REPUBLIC OF CAMEROON

PEACE - WORK – FATHERLAND

MINISTRY OF HIGHER EDUCATION

REPUBLIQUE DU CAMEROUN

PAIX – TRAVAIL – PATRIE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

CEF440

INTERNET PROGRAMMING AND MOBILE PROGRAMMING

BY

GROUP 25

NAME	MATRICULE
FOFIE FOPA ELISABETH	FE21A189
MBISHU FABRICE YENVEN	FE21A232
EDI EDISON FORNANG	FE21A178
TAKOH CLOVERT CLOVERT NFUA	FE21A311
KIMBI CYRIL BONGNYU NFOR	FE21A216

COURSE INSTRUCTORS: DR NKEMINI Valery **APRIL 2024**

Contents

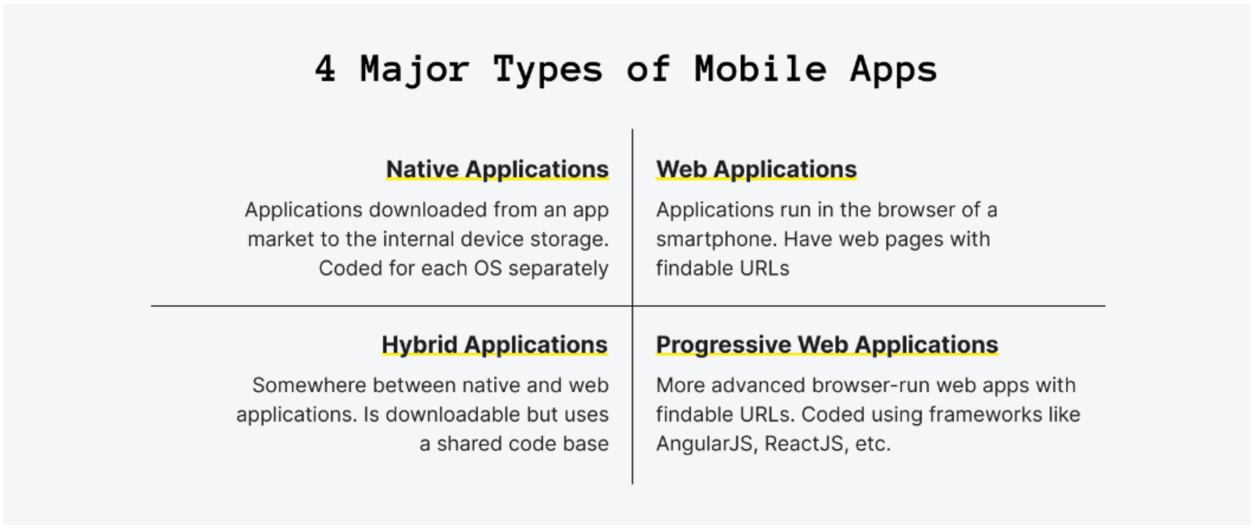
1. MAJOR TYPES OF MOBILE APPS AND THEIR DIFFERENCES (NATIVE, PROGRESSIVE WEB APPS, HYBRID APPS)

A mobile application is a kind of software that is created for being run on a mobile device. This includes not only smartphones but also tablets and smartwatches.

There exist various types of mobile applications which differ by purpose and the ways they operate on mobile devices. Shortly mentioning the latter, mobile apps may be coded differently, for instance, depending on the operating systems (OS) of the devices they'll be used on or if they'll be launched in browsers instead of being downloaded to the device's internal storage.

Accordingly, the programming languages and frameworks to create mobile applications vary as well. For instance, Flutter, React Native, and Ionic are considered among the best mobile app development frameworks.

A cross-platform app is an application that works on multiple platforms, such as Android, iOS, or Windows.



4 Major Types of Mobile Apps

Native Applications
Applications downloaded from an app market to the internal device storage. Coded for each OS separately

Web Applications
Applications run in the browser of a smartphone. Have web pages with findable URLs

