**UNIVERSITY OF BUEA**



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**DERPARTMENT OF COMPUTER ENGINEERING**

**CEF440: Internet Programming and Mobile Programming**

**COURSE INSTUCTOR: Dr. Valery Nkemeni**

**Task 1: Mobile App Development Process**

**BY GROUP 3**

**TABLE OF CONTENTS**

[INTRODUCTION 2](#_Toc14820)

[WHAT IS A MOBILE APPLICATION? 3](#_Toc29232)

[How are mobile apps built? 3](#_Toc7283)

[How does a mobile app work? 4](#_Toc39)

[Advantages of mobile apps 4](#_Toc22979)

[Disadvantages of mobile apps 4](#_Toc31288)

[Types of mobile applications 5](#_Toc7857)

[Comparison between the different types of mobile applications (google.com) 6](#_Toc15810)

[When to Choose Each: 7](#_Toc15111)

[WHAT ARE MOBILE APP PROGRAMMING LANGUAGES? 8](#_Toc1854)

[Types of mobile application programming language 8](#_Toc10231)

[Comparism between the different Mobile Programming Languages 9](#_Toc2828)

[DEVELOPMENT FRAMEWORK 10](#_Toc3682)

[Types of Software Frameworks 10](#_Toc5918)

[Features of Software Frameworks 10](#_Toc10149)

[Use Cases 10](#_Toc3341)

[Advantages and Disadvantages 11](#_Toc4688)

[MOBILE APP ARCHITECTURES AND DESIGN PATTERNS 12](#_Toc12994)

[Definition 12](#_Toc3756)

[Importance of a Good Mobile Application Architecture 12](#_Toc6494)

[The Key Elements in Mobile Application Architecture Designing 12](#_Toc23399)

[The Different Layers of Mobile App Development Architecture 13](#_Toc30369)

[What are Real-world Mobile App Architecture Examples? 14](#_Toc10721)

[Let’s check out different modern app architectures. 14](#_Toc18051)

[Key Attributes of a Result-Driven Mobile App Architecture 16](#_Toc15851)

[What are Mobile App Architecture Best Practices? 17](#_Toc24190)

[How to Choose a Good Mobile Application Architecture? 18](#_Toc23632)

[Design Patterns for Mobile Development 19](#_Toc10343)

[Important Topics for Mobile Development Design Patterns 19](#_Toc1769)

[Architectural design patterns 19](#_Toc10435)

[Design pattern 21](#_Toc14854)

[REQUIREMENTS ENGINEERING PROCESS 24](#_Toc21748)

[What is Requirements Engineering? 24](#_Toc15991)

[Requirements Engineering Process 24](#_Toc8123)

[Tools Involved in Requirement Engineering 27](#_Toc31862)

[Advantages of Requirements Engineering Process 27](#_Toc5857)

[Disadvantages of Requirements Engineering Process 28](#_Toc24381)

[MOBILE APP DEVELOPMENT COST ESTIMATION 29](#_Toc769)

[I. Understanding App Development Costs: A Breakdown 29](#_Toc5659)

[II. Factors influencing app development costs 30](#_Toc10036)

[Traditional Mobile App Development - Coding 30](#_Toc9529)

[No-Code Development 30](#_Toc10590)

[Low-Code Development 31](#_Toc25224)

[CONCLUSION 32](#_Toc26030)

[REFERENCES 32](#_Toc4605)

# INTRODUCTION

Mobile app development is a rapidly evolving field that plays a crucial role in modern digital solutions. The process involves selecting the right app type (native, hybrid, or progressive web app), programming language, and development framework to ensure optimal performance and user experience. Additionally, developers must consider architectural design patterns, requirement engineering, and cost estimation to build scalable, efficient, and cost-effective applications. This task explores these key aspects, providing insights into their differences, advantages, and best practices in mobile app development.

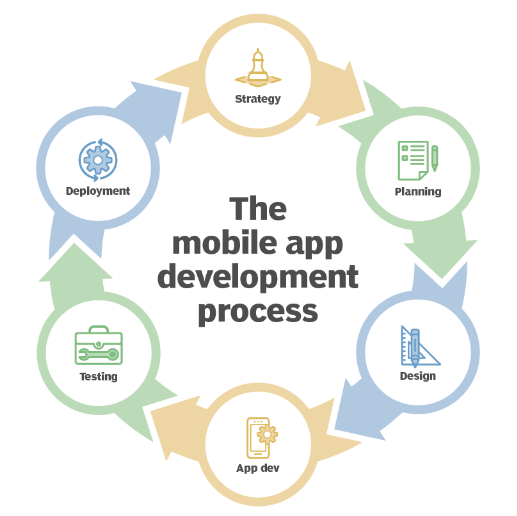
# WHAT IS A MOBILE APPLICATION?

A mobile app (or mobile application) is a software application developed specifically for use on small, wireless computing devices, such as [smartphones](https://www.techtarget.com/searchmobilecomputing/definition/smartphone) and [tablets](https://www.techtarget.com/searchmobilecomputing/definition/tablet-PC), rather than desktop or laptop computers.

Mobile apps are sometimes categorized according to whether they are web-based or [native apps](https://www.techtarget.com/searchsoftwarequality/definition/native-application-native-app), which are created specifically for a given platform. A third category, [hybrid apps](https://www.techtarget.com/searchsoftwarequality/definition/hybrid-application-hybrid-app), combines elements of both native and web apps.



## How are mobile apps built?



Mobile apps are built using a variety of programming languages and frameworks, and they can be downloaded and installed from app stores such as the Apple App Store or [Google Play](https://www.techtarget.com/searchmobilecomputing/definition/Google-Play-Android-Market)store.

Mobile apps are designed to provide a wide range of functions and services and with consideration for the demands, constraints and capabilities of the devices they're built for. For example, a [gaming](https://www.techtarget.com/whatis/definition/gaming) app might take advantage of the [iPhone's](https://www.techtarget.com/searchmobilecomputing/definition/iPhone) accelerometer.

## How does a mobile app work?

Mobile apps are designed to run on specific mobile operating systems such as [iOS](https://www.techtarget.com/searchmobilecomputing/definition/iOS), [Android](https://www.techtarget.com/searchmobilecomputing/definition/Android-OS) and Windows Phone. When a mobile app is downloaded and installed on a device, it is stored in the device's memory and is launched using the device's operating system.

When a user opens a mobile app, the app communicates with the device's [operating system](https://www.techtarget.com/whatis/definition/operating-system-OS) and other built-in software components to access the device's hardware and services such as the camera, [GPS](https://www.techtarget.com/searchmobilecomputing/definition/Global-Positioning-System) and internet connection. The app then uses this information to provide its specific functions and services to the user.

## Advantages of mobile apps

There are numerous advantages to using mobile apps, namely:

1. ****Convenience.**** Mobile apps can be downloaded and installed on a device, allowing users to access the app's functions and services at any time, from anywhere.
2. ****Personalization.**** Mobile apps can be customized to meet the specific needs of individual users, providing a personalized experience.
3. ****Offline access.**** Many mobile apps can be used offline, providing access to important information and features even when an internet connection is not available.
4. ****Push notifications.**** Mobile apps can send [push notifications](https://www.techtarget.com/searchmobilecomputing/definition/push-notification) to users, providing real-time updates on important information and events.

