# download

FACULTY OF ENGENEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGENEERING

|  |  |
| --- | --- |
| **NAMES** | **MATRICULES** |
| NSEM CONFIDENT NJOCK | FE21A285 |
| NZEMTEJUH SYLVANUS | FE21A296 |
| NKWENKAM JENNIFER | FE21A279 |
| VTALAH DE GERAUD | FE21A329 |
| OLETIA AJONGAKUE SYNCLAIR | FE21A190 |

## CEF440: INTERNET PROGRAMMING AND MOBILE PROGRAMMING

**TASK 1: REPORT**

**Supervisor: Dr Nkemeni Valery**

**TABLE OF CONTENTS**

**[1. QUESTION 1 3](#_Toc16217)**

[1.2. Evolution of web and mobile apps 3](#_Toc16217)

[1.4. Key differences: web apps vs. native apps](#_Toc5565)

[1.4.1. Functionality 4](#_Toc16314)

[1.4.2. User experience](#_Toc7352)

[1.4.3. Performance](#_Toc8273)

[1.4.4. App development](#_Toc892)

[1.4.5. Customer reach](#_Toc5899)

[1.5. Key differences: native apps vs. hybrid apps](#_Toc20569)

[1.5.1 App development](#_Toc1770)

[1.5.2. Cost efficiency](#_Toc16507)

[1.6. Key differences: hybrid apps vs. progressive web apps 5](#_Toc11755)

[1.6.1. Organic reach](#_Toc16734)

[1.6.2. Performance](#_Toc13615)

[1.6.3. Maturity](#_Toc19178)

[1.7. When to use web apps vs. hybrid apps vs. native apps?](#_Toc798)

[1.7.1. Time to market](#_Toc18310)

[1.7.2. Customer requirements](#_Toc26550)

[1.7.3. Marketing strategy](#_Toc24989)

[1.7.4. Complexity 6](#_Toc15479)

[1.8. web apps vs. hybrid apps vs. native apps](#_Toc2664)

**2. QUESTION 2........................................................................................................................................................................................................**

[2.1. Importance of Choosing the Right Programming Language for App Development 7](#_Toc7303)

[2.1.1. Fitting In: Platform Acceptance](#_Toc27650)

[2.1.2. Building Blocks: Libraries and Frameworks](#_Toc16123)

[2.1.3. Playing with Others: Third-Party Integrations](#_Toc18695)

[2.1.4. Speed Demons vs. Efficiency Champs: Performance and Efficiency](#_Toc3179)

[2.1.5. Specialized Tools: Domain-Specific Capabilities](#_Toc7765)

[2.1.6. Growing Your App: Extensibility and Customization](#_Toc4675)

[2.2. Best Programming Languages for Mobile App Development 8](#_Toc16294)

[2.2.1. Swift](#_Toc18791)

[2.2.2. Kotlin](#_Toc11799)

[2.2.3. Java](#_Toc4898)

[2.2.4. Dart (used with Flutter)](#_Toc8158)

[2.2.5. JavaScript (used with React Native)](#_Toc9859)

[2.2.6. C# (used with Xamarin)](#_Toc10940)

[2.2.7. Objective-C 9](#_Toc14728)

[2.2.8. Python](#_Toc20205)

[2.2. 9. HTML/CSS](#_Toc19663)

[2.2.10. Rust](#_Toc20132)

[2.2.11. Ruby](#_Toc30118)

[2.2.12. Prolog](#_Toc24145)

[2.2.13. LISP 10](#_Toc8616)

[2.2.14. Malboge](#_Toc23628)

[2.2.15. Go](#_Toc26707)

[2.3. Conclusion](#_Toc29334)

**3. QUESTION 3........................................................................................................................................................................................................**

[3.1. Comparing the various features of the different frameworks. 11](#_Toc25144)

**4. QUESTION 4........................................................................................................................................................................................................**

[4.1. Design Patterns for Mobile Development 12](#_Toc23494)

[For Example: 13](#_Toc31407)

[For Example: 14](#_Toc3180)

[For Example:](#_Toc6204)

[For Example: 15](#_Toc19558)

[For Example: 16](#_Toc6406)

[For Example:](#_Toc26620)

[4.1.8. Dependency Injection (DI) Method Design Pattern](#_Toc19466)

[For Example:](#_Toc7871)

[4.1.9. Adapter Method Design Pattern](#_Toc5367)

[For Example:](#_Toc26236)

[4.1.10. Strategy Method Design Pattern](#_Toc6098)

[For Example:](#_Toc2765)

[4.1.11. Composite Method Design Pattern 17](#_Toc29397)

[For Example:](#_Toc13825)

[4.2. Conclusion](#_Toc36)

**5. QUESTION 5........................................................................................................................................................................................................**

[5.1. What Are Mobile App Requirements?](#_Toc5748)

[5.2. Mobile App Requirements Benefits](#_Toc9805)

[5.3. How to Gather and Document App Requirements 18](#_Toc11157)

[5.4. Mobile App Requirements Types](#_Toc13746)

[5.5.1. Define Your App Idea and Purpose](#_Toc24617)

[5.5.2. Gather and Align the App and Business Objectives or Goals 19](#_Toc585)

[5.5.3. Run a Market Analysis and Competitor Analysis](#_Toc906)

[5.5.4. Determine Scenarios and a User Persona 20](#_Toc9304)

[5.5.5. Gather and Prioritise Functional and Non-Functional Requirements](#_Toc496)

[5.5.6. Design Use Cases and User Stories 21](#_Toc24597)

[5.5.7. Write an App Requirements Document 23](#_Toc31758)

[5.5.8. Deploy Prototyping and Wireframing 25](#_Toc29529)

[5.5.10. Apply Agile Methodology](#_Toc8326)

**6. QUESTION 6............................................................................................................................................................................…………………**

[7.0.REFERENCES](#_Toc11157) 27

1. **QUESTION 1:Review and compare the major types of mobile apps and their differences (native,progressive web apps, hybrid apps)**

An application is a software that lets you exchange information with customers and help them complete specific tasks. Different types of applications, or apps, are based on their development method and internal functionality. Web apps are delivered over an internet browser. Users don't need to install them on their devices. Native apps, on the other hand, are built for a specific platform or device type. The user must install the appropriate software version on their device of choice. Hybrid apps are native applications with a web browser embedded inside them.

* 1. Why are there so many different types of applications?

Application development started with the invention of computers. When desktops and personal computers were the only devices, companies installed applications on premises or in a company's data center then delivered functionality via corporate networks. Most early applications used a client-server architecture. An on-premises server centrally stored and processed data. Users had to install separate client apps on their devices. The client apps communicated with the server. Different client app versions had to be developed and installed for multiple platforms. For instance, a Windows machine required a different version than a Linux machine.

### ****Evolution of web and mobile apps****

### With the internet's growth, servers and clients could be located anywhere on the globe. Web applications emerged to reach more users and give more flexibility than applications had to that point. Instead of installing client apps, users could access server functionality straight from the browser. At the same time, the development of mobile devices created newer platforms for application delivery. Users had more choices in how they wanted to access software functionality. They could use the browser or install an app on their device of choice.

### ****Challenges in app development****

Businesses today have to develop different types of web and mobile applications to reach multiple user groups. Here are some examples:

* Web apps for users who want to work on the browser
* Windows and macOS applications for desktop users
* Android apps for Android mobile device types
* iOS apps for iOS devices

Today, app developers have to write the same software in different languages. They also test, package, and deploy the same software for various platforms. As a result, new feature releases, bug fixing, and software maintenance become time-consuming and expensive. Various design solutions, such as containers and service-oriented architecture, address the issue. Different design approaches led to the creation of different types of web and mobile

## Key differences: web apps vs. native apps

The term web app indicates an app you can access from the browser of a desktop or mobile device. The term native app suggests an app you can download and install on your device. A native mobile app is developed specifically for a mobile device. The terms native app, native mobile app, and mobile app are often used interchangeably to refer to the same type of software. Some key differences between native apps and web apps are given below.

