REPUBLIQUE DU CAMEROON
PAIX-Travail-Patrie
MINISTRE DE L'ENSEIGNEMENT SUPERIEUR

FACULTE D'INGINERIE ET TECHGNOLOGIE

REPUBLIC OF CAMEROON
Peace-Work-Fatherland
MINISTER OF HIGHER EDUCATION

FACULTY OF ENGINEERING AND TECHNOLOGY

********** UNIVERSITY OF BUEA **********

COURSE TITLE: INTERNET PROGRAMMING AND MOBILE PROGRAMMING

PROJECT: TASK ONE

COURSE CODE: CEF 440

PRESENTED BY:

NAME	REGISTRATION NUMBER
IHIMBRU ZADOLF ONGUM	FE21A203
CHE KASSINA KUM	FE21A158
NFOUA EUGENE MGBA	FE21A257
FONJI DANIEL KUKUH	FE21A194
EPIE MUKEH SANDRA	FE21A185

COURSE FACILITATOR: **APRIL 2024**

Dr. VALERY NKEMENI

Table of Contents

1.Review and comparison between the major types of mobile apps (Web, Hybrid and Native app)

1.1 Review

The mobile app development industry is constantly evolving. Given the benefits of mobile apps, their structure, architectural programming, and languages become essential. These languages determine the development speed, testing bandwidth, flexibility & scalability of mobile apps. Thus, there is substantial demand for app development languages and the right developers who can work with them. Below is an elaboration on some of the most popular programming languages for mobile app development.

1.2 Comparison

1.2.1 WEP APP

A web app or browser-based app can deliver similar functionality to an app as a website. In fact, with a little creativity, you can keep the differences to a minimum and design a web app so it looks and feels pretty much like a native app. There are different approaches to help you create successful mobile websites, such as responsive and adaptive design.

- **Progressive Web Apps (PWAs):** These web applications deliver an app-like experience using modern web capabilities. They work on any standards-compliant browser platform. Features like offline access, push notifications, and device hardware integration make them similar to native apps. PWAs don't require app store downloads to simplify maintenance and updates.
- **Responsive Web Apps (RWAs): RWAs** adapt to different screen sizes. They use responsive design to adjust the layout and content automatically. This ensures a consistent and smooth experience across all devices, from smartphones to desktops. RWAs prioritize fluid design and cross-device compatibility for diverse internet access methods.

1.2.2 **NATIVE APP**

NATIVE APP A native app only runs on a specific mobile operating system. It won't run on other mobile operating systems. So, for example, if you develop a native app for iOS, you'd do the development in the Xcode environment using Swift. For Android, developers commonly use Android Studio and Java. Native apps can normally access all the functionality of the chosen device easily. You can run them without error on the device if developed properly. However, this comes with a trade-off. If you want your app to run on iOS and Android, you will have to develop the app twice, once for each operating system. This can make the development process both slower and more expensive. Many companies will develop their app for a single operating system when they choose the native route. If the app is successful in that environment, they will then go back and recreate it for other operating systems.

1.2.2 **HYBRID APP**

A hybrid app works on multiple platforms. You write it with a single standard code language (such as C# or a combination of HTML5 and JavaScript) and then compile and execute it on each platform. The use of plugins for that operating system will manage device-specific interactions. Hybrid development environments include: 1. Ionic: A popular open-source framework for building hybrid mobile apps using web technologies such as HTML, CSS, and JavaScript. 2. React Native: A framework for building mobile apps using React, a popular JavaScript library for building user interfaces. 3. Xamarin: A framework for building mobile apps using C# and .NET, which allows developers to share code across multiple platforms.

