CHAPTER 1

Introduction to Mobile Application Development

1.1 Overview of Mobile Application Development

Mobile application development is the process of making software for smartphones, tablets and digital assistants, most commonly for the Android and iOS operating systems. The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser. The programming and markup languages used for this kind of software development include Java, Swift, C# and HTML5.

There are two dominant platforms in the modern smartphone market. One is the iOS platform from Apple Inc. The iOS platform is the operating system that powers Apple's popular line of iPhone smartphones. The second is Android from Google. The Android operating system is used not only by Google devices but also by many other OEMs to built their own smartphones and other smart devices.

1.2 Web Development Vs App Development:

Aspect	Web Development	App Development
Definition	Creating websites and web-based applications accessible through web browsers.	Creating mobile applications that can be downloaded and installed on mobile devices.
Purpose	Provide online presence for businesses or individuals.	Provide userfriendly experience for user and increase engagement.
Design	Responsive design, optimized for search engines, and fast loading.	User-friendly interfaces, and device-specific features like push notifications, GPS, and camera.
Functionality	Accessible from different devices with different screen sizes	Work on different operating systems, and utilize device-specific features.
Target Audience	Desktop and laptop users with longer attention spans.	Mobile users who expect instant gratification and demand convenience.
User Behavior	Looking for information or products.	More likely to make in-app purchases, and spend more time on apps than mobile websites.
Technology Requirements	HTML, CSS, JavaScript, Angular or React	Java, Kotlin, Swift, Flutter, React Native, or Ionic
Skillset	Strong understanding of web technologies	Strong understanding of mobile

	and frameworks.	operating systems and programming languages, familiarity with app development frameworks
Cost	Less expensive	More expensive due to platform-specific development.
Time	Quicker to develop	Longer to develop and test due to compatibility with different devices and operating systems.

1.3 Types of Mobile Applications(Native, Hybrid, Web)

Native Apps

Native apps are built for specific operating systems on mobile devices. That means the app can run on Android devices or Apple iOS devices but not both. This is why many businesses hire software developers who specialize in particular operating systems. Native apps can be developed with a wide variety of programming languages, from Java to Python to C++.

There are a number of advantages to native apps. The chief benefit is performance. These apps are faster than hybrid apps, as well as more consistent. Additionally, the user experience (UX) tends to be superior, because they use the device's user interface (UI). These apps can also access different features that come with the device itself, such as bluetooth and GPS.

Hybrid Apps

Instagram and Gmail are 2 extremely popular examples of hybrid apps. These are technically web apps, but they behave and act like native apps. They run within an app-embedded web browser. Unlike native apps, however, hybrid apps can function on multiple platforms and operating systems.

The main advantage of choosing hybrid apps is that development is streamlined since businesses only need to build one version of the product. This also means that the development process is typically quicker than that of native apps, as well as less costly. Apps can also function offline, and updates are easier to facilitate.

Web Apps

Web apps run in a web browser. They are accessed on a mobile device and don't require downloading on the part of the user. Commonly, developers use traditional web development languages like HTML5, JavaScript, CSS, and others to create web apps.

As with hybrid apps, a major advantage of web apps is that they don’t require multiple codebases or customization to unique operating systems. Therefore, development time can be fairly quick, and the app won’t be expensive to create. They also behave responsively, adapting to the UI of the specific device on which it is functioning.

CHAPTER 2

Overview of mobile platform

2.1 Mobile Application Platforms(IOS, Android, Cross-platform)

2.2 Choosing the right platform

Mobile app development has become an essential part of business strategies for companies of all sizes. However, one of the most critical decisions you'll need to make is choosing the right platform for your mobile app. Whether it's iOS, Android, or both, this decision can significantly impact the success of your app.

The factors to consider when choosing the right platform for mobile app development.

Define Your Target Audience – Geography

Understanding your target audience is the first step in choosing the right platform. Different platforms attract different demographics. iOS users tend to be more affluent and willing to spend money on apps, while Android has a larger market share but is more diverse in terms of user.

App Purpose and Functionality:

Consider the purpose and functionality of your app. Some apps may require specific features or hardware only available on one platform. For instance, if your app relies heavily on features like augmented reality (AR), you may want to prioritize iOS development, as Apple devices often have better AR capabilities.

