UNIVERSITY OF BUEA

REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

P.O. Box 63,
Buea, South West Region
CAMEROON
Tel: (237) 3332 21 34/3332 26 90
Fax: (237) 3332 22 72

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER ENGINEERING

INTERNET PROGRAMMING AND MOBILE DEVELOPMENT

By:

Group 6

NAMES	MATRICULE	Speciality
KOUETE TEPE KENNETH	FE21A220	Software
ZELEFACK MARIE	FE21A330	Software
EFUETAZOH ASONG RODERIC	FE21A179	Network
TAKO DIEUDONNEVOJONG	FE21A310	Software
BUINFUN MONIE JULIUS	FE21A154	Software

Supervisor:

Dr Nkemeni Valery

University of Buea

2nd Semester 2023/2024 Academic Year

Abstract

In the ever-evolving landscape of technology, mobile applications have carved out a niche that's impossible to ignore. This report is a deep dive into the world of mobile app development, an exciting realm where creativity meets technical prowess. We'll explore the diverse types of mobile apps that populate our digital ecosystem, the programming languages that give them life, and the development frameworks that provide the scaffolding for their creation. We'll also delve into the intricate architectures and design patterns that underpin these apps, the art of requirement engineering that ensures they meet user needs, and the process of cost estimation that keeps the wheels of development turning. Our aim? To offer you a holistic understanding of the multifaceted process of mobile app development. So, whether you're a seasoned developer or a curious onlooker, we invite you to join us on this enlightening journey.

TABLE OF CONTENT

LIST OF FIGURES

INTRODUCTION

In the palm of our hands, we hold a world that's constantly evolving and expanding - the world of mobile applications. As our reliance on smartphones continues to grow, so does the demand for mobile apps that cater to every aspect of our lives. From ordering food to tracking our fitness, these apps have woven themselves into the fabric of our daily routines. But have you ever wondered about the magic that brings these apps to life? That's where mobile application development comes in. It's a fascinating field that's as dynamic as the apps it creates, and it's a skill that's becoming increasingly important in our digital age. This report is your gateway to understanding this exciting world. We'll peel back the layers of mobile app development, exploring everything from the types of apps and programming languages to the frameworks, architectures, and design patterns used. So, buckle up and get ready for an enlightening journey into the heart of mobile app development!

OBJECTIVE

The objective of this report is to review and compare the major types of mobile apps (native, progressive web apps, hybrid apps), mobile app programming languages, and mobile app development frameworks. It also aims to study mobile application architectures, design patterns, the process of collecting and analysing user requirements for a mobile application (Requirement Engineering), and how to estimate mobile app development cost.

CHAPTER 1. MAJOR TYPES OF APPS AND THEIR DIFFERENCES [1]

1.

Mobile apps have become an integral part of our daily lives, providing a wide range of functionalities and experiences. There are several types of mobile apps, each with its own unique characteristics and advantages.

1.1.Types of mobile Apps

1.1.1. Native Apps

Native apps are applications for mobile devices built with the native language of chosen platforms: Objective C and swift for IOS, Java and Kotlin for Android. A native programming language is the primary or default language used on a particular platform. Examples of Native Apps Pre-installed apps like calculator, contacts, File Manager etc on new devices are native apps

Advantages

- It works well offline.
- High-level security is guaranteed by the OS itself.
- Fewer bugs and technical issues since the app is written in the same language as the platform and doesn't need any translation

Disadvantages

- More development time, especially when building the app for multiple platforms.

1.1.2. Progressive Web Apps (PWA)

These are web apps that simulate the behaviour of native apps (the can send notifications, use the camera, ask for location etc) and are typically accessed via a web browser of your choice but can be installed and accessed on a device. Examples of PWA are Spotify, Alibaba.com, Uber, Pinterest etc.

Advantages

- Accessible via a website

- Cheap and faster in development

Disadvantages

- Offline functionalities are Limited

1.1.3. Hybrid Apps

`These are mobile applications that combines elements of both native app and web-based apps. In simple terms, a hybrid mobile app is like a website that has been packaged and optimized to run on a mobile device while also having some native capabilities. Example: Facebook, Twitter etc. These apps are hybrid apps, combining web-based and native components. These app's user interface and core functionalities are built using web technologies but are also integrated with native device features such as push notifications, camera access and location services.

Advantages

- Hybrids apps can be developed once and deployed across multiple platforms such as (IOS, Android and windows) with minimal code changes.

- Hybrid apps can access native device feature such as camera, GPS etc.

Disadvantages

- Has limited offline functionalities

1.2.Differences between Native, Hybrid and PWAs Apps

Table 1.1 below shows the differences between different types of mobile apps

Table 1.1. Difference between Native, PWA and Hybrid Apps

Features	Native Apps	Hybrids Apps	PWAs

Language	iOS- Swift Android- Java and Kotlin	HTML, CSS and JavaScript	HTML, CSS and JavaScript
Platform	iOS or Android	WebView	Web browser
Security and Maintenance	Highest security and Maintenance	Safe Maintenance	Lowest Security and Maintenace
Access to mobile feature	Yes	Limited	No
Development Process	Hardest and Slowest	Easy and Quick	Easiest and quickest

CHAPTER 2: MOBILE APP PROGRAMMING LANGUAGES [2]

2. Introduction

Let's think of it this way. You're about to embark on an exciting journey to create a mobile app. It's like learning a new language before visiting a foreign country. You've got Java, the seasoned traveller, reliable and full of wisdom. Swift is like the adventurous backpacker, quick, efficient, and full of energy. And JavaScript, well, it's the versatile globetrotter, comfortable in any environment. Picture out the most common mobile application programming languages

Figure 2.1. Common Mobile Application Programming Languages

Each one has its own charm and choosing the right one can make all the difference in your journey. So, let's dive in and get to know these languages a bit better, shall we?

2.1. Developing mobile apps for the Android platform

Picture this: you're about to create an app for Android, the world's most popular mobile platform. But how do you speak Android's language? That's where programming languages come in. They're like the secret codes that bring your app to life. Ready to crack the code? Let's get started!

2.1.1. Common Programming Languages

JAVA

Java was the default language to write Android apps since the Android platform was introduced in 2008. Java is an object-oriented programming language that was originally developed by Sun Microsystems in 1995 (now, it is owned by Oracle). 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.

KOTLIN

In 2017 Google announced that it will support Kotlin as an alternative first-class language for Android programming. It 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.

2.1.2. Comparism between java and kotlin

Similarities

- **Interoperability:** Both are fully interoperable, this allow developers to gradually migrate from java to kotlin or use both languages in the same codebase.
- **Object-Oriented Programming**: both languages are object oriented and share similar concepts such as encapsulation , polymorphism etc.
- **Android Development**: both Java and kotlin can be used for Android app development. They have access to the same Android SDK and can build apps with similar functionality.

