****

**REPUBLIC OF CAMEROON**

PEACE-WORK-FATHERLAND

THE UNIVERSITY OF BUEA

P.O Box 63,

Buea,South West Region

Cameroon

Tel: (237) 3332 21 34/3332 26 90

Fax: (237) 3332 22 72

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**Dean: Agbor Dieudonne Agbor PhD**

**Vice-Dean : Nde D. Nguti, PhD**

**Faculty Officer : Mme Ekpombang Patience Taku**

**CEF440 – TASK1 : BASIC DEFINITIONS IN INTERNET PROGRAMMING AND MOBILE PROGRAMMING**

Date:

Presented by

**GROUP 4**

|  |  |  |
| --- | --- | --- |
| SN | NAME | MATRICULE NUMBER |
| 1 | DJOUMESSI NGUEFACK IVAN | FE21A174 |
| 2 | NJOYONG GODWILL | FE21A273 |
| 3 | TETUH WIBINAG ENOGONWEI | FE21A322 |
| 4 | DJOUMESSI DONGMO RONSARD CARNEGIE | FE21A172 |
| 5 | GATCHUINNE MOMO MURIELLE CYNTHIA | FE21A200 |

**Valery Nkemeni, PhD**

**Course Supervisor**

[What are native applications? 3](#_Toc162880802)

[**Advantages of native apps** 4](#_Toc162880803)

[Disadvantages of native apps 4](#_Toc162880804)

[**What are web applications?** 5](#_Toc162880805)

[Advantages of web apps 5](#_Toc162880806)

[Disadvantages of web apps 6](#_Toc162880807)

[**What are progressive web applications?** 6](#_Toc162880808)

[**What are hybrid mobile apps?** 9](#_Toc162880809)

[Advantages of hybrid apps 9](#_Toc162880810)

[Disadvantages of hybrid apps 10](#_Toc162880811)

[**Hybrid app vs. web app** 10](#_Toc162880812)

[**Native app vs. hybrid app** 11](#_Toc162880813)

[**Hybrid app vs. progressive web app** 11](#_Toc162880814)

[**Questions to ask before choosing a native, hybrid, or web app** 12](#_Toc162880815)

[Which type of application should your business pay a developer team to create? 13](#_Toc162880816)

[Summary of Differences: Progressive Web Apps, Hybrid Apps, and Native Apps 13](#_Toc162880817)

[ Singleton Method Design Pattern 22](#_Toc162880818)

[ Factory Method Design Pattern 22](#_Toc162880819)

[ Observer Method Design Pattern 22](#_Toc162880820)

[ Dependency Injection (DI) Method Design Pattern 22](#_Toc162880821)

[ Adapter Method Design Pattern 22](#_Toc162880822)

[ Strategy Method Design Pattern 22](#_Toc162880823)

[ **Composite Method Design Pattern** 23](#_Toc162880824)

[ Model View Presenter (MVP) Architecture 24](#_Toc162880825)

[ Model View View Model (MVVM) Architecture 24](#_Toc162880826)

[ VIPER Architecture 25](#_Toc162880827)

[User requirements focuses onwho you are targeting with your application. 28](#_Toc162880828)

[**Methods for Estimating Mobile App Development Costs** 30](#_Toc162880829)

[**Tips for Accurate Cost Estimation** 31](#_Toc162880830)

[Ensuring Clear Communication with the Development Team 31](#_Toc162880831)

[Taking into Account Unexpected Costs 31](#_Toc162880832)

[Regularly Revisiting and Revising the Budget 31](#_Toc162880833)

**Review and compare the major types of mobile apps and their differences**

What's An APP?

An app is software that allows you to exchange information with customers and help them complete specific tasks. The different types of applications differ in their development method and internal functionality. Web applications are delivered via a web browser. Users don't need to install them on their devices. Native apps, on the other hand, are designed for a specific platform or device type. The user must install the correct version of the software on the device of their choice. Hybrid apps are native, but have a web browser built in.

What are native applications?

When you think of an app, you likely picture a branded little icon on your smartphone’s screen.

In reality, mobile apps from an app store are just one type of app. They’re called native mobile apps.

Native apps developed for Android are [written in Java](https://www.upwork.com/resources/java-tips-app-development-project), while apps developed for iOS are written in Swift (you may find older iOS apps written in Objective-C).

Native mobile apps are typically faster and more reliable than hybrid or web apps, which lets them deliver a better user experience (UX).

They also let you interact with a device’s application programming interface (API) and internal hardware, granting your company’s app access to features like:

* Cameras
* User contact lists
* The device’s microphone
* Device location tracking

The ability to interact with a device’s operating system is a big reason many companies decide to develop native mobile applications.

However, native features come with larger upfront development costs that organizations with smaller budgets likely can’t handle.

Higher development costs are a guarantee if you intend for your application to be on more than one device, as this requires your business to create two mobile apps: one for [Apple’s App Store](https://www.apple.com/app-store/) and another for [Android’s Google Play Store](https://play.google.com/store/).

**Advantages of native apps**

Native apps are a popular choice for companies due to their performance and various features. A few reasons native apps are appealing in many situations include:

* Superior performance. Native applications tend to run smoothly, even when running heavier graphics.
* Platform-specific features. Creating your app for iOS or Android grants your business access to platform-specific features.
* Superior user interface. Native applications tend to outperform hybrid and web applications because they must meet platform-specific performance standards.
* App store visibility. Unlike web apps and hybrid apps, native apps offer your business greater visibility because they’re featured on app stores. This makes it easier for potential new users to discover your brand.

The principal advantage of native apps is that they optimize the user experience. They look and perform better because they are designed and developed specifically for that platform.

Disadvantages of native apps

. Some disadvantages you may discover when developing a native app include:

* Greater upfront costs. Expenses for native apps can be higher than hybrid and web apps, especially if your business intends to get your app on more than one platform. This may require your business to hire two development teams.
* Requires experienced app developers. If you intend to build an app for the Apple App Store, you’ll need to hire an app developer proficient with Swift. If you plan to get your app on the Android App Store, your app developer must know Java. Your team could attempt to build the app on their own, but there’s a considerable learning curve to these programming languages (and the resulting app may not be as polished or robust as one made by a professional).
* Start over with different platforms. If you want to build and launch an app on more than one platform, you must start again from both a design and development perspective for each platform.

**What are web applications?**

A [web application](https://www.upwork.com/resources/what-is-web-application-development) is more or less a website that looks and feels more like a mobile application. Unlike native applications, web apps can run on various internet browsers, like Chrome or Safari, and are written using JavaScript or HTML/CSS.