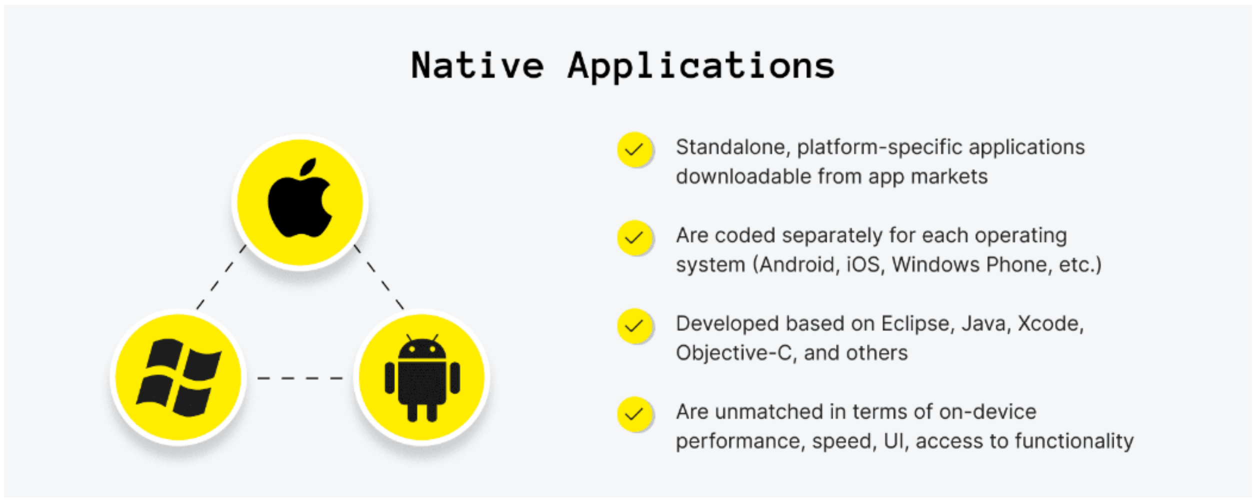
Hybrid Applications
Somewhere between native and web applications. Is downloadable but uses a shared code base

Progressive Web Applications
More advanced browser-run web apps with findable URLs. Coded using frameworks like AngularJS, ReactJS, etc.

1.1 Native Applications

Native mobile apps are applications downloaded from an app market to the internal device storage. They are usually coded for each device separately.

When building native apps developers have to code the app at least twice for Android and iOS so that the audience would have the chance to get it. Twice because the two most popular mobile OS are based on different programming languages, so the code is not reusable.


Native Applications
Standalone, platform-specific applications downloadable from app markets
Are coded separately for each operating system (Android, iOS, Windows Phone, etc.)
Developed based on Eclipse, Java, Xcode, Objective-C, and others
Are unmatched in terms of on-device performance, speed, UI, access to functionality

The best-fit software programming languages for native application development fully depend on the operating systems and their mobile application requirements. For instance, Android applications are usually developed based on Eclipse and Java while iOS apps use Xcode and Objective-C. But then again there are specially designed SDKs (software development kits) and frameworks that can help speed things up.

1.1.1 Native Application Advantages

- On the bright side, because native applications are standalone and platform-specific, they are practically unrivaled in terms of usability and compatibility.

- They can boast unmatched on-device performance, advanced security, the capability of working offline, tend to have fewer bugs, and are tailored to make the most of each device's native UI.

- Plus, they have no roadblocks to accessing the smartphone's hardware and features like the microphone, GPS, or camera.

- Since the application may be up on app markets, this could mean additional discoverability.

- It is possible to charge users for downloading the app, providing a monetization path.

1.1.2 Native Application Disadvantages

- These types of mobile applications are quite costly to develop. Launching a native app requires additional investment and possibly time. And their maintenance won't come easy nor cheap either.

- After being coded for separate OSs, these apps have to go through the process of approval to become available on the app markets. This may also be time-consuming and can possibly delay your mobile app launch on the market.

-The step that follows launch on the app markets may be even harder as you'd need to make your users manually install the app on their devices to use it.

- There’s no guarantee that users won't delete the app after installing if it isn't used regularly. This is a common scenario since native applications are usually weighty, and not everyone likes their smartphone's internal memory to be clogged with applications they might not use often.

- Native applications require users to make updates. Whenever the application gets a new version rolled out even with minor changes, the users must make corresponding updates on their mobile devices. At times, they'll lose access to the app completely until the update is made.

Hence, opting for classic native application creation might not be the right path for all businesses or ideas.

1.1.3 Examples of Native Application

Google Play or AppStore, comprises of dozens of mobile apps categories to choose from. Among them, there are plenty of native applications available for download. WhatsApp Messenger is a native app example.

1.2 Hybrid Applications

Hybrid mobile apps are applications that combine web apps with the native capabilities of a platform. They are developed in one go for all platforms, which means less time and money spent on cross-platform development. This brings us to one more mobile application type.

Hybrid mobile apps are similar to the cross-platform mobile app development concept, hybrid application development values the reusability of code and accessibility on numerous devices. This solution is somewhat of a middle ground between native applications and web apps.