Differences

Table 2.2. shows the difference between kotlin and Java

Table 2.2. Difference between Java and Kotlin

Features	Kotlin	Java
Syntax	More concise and expressive	Less concise and expressive
Backward compatibility	Not as strong as java, may require new language features that require the new version of kotlin compiler	Very strong ensuring that previously written code in older versions run smoothly in newer versions
Functional programming	Has better support for functional programming , it provides highordered functions, lambda functions and built-in functions making it easier to style functional code	has less support for functional programming concepts
Couroutines	Has a native support for couroutines which are a strong mechanism for writing asynchronous and concurrent code	Does not support couroutines ,but uses threads, promises etc to handle concurrent and asynchronous code which are generally more compledx

2.2. Developing mobile apps for the iOS platform

2.2.1. Programming Languages

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.

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.

2.2.2. Comparison Between The Programming Languages

Similarities

- **IOS development:** Both are used for IOS development
- **Interoperability**: swift is designed to be highly interoperable with Objective-C. Meaning objective c code can be used in swift and vice versa.
- **Object Oriented Programming**: both are object oriented, and support concepts such as classes, objects, inheritance etc.

Differences

Features	**Swift**	**Objective-C**
Syntax	More modern and concise	Verbose syntax
Performance	Faster	slower
Compatibility	Uses Automatic Reference Counting (ARC) and ARC-swift	Uses only ARC
	for memory management making it more efficient	
Learning-curve	Generally considered easier to learn as it is more readable and less complex syntax	Generally considered more difficult

2.3. Developing mobile apps for Cross platforms

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.

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.

HTML AND 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 **Untapped and Untitled Goose Game** employ HTML/CSS for their cross-platform compatibility and rapid development cycles.

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.

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.

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.

So, 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 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.

To make informed decisions in this competitive landscape, it is crucial to align language choices with project requirements, target platforms, and developer proficiency

CHAPTER 3: REVIEW AND COMPARE MOBILE APP DEVELOPMENT FRAMEWORKS BY COMPARING THEIR KEY FEATURES

3. Introduction

Mobile app development frameworks have come a long way. In the past, developers had to create separate codebases for iOS and Android platforms. Cross-platform programming emerged as an alternative approach, but challenges persisted.

Today, mobile app frameworks continue to evolve, adapting to industry trends and technological advancements. The future holds exciting possibilities.

3.1. Evolution of App Development Frameworks

Mobile app development frameworks have evolved due to technology advancements and changing development paradigms. We can expect further innovations in artificial intelligence, blockchain integration, and immersive technologies.

In the year 2024, with the ongoing expansion of mobile technology, the decision on the most fitting **mobile development** framework is increasingly vital for both developers and companies. The field is overflowing with alternatives, with every option offering distinctive attributes, user-friendliness, and performance potential. Opting for a framework that matches your project's objectives, technical requirements, and user anticipations is not merely a matter of thorough preparation; it's a deliberate move in the current aggressive application marketplace.

Mobile app frameworks are platforms and tools that provide pre-built components and libraries for the streamlined development of mobile

3.2. Common Mobile Development Frameworks

Here are some popular mobile app development frameworks:

- React Native: Developed by Facebook, React Native is a JavaScript framework for building native apps for Android and iOS. It allows developers to write code once and run it on both platforms, saving time and effort.

- Flutter: Flutter is a UI toolkit from Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It uses the Dart language and offers a rich set of pre-designed widgets.
- Xamarin: Xamarin is a Microsoft-owned framework that allows you to build apps for Android, iOS, and Windows using C# and .NET. It provides tools to directly invoke platform-specific APIs.
- Ionic: Ionic is a popular, free framework for building cross-platform mobile apps using web technologies like HTML, CSS, and JavaScript1. It focuses on performance, and leverages hardware acceleration.

Here's a summary Comparison with respect to key matrices in table format: **table 3.1**

Table 3.1. Comparison of mobile App Frameworks with respect to key matrices

Framework	**Language**	**Performance**	**Cost & Time to Market**	**UX & UI**	**Complexity**	**Community Support**
React Native	JavaScript	High	Moderate	High	Moderate	Strong
Flutter	Dart	High	Moderate	High	Moderate	Growing
Xamarin	C#	High	Moderate	High	High	Strong
Ionic	HTML, CSS, JavaScript	Moderate	Low	High	Low	Strong

CHAPTER 4: MOBILE APPLICATION ARCHITECTURES AND DESIGN PATTERNS.[4]

4.

4.1. Mobile App Architecture

4.1.1. What Is Mobile App Architecture?

Mobile app architecture is the structural design and organization of a mobile application, outlining how various components and modules of the app are interconnected and work together to achieve its functionality. It serves as the blueprint for the development of a mobile app, defining the framework for building, maintaining, and expanding the application.

Mobile app architecture includes various layers or components, each with specific responsibilities, and it determines how data is processed, presented to users, and interacted with. It plays a critical role in the app's performance, scalability, maintainability, and security.

4.1.2. Key Components of Mobile App Architecture

The key components of mobile app architecture consist of the following layers:

- **User Interface (UI) Layer**

The UI layer is responsible for the presentation of the app to the user. It includes the visual elements and components that users interact with, such as screens, buttons, forms, navigation menus, and any graphical elements. It manages the layout and appearance of the app.

Common technologies used in the UI layer include UI frameworks, user interface libraries, and design tools that help create a visually appealing and responsive user experience.

- **Application Logic Layer**

The application logic layer, also known as the business logic layer, houses the core functionality of the app. It includes algorithms, business rules, and processes that control the app's behaviour.

This layer processes user input, orchestrates data retrieval and storage, and ensures the correct operation of the app's features.

The application logic layer uses programming languages, software libraries, and frameworks specific to the platform (e.g., Android, iOS) to implement the app's functionality.

- **Data Layer**

The data layer manages data storage, retrieval, and communication with external data sources. It includes databases, server APIs, and any data repositories that the app interacts with. This layer ensures data integrity, security, and availability.

4.1.3. Types of Mobile App Architecture

There are primarily five types of mobile application architecture. Let's discuss each in detail here:

a. Monolithic Architecture

In monolithic architecture, all components and modules of an application are tightly integrated into a single, unified unit. In a monolithic architecture, the entire application, including the user interface, application logic, and data storage, is bundled together as a single codebase and runs within a single process.