1.3 Differences

Differences

Feature	Native App	Web App	Hybrid App
Development Language	Platformspecific (e.g., Swift for iOS)	Web tech-Nologies (HTML, CSS, JavaScript)	Combination of native and web technologies
Performance	High with smooth interactions	Dependent on browser and internet speed	Good, but may not match mobile apps
Device Feature Access	Full access to device hardware	Limited access	Moderate access depends on plugins
Offline Access	Generally available	Limited or none	Often available

Compatibility	Specific to each platform (iOS, Android, etc.)	Universal across all devices with a web browser	Broad, but with some platformspecific tweaks
Development Cost	Higher due to platform specificity	Lower	Moderate, less than native
Maintenance	Frequent updates required	Easier to update, centralized	Regular updates needed, but fewer than separate mobile apps for each platform
Internet Dependency	Low	High	Low
User Experience	Optimal, tailored to each platform	Varies, generally good	Good, aims to mimic native experience

2.Review and comparison of mobile app programming languages

2.1 . Swift

As the official language for iOS app development in 2022, Swift is still considered a younger language, overtaking Objective-C as the language of choice by Apple after it was introduced in 2014.

Swift is very user-friendly, ideally suited to new programmers, focusing on expressiveness, safety, and speed. However, Swift is highly dependent on third-party tools and lacks the maturity of other languages. Still, there is no more natural choice for iOS development than swift.

2.2. Kotlin

The official Android programming language is Kotlin, although it is one of many languages that we can use for Android app development. Kotlin is a simple language with a powerful, clean syntax and combines both acquisitive and functional programming features to allow for faster compilation.

2.3 JavaScript

JavaScript is a top language choice by many developers for mobile apps in coordination with frameworks such as React Native, Cordova,
Native was created by Facebook, giving it the backing to make it a dominant player in the development of mobile apps, particularly those focused on native UI elements – without the need to know native programming languages such as Swift or Kotlin.

2.4. Objective-C

Once the official language for iPhones, Objective-C remains a common iOS programming language due to its stable performance, resource availability, and compatibility with C++.

2.5. Java

Once the official language for iPhones, Objective-C remains a common iOS programming language due to its stable performance, resource availability, and compatibility with C++.

2.6. Python

Python has been around since the 80s and is still considered the top programming language, making it an ideal beginner choice due to its versatility in creating mobile apps and a wide variety of other software. It is English-syntax based with many robust libraries. Although not always considered a “mobile app coding language,” given the availability of native language options, Python is often relied upon for more specialized mobile development in areas such as data science.

Figure 1: **Popular Programming Languages for Mobile Apps**

3. Review and compare mobile app development frameworks by comparing their key features (language, performance, cost & time to market, UX & UI, complexity, community support) and where they can be used.

3.1 Table of comparison

Framework	Language	Performance	Cost & Time to Market	UX & UI	Complexity	Community Support	Where it can be used
React Native	JavaScript	**High**: Utilizes native components via JavaScript bridge. Performance can suffer for complex UI animations.	**Moderate**: Single codebase for iOS and Android reduces development time. Some platform-specific customization may be needed.	**Good**: Native-like feel with platform-specific UI components.	**Moderate**: Learning curve due to JavaScript and platform-specific nuances.	**Large**: Extensive community support with active development and frequent updates.	Cross-platform (iOS, Android)
Flutter	Dart	**High**: Compiles to native code, providing near-native performance.	**Low**: Hot reload feature speeds up development. Single codebase for iOS and Android reduces time to market.	**Excellent**: Customizable widgets for pixel-perfect designs.	**Moderate**: Learning curve for Dart language and Flutter's reactive programming model.	**Growing**: Active community with increasing adoption and support.	Cross-platform (iOS, Android)
Xamarin	C#	**High**: Compiles to native code, providing near-native performance.	**Moderate**: Shared codebase reduces development time, but platform-specific adjustments may be required.	**Good**: Native UI with Xamarin.Forms or platform-specific UI with Xamarin.iOS and Xamarin.Android.	**Moderate**: Learning curve for C# and Xamarin APIs.	**Large**: Established community with strong Microsoft support.	Cross-platform (iOS, Android)
Ionic	HTML/CSS/JS	**Moderate**: Performance depends on the performance of the	**Low**: Rapid development with web technologies. Easy deployment to	**Good**: Utilizes web technologies for flexible and responsive UI.	**Low**: Easy to learn for web developers. Limited native features may	**Large**: Active community with extensive documentation and plugins.	Cross-platform (iOS, Android)