Hybrid Applications
Flutter
Native Script
HTML
CSS
JS
</>
Is somewhat of a middle ground between native applications and web apps
Have one code base for multiple OS
Developed using CSS, HTML, and Javascript and are then wrapped into the platform-specific app containers. Frameworks like Flutter and NativeScript are also common

They are developed using web technologies like CSS, HTML, and Javascript and are then wrapped into the platform-specific app containers (meaning that they integrate well both with iOS and Android). Hybrid applications are also commonly developed using specially designed mobile application development frameworks like Flutter or React Native.

1.2.1 Hybrid Applications Advantages

- Cost-effectiveness and an accelerated time to market are the major benefits of this mobile app type compared to native applications.

- We don't have to work on separate mobile apps in terms of iOS and Android code. Hybrid apps have one code base for multiple OS, so maintenance is simple. This partially explains why hybrid applications are currently gaining popularity and why they've become one of the mobile application trends.

- Hybrid apps exist in the form of downloadable applications from app markets, so technically they're sort of like native apps.

- They can be accessed offline or with a bad internet connection (but only to a limited extent).

- They're capable of supporting the built-in smartphone hardware and features (just like native applications do but aren't compatible with all the functionality).

1.2.2 Hybrid Applications Disadvantages

- Their speed may be dependent on the browser (so, regular native apps usually operate faster).

- They aren't as interactive and quick as native applications.

- Hybrid applications might not perform flawlessly if the application is too complex since this requires more power and speed.

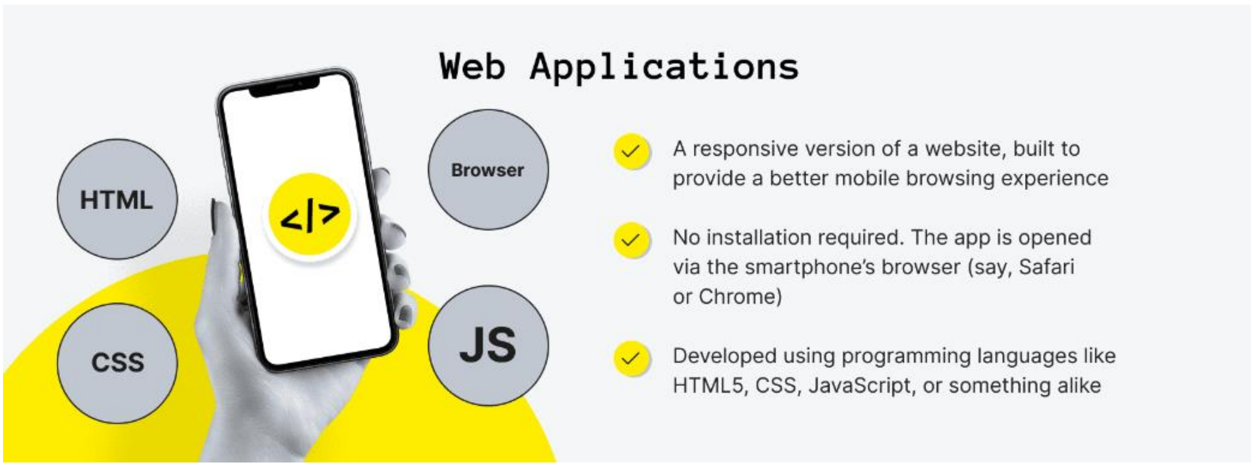
1.2.1 Hybrid Applications Examples

There are tons of hybrid applications today, including Instagram, Gmail, Facebook, and many others.

1.3 Web Applications

Web apps are a responsive version of a website, built to provide a better mobile browsing experience when compared to using the website's regular mobile version.

A web app makes a website responsive, it is launched via the browser (so you don't have to force users to install the app to the device to use it like in the case with native apps). Mobile websites are the version of a website that people see when opening a site from a mobile browser, but they are commonly unoptimized, not user-friendly, slow, and look like the shrunk desktop website version.


Web Applications
HTML
CSS
Browser
JS
</>
A responsive version of a website, built to provide a better mobile browsing experience
No installation required. The app is opened via the smartphone's browser (say, Safari or Chrome)
Developed using programming languages like HTML5, CSS, JavaScript, or something alike

Once again, web applications are basically websites that are designed to be responsive and work well on a mobile device. Hence, they are developed using programming languages like HTML5, CSS, JavaScript, or something similar.

