UNIT 10 MOBILE SOFTWARE ENGINEERING

Structure **Page Nos.**

10.0 INTRODUCTION

The rapid proliferation and ubiquity of mobile, smart devices in the consumer market has forced the software engineering community to quickly adapt developmental approaches conscious of the novel capabilities of mobile applications. The

combination of computing power, access to novel on board sensors and ease of application transfer to market has made mobile devices the new computing platform for businesses and independent developers. However, the growth of this new computing platform has outpaced the software engineering work tailored to mobile application development.

These powerful development tools and frameworks greatly simplify the task of implementing a mobile application. However, they are predominantly focused on the individual developer who is trying to create an application as quickly as possible. For small and medium-sized mobile applications that can be built (and easily updated) by a single developer, they represent a vast improvement on the previous generations of tools, and encourage developers to adhere to the important principles of abstraction and modularity that are built into the platform architectures.

However, as mobile applications become more complex, moving beyond inexpensive recreational applications to more business critical uses, it will be essential to apply software engineering processes to assure the development of secure, high-quality mobile applications. While many "classic" software engineering techniques will transfer easily to the mobile application domain, there are other areas for new research and development.

10.1 OBJECTIVES

After going through this unit, you should be able to learn,

- Transition from Design to Coding of Mobile Applications
- Elements of Mobile Applications
- Approaches to the Development of Mobile Applications

10.2 TRANSITION FROM DESIGN TO CODING OF MOBILE APPLICATIONS

Transition holds information about animations that will be run on its targets during a scene change. Subclasses of this abstract class may choreograph several child transitions they may perform custom animations themselves. Any Transition has two main jobs:

(1) capture property values, and (2) play animations based on changes to captured property values. A custom transition knows what property values on view objects are of interest to it, and also knows how to animate changes to those values.

10.2.1 Create a transition

Once you have defined the starting scene and the ending scene you want to change between, you need to create a Transition object that defines an animation. The framework enables you to specify a built-in transition in a resource file and inflate it in your code or to create an instance of a built-in transition directly in your code.

10.2.2 Transition of Design to Coding using Wireframes

Wireframes are intended to use to demonstrate the functionalities, user interactions and screen flows, without explicitly specifying the actual screen components should

look like and how the components should behave in order to keep the upfront development effort and cost in its lowest minimum.

Android wireframes are screen sketches of Android apps (figure 10.1). It helps you present and explain design ideas of applications to customers, which ultimately leads to a consensus on the ideas transition into proposed.

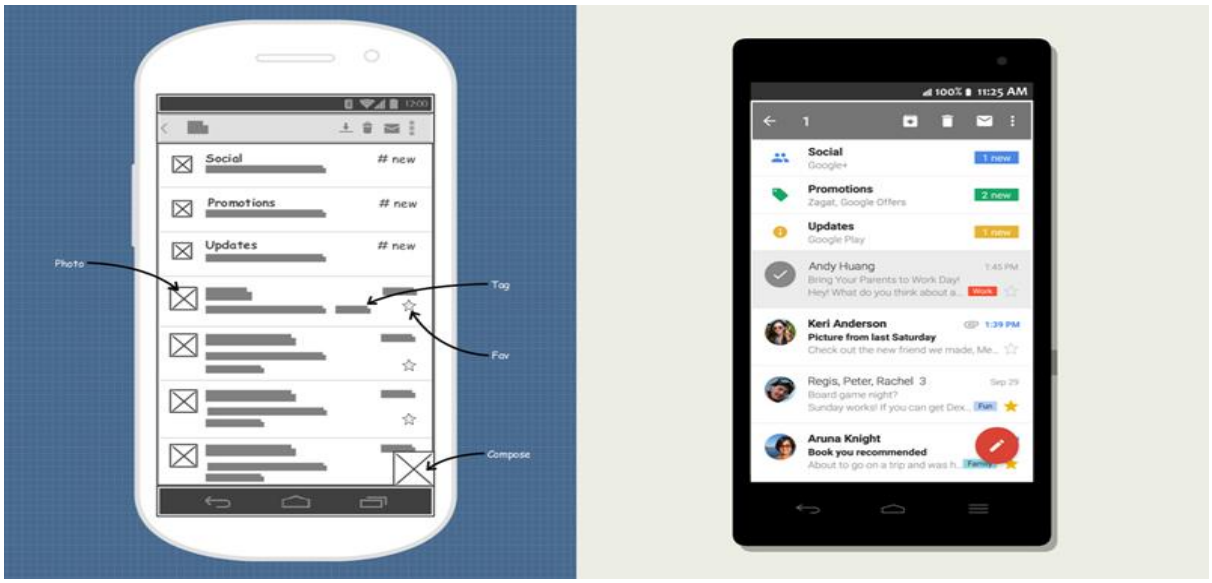


Figure 10.1:Android Wireframe Screen Sketches

10.2.3 Prototyping mobile UI animations – the takeaway

Animated mobile transitions can be charming, useful and user-centric. They guide users through a mobile application experience and ensure that both user goals and conversion goals are catered to. But mobile UI designers have to resist the temptation to animate for mobile animation's sake. Animations should always be relevant, targeted and with a purpose. That way, your UI will be as effective as these inspiring examples of animated mobile transitions.