		WebView componen t.	multiple platform s.		require addition al plugins.		
NativeScript	JavaScript	**High**: Direct access to native APIs for optimal performanc e.	**Moderate**: Shared codebase reduces development time. Platform-specific adjustments may be needed.	**Good**: Native UIcomponents forseamless user experience.	**Moderate**: Learning curve due to JavaScript and native platform APIs.	**Moderate**: Supportive community with regular updates and plugins.	Cross-platfor m(iOS, Android)
Swift (iOS)	Swift	**High**: Native code execution ensures optimal performanc eon iOS devices.	**High**: Swift's modern language features and native development environment can speed up development but limited toiOS only.	**Excellent**: Leverages nativeiOS UI components forbest user experience.	**High**: Swift can have a steep learning curve for developers new to the language and iOS developmen t.	**Large**: Robust community supported byApple's ecosystem.	iOS only
Kotlin (Android)	Kotlin	**High**: Kotlin compiles to bytecode and runs on the JVM, providing performanc e comparable to Java.	**High**: Kotlin's concise syntax and interoperabilit ywith Java can accelerate development.	**Excellent**: Utilizes modern Android UI toolkit (Jetpack)for flexible and intuitive UI.	**Moderate**: Kotlin's syntax may require adjustment for developers familiar with Java.	**Large**: Kotlin isofficially supported by Google with growing community adoption.	Androi donly

4. Study mobile application architectures and design patterns

4.1 **Definition**

- An application architecture describes the patterns and techniques used to design and build an application. The architecture gives you a roadmap and best practices to follow when building an application, so that you end up with a well-structured app. Software design patterns can help you to build an application.
- Mobile app architecture includes various layers or components, each with specific responsibilities, and it determines how data is processed, presented to users, and interacted with. It plays a critical role in the app's performance, scalability, maintainability, and security.

4.2 **Key components of Mobile App Architecture**

- **User Interface (UI) Layer :** The UI layer is responsible for the presentation of the app to the user. It includes the visual elements and components that users interact with, such as screens, buttons, forms, navigation menus, and any graphical elements. It manages the layout and appearance of the app. Common technologies used in the UI layer include UI frameworks, user interfacelibraries, and design tools that help create a visually appealing and responsive user experience.
- **Application Logic Layer :** The application logic layer, also known as the business logic layer, houses the core functionality of the app. It includes algorithms, business rules, and processes that control the app's behavior. This layer processes user input, orchestrates data retrieval and storage, and ensures the correct operation of the app's features.
- **Data Layer :**The data layer manages data storage, retrieval, and communication with external data sources. It includes databases, server APIs, and any data repositories that the app interacts with. This layer ensures data integrity, security, and availability. In the data layer, technologies like databases (SQL or NoSQL), RESTful or GraphQL APIs, and caching mechanisms are commonly used. Data management frameworks, Object-Relational Mapping (ORM) libraries, and data synchronization tools may also be part of this layer

4.3 **Types of Mobile App Architecture**

- **Monolithic Architecture**

In monolithic architecture, all components and modules of an application are tightly integrated into a single, unified unit. In a monolithic architecture, the entire application, including the user interface, application logic, and data storage, is bundled together as a single codebase and runs within a single process.

Some advantanges of Monolithic Architecture include;

● Easier to develop, test, and initially deploy

● A wealth of knowledge and tools available for monolithic development

● Does not require communication between separate services

- **Microservices Architecture**

In a microservices architecture, an application uses small, independent, and loosely coupled services that operate simultaneously to provide its functionality.

Instead of building a monolithic application where all features and functions are tightly integrated into a single codebase, a microservices architecture breaks down the application into a collection of individual services, each responsible for a specific set of tasks or functionalities.

Some advantages of Microservices Architectures include

● Microservices can be individually scaled up or down as needed

● Flexibility to choose the most appropriate technology for each service

● Smaller, focused teams can work on individual services, leading to faster development cycles and quicker time-to-market for new features. 3) Model-View-Controller (MVC) Model-View-Controller (MVC) is a widely used architectural design pattern for develop

- **Model-View-Controller (MVC)**

Model-View-Controller (MVC) is a widely used architectural design pattern for developing software applications, particularly in web and mobile applications.