1.3.1 Web Applications advantages

- Cross-platform and cross-device accessibility using a smartphone browser without installation.

- Increased user satisfaction with their mobile experience (if compared to opening a static website version that's poorly optimized for mobile).

- No internal device storage nor updates are required on the user's end.

- Not missing out on SEO (because web applications have URLs like any website, search bots see the pages separately and people can find them via a fast Google search or direct URL address input).

- They are cheaper to develop than native mobile applications.

1.3.2 Web Applications Disadvantages

- If the users fail to have decent access to the internet, they won't have the chance to enjoy the web app to the fullest until the connection is back.

- When a user closes the web page, they'll have to find a way to reopen it in the browser and return to the website again (to compare, native applications are downloaded to the device and are accessible in one tap).

- There may also be additional restrictions with such types of apps due to limited hardware accessibility (such as with sending push notifications).

- Web applications might be far from perfect in terms of UX, security, and speed.

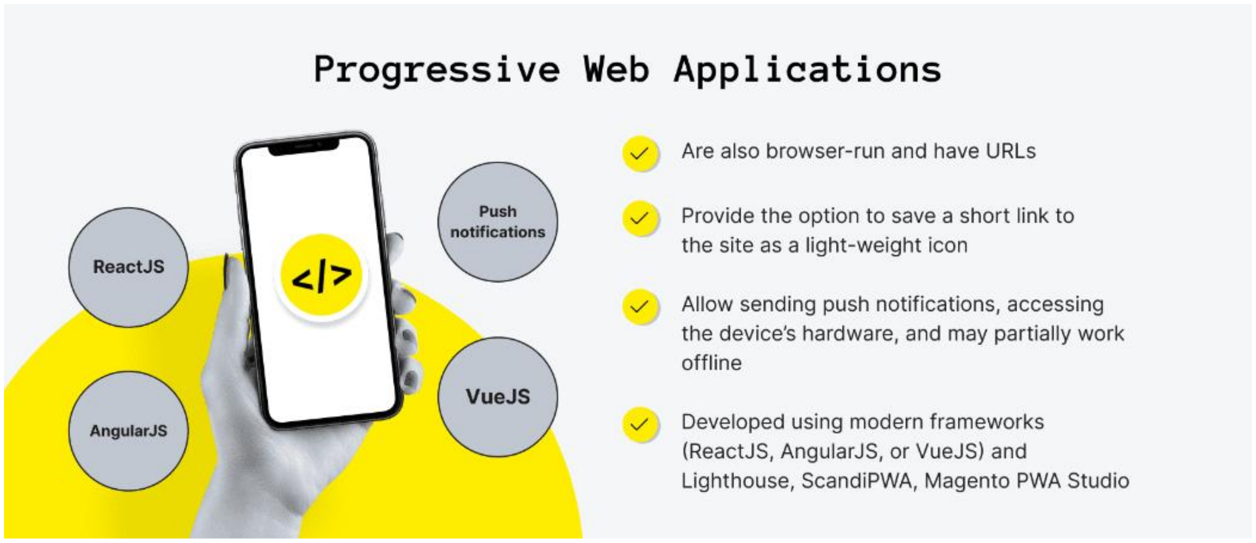
1.3.3 Progressive Web Applications examples

Numerous websites have chosen custom web application development and made web apps to improve the mobile experience of their users, including Trello.

1.4 Progressive Web Applications

Progressive Web Apps (PWA) are web apps that simulate the behavior of native mobile apps (they can send notifications, use the camera, ask for location, etc.) They are typically accessed via a web browser of your choice but can be installed and accessed on a device. PWAs are radically accessible, making them a great choice for dynamic eCommerce solutions.

Web application development is constantly moving forward, and the progressive web application (PWA) solution emerged not too long ago. This web-based technology overcomes many of the previously existing web app issues (including the ability to be used offline or send push notifications).


Progressive Web Applications
ReactJS
AngularJS
Push notifications
VueJS
</>
Are also browser-run and have URLs
Provide the option to save a short link to the site as a light-weight icon
Allow sending push notifications, accessing the device's hardware, and may partially work offline
Developed using modern frameworks (ReactJS, AngularJS, or VueJS) and Lighthouse, ScandiPWA, Magento PWA Studio

Progressive web apps are built using modern frameworks like React.js, Angular.js, or Vue.js. Lighthouse, ScandiPWA, and Magento PWA Studio are also commonly used for their development.

1.4.1 Progressive Web Applications advantages

- PWAs are also browser-run and have URLs (so the pages can be promoted in terms of SEO and found via the web search).