Web apps have a lower entry barrier than native apps due to their comparative simplicity and tend to be cheaper and easier to develop. The tradeoff is that web apps also tend to be slower and have less intuitive designs for their user interface (UI).

In addition to these downsides, Apple and Android app stores can’t feature web apps. This means your application may have less organic visibility. But what web apps lack in discoverability, they make up for in accessibility.

Even though users can’t find web apps on app stores, anyone can access them from any mobile device with a web browser. And these days, almost every device—from smartwatches to smart TVs—has web browser functionality.

Advantages of web apps

Many users love web apps because they’re easy to develop and maintain. Some reasons people choose to develop web apps include:

* Easy maintenance. Unlike native apps, web apps use a common code base across different platforms. If something goes wrong with the application, you don’t have to fix both Android and iOS apps.
* Easy accessibility. As long as your users use the right browser, they can access your web application without downloading anything to their devices.
* Less expensive. Web apps have a lower development cost than native apps. They’re also easier to make.
* Less user maintenance. With native apps, users typically have to manually update the app via the app store. On the other hand, web apps let your company update the app without the user lifting a finger.‍
* No app marketplace approval. Web apps aren’t on the app marketplace, so your business won’t have to go through any approval process to be visible to the public. This means you can release the app whenever you want and in whatever format you choose.

Disadvantages of web apps

Web apps are often easier to develop and maintain, but this doesn’t mean they’re a perfect solution in all scenarios. We cover a few disadvantages when developing or running an app for web browsers:

* Smaller scope. While web apps are simpler to make, they also lack native features. For instance, they can’t use device features like location tracking or push notifications.
* More complex UX. Web apps are only available on browsers. While this makes them more accessible, it also complicates the user experience because it takes more steps to get to your application.
* Less focus on your app. A native app covers your users’ entire screen, making it harder for them to get distracted and leave the app. Web applications lack this feature.‍
* Less speed. Web apps tend to be slower, less responsive, and less interactive than native apps.

**What are progressive web applications?**

A Progressive Web App (PWA) is a type of web application that uses modern web technologies to deliver an app-like experience to users. It aims to combine the best features of both web and native mobile applications.

PWAs are designed to be reliable, fast, and engaging, providing an immersive user experience similar to that of native apps. They can be accessed through a web browser, eliminating the need for users to download and install them from an app store.

Key features of Progressive Web Apps include:

1. Responsive Design: PWAs are built to be responsive and adaptive, offering a consistent experience across different devices and screen sizes.
2. App Shell: times and offline functionality. PWAs make use of an app shell model, which separates the core application infrastructure from the dynamic content. This allows for faster loading
3. Service Workers: Service workers are JavaScript files that run in the backgroun d and enable PWAs to work offline or with a poor network connection. They can cache data, handle push notifications, and perform background synchronization.
4. App Manifest: PWAs have a web app manifest, which is a JSON file that provides metadata about the application, such as its name, icons, and color scheme. This allows PWAs to be installed on a user's device and appear like a native app.
5. Push Notifications: PWAs can send push notifications to users, even when the application is not actively running. This helps in engaging users and providing timely updates.
6. Secure Connections: PWAs are required to be served over HTTPS to ensure data security and prevent unauthorized access.

The main advantage of PWAs is that they offer a seamless user experience across different platforms, without the need for separate development for each specific platform. They can be easily discovered through search engines and shared via URLs, making them highly accessible. Additionally, they eliminate the need for regular app updates since the latest version is always served from the web.

Advantages of Progressive Web Apps:

1. Cross-platform Compatibility: PWAs are designed to work on multiple platforms, including desktops, mobile devices, and tablets. They can adapt to different screen sizes, operating systems, and browsers, providing a consistent experience across platforms.
2. Offline Functionality: One of the significant advantages of PWAs is their ability to work offline or in areas with a poor network connection. By utilizing service workers and caching, PWAs can store and display content even when the user is offline, enhancing user engagement and productivity.
3. Discoverability: PWAs can be easily discovered through search engines, as they have URLs and are indexable by search engines. This allows users to find and access PWAs through web search, increasing their visibility and accessibility.
4. No App Store Dependency: Unlike native mobile apps, PWAs do not require distribution through app stores. Users can access them directly through a web browser, eliminating the need for app store approvals and updates. This gives developers more control over the distribution and update process.
5. Cost-Effectiveness: Developing a single PWA that works across platforms can be more cost-effective than developing separate native apps for each platform. With PWAs, developers can target a broader audience with less effort and cost.

Disadvantages of Progressive Web Apps:

1. Limited Native Functionality: While PWAs can provide many app-like features, they have limitations compared to native apps. They may not have access to all the hardware and software capabilities of a device, such as certain sensors or device-specific functionalities. However, with advancements in web APIs, these limitations are gradually being addressed.
2. Browser and Platform Support: Although modern web technologies are widely supported, not all browsers and platforms fully support all PWA features. Some features may have limited support or behave differently across different browsers or operating systems. This can create inconsistencies in the user experience.
3. Limited User Awareness: While PWAs can be indexed by search engines, users may not be aware that a particular website offers a PWA experience. Unlike native apps that are prominently featured in app stores, PWAs rely on effective marketing and promotion to gain user recognition and adoption.
4. Security Concerns: PWAs, like any web application, are susceptible to web-based security vulnerabilities, such as cross-site scripting (XSS) or cross-site request forgery (CSRF). However, by adhering to best security practices and utilizing secure connections (HTTPS), these risks can be mitigated.

‍

**What are hybrid mobile apps?**

As the name implies, [hybrid mobile apps](https://www.upwork.com/resources/should-you-build-a-hybrid-mobile-app) combine elements of web apps and native apps. Hybrid mobile apps can be installed on the device and run on a web browser, so they sit somewhere between native apps and web apps.

These apps are built in two parts:

1. The backend code
2. The native shell (which makes it downloadable on app stores)

[Progressive web apps](https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps) (PWA) are similar to hybrid apps. Both are a bit of a mix between native and web apps, but there are also some major differences.

For one, PWAs aren’t available on app stores, but users can still create an app icon for the PWA on their smartphone. They’re also cheaper to make, easily customizable and tend to offer better speed than hybrid apps in areas with slow internet.