### ****Functionality****

Web apps only give users access to interactions supported by web browsers. Even though a web application has rich design elements, it cannot access device features. Native mobile apps, on the other hand, let users interact with their devices' internal hardware and operating systems. You can grant users access to native features like:

* Device location tracking
* Device microphone and cameras
* User contact lists
* Touch gestures, device tilt, and other user interactions
* Device security features like a fingerprint scan or face recognition

### ****User experience****

Web apps lack consistency in user experience due to their heavy dependency on browsers. Certain features or images may look different on different browsers. Buttons and menu bar features may be challenging to access from mobile browsers. Browser window resizing may impact the look, feel, and functionality of the web application. Users tend to have a better experience on native mobile apps. For instance, the native app fills the screen and takes control of the entire device. Users get more out of the native app because they are comfortable with the interactions. The native app can also send push notifications to users and get them to re-engage.

### ****Performance****

Native applications give better performance when compared to web apps. They are faster, more responsive, and more interactive. However, the onus is on the user to maintain the native app performance. The user must download and install regular software updates to keep the app running optimally. Web applications are slower and less responsive, but they give you more control over performance. Software updates benefit all users immediately.

### ****App development****

Web apps are comparatively simpler, cheaper, and faster to develop. Time to market is shorter because of a straightforward app development process. They are also easier to maintain because you only have to test and update a single codebase. Native apps require a heavier financial investment. They also need development teams with cross-platform development experience. For instance, a developer specializing in native iOS apps may not be the best choice for building native Android apps.

### ****Customer reach****

Web apps have limited customer reach, as users require an internet connection to access the app. In the case of mobile web apps, there is a multi-step access process because users first have to open the mobile browser then find the app. Conversely, you can design native apps to work offline on the user's device. Native apps also give more discoverability because they are on app stores. You can run marketing campaigns within the app store to reach a wider or newer customer base.

## Key differences: native apps vs. hybrid apps

A hybrid app is a particular type of native app. Like native apps, users can download and install a hybrid app from app stores. However, the internal structure of native and hybrid apps is very different. Internally, hybrid apps are more like web apps. Hybrid apps lie somewhere between native and web apps.

### ****1.5.1 App development****

In a native app, your developers have to rewrite and redesign all the app functionality in the native development language. A hybrid app lets you write the app functionality in a single codebase. You can then wrap your code in a lightweight native app shell or container. The container enables you to take advantage of native features in your mobile devices, like hardware, calendars, and notifications.

### ****1.5.2. Cost efficiency****

Hybrid apps achieve the same performance and user experience as native apps at a lower cost. Your developers can build them using commonly used app development languages and technologies like JavaScript, CSS, and HTML5. They can then integrate them with hybrid app development frameworks like Ionic, Cordova, or React Native. Both time and cost of development are lower, but you can still upload them to an app store to enjoy the same reach and discoverability.

## Key differences: hybrid apps vs. progressive web apps

Progressive web apps are the result of advancements in browser technologies. Modern browsers let you give your users a native-app-like experience from the web app itself. You can achieve this by integrating a JavaScript framework around your existing web app. Your progressive web app can send notifications via the mobile browser, track user location, and so on. Like hybrid apps, progressive web apps also lie between native and web apps. However, there are some key differences.

### ****Organic reach****

You can deliver both progressive web apps and hybrid apps from app stores. However, progressive apps rank higher in search engine results with no additional effort. In addition, you get better search results than hybrid apps with the same keyword targeting.

### ****Performance****

In most cases, progressive web apps tend to be lighter in size than hybrid apps. They utilize less mobile storage and memory. However, the underlying technologies are non-native. Browser dependency could result in increased mobile battery consumption for users.

### ****Maturity****

Progressive web app technology is relatively new compared to hybrid or web apps. Consequently, developer and community support for progressive apps are still evolving. Hybrid app technology is more mature, and development remains less expensive.

## When to use web apps vs. hybrid apps vs. native apps?

Large companies have to use a combination of native, hybrid, and web apps to reach the widest possible customer base. Before choosing the best app type for your use case, you can consider the following factors.

### ****Time to market****

Early stage start-ups prefer using web apps to release a minimum viable product for customers as soon as possible. Native and hybrid apps require comparatively more time, planning, and effort to launch successfully.

### ****Customer requirements****

Some products and services have large customer bases that use mobile apps regularly to complete tasks. In this case, native applications are preferred over hybrid and web apps.

### ****Marketing strategy****

For some companies, their app development project is often tied closely to their marketing goals. They use a progressive web app to reach the largest possible audience and get initial sign-ups. The web app may have limited functionality or offer full-feature free trials for a limited period. The company then uses native or hybrid mobile apps to enhance the experience of paying customers.

### ****Complexity****

In some instances, the mobile app functionality may be so complex that there is no option but to develop hybrid apps or native apps that support the requirements. For example, mobile banking apps require native features to keep fingerprint authorization features.

## web apps vs. hybrid apps vs. native apps

|  |  |  |  |
| --- | --- | --- | --- |
| **CHARACTERISTICS** | **WEB APPS** | **HYBRID APPS** | **NATIVE APPS** |
| **Usage** | Users can access directly from a browser | Users have to install the app on their device of choice | Users have to install the app on their device of choice |
| **Internal working** | Client code in the browser communicates with remote server-side code and databases | Client code and browser code wrapped in a native shell or container | Client code written in technology and language specific to the device or platform it will be installed on |
| **Native device features** | Not accessible | Accessible | Accessible |
| **User experience** | Inconsistent and dependent on the browser being used | Consistent and engaging | Consistent and engaging |
| **Access** | Limited by browser and network connectivity | One-step access with offline features | One-step access with offline features |
| **Performance** | Slower and less responsive | Faster, but may consume more battery power | Performance can be optimized to device |
| **Development** | Cost-efficient, faster time to market | Cost-efficient, faster time to market | Expensive, slower time to market |

1. **QUESTION 2:Review and compare mobile app programming language**

The world is app-hungry! With over [6.4 billion smartphone users globally (Statista, 2023),](https://www.statista.com/outlook/dmo/app/worldwide" \l "revenue" \t "https://ellow.io/best-programming-language-for-mobile-app-development/_blank)the demand for skilled app developers is exploding. Firms specializing in mobile app development are witnessing a surge, as businesses of all sizes scramble to find talent to craft their digital experiences. But amidst this exciting chaos, a crucial question arises: “Which programming language should you choose to build your app dreams?” Selecting the right language is not just about syntax and frameworks, it impacts your app’s functionality, performance, and ultimately, your career trajectory.  With new languages emerging and existing ones evolving, navigating the vast options can feel overwhelming. Fear not, aspiring developer! This article serves as your guide, delving into the world of mobile app development languages.  We will explore their strengths, and weaknesses, and how they shape your app’s journey, empowering you to make the perfect choice for your coding adventure.

## ****Importance of Choosing the Right Programming Language for App Development****

Choosing the right programming language for your app is like picking the right tool for the job. It impacts everything from how your app works to how smoothly it runs and its success. Here is why selecting the perfect language is crucial:

### ****Fitting In: Platform Acceptance****

Just like you wouldn’t use a hammer on a screw, some languages are best suited for specific platforms. For iOS apps, Swift or Objective-C are the go-to choices, while Ruby, JavaScript, or even web development languages might be better for multi-platform apps.