Transitions should offer a truly exclusive experience with a sliding, flipping, turning or popping effect and Effects any event can have a popping, fading, sliding, folding, dropping, bouncing effect – and many more.

10.2.4 Application Wire framing Features

Speed up your wire framing process with a modern tool. Designing wireframes that behaves like the real deal and start getting real feedback from users testing your app wireframe. Accelerate innovation. Avoid rework. Launch the right app.

A final prototype is made with all of the features incorporated into the application design and sent to the development team for coding. After this one iteration, an MVP is made and send to the client and if and as there are changes required, they are made in the final design of the application. Every mobile application is made a success through its functionality, code and calculations, regardless of how it looks and works", this sounds like an absolute fluke.

After all, today when we talk about mobile applications there is an immense need for an exceptional UI and UX as much as a perfect backend code. In fact, UI/UX and

visuals are the reason why some mobile applications become an instant hit in the market and some are never downloaded in the first place.

Animated transitions can make the difference between a great mobile app and transitions – those little animations that make UI elements visible or invisible – often go unnoticed, but when executed right they contribute to a seamless user experience.

UI is the User Interface and UX stands for User Experience. For the fundamentals of it, the app UX design process steps are more concerned with user retention which is experience based and more subjective whereas, the UI design development process is consists of the visuals and graphics of an app. UI is a sub-part of UX, although the UI/UX design is an art, the mobile app design process of creating the graphics of an application is more systematic and technical.

Transition's mobile applications support increase your business agility and taking advantage of mobile technologies improves productivity and provides added value. Companies who embrace today's world as "mobile-first" find that their processes run smoother and simpler, while their business is more scalable, with mobile applications. Transition's system architecture ensures our mobile apps are secure, fully functional, and embedded into your software and business workflows.

Building on this robust and reliable foundation, we develop user-focused, future-proof iOS, Android and cross-platform mobile applications in short-timescales.

10.2.5 Animate layout changes using a transition

Android's transition framework allows you to animate all kinds of motion in your UI by simply providing the starting layout and the ending layout. You can select what type of animation you want (such to fade the views in/out or change the view sizes) and the transition framework figures out how to animate from the starting layout to the ending layout.

The transition framework includes the following features:

- **Group-level animations**: Apply one or more animation effects to all of the views in a view hierarchy.
- **Built-in animations**: Use predefined animations for common effects such as fade out or movement.
- **Resource file support**: Load view hierarchies and built-in animations from layout resource files.
- **Lifecycle callbacks**: Receive callbacks that provide control over the animation and hierarchy change process.

The basic process to animate between two layouts is as follows (figure 10.2):

1. Create a Scene object for both the starting layout and the ending layout. However, the starting layout's scene is often determined automatically from the current layout.
2. Create a Transition object to define what type of animation you want.
3. Call TransitionManager.go() and the system runs the animation to swap the layouts.
4. The diagram in figure 1 illustrates the relationship between your layouts, the scenes, the transition, and the final animation.

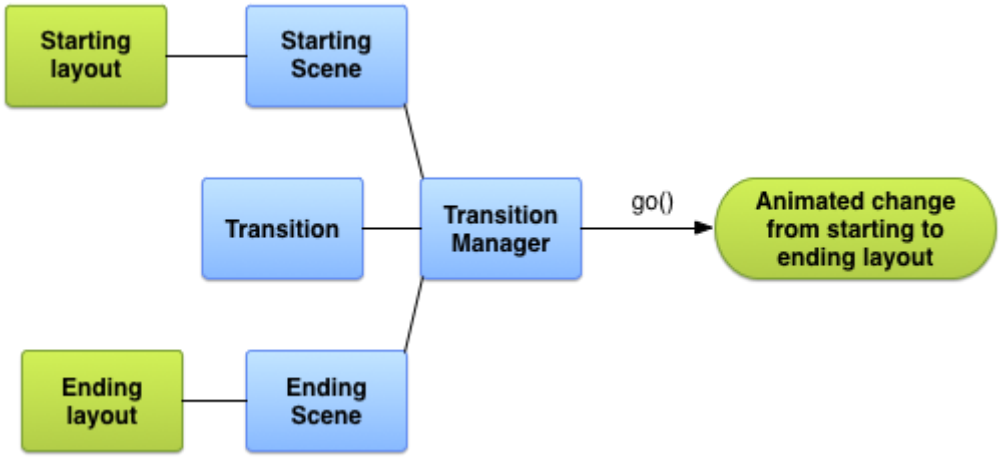


Figure 10.2:Basic illustration of how the transition framework creates an animation

Create a scene

Scenes store the state of a view hierarchy, including all its views and their property values. The transitions framework can run animations between a starting and an ending scene.