‍

Advantages of hybrid apps

Hybrid apps are a popular choice for many businesses and organizations. If you develop a hybrid app, you’ll enjoy:

* Developer productivity. Hybrid apps can be built with common web technologies across both the front and back end, reducing development time.
* Cheaper but greater discoverability. Hybrid apps are simpler and have fewer upfront costs than native apps, and they can also be featured on app stores.
* Internal API access. Unlike web apps, hybrid apps can use device features like location tracking and push notifications.‍
* Cross-platform availability. Hybrid apps are built using a common codebase, so they can be used on both Android and iOS devices.

Disadvantages of hybrid apps

Hybrid apps provide several benefits, but there are also drawbacks to consider. Some obstacles you may face if you decide to develop a hybrid app include:

* Inconsistent user experience. Hybrid app performance depends on the user’s internet speed, meaning the UX for hybrid applications can be inconsistent.
* Learning curve. Hybrid apps aren’t built merely using JavaScript or CSS. They must integrate with hybrid app development frameworks like React Native, Ionic, or Cordova—all of which have a learning curve. Cordova is one of the primary tools for connecting to native software development kits (SDKs), which allows hybrid apps to use certain native features.

**Hybrid app vs. web app**

Perhaps you prefer the convenience of a web app, but you’d also like to have some of the features of a native app. Hybrid apps are similar to web apps in many ways but are not identical. Key differences include:

* Visibility. Hybrid apps can run in a web browser, but it’s also possible to feature them on the app store. This often gives them greater visibility than web apps.
* API access. Unlike web apps, hybrid apps can access a device’s push notifications and location tracking.
* Ease of development. Hybrid apps require knowledge of additional development frameworks besides JavaScript and CSS, making them more difficult to put together.
* Speed. Mobile web apps are sometimes slower and less responsive than native apps. Hybrid apps may offer faster performance, although the app’s functionality largely depends on the user’s internet speed.

**Native app vs. hybrid app**

Native apps are enticing for several reasons, but developing one can be quite a chore. A hybrid app might be a reasonable solution if you don’t want to take the time or spend the money on a native app. Like native apps, hybrid apps are visible in the app store and can provide API access to use location tracking and push notifications.

At the same time, you shouldn’t expect an identical experience from native and hybrid apps. Things might look different between each choice, including:

* User interface. The user interface on native apps is often more consistent than what you may experience on a hybrid app.
* Development. Hybrid apps use common web technologies, making them an easier project to take on for mobile app development teams who lack the experience or knowledge necessary to build a native app.
* Platform-specific features. Hybrid apps are accessible on iOS and Android devices. Unlike native apps, you can’t build platform-specific features into your app.

**Hybrid app vs. progressive web app**

1. Technology Stack:
   * Hybrid Apps: Developed using web technologies wrapped in a native container.
   * Progressive Web Apps: Developed using modern web technologies, accessed through web browsers without a native wrapper.
2. Distribution and Installation:
   * Hybrid Apps: Distributed through app stores, requiring installation.
   * Progressive Web Apps: Accessed directly through web browsers, no app store installation required.
3. Platform Compatibility:
   * Hybrid Apps: Can run on multiple platforms using a single codebase.
   * Progressive Web Apps: Inherently cross-platform, can run on any device with a modern web browser.
4. Native Integration:
   * Hybrid Apps: Can access device-specific features using plugins or APIs.
   * Progressive Web Apps: Have limited access to native device features compared to hybrid or native apps.
5. Offline Functionality:
   * Hybrid Apps: Can provide some offline functionality using caching and local storage.
   * Progressive Web Apps: Designed to work offline using service workers and caching.
6. User Discovery:
   * Hybrid Apps: Benefit from app store listings for user discovery.
   * Progressive Web Apps: Discovered through search engines and URLs, rely on marketing and promotion.

**Questions to ask before choosing a native, hybrid, or web app**

Now that you know more about the different types of apps, consider which choice might be the right fit for you. Consider the following additional questions as you weigh the pros and cons:

* How soon do you need a mobile app? Since native apps require more work, they may take longer to develop than a web or a hybrid app.
* What’s your budget for the app development project? You’ll probably spend less on a web or hybrid app than on a native app.
* What features do you want to include in your app? Native apps sometimes offer more extensive functions and new features. It’s important to decide which features you want to include.
* How will the app serve your business priorities? What outcome are you hoping to achieve from offering an app to your customers? This may dictate what type of app you build.
* Are you catering to a specific platform? Do you want to create something offering compatibility with any operating system or are you specifically marketing your app to iPhone or Android users?
* What kind of user experience do you want to provide? Offering a strong user experience will make your app more likely to grow. It’s possible for a web app to have a strong user experience, but native apps sometimes make this easier to provide.
* How important is offline connectivity? Your users can’t access a web app without an internet connection.
* How often will you make updates? Offering a native or hybrid app means users must visit the app store to install the new version.

Ready to build your own web, hybrid, or native mobile app? [Hire a mobile app developer](https://www.upwork.com/hire/mobile-app-developers/) on Upwork who can assist you with your software development.

You’ll save yourself from the time-consuming process of web development and still end up with a great app your customers love.

Which type of application should your business pay a developer team to create?

By now, you should have a pretty clear understanding of hybrid apps, native apps, web apps, and their differences.

A native app is probably the best choice for businesses with a large budget because this type of app delivers high performance and a consistent user experience.

A web application might be the most affordable option for startups or small businesses with a smaller budget.

A hybrid app may meet your needs if your business is somewhere in the middle—desiring visibility on the app marketplace but not wanting the hassle of building two applications for iOS and Android.

Regardless of which direction you go, aim to deliver the best user experience possible. Otherwise, all the time and money you devote to building the app will go to waste because your users won’t enjoy using it.

## Summary of Differences: Progressive Web Apps, Hybrid Apps, and Native Apps

|  |  |  |  |
| --- | --- | --- | --- |
| **Characteristics** | **Web application** | **Hybrid Application** | **Native app** |
| Usage | Users can access directly from a browser | Users need to install the app on the device of their choice | Users need to install the app on the device of their choice |
| Inner workings | Client code in the browser communicates with remote server-side code and databases | Client code and browser code encapsulated in a native shell or container | Client code written in technology and language specific to the device or platform on which it will be installed |
| Native device features | Not accessible | Accessible | Accessible |
| User Experience | Inconsistent and browser-dependent | Consistent and engaging | Consistent and engaging |
| Access | Limited by browser and network connectivity | One-step access with offline features | One-step access with offline features |
| Performance | Slower and less responsive | Faster, but can consume more battery | Performance can be optimized for the device |
| Development | Cost-effective and faster time-to-market | Cost-effective and faster time-to-market | Costly and slower time-to-market |
|  |  |  |  |

**2.Review and Compare Mobile App Programming Languages**.

**Introduction**

In this modern computing era, mobile app development continues to rapidly evolves and dynamic. Mobile programming languages have different criteria, strength and weakness that can impact the choice of a developer when it comes to choosing the best programming language. The choice of a programming language can impact the development process and the app’s outcome. Choosing the best programming language can facilitate coding and optimize app performance. Below I will provide you the best mobile programming languages.