Key Characteristics of Monolithic Architecture:

- **Single Codebase**: In a monolithic architecture, all the code that makes up the application is part of one codebase.
- **Tight Coupling**: Components and modules within the monolith are tightly interconnected, meaning they are interdependent and often share resources and libraries. Changes to one part of the code can have ripple effects on other parts, making maintenance and updates more challenging.

- **Single Deployment Unit**: Monolithic applications are typically deployed as a whole. When you need to make changes or updates to the application, you redeploy the entire monolith, which can result in downtime during updates.
- **Shared Data Storage**: Data storage is centralized, and all components of the application share access to the same database or data store.

Advantages of Monolithic Architecture:

- Easier to develop, test, and initially deploy
- A wealth of knowledge and tools available for monolithic development
- Does not require communication between separate services

b. Micro services Architecture

In a microservices architecture, an application uses small, independent, and loosely coupled services that operate simultaneously to provide its functionality.

Instead of building a monolithic application where all features and functions are tightly integrated into a single codebase, a micro services architecture breaks down the application into a collection of individual services, each responsible for a specific set of tasks or functionalities.

Key Characteristics of Micro services Architecture:

- **Service Independence**: Each service is a standalone unit with its codebase, database, and possibly even its technology stack.
- **Loose Coupling**: Services communicate with each other through well-defined, lightweight interfaces such as APIs (Application Programming Interfaces).
- **Small and Focused**: Micro services are generally small in size and focused on a specific piece of functionality. Each service should perform a single task well.
- **Distributed Deployment**: Micro services are often deployed independently. This means that different services can be hosted on separate servers or containers.

- **Polyglot Technology**: In a micro services architecture, different services can use different programming languages and technologies based on the specific requirements of the service.
- **Independent Data Management**: Services may have their own data stores, which can be a database, key-value store, or other data storage solutions.

Advantages of Microservices Architecture:

- Microservices can be individually scaled up or down as needed
- Flexibility to choose the most appropriate technology for each service
- Smaller, focused teams can work on individual services, leading to faster development cycles and quicker time-to-market for new features.
- Failure in one service is less likely to impact the entire application

Easier Maintenance **Model-View-Controller (MVC)**

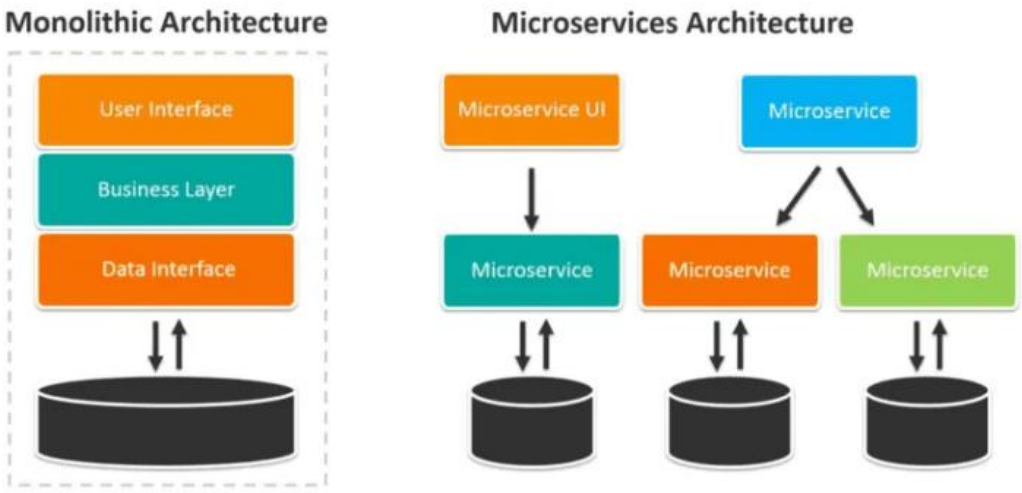

Figure 4.1. monolithic vs MicroService Architecture

c. Model-View-Controller (MVC)

Model-View-Controller (MVC) is a widely used architectural design pattern for developing software applications, particularly in web and mobile applications.

It divides the application into three interconnected components, each with a specific role and responsibility. The primary purpose of the MVC pattern is to separate the concerns of an application, making it easier to develop, maintain, and extend.

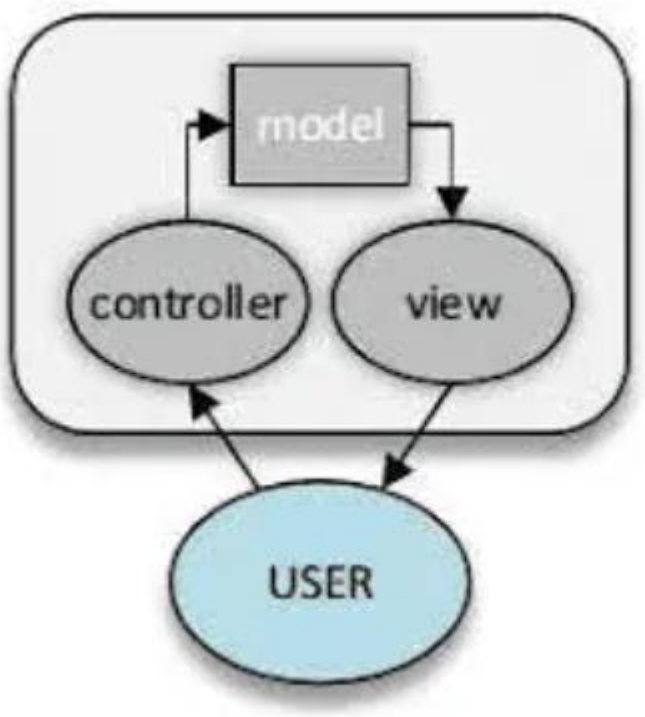

Figure 4.2: MVC architecture

a. Model:

It represents the data and business logic of the app. It encapsulates the core functionality and data management of the application. This component is responsible for data storage, retrieval, validation, and processing. It responds to requests from the Controller and notifies the View when the data changes.

b. View:

Its responsibility is to render the user interface and present data to the user. It displays the information from the Model to the user in a visually appealing and understandable format.

c. Controller:

It works like an intermediary between the Model and the View. It receives user input, processes it, and communicates with the Model to retrieve or update data. The Controller decides which View should be displayed to the user based on the user's actions

4.2. Mobile App Design Patterns[4-1]

The best mobile app design patterns are special because they respect all the rules and principles we described above. This is precisely what makes them so popular and frequently used, so we want to present you some of these solutions. There are good examples of awesome mobile app designs. Let's check out our top 5 picks:

Full-Screen Navigation

The name of this pattern clearly shows its purpose. Full-screen navigation uses the entire homepage to ensure simple and intuitive user experience, taking them from generic content to the more specific pages one step at the time.