You can create your scenes from a layout resource file or from a group of views in your code. However, the starting scene for your transition is often determined automatically from the current UI.

A scene can also define its own actions that run when you make a scene change. For example, this feature is useful for cleaning up view settings after you transition to a scene.

Create a scene from a layout resource

You can create a Scene instance directly from a layout resource file. Use this technique when the view hierarchy in the file is mostly static. The resulting scene represents the state of the view hierarchy at the time you created the Scene instance. If you change the view hierarchy, you have to recreate the scene. The framework creates the scene from the entire view hierarchy in the file; you cannot create a scene from part of a layout file.

To create a Scene instance from a layout resource file, retrieve the scene root from your layout as a ViewGroup instance and then call the Scene.getSceneForLayout() function with the scene root and the resource ID of the layout file that contains the view hierarchy for the scene.

Define layouts for scenes

The code snippets in the rest of this section show you how to create two different scenes with the same scene root element. The snippets also demonstrate that you can load multiple unrelated Scene objects without implying that they are related to each other.

The example consists of the following layout definitions:

- The main layout of an activity with a text label and a child layout.
- A relative layout for the first scene with two text fields.

- A relative layout for the second scene with the same two text fields in different order.

The example is designed so that all of the animation occurs within the child layout of the main layout for the activity. The text label in the main layout remains static.

This layout definition contains a text field and a child layout for the scene root. The layout for the first scene is included in the main layout file. This allows the app to display it as part of the initial user interface and also to load it into a scene, since the framework can load only a whole layout file into a scene.

The layout for the second scene contains the same two text fields (with the same IDs) placed in a different order.

Generate scenes from layouts

After you create definitions for the two relative layouts, you can obtain a scene for each of them. This enables you to later transition between the two UI configurations. To obtain a scene, you need a reference to the scene root and the layout resource ID.

In the app, there are now two Scene objects based on view hierarchies. Both scenes use the scene root defined by the Frame Layout element in res/layout/activity_main.xml.

Create a scene in your code

You can also create a Scene instance in your code from a ViewGroup object. Use this technique when you modify the view hierarchies directly in your code or when you generate them dynamically.

To create a scene from a view hierarchy in your code, use the Scene (sceneRoot, viewHierarchy) constructor. Calling this constructor is equivalent to calling the Scene.getSceneForLayout() function when you have already inflated a layout file.

The following code snippet demonstrates how to create a Scene instance from the scene root element and the view hierarchy for the scene in your code:

Create scene actions

The framework enables you to define custom scene actions that the system runs when entering or exiting a scene. In many cases, defining custom scene actions is not necessary, since the framework animates the change between scenes automatically.

Scene actions are useful for handling these cases:

- Animate views that are not in the same hierarchy. You can animate views for both the starting and ending scenes using exit and entry scene actions.

- Animate views that the transitions framework cannot animate automatically, such as ListView objects.

To provide custom scene actions, define your actions as Runnable objects and pass them to the Scene.setExitAction() or Scene.setEnterAction() functions. The framework calls the setExitAction () function on the starting scene before running the transition animation and the setEnterAction () function on the ending scene after running the transition animation.

10.2.6 Apply a transition

The transition framework represents the style of animation between scenes with a Transition object. You can instantiate a Transition using several built-in subclasses, such as AutoTransition and Fade, or define your own transition. Then, you can run the animation between scenes by passing your end Scene and the Transition to TransitionManager.go().

The transition lifecycle is similar to the activity lifecycle, and it represents the transition states that the framework monitors between the start and the completion of an animation. At important lifecycle states, the framework invokes callback functions that you can implement to make adjustments to your user interface at different phases of the transition.

- An **enter** transition determines how views in an activity enter the scene. For example, in the *explode* enter transition, the views enter the scene from the outside and fly in towards the center of the screen.
- An **exit** transition determines how views in an activity exit the scene. For example, in the *explode* exit transition, the views exit the scene away from the center.
- A **shared elements** transition determines how views that are shared between two activities transition between these activities. For example, if two activities have the same image in different positions and sizes, the *changeImageTransform* shared element transition translates and scales the image smoothly between these activities.

Android supports these enter and exit transitions:

- *explode* - Moves views in or out from the center of the scene.
- *slide* - Moves views in or out from one of the edges of the scene.
- *fade* - Adds or removes a view from the scene by changing its opacity.

Any transition that extends the Visibility class is supported as an enter or exit transition. For more information, see the API reference for the Transition class.