### 

### **2.1.2. **Building Blocks: Libraries and Frameworks****

Think of [libraries and frameworks](https://ellow.io/best-mobile-app-development-platforms/)as pre-built toolkits. They save you time and effort by providing ready-made solutions for common tasks like database interaction or user interface creation. Choose a language with robust libraries and frameworks that match your app’s needs.

### **2.1.3. **Playing with Others: Third-Party Integrations****

Your app might need to connect with external services, APIs, or other platforms. Make sure the language you choose plays well with others. Check for available APIs, SDKs (software development kits), and a supportive community to help with integrations.

### **2.1.4. **Speed Demons vs. Efficiency Champs: Performance and Efficiency****

Some languages are known for their speed and control, like C and C++, making them ideal for resource-intensive tasks. But don’t forget about development efficiency. Languages like Python or JavaScript might be faster to develop with but may have some performance trade-offs. Find the right balance for your app’s needs.

### **2.1.5. **Specialized Tools: Domain-Specific Capabilities****

Not all languages are created equal. Some excel in specific areas like data analysis (R and MATLAB) or blockchain development (Solidity). If your app deals with a particular domain, consider languages with specialized features and tooling to streamline development.

### **2.1.6. **Growing Your App: Extensibility and Customization****

As your app evolves, you should add new features or functionalities. Choose a language that provides flexibility and customization options. C++ or Java offer features like inheritance and custom annotations, allowing you to create reusable components and even extend the language itself.

## ****Best Programming Languages for Mobile App Development****

### ****Swift****

Developed by Apple, [Swift](https://ellow.io/best-mobile-app-development-platforms/)is the go-to language for ****iOS app****development. Its clean syntax and powerful features make it easy to write and maintain code.

Swift offers safety features like optional and automatic memory management, reducing the chances of runtime errors. Popular apps like ****Airbnb and LinkedIn****use Swift for their iOS versions due to its performance and reliability.

### **2.2.2. **Kotlin****

Kotlin has gained popularity for Android app development due to its interoperability with Java and concise syntax. It offers features like null safety and extension functions, enhancing productivity and code safety. Apps like ****Pinterest and Trello****have embraced [Kotlin](https://ellow.io/hire-developer/hire-kotlin-developers/) for its modern language features and seamless integration with existing Java codebases.

### **2.2.3. **Java****

As a long-standing player in the mobile app development scene, [Java](https://ellow.io/go-vs-java/)remains a robust choice for Android app development. Its platform independence, strong community support, and extensive libraries make it a versatile language. Apps like ****WhatsApp and Instagram****rely on [Java for their Android](https://ellow.io/top-five-java-certifications/)versions due to its stability and scalability.

### **2.2.4. **Dart (used with Flutter)****

Dart, paired with Google’s [Flutter framework](https://ellow.io/best-mobile-app-development-platforms/), enables cross-platform app development with native-like performance. Its Just-in-Time compilation allows for fast development cycles, while Ahead-of-Time compilation ensures efficient production builds. Flutter powers apps like ****Alibaba and Google Ads****due to its fast development pace and expressive UI capabilities.

### **2.2.5. **JavaScript (used with React Native)****

React Native leverages [JavaScript](https://ellow.io/typescript-vs-javascript/)to build cross-platform mobile apps with a single codebase. Its component-based architecture and hot reloading feature accelerate development. Popular apps like ****Facebook and Instagram****use [React Native](https://ellow.io/best-mobile-app-development-platforms/)for their ability to deliver native-like performance and seamless user experiences a [cross platforms](https://ellow.io/cross-platform-vs-native-mobile-development/).

### **2.2.6. **C# (used with Xamarin)****

[Xamarin](https://ellow.io/best-mobile-app-development-platforms/), powered by C#, allows developers to build [native](https://ellow.io/cross-platform-vs-native-mobile-development/)Android, iOS, and Windows apps with a shared codebase. Its strong integration with Visual Studio and access to native APIs ensure high performance and platform-specific functionalities. Apps like ****UPS and Alaska Airlines****rely on Xamarin for its code-sharing capabilities and native user experiences.

### **2.2.7. **Objective-C****

While Swift has largely replaced Objective-C for iOS development, it remains relevant for maintaining legacy codebases. Objective-C offers dynamic messaging and runtime reflection, allowing for flexible app development. Apps like ****Airbnb and Uber****initially relied on Objective-C before transitioning to Swift.

## **2.2.8. **Python****

[Python’s](https://ellow.io/global-companies-that-use-python/)simplicity and versatility extend to mobile app development, particularly with frameworks like Kivy and BeeWare. Its readable syntax and extensive libraries facilitate rapid prototyping and development. Apps like ****Instagram and Dropbox****use Python for [backend services](https://ellow.io/cross-platform-vs-native-mobile-development/)and automation tasks, showcasing its flexibility beyond mobile development.

### **2.2. 9. **HTML/CSS****

Web technologies like [HTML](https://ellow.io/advantages-and-disadvantages-of-html/)and [CSS](https://ellow.io/advantages-and-disadvantages-of-css/), combined with frameworks like Cordova and PhoneGap, enable the building of hybrid mobile apps. Their familiarity and ease of use make them accessible to web developers. Apps like ****Untappd and Untitled Goose Game****employ HTML/CSS for their cross-platform compatibility and rapid development cycles.

### **2.2.10. **Rust****

Rust’s focus on safety and performance makes it a compelling choice for mobile app development, especially for resource-intensive applications. Its ownership model and zero-cost abstractions ensure memory safety and prevent data races. While less widely adopted in the mobile space, Rust is gaining traction for apps like ****Firefox Focus****due to its security and efficiency benefits.

### **2.2.11. **Ruby****

Although primarily associated with web development, Ruby can be used for mobile app development with frameworks like RubyMotion. Its elegant syntax and developer-friendly features promote rapid development. Apps like ****Basecamp and Couchsurfing****have utilized Ruby for its productivity gains and expressive codebase.

### **2.2.12. **Prolog****

Prolog’s declarative programming paradigm makes it suitable for certain types of mobile applications, particularly those involving rule-based logic or expert systems.  Its built-in pattern matching and backtracking capabilities simplify complex problem-solving tasks. While not commonly used for mobile development, Prolog finds applications in niche domains like educational apps and decision support systems.

### **2.2.13. **LISP****

Lisp’s flexibility and metaprogramming capabilities make it a niche choice for mobile app development, particularly in AI and game development. It’s homoiconicity and macro system enable powerful abstractions and domain-specific languages.  Apps like ****Autodesk SketchBook and WolframAlpha****demonstrate Lisp’s potential for innovative and specialized applications.

### **2.2.14. **Malboge****

Malboge, known for its esoteric nature and challenging programming style, is not a practical choice for mainstream mobile app development. Its intentionally obscure syntax and unconventional execution model make it unsuitable for real-world projects. While it may serve as a curiosity or educational tool, Malboge lacks the practicality and support required for mobile app development.

### **2.2.15. **Go****

Go’s simplicity, concurrency support, and fast compilation times make it an appealing option for mobile app development, particularly for backend services and command-line tools. Its built-in garbage collection and static linking simplify deployment and maintenance.  While not as prevalent as other languages in the mobile space, [Go](https://ellow.io/go-vs-java/) is gaining traction for apps like Docker and Kubernetes due to its performance and scalability benefits.