Full-screen navigation is ideal when you have to add tons of features to the single app. With this approach, you are not forcing users to face all components instantly. Instead, let them choose their path through the app.

The disadvantage of this concept is the inability to display any other options except for the navigation. However, it's something most developers are willing to give up on if they have to design an all-encompassing mobile app.

Side Drawer

Instead of dedicating the whole homepage to the navigation, you can use a side drawer to hide most of the orientation elements. This pattern (also known as the hamburger menu) hides the navigation bar on the left side of the screen, letting users decide when to reveal it completely.

Side drawer is excellent for secondary features – the ones that don't repeat too often. However, it's not the perfect pattern if your app requires instant navigation due to the low discoverability. Additionally, side drawer forces users to take an extra action, which is never a good thing to do in the digital environment.

The Skeleton Pattern

Speed is one of the most essential features in mobile app usage. Every millisecond you waste can chase users away, but the skeleton pattern solves the problem by making your product appear faster than usual.

For instance, Facebook uses this pattern because it’s able to instantly display at least some part of the content, thus successfully replacing the launch screen. That way, consumers can immediately notice a few features, which drastically increase retention. On the other hand, the skeleton is not necessary if users don’t launch your app at least once a day.

Tab Bar

The tab bar pattern is not a novelty because we know it from the desktop design. It’s the option that enables direct access to the essential destinations, which makes it perfect for simple apps with up to five different navigation options.

CHAPTER 5: HOW TO COLLECT AND ANALYSE USER REQUIREMENTS FOR A MOBILE APPLICATION (REQUIREMENT ENGINEERING)[5]

5.1. What is requirement Engineering ?[5-1]

Requirements engineering (**RE**) is the process of defining, documenting, and maintaining requirements in the engineering design process. It is a common role in systems engineering and software engineering.

The first use of the term requirements engineering was probably in 1964 in the conference paper "Maintenance, Maintainability, and System Requirements Engineering", but it did not come into general use until the late 1990s with the publication of an IEEE Computer Society tutorial in March 1997 and the establishment of a conference series on requirements engineering that has evolved into the International Requirements Engineering Conference.

In the waterfall model. requirements engineering is presented as the first phase of the development process.

5.2. Why is requirements Engineering important ?[5-1]

To turn your idea into a shippable mobile app, you need a team of developers. But finding the right team isn't the hard part. The hard part is explaining your mobile app vision to developers so clearly that they conceive it the way you do.

Problems:

One limited study in Germany presented possible problems in implementing requirements engineering and asked respondents whether they agreed that they were actual problems. The results were not presented as being generalizable but suggested that the principal perceived problems were incomplete requirements, moving targets, and time boxing, with lesser problems being communications flaws, lack of traceability, terminological problems, and unclear responsibilities. So, from this study we can see that we need requirement engineering because of the following reasons:

- **Increase your own certainty.** Discussing requirements for your mobile app makes everything clearer. Objectives, perspectives, features, constraints — your product

vision starts to take shape. Determining product requirements moves you from fuzzy statements to tangible tasks with thorough deadlines, budgets, and quality criteria.

- **Make your ideas clear to developers.** Clear product requirements reduce the expectation gap between the mobile app you want and what developers deliver.
- **Get prompt development and delivery.** With documented mobile app requirements in sight, your development team can better understand your project, set priorities, and reduce rework.
- **Make sure the final app meets your quality expectations.** Thanks to the acceptance criteria stated in a PRD, your team can easily determine whether you will be satisfied with the delivered app.
- **Reduce scope creep.** A high-quality requirements specification prevents you from developing unnecessary features, prevents your team of developers from working at cross-purposes, and guards your whole development team from becoming overloaded.
- **Spend less.** Since well-thought-out requirements contribute to a focus on the essentials, reduce rework, and speed up development, they save you money.

According to Boehm's research(Barry Boehm, University of Southern California Victor R. Basili, University of Maryland), rework can cost about 40% to 50% of the total cost of all software development. And a major part of rework is caused by requirements errors. **IBM once claimed this**:

"Time not spent in requirements is time spent in rework — at 200 times the cost."

5.3. Types of product requirements

When you get an idea to make an app, you need to ask yourself three main questions:

- **Why?** Why do you need a mobile app? To help people with your unique experience, get an extra revenue stream, as an investment — what is your goal?
- **Who?** Who will use your app? Think of your target users' pains, problems, needs, and preferences. What solution do users expect to get from your app?

- **How?** How will you achieve your desired business outcomes and meet users' expectations? Think of the functionality your app should provide.

Answers to these questions form **three main levels of requirements** for mobile app development: business requirements, user requirements, and system requirements.

Each level also has an assortment of functional and non-functional requirements.

Functional requirements relate to your app's operation and features you're going to implement.

Non-functional requirements define characteristics and constraints that aren't connected to functional requirements. In most cases, non-functional requirements relate to:

- **Attributes of the developed product** like performance, reliability, availability, and usability.
- **The app development process**, describing development methodologies, standards, coding languages, time restraints, security, etc.
- **The external environment**, taking into account your app's connection to other systems and hardware components, alignment with corporate policy, government regulations, and so on.

The types of requirements are as follows:

5.3.1. Business requirements

When writing your business requirements, we focus on reasons why building a mobile application is essential for our business, the changes the app will entail, and the outcomes we expect it will deliver. To keep our product vision clear to our development company, you should record our business requirements in a mobile app business requirements document (BRD).

Based on the vision and scope document proposed by **Karl Wiegers** in the third edition of *Software Requirements*, we can see the following BRD structure(on **Table 5.1**):

Table 5.1. Proposed Structure of Business requirement

1. Business requirements	
Background	Describe the situation that led you to the idea of creating a mobile app, the overall goal(s) for your project, and improvements you suppose it will bring to your business.
Business opportunity	Highlight strengths and advantages of your app compared to existing solutions on the market. Describe how your mobile app will keep up with market trends and ever-evolving technologies.
Business objectives	Summarize what benefits you expect to get from building a mobile app in a quantitative and measurable way. Your objectives must be **SMART** (specific, measurable, achievable, relevant, and time-bound). An objective may read like this: "I want to bring in $X in revenue and return Y% on investment within Z months."
Success metrics	Determine what indicators will help stakeholders understand that your project has achieved success. For example, for an e-commerce app, to bring in $X in revenue within Z months, a good goal could be getting two cross-sales on 80% of orders.
Vision statement	You can describe your product vision using the following format: • **For** (target users)

	• **Who** (need or want to change something) • **The** (your product name) • **Is** (a mobile app) • **That** (will provide unique valuable functionality, key benefit) • **Unlike** (current business model or competitors) • **My product** (advantages that differentiate your app from competing apps)
Monetization model	From the outset of your project development, define how your mobile app will generate revenue.
Business risks	Think of possible situations that can adversely affect your mobile app development. For example, what will you do if you get too few downloads? You need primarily to estimate the probability of this risk and how it will impact the whole project. Then plan actions to control, mitigate, or eliminate the risk. Involve other stakeholders to join in the decision-making.
Assumptions and dependencies	Business assumptions reflect your observations of ways you can achieve desired business objectives. Given the objective to bring in $X in revenue within Z months, your assumption can be that a new app will attract 100 monthly active users who will spend on average $15 a month. Highlight external factors that your mobile app development depends on, such as third-party suppliers, partners, other business projects, industry standards, or legislation.

