

PEACE-WORK-FATHERLAND

MINISTRY OF HIHGER EDUCATION

UNIVERSITY OF BUEA

PAIX-TRAVAIL-PATRIE

MINISTERE DE L’ENSEIGNEMENT SUPERIEUR

UNIVERSITY OF BUEA

**UNIVERSITY OF BUEA**

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

CEF 440: INTERNET PROGRAMMING AND MOBILE PROGRAMMING

INTRODUCTION TO MOBILE APP DEVELOPMENT

Presented by:

EMADE ROSINE NSAH..........................................................FE20A032

ETUGE GIDEON......................................................................FE20A036

NOUGHUE LEMOUPA FRANK..............................................FE20A092

TAJOUEGO DJUIDJA ESTRELA............................................FE20A107

TAMAH JUSTENE....................................................................FE20A108

**Course Instructor**: Dr Nkemeni Valery

1. **WHAT ARE THE VARIOUS TYPES OF MOBILE APPLICATIONS THAT EXIST AND THEIR DIFFERENCES**

There exist different types of classifications of mobile applications; two of interest will be platform dependency and functionality. we could also classify by architecture.

Our focus is on platforms because they affect performance highest.

Based on the platform dependency category, there exist two types of mobile application: native and non-native applications

It is worth noting that non-native have four sub-categories varying from each other.

| **Native applications** | **Non-native applications** |
| --- | --- |
| Higher performance | Lower performance |
| Platform specific that is they share one codebase | Non-platform specific that is have one codebase |
| Can access hardware, software and firmware features completely and efficiently | Cannot access hardware, software and firmware features |
| Codebase for native applications are compiled to binary format which are environment specific | code base for non-native applications are compiled and their source code are stored in the application |
| The applications are well secured | The applications are less secured |
| The applications are built either with java, kotlin(android) , swift, objective C(iOS),C#(windows) | Applications are built using web technologies |
| Don't need a framework to communicate with the OS kernel | Need a framework to communicate with the OS kernel |
| Examples of such applications include whatsapp | Example of such applications include twitter,instagram |

1. **DEEP REVIEW OF PROGRAMMING LANGUAGES USED FOR MOBILE PROGRAMMING**

While reviewing mobile programming languages,the most sensitive aspects are below:

1. Environment: mobile programming languages are more efficient on some environments than others example java on android and swift on iOS
2. Performance: some mobile programming languages have an overall higher performance. This is due to their nature and structure, compiled languages like java,kotlin and C++ are more efficient than interpreted languages like python. Also,memory management will affect performance as in the case of C++ being better than java in gaming development as the first utilizes specific keywords for memory management and the later uses an automatic garbage removal system. A good example is the popular game minecraft moving from java to C++
3. Frameworks: Mobile programming languages are often used along frameworks to build mobile applications thus the popularity and the ability of the framework to communicate with the operating system on the mobile
4. User interface: Mobile programming languages need to render GUI components for a better UI/UX design. This is seen in native applications being better than non-native as GUI is rendered by **views** for both android and iOS in which native programming languages excel
5. Integration with device features: Mobile programming languages need to integrate device features such as cameras, GPS, and sensors in which native programming languages excel
6. Cross-platform compatibility: mobile programming languages that enable cross platform development provide a faster development so are adequate for SDLC methods
7. Community support: The size and activity of a mobile programming language's community can impact its popularity and usefulness, including factors such as documentation, tutorials, and online forums specific to mobile development.

The TIOBE and Github yearly state of the Octoverse provide different type of classification as follows:

1. TIOBE index: Here, programming languages are ranked according to how widely they are used. The index is updated every month, and many indicators, including open-source projects, job posts, and search engine results, are used to establish the ranking.

The ranking for programing languages as ranked by TIOBE for March 2023 are:

1. PYTHON
2. C
3. Java
4. C++
5. C#
6. Visual basics
7. Javascript
8. SQL
9. PHP
10. Go
11. Assembly
12. Matlab
13. Delphi
14. Scratch
15. Classic visual basics
16. R
17. Fortran
18. Ruby
19. Rust
20. Swift
21. CAS
22. FoxPro
23. Cobol
24. Perl
25. F#
26. Objective C

Kotlin dart were 35 and 36 according to the ranking respectively,typescript is 46

GitHub yearly state of the Octoverse: This is an annual report published by GitHub that provides insights into the state of the software development industry. The study contains information on many different subjects, including programming languages, open-source initiatives, and developer networks. An updated analysis on the condition of the software development sector, including details on use and trending programming languages, will be available in the 2023 GitHub yearly state of the Octoverse report.