It divides the application into three interconnected components, each with a specific role and responsibility. The primary purpose of the MVC pattern is to separate the concerns of an application, making it easier to develop, maintain, and extend.

● Model: It represents the data and business logic of the app. It encapsulates the core functionality and data management of the application. This component is responsible for data storage, retrieval, validation, and processing. It responds to requests from the Controller and notifies the View when the data changes.

● View: Its responsibility is to render the user interface and present data to the user. It displays the information from the Model to the user in a visually appealing and understandable format.

● Controller: It works like an intermediary between the Model and the View. It receives user input, processes it, and communicates with the Model to retrieve or update data. The Controller decides which View should be displayed to the user based on the user's actions.

- **Model-View-ViewModel (MVVM)**

Model-View-ViewModel (MVVM) is an architectural design pattern used primarily in software development for building user interfaces. MVVM is especially popular in modern mobile and web applications.

It's an evolution of the Model-View-Controller (MVC) pattern, designed to enhance the separation of concerns and improve the testability and maintainability of code.

● Model: The Model represents the application's data and business logic. It is responsible for data storage, retrieval, validation, and processing.

● View: The View represents the user interface and is responsible for displaying data to the user. In MVVM, the View is the visual component of the application, such as a screen in a mobile app or a web page in a web application.

● ViewModel: The ViewModel serves as an intermediary between the Model and the View. It encapsulates the presentation logic, transforms the data from the Model into a format that is suitable for the View, and handles user interactions.

5. Study how to collect and analyse user requirements for a mobile application (Requirement Engineering)

Collecting and analyzing user requirements is crucial for building a successful mobile application. Here are some steps that follows:

5.1 Define App Idea and Purpose:

• Clearly define the problem your app solves or the value it provides.

• This foundational step helps attract the right users and guides requirement gathering.

5.2 Conduct Market and Competitor Analysis:

• Research existing apps in your target space.

• Identify their strengths, weaknesses, and any unmet user needs they present.

• This analysis can spark feature ideas and inform your approach.

5.3 Identify Your Target Users:

• Develop user personas - fictional representations of your ideal users.

• Include demographics, behaviours, goals, and challenges they face.

• Personas help you understand user needs and tailor the app accordingly.

5.4 User Research Techniques:

• There are various methods to gather user requirements

o Surveys and Questionnaires: Gather quantitative data on user preferences and pain points through online surveys or paper questionnaires.

o Interviews: Conduct in-depth interviews with target users to understand their motivations, frustrations, and app usage habits.

o User Testing: Observe users interacting with prototypes or mockups of your app to identify usability issues and gather feedback on features.

5.5 Analyze and Prioritize Requirements:

• Once you have collected data, analyze it to identify common themes and user needs.

• Group similar requirements and prioritize them based on importance, feasibility, and user impact.

• Not all ideas can be implemented initially, so prioritization helps focus development efforts.

Additional Tips:

• Active Listening: During user research, actively listen and avoid leading questions. Let users express their needs freely.

• Iteration is Key: Don't expect to get everything perfect initially. Gather feedback throughout the development process and be prepared to iterate on your requirements.

• Analytics Integration: Consider integrating app analytics tools to track user behavior and identify areas for improvement after launch.

6. Estimating mobile app development cost.

6.1 **Review**

How much does it cost to make an app? Likely, this is one of the first questions that app brands and developers need to know an answer for, when they launch an app development project. An app development cost estimate is the most basic and yet crucial component of a budget for any mobile app development project. Quite often a company owner finds herself trying to answer these two questions – how much does it cost to create an app for my business via hiring an app development company versus how much does it cost to develop an app internally.

Determining the cost of mobile app development isn't a one-size-fits-all process. It involves several factors such as features, complexity, platforms d that vary from one project to another.