Development Cost and Resources (Budget)

The choice of platform also depends on your budget and available resources. iOS development typically requires a Mac, and you must develop in Swift or Objective-C. Android development, on the other hand, is more flexible and can be done on various operating systems, including

Windows and Linux. Assess your development team's skills and budget to make an informed decision.

Market Share and Revenue Potential (ROI):

When boosting in-app revenue, developers often adopt different strategies for iOS and Android apps, tailoring their approaches to their target audiences.

iOS users have a reputation for being more willing to spend on applications and tend to be more generous with their spending compared to Android users. According to a market study by Wolfgang Digital, the average Android user spends roughly three times less on apps than their iPhone counterparts.

For iOS: Consider selling the app, implementing premium in-app purchases, or offering purchases that remove ads to cater to the user base's spending habits.

For Android: Focus on in-app purchases or provide options for purchases that unlock access to all application features to align with the preferences of your Android audience.

Development Time:

The time required to develop an app can also vary between platforms. Android app development can sometimes take longer due to the need for extensive testing on multiple device models and OS versions. iOS, with its limited device variations, can streamline the development process.

Maintenance and Updates:

Consider the long-term maintenance and updates your app will require. You'll need to keep up with changes in operating systems, device models, and user expectations. Factor in the cost and effort of maintaining your app on your chosen platform.

Cross-Platform Development:

To target iOS and Android without developing separate apps, consider cross-platform development tools like Flutter, React Native, or Xamarin. These frameworks allow you to write code once and deploy it on multiple platforms.

2.3 Mobile Application Development Lifecycle

1. The Research/Planning Stage

Your first step should be to dive deep into the research phase- figure out your market and existing competitor apps. Brainstorm on details like the purpose of your app, your target audience, preferred platforms, app development language & frameworks, features your competitor app offers (and if you'd want the same/different ones), a timeline of development & launch, and how you'd want to market it. Strategic business analysis at an early stage will help you with calculating the right ROI, return on investment factor that will eventually guide you in deciding & maintaining the budget.

2. Wireframes

The next step is where you document and wireframe your application. Drawing detailed sketches of your vision of the app (how it would look, the features it would have, etc) greatly helps in bringing it to life in the later stages. Post sketching, wireframes come to refine all your ideas. Now you can arrange all your design components accurately and see if there are any visible usability issues. This step aims to draw a clear understanding of how your proposed functional app will look once all your features & ideas are infused together. Needless to say, let all your creativity kick in at this stage.

3. Technical Feasibility & Back-End Assessment

Once you have a clear understanding of your visuals, you need to consider whether the back-end systems can support your app's functionality. Think- APIs, data diagrams, data integration, servers, push notifications, etc. You will have different requirements depending on whether it is an android app development or an ios app development lifecycle. The platform formats will also need a slightly different back-end working based on whether it's on a smartphone, wearables, tablets, etc. By the end of this step, you might figure out that some of the initial functionality isn't feasible for you. Thus, helping you rethink & review those features.

4. Prototyping

Now that you're at this stage, you should build a rapid prototype. One cannot truly comprehend the experience of an app without touching it & checking how the workflows. Build a prototype and get it into a user's hand to get a quick idea of how it works & feels. Take honest feedback as this will point you in the right direction in future development stages.

5. Design

Once prototyping is done, you can dive into coding. Here, your UX and UI designers take over. Your user experience (UX) designer builds interaction between different design elements while your user interface (UI) designer builds the overall look and feel of your app. What you get here are visual directions & blueprints. These act as a guide for informing your engineers of your vision with the final product and how its interaction with the customers should feel.

6. Develop

This phase starts quite early in some sense. From working on a functional prototype and reviewing it with every step, it’s all a part of the development stage. But as this stage progresses further, the core functionalities are deeply tested. The app is then moved to the deployment phase. All the bugs are also fixed & taken care of. If it’s a complex project, the large application is broken down into smaller modules and dealt with in parts before putting everything together, so that the app is ready for release.

7. Testing

In the mobile app development lifecycle, testing plays a very important role. It’s often a good idea to start testing early as this could help with keeping the final costs in check. The farther one goes in the development life cycle, the costlier it is to fix bugs and other technical issues. At this stage, the app should ideally be tested for every aspect including usability, compatibility, interface, security checks, stress, and performance. Inviting some of your target audience to test it would be a good idea. This feedback from beta users will indicate whether your app can work in a real-world situation.