2. Scope and limitations	
List of features	Here we define what features our app must, should, could, and won't provide based on your business objectives, time and financial resources, and problems with existing business solutions, if any.
Scope of initial release	Here we define what features we should develop first. For help deciding.
Scope of subsequent releases	This section describes features that aren't so critical to be developed first because of their complexity, high cost, or low profitability. We can implement them in future app releases.
Limitations and exclusions	Here we List features that we have to cut from the project scope. we can add them to subsequent releases.
3. Business context	
Key stakeholders	Create profiles of everyone somehow related to your project: those who take an active part in mobile app development, who depend on its outcome, and who impact its outcome. To get the

3. Business context	
	ball rolling, we can start from our corporate organizational chart.
Project priorities	Agree on features, quality, schedule, budget, and team size. Prioritize the factors that lead to your project's success and define constraints on project development. Discuss the degree of freedom you can grant your project manager to accomplish tasks that lead to project success within the existing constraints.
Deployment considerations	Here Describe possible improvements you want to make for your mobile application to expand its market share. These can be extra features to reach audiences in other countries or new cloud data storage to make your app more adaptive.

we can **represent our project scope** using different tools. The most comprehensive is a **lean canvas**. It represents the segments of a business plan crucial for developing documentation for all mobile applications: groups of users and their main problems, solutions your app is going to provide along with a unique value proposition (UVP), and other advantages. In the lean canvas model, you can describe the channels you'll use to attract target users and key metrics that will tell you how your business is doing. A lean canvas also helps you determine the monetization model for your mobile app along with other potential revenue streams.

5.3.2. User requirements

After identifying your business requirements, it's time to focus on your users' needs. You need to outline potential aims with which users come to your app and the actions they will take to meet these aims. But whose opinion should you consider when drafting user requirements?

The trouble is, there's no single type of app user. On the contrary, there are many types of users asking for different things: investors, business owners, end users, developers, distributors, regulators, marketing staff, and others. Your task is to hear everyone and find the balance between the needs of different user groups.

When it comes to user requirements, it's sensible to start with these three steps:

Step 1 — Classify users. Group all stakeholders into user classes. You can sort them according to the following criteria for mobile applications:

- Access level (guest, regular user, paying user, provider, administrator)
- Tasks they perform (find, view, read, select, buy, share, comment)
- App features they use (searching, mapping, sorting, comparing, paying, etc.)
- Frequency of visits (daily, monthly)
- Platforms used (iOS or Android)
- Native language (or other demographics like location, gender, education, and family status.)
- **Step 2** — Identify product champions. Choose individuals who can represent each group of users and communicate user requirements to your project manager. Being a good product champion means having a clear vision of the benefits your app will bring to users. In turn, product champions must be actual users to perfectly understand users' pains and urgent needs.

- **Step 3** — Agree on the requirements decision-makers for your project. Agree on representatives of each group of users with stakeholders. Be careful not to overlook any

stakeholder to avoid complaints that the final app doesn't meet a stakeholder's expectations.

After identifying eligible user representatives, get their input on two types of user requirements to develop an app. See the propose structure of these requirements on **Table 5.2.**

Table 5.2. Proposed structure of user requirements

User requirements	
Functional user requirements	Here, we outline tasks users want to perform within our mobile app and list possible user–app interactions. Based on this data, we can derive the core functionality our app must provide to enable these interactions to happen.
Non-functional user requirements	Here, we gather users' expectations related to our mobile app's level of performance, security, usability, and so forth.
Deployment considerations	Here, we describe possible improvements we want to make for our mobile application to expand its market share. These can be extra features to reach audiences in other countries or new cloud data storage to make your app more adaptive.

Record feedback from users in a **user requirements document (URD)**. To do this, you can use the following techniques:

- **A user persona:** it is a useful tool that allows you to visualize your target users. For each user persona, we choose a name and a photo, then list the user's needs, wants, and aims. Write key reasons why the persona will use our app. Here is an example of a user persona for picture sharing mobile app:

Figure 5.1. Example of user persona

- **User stories.** Itemize actions users will perform within your app to meet their goals. Then arrange these actions in a natural sequence to determine a typical user journey through your app. Depending on your project's scope, you can primarily outline epics — intricate user actions that you can decompose into smaller steps users will take while using your app.

In Agile development, user stories are often put into a product backlog. While negotiating the scope of software development for the first and subsequent releases, the client and the **development team** will consider user stories from the backlog and select the most

relevant. By arranging user stories, they can form a product roadmap that clearly defines what app features they should implement and when. The example below is related to the two most common basic epics for any mobile app:

Epics	Welcome/Onboarding	Authorization
User stories	Splash screen + app carousel	Sign up with a phone number
Why	As an unregistered user, I want to have a sneak peek of the app so that I'll be aware of its main features prior to registration	As an unregistered user, I want to get registered to the app so that I'll get personalized content
What	App carousel should include up to three blocks that a user can swipe through to find out more about the app	Registration form fields should include: • phone number (phone input field with the US phone number format); • password (text input field that can't be blank and can't be less the two symbols); • terms and conditions (in the form of hyperlinked text).

Figure 5.2. Basic Epics for a mobile App

5.3.3. System requirements

A complete product requirements document for a mobile app should contain requirements on how your app must operate. Resist the lure of hastily writing a technical design document for a mobile application based only on users' wants and business needs. We need to talk to developers. They'll give us feedback on whether it's technically possible to realize our original plans for the app's functionality. In talking to developers, you'll reveal potential threats to our project development and can collectively establish a plan B to sidestep them.