Android also supports these shared elements transitions:

- *changeBounds* - Animates the changes in layout bounds of target views.
- *changeClipBounds* - Animates the changes in clip bounds of target views.
- *changeTransform* - Animates the changes in scale and rotation of target views.
- *changeImageTransform* - Animates changes in size and scale of target images.

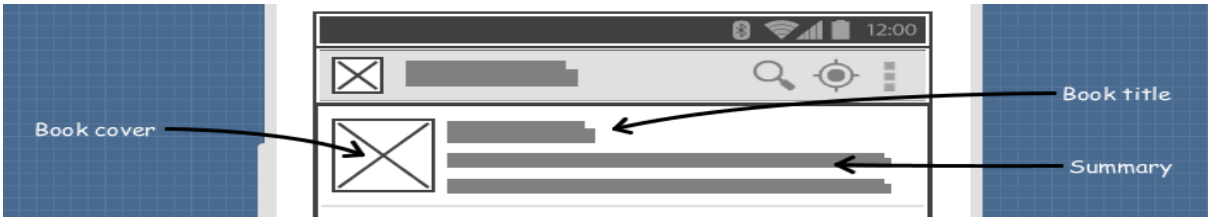
10.2.7 How to use Android wireframe effectively?

The use of wireframe can bring many benefits to both the development team and clients, but this is the case only when you use it wisely and properly. A typical misuse of wireframe is to treat it as a replacement of screen design. This makes the production and refinement difficult and costly, reducing the usefulness of Wireframing. In this section, we will go through some of the effective Wireframing tips.

- A wireframe is intended to be simple and just enough. It is simple so that it can be produced quickly and easily, and makes no hesitation for a discard and re-work. The low-fi presentation also makes it more comprehensive and communicative. Therefore, do not need to spend too much time on beautifying the drawing, aligning things, or using pretty typology and etc.

- In a wireframe, instead of showing any actual content, we can replace a large chunk of text (the actual context) with a placeholder of text. This is to avoid time being spent on preparing the content unnecessarily, and to prevent the readers from being distracted by the text content. But if the displaying of text is needed, you may consider placing some dummy text there instead. You can easily find a dummy text generator on the internet.

- The use of annotation helps you describes an element (e.g. "Company logo") or to explain something related to its behaviour (e.g. "Hide in 5 seconds"). Use it if necessary. But again, don't attempt to document each of the wireframe elements. You should only use annotation whenever it necessary.



Wireframes can be hand-drawn, but we usually create wireframes with software for more efficient and easier to manage of our works. Besides, some wireframe software provides you with features that paper-and-pencil cannot accomplish. Here are three of them:

- State - The wireframing tool of Visual Paradigm supports the concept of state, which allows you to create a child wireframe based on an existing one. It is not only save you time in creating a screen flow with a sequence of similar child wireframes, it also makes refinements of the related child wireframes much easier (as we make changes in the initial state of a wireframe, the changes will also be reflected in all its' child states wireframes)

- Storyboard - A storyboard presents the screen flow of a particular scenario. It makes the wireframes more manageable and the presentation much easier.

- Managing wireframes by User story - User story is an agile tool for recording user's concerns and requirements. To include wireframes as part of a user story's scenario shows how user will use the feature in performing part of their job described in that user story. Besides, when developer start implementing the user story, they can check the wireframe to gain ideas about user's expectations.

Once these screens are sent off to the client, developers begin a **collaborative design** process. They often find that clients like different parts of each concept – the button from this one, the colours in that one – so we'll try to pull these together and refine them into a unified design, rather than a Frankenstein's Monster of various designs.

This gives us a finalized visual design, which can be combined with the UX flow to create a working **prototype**. This is essentially a collection of images of each screen, with tappable hotspots that make it possible to move from screen to screen as if you're using the real application. It can be installed on a device, meaning it's accessible from an icon just like any other application, and often find that people don't realize it's not a fully-functional application.

10.2.8 Benefits of using wireframe

1) Clarify user interface 2) Early consideration of usability 3) Engaged clients 4) Cost-efficient

Full-blown, high-fi screen designs takes time to develop, which lead to high development cost. Wireframing is easy to create and edit. It is an inexpensive way to create basic screen sketches. It also makes tweaking or even overhauling sketches simple and inexpensive.

Check Your Progress 1

1. Transition holds …..….. that will be run on its targets during a scene change. ……

10.3 ELEMENTS OF MOBILE APPLICATIONS

The Elements of Mobile Application Development

Knowing the essential elements in Mobile Application Development will help increase your popularity and following. You need a strategy that will improve the customer engagement and make the mobile application development process a success.

You may not be the first person to come up with an application idea, and in most cases, developers tend to improve on what already exists. For this reason, there are many products in application stores that have similar purposes, and as you can imagine, the nature of the competition you are likely to face is high.