## Disadvantages of mobile apps

However, there are some disadvantages and considerations to keep in mind when using mobile apps. These include:

1. ****Limited functionality.**** Mobile apps are designed to provide specific functions and services, and they might not be able to provide the same level of functionality as a desktop application.
2. ****Limited compatibility.**** Mobile apps are designed to run on specific mobile operating systems, and they might not be compatible with all devices.
3. ****Security concerns.**** Mobile apps can access sensitive information on a device, and they might not have the same level of security as a desktop application.
4. ****Limited updating capability.**** Some mobile apps might not be easily updated, and users might need to download a new version of the app to access the latest features and [bug](https://www.techtarget.com/searchsoftwarequality/definition/bug) fixes.

While mobile apps offer convenience, personalization and offline access, they also have limitations in terms of functionality, compatibility, security and updating capability. Business software buyers should consider the advantages and disadvantages of mobile apps when deciding on a distribution method for their software.

As the mobile market continues to grow, mobile apps are becoming an increasingly popular way for businesses to engage with customers and employees, providing a convenient, personalized and secure way to access important information and perform tasks on the go.

## Types of mobile applications

There are ****different types of mobile applications****. Members of the [mobile app development](https://www.bitstudios.com/service/mobile-app-development-services/) community agree that there are four types. They are Native apps, hybrid apps, [web apps](https://www.bitstudios.com/service/web-application-development-services/), and progressive web apps. Let us tackle each of these in the following sections.

1. **Native apps**

These are designed for specific mobile operating systems like Android, IOS, or Windows. Here, Android is developed using Java and IOS uses Swift or Objective C. They are published on Play Store or App Store.

**Examples:** Google Maps, Spotify, Telegram, WhatsApp, etc...

**Technologies used:** C++, Java, Kotlin, Objective-C, Python, React Native, Swift.

**Advantages:**

They have best performance.

More secure.

These are user interactive.

They have access to full features of device.

**Disadvantages:**

They are developed only on single platform.

These are expensive to maintain.

1. **Web Apps**

Mobile web applications are optimized for mobile devices and they are accessible through a web browser on a smartphone. These are developed using HTML/CSS/Javascript. They runs with the help of web browser applications like chrome, safari, Firefox etc.

**Examples:** Amazon, Canva, Netflix, Walmart etc.

**Technologies used:** C#, C++, HTML, CSS, Javascript, Java, Perlin, PHP, Ruby, Python, Typescript.

**Advantages:**

It is easy to development.

Easy to maintain.

These are cheaper than Native and Hybrid apps.

They works on all platforms.

**Disadvantages:**

These are device browser dependent.

Poor user experience.

Slow performance.

These are less interactive with device utilities.

1. **Hybrid apps**

Hybrid apps combine elements of both native and web apps, using a single codebase to run on multiple platforms. These are deployed on container that uses mobile WebView object.

**Examples:** Facebook, Gmail, Instagram, LinkedIn, Twitter, Uber, Yelp.

**Technologies Used:** HTML5, CSS, Flutter, JavaScript, Appy Pie, AppMakr, Ionic, Objective-C, Swift, React Native, Wrap-Up.

**Advantages:**

These are cheaper to maintain.

These are easy to develop.

They have access to features of device.

Faster development due to single codebase.

**Disadvantages:**

These are slower as compared to native apps.

Less interactive than native apps.

1. **Progressive Web Apps (PWA)**

PWAs are web apps that provide a native app like experience. These are designed and developed similar to web apps. These apps take support of services workers. These are published on play store or app store.

**Examples:** Pinterest, Starbucks, Adidas, Tinder, Trivago.

**Technologies used:** AngularJS, CSS, HTML5, JavaScript, Lighthouse, Polymer, PWA library, VueJS, WebAssembly.

**Advantages:**

They provides fast and responsive performance than web pages.

These are easy to maintain.

These are cheaper than Native and Hybrid apps.

Works on all platforms.

They can work offline.

**Disadvantages:**

They have limited hardware and software support.

Technical options such as Bluetooth or NFC cannot be used.

## Comparison between the different types of mobile applications (google.com)

Comparison between **Native Apps, Web Apps, Hybrid Apps, and Progressive Web Apps (PWAs):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Feature** | **Native Apps** | **Web Apps** | **Hybrid Apps** | **Progressive Web Apps (PWAs)** |
| **Definition** | Applications built specifically for a platform (Android/iOS) using native languages. | Web-based applications accessed via browsers, running on any device with an internet connection. | A mix of native and web apps, using a web view wrapped in a native shell. | Advanced web applications that provide native-like experience in browsers. |
| **Development Languages** | Swift, Kotlin, Java, Objective-C | HTML, CSS, JavaScript | HTML, CSS, JavaScript with frameworks like Ionic, React Native | HTML, CSS, JavaScript with service workers and APIs |
| **Performance** | High performance, optimized for hardware and OS | Lower performance, depends on browser capabilities | Moderate performance, better than web but not as good as native | Good performance, improved caching and offline capabilities |
| **Access to Device Features** | Full access to device hardware (Camera, GPS, Sensors, etc.) | Limited access via browser APIs | Moderate access through plugins like Cordova or Capacitor | Limited access, but improving with modern web APIs |
| **Installation** | Installed via App Store (iOS) or Play Store (Android) | No installation needed, accessed through a browser | Installed from app stores but works using a web backend | Can be installed on the home screen without a store |
| **Offline Functionality** | Fully functional offline | Requires internet connection | Some offline capabilities if cached | Works offline using service workers |
| **Update Process** | Requires manual update via App Store/Play Store | Updates automatically as it is web-based | Updates automatically if changes are made on the web part | Updates automatically in the background |
| **Development Cost** | High (separate development for iOS & Android) | Low (single codebase) | Moderate (single codebase but requires native shell) | Low to moderate (single codebase with additional optimizations) |
| **User Experience (UX)** | Best UX, optimized for platform | UX depends on browser speed and responsiveness | Near-native experience but can be inconsistent | Smooth and responsive UX, close to native |
| **App Store Presence** | Available in Play Store/App Store | Not available, accessed via URL | Available in stores | Not available in stores but can be added to the home screen |

## When to Choose Each:

**Native Apps:** When you need high performance, full device access, and the best user experience. (e.g., Games, High-end productivity apps)

**Web Apps:** When you want easy access across multiple devices without installation. (e.g., News websites, Blogs)

**Hybrid Apps:** When you need a balance between cost, performance, and device features. (e.g., Social media apps, E-commerce apps)

**PWAs:** When you want a web-based solution with offline functionality and better performance. (e.g., E-commerce platforms, News portals, Small business apps)

# WHAT ARE MOBILE APP PROGRAMMING LANGUAGES?

Mobile app programming languages are specialized coding languages used to develop applications for mobile devices like smartphones and tablets. These languages determine how an app functions, its performance, and its compatibility with different operating systems (iOS, Android, etc.).

## Types of mobile application programming language

1. **Swift:**

Swift is a powerful and intuitive programming language for all Apple platforms, with a concise-yet-expressive syntax and modern features developed uniquely for the modern iOS development process. The main advantages of Swift are the ability to enhance the readability, speed, and performance of the system. In comparison to Python, it boosts the speed of the system up to eight times. Swift can also reduce errors in the development process and ensure a reliable and trendy mobile application.