## **2.3. **Conclusion****

When considering the top programming languages for app development in 2024, it is essential to prioritize versatility, platform compatibility, performance, and security. Python stands out for its adaptability, while JavaScript excels in creating both websites and mobile apps. Swift remains the go-to choice for iOS app development, while Java remains strong for [cross-platform](https://ellow.io/cross-platform-vs-native-mobile-development/)and corporate applications. Rust emerges as a formidable option for its combination of performance and security, while Kotlin gains ground for [Android app development](https://ellow.io/best-programming-languages-for-android-app-development/) over Java. To make informed decisions in this competitive landscape, it is crucial to align language choices with project requirements, target platforms, and developer proficiency. Leveraging platforms like [ellow.io](https://ellow.io/)can streamline the process of hiring skilled mobile app developers, ensuring that your projects stay on track and ahead of the curve in this rapidly evolving industry.

1. **QUESTION 3:Review and compare mobile app development frameworks by comparing their key features (language, performance, cost & time to market, UX & UI, complexity, community support)**

**and where they can be used.**

Frameworks are the backbone of mobile app development. They are essential to building dependable apps quickly. When choosing a framework, you can choose between native or cross-platform applications. Though both approaches have advantages, cross-platform applications work on multiple devices regardless of their operating system.

Different companies use different technologies to build and design great apps. Various company requirements have led to emergence of four major frameworks in the market - React Native, Flutter, Ionic, and Xamarin.

If you are considering building a cross-platform application, you must understand the basics of the available frameworks and their unique strengths and features. Doing so will help you choose one that is relevant to you. That is why we have put together this comparison. We will help you understand they key differences between Flutter, React Native, Ionic, and Xamarin.

## ****Comparing the various features of the different frameworks.****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Key features** | **React Native** | **Flutter** | **Ionic** | **Native Script** | **Xamarin** |
| **Language[scripting, procedural]** | [JavaScript, TypeScript] | [Dart, Dart] | [HTML, CSS, JavaScript/ TypeScript] | [JavaScript, TypeScript] | [XAML and/or Xamarin.forms] |
| **Performance** | Can be leveraged to build apps similar to native apps. It allows developers to use native modules for scripting code for complex operations. | Flutter top all of the apps in performance. It offers amazing speed as It uses Dart. | Doesn’t offer performance similar to native apps.  Since ionic tools use web technologies to render the app, it teds to reduce the speed. | Good performance with direct access to native APIs, resulting in native-like performance. | Can be used to develop apps similar to native apps. It focus on business rationale. On the other hand, Xamarin. Forms work with concept of broad code sharing. This may decrease the performance of the code in various operations. |
| **Cost and Time to market** | Cost-effective and offers rapid development with its code reusability and hot reloading features, reducing time to market. | Offers fast development cycles with hot reload, reducing development time and cost. | Cost-effective with quick development cycles, suitable for rapid prototyping and MVP development. | Cost-effective with code sharing capabilities, reducing development time and cost. | Xamarin can be cost-effective in the long run due to code sharing capabilities, although initial development may take longer compared to some other frameworks. However, it offers faster time to market compared to fully native development for multiple platforms. |
| **UX & UI** | Offers native-like UI/UX with access to native components and libraries. | Provides highly customizable UI with its rich set of widgets, enabling developers to create visually appealing and interactive interfaces. | Provides a wide range of UI components for building responsive and visually appealing interfaces. | Offers native UI rendering for a native look and feel, along with access to native components. | Provides access to native UI components and platform-specific APIs, allowing for the creation of highly polished and native-like user experiences. |
| **Complexity** | Moderate complexity, suitable for both simple and complex applications. | Moderate complexity, suitable for both simple and complex applications. | Low to moderate complexity, suitable for simple to moderately complex applications. | oderate complexity, suitable for both simple and complex applications. | Moderate complexity, suitable for both simple and complex applications. Xamarin.Forms, a sub-framework, can reduce complexity by enabling shared UI code across platforms. |
| **Community support** | React Native was created when a developer community was looking for an alternative to the combination of Native React and better mobile application development.  React Native has one of the best communities with over 92.6K stars and 3000+ contributors working to enhance the platform on [GitHub](https://github.com/facebook/react-native" \o "GitHub" \t "https://theonetechnologies.com/blog/post/_blank). Hence, it has a large community of developers across the globe and they are very active on QA sites and forums like Stack Overflow. | Flutter has good community support. It has over 110K stars on [GitHub](https://github.com/flutter/flutter" \o "GitHub" \t "https://theonetechnologies.com/blog/post/_blank), which is very close to React Native. Also, they have 700+ contributors working to enhance the framework and provide the best development experience. Flutter developers are also active on different QA websites. | Strong community support with a large number of plugins and resources available. | Active community support with a range of plugins and resources available. | When it comes to community support, Xamarin has very limited community support among the three platforms. What we think is that maybe the platform is not free, the users have to purchase this cross-platform.  As a Xamarin developer, you will not find instant help and support from the Xamarin community on the Internet. Therefore, on GitHub, the number of users, repository, and contributors are nowhere near the React Native framework. |
| **Use cases** | Ideal for cross-platform app development, especially for applications requiring frequent updates or those targeting multiple platforms. | Suitable for building high-performance, visually rich cross-platform applications, especially those requiring custom UI designs. | Ideal for developing cross-platform applications with simple to moderate UI requirements, such as content-based apps or enterprise applications. | uitable for cross-platform app development, especially for applications requiring direct access to native APIs or requiring a high level of customization. | Ideal for building cross-platform applications, especially for enterprises or businesses already using the Microsoft technology stack. Xamarin is suitable for applications requiring platform-specific features, high performance, and native-like user experiences. |

1. **QUESTION 4:Study mobile application architectures and design patterns**

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

* + 1. **MODEL VIEW CONTROLLER**

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.

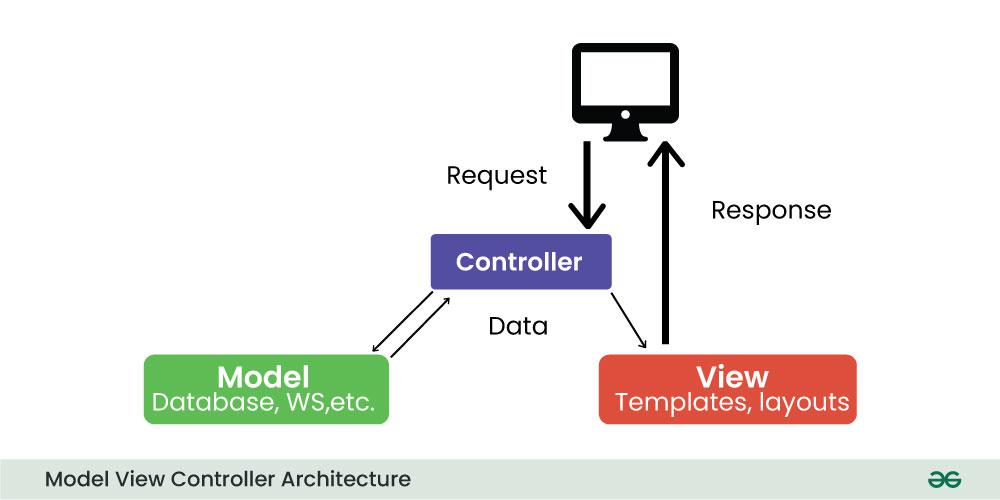


Figure 1: model view controller architecture

* **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).

* + 1. **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.

