lines between web and mobile experiences. Built with standard web technologies (HTML, CSS, JavaScript), they function within a web browser but deliver an app-like experience. Users can install PWAs on their home screens, enabling them to launch the app independently and even access limited functionalities offline, thanks to service workers – a technology that caches data for later use.

PWAs boast several advantages:

* **Cross-platform compatibility:** A single codebase caters to diverse user bases across various operating systems.
* **Reduced development time and cost:** Eliminates the need for separate native development efforts.
* **Improved discoverability:** Accessible through web searches, leveraging existing SEO strategies.

However, PWAs come with limitations. Access to certain device features may be restricted compared to native apps. Additionally, due to their reliance on web browsers, PWAs might experience performance inconsistencies across different platforms and devices.

**Hybrid Apps: A Compromise**

Hybrid apps attempt to strike a balance between native and web app functionalities. They utilize a native app container that embeds a webview component, essentially a mini-browser within the app. Developers leverage web technologies like HTML, CSS, and JavaScript to build the core functionalities, while the native container provides access to some device features.

Hybrid apps offer a middle ground:

* **Cross-platform development potential:** Shares a significant portion of codebase across platforms.
* **Partial access to native features:** Enables basic device integration through the native container.

However, compromises are inevitable. Hybrid apps might encounter performance bottlenecks compared to native apps, as communication between the native container and webview can add overhead. Additionally, the level of device integration and feature access may be limited compared to fully native applications.