6.2 Factors that Influence Mobile Development Cost

1. **Features and functionalities of the App:**

A simple app with basic functionality will cost significantly less than a complex app with advanced features like data synchronization, user authentication, complex backend. More features will require more more number of screens/pages in the app. More screens mean more design, development, and testing effort, thus higher cost.

2. **Platforms (iOS, Android, Both):** The type of platform chosen also affects the cost. Developing for one platform (iOS or Android) is less costly than creating a cross-platform app. However, depending on your target audience, you may need to consider developing for both platforms. Also, the cost for developing an iOS app is higher than that of an android app.

3. **Design (UI and UX):** Complex designs with custom graphics, animations, and transitions (like Amazon mobile app) will require more effort, hence increased costs. Simpler designs, (such as a simple calculator app) on the other hand, will be less expensive but might not stand out in the crowded app market.

4. **Third-party integration and Backend Development:** Building a mobile app that fetches data from a server or requires real-time synchronization (like in a chat app), a backend server will need to be developed, increasing costs. Similarly, if your app needs to interact with other apps or services using APIs, it will require additional development effort.

5. **Location and Development Team:** The location of your development team significantly influences the cost due to varying labor rates. Developers in North America tend to be more expensive than those in Africa. Additionally, a larger team with specialized roles (UI/UX designers, front-end and back-end developers, QA testers) will ensure a high-quality app but will also increase costs.

6. **Maintenance and Updates:** Post-launch costs like server costs, emergency maintenance, app updates, and customer support are often overlooked but should be factored into the overall budget. Regular updates to keep up with platform changes (like new OS releases) and new feature additions are also a part of the total cost.

7. **Development Time:** Development time depends on factors like the complexity of features, integration requirements, testing, and iteration cycles. Agile development methodologies, which involves iterative development and frequent feedback, may affect the cost compared to traditional waterfall methods.

6.3 Method for Estimating Mobile App Development Costs

1. **Fixed Price Model**: Ideal for small projects with well-defined and unchanging requirements, this model ensures clients know the total project cost upfront. While it aids in budget planning, it lacks flexibility for changes once development begins.

2. **Time and Material Model:** Suited for projects with evolving or undefined requirements, clients are billed based on actual time and resources utilized. This model allows for flexibility during development but may result in costs exceeding initial estimates if project scope expands.

3. **Dedicated Team Model:** Tailored for long-term projects with changing requirements, clients benefit from a dedicated team working exclusively on their project. Costs are based on team size and time spent, offering control over the development process and ensuring team commitment.

6.4 How to Reduce Mobile App Development Costs

Set clear goals for your app: Comprehensive planning at the outset can save both time and money down the line. A clear understanding of your target audience, the app's purpose, and its key functionality requirements is crucial. This also helps avoid unnecessary revisions or feature additions in later stages, thus preventing cost overruns. A handy framework to use is the **SMART** (Specific, Measurable, Attainable, Relevant, and Time bound) goals framework

Develop a cross-platform app: Cross-platform development is an increasingly popular approach to mobile app development. According to NetGuru, building a cross-platform app can be up to 30% cheaper than building 2 separate native apps. That's because you only need a single team of engineers to build a cross-platform app.

And the app uses a single codebase to run on both operating systems. This significantly reduces development time and that can save you a lot of money.

Adopting a Minimum Viable Product (MVP) Approach: Prioritizing the development of a minimum viable product (MVP) is a great way to reduce your mobile app development costs. This strategy focuses on developing and releasing a version of the app with only the most essential features that bring users the most value. MVP approach can significantly lower mobile app development costs, a valuable advantage in the face of limited budgets.

6.5 Reference

- https://itcraftapps.com/blog/estimating-mobile-app-development-costs-a-comprehensiveguide/
- https://www.businessofapps.com/app-developers/research/app-development-cost/
- https://www.octalsoftware.com/blog/mobile-app-architecture-guide
- https://theonetechnologies.com/blog/post/flutter-vs-react-native-vs-xamarin-top-cross- platform-mobile-app-development-framework
- https://surf.dev/mobile-app-development-frameworks/
- https://www.interaction-design.org/literature/article/native-vs-hybrid-vs-responsive-whatappflavour-is-best-for-you