- They put page loading speed in the spotlight (while native apps still perform faster, PWAs use optimized rendering to accelerate the page loading times).

- Progressive web apps are immersive and may resemble the look of native applications.

- They also provide the option to save a short "Add to Home Screen" link to the site (and this link will look like any other app icon on the smartphone's screen). Importantly, these bookmarks are low-weight.

- These types of mobile applications allow sending push notifications, accessing the device's camera and other functionality, and may partially work offline.

- Notably, when rebuilding a website into a PWA, the desktop version of the website can get optimized as well.

- Progressive web apps cost less to develop when compared to native applications.

1.4.2 Progressive Web Applications Disadvantages

- These apps aren't available in app stores and can be saved to the home screen only after the website was accessed via a browser.

- PWA technologies are advancing, yet there is still room for improvement, including offline use and device functionality access limitations on iOS.

But the main question is can a PWA be considered a mobile app substitute? Perhaps, not. They target different audience groups and have different aims.

1.4.3 Progressive Web Applications examples

The implementation of progressive web apps has been especially seen in the e-commerce sector recently. Many renowned companies including Starbucks have converted their websites into PWAs.

2. MOBILE PROGRAMMING LANGUAGES

Businesses of all types are increasingly turning to Indian firms that specialize in mobile app development to find qualified personnel to design and build their mobile applications. With so many programming languages out there, it’s important to pick the one that works best for creating apps.

Choosing the proper programming language is crucial when creating apps. It determines how easy it will be to create and manage your application as well as its effectiveness, scalability, and performance. Due to the advent of new languages and the development of existing ones, it can be challenging to navigate through the multitude of alternatives available.

Selecting the proper programming language is crucial when creating apps for a variety of reasons. Your app’s functionality, performance, scalability, maintainability, and developer productivity will all be significantly impacted by the programming language you choose.

2.1 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:

1. 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. Building Blocks: Libraries and Frameworks

Think of libraries and frameworks 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.

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.

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.

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.

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.

2.2 Best Programming Languages for Mobile App Development

1. Swift

Developed by Apple, Swift 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. 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 for its modern language features and seamless integration with existing Java codebases.

3. Java

As a long-standing player in the mobile app development scene, 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 versions due to its stability and scalability.

4. Dart (used with Flutter)

Dart, paired with Google's Flutter framework, 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.

5. JavaScript (used with React Native)

React Native leverages 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 for their ability to deliver native-like performance and seamless user experiences a cross platforms.

6. C# (used with Xamarin)

Xamarin, powered by C#, allows developers to build native 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.

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.

8. Python

Python's 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 and automation tasks, showcasing its flexibility beyond mobile development.

9. HTML/CSS

Web technologies like HTML and 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.

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.

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.

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.

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.

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.

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 is gaining traction for apps like Docker and Kubernetes due to its performance and scalability benefits.

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 and corporate applications. Rust emerges as a formidable option for its combination of performance and security, while Kotlin gains ground for Android app development over Java.

3. MOBILE DEVELOPMENT FRAMEWORKS

Mobile development frameworks are essential tools for developers to create applications for mobile devices efficiently. These frameworks provide a structured approach to app development, offering pre-built components, libraries, and tools to streamline the process. some popular mobile development frameworks:

3.1 React Native

Developed by Facebook, React Native enables developers to create mobile apps using JavaScript and React, facilitating cross-platform development for iOS and Android from a single codebase. Additionally, React Native provides a platform for web application development.

In recent years, Expo has emerged as a lightweight variant of React Native, offering a toolchain and platform that simplifies the development, building, and deployment processes. Expo streamlines React Native development by providing pre-configured tools, a development environment, and access to native APIs, making it accessible to developers of varying skill levels.

With Expo, developers can leverage over-the-air updates, pre-built UI components, and a client app for quick project previews on iOS and Android devices, further enhancing the efficiency and accessibility of React Native app development.

3.2 Flutter

Developed by Google, Flutter is an open-source UI software development kit designed for crafting natively compiled applications across mobile, web, and desktop platforms, all from a unified codebase.

Powered by the Dart programming language, Flutter offers a comprehensive suite of customizable widgets, enhancing the developer experience and expediting the creation of visually appealing user interfaces..

3.3 Xamarin

Developed by Microsoft, Xamarin is a framework that allows developers to build cross-platform mobile apps using C#. It leverages the .NET framework and provides access to native APIs, offering high performance and native-like user experiences.