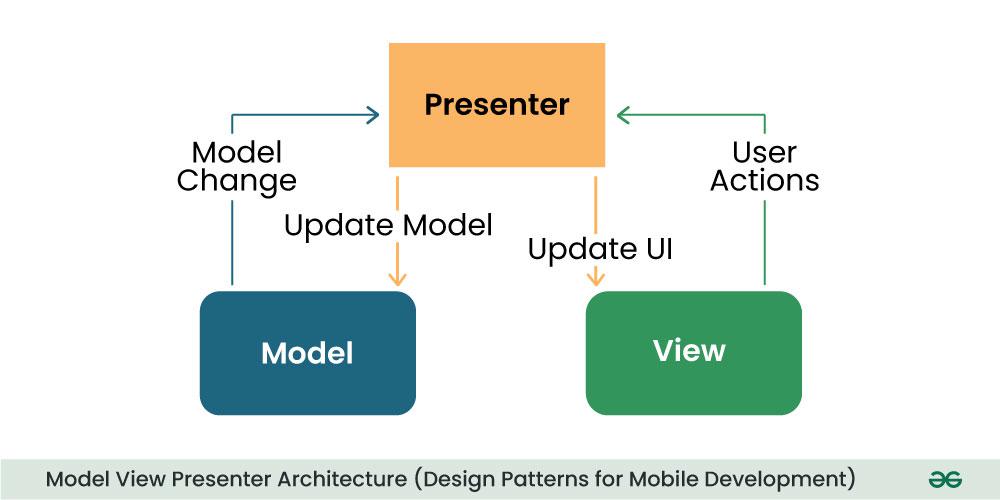


Figure 2: model view presenter architecture

* **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.

* + 1. **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.

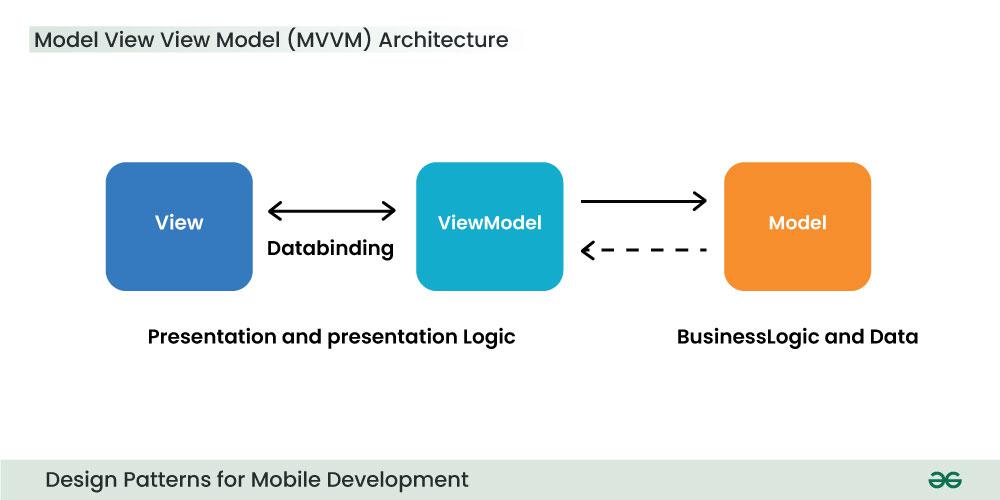


Figure 3: model view viewmodel architecture

* **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.

* + 1. **VIPER ACHITECTURE**

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.

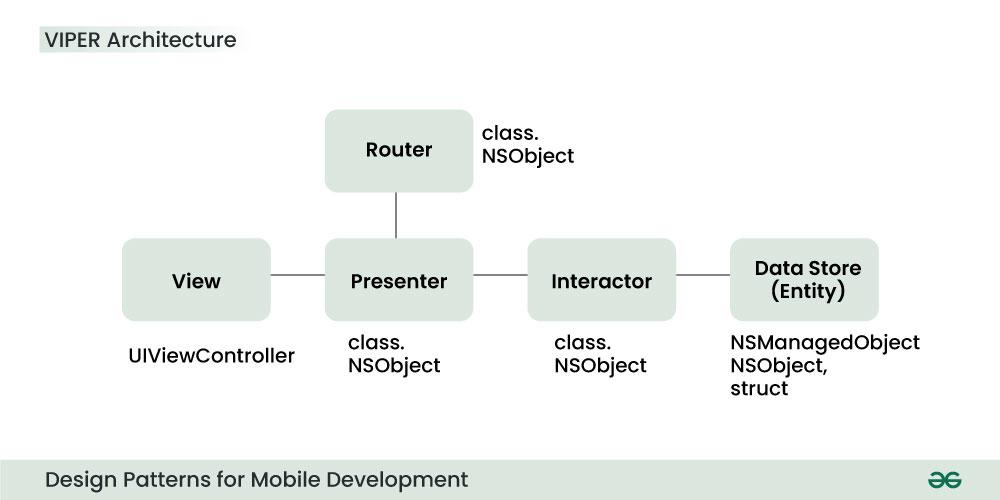


Figure 4: viper architecture

* 1. **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.
  2. **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.
  3. **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.
  4. **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.
  5. **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.
     1. **SINGLETON METHOD 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.

* + 1. **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.

* + 1. **OBSERVER METHOD 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.

## Conclusion

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

1. **QUESTION 5:Study how to collect and analyse user requirements for a mobile application (Requirement**

**Engineering)**

## 

## What Are Mobile App Requirements?

Mobile app requirements document the business logic, technical specifications, and development guidelines for mobile app developers to design the application of your business dreams. It includes the key app’s features, app user personas, and business goals to ensure that multiple team members are on the same page before the software development process commences.

### Mobile App Requirements Benefits

At Requiment, we often see that requirements-gathering for mobile or other applications has [benefits](https://www.requiment.com/benefits/), including:

* A better overall project development
* Improved project management throughout the software development process
* Enhanced efficiency to document the technical aspects required for an optimal product
* A higher probability for the right target audience that meets the business requirements
* Smoother app development with a better user persona and user story
* Higher chance of success in developing the desired features for the target user
* Target users help develop the right user interface to improve user experience
* A competitive advantage over other apps
* The ability to target different user groups in one mobile app requirements document
* Better scalability with all the features designed for a specific market
* App technology that enhances user loyalty and satisfaction
* Cost optimisation by removing unnecessary features

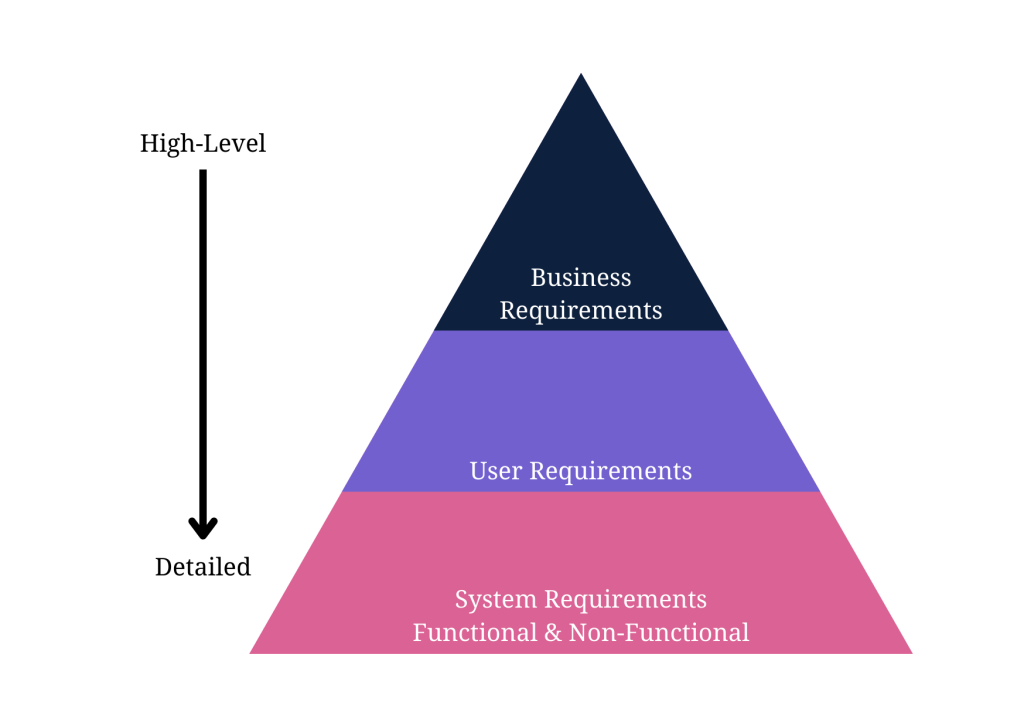
## How to Gather and Document App Requirements

We enable users and fellow mobile app developers to understand and prepare everything necessary to develop a highly competitive application. So, let’s start by sharing the basics and requirements gathering process for multiple team members to hop aboard the development train.