1. **Kotlin:**

Kotlin is a statically typed, object-oriented programming language that is interoperable with the Java virtual machine (JVM), Java Class Libraries and Android . it is used in High-performance Android apps (e.g., Netflix, Pinterest).The Kotlin programming language was originally designed to improve the [Java](https://www.theserverside.com/definition/Java) programming language and is often used in conjunction with Java. Despite being the preferred development language of Android, Kotlin's interoperability with Java has led it to be used with many application types.

1. **Java:**

Java is an extremely transferable programming language used across platforms and different types of devices, from smartphones to smart TVs. It's used for creating mobile and web apps, enterprise software, Internet of Things (IoT) devices, gaming, big data, distributed, and cloud-based applications among other types. It is characterised by High quality learning resources, Inbuilt functions and libraries and High-quality development tools.

1. **Dart:**

Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform paired with a [flexible execution runtime platform](https://dart.dev/overview" \l "platform) for app frameworks.Dart is a versatile programming language developed by Google, primarily used for front-end development. Its main use case is in Flutter development (Cross-Platform Apps), Google's UI toolkit for building cross-platform mobile, web, and desktop apps from a single codebase.Example: Apps like Google Ads, eBay Motors, and BMW use Flutter, powered by Dart.

1. **C#:**

C# is a cross-platform general purpose language that makes developers productive while writing highly performant code.C# programs can run on many different devices, from Internet of Things (IoT) devices to the cloud and everywhere in between.It is Used before Swift was introduced in 2014· Used before Swift was introduced in 2014 it has a More complex syntax compared to Swift its Still supported but gradually being phased out

1. **JavaScript:**

JavaScript, originally designed for web development, has evolved into a full-stack language that can also be used for mobile app development. With the rise of frameworks like React Native, Ionic, and NativeScript, JavaScript enables developers to create cross-platform mobile applications that run on iOS and Android with a single codebase.

JavaScript has a Cross-Platform Compatibility ,Rich Ecosystem and thousands of libraries and frameworks.

JavaScript can be used in mobile development through three main approaches:

* **Hybrid Apps (Web-Based)**
  + Hybrid apps use HTML, CSS, and JavaScript wrapped inside a native WebView (a browser-like component). These apps are platform-independent but rely on web technologies.
* **Cross-Platform Native Apps**
  + Unlike hybrid apps, cross-platform native frameworks use JavaScript to create truly native mobile apps by interacting with the native APIs.
* **Progressive Web Apps (PWAs)**
  + PWAs are web apps that act like mobile apps by using service workers for offline capabilities. Users can install them on their home screen without downloading them from an app store.

## Comparism between the different Mobile Programming Languages

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Feature** | **JavaScript (React Native, Ionic, Cordova)** | **Swift (iOS)** | **Kotlin (Android)** | **Flutter (Dart)** | **C# (Xamarin, MAUI)** | **Java (Android)** |
| **Platform Support** | iOS & Android (cross-platform) | iOS only | Android only | iOS & Android (cross-platform) | iOS & Android (cross-platform) | Android only |
| **Performance** | Medium to High (near-native in React Native) | High | High | High (compiles to native) | Medium to High | High |
| **Code Reusability** | Yes (write once, run on both) | No | No | Yes | Yes | No |
| **Ease of Learning** | Easy (especially for web developers) | Moderate | Moderate | Moderate | Easy | Moderate |
| **Development Speed** | Fast (hot reload, reusable components) | Slower | Slower | Fast (hot reload) | Fast | Slower |
| **UI Customization** | Limited (relies on native UI components) | Full | Full | Full (custom UI engine) | Full | Full |
| **Native API Access** | Limited (requires bridges or plugins) | Full | Full | Full | Full | Full |
| **App Size** | Larger due to framework overhead | Smaller | Smaller | Larger due to Flutter engine | Larger | Smaller |
| **Community Support** | Large (React Native, Ionic, Cordova) | Large | Large | Growing fast | Medium | Large |
| **Best For** | Cross-platform apps, web-to-mobile transition | iOS apps, Apple ecosystem | Android apps, Google ecosystem | Beautiful, high-performance cross-platform apps | Enterprise apps, Microsoft integration | Android apps, enterprise applications |
| **Used By** | Facebook, Instagram, Airbnb | Apple, Lyft, LinkedIn | Google, Pinterest, Netflix | Google Ads, Alibaba, eBay | Microsoft apps, Siemens | Twitter, Spotify, LinkedIn |

# DEVELOPMENT FRAMEWORK

A **software framework** is a pre-established platform that provides a foundation for developing software applications. It offers a standard way to build and deploy applications by supplying a collection of pre-written code, components, and tools. This structure streamlines the development process, allowing developers to focus on crafting application-specific functionalities without reinventing fundamental components. ​[Futurbyte+6Wikipedia+6Polyxer+6](https://en.wikipedia.org/wiki/Software_framework?utm_source=chatgpt.com" \t "_blank)

## Types of Software Frameworks

Software frameworks can be categorized based on their application domains:

1. **Web Frameworks**: Designed for web application development, these frameworks handle aspects like HTTP requests, database interactions, and user authentication. They are divided into:​

- **Front-end Frameworks**: Focus on building dynamic user interfaces. Examples include React, Angular, and Vue.js.​[BrowserStack](https://www.browserstack.com/guide/web-development-frameworks?utm_source=chatgpt.com" \t "_blank)

**- Back-end Frameworks**: Manage server-side logic and database operations. Examples include Node.js, Django, and Ruby on Rails. ​[BrowserStack](https://www.browserstack.com/guide/web-development-frameworks?utm_source=chatgpt.com" \t "_blank)

1. **Mobile Development Frameworks**: Facilitate the creation of mobile applications for platforms like iOS and Android. Examples include Flutter, React Native, and Xamarin.​
2. **Desktop Application Frameworks**: Support the development of desktop applications across various operating systems. Examples include Electron and Qt.​
3. **Enterprise Application Frameworks**: Assist in building large-scale, complex business applications. Examples include Java EE and .NET Framework.​

## Features of Software Frameworks

Common features across various frameworks include:

**\* Code Reusability**: Encourage the reuse of code components, reducing redundancy and development time.​[DailyBot+1Orient Software+1](https://www.dailybot.com/insights/the-top-software-development-frameworks-you-need-to-know?utm_source=chatgpt.com" \t "_blank)

**\* Modularity**: Promote a modular architecture, making it easier to manage and scale applications.​

**\* Security**: Often come with built-in security features to protect against common vulnerabilities.​[ValueCoders](https://www.valuecoders.com/blog/technology-and-apps/frameworks-selection-for-projects/?utm_source=chatgpt.com" \t "_blank)

**\* Community Support**: Popular frameworks typically have extensive communities, offering support, plugins, and extensions. ​[ValueCoders+1Spiceworks Inc+1](https://www.valuecoders.com/blog/technology-and-apps/frameworks-selection-for-projects/?utm_source=chatgpt.com" \t "_blank)

## Use Cases

**- Web Frameworks**: Utilized for developing websites, e-commerce platforms, and web services.​

**- Mobile Development Frameworks**: Used to create native or cross-platform mobile applications.​

**- Desktop Application Frameworks**: Employed in building software like media players, browsers, and productivity tools.​

**- Enterprise Application Frameworks**: Applied in developing customer relationship management (CRM) systems, enterprise resource planning (ERP) software, and other large-scale business applications.​

