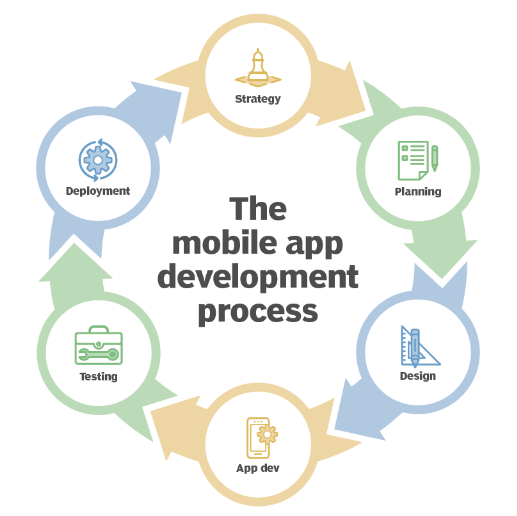
# What Is a Mobile Application? (https://www.techtarget.com/whatis/definition/mobile-app)

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

# (https://www.geeksforgeeks.org/types-of-mobile-application-appium/)

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.

## 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.

## 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.

## 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.

## 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)

# An Overview of Architecture for Mobile Apps (https://radixweb.com/blog/guide-to-mobile-app-architecture)

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**

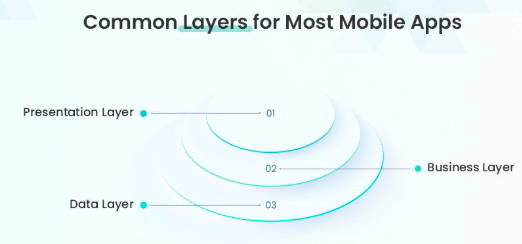
When formulizing your mobile app information architecture, determine if your users require real-time updates or notifications. Although push notifications are attractive, they are also expensive and drain phone data and battery.

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.**

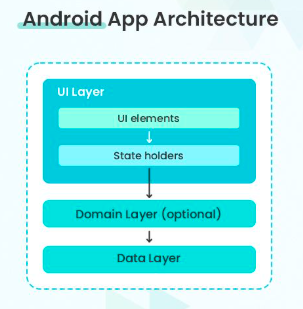
**Android Mobile App Architecture**

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



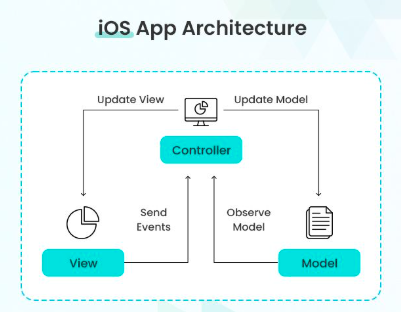
**iOS Mobile App Architecture**

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 (https://www.geeksforgeeks.org/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.

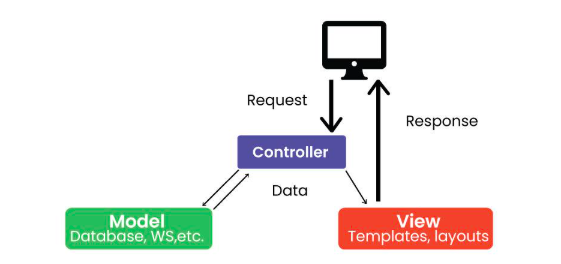
**Important Topics for Mobile Development Design Patterns**

* 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

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.

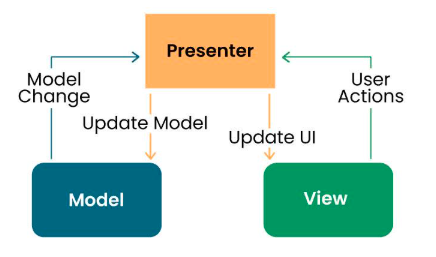
**Controller:** Processes user input and controls data flow between Model and View.

**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.

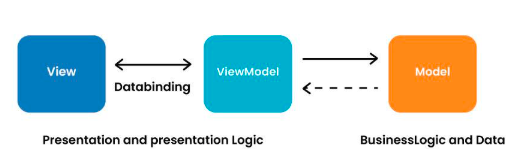
**Designer:** Acts as an intermediary processing user input and updating the View and Model.

**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.

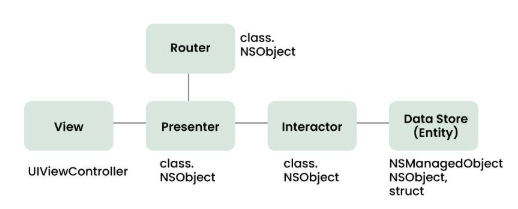
**ViewModel:** Acts as an interface between the Model and the View, which contains the reference logic.

**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.

**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.