**Best Native iOS Mobile Programming Languages**

1.Swift

Swift is a modern, open-source programming language developed by Apple for ios app development. It is known for its speed, safety features, expressive syntax and hence empower developer to create robust applications.

Advantages.

* **Safety and Performance:** Swift eliminates common programming errors through features such as optimal typing and automatic memory management hence resulting in a safer and more stable app.
* **Modern syntax:** Swift syntax is concise, expressive and user friendly making easier to read and write code.
* **Active Community:** Swift has a thriving developer community, ensuring regular update and numerous open source libraries and frameworks.

Disadvantage.

* **Platform Limitation:** Swift is primarily developed for iOS app development thus limited for cross- platform development.

2. **Objective C or C#**

Objective C was the primary language used for iOS and macOS app development before the introduction of swift Objective C is known for its dynamic runtime and extensive frameworks and libraries. C# provide a robust and familiar environment for creating powerful mobile app.

Advantages

* **Mature and Stable:** C# has been used for a long time and has a mature and stable ecosystem. It has been tested and refined for years making reliable for developing robust app.
* **Interoperability with C:** C# is a superset of c programming language which implies C libraries can be used to code. This provides developers with flexible and leverage existing C libraries to be part of C# projects.
* **Legacy Codebase:** C# is widely used in IOS app development before the introduction of swift. Working on a project with existing C# codebase using objective C makes it easier to maintain and extend the codebase without the need for complete rewrite.

Disadvantage

* **Syntax are complicated and tedious:** C# is an older language and compare to swift whose syntax resembles plain modern English, Objective C syntax is more tedious and difficult to write.

**Comparing objective C and Swift**

* **Performance:** Swift is faster compare to C#. This is why swift replaced objective C as an Apple’s native app development language.
* **Syntax:** C# syntax is complicated and long. It using many brackets, symbols and semicolons. Meanwhile swift is easy to learn and similar to English with few lines of codes
* **Time-Efficiency:** Swift is time efficient due to it concise syntax and reduces time spending in writing code. Swift has fast feedback to debug and faster compile time meanwhile C# is lengthy and complex.

**Best Native Andriod Mobile Programming Languages**

1**.Java.**

Java is a widely used mobile app programming language known for its “write once, runaway” philosophy. Java’s key features are object-oriented, garbage collection and extensive libraries. It’s stability, vast community support makes java the best choice for developer building scalable and reliable android application.

Advantages

* **Platform independency:** Java “write once, runaway” capability allows developers to build applications that can run on various platforms including android, desktop, web and embedded systems.
* **Robustness and Scalability:** Java’s strong typing and exceptional handling provide a robust framework for building large scale applications with minimal errors.
* **Broad community Support:** Java has a massive developer community, providing access to abundant resources, frameworks and libraries.

Disadvantage

* **Heavy Code:** Java’s syntax can be complex, resulting in long lines of codes compare to more concise language like kotlin.

2.Kotlin.

Kotlin is a modern ,statically –typed programming language known for its seamless interoperability with Java. It is designed to be concise, safe and interoperable offering features such as data classes and co-routines for asynchronous programming.

Advantages

* **Interoperability with Java:** Kotlin is fully interoperable with Java, enabling developer to use developer to adopt Kotlin.
* **Concise and expressive syntax.** Kotlin’s syntax is more expressive than Java and hence increasing developer productivity sine the codes are not complex as compare to Java.
* **Null Safety:** Kotlin includes null safety features that help eliminate null pointer exceptions, a common source of bugs in Java.

Disadvantages

* **Smaller Community than Java:** Although Kotlin’s community is rapidly growing it’s still smaller compare to Java.

**Comparing Java and Kotlin**

* **Null Safety:** Kotlin includes built-in null safety features, whereas Java does not. This makes Kotlin less prone to null pointer exceptions.
* **Extension Function:** Kotlin supports extension functions, allowing developers to add new function to existing classes. Java lack this features.
* **Syntax:** Kotlin has a more concise and expressive syntax compared to java. It reduces boilerplate code and provides a more readable and modern language structure.

**Best Cross-platform mobile programming Language**

1. Javascript (React Native).

React Native is a popular Javascript framework that allows developer to build Native mobile app for iOS and Android using a single codebase. It utilizes Javascript to creates reusable UI component that are translated to native code at runtime.

**Advantages**

* **Cross-platform development:** React Native enable developers to write code once and deloys it on both iOS and Android platform, saving time and resources.
* **Reusability:** React Native promotes code reusability, allowing developers to share a significant portion of their codebase between platforms. This streamlines development and maintenance efforts, as changes made to shared code automatically reflect on both platforms.
* **Large developer community and ecosystem:** JavaScript has a vast and active developer community, providing access to many resources, libraries, and frameworks. React Native benefits from this ecosystem, ensuring continuous support, updates, and contributions.

**Disadvantage**

* **Platform limitations:** Although React Native aims for cross-platform compatibility, certain platform-specific functionalities may require additional native coding or third-party libraries, adding complexity to the development process.

2.Flutter(Dart).

Flutter is an open-source UI to build natively compiled mobile, web and desktop application from a single codebase. Flutter utilizes the Dart programming language, also developed by Google, as its primary language.

**Advantages**

* **Cross-platform development**: Flutter allows developers to write code once and deploy it on multiple platforms, including iOS, Android, web, and desktop. This reduces development time , efforts and ressources.
* **Strong community and support**: Flutter has a growing and enthusiastic community, providing ample resources, packages, and third-party libraries.
* **Fast and customizable UI**: Flutter offers a rich set of pre-built UI components called widgets, enabling developers to create visually appealing and highly customizable user interfaces.

**Disadvantage**

* **Limited libraries and integrations:** Compared to more established languages like Java or Swift, the Flutter ecosystem may have fewer libraries and integrations available.