## Advantages and Disadvantages

**Advantages:**

**\* Efficiency**: Frameworks reduce repetitive coding, enabling faster development.​[TechAffinity+4ValueCoders+4DailyBot+4](https://www.valuecoders.com/blog/technology-and-apps/frameworks-selection-for-projects/?utm_source=chatgpt.com" \t "_blank)

**\* Consistency**: Provide a standardized approach, ensuring uniformity across projects.​[ValueCoders+4Polyxer+4Wikipedia+4](https://polyxer.com/blogs/software-development-frameworks/?utm_source=chatgpt.com" \t "_blank)

**\* Maintainability**: Structured codebases are easier to maintain and update.​

**\* Scalability**: Facilitate the scaling of applications as they grow. ​[ValueCoders](https://www.valuecoders.com/blog/technology-and-apps/frameworks-selection-for-projects/?utm_source=chatgpt.com" \t "_blank)

**Disadvantages:**

**\* Learning Curve**: Some frameworks have complex architectures that require time to master.​

**\* Overhead**: May introduce performance overhead due to their generic nature.​

**\* Flexibility Constraints**: Impose certain design patterns and structures, which might limit flexibility for specific project requirements.

# MOBILE APP ARCHITECTURES AND DESIGN PATTERNS

Have you ever found yourself curious about how apps function despite different components like UI, database, APIs, etc? It’s the magic of an app’s architecture that combines everything and defines a seamless user flow. The architecture sets the rules for the connection between components and the app.

## Definition

In a simple definition, a mobile app architecture is a combination of model/design and techniques used to build a mobile application ecosystem. It functions as the blueprint for a mobile application that will take shape according to the architecture. It is a collection of UI/UX, data flow, tech stack, and everything that makes an app functional and useful. The architecture also defines the entry points for users and various components along with their interactions.

## Importance of a Good Mobile Application Architecture

The significance of mobile app architecture cannot be underestimated and it’s crucial to the success of your app. The following points state all the importance of good architecture for your mobile application.

1. **Speed and quality**

The speed and quality of your mobile application strongly rely on the underlying architecture. Undoubtedly, architecture defines the way different components interact and behave to make the whole app functional.

Architecture is the backbone of an application that provides shape and structure. Users like apps that are smooth-functioning and intuitive, and they hate laggy and unresponsive apps. An efficient architecture yields a superior user experience by providing quick app response. So, pay attention to the architecture design and follow the best tips for mobile app development to create a winning application.

1. **Compatibility**

If you want a mobile application that is compatible with different devices and mobile platforms, choosing the right architecture is vital. Different architectures for mobile apps provide different levels of compatibility with diverse systems.

Compatibility refers to a mobile app’s ability to share and receive data from other systems or perform intended functions without restrictions to interfaces. With a solid architecture, you can build a system that can interact with other systems and provide the requisite compatibility.

1. **Scalability and Adaptability**

Imagine if you want to add new features to your application that you have built without a defined architectural pattern, how difficult and complex it would be. You won’t find a way to add new functionality to your app without denting the integrity of the app.

Hence, mobile application architecture is crucial to make your app scalable and adaptive to incorporate any new requirements easily. Well-defined mobile app architectures make it easy to integrate additional features and modify the app as per your requirements.

## The Key Elements in Mobile Application Architecture Designing

You need to create a robust architecture for your app that can stand the test of time. Creating it requires consideration of different factors as follows.

**Determine the Device Requirements**

In order to build a solid architecture for your mobile application, you should consider the device and hardware specifications you wish to target. It would require an understanding of the resolution, screen size, memory, CPU, storage, and other characteristics of the target devices, along with the development environment and tools. Since the application depends on the device features and hardware for its functioning, it’s crucial to get the details of the device it would run on.

**Handle Network Fluctuations**

Through the lifecycle of your mobile application, it will encounter cases when the internet connection will be weak or fully unavailable. Hence, your app should be ready to handle the worst network conditions. It is crucial to create an application architecture that is capable of dealing with any state of network connections.

**User Interface**

The importance of UI/UX cannot be overstated and it should be designed to keep users engaged with your application and provide a seamless experience. When it comes to creating an app architecture diagram, you should include the aspects of UI/UX to craft an app that resonates with your audience.

**Push Notifications**

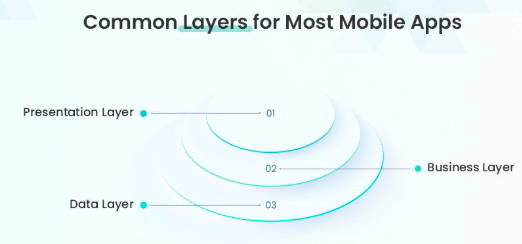
When formulizing your mobile app information architecture, determine if your users require real-time updates or notifications. Although push notifications are attractive, they are also expensive and drain phone data and battery.

There are different stages of developing a mobile app covering various technologies and layers. Every stage of app development covers different aspects from the user interface design to backend logic.

The mobile application architecture is made up of different layers that stack together to form the whole app ecosystem. Dividing the architecture into layers offers numerous benefits including reusability, scalability, flexibility, and maintenance.

## The Different Layers of Mobile App Development Architecture

The architecture of a mobile application has multiple layers that communicate with each other and pass on data to trigger various functionalities. The following are the common layers for most mobile apps.



**Presentation Layer**

The presentation layer sits on top of an app architecture stack and defines how an application will be presented to the end users. It’s a user interface and communication layer where the users interact with the application. It offers presentation services collecting and displaying user data to the end users. The presentation layer is formed from the UI/UX of a mobile application. ( This is the layer that handles user interactions and represents the face of a mobile application where users can find all kinds of GUI elements.)

**Business Layer**

This layer is related to the business logic for the application. It provides a set of rules and algorithms that manage the flow of data in the context of a business. Hence, the business layer defines how data will be generated, processed, stored, and used for a mobile application.

The primary function of this layer includes logging, data caching, security, data validation, and exception management. Depending on the operations of a mobile app, the business layer can exist on a device or on a server.

**Data Layer**

Mobile applications need a secure and efficient mechanism for data transactions. This responsibility is handled by the data layer which ensures the transfer and receipt of data seamlessly. This layer consists of various components like service agents, data access components, data utilities, etc, to enable data transactions within an app. When it comes to designing the data layer, the mobile app developer needs to consider the ease of modification as per the change in requirements and maintenance.

## What are Real-world Mobile App Architecture Examples?

When it comes to mobile architecture design there are different patterns already existing for various platforms and operating systems. Your mobile app architecture can vary based on the type of users or platforms you wish to target.

For example, the architecture for Android vs iOS apps differs in many aspects. A typical Android application is composed of multiple app components like fragments, activities, content providers, and services. The app also requires a manifest file where all these components are declared. On the other hand, iOS apps reside in the IPA container or the Bundle directory where all the app data is compressed and packed.

## Let’s check out different modern app architectures.

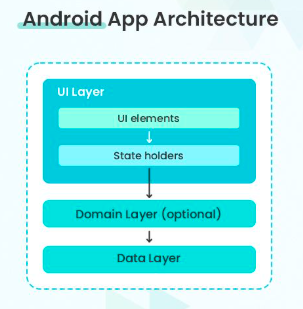
**Android Mobile App Architecture**

Mobile apps that are specially developed for Android devices are native apps. Native apps are mobile apps developed specifically for a particular operating system like Android. There are a variety of manufacturers when it comes to Android devices. So, these apps have to meet different requirements and use Java and Kotlin languages. Thus, a single architecture will not work for Android mobile app development.