Entrepreneurs in tech valleys all across the globe have already paid attention to this and are trying to make the best of the available opportunities. There's no denying the fact, that mobile applications have become a part of our everyday lifestyle and most of us remain dependent on them constantly for taking care of several critical operations.

As easy as it may sound using them, it is a whole different ball game when it comes to developing each application. Few things to keep in mind while developing a mobile application:

10.3.1 Product Strategy

The first component of a successful mobile application is a thorough product strategy that solves your immediate business goals and addresses long-term growth opportunities.

There are many business cases for mobile applications and countless opportunities to streamline processes or introduce new customer value streams with mobile solutions. The key is choosing one that makes the most impact within the context of a broader business strategy.

The same type of project analysis techniques used in other areas of business can apply to mobile application development prioritization. It's essential to start thinking about the mobile application development project with this frame of mind, so one can develop a product that delivers intentional and measurable business value.

10.3.2 Comprehending what your application needs to deliver

Before developing an application for your business, give in some time to understand what all aspects of your business the application needs to cover. It is important to place the priorities of your business out there through your application because this is how both your business and application will be able to succeed in the long run.

10.3.3 User Interface Design (Ui)

Work on your User Interface Design (UI) and application usability, which are the subsets of user experience design. You will have to come up with decent and useful features.

10.3.4 Screen Resolution

When building your mobile application, you should also consider the screen size and resolution of various mobile devices that operate on the platforms you wish to launch your application. In terms of resolution, you have to achieve the right pixels per inch or standard screen resolution for your application. More about this information is available on the application store technical guidelines.

10.3.5 Graphics has to be appealing

Appearance is very vital to this process and in the end, the outcome has to be easy and comfortable to the user's eyes. The graphics of your application have the tendency to determine its failure and success, so it will be a big mistake to ignore it. It's necessary that the application automatically adjusts to all screen sizes without any issues and looks just as good when users are accessing it on a mobile phone screen of any size.

Applications with quality graphic art or design are captivating to the eyes of the users. When you do not wish to use text, it is essential to know that a picture is worth a thousand words. Therefore, you need to use high quality and intuitive images, animations, visuals and designs to keep your users engaged.

10.3.6 User Experience Design (UX)

The interaction between your application and the users should create positive emotions and attitudes. It is more about how users feel when they are using your application. You need to design your application with the users in mind. You need to

come up with intuitive and interactive designs, making east easy for the user to know the right thing to do.

Look in to the details of the User requirements and identify a UX strategy

UX is a two-part process beginning with a scientific, disciplined approach to understanding the users' problems, from their perspective. Identify and articulate the problem areas to produce a much more successful solution. Analyze the output of this initial user research and problem definition in a well-articulated UX strategy.

10.3.7 Good Speed

The speed matters the most. If the application hangs and stuck during the use, there's a great possibility that users get annoyed and stop using the application. Since there are more than enough alternatives and options available in the market, no one would wait for a snail-motion application and quickly moves to another better alternative. Everyone now wants a simple, easy, and quick way to their solutions.

10.3.8 User-friendly Navigation

Most users tend to show less patience to an application that has poor navigation. It pays to have a simple application that allows users to find and use what they need with ease. You should aim to provide a clutter-free experience. Focus on delivering a simple and easy to use application.

You do not need a complex interface and navigation system to appear modern, and sometimes simplicity is the ultimate sophistication. Most users do not have all the time to waste trying to explore your application; there are enough puzzle games in the application store for them to play.

10.3.9 Application Content

In Mobile Application Development, you need text to label your buttons, provide guidelines and explain specific terminologies. The content in your application should be explicit and precise. Stuffing keywords for SEO optimization may distort the message you are trying to communicate. Applications with updated content and information look new to all users, potential and repeat customers. Most experts talk about delivering accurate and specific content.

There are a few different routes you can take to make your online product ready for mobile devices. The option you choose largely depends on three main factors:

- The quality of the experience you want your application to have.
- The complexity of the features you need for your application to work.
- Your budget.

10.3.10 Mobile device performance

- **Mobile Application Size**

Size matters especially when the user has limited storage capacity and slow internet. A well-designed and straightforward application tends to consume less space. Size matters because it keeps the mobile device's CPU and RAM free, making your application easy and fast to load.

- **Battery and Processor Usage**

Some applications will overwork the processor leading to high battery usage. Gaming applications are known to consume more juice from a mobile device. The development of your mobile application should be in such a way that it does not consume too much energy.

10.3.11 Social Sharing Option

When listening to music, reading an article or playing a game, some users may want to share the experience with others on social media. For this reason, an application should be able to integrate with popular social networking sites for easy and fast sharing. With a built-in viral mechanism that promotes social sharing, you let your customer market the application for you.