### Mobile App Requirements Types

Mobile apps have different requirement types to collect, including:

* + 1. **Business requirements**  are high-level requirements that ensure the app will align with business objectives, and the project’s scope, and identify the key stakeholders.
    2. **User requirements** are valuable insights into what your target audience needs and wants, how you can solve their problems, and what the audience experiences from your prototyping app.
    3. **Product or system requirements** are non-functional requirements and functional requirements that include technical requirements and technical specifications for the engineering team.

Figure 5. Mobile app requirement pyramid

* 1. **STEP FOR REQUIREMENT GATHERING**

### Define Your App Idea and Purpose

Mobile development requirements-gathering starts with a business idea. The first information you need is the idea or purpose of the mobile app.What purpose will it serve? Does it offer a solution to a potential problem?You need to identify a problem the app will solve to recognise the idea or purpose behind it. [Requirements gathering and management for mobile apps](https://www.requiment.com/requirements-gathering-and-management-for-mobile-apps/) require some effort with massive results.

### Gather and Align the App and Business Objectives or Goals

An app idea is fruitless without understanding business needs, business goals, and business rules.This step encourages you to gather business requirements to understand how the enterprise aligns with the idea from the first step. Gathering business requirements to document involves these steps:

1. Identify the stakeholders for the right mobile application software development based on the business idea.
2. Define clear and concise business goals and objectives to understand the project’s scope.
3. Elicit stakeholder requirements and user requirements with elicitation techniques.
4. Document the requirements in a business requirements document.
5. Validate your requirements with stakeholders for a further transparent and opportunistic process.

So, what requirements-gathering elicitation techniques could you use to gather stakeholder requirements for the business requirements document?. Elicitation techniques work for any requirements-gathering type. Here are some popular and successful techniques to use:

* Analyse similar documents
* Analyse similar external and internal interfaces
* Brainstorm use cases and user stories
* Create user stories and use cases
* Hold stakeholder focus groups
* Host requirements workshops
* Interview all the relevant stakeholders
* Observe documents and case studies
* Prototype visual examples for feedback
* Reverse engineer the processes
* Use online surveys/questionnaires
* Validate ideas with stakeholders

### 

### Run a Market Analysis and Competitor Analysis

Conduct a market or competitor analysis to truly understand the user’s perspective and design the appropriate user personas.

It also helps your team gather more user requirements for the development company.

The following steps explain the process of gathering user and competitor requirements:

1. Identify the direct, indirect, secondary, and substitute competitors for the mobile app. Remember to recognise any businesses offering similar mobile app services or products and those offering different products in a broader niche umbrella.
2. Gather competitor information, including products, descriptions, pricing structures, geographic reach, engaging promotions, target market positioning, business reputation, user profiles, and key partnerships to understand what your product needs to compete against.
3. Use a SWOT analysis to determine your competitor’s strengths, weaknesses, opportunities, and threats. You could learn from another app’s mistakes to improve your requirements and identify possibly unique features other apps don’t provide.

The SWOT analysis in a table will help you see what the new product needs to do to compete better with the top market competitors. Rank each competitor from 1-10 on each key element. Then, rank what your mobile app aims to have in the requirements, looking for opportunities to improve the numbers.

### Determine Scenarios and a User Persona

The next major step in mobile app requirements-gathering is to design user personas and scenarios to guide the requirements. A user persona fictionalises the target users for the mobile app. It should describe the ideal person who uses the app, with some flexible aspects for alternate users.

The ultimate user person could include the following details about target users:

* Age (also, typical generational qualities)
* Behavioural considerations
* Gender (including non-binary if relevant to the product)
* Geographic location
* Goal or problem the app addresses or solves
* Goal quotes or principles
* Goal-related frustrations
* Motivation to use the app
* Range of hobbies and daily activities
* Typical occupation range

How would you transform user personas into scenarios?. Create a persona scenario or storyboard by focusing on the goals, how their typical behaviours affect them, and how the persona’s background motivates them to respond differently. Write the scenario as a short paragraph for starters, as you’ll design user stories later. Meanwhile, [identifying stakeholders for requirements gathering](https://www.requiment.com/identifying-stakeholders-for-requirements-gathering/) means creating personas.

### Gather and Prioritise Functional and Non-Functional Requirements

Your user and business requirements are shaped through the initial steps of app requirements gathering. You still need to document them, but you’ll do that soon enough. Meanwhile, start prioritising the functional and non-functional requirements for the technical details, which also design use cases. [Functional and non-functional requirements](https://www.requiment.com/what-are-functional-and-non-functional-requirements/) differ. First, determine which functional app requirements the project needs. Here are some examples of functional mobile requirements:

* A complete description of a feature the app offers or software interfaces.
* How the app allows users to sign up, verify accounts, or subscribe to a newsletter.
* Buttons and dashboards users interact with to complete a specified task.
* External and internal interfaces users interact with on the app.
* The necessary administrative functions for different user classes.
* Transaction adjustment, correction, and cancellation functions.

Secondly, determine the non-functional requirements necessary to run your app.

Here are some examples of non-functional requirements in mobile development:

* How fast the app responds to user input.
* How the app protects user and business data.
* Whether the app can work on multiple platforms.
* How much data does the app store and is it scalable?
* How reliable and maintainable the app remains.
* Does the app comply with local laws and regulations?

Next, you’ll prioritise the non-functional requirements (NFRs) and functional requirements for an app. Priorities determine the tech stack and importance of each function. The MoSCow prioritisation technique helps with any requirements prioritisation before documenting the requirements. The technique requires you to put every technical requirement into one of four categories:

* **Must Have** – The highest-level requirements are critical to the requirements document to ensure the project’s success.
* **Should Have** – The second highest-level specifications are necessary for the project but won’t delay the progress of development or success.
* **Could Have** – The medium-level specifications could enhance user experience but aren’t dealbreakers if you don’t develop them right away.
* **Won’t Have** – The low-level requirements aren’t important to stakeholders at the time of requirements documentation and won’t affect the development process.

Include requirements for user experience (UX) and user interface (UI) with your functional and non-functional requirements. It helps to have these requirements in place before designing use cases and documenting a key user experience and user flow requirement for app development. [Requirements prioritisation](https://www.requiment.com/requirements-prioritisation-making-informed-decisions/) simplifies decision-making for your app.

### Design Use Cases and User Stories

A mobile system requirements document won’t be complete unless you add use cases and stories. Design them before documenting the specifications for the development company or team. Use stories and cases to add visual representation to your documents for mobile app development documentation.