**Comparing Javascript and Flutter**

* **Speed:** Javascript is an interpreted language so it is lighter and faster.However ,Dart is much more faster compared to Javascript.
* **Ease to Use:** Javascript has been longer in the industry so it is mature and stable language .Dart is hard for developpers to find solution for specific problems .In terms of coding style and syntax it looks like java so Javascript is easier to use compare to Dart.
* **Support Typing:** Javascript gives best support for dynamic and duck typing.Compare to Javascript ,Dart is a more typer safer and it support both open and robust prototyping.
  1. Compare Mobile App Developoment Frameworks By Comparing In Their Key Features

1. Frontend Frameworks:

Mobile app development frameworks are the backbone of app creation, providing a structured and efficient way to build applications for various platforms, such as iOS, Android, and even the web.

Mobile app development frameworks are sets of pre-built tools, libraries, and best practices that simplify and standardize the app development process. These frameworks come in various flavors, each catering to specific development needs and project requirements.

They can be categorized in to the following:

Native App Development Frameworks

1. Android development frameworks
   1. Kotlin

Kotlin, endorsed by Google as an official Android development language, has gained popularity rapidly. It offers concise syntax, enhanced safety features, and seamless interoperability with Java, making it a compelling choice for modern Android app development.

1. **Programming Language**: Kotlin
2. **Performance:**

* Developers can reuse code across Android, iOS, web, desktop, and server-side while keeping native code if needed.
* Kotlin Multiplatform can be seamlessly integrated into any project. Developers can utilize platform-specific APIs while making the most of both native and cross-platform development.
* Thanks to [Compose Multiplatform](https://www.jetbrains.com/lp/compose-multiplatform/), a modern declarative cross-platform UI framework by JetBrains, developers have full code-sharing flexibility and the ability to share both the logic and the UI.
* There's no need to introduce a new language to your codebase when you already use Kotlin for Android. You can reuse your Kotlin code and expertise, which makes migrating to Kotlin Multiplatform less risky compared to other technologies.

1. **Cost And Time To Market:**

|  |  |  |
| --- | --- | --- |
| **Complexity Level** | **Average Cost** | **Development Time** |
| Simple | $30,000-$50,000 | 2 to 4 Months |
| Moderate | $60,000-$90,000 | 5 to 9 Months |
| High | $100,000-300,000 | 10 to 15 Months |

1. **UI-UX complexity:**

Modern Language for Android Development: Kotlin is a modern, concise, and expressive programming language that offers many features to streamline Android app development. It is fully interoperable with Java, making it easy for developers to transition from Java to Kotlin and leverage existing Java libraries and frameworks.

UI Development with Android UI Toolkit: When developing Android apps with Kotlin, developers typically use Android's UI toolkit, which includes XML layouts for defining the app's UI and Kotlin code for handling UI logic and interactions. Kotlin's concise syntax and powerful features, such as extension functions and null safety, enhance productivity and maintainability when working with Android UI components.

1. **community support:**

Even though this cross-platform mobile development framework is one of the youngest on our list, it has a mature community. In November 2023, JetBrains promoted it to [Stable](https://blog.jetbrains.com/kotlin/2023/11/kotlin-multiplatform-stable/). It's growing fast and is already making a distinct impression on today's market. Thanks to its regularly updated [documentation](https://www.jetbrains.com/help/kotlin-multiplatform-dev/get-started.html) and community support, you can always find answers to your questions. What's more, many [global companies and startups already use Kotlin Multiplatform](https://www.jetbrains.com/help/kotlin-multiplatform-dev/case-studies.html) to develop multiplatform apps with a native-like user experience.

**Mobile app examples:** McDonald's, Netflix, Forbes, Cash App, Philips.

1. **IOS development Frameworks**

Hybrid App Development Frameworks

1. Ionic

Ionic is an open-source framework for building cross-platform mobile apps using web technologies. It offers a library of pre-designed UI components, a powerful command-line interface (CLI), and extensive documentation. Ionic apps can run on iOS, Android, and the web.

1. **Programming Language**: HTML, CSS, JavaScript

Mobile apps example: T-Mobile, BBC (Children's & Education apps), EA Games.

1. **Cost And Time To Market:**
2. UI Complexity:
   1. Ionic Components: Ionic offers a rich library of UI components, including buttons, cards, lists, forms, navigation components, and more. These components are styled to provide a consistent and modern look across different platforms.
   2. Customization: Developers can customize the appearance and behavior of Ionic components using CSS or Sass (Syntactically Awesome Style Sheets), allowing for flexibility in UI design. Additionally, Ionic provides theming capabilities to create custom themes and styles for the app.
3. UX Complexity:
   1. Navigation: Ionic provides navigation components and patterns, such as tabs, side menus, and routing, to facilitate navigation within the app. Designing intuitive navigation structures and user flows is essential for a seamless user experience.
   2. Performance Optimization: Performance optimization is crucial for delivering a smooth and responsive user experience. Developers need to optimize UI rendering, minimize app startup time, and optimize network requests to ensure optimal performance across devices and platforms.
4. **Community Support**

There's constant activity on the Ionic Framework Forum, where community members exchange knowledge and help each other overcome their development challenges.

#### LIMITATIONS OF IONIC

Ionic was built on top of Apache Cordova, but [it has not been Cordova-based](https://ionicframework.com/blog/ionic-isnt-cordova-anymore/) since May 2021. The App Center service didn’t accept calls from the Cordova SDK starting from April 1, 2022. Therefore, Cordova is not the appropriate option for new projects, though it still can be used in production. In turn, Ionic can still be a choice for cross-platform app development.

Also, a significant limitation is a rapid decline in the popularity of Ionic, which means that finding engineers to develop on Ionic or support an existing app is becoming increasingly difficult.

1. React Native

Developed by Facebook, React Native is a popular choice for building mobile apps using the React JavaScript library. It enables developers to create near-native experiences for both iOS and Android while maintaining a single codebase.

* 1. **Programming Language**: JavaScript
  2. **cost and time to market:**

|  |  |  |
| --- | --- | --- |
| **Complexity Level** | **Average Cost** | **Development Time** |
| Simple | $30,000-$50,000 | 2 to 4 Months |
| Moderate | $60,000-$90,000 | 5 to 9 Months |
| High | $100,000-300,000 | 10 to 15 Months |

* 1. **ux-ui complexity:**
* Component-Based UI Development: React Native utilizes a component-based architecture, where UI elements are modularized into reusable components. This approach facilitates efficient UI development, enabling developers to create complex interfaces by composing smaller, reusable components.
* Native-Like User Experience: React Native provides access to native APIs and components, allowing developers to build apps with a native look and feel. With features like native navigation, animations, and gestures, React Native apps can deliver a seamless and intuitive user experience similar to that of native mobile applications.