10.3.12 Security

The security of your application is the most crucial factor to consider above all other considerations. You need to think about all the possible ways you can make your users safe. This factor takes into account the type of technology you use to encrypt user information and data.

10.3.13 Power Consumption – many aspects of an application affect its use of the device's power and thus the battery life of the device. Dedicated devices can be optimized for maximum battery life, but mobile applications may inadvertently make extensive use of battery-draining resources.

10.3.14 Compatibility

If your mobile application is to be available for all users, then you'll have to consider the iOS or Android versions available. As one upgrades from one version to the other, application updates should also be available. Creating unique features for various platform versions makes it easy for the user to see the change and keep them engaged.

Overall, your application has to be simple and loads quickly. Application users are always looking for something new and attractive to try. The success of your project relies on the ten most important elements of mobile application development.

10.3.15 Simplicity

Simplicity is an increasingly increased demand for mobile application development. Not all of the mobile application users are pro. Give them the ease of use and thus increase the chances of success for your application. Make your application easier, simpler and quicker.

10.3.16 Offline function

Making an application usable even without the network coverage is definitely a plus. The availability of the internet is questionable in some areas of the world. If your application works with internet support, there are more chances of increased use and popularity. Also, this would enable the application to have more users from developing countries.

10.3.17 Have a high performance

A poorly performing mobile applications get usually abandoned by users. An important component that gauges the success of an application is how the application perform. A majority of mobile application users stops using an application that perform poor more than twice. Performance is an extremely important component for application success, and only high-performance applications get in the list of successful applications.

10.3.18 Provide a Feedback Option

An easy pathway for feedback contributes to building a right kind of mobile application development process. This also helps to improve user experience. Make sure you include a feedback and review option so that your application can make business better by hearing from what people about your application. In addition, this also helps the mobile application developers to improve with regards to complaints and make improvements.

10.3.19 Regular Updates

Make quality improvements and add necessary features because regular updates based on the users' experience and reviews engage the users in the most practical way. Frequent updates will also help improve the rating of your application. with every update, you may deal with some new suggestions, complaints, and feedback. Keep up with them and stay successful with your mobile application.

10.3.20 Marketing

A competent mobile application development company never underestimates the importance of marketing. It is the final ingredient for the success recipe of a mobile application. Through marketing, more people know about the solution that you provide through your application. Contrarily, if there is a lacking of good marketing, no matter how good your application is, people won't know about it. This ultimately will add to application failure. Make sure you do the right marketing of your application. It is essential to convey the word to clients, users, and customers.

Check Your Progress 2

1. Write few elements of mobile application development?
   ……

2. What is user friendly navigation?
   …….

10.4 APPROACHES TO THE DEVELOPMENT OF MOBILE APPLICATIONS

The purpose of this topic is to put the spotlight on different mobile development approaches and guide to the best way to develop applications based on what you need.

Types of Mobile Development Approaches are;
1) Native Application Development **2)** Cross-Platform Application Development **3)** Hybrid Application Development **4)** Rapid Mobile Application Development: RMAD **5)** Progressive Web Applications: PWAs

A Step in the Right Direction
Having considered the different types of mobile development approaches, their main features, and advantages, it is obvious that every approach is different and so will be the resulting application.

For project owner, it should be in the best interest to define the purpose of the mobile application in accordance with the exact needs of the target audience. This will help to estimate the project timeline, the financial and technical resources needed, after which it will be able to choose the most favourable development approach to follow.

Applications are distinct in their own ways. If the goal and desire is to develop a long-term project, an application that will be responsive, have a superior quality user experience, smooth performance, and reliable security, then the native development approach is very advisable. On the other hand, if looking to compromise in budget plans or you desire speed in development time, it can consider the cross-platform approach.

10.4.1 Native Application Development

Native application development is the use of platform-specific programming languages, software development kits, and development environments offered by the OS providers. In other words, if planning an application for iOS and Android, these applications will be developed separately for each platform with the use of completely different technology stacks. Native applications support all available features of the platform and compatible devices. Furthermore, native applications have higher performance and responsiveness.

Prime Reasons to Choose Native Mobile App Development

However, there are ample other convincing reasons to embrace this platform for app development:

1. Access to Complete Device Features
2. Scalability
3. Offline Performance
4. Stability
5. Cost

Advantages of Native Application Development Approach

- Overall, native applications have the best performance
- Platform-specific UI implementation
- 100% support of OS features
- Total access to hardware-related features
- Clear application update path and supported toolset
- Native applications are highly reliable, secure, and responsive

Let’s take a closer look at what Apple and Google offer as a native mobile technology stack for their platforms.

Programming Languages: Objective-C and Swift

Objective-C

Objective-C is one of the high-level and object-oriented programming languages used for developing native applications for iOS. It is modeled like the C language, possesses a dynamic runtime environment, and it was predominant for the development of iOS applications initially, before the inception of the Swift programming language.