After constructive dialogue with our team, we have to write down the agreed requirements in a **software requirements specification (SRS**) for a mobile application that contains the following blocks(**see Table 5.3**)

Table 5.3. **System requirements**

System requirements	
Functional requirements	List the features developers can build to enable users to complete tasks according to your business requirements. To do this, use existing mind maps or user stories. After defining what your app will do, assign a unique name and number to every functional requirement along with a short description, rationale, and status.
Subsystem requirements	Describe requirements for developing an app from the perspective of software and hardware subsystems. For example, if you're going to build a fitness activity tracking app, you'll need to write requirements for wearable trackers that will synchronize with the app.
Business rules	Since every business is subject to laws, policies, and industry standards, these will be obvious sources of requirements for an SRS. Here's a shortlist of requirement sources: • Corporate policy • Government regulations • Industry standards • User roles and permission ratings • If-then models of user behavior

System requirements	
	◦ Computational algorithms, if any
Data requirements	When developing a mobile app, you need to create, store, modify, display, delete, process, and use massive amounts of data. To manage data flows, you need to: ◦ outline a logical model of data entity interactions ◦ define entities in the data dictionary ◦ specify how the system must enforce data analysis, retention, or disposal ◦ choose types of data reports (spreadsheets, charts, dashboards, etc.)
Quality attributes	Writing clear quality criteria ensures that developers will meet your expectations with the end product. You need to consider quality attributes that are important to: ◦ your business and users, such as usability, performance, and security (**external attributes**) ◦ developers, such as efficiency, modifiability, and portability (**internal attributes**) Discuss what attributes are critical to your app's success with other stakeholders and prioritize them. Write specific expectations for each attribute using **fit criteria** — a quantification of the requirement that describes the standard your app must reach. Translate quality attributes

System requirements	
	into technical specifications and write acceptance tests for your team to enable them to check results.
External interfaces	This part of a functional requirements document for a mobile application is needed to ensure that your app will communicate properly with users and external hardware or software systems. In an SRS, you need to write down requirements for: • **User interfaces.** Specify the design of your mobile app's screens (standards for fonts, icons, color schemes, images, screen size, layout, resolution, and so forth) • **Software interfaces.** Describe interactions between your app and other software components including other apps, websites, libraries, databases, and tools. • **Hardware interfaces.** Outline each of the supported device types, data and control interactions between software and hardware, and communication protocols to be used. • **Communications interfaces.** In an SRS for your mobile app, state requirements for any communications functions your app will use, including in-app messages, push notifications, emails, and network protocols.
Constraints	Record constraints that restrict your mobile app's design, operation, and implementation. First of all, check whether your mobile app specification aligns with Apple App Store and Google Play Store requirements. Additionally, specify other system constraints imposed, for instance, by

System requirements	
	the programming language used or rules of using third-party APIs or content.
Localization requirements	If you want your app to be used in countries, cultures, and geographic locations that differ from those in which it was created, then you should set requirements for changing: • Currency • Date, number, address, and telephone number formats • Language (including national spelling conventions, local dialects, directions) • Functionality to comply with regulations and laws • Content in consideration of cultural and political issues • Time zones • Weights and measures • Other variables

5.4. Summary of Activities Involved in drafting a document for requirements

After understanding what requirement Engineering is and understanding the different types of requirements for a mobile App, let see how to draft a document of requirements. Drafting a document of requirements for your mobile app project is commonly about performing four activities:

- **Elicitation:** This involves asking what users expect from a new product, listening to what they say, and watching what they do.
- **Analysis:** This involves processing user feedback to understand, classify, and relate this information to possible mobile app functionalities.
- **Documenting:** It's essential to document your app idea as a product hypothesis and product objectives.
- **Creating User Personas and User Stories**: These help you understand your users better and design your app according to their needs.
- **Developing App Wireframes or an App Prototype:** This gives a visual representation of what the app will look like and how it will function.
- **Listing App Features and Functional Requirements**: This helps you understand what features your app needs to have and how they should function.
- **Deciding on the App Tech Stack and Technical Performance Characteristics**: This involves deciding on the technologies you will use to build your app and the performance characteristics it needs to have1.

Chapter 6: HOW TO ESTIMATE MOBILE APP DEVELOPMENT COST.

6.

6.1. Factors Influencing Mobile App Development Cost[6]

There are two main stages impacting the cost of developing a mobile App:

6.1.1. Technical Expertise:

6.1.1.1. Programming Fundamentals: A developer's grasp of core programming concepts (variables, functions, loops) affects development speed and efficiency. Strong fundamentals typically translate to cleaner code and fewer bugs, reducing development time and overall cost.

6.1.1.2. Mobile-Specific Languages: Proficiency in languages relevant to the target platform **(Java/Kotlin for Android, Swift/Flutter/React Native for iOS)** is crucial. Developers with this expertise can work faster and avoid compatibility issues, minimizing development costs.

6.1.2. Planning & Design Complexity:

6.1.2.1. App Functionality: Complex functionalities with extensive features require more development time, pushing the cost upwards. Simpler apps with basic features are faster and cheaper to develop.

6.1.2.2. Platform Selection: Developing for both iOS and Android doubles the workload, significantly increasing the cost. Choosing a single platform (based on the target audience) is a more cost-effective approach.

6.1.2.3. User Interface Design: A complex, feature-rich UI design necessitates more design hours, impacting the cost. A well-planned, intuitive UI can be achieved efficiently, reducing design expenses.

6.2. The formula for Calculating Mobile App Development Cost [4].

Here's a structured approach to help you gauge the approximate cost of your mobile application.

6.2.1. Total Development Time (TDT): This encompasses all phases of app development, such as planning, design, coding, testing, and deployment.

6.2.2. Hourly Rate (HR): This denotes the cost incurred per hour by the development team you engage, which can fluctuate based on factors like location and expertise.

6.2.3. Complexity Factor (CF): This is a variable multiplier that reflects the intricacy of your app's features:

- Basic Features: CF = 1.0
- Moderate Features: CF = 1.5
- Advanced Features: CF = 2.0
- Extensive Features: CF = 2.5

6.2.4. Platform Factor (PF):

This factor indicates the number of platforms your app targets:

- Single Platform: PF = 1.0
- Dual Platform: PF = 1.2
- Multi-Platform: PF = 1.5

6.2.5. Design Cost (DC): This includes the expenses associated with crafting your app's **user interface (UI)** & **user experience (UX)**, influenced by the desired design complexity.

6.2.6. Estimation Formula:

*Estimated Cost = TDT * HR * CF * PF + DC*

For example, let's consider a scenario where you're developing a moderately complex app (CF = 1.5) for both iOS and Android platforms (PF = 1.2). Your projected development time is 1,000 hours, with a budgeted hourly rate of $50 (HR = $50). Moreover, the design cost is estimated at $10,000 (DC).

Plugging these values into the formula:

Estimated Cost = 1,000 hours * $50/hour * 1.5 * 1.2 + $10,000 = $120,000.

This gives you an estimated cost of $120,000 for your mobile app project.