**Community Support:**

As one of the most popular cross-platform app development frameworks, React Native has a large and strong community of developers who share their technical knowledge. Thanks to this community, you can get the support you need when building mobile apps with the framework.

* 1. **Performance**

Developers can see their changes in their React components immediately, thanks to the Fast Refresh feature.

One of React Native's advantages is a focus on the UI. React primitives render to native platform UI components, allowing you to build a customized and responsive user interface.

In versions 0.62 and later, integration between React Native and the mobile app debugger Flipper is enabled by default. Flipper is used to debug Android, iOS, and React native apps, and it provides tools like a log viewer, an interactive layout inspector, and a network inspector.

* 1. **Mobile app examples:** Microsoft’s Office, Skype, Xbox Game Pass,Meta’s Facebook, desktop Messenger, and Oculus

#### LIMITATIONS OF React native Script

Native Script has some bottlenecks, such as long-term testing and paid UI components. Also, since the Native Script community has not been very active lately, there are a small number of third-party libraries and plugins. This may impose additional limitations on development.

**Cross platform Mobile App Development Frameworks**

1. Flutter

Flutter, developed by Google, is an open-source UI software development kit for building natively compiled applications for mobile, web, and desktop from a single codebase. It’s known for its expressive and flexible user interface components.

* 1. **Programming Language**: Dart
  2. **Cost and Time to market:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Flutter app Complexity** | **App aspects defining complexity** | **Flutter development timeline** | **Average prices** |
| Simple flutter based app | Basic user interface, MVP features | 2-3 months | $40,000-$60,000+ |
| Medium flutter based app | Custom interface, advanced functionality | 3-6 months | $60,000-$120,000+ |
| Complex flutter based app | Bespoke interface, advanced features of high complexity | 6-14 months | $120,000-$200,000+ |

* 1. **UX-UI Complexity:**
     1. Versatile UI Design: Flutter offers a wide range of customizable widgets and layout options, allowing developers to create visually rich and complex user interfaces effortlessly.
     2. Optimized User Experience: With built-in support for animations, gestures, navigation, and performance optimization, Flutter enables developers to deliver smooth, responsive, and engaging user experiences across various platforms.
  2. **Community Support:**
     1. Flutter has a relatively active community of users around the world and is widely used by many developers. According to the [Stack Overflow Trends](https://insights.stackoverflow.com/trends?tags=flutter%2Creact-native), the usage of Flutter has been trending up over time, based on the increasing use of the corresponding tag.
  3. **Performance:**
     1. Flutter's hot reload feature allows you to see how your application changes as soon as you modify your code, without you having to recompile it.
     2. Flutter supports Google's Material Design, a design system that helps developers build digital experiences. You can use multiple visual and behavioral widgets when building your app.
     3. Flutter doesn't rely on web browser technology. Instead, it has its own rendering engine for drawing widgets.

**Mobile App Examples**: ebay, Alibiba, Google Pay, BlaBlaCar.

1. FLUTTER’S LIMITATIONS

Flutter perfectly covers about 90% of business cases, especially projects that focus on a pleasant design and great UX for end users. However, in cases of native-dependent-functionality as core features, augmented reality development, and game development with 3D elements, its capabilities may not be sufficient and it is more effective to resort to native

Android and iOS application development.

1. Xamarin(.NET MAUI)

Xamarin, a Microsoft-owned platform, allows developers to build cross-platform apps using C# and .NET. It offers a wide range of libraries and tools for building native user interfaces and accessing platform-specific APIs.

**Programming Language**: C#

**Cost And Time to Market:**

|  |  |  |
| --- | --- | --- |
| **Complexity Level** | **Average Cost** | **Development Time** |
| Simple | $30,000-$50,000 | 2 to 4 Months |
| Moderate | $60,000-$90,000 | 5 to 9 Months |
| High | $100,000-300,000 | 10 to 15 Months |

UI Complexity:

* + Xamarin.Forms provides a streamlined approach to creating cross-platform user interfaces using XAML or C#, offering a wide range of built-in controls and layouts.
  + Custom renderers allow for additional customization of UI elements, but may increase complexity, especially when dealing with multiple platforms.

UX Complexity:

* + Navigation structures and performance optimization are crucial aspects of UX design in Xamarin.
  + Developers need to ensure intuitive navigation flows and optimize performance to deliver a smooth and responsive user experience across platforms.

**Performance:**

* .NET MAUI provides cross-platform APIs for accessing native device features, like the GPS, accelerometer, and battery and network states.
* There's a single project system which is enabled using multi-targeting to target Android, iOS, macOS, and Windows.
* With the support for .NET hot reload, developers can modify their managed source code while the app is running.

**Community Support:** Even though .NET MAUI is still a relatively new framework, it has already gained traction among developers and has an active community on Stack Overflow and Microsoft Q&A.

**Mobile app examples:** NBC Sports Next, Escola Agil, Irth Solutions.

1. Native Script

**Programming Language**: JavaScript, TypeScript

**Mobile app examples**: Daily Nanny, Strudel, Breethe.

**Performance:**

* NativeScript allows developers to easily access native Android and iOS APIs.
* The framework renders platform-native UIs.
* Apps built with NativeScript run directly on a native device without relying on WebViews, a system component for the Android OS that allows Android applications to show content from the web inside an app.
* NativeScript offers various plugins and pre-built app templates, eliminating the need for third-party solutions.

UI Complexity:

* 1. Native UI Components: NativeScript provides access to native UI components for building the user interface of mobile applications. Developers can use XML markup or JavaScript/TypeScript to define UI layouts and components, leveraging platform-specific styling and behavior.
  2. Custom UI Components: NativeScript allows developers to create custom UI components using JavaScript/TypeScript and CSS, providing flexibility in UI design and functionality. Custom components can be reused across the application, but their implementation may increase development complexity.

1. UX Complexity:
   1. Navigation and Flow: Designing intuitive navigation structures and user flows is essential for a positive user experience. NativeScript offers navigation patterns such as stack navigation, tab navigation, and side drawer navigation to facilitate seamless navigation within the app.
   2. Performance Optimization: Performance optimization is crucial for delivering a smooth and responsive user experience. Developers need to optimize UI rendering, minimize app startup time, and optimize network requests to ensure optimal performance across platforms.

Community Support :NativeScript is based on well-known web technologies like JavaScript and Angular, which is why many developers choose this framework. Nevertheless, it's usually used by small companies and startups.

#### LIMITATIONS OF MAUI

Despite all the features, .NET MAUI is still a new technology that lacks some essential controls.Also, UX is not MAUI’s strong point. Therefore, the technology is suitable for projects that focus on functionality rather than appearance and user experience.