However, the most common Android architecture in enterprise mobile app development is the Clean architecture. The Clean architecture is based on layers and inversion of code principles. It is composed of presentation, business, and data layers. Each layer is independent and exchange data through interfaces.

**Clean architecture offers benefits like:**

* Easy testing and troubleshooting
* UI is separated
* Not dependent on external libraries, frameworks, and databases



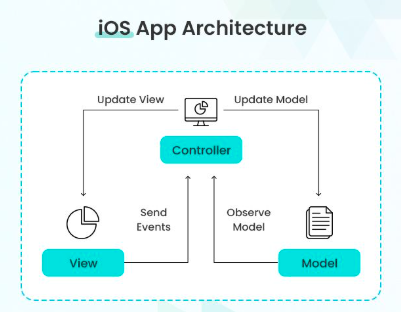
**iOS Mobile App Architecture**

Objective-C and Swift are two languages to develop native iOS applications. While there are no specific guidelines for Android apps, Apple guides on developing apps using an architecture with an MVC pattern. Although iOS developers can choose any architecture, it’s the most common pattern for iOS mobile app development. MVC is a 3-tier architecture for mobile applications. Here are the different layers it has:

* **Model:** The model layer manages the data, rules, and logic of an application and consists of model objects, parsers, networking code, etc.
* **View:** It’s the presentation layer for MVC architecture. There are elements for user interactions, and it doesn’t have any business logic.
* **Controller:** This layer establishes communication between the two other layers that are the Model and the View layers.

**MVC architecture provides benefits like:**

* Faster app development process
* Clear communication between different layers
* Easy to maintain and simple app architecture



**Hybrid Mobile App Architecture**

These mobile apps are a hybrid of native and web applications. These apps have a native app shell in which web-based content is loaded. Web technologies are used to build hybrid apps like HTML, CSS, and JavaScript. These apps can access the hardware features of a mobile app.

Hybrid mobile apps use a single codebase for different mobile platforms. So, a single hybrid app will function on different platforms whether Android or iOS.

**Hybrid app architecture has benefits like**:

* Faster development time
* Single codebase for different platforms
* Cost-effective and cross-functional
* Access hardware features

**Cross-platform App Architecture**

Another popular mobile app development architecture is cross-platform apps. Just like hybrid apps, there is a single codebase for various platforms, and they are compatible with Android and iOS. But they are built on frameworks.

There are several cross-platform app development frameworks and technologies like Flutter, Ionic, Xamarin, React Native, etc. Compared to hybrid apps, these apps offer an experience that is much closer to native apps, and they are better in performance.

**Cross-platform apps provide benefits like:**

* Single codebase for multiple platforms
* Native app-like performance and quality
* Access to hardware features
* Faster loading speed
* Cost-effective than native apps

## Key Attributes of a Result-Driven Mobile App Architecture

An app architecture needs to be flexible, extensible, modifiable, and possess other qualities for a stable and robust mobile application. Here are some key characteristics of a well-designed and impeccable app architecture.

**Reusability**

With code reusability, developers can develop and deploy application features at a faster speed. An efficient mobile app architecture is built with code reusability in mind and allows developers to quickly build and release application updates and versions.

**Sustainability**

Changes in the environment where a mobile app resides are uncontrollable. For instance, technological changes or a shift in market demand. Hence, your app needs to be ready to adjust according to these changes. It calls for a mobile app architecture design that is resilient to these changes. A sustainable mobile app is capable of handling changes like improved technologies and servers.

**Performance**

Mobile app developers prioritize user experience and functionality when it comes to app development. Users expect that applications, whether Android or iOS, should respond to their taps instantly and execute tasks. Hence, choose an architecture that offers excellent performance.

**Extensibility**

Changes in user needs require you to integrate more functionality over time. If your app is built on a flexible architecture, it will be easy to implement new features, or it will be a complex process otherwise. Hence, dividing app components to form a loosely coupled architecture will simplify the integration of features.

**Scalability**

You need an app architecture that supports easy scalability to accommodate new users and increased load on your app.

**Security**

In the modern world, security has become so vital that big businesses spend millions to beef up security. Consider an app architecture that provides a high level of security and ensures compliance with various standards.

**Testability**

If your app architecture has high testability, it will be more reliable as the chances of bugs in it are low. A testable architecture makes it easy to discover and fix errors or bugs early in the development phase, making the final application well-performing and versatile.

**Intuitiveness**

Apart from making your app flexible and scalable, you should also focus on user experience. Build your app with a user-friendly interface that is easy to understand.

## What are Mobile App Architecture Best Practices?

From performance to scalability, all vital features of your mobile app depend on its architecture. Therefore, it is important to choose the right architecture that meets your requirements and offers excellent user experience. It will help to save your money and precious time by eliminating reworks in the future.

**You can start with a good mobile app architecture diagram with these best practices.**

* Choose an architecture that separates the UI layer from the data layer. It will provide reusable code and simplify the process of change.
* Work with good software engineering practices and principles. Follow principles like KISS, YAGNI, SOLID, DRY, etc, when defining the architecture and developing the app.
* For APIs, you can leverage simple and lightweight data formats such as JSON.
* Stay updated with the current mobile app development trends to develop an application as per modern user needs.

## How to Choose a Good Mobile Application Architecture?

Now that you know how crucial an architecture for a mobile application is, you must weigh the pros and cons of different architectures to pick the best one. You can keep our advice below in mind when it comes to choosing an appropriate mobile app architecture design.

* You should consider developing native applications if you don’t have a shoestring budget. It offers many advantages including intuitive functionality and performance.
* Does your target audience include Android and iOS users? Choose a cross-platform app or create native apps that will serve users on various platforms.
* You can choose a hybrid solution to help access your brand from a more diverse set of devices.

There are different principles and architectural patterns when it comes to developing mobile apps, each with its own strengths and limitations. You cannot compromise on your app architecture because it’s the foundation of your app and its performance and stability depend on the architecture.

The decision of choosing an architecture for your mobile app is influenced by several factors including business requirements, application type and functionality, tech stack, flexibility, development timeframe, etc. You must evaluate different app architectures based on your needs and constraints to pick the right architecture for your mobile app.

# Design Patterns for Mobile Development

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.

## Important Topics for Mobile Development Design Patterns

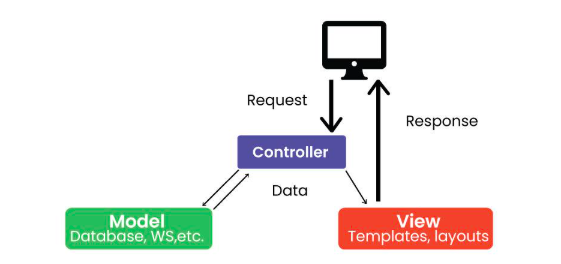
* Model View Controller (MVC) Architecture
* Model View Presenter (MVP) Architecture
* Model View View Model (MVVM) Architecture
* VIPER Architecture
* Singleton
* Factory Method
* Observer
* Dependency Injection (DI)
* Adapter
* Strategy
* Composite
* Conclusion

## Architectural design patterns

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

**Model View Controller (MVC) Architecture**

MVC is a design model that separates an application into three interacting parts: Model, View, and Controller. This separation allows for better code design and modularization.