The ranking for programing languages as ranked by Github for 2022 are:

1. JavaScript
2. Python
3. Java
4. Typescript
5. C#
6. C++
7. PHP
8. Shell
9. C
10. Ruby

**From the above rankings the languages obtained for mobile development are:** javascript,python,java,typescript,C#,C++,kotlin,dart,Go,swift which interest us. Server side languages like PHP and SQL are not ranked here.

**Ranking the mobile languages that made it according to performance is as below:**

**Compiled:** they are more performant. This include kotlin, dart,java,Go,C#,C++,swift and Objective C

**Interpreted**: they include typescript, python,javascript

**The ranking for programming languages as ranked by Versatility**(the ability to run on different platforms with the same codebase) are: typescript,javascript,python

In conclusion,building a highly performant application without time constraint making use of most of the hardware and software features will be done by compiled programming languages factoring in their various operating systems while building a less demanding application which requires speed in the development will be done with versatile languages. It's because of these reason that gaming engines are built using C#/C++/C and most social media applications are built using progressive web/hybrid technology

1. **WHAT IS MOBILE APPLICATION DEVELOPMENT FRAMEWORK**

This is an embedding software/firmware that provides for design and support for mobile applications providing various libraries and components that provide a fundamental structure to support the development of applications for a specific environment

1. **CLASSIFICATION OF MOBILE APPLICATION DEVELOPMENT FRAMEWORKS BASED ON**
2. **Programming Languages:**
3. **JavaScript-based frameworks**:
4. React native: this framework uses react and JavaScript to build mobile applications. It allows-s developers to develop code once and deploy it on both iOS and Android platforms
5. Ionic: it uses angular and JavaScript to create mobile applications for various platforms. Ionic is an open source, HTML-based framework that is designed to build hybrid mobile applications
6. Phone Gap (Apache Cordova): Cordova is a platform for building native mobile applications using HTML,CSS and JavaScript. It allows developers to build mobile applications for multiple platforms with a single codebase

**2. swift-based frameworks**:

1. Swift UI: this framework is built for developing user interfaces for iOS, IpadOS, macOS, and watchOS. Swift is a declarative framework and allows you to write less code and reuse it more effectively
2. UIKit: it's a framework designed for building iOS applications with the swift programming language. It is a mature framework and provides a wide range of supporting classes and components for building an iOS applications

3. **java-based frameworks:**

1. Android SDK: this framework is used to develop native android applications. It includes a set of development tools, libraries and APIs for building android applications

4. **Dart-based frameworks:**

1. Flutter: flutter is an SDK for building high-performance, high-fidelity, applications for iOS, android, web and desktop from a single codebase using the dart programming language. This SDK offers a rich set of customizable widgets and flexible APIs that can integrate with any third party device

B. **Complexity:**

1. **Low code/no-code frameworks**: these are frameworks that allow non-technical users to create mobile applications without writing any code. Examples include Appy pie and Mendix
2. **Native frameworks**: these are frameworks designed by the operating system to take the most advantage of the hardware and software but turn to have a higher a complexity

C. **User interface:**

1. **Native UI frameworks**: These are UI frameworks that are specific to a particular platform and provide access to all the native UI components and features. Examples include UIKit for iOS, Android SDK for Android, and Windows. UI.Xaml for Windows Mobile.

2. **Cross-platform UI frameworks**: These are UI frameworks that allow developers to create UI components that can be used across multiple platforms. Examples include React Native, Xamarin. Forms, and Flutter.

3. **Web-based UI frameworks:** These are UI frameworks that use web technologies like HTML, CSS, and JavaScript to create mobile apps with a web-based user interface. Examples include Ionic, PhoneGap, and Cordova.

4. **Game engine UI frameworks**: These are UI frameworks that are specifically designed for creating mobile games and provide access to game-specific features and functionality. Examples include Unity and Unreal Engine.

5. **Custom UI frameworks:** These are UI frameworks that are created by developers themselves for their specific app needs. They may use a combination of native, cross-platform, web-based, or game engine technologies.

**D. Performance**

1. **High-performance frameworks:**

1. **Native frameworks**: These frameworks include the Android SDK, UIKit, and SwiftUI. They offer the best performance because they are built specifically for the platform they will run on.