**Backend development Frameworks**

1. **Django**

Written primarily for Python Developers, the DRY (DON'T REPEAT YOURSELF) principal [Django](https://www.djangoproject.com/) supports won't let you reuse existing code modules at any cost.

Use its backend customizations & scalable programs to create mobile/ website applications that can cater to many client requests at a high speed of development.

Its excellent REST API support allows different content management systems to integrate, automate, and improve tasks of variable complexity. A few features of Django are

* Highly robust & secure multifactor authentication
* Middleware services that simplify content management
* Flexible & seamless integration with 3rd party applications
* A risk-free community supports
* A superb development documentation

1. **ExpressJS**

The popularity of [ExpressJS](https://expressjs.com/) JavaScript framework is gaining momentum across every sphere worldwide. Being an unopinionated part of the MEAN & MERN stack, use its hybrid app development flexibility for all user requests.

Make your business expressive with the POST, GET, DELETE, & PUT requests this JS framework handles at higher flexibility.

Take a look at the features/ functionalities this assertive framework intuitively entertains.

* Same performance behavior for client & server-side routing
* Middleware functions with higher response time
* Routable Debugging modules whose request flow logs info. better
* Extensive Utility templates
* HTTP methods that effortlessly reduce programming complexities

1. **Laravel**

It is an absolute and scalable server-side framework well-versed in every in-and-out data manipulation, which iteratively rules out unprofessional app research semantics.

Spotify, X(Formerly Twitter), & Udemy are using [Laravel](https://laravel.com/) as their backend framework and heading towards excellence with its elegant & expressive syntax.

Let's get quickly started with learning about Laravel features in short: -

* Its modern & secure CLI (Command Line Interface) works brilliantly, even at low resolutions.
* A non–trivial database migration
* Extensive libraries & API support
* Intact and innovative security fulfilling every small & big business requirement

1. **Mobile Applications Architecture and design patterns.**

**Introduction**

Design patterns are reusable solutions to common software development problems. They have had a significant impact on software development, including mobile app development. The implementation of mobile apps has established some proven models and standards to overcome the challenges and limitations of mobile app development.

Most mobile applications were built with low code and were not based on architecture. Mobile app development with the right design patterns can effectively integrate user interfaces with data models and business logic. This will affect the quality of your source code.

There are very few architectural design patterns available for mobile development.

**a. Monolithic Architecture**:

* In a monolithic architecture, all components of the app are tightly coupled within a single codebase.
* This simplicity can make development and testing easier, but it can lead to challenges in scaling and maintenance.

**b. Microservices Architecture:**

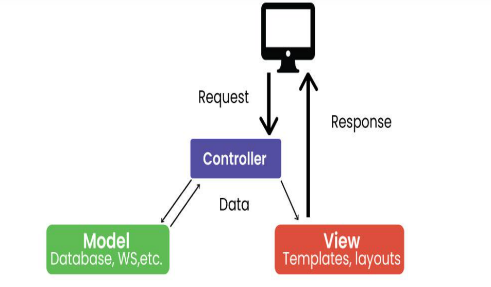
* Microservices architecture involves breaking the app into smaller, independent services that communicate with each other via APIs.
* It promotes flexibility, scalability, and easier maintenance but can introduce complexity.

**c. Layered Architecture:**

* A layered architecture divides the app into distinct layers, such as presentation, business logic, and data.
* This approach simplifies management and separation of concerns.

**d. Model-View-Controller (MVC):**

* MVC is a design pattern that separates the app into three interconnected components: the model (data), the view (UI), and the controller (business logic).
* It's widely used for building mobile apps, especially in iOS and Android development.

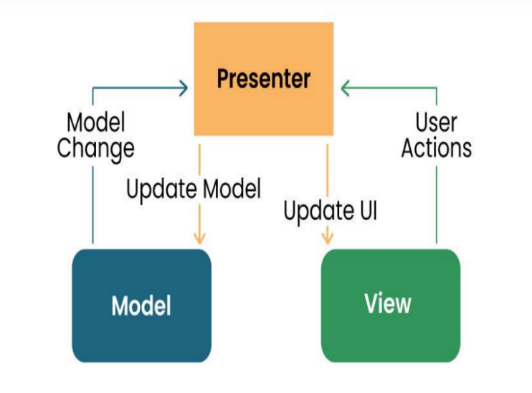


**Model**: Represent Represents application data and business logic.

**View:** Displays data to the user.

**Controller:** Processes user input and controls data flow between Model and View.

**e. Model-View-Presenter (MVP):**

* MVP is a variation of MVC where the presenter acts as an intermediary between the view and the model.
* It enhances testability and separation of concerns.

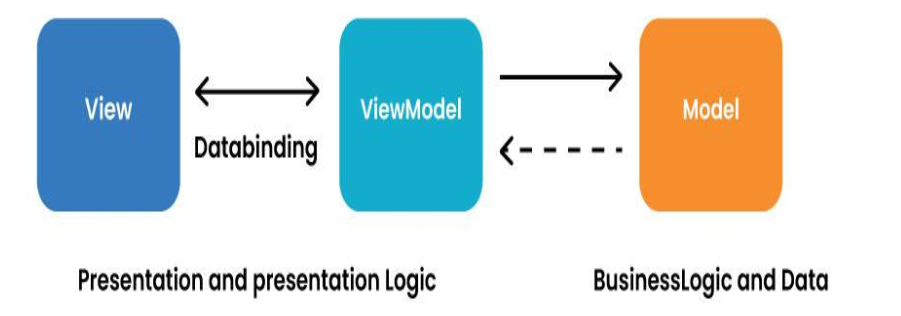
**Model:** Manages data and business logic.

**View**: Represents the user interface.

**Designer:** Acts as an intermediary processing user input and updating the View and Model.

**f. Model-View-ViewModel (MVVM):**

* MVVM is a design pattern popular in Android development.
* It separates the UI (View), business logic (ViewModel), and data (Model) into distinct components.



**Model:** Represents data and business logic.

**View:** Represents the user interface.

**ViewModel:** Acts as an interface between the Model and the View, which contains the reference logic.

**5-HOW TO COLLECT AND ANALYSE USER REQUIREMENTS FOR MOBILE APP DEVELOPMENT**

**Introduction**

According to the software development life cycle the first phase of this process is the requirement gathering phase followed by requirement analysis phase. These phases are very essential for the overall efficiency of a software and acts as a blue print for quality assurance.