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

**View:** Displays data to the user.

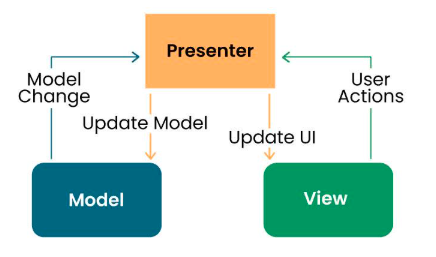
**Controller:** Processes user input and controls data flow between Model and View.

**For Example:**

Imagine a mobile weather app. The model stores weather information, the View displays it to the user, and the controller handles user interactions such as updating the displayed location or converting units (e.g. from Celsius to Fahrenheit).

**Model View Presenter (MVP) Architecture**

MVP is a new architecture that separates an application into three parts: Model, View, and Presenter. This is similar to MVC but puts more responsibility on the Teacher to manage the interaction between Model and View.



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

**View:** Represents the user interface.

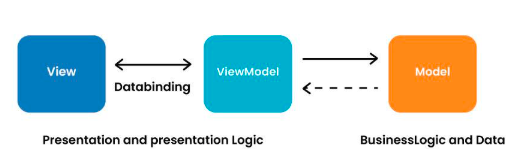
**Designer:** Acts as an intermediary processing user input and updating the View and Model.

**For Example:**

In a note-taking app, the Model would store the text, the View would display it, and the provider would handle user input such as typing, editing, or deletes the process.

**Model View View Model (MVVM) Architecture**

MVVM is a design model widely used in mobile development, especially in frameworks like Android’s Jetpack. Its purpose is to separate the application into three parts: Model, View, and ViewModel.



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

**View:** Represents the user interface.

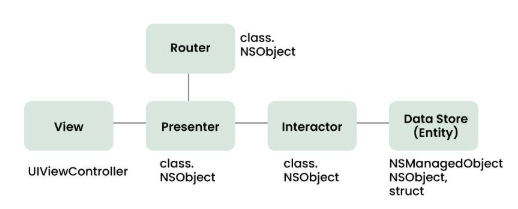
**ViewModel:** Acts as an interface between the Model and the View, which contains the reference logic.

**For Example:**

In an e-commerce application, the Model contains product data, the View displays product information, and the ViewModel manages interactions, such as adding items to a cart.

**VIPER Architecture**

VIPER stands for View, Interactor, Presenter, Entity, and Router. VIPER is primarily based at the clean architecture ideas, which purpose to separate the concerns of different layers of the utility. Each layer has a single duty and communicates with different layers through properly-defined interfaces.



**Let’s briefly explain the function of every element:**

**View:** This is the consumer interface layer, wherein the perspectives and look at controllers are defined. The view is chargeable for showing the information provided by way of the presenter and forwarding the person moves to the presenter.

**Presenter:** This is the presentation layer, where the good judgment for formatting and imparting the records is defined. The presenter is liable for fetching the records from the interactor, reworking it right into a suitable layout for the view, and updating the view hence. The presenter additionally handles the consumer movements acquired from the view and calls the router to navigate to other screens.

**Interactor:** This is the enterprise good judgment layer, where the common sense for manipulating the data and interacting with external services is described. The interactor is accountable for gaining access to the facts from the service layer, acting any vital operations on it, and returning it to the presenter. The interactor additionally communicates with the entity layer to store and retrieve the information fashions.

**Entity:** This is the information layer, wherein the data models and systems are described. The entity is responsible for representing the data in a constant and coherent manner throughout the software. The entity layer also can encompass records get entry to gadgets (DAOs) or repositories that summary the information of records patience and retrieval.

**Router:** This is the navigation layer, where the logic for routing and transitioning among different monitors is defined. The router is chargeable for developing and providing the view controllers, passing any vital facts to them, and coping with any dependencies or configurations. The router also communicates with the presenter to get hold of the navigation requests and execute them.

## Design pattern

**Singleton Method Design Pattern**

The singleton policy ensures that there is only one instance of a class and provides global access. This is especially useful when you want to manage a single instance of an object or control access to a delayed object.

**For Example:**

Singleton can be used to manage player’s score in mobile game. There can only be one instance that is responsible for tracking scores and is updated throughout the game.

**Factory Method Design Pattern**

The Factory Method model defines an interface for creating an object but allows subclasses to modify the type of the created object. Especially useful when you need to create objects with a common interface but different functionality.

**For Example:**

In a mobile app that supports multiple payment gateways, payments can be made using the Factory Method. Each payment gateway (e.g., PayPal, Stripe) is a small business and provides its services.

**Observer Method Design Pattern**

The observer structure defines one to many dependencies between objects, so when one object changes its state, all its dependents are automatically notified and updated. This is useful for scheduling distributed events.

**For Example:**

In the reports app, many features (Observers) such as the title widget, the report feed view, and the notification provider (Themes) can subscribe to updates when new information arrives. The observer model ensures that they are created all registered parts report, and accordingly You can update it.

**Dependency Injection (DI) Method Design Pattern**

Dependency Injection is a method of providing class dependencies from the outside, rather than creating them in the class. It improves code modularity and testability by making classes independent of their dependencies.

**For Example:**

In an Android app, instead of creating a single database connection object in the class, you can place the database object externally, allowing you to easily test and modify database operations.

**Adapter Method Design Pattern**

The adapter configuration allows you to use the interface of an existing class as a link to a new one. It is often used to work with others without modifying the source code of existing classes.

**For Example:**

If you want to use a third-party library that provides data in a different way, you can create an adapter that will convert the library's results to the format your app would expect and make sure that they are compatible meet without changing the library code.

**Strategy Method Design Pattern**

The strategy model defines a family of algorithms, contains each of them, and provides them with flexibility. It allows you to select the appropriate algorithm at runtime. This example is useful when you want to provide different options for a task.

**For Example:**

In a weather application, you can use various methods to retrieve weather information, such as using a REST API, WebSocket, or local storage. The user can change these options, and the app adapts to his preferences.

**Composite Method Design Pattern**

A composite pattern allows you to arrange objects in a tree structure to represent a part-of-the-whole structure. This is helpful when you have to deal with individual objects and sets of objects accurately.

**For Example:** You can use Composite pattern to create complex shapes from simple shapes in the mobile drawing app. Complex designs can contain individual designs, allowing users to manipulate and categorize resources as needed.

Design processes play an important role in mobile app development by providing proven solutions to common software design challenges. Using this framework allows developers to create maintainable, extensible, and efficient applications. Understanding when and how to apply these options can significantly improve the quality of your mobile app codebase. Whether you’re working for Android, iOS, or any other mobile platform, a solid understanding of these design patterns will allow you to create robust, scalable mobile applications.

# **REQUIREMENTS ENGINEERING PROCESS**

## What is Requirements Engineering?

Requirements engineering is the systematic and strict approach to the definition, creation, and verification of requirements for a software system OR simply the process of identifying, eliciting, analyzing, specifying, validating, and managing the needs and expectations of stakeholders for a software system. To guarantee the effective creation of a software product, the requirements engineering process entails several tasks that help in understanding, recording, and managing the demands of stakeholders.