6.3. Estimated Development Costs of Successful Mobile App

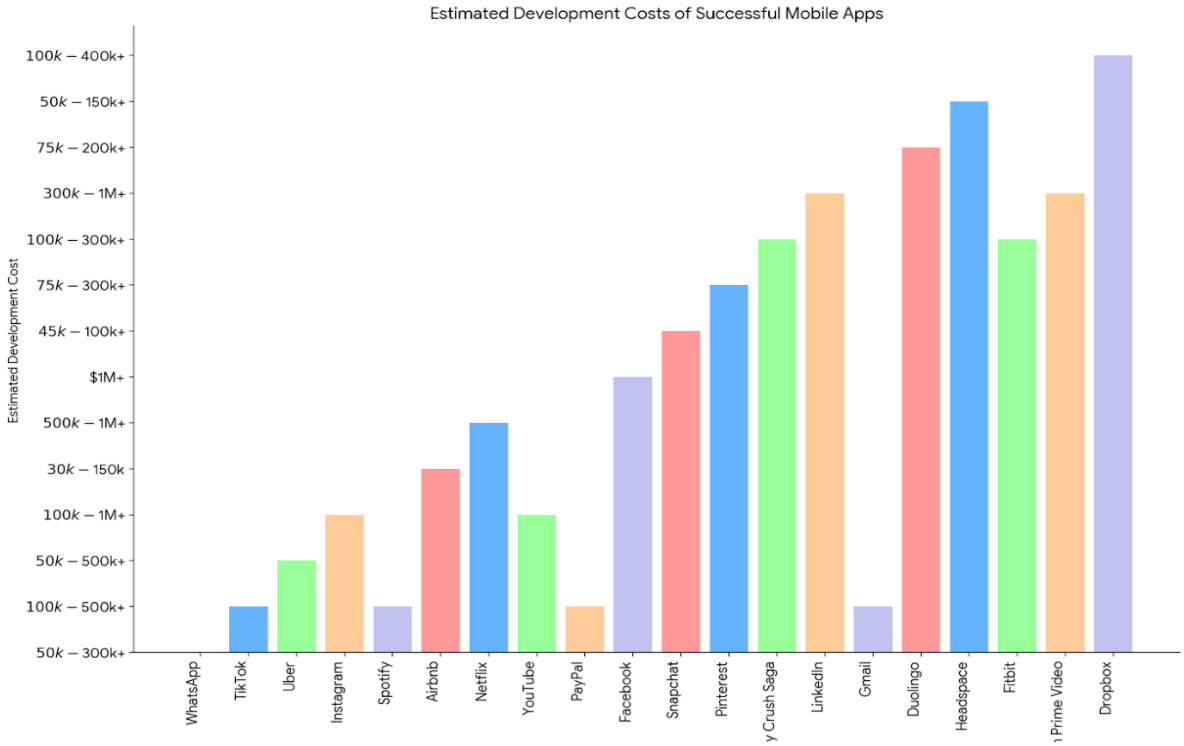

Figure 6.1. Bar chart of the Cost of successful mobile Applications

The mobile app landscape is brimming with innovation, catering to a vast array of needs and interests. This chart showcases a selection of highly successful apps across various categories, along with estimated development costs. From social media giants like Facebook and Instagram to productivity tools like Dropbox and Gmail, the range highlights how app development can target different user demographics and behaviours.

Interestingly, while complex social networking platforms with massive user bases tend to incur higher costs (over $1 million), some highly engaging games and gratified learning apps can be developed for under $300,000. This underlines the importance of a well-defined concept and target audience even for moderately budgeted apps.

6.4. What Are the Different App Costs Based on Business Categories?

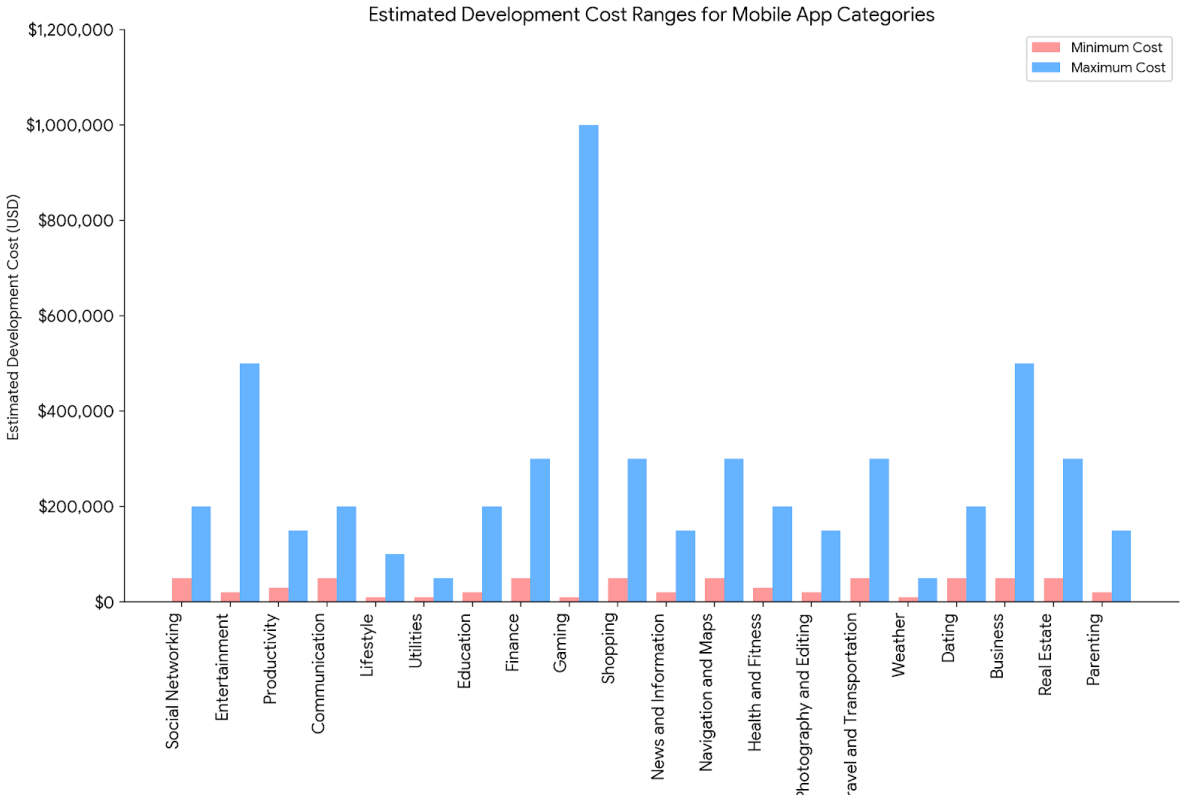

Figure : 6.2. Bar chart showing The cost of Mobile applications based on Business Category.