8. Deployment

Finally, at this point, your app should be ready to submit. Select a day to keep the formal launch. Launching policies will vary for android app development & iOS app development. Keep those factors in check. And lastly, keep in mind that this isn’t the end of your process. In fact, it’s a new beginning of its own. Once the app is out there, feedback will start pouring in, which you will need to accommodate into the future versions of your app. Thus, starting the new development cycle. Keep a check on your resources and your long-term commitment to your project throughout this step.

CHAPTER 3

Mobile Application Architecture and Security

Introduction to Model-View-Controller architecture platform.

The Model-View-Controller (MVC) framework is an architectural/design pattern that separates an application into three main logical components Model, View, and Controller. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other. It was traditionally used for desktop graphical user interfaces (GUIs).

Features of MVC :

It provides a clear separation of business logic, Ul logic, and input logic.

It offers full control over your HTML and URLs which makes it easy to design web application architecture.

It is a powerful URL-mapping component using which we can build applications that have comprehensible and searchable URLs.

It supports Test Driven Development (TDD).

Controller:

The controller is the component that enables the interconnection between the views and the model so it acts as an intermediary. The controller doesn’t have to worry about handling data logic, it just tells the model what to do. It processes all the business logic and incoming requests, manipulates data using the Model component, and interact with the View to render the final output.

View:

The View component is used for all the UI logic of the application. It generates a user interface for the user. Views are created by the data which is collected by the model component but these data aren’t taken directly but through the controller. It only interacts with the controller.

Model:

The Model component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. It can add or retrieve data from the database. It responds to the controller’s request because the controller can’t interact with the database by itself. The model interacts with the database and gives the required data back to the controller.

Popular MVC Frameworks:

Some of the most popular and extensively used MVC frameworks are listed below.

Ruby on Rails

Django

CherryPy

Spring MVC

Catalyst

Rails

Zend Framework

Fuel PHP

Laravel

Symphony

Working of the MVC framework with an example:

Let’s imagine an end-user sends a request to a server to get a list of students studying in a class. The server would then send that request to that particular controller that handles students. That controller would then request the model that handles students to return a list of all students studying in a class.

The flow of Data in MVC Components

The model would query the database for the list of all students and then return that list back to the controller. If the response back from the model was successful, then the controller would ask the view associated with students to return a presentation of the list of students. This view would take the list of students from the controller and render the list into HTML that can be used by the browser.

The controller would then take that presentation and returns it back to the user. Thus ending the request. If earlier the model returned an error, the controller would handle that error by asking the view that handles errors to render a presentation for that particular error. That error presentation would then be returned to the user instead of the student list presentation.

As we can see from the above example, the model handles all of the data. The view handles all of the presentations and the controller just tells the model and view of what to do. This is the basic architecture and working of the MVC framework.

The MVC architectural pattern allows us to adhere to the following design principles:

1. Divide and conquer. The three components can be somewhat independently designed.

2. Increase cohesion. The components have stronger layer cohesion than if the view and controller were together in a single UI layer.

3. Reduce coupling. The communication channels between the three components are minimal and easy to find.

4. Increase reuse. The view and controller normally make extensive use of reusable components for

various kinds of UI controls. The UI, however, will become application-specific, therefore it will not be easily reusable.

5. Design for flexibility. It is usually quite easy to change the UI by changing the view, the controller, or both.

3.2 Introduction to data storage options in Mobile application

Different types of data storage options in mobile application.

list of options of data storage techniques:

-Shared Preferences: It's used to store data in key-value pairs and you can then easily access it from the SharedPreferences interface.

-SQLite: Provides a structured data store and you can use it to store private data in an Android application

-File storage: Used to store raw files in sd card or memory card, storing mp3 files for music, etc.

-Content providers: Used to store semi-structured data with user-configurable data access.

-Cloud storage: Used to store data in third-party data storage, for example, Google Drive.

Shared Preferences

Shared preferences as a data store are used when you can store your data in key-value pairs in the application and you need this data to stay even when the app is closed, for example, user preferences of the application. Suppose, you have a weather application that you built and you allow your user to select the unit degree Celsius or Fahrenheit as a preference in the application. And you don't want the user to select it every time when he opens your awesome weather application. This can be a good opportunity to use Shared Preferences to store user preferences in key-value pairs.

SQLite

SQLite is used to store more structured data locally in the mobile application, it is used to store structured data like your store in an RDBMS, it provides similar features to MySQL or any other SQL database but with limited functionality. Some features like stored procedures are not available. SQLite is very helpful when you would like to store complex data in your android application or a large amount of data that can be used by the application to provide a better user experience.