Requirement gathering can be done using different methods such as

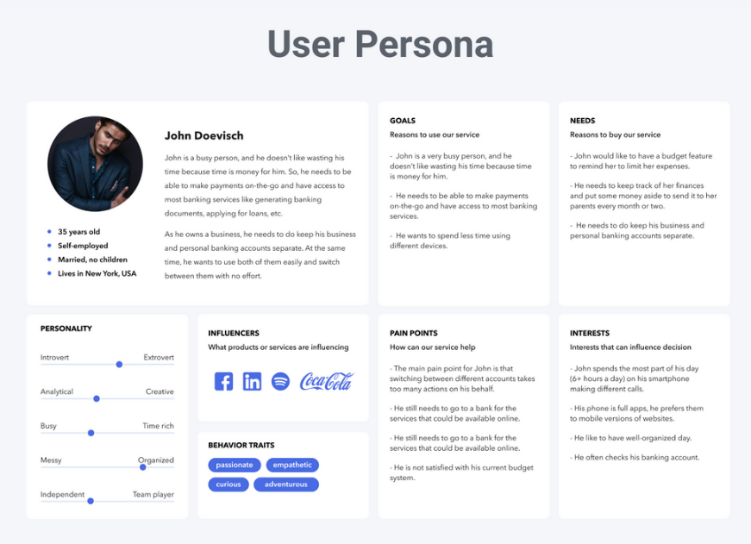
* Focus group meetings
* Surveys
* By examining pre-requisite data to bring about statistical facts
* Questionnaires

All these requirements should be gotten from the targeted market, the target market could be a group of people such as a company or a research institute, a particular geographical location such as Africa or maybe Cameroon, maybe a specific gender, or a specific group age.

User requirements focuses onwho you are targeting with your application.

* What are your target audience’s age group, lifestyle, and social background?
* Which geographical regions and languages are you targeting?
* What issues are your potential users facing that your app can solve?
* With which requests are they turning to your app?
* Which device types (smartphone, tablet, etc.) and OS (Android, iOS) do your potential users prefer?
* What are the parameters they value a lot in a digital product (e.g., simplicity, unusual design, wearables integration)

Based on your input, we usually create a user persona – these are imaginary users of your future app. We describe the smallest details about them, from work life to hobbies. Such methodology helps us have a clear understanding of who we are designing an app for and what they would be happy to see.



After the requirement gathering phase these requirements have to be analyzed. This analysis is done by the business analyst although some institutes have a requirement analyst. Analysis is very important as it helps:

* Capturing user stories or use cases
* Defining system constrains and interfaces
* Prioritized requirements based on their importance and feasibility
* Indicate a measurement of the total budget
* Outline functional and non-functional requirements

1. **Functional requirements:**

Functional requirements describe the specific functionalities, features and behaviors that a software system must exhibit in other to fulfil the needs of the user.

It typically answers questions like what should the system do?

Examples of this requirements include

* The system should allow users to create accounts and login
* The system should allow the prices of each item in a shopping cart

1. **Non-functional requirements:**

Also known as quality attributes or constraints, define the characteristics and qualities that the software system should possess. They are not directly related to specific functionalities but rather focus on aspects of the system’s performance, usability, security, scalability and other qualities that impacts its overall effectiveness. Examples include:

* Security
* Reliability
* Usability

**6. study how to estimate mobile app development cost**

**Methods for Estimating Mobile App Development Costs**

**Fixed Price Model**: Under this model, [itCraft](https://itcraftapps.com/about-us/) estimates the total cost of the project based on the defined requirements and specifications before the development starts. This method is most suitable for small projects with clear, unchanging requirements. The main advantage is that the client knows the exact cost upfront, helping with budget planning. However, it lacks flexibility for changes or additions once the project is underway.

**Time and Material Model**: This model is based on the actual time and resources utilized on the project. Clients are billed according to the hours worked by the development team. This approach is suitable for projects where requirements are expected to evolve or are not fully defined at the beginning. It offers high flexibility as changes can be made during the development process, but the final cost might exceed the initial estimates if the project scope expands or unexpected challenges occur.

**Dedicated Team Model**: In this model, itCraft provides a dedicated team of professionals who work exclusively on your project. The cost is based on the team size and composition, and the time they spend on the project. This model is ideal for long-term projects where requirements might change over time. It gives you greater control over the development process and ensures that the team is fully invested in your project.

At itCraft, we understand that each project is unique. Therefore, we offer these different [pricing](https://itcraftapps.com/mobile-app-development-cost/) models to cater to the specific needs and requirements of each client. Our main goal is to deliver high-quality mobile app solutions in a cost-effective and transparent manner. Before deciding on a pricing model, we thoroughly discuss the project with the client to ensure that the chosen model aligns with their project goals and budget.

## **Tips for Accurate Cost Estimation**

### Ensuring Clear Communication with the Development Team

Open and continuous communication with your **development team is critical for accurate cost estimation.** As a client, you should be clear and precise about your requirements, expectations, and budget. At itCraft, we value transparency and encourage clients to discuss their project thoroughly with us. This helps in understanding the project scope better, eliminating misunderstandings, and providing an accurate estimate.

### Taking into Account Unexpected Costs

No matter how well you plan, there will always be some unforeseen expenses during the app development process. These could be due to sudden changes in requirements, extra time spent on bug fixing, or additional features requested by users after launch. Therefore, it’s wise to set aside a contingency fund as part of your budget. This prepares you for unexpected costs and ensures that the development process isn’t hampered due to budget constraints.

### Regularly Revisiting and Revising the Budget

Mobile app development is a dynamic process, and costs can fluctuate based on a variety of factors. For example, additional features may be needed, or development could take longer than expected. It’s important to revisit and revise your budget periodically, keeping track of the actual spend against the estimated cost. This will help identify if the project is on track financially, and if not, corrective actions can be taken early.

Refences:

1. https://www.cleveroad.com/blog/flutter-app-development-cost/
2. <https://kotlinlang.org/docs/cross-platform-frameworks.html#flutter>
3. https://www.sculptsoft.com/choosing-the-right-mobile-app-development-frameworks/
4. https://appetiser.com.au/blog/android-app-development-cost/
5. https://quokkalabs.com/blog/which-backend-frameworks-are-impacting-mobile-web-app-development-immensely/
6. [**https://www.geeksforgeeks.org/design-patterns-for-mobile-development/**](https://www.geeksforgeeks.org/design-patterns-for-mobile-development/)
7. **https://www.linkedin.com/pulse/understanding-mobile-app-architecture-comprehensive-singh-dave--ukyzf/**