#### How to Design a Use Case for App Requirements Documents

A use case diagram lets everyone at the development company visualise an overview of how users will interact with the app. It’s an overview that includes actors, how actors interact with the app, and the sequence of interactions actors will deploy. Here’s a simple example of a use-case diagram that shows an overview of how actors interact with functional features while non-functional features interact with the app from the back-end stack:

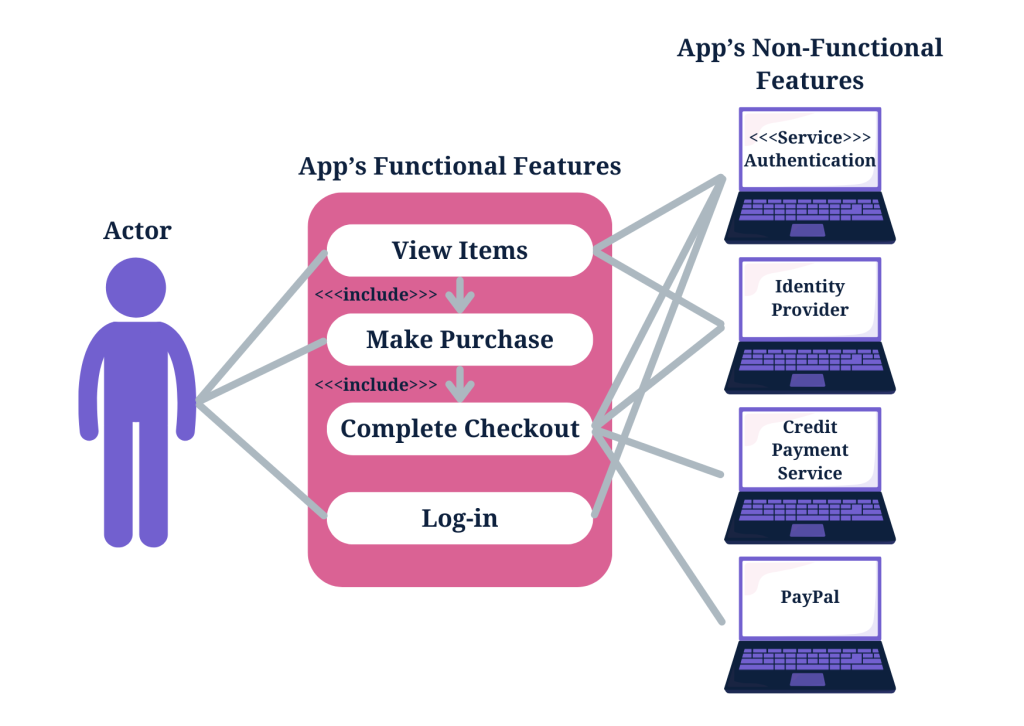


Figure 6: how to design usecase

#### How to Design a User Story for App Requirements Documents

A story evolves the user persona you wrote earlier. Creating user stories lets the development company design an app that meets acceptance criteria. Transforming a user persona into a story means you must follow the INVEST acronym to guide the criteria. In other words, every user story must meet the following factors:

* **Independent**– stories must be independent from each other.
* **Negotiable**– The what and why of stories should be concrete, while the how shouldn’t be.
* **Valuable** – Stories should add value for all the stakeholders.
* **Estimable** – Each story should be capable of estimation.
* **Small**– Stories must be small enough for sprint completion.
* **Testable** – Stories must be testable with written acceptance criteria possible immediately.

Here’s a criteria example that depicts a good user story:

1. A business logo displays when users load the application.
2. Animation includes enlarging the logo until it fills the screen and a slow sideways transition to the next screen.
3. The user clicks on a dropdown menu on the top-right tab to search the categories, which include information about the company, shopping categories, FAQS about payment, a help guide, and a Contact Us page as some key features.
4. The user clicks on the shopping categories to open a new dropdown menu that includes hats, shirts, pants, and shoes.
5. Users click on hats, and the screen changes to show the products.
6. The animation includes an enlarging logo with a sideways screen transition.
7. Users click on the add to cart buttons when seeing a desired product.
8. The button fills with a tick, as the top screen shows a banner that the item was added to the cart.
9. The user clicks on the top-left dashboard button to enter the cart.

The criteria can go on but keep it short, as you’ll use different user stories, including one where the user checks an item out of their cart and makes payment.

### Write an App Requirements Document

Delivering a proper app requirements document means you need to know how to write a mobile app requirements document.Mobile application development relies on the requirements document to design proper flow or the best app features and hit the right target audience.

**HOW TO WRITE A MOBILE APP DEVELOPMENT DOCUMENT**

#### Step 7a: Formulate the App’s Idea Statement

Every app requirements document should include an idea statement that lets every stakeholder and software developer understand the document before diving into the details. Start your app requirements document with a simple single-sentence statement that aligns with the app’s idea.

#### Step 7b: Document All Relevant App Details

A detailed description of development plans in the requirements document is instrumental to completing the documentation. A successful mobile app requirements document includes more than the details of an application and its functions for the development team.

##### Descriptions to Include in a Mobile Requirements Document

A mobile application requirements document should include a detailed description of functional and technical requirements and the app’s functionality to properly capture and represent the project’s scope.

* Business requirements
* User requirements
* Software requirements specification
* Technical specifications
* Functional specifications
* Non-functional requirements
* Hardware interfaces
* A list of must-have features
* Unique app features
* Internal and external interfaces
* Non-functional key metrics
* User stories
* Acceptance criteria

#### Step 7c: Prepare a Navigation Sequence

The development team requires a simple navigation sequence they can follow during software development. The mobile app requirements document outlines the sequence in which the software development process flows. Add the details from step 7b in a sequence that every developer can easily grasp.

#### Step 7d: Add Requirements Formats for Visuals

Add your user stories from a user’s perspective and the use case overviews you designed to the app requirements document to help stakeholders and the development team understand every aspect of the app requirements document. Successful software development means knowing the intended users.

#### Step 7e: Add Cost Optimisation Details

The development team, stakeholders, and the client will appreciate a cost-benefit analysis to ensure cost optimisation throughout the software development process. You’ll find guidelines for cost optimisation in your business objectives or budget. Business analysts also insist on adding a cost-benefit analysis to an app requirements document to meet the business needs and have a greater chance of success against competing apps. Did you know that [poor requirements management can lead to projects going over budget](https://www.requiment.com/why-poor-requirements-management-can-lead-to-projects-going-over-budget/)?

#### Step 7f: Add Communication Protocols

Ensure another key element is in your document before delivering the mobile app requirements document to a project manager, stakeholder, developer, or operating environment. Add communication methods for a collaborative process. Collaboration relies on dependable communication. The [importance of effective communication in requirements gathering](https://www.requiment.com/the-importance-of-effective-communication-in-requirements-gathering/) outlines why you need it.

#### Alternate App Requirements Document Process

Alternatively, use a mobile app requirements document template to capture the requirements for app development. A mobile app development requirements document template speeds up the process for non-business analysts or requirements analysts. The requirements document focuses on product or system requirements to develop an app. It’s a basic app requirements document template that app stores and development teams can follow.