## Requirements Engineering Process

* Feasibility Study
* Requirements elicitation
* Requirements specification
* Requirements for verification and validation
* Requirements management

**1. Feasibility Study**

The feasibility study mainly concentrates on the prerequisite analyses processes it can futher be divided into:

**Technical Feasibility**: In Technical Feasibility current resources both hardware software along required technology are analyzed/assessed to develop the project. This technical feasibility study reports whether there are correct required resources and technologies that will be used for project development.

**Operational Feasibility**: In Operational Feasibility degree of providing service to requirements is analyzed along with how easy the product will be to operate and maintain after deployment.

**Economic Feasibility**: In the Economic Feasibility study cost and benefit of the project are analyzed. This means under this feasibility study a detailed analysis is carried out will be cost of the project for development which includes all required costs for final development hardware and software resources required, design and development costs operational costs, and so on. After that, it is analyzed whether the project will be beneficial in terms of finance for the organization or not.

**Legal Feasibility**: In legal feasibility, the project is ensured to comply with all relevant laws, regulations, and standards. It identifies any legal constraints that could impact the project and reviews existing contracts and agreements to assess their effect on the project’s execution.

**Schedule Feasibility**: In schedule feasibility, the project timeline is evaluated to determine if it is realistic and achievable.

**2. Requirements Elicitation**

It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of the same type, standards, and other stakeholders of the project. The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping, etc. Requirements elicitation is the process of gathering information about the needs and expectations of stakeholders for a software system. This is the first step in the requirements engineering process and it is critical to the success of the software development project. The goal of this step is to understand the problem that the software system is intended to solve and the needs and expectations of the stakeholders who will use the system.

Several techniques can be used to elicit requirements, including:

**Interviews:** These are one-on-one conversations with stakeholders to gather information about their needs and expectations.

**Surveys:** These are questionnaires that are distributed to stakeholders to gather information about their needs and expectations.

**Focus Groups:** These are small groups of stakeholders who are brought together to discuss their needs and expectations for the software system.

**Observation:** This technique involves observing the stakeholders in their work environment to gather information about their needs and expectations.

**Prototyping:** This technique involves creating a working model of the software system, which can be used to gather feedback from stakeholders and to validate requirements.

It’s important to document, organize, and prioritize the requirements obtained from all these techniques to ensure that they are complete, consistent, and accurate.

**3. Requirements Specification**

This activity is used to produce formal software requirement models. All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality. During specification, more knowledge about the problem may be required which can again trigger the elicitation process. The models used at this stage include ER diagrams, data flow diagrams(DFDs), function decomposition diagrams(FDDs), data dictionaries, etc.

Requirements specification is the process of documenting the requirements identified in the analysis step in a clear, consistent, and unambiguous manner. This step also involves prioritizing and grouping the requirements into manageable chunks.

The goal of this step is to create a clear and comprehensive document that describes the requirements for the software system. This document should be understandable by both the development team and the stakeholders.

Several types of requirements are commonly specified in this step, including

**Functional Requirements:** These describe what the software system should do. They specify the functionality that the system must provide, such as input validation, data storage, and user interface.

**Non-Functional Requirements:** These describe how well the software system should do it. They specify the quality attributes of the system, such as performance, reliability, usability, and security.

**Constraints:** These describe any limitations or restrictions that must be considered when developing the software system.

**Acceptance Criteria:** These describe the conditions that must be met for the software system to be considered complete and ready for release.

Once the requirements are specified, they must be reviewed and validated by the stakeholders and development team to ensure that they are complete, consistent, and accurate.

**4. Requirements Verification and Validation**

**Verification:** It refers to the set of tasks that ensures that the software correctly implements a specific function.

**Validation:** It refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements. If requirements are not validated, errors in the requirement definitions would propagate to the successive stages resulting in a lot of modification and rework. The main steps for this process include:

* The requirements should be consistent with all the other requirements i.e. no two requirements should conflict with each other.
* The requirements should be complete in every sense.
* The requirements should be practically achievable.

Reviews, buddy checks, making test cases, etc. are some of the methods used for this.

Requirements verification and validation (V&V) is the process of checking that the requirements for a software system are complete, consistent, and accurate and that they meet the needs and expectations of the stakeholders. The goal of V&V is to ensure that the software system being developed meets the requirements and that it is developed on time, within budget, and to the required quality.

It’s important to note that V&V is not a one-time process, but it should be integrated and continue throughout the software development process and even in the maintenance stage.

**5. Requirements Management**

Requirement management is the process of analyzing, documenting, tracking, prioritizing, and agreeing on the requirement and controlling the communication with relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible to incorporate changes in requirements specified by the end users at later stages too. Modifying the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process.

Requirements management is the process of managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant. The goal of requirements management is to ensure that the software system being developed meets the needs and expectations of the stakeholders and that it is developed on time, within budget, and to the required quality.

Several key activities are involved in requirements management, including:

**Tracking and controlling changes:** This involves monitoring and controlling changes to the requirements throughout the development process, including identifying the source of the change, assessing the impact of the change, and approving or rejecting the change.

**Version control:** This involves keeping track of different versions of the requirements document and other related artifacts.

**Traceability:** This involves linking the requirements to other elements of the development process, such as design, testing, and validation.

**Communication:** This involves ensuring that the requirements are communicated effectively to all stakeholders and that any changes or issues are addressed promptly.

**Monitoring and reporting:** This involves monitoring the progress of the development process and reporting on the status of the requirements.

## Tools Involved in Requirement Engineering

* Observation report
* Questionnaire ( survey, poll )
* Use cases
* User stories
* Requirement workshop
* Mind mapping
* Roleplaying
* Prototyping

## Advantages of Requirements Engineering Process

* Helps ensure that the software being developed meets the needs and expectations of the stakeholders
* Can help identify potential issues or problems early in the development process, allowing for adjustments to be made before significant
* Helps ensure that the software is developed in a cost-effective and efficient manner
* Can improve communication and collaboration between the development team and stakeholders
* Helps to ensure that the software system meets the needs of all stakeholders.
* Provides an unambiguous description of the requirements, which helps to reduce misunderstandings and errors.
* Helps to identify potential conflicts and contradictions in the requirements, which can be resolved before the software development process begins.
* Helps to ensure that the software system is delivered on time, within budget, and to the required quality standards.
* Provides a solid foundation for the development process, which helps to reduce the risk of failure.

## Disadvantages of Requirements Engineering Process

* Can be time-consuming and costly, particularly if the requirements-gathering process is not well-managed
* Can be difficult to ensure that all stakeholders’ needs and expectations are taken into account
* It Can be challenging to ensure that the requirements are clear, consistent, and complete
* Changes in requirements can lead to delays and increased costs in the development process.
* As a best practice, Requirements engineering should be flexible, adaptable, and should be aligned with the overall project goals.
* It can be time-consuming and expensive, especially if the requirements are complex.
* It can be difficult to elicit requirements from stakeholders who have different needs and priorities.
* Requirements may change over time, which can result in delays and additional costs.
* There may be conflicts between stakeholders, which can be difficult to resolve.
* It may be challenging to ensure that all stakeholders understand and agree on the requirements.