File Storage

In Android applications, we can choose to use the device storage space in order to store the data in it for the application we build. This type of storage is useful when the data is like an image, audio, video, or a file like that. These files don’t need to be searched internally and can be accessed in the app for storing large files. The application should be able to parse this data in order to use this while running though. Like you will need a video player specific to the format you’re storing the file in the device if it’s not supported by your Android device.

There can be two options here one is internal storage and the other is external storage, I recommend external storage as people have a large amount of external storage on their phones generally as they use an SD card in their devices.

Though an option can be provided in this case to the user if they would like to store the files in their internal phone storage or external storage.

Content Providers

Suppose you have an application which provides data to other applications on the device, you store about 5 rows in a database including:

email, name, phone number, profile photo and password

But to other applications, you would only like to expose a part of this information which is you can assume only the profile picture of the user.

In this scenario, content providers can help you out and you can use them in order to achieve this. Content Providers can help you with restricting the data access for all the data you have but providing access to the data you want other apps to access in your device.

Cloud Storage

You can use cloud storage and access it through various APIs provided by Microsoft, Google and other companies. This can help you back up your application data on the internet which can be restored in case you lost your device and got a new one. For example, WhatsApp allows you to store your device backup in google drive.

3.3 APIS

What is an API?

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau’s software system contains daily weather data. The weather app on your phone “talks” to this system via APIs and shows you daily weather updates on your phone.

API stands for Application Programming Interface. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.

API architecture is usually explained in terms of client and server. The application sending the request is called the client, and the application sending the response is called the server. So in the weather example, the bureau's weather database is the server, and the mobile app is the client.

How to use an API?

The steps to implement a new API include:

Obtaining an API key. This is done by creating a verified account with the API provider.

Set up an HTTP API client. This tool allows you to structure API requests easily using the API keys received.

If you don't have an API client, you can try to structure the request yourself in your browser by referring to the API documentation.

Once you are comfortable with the new API syntax, you can start using it in your code.

Need For Mobile Application Security

It is also important to develop an application that is safe and secured. Previously, cyber-security was preferred in computers, laptops, and other networking devices vulnerable to malware attacks by hackers. But in recent years, there has been a paradigm shift in the cyber-attacks from traditional computer software to mobile applications. The main reason is the increased usage of mobile devices in daily life. Therefore, it is essential to secure the applications to avoid any cyber-attack.

Mobile app security is indispensable to protect mobile devices from various adverse effects such as

Data Theft

Poor mobile app security could lead to losing sensitive data such as customer information, login credentials, financial details, etc. The hackers will use the vulnerabilities available in the applications to gain access to crucial information. This affects both the customer as well as the company. It is detrimental to a business as it loses the trust among the users.

Financial Loss

The data available in mobile banking applications can include financial details such as credit card and debit card numbers, CVV, etc. If a banking app is compromised, it becomes easy for the hackers to get hold of the entire mobile. The hackers can make a transaction on the customer's mobile without their knowledge. The lack of mobile app security also causes financial losses to the company through fines, compensation, restoration, etc.

Intellectual Property Heist

Intellectual property is also a property that incorporates human intelligence such as copyrights, patents, etc. All mobile applications have their base code on which the app is designed. This base code is intellectual property. Generally, the hackers try to abstract the base code of the successful apps to create their clones. Such cloning apps are created to trick the users into downloading the false app under the impression of the original app. These cloned apps can also be used to spread malware on mobile devices.

Reputation Damage

Security breaches in the mobile application can damage the company's reputation. Once the user data is exposed, the customer's confidence in the app company is ruined. This results in the loss of brand name. Every business works based on user trust. If the trust and reputation are lost, it isn't easy to rebuild them.

Causes of Security Threats in Mobile Apps

Exploiting The Application Platform

A mobile application platform is where all the applications are available for download. The examples are Google play store and Apple store. These platforms have certain guidelines for application security such as android intents, platform permissions, keychains, etc. The improper usage of the platform's operating system might result in data leakage during android communication.

Insecure Data Storage

Here, the security threat is caused by storing data in an unsafe manner. This vulnerability is mainly caused by the compromised operating system, jailbroken devices, improper handling of data cache and framework vulnerabilities. This leads to the hacking