2. **React Native**: React Native is a high-performance framework that offers near-native performance. It achieves this by using a bridge that connects the JavaScript code to the native platform code.

1. **Medium-performance frameworks:**

1**. Xamarin**: Xamarin is a cross-platform mobile app development framework that uses C# and .NET to build mobile apps. Xamarin offers good performance because it has access to the native capabilities of the platform.

2. **Flutter:** Flutter is a high-performance framework that offers good performance by using its own rendering engine and rendering pipeline. It also provides a Just-In-Time (JIT) compilation for faster development cycles.

1. **Low-performance frameworks:**

1**. Apache Cordova/ PhoneGap**: Cordova and PhoneGap are hybrid mobile app development frameworks that use web technologies like HTML, CSS, and JavaScript. The performance of these frameworks is generally low because they rely on a bridge to access the native platform capabilities.

**2. Ionic:** Ionic is another hybrid mobile app development framework that uses web technologies to build mobile apps. The performance of Ionic is lower than the native and high-performance frameworks because it also relies on a bridge to access some of the platform capabilities.

Generally, the performance of a mobile app development framework depends on many factors, including the platform, programming language, and optimization techniques used by the framework. As a developer, it is essential to consider the performance characteristics of a framework when selecting one for your mobile app development project to ensure that it aligns with your application's requirements.

**E. Community Support:**

The frameworks below have been grouped according to their strengths

**1. Strong Community Support**: These frameworks usually have a large and active community of developers who share their knowledge, provide support, and contribute to the development of the framework. Examples of such frameworks include React Native, Flutter, and Ionic.

**2. Moderate Community Support:** These frameworks have a smaller community compared to the ones with strong support, but still have active developers who contribute to the framework. Examples of such frameworks include Xamarin, NativeScript, and PhoneGap.

**3. Weak Community Support:** These frameworks have a smaller and less active community of developers. They may still be useful for some specific use cases, but may not receive frequent updates and support from the community. Examples of such frameworks include Sencha Touch and jQuery Mobile.

In general, frameworks with strong community support are more likely to have regular updates, bug fixes, and feature enhancements. This makes it easier for developers to build high-quality apps and address any issues that arise during development

**F. Cost and time**

1. **Low-cost and fast development frameworks:**

1**. React Native:** React Native is a popular framework for building cross-platform mobile apps that offers cost and time advantages because it uses a single codebase for both iOS and Android platforms.

**2. Flutter:** Flutter is another popular framework that offers low-cost and fast development by using its own rendering engine and rendering pipeline. It also provides a hot reload feature, which makes the development process faster.

1. **Medium-cost and moderate development frameworks:**

**1. Xamarin:** Xamarin is a cross-platform mobile app development framework that uses C# and .NET to build mobile apps. Xamarin offers moderate development time and cost because it has access to the native capabilities of the platform and reduces the time and costs of developing, testing and maintaining two separate codebases for iOS and Android.

**2. Ionic**: Ionic is a hybrid mobile app development framework that uses web technologies to build mobile apps. The framework offers moderate development time and cost because it allows developers to use their existing web development skills, which reduces the learning curve and time to market for the first-time mobile app developers.

1. **High-cost and long development frameworks:**

1. **Native frameworks**: Native frameworks like Android SDK, UIKit and SwiftUI are high-cost and long development frameworks. They require developers with specific skills and lengthy development cycles, which makes the development process longer and more expensive.

**2. PhoneGap:** PhoneGap is a hybrid mobile app development framework that uses web technologies, but it requires additional development work for building a separate application for each platform (Android and iOs). This increases the cost and development time if you decide to deploy to both platforms.

Basically the cost and time characteristics of a mobile app development framework depend on the complexity of the app and the level of optimization required for the development process.

1. **HOW DO YOU COLLECT AND ANALYZE REQUIREMENTS FOR A MOBILE APPLICATION TO BE DEVELOPED**

There are basically two types of Requirements as concerns with mobile developments. That is, Functional and nonfunctional requirements.

**Functional requirements:** This are product features or functions that developers must implement to enable users to accomplish their tasks

**Nonfunctional requirements**: These are not related to the system's system functionality, but rather define how the system should perform.