3.4 Ionic

Ionic is an open-source framework that uses web technologies such as HTML, CSS, and JavaScript to build cross-platform mobile apps. It utilizes AngularJS for application logic and provides a library of pre-designed UI components.

3.5 PhoneGap / Apache Cordova

PhoneGap, now known as Apache Cordova, is a framework that enables the development of mobile apps using web technologies. It wraps HTML, CSS, and JavaScript code into a native container, allowing developers to access device features through plugins.

3.6 NativeScript

Developed by Progress, NativeScript is an open-source framework for building native mobile apps using JavaScript or TypeScript. It provides access to native APIs and UI components, allowing developers to create truly native experiences.

3.7 SwiftUI

Introduced by Apple, SwiftUI is a framework for building user interfaces across all Apple platforms, including iOS, macOS, watchOS, and tvOS. It enables developers to create declarative UIs using Swift, reducing boilerplate code and enhancing productivity.

3.8 Kotlin Multiplatform Mobile (KMM)

Kotlin Multiplatform Mobile is a framework developed by JetBrains for sharing code between iOS and Android platforms using Kotlin. It allows developers to write business logic once and share it across multiple platforms, improving code consistency and reducing duplication.

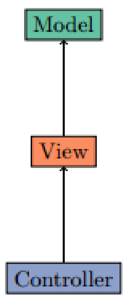
These frameworks offer different approaches and features, catering to the diverse needs of mobile app developers. The choice of framework often depends on factors such as project requirements, developer preferences, platform support, and ecosystem maturity.

4 MOBILE APPLICATION ARCHITECTURES AND DESIGN PATTERNS

Mobile application architectures and design patterns are crucial elements in creating robust, maintainable, and scalable mobile apps. They provide a blueprint for organizing code, managing complexity, and ensuring that the app meets both functional and non-functional requirements. Here are some common mobile application architectures and design patterns:

4.1 Model-View-Controller (MVC)

The Model-View-Controller architecture separates an application into three interconnected components. The Model represents the data and business logic, the View handles the user interface, and the Controller acts as an intermediary, handling user input and updating the model accordingly. MVC promotes code reusability and separation of concerns, making it a widely-used design pattern in mobile development.



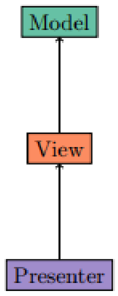
4.2 Model-View-ViewModel (MVVM)

MVVM extends the MVC pattern with a stronger separation of concerns. The Model remains responsible for data and business logic, while the View is responsible for UI rendering.

The ViewModel acts as a mediator between the View and Model, providing data and behavior to the View. This pattern is particularly popular in frameworks like Xamarin and Angular for its support of data binding and testability.

4.3 Model-View-Presenter (MVP)

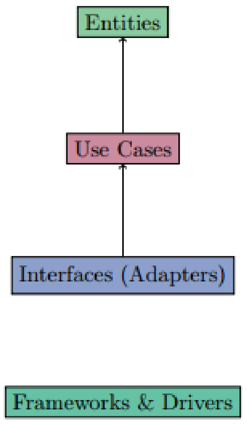
MVP is another variation of MVC, emphasizing a passive view. The Model represents the application's data and logic, while the View renders the UI and forwards user input to the Presenter. The Presenter acts as an intermediary between the View and Model, updating the View based on user interactions and changes in the Model. MVP enhances testability and separates concerns, making it suitable for large-scale applications.



4.4 Clean Architecture

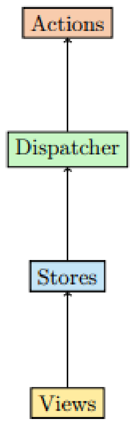
Clean Architecture, proposed by Robert C. Martin, emphasizes separation of concerns and layers with clear dependencies.

The architecture is divided into layers, including Entities representing business objects, Use Cases containing application logic, Interfaces (Adapters) connecting inner layers with external frameworks, and Frameworks & Drivers containing implementation details. Clean Architecture promotes scalability, testability, and maintainability of mobile applications.



4.5 Flux Architecture

Developed by Facebook for React applications, Flux is an architectural pattern that emphasizes unidirectional data flow. It consists of four main components: Actions representing user interactions or events, a Dispatcher dispatching actions to stores, Stores containing application state and logic, and Views reacting to changes in stores and updating the UI. Flux provides a predictable state management solution suitable for complex, interactive applications.