# MOBILE APP DEVELOPMENT COST ESTIMATION

## Understanding App Development Costs: A Breakdown

Let’s break down the different stages involved in a mobile app development project. This way we’ll get a good understanding of how app development costs are formed.

**1. Discovery**

This phase is for understanding the purpose of the app: how it’s used for engaging its users, and what are the business needs of the app. It determines the overall direction and features required. The phase can involve tasks like market research and user research. This phase typically costs from 1 000 to 10 000 dollars. ([Business of Apps](https://www.businessofapps.com/app-developers/research/app-development-cost/), [Mobiversal](https://blog.mobiversal.com/app-development-cost-breakdown.html))

**2. App Design (UI and UX)**

User experience (UX) and user interface (UI) design are critical for creating an engaging and intuitive app.

In this phase, specifications and requirements for the app are defined. It includes creating wireframes, mockups, and user flows, as well as finalizing the app's design elements and user experience. Designs from this stage inform the development process and ensure a cohesive vision.

The cost of this stage can vary largely depending on the approach and the app: typical costs are between 5 000 and 55 000 dollars.

**3. Mobile App Development & Infrastructure**

This stage involves the actual coding or building of the app. Developers bring the design and functionality to life, implementing features, integrating APIs and databases, and ensuring seamless performance across different devices and platforms.

The infrastructure of the app is also set up at this stage, including the server, control panel and ensuring scalability to the desired number of users.

This phase typically costs something between 2 000 and 60 000 dollars.

**4. App Testing and Quality Assurance**

Before launching the app, the app needs to be tested. This includes functional testing, usability testing, performance testing, and bug fixing.

Testing ensures that the app is stable, reliable, and meets the desired standards of quality.

This stage typically costs 5 000 dollars or less.

**5. Publishing to App Store and Google Play**

Once the app is ready, it needs to be submitted to app stores for distribution. Each platform, Apple App Store and Google Play Store, has its own set of guidelines and review processes.

The review process typically takes about a week or less, but the duration can vary.

App publishing involves creating app store listings, providing app descriptions, screenshots, and promotional materials.

This stage isn’t as significant in costs as the others: the fees are a one-time 25 dollar cost for Google Play Store and a yearly subscription of 99 dollars for Apple’s App Store.

**6. Ongoing maintenance and updates**

While the initial development phase is crucial, ongoing maintenance and updates are equally important to keep the app running smoothly and meeting user expectations.

It is usually suggested that app maintenance can cost 15-20% of the total development cost on a yearly basis. However, the maintenance cost of the app during its first year after launch can hike up to 50% of the initial app development cost.

Maintenance costs that require experts – customer support, updates, bug fixes – can be included in the initial pricing of app development, have a fixed monthly add-on fee, or they can be charged on a price per hour basis.

## Factors influencing app development costs

Now you know what stages there are in the app development process – and what costs can be involved in them. There’s a lot of variance between app development costs. What are the factors that determine the cost of the app exactly?

Development method: Traditional Mobile App Development vs. No-Code vs. Low Code

When it comes to app development, there are different approaches to consider. Custom development, coding, or traditional mobile app development – whatever you call it – is not the only option anymore.

Each method has its own advantages and considerations in terms of time, cost, and technical expertise required.

Let’s look into each one:

### Traditional Mobile App Development - Coding

Traditional app development involves writing code from scratch using programming languages.

**Costs:** Demands more time and resources due to the extensive coding, testing, and debugging involved. This increases costs as more hours are required for each stage.

**Time to market:** Takes longer due to the manual coding process.

**Customization:** Offers maximum flexibility and customization but requires skilled developers with expertise in the chosen programming language.

**Technologies and Languages:** Allows developers to select from a wide range of technologies and programming languages based on project requirements.

### No-Code Development

[No-code app development platforms like Choicely](https://www.choicely.com/app-builder) have gained popularity in recent years as they enable the creation of functional apps without the need for extensive coding knowledge.

**Costs:** [No-code development](https://www.choicely.com/blog/no-code-mobile-app-development) eliminates the need for dedicated developers, which can significantly reduce costs.

**Time to market:** No-code app builders provide pre-built components and visual interfaces that allow users to create apps by simply dragging and dropping elements. This accelerates the development process, reducing time to market.

**Customization:** Some of the no code app builders have limitations when it comes to complex functionalities or highly customized features. It's important to evaluate the platform's capabilities and ensure it aligns with the specific requirements of your app. Choicely supports custom elements created on top of no code apps. With no code, individuals without extensive coding knowledge can create apps. This democratizes app development, making it accessible to a wider audience.

**Technologies:** Solutions might only cater to one technology, or programming language. [Choicely app builder](https://www.choicely.com/app-builder) produces native iOS and Android apps and supports multiple languages. The platform doesn't require developers to learn a new one as the most common languages are covered. Still, native languages are the preferred and the most straightforward languages to be implemented with Choicely.

### Low-Code Development

Low-code development lies between traditional coding and no-code development. It streamlines the development process by providing visual interfaces for app creation, combined with some level of coding flexibility. Here are a few key considerations:

**Costs**: Because of the streamlined of the app development process, apps built with low code solutions are more affordable than custom coded apps.

**Time to market:** Low-code platforms simplify the development process and the time to market typically sits somewhere between custom coding and no code development.

**Customization:** Complex app requirements may require custom coding and additional technical expertise.

**Technologies:** The range of technologies and devices supported by low-code platforms may vary. It's essential to evaluate platform compatibility with desired features and target platforms (iOS, Android, or cross-platform).

 Considering these different development methods helps you make an informed decision based on your project requirements, available resources, and desired cost and timeframes.

# CONCLUSION

The mobile app development process involves several crucial aspects, including app types, programming languages, frameworks, architectures, requirement engineering, and cost estimation. Understanding the differences between native, hybrid, and progressive web apps helps developers choose the best approach based on performance, compatibility, and user experience. Additionally, selecting the right programming language and framework significantly impacts development speed, cost, and scalability.

A well-structured mobile app architecture and design pattern ensures maintainability and efficiency, while a thorough requirement engineering process helps define user needs and project scope. Finally, an accurate cost estimation is essential for budgeting and resource allocation. Mastering these elements enables developers to create high-quality, efficient, and user-friendly mobile applications.

# REFERENCES

* WHAT IS A MOBILE APPLICATION? ( <https://www.techtarget.com/whatis/definition/mobile-app> )
* Types of mobile applications ( <https://www.geeksforgeeks.org/types-of-mobile-application-appium/> )
* Comparison between the different types of mobile applications ( google.com )
* WHAT ARE MOBILE APP PROGRAMMING LANGUAGES?

**<https://www.ibm.com>**

<https://www.builder.ai/>

<https://newwavesolution.com>

<https://developer.ibm.com>

<https://developer.apple.com/swift/>

<https://www.techtarget.com>

<https://aws.amazon.com/>

<https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting>

<https://dart.dev/overview>

* DEVELOPMENT FRAMEWORK ( )
* Mobile app architectures and design patterns ( <https://radixweb.com/blog/guide-to-mobile-app-architecture> )
* Design Patterns for Mobile Development ( <https://www.geeksforgeeks.org/design-patterns-for-mobile-development/> )
* REQUIREMENTS ENGINEERING PROCESS ( <https://www.geeksforgeeks.org/software-engineering-requirements-engineering-process/> )
* MOBILE APP DEVELOPMENT COST ESTIMATION ( )