These requirements could be collected and analyzed easily by grouping them into the following categories:

1) **Business requirements:** Here, high level statements of goals, objectives and needs are collected and analyzed. Business requirements could be collected and analyzed by;

a. **Identify the Business Goals and Objectives:** The first step in collecting requirements for mobile app development is to understand the business goals and objectives. This will help you identify the purpose of the app and the target audience.

b. **Conduct Market Research:** Conduct market research to understand the needs and preferences of your target audience. This will help you identify the features and functionalities that are most important for the app.

c.  **Create a User Persona:** Develop a user persona that represents your target audience. This will help you understand their needs, preferences, and behaviors.

d. **Define Functional Requirements**: Based on the business goals, user persona, and market research, defines the functional requirements for the app. These should include the features and functionalities that are essential for the app to meet its objectives.

e. **Prioritize Requirements:** Prioritize the requirements based on their importance to the business and users. This will help you focus on the most critical features and functionalities during development.

f. **Create Use Cases:** Create use cases that describe how users will interact with the app. This will help you identify any gaps in functionality or usability.

g. **Test and Validate Requirements:** Test and validate the requirements to ensure they meet the business goals and user needs. This can be done through user testing, surveys, or other feedback mechanisms.

2) **User requirements:** Here, the needs of discreet stakeholders are specify to define what they expect from a particular solution. User requirements for mobile developments could be collected and analyzed by;

a) **Identify User Needs:** The first step in collecting requirements for mobile app development using a user logic approach is to identify the needs of the users. This can be done through surveys, interviews, or other feedback mechanisms.

b) **Develop User Stories**: Develop user stories that describe how users will interact with the app. These stories should focus on the user's goals and objectives, rather than the technical details of the app.

c) **Create User Flows:** Create user flows that describe the steps users will take to achieve their goals within the app. This will help you identify any gaps in functionality or usability.

d) **Define Functional Requirements:** Based on the user stories and user flows, define the functional requirements for the app. These should include the features and functionalities that are essential for the app to meet user needs.

e) **Prioritize Requirements:** Prioritize the requirements based on their importance to the users. This will help you focus on the most critical features and functionalities during development.

f) **Test and Validate Requirements:** Test and validate the requirements to ensure they meet user needs. This can be done through user testing, surveys, or other feedback mechanisms.

g) **Iterate and Refine:** Use the feedback from testing and validation to iterate and refine the requirements. This will help ensure that the app meets user needs and is easy to use.

3) **Technical requirements**: Here, the factors required to deliver a desired function or behavior from a system to satisfy a user’s standards and needs are collected and analyzed. These could be achieved by;

a) **Identify Technical Requirements:** The first step in collecting requirements for mobile app development using a technical approach is to identify the technical requirements of the app. This can include the platform, programming language, database, and other technical specifications.

b) **Define Architecture**: Define the overall architecture of the app, including the software components, modules, and interfaces. This will help ensure that the app is scalable, maintainable, and extensible.

c) **Develop Use Cases:** Develop use cases that describe how users will interact with the app from a technical perspective. These use cases should focus on the technical details of the app, such as APIs, data storage, and security.

d) **Create Technical Requirements Document:** Based on the technical requirements and use cases, create a technical requirements document that outlines the technical specifications of the app. This document should include details such as system requirements, database schema, and API specifications.

e) **Prioritize Technical Requirements:** Prioritize the technical requirements based on their importance to the overall functionality and performance of the app.

f) **Test and Validate Technical Requirements:** Test and validate the technical requirements to ensure they meet the functional and performance requirements of the app. This can be done through unit testing, integration testing, and other types of testing.

g) **Iterate and Refine**: Use the feedback from testing and validation to iterate and refine the technical requirements. This will help ensure that the app is technically sound and meets the functional and performance requirements.

1. **HOW TO ESTIMATE THE MOBILE APPLICATION DEVELOPMENT COST**

Estimating the cost of mobile application development can be a challenging task as it depends on several factors such as the complexity of the applications, the platform(s) it will be developed for, the features and functionalities required, the level of design and user experience and the development team’s rates and location. Below are some key steps to consider when estimating the cost of the mobile application development:

1. Define the project requirements in detail, including the features and functionalities required, the platform it will be developed for, the level of design and user experience as well as any other relevant detail
2. Determine the size and composition of the development team required to complete the project including developers, designers, project managers and QA testers
3. Calculate the hourly rates of the team members based on their skills and experience level. This can be done by researching industry rates to similar roles and locations
4. Estimate the project timeline for each stage of the development process including design, development, testing and deployment
5. Consider any additional costs that may be incurred such as third-party API integration, hosting fees or application store submission fees
6. Review the estimate and adjust it as necessary based on any changes in requirements or scope