The table below shows the estimated cost of a mobile application based on business category.

Table 6.1.Estimated cost of mobile application base on the industry

App Category	Estimated Cost Range	Features Included
Social Networking	**$50,000 – $200,000**	**User profiles, news feeds, photo/video sharing, messaging/chat, notifications, privacy settings, friend/follow system, groups/pages, and event management.**

Entertainment	**$20,000 – $500,000**	**Streaming audio/video content, personalized recommendations, playlists, offline mode, social sharing, user-generated content, in-app purchases, gaming elements (for gaming apps within this category).**
Productivity	**$30,000 – $150,000**	**Task lists, calendars, reminders, notes, document editing/viewing, cloud synchronization, collaboration tools, time tracking, goal setting, file management, and integrations with other productivity tools.**
Communication	**$50,000 – $200,000**	**Instant messaging, voice/video calling, group chats, file sharing, emojis/stickers, status updates, voice messages, video conferencing, encryption/security features.**
Lifestyle	**$10,000 – $100,000**	**Fitness tracking, meal planning, meditation guides, habit tracking, fashion/style tips, travel planning, home decor ideas, event reminders,**

		budgeting tools, and recipe sharing.
Utilities	**$10,000 – $50,000**	**File management, device optimization, QR/barcode scanner, flashlight, calculator, unit converter, battery saver, VPN, language translator, screen recorder, widget support.**
Education	**$20,000 – $200,000**	**Online courses/classes, quizzes/tests, interactive lessons, progress tracking, offline access, discussion forums, video lectures, virtual classrooms, certification programs, and teacher/student profiles.**
Finance	**$50,000 – $300,000**	**Account management, budget tracking, expense/income tracking, bill payment, investment tracking, financial news, credit score monitoring, goal-based saving, secure transactions, and receipt scanning.**

Gaming	**$10,000 – $1,000,000**	**Various genres (puzzle, action, strategy, simulation, etc.), levels/challenges, multiplayer mode, in-app purchases, virtual currencies, leaderboards, achievements, social sharing, customizable characters/avatars.**
Shopping	**$50,000 – $300,000**	**Product browsing/searching, shopping cart, payment gateway integration, order tracking, wish lists, discounts/offers, reviews/ratings, loyalty programs, personalized recommendations, and social sharing.**
News and Information	**$20,000 – $150,000**	**News articles, RSS feeds, personalized news feeds, offline reading, push notifications, breaking news alerts, topic/category customization, bookmarking/favorites, multimedia content (images, videos), and comment sections.**
Navigation and Maps	**$50,000 – $300,000**	**GPS navigation, real-time traffic updates, route planning, voice-guided**

		directions, offline maps, points of interest (POI) search, location sharing, alternative routes, public transit information, and augmented reality features.
Health and Fitness	**$30,000 – $200,000**	**Activity tracking, workout plans, exercise demonstrations, calorie/step counting, progress tracking, meal logging, sleep monitoring, hydration reminders, heart rate monitoring, and integration with wearable devices.**
Photography and Editing	**$20,000 – $150,000**	**Photo/video capture, filters/effects, cropping/rotating, retouching tools, collage making, slideshow creation, social sharing, cloud backup, metadata editing, geotagging, and watermarking.**
Travel and Transportation	**$50,000 – $300,000**	**Flight/hotel booking, itinerary management, travel guides, currency conversion, language translation, reviews/ratings, local transportation information, weather forecasts, offline maps,**

		travel insurance, and loyalty programs.
Weather	**$10,000 – $50,000**	**Current weather conditions, hourly/daily forecasts, severe weather alerts, radar/satellite maps, customizable locations, weather widget, UV index, air quality index, sunrise/sunset times.**
Dating	**$50,000 – $200,000**	**Profile creation, swiping/matching mechanism, messaging/chat, photo verification, location-based matching, privacy controls, in-app gifts, video calling, event/meetup organizing, compatibility quizzes.**
Business	**$50,000 – $500,000**	**CRM (Customer Relationship Management), ERP (Enterprise Resource Planning), project management, invoicing/billing, employee tracking, analytics/reporting, inventory management, customer support, and secure communication channels.**

Real Estate	**$50,000 – $300,000**	**Property listings, search filters, mortgage calculators, virtual tours, agent profiles, neighborhood information, price trends, saved searches, appointment scheduling, real-time notifications, contract/document management.**
Parenting	**$20,000 – $150,000**	**Child monitoring, growth tracking, milestone reminders, parenting tips/articles, family calendar, educational content/games, health records, emergency contact list, chore/allowance management, community forums.**

6.5. How Much Does it Cost to Develop a Mobile App?

The cost of a mobile app can vary significantly depending on what you envision. Here's a simplified breakdown:

- Simple apps with basic features can cost as little as $10,000.
- Moderately complex apps with more features can range from $50,000 to $150,000.
- If you're looking at an app with extensive features and functionalities, like social media or gaming apps, the cost can reach well over $1 million.

CONCLUSION

In our cosmic journey through mobile app development, we've explored diverse territories: native realms, the web's progressive waves, and hybrid constellations. Our trusty starships—Java, Kotlin, and Swift—propelled us across the galaxies of Android and iOS. Guided by the luminous frameworks React Native and Flutter, we charted our course. Nebulae of architecture—MVC, MVVM, Clean—shaped our cosmic blueprints. The stardust of requirements engineering aligned our orbits, and our warp drives estimated costs. As we prepare to dock our starships, let's remember: mobile app development isn't just code; it's cosmic creation, lighting up lives.

REFERENCE

[1] Oskar Mieczkowski (2024) What are native, Hybrid and PWA Mobile Apps? Which Should You Choose? https://www.monterail.com/blog/native-hybrid-pwa-mobile-apps-differences

[2] Vaishnavi J.(2024).Best programming languages for mobile app development https://ellow.io/bestprogramming-language-for-mobile-app-development/

[3] 2024's Top Mobile App Development Frameworks: Build Smarter & Faster - Full Scale

[4]Mobile App Architecture - A Comprehensive Guide 2024 (octalsoftware.com)

[4-1]Mobile App Design Patterns: The Ultimate Guide to Them (designwebkit.com)

[5]How to Write a Mobile App Requirements Document (+ Free Template Download) - Mind Studios (themindstudios.com)

[5-1]Wikipedia: Requirements engineering - Wikipedia

[6] App development cost in 2023 (factors, estimates & examples) (devsolutely.com)

[6-1]Mobile App Development Costs: A Complete Breakdown in 2024 (branex.com)