There are still plenty of mobile enterprise applications operating on the Objective-C code, irrespective of the rise of Swift. This could be because it is a difficult process to switch to Swift, especially if you have existing applications written in Objective-C that need regular updates. All the same, Objective-C is very well-preferred by some developers for projects of complexity and software engineering challenges.

Some other advantages of Objective-C include:

- Works with C++
- Highly recommended and well-tested language
- Expressive message syntax
- Simple and effective use of private APIs
- Multiple third-party libraries
- Large community support

Swift

The prominence and rise of the Swift language are as a result of its use for creating Apple iOS and Linux OS applications. It is proclaimed to be a multi-purpose and multi-paradigm programming language. However, the main reason for its popularity is effective performance, when compared to other favorable languages.

Swift has the advantage of being an open-source programming language with code less prone to error because it gives inline support that can manipulate data and text strings. Additionally, Swift includes dynamic libraries. These libraries can be uploaded directly to memory, thereby slashing the size of applications developed and significantly increasing their performance.

The employment of Swift to develop applications for iOS simplifies the task of development, plus, maintenance is stress-free. It also enhances code reusability, and readability.

Some other advantages of Swift include:

- Multi-feature programming language
- Operates very fast
- Compatible with Objective-C
- Functions with Cocoa Touch frameworks and Apple’s Cocoa
- Multi-device support
- Community support

Toolset: AppCode and Xcode

There are two major toolsets used for iOS application development: Xcode and AppCode.

AppCode

AppCode is a fast upcoming tool for programing native iOS application and comes as an alternative to Xcode. Some developers have put out the word that it enables faster coding than its rival Xcode. Nonetheless, it has a disadvantage. AppCode can only design interfaces with the use of written code, unlike Xcode. AppCode supports various programming languages for the Swift development environment, such as JavaScript, C, C++, and Objective-C.

Apple Xcode

Xcode can be named the official IDE of Apple. Therefore, it is an appropriate choice of a toolset for iOS application developed. The full-featured development environment of Xcode enables the production of not only mobile applications, but desktop applications too. To buttress being a good user interface development tool, it works best for gaming applications.

Xcode-developed applications can work on numerous Apple Inc products. Moreover, it is equipped with the support for repositories from Git, documentation, instrumentation, debugging tools, and it can develop user interfaces by using a graphical editor. Furthermore, it is compatible with a list of programming languages such as C, C++, Python, Objective-C, Objective-C++, Java, AppleScript, Ruby, Rez, Swift, and so on.

Android Application Development Stack

Programming Languages: Java, Kotlin

Kotlin

Kotlin is a more recent programming language in comparison to Java. In any case, it's an appropriate technology option for Android Studio that is fast becoming more stable. Purposefully, it is not just object-oriented but also an open-source programming language. Furthermore, it is highly-functional and supports higher-order functions.

Kotlin has powerful syntax with fewer errors than other languages used to develop native Android applications. Equally important, it is faster to code, clean, lightweight, and less verbose. Originally based on the Java Virtual Machine, interoperability with Java makes it possible for developers to combine codes from Java-based projects with Kotlin.

Benefits of Kotlin are:

- Runs on multi-platform
- Concise and secure code
- Interoperability with Java
- Functions can be extended
- Big community support
- Easy code maintenance

Java

There is no debate to it that Java is said to be one of the most widely used programming languages. Without compromising, it is dominantly the most used class-based, object-oriented language for developing Android applications. Java is ideal for mobile and web applications, Big Data, among others. On top of that, it is also famous for rich open-source tools and libraries that facilitate the easy and fast development of top-notch applications.

Top development speed and product quality are the most prominent advantages of Java. However, it has a downside of being comparatively slow and consuming memory space. Nevertheless, you can consider Java for complex mobile projects, especially if you want scalability and robustness. Big players such as Google continuously work to make the Java language better. The main parts of the Android OS are programmed with Java.

Further advantages of Java:

- Highly secure
- Great network capability
- Portable and scalable code
- Automatic management of memory
- Robust programming language
- Easy to use and compile
- Not limited to any platform

Toolset

For a native integrated development environment, Android Studio and Eclipse are the most favoured software.

Eclipse

The Eclipse software development kit (SDK) is another free, open-source mobile development software. Originating from IBM visual and primarily coded in Java, its main purpose is for the development of Java applications. The good news here is that developers can extend their use of Eclipse by coding plug-in modules specially for themselves, or by using other plug-ins as development toolkits.

Android Studio

The integrated development environment from Google is called Android Studio. Usually, developers employed it to edit written code, handle performance tooling, and debugging. With such characteristics as flexible development and an immediate build/deploy system, Android studio enables making unique, high-quality Android applications. Notably, there is a major difference if compared to Xcode, which functions specifically for Mac OS. The Android studio works with other PC operating systems.