4.6 Redux

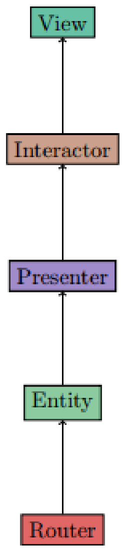
Redux is a predictable state container commonly used with React applications. It centralizes the application state in a single store and uses pure functions called reducers to update the state in response to actions.

Redux simplifies state management and enhances predictability by enforcing a single source of truth for application state.

4.7 VIPER (View, Interactor, Presenter, Entity, Router)

VIPER is an architectural pattern derived from Clean Architecture, with additional components for handling navigation and user interactions.

It divides the application into five layers: View for UI rendering, Interactor for business logic, Presenter mediating between View and Interactor, Entity representing data models, and Router managing navigation between modules or screens. VIPER promotes modularity, testability, and maintainability, particularly in large, complex mobile applications.



5 USER REQUIREMENTS FOR A MOBILE APPLICATION

Mobile app requirements document the business logic, technical specifications, and development guidelines for mobile app developers to design the application of your business dreams.

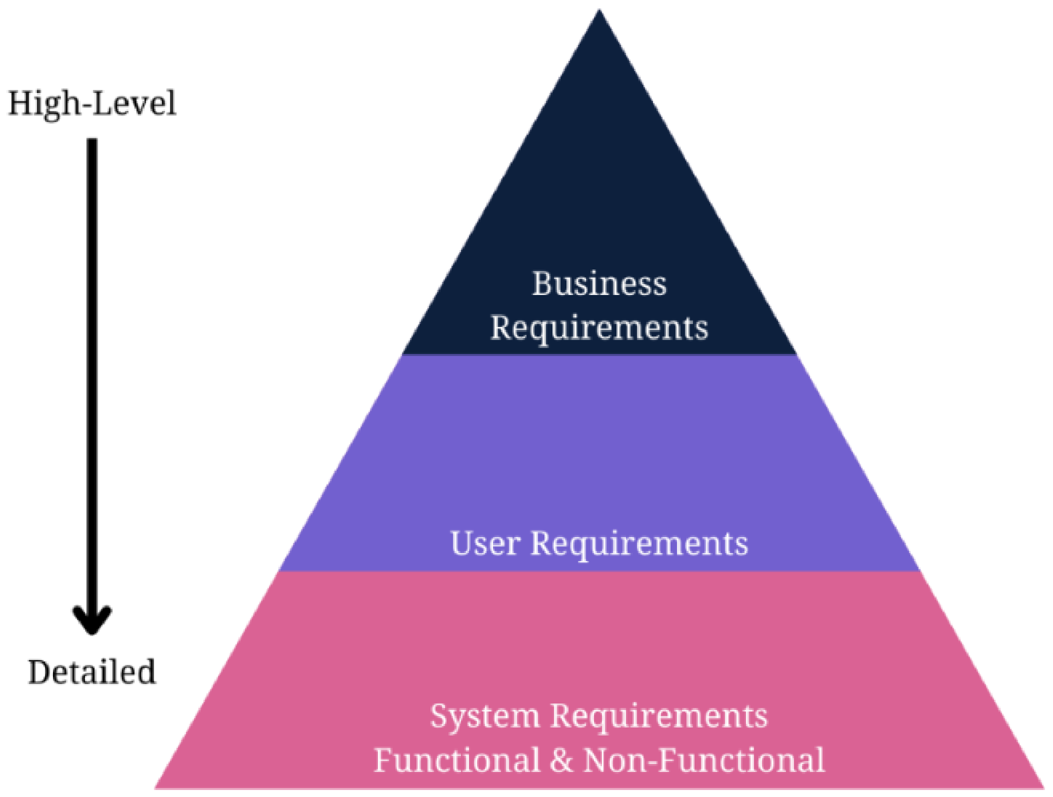
It includes the key app's features, app user personas, and business goals to ensure that multiple team members are on the same page before the software development process commences.

Mobile apps have different requirement types to collect, including:

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.

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.

Product or system requirements are non-functional requirements and functional requirements that include technical requirements and technical specifications for the engineering team.



- A functional specification document will outline the product's features and how users interact with the app.
- On the other hand, non-functional requirements include the technical specification for the app's quality attributes.
- For example, a functional requirement refers to the sign-up button users click on before using your services.
- However, non-functional requirements refer to how fast the system responds to the user clicking on that button and how the system protects their data while clicking the button.

The following steps to gather requirements for a mobile application to ensure you meet the business requirements and key features necessary to develop a profitable product.

5.1: 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?

We need to identify a problem the app will solve to recognise the idea or purpose behind it.