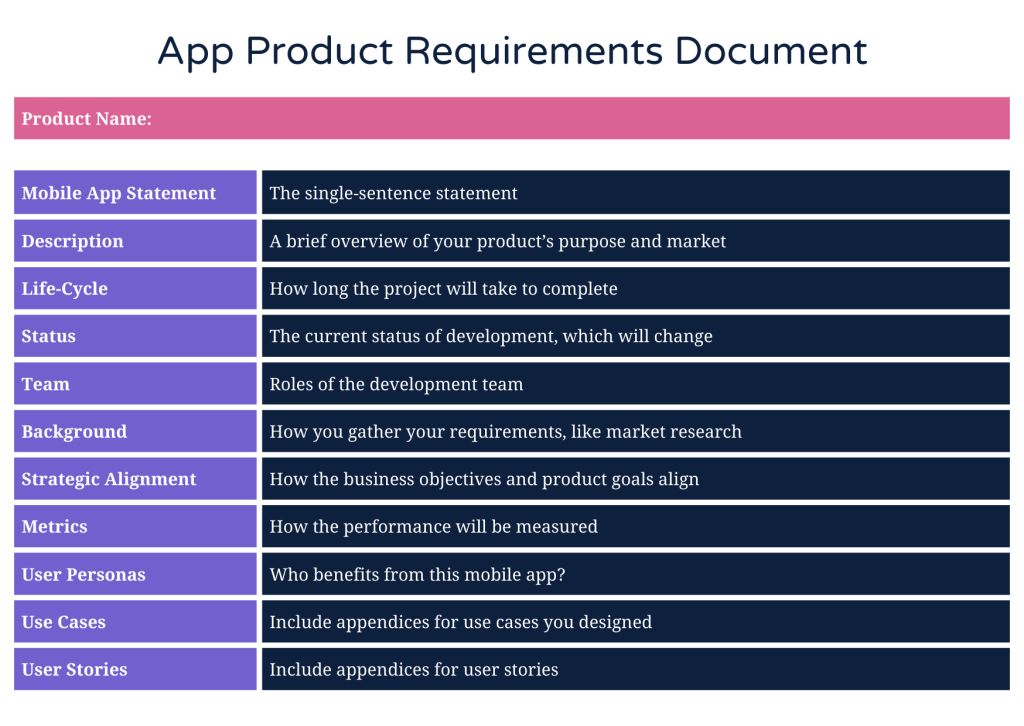
****

Figure 7: app product requirement document.

### Deploy Prototyping and Wireframing

Prototyping and wireframing let you design the user flow of user interfaces and basic app functions. It also lets you test and validate layouts and transitions between app pages. Here are the steps to wireframe your requirements for an app, which you will then validate in the next step:

1. Map the target user flow.
2. Sketch the flow’s core part.
3. Set a mobile wireframe.
4. Determine the layout with boxes.
5. Use design patterns.
6. Add intended copy.
7. Connect the app’s pages to design a flow.
8. Design a prototype.
9. Release the initial design to gather feedback.
   * 1. **Validate the App Requirements**

Validation is a quality control process you use before launching the final product based on your requirements. The prototype app collects feedback from stakeholders, and you can invite stakeholders to verify that the app meets the documented requirements. Use the feedback for the final step. The [benefits of collaborative requirements gathering](https://www.requiment.com/the-benefits-of-collaborative-requirements-gathering/) share insights about feedback and collaboration.

### Apply Agile Methodology

Agile methodology in requirements-gathering means you’ll always adapt the requirements document as per the feedback from stakeholders, testing, and initial product releases. Agile methodology focuses on user experience and constant testing and validation to further improve your application. You should know the [importance of updating requirement documentation throughout a project](https://www.requiment.com/the-importance-of-updating-requirement-documentation-throughout-a-project/).

**Conclusion**

Gathering the right requirements for a mobile development requirements document follows ten steps, some containing sub-steps with easy-to-follow guidelines.

1. **QUESTION 5:Study how to estimate mobile app development cost.**

App development costs can vary between 16 000 and 500 000 US dollars, according to different sources. The cost depends on multiple factors, like requirements, development methods used and local pricing. In the end, app development can cost practically anything as the pricing is up to the seller, and is affected with buyers’ various needs.

Simply put, you could calculate app development costs with a simple multiplication:

**Total App Development time(in hours) x Hourly Rate.**

However, that doesn’t take into account all other costs like maintenance costs which are a major cost factor during the app’s lifecycle.

**HOW TO ESTIMATE MOBILE APP DEVELOPMENT COST.**

Estimating the cost of mobile app development involves several factors and considerations

Estimating the cost of mobile app development involves several factors and considerations. Here's a guide on how to effectively estimate the cost:

* 1. **Define Requirements:** Clearly define the requirements of the mobile app including features, functionalities, platforms (iOS, Android, or both), integrations, user interface design, and any other specific needs.
  2. **Break Down Features:** Break down the requirements into smaller features or modules. This helps in estimating the cost more accurately by assessing the complexity and effort required for each component.
  3. **Research Development Rates:** Research and gather information on the average development rates in the region or country where you intend to hire developers. Rates may vary based on the level of expertise, location, and complexity of the project.
  4. **Choose Development Approach:** Decide whether to build the app in-house, hire a development agency, or outsource the project to freelancers. Each approach has its own cost implications, so choose the one that best fits your budget and requirements.
  5. **Estimate Development Hours:** Break down the development process into tasks and estimate the number of hours required for each task. Consider factors such as coding, testing, debugging, design, project management, and deployment.
  6. **Consider Additional Costs:** Factor in additional costs such as app store fees (for publishing the app), third-party integrations, maintenance, updates, hosting, marketing, and support services.
  7. **Account for Contingencies:** Include a contingency buffer in your cost estimation to account for unforeseen challenges, changes in requirements, or delays in the development process.
  8. **Review Similar Projects:** Review similar projects or mobile apps in the market to get an idea of the costs involved. This can provide valuable insights into the typical budget required for a project of similar scope and complexity.
  9. **Get Quotes:** If you're outsourcing the development, obtain quotes from multiple vendors or development agencies. Compare the quotes based on the scope, quality, timeline, and cost to make an informed decision.
  10. **Create a Budget Plan:** Based on the estimates and quotes gathered, create a detailed budget plan outlining the expected costs for each phase of the development process.
  11. **Factor in Post-launch Costs:** Don't forget to consider post-launch costs such as maintenance, updates, marketing, and ongoing support. These costs are essential for ensuring the long-term success and sustainability of the mobile app.
  12. **Review and Adjust:** Periodically review and adjust your cost estimates as the project progresses, taking into account any changes in requirements, scope, or market conditions.

By following these steps and considering various factors, you can effectively estimate the cost of mobile app development and plan your budget accordingly. Also it is good to keep in mind that cost estimation is an iterative process, so be prepared to refine your estimates as you gather more information and progress through the development lifecycle.

1. **References:**
2. ) Vaishnavi Jonna, (February 23th, 2024). Review and compare mobile app development frameworks by comparing their key features. <https://ellow.io/best-programming-language-for-mobile-app-development/>
3. ) Kiran Beladiya, (Jun 23, 2023). Review and compare mobile app development frameworks by comparing their key features. <https://theonetechnologies.com/blog/post/flutter-vs-react-native-vs-xamarin-top-cross-platform-mobile-app-development-framework>
4. ) Girish R(not known). Review and compare mobile app development frameworks by comparing their key features. <https://www.fingent.com/blog/react-native-flutter-ionic-xamarin-a-comparison-between-the-top-mobile-app-development-frameworks/>
5. ) mobidev.biz (March 27, 2024). Review and compare mobile app development frameworks by comparing their key features. <https://mobidev.biz/blog/cross-platform-mobile-development-frameworks-comparison>

) Anujpato5vy ( 2023) . mobile application architectures and design patterns. <https://www.geeksforgeeks.org/design-patterns-for-mobile-development/>

1. ) Catherine Beaton (Feb, 18 2024) . how to collect and analyse user requirements for a mobile application. <https://www.requiment.com/requirements-gathering-and-management-for-mobile-apps/>
2. )Tuhin Bhatt (March 18 2024). Study how to estimate mobile app development cost. <https://www.intelivita.com/blog/app-development-cost/>
3. )aws (). Review and compare the major types of mobile apps and their differences. <https://aws.amazon.com/compare/the-difference-between-web-apps-native-apps-and-hybrid-apps/>