10.4.2. Cross-Platform Application Development

A cross-platform mobile application, as the name implies, simply means an application that can run on different platforms. The cross-platform approach is a good alternative to native development, as it solves the challenge of creating a separate application for each mobile platform. In other words, this concept allows to deliver applications across multiple platforms simultaneously with the use of languages and tools different from native toolsets offered by Google and Apple.

Advantages of Cross-Platform Development Approach

- Uniformity across all platforms
- Effective for budget control or low funding
- Implementation is easy
- Publishing can be at once for all platforms
- The source code can be reusable
- High demographic coverage
- Fast time to market

10.4.3 Hybrid Application Development

The blend of native and web applications is known as hybrid app or hybrid application. In a computing device, native applications are installed for a certain program. But, to run web applications, multiple frameworks are available over the internet. Hybrid apps are often mentioned in the context of mobile computing.

Hybrid application development is classified as a form of cross-platform. In this development approach, the application core is developed by the use of standard web technologies and tools like JavaScript, CSS, and HTML5, and then executed within a native shell.

Resulting applications from the hybrid development approach have the speed of a regular web application and the user experience similar to any type of native mobile app. Notably, the use of a single code base enables deployment to all platforms lowering the cost if compared to native applications. Lastly, hybrid applications with hardware-dependent features have access to device hardware components and native platform libraries.

Advantages of Hybrid Development Approach

- Single development team
- Short time to market
- The easy portability of code
- Capable use of hardware components
- Same user experience as a native mobile application
- Lower cost of development
- Ability to work on and offline

10.4.4 Rapid Mobile Application Development: RMAD

The Rapid Mobile Application Development approach is used to develop cross-platform applications in a short time. It involves the usage of specific code-free or low-code development tools to program simple applications for various business solutions.

To start with, this approach is very similar to rapid application development methodology based on little panning, initial prototyping, recyclable software components, and the use of the adaptive process.

It can be applied with several development platforms, examples are Alpha Software, MobileFrame, and MobileSmith. They all possess the no-code and low-code options.

Advantages of MobileSmith:

Rapid mobile developers declare the functions and features of applications on the frontend and the backend translates the specifications into code. In this case, the frontend uses metadata to function. Effectively, this is done by summarizing the major

information about the functions of an application like asset managers or user interface elements in a database. Thus, it cuts off the need for database coding.

Advantages of RMAD Approach

- Secure
- Works online and offline
- Good backend data integration
- Works on all platforms including web
- Code-free development
- Little experience as a developer is need
- Reusable components
- Low cost of production

10.4.5 Progressive Web Applications: PWAs

The last type of development approach, but not the least, which we cannot exclude is the Progressive Web Application development. In this case, the engagement of HTML is principal here.

Progressive Web Apps are the future of the world. Progressive Web Apps merge the web world and the native app world. Simply put, PWAs are the native app like web applications that are making everyone's lives much easier.

Advantages of Progressive Web Applications

- Easy application maintenance
- The use of a single codebase
- Mobile-friendly UI
- You don't need an application store to host or download
- Works on and offline

Check Your Progress 3

1. Which application development is the use of platform-specific programming languages?
   ……

2. Which approach can be used to run application the on different platforms?
……
3. Which application Development approach is used to develop cross-platform applications in a short time?
……

10.5 SUMMARY

Mobile applications are vital to business growth. Therefore, the process and strategy of mobile application development must be designed in a way that various parameters are taken into consideration. It hopes that found the outlines of the transition of design to development, important elements to consider in development with various approaches are recommended for any application development project useful. It is more involved than simply having an idea for a product or service that it had like to introduce by just building an application.

10.6 SOLUTIONS / ANSWERS

Check Your Progress 1

1. Transition holds…..that will be run on its targets during a scene change. information about animations

Check Your Progress 2

1. **Write few elements of mobile application development?**

   Speed, Application Content, Graphic Design, Navigation, Mobile Device Performance

2. **What is user friendly navigation?**

   Focus on delivering a simple and easy to use application.

Check Your Progress 3

1. **Which application development approach is used in platform-specific programming languages?**

   Native Application Development

2. **Which approach can be used to run application the on different platforms?**

   Cross Platform Development

3. **Which application Development approach is used to develop cross-platform applications in a short time?**

   Rapid Mobile Application Development

10.7 FURTHER READINGS

1. Software Engineering – A Practitioner's Approach by Roger S. Pressman
2. Mobile Application Development: A Developer Survey by Agrawal, S. and A.I. Wasserman
3. Agile Project Management with Scrum by Schwaber, K.

Reference Websites

http://developer.apple.com/iphone/index.action

http://developer.android.com/guide/index.html

http://scrum.org