Requirements gathering and management for mobile apps require some effort with massive results.

5.2 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:

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

5.3 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 the team gather more user requirements for the development company.

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

- Identify the direct, indirect, secondary, and substitute competitors for the mobile app. Remember to recognize any businesses offering similar mobile app services or products and those offering different products in a broader niche umbrella.
- 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.
- 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.

5.4 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 functionalizes 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)
- Behavioral 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 means creating personas.

5.5 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 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, the non-functional requirements (NFRs) and functional requirements are prioritized for an app.

Priorities determine the tech stack and importance of each function.

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.

5.6 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.

5.7 Write an App Requirements Document

- 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.

- 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.

- 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 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.

- 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.

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.

- 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.

5.8 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:

- Map the target user flow.
- Sketch the flow's core part.
- Set a mobile wireframe.
- Determine the layout with boxes.
- Use design patterns.
- Add intended copy.
- Connect the app's pages to design a flow.
- Design a prototype.
- Release the initial design to gather feedback.

5.9 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 share insights about feedback and collaboration.

5.10 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.

6 ESTIMATING MOBILE APP DEVLOPMENT COST

6.1 Define Your App's Objective and Features

To estimate the cost of your app, start by determining its goal and features. What do you want your app to achieve? What functionalities do you need? Make a list of features, prioritizing them based on importance and complexity. This will lay the groundwork for estimating the cost.

For example, your app may aim to help users find local restaurants and make reservations. Key features could include searching for restaurants by cuisine, location or name, viewing restaurant details and booking reservations directly app. More complex features like online food ordering may require additional development hours. Clearly defining your app's purpose and minimum viable product will allow for a more accurate cost estimate.

6.2 Choose the Platform

Deciding between iOS, Android, or both platforms is crucial for determining development costs. Developing separate apps for each platform requires more resources compared to creating a cross-platform app. Each platform has unique design principles and development complexities that impact the overall cost.

For instance, building native iOS and Android apps would mean employing developers skilled in Swift/Objective-C and Java/Kotlin respectively, whereas a cross-platform framework like React Native or Flutter allows using the same codebase for both stores with some customization. Consider your target users and distribution strategy when deciding on platforms.

6.3 Evaluate App Design Complexity

The design of your app plays a significant role in its appeal and usability. Consider whether your app would benefit from a simple and minimalistic design or a more elaborate and visually captivating interface. Keep in mind that intricate designs with custom animations and interactions can increase development costs. For example, a basic design with one or two screen views per section may take 50–100 hours whereas an immersive design with advanced motion graphics could easily exceed 200 hours for the visual design work alone.

6.4 Estimate Development Hours

Once you have a clear understanding of your app's features and design, estimate the number of development hours needed for each task. Break down the project into smaller components and allocate hours based on complexity. Using industry benchmarks or seeking guidance from developers can help you make a more accurate estimation.

For instance, a login screen may take 10 hours whereas integrating payment processing could take 30–50 hours depending on the provider. Don't forget to factor in testing, bug fixes and other non-development tasks into your hour estimates.

6.5 Research Development Rates

Hourly rates for mobile app development vary depending on regions and development teams. Take the time to research average rates in major cities like San Francisco, New York, London and Bangalore which range from $50–150 per hour depending on the seniority and expertise of the developers. Make sure to check rates on reputable freelancing sites and job boards.

Remember that a higher rate doesn't always guarantee better quality. Read reviews from past clients, examine examples of applications developed by the team in their portfolio, and evaluate the specific technical skills and experience levels of the core team members before making a decision. Pay close attention to reviews that comment on factors like responsiveness, ability to work within budgets and timelines, and quality of project management.

6.6 Consider Additional Expenses

In addition to development hours, there are other costs to consider. Quality assurance and testing are essential for ensuring all features and functionality of the application work as intended across different device types and operating system versions, so allocate at least 15–20% of the development budget for thorough testing and fixing any bugs found.

Also, factor in the current publishing fees which are $99 per year for the Apple App Store and $25 per upload for the Google Play Store when estimating total costs. Regular updates to address compatibility issues, bugs and add new features should also be budgeted for, estimating at least 2–3 updates per year.

6.7 Include a Contingency Budget

Unexpected challenges like changes to project scope, integration issues between features, or bugs that take significant time to resolve can arise during development, leading to delays and extra expenses. It's wise to include a contingency budget of 15–20% of the total estimated development costs to help account for any unforeseen obstacles that may come up over the course of the project. This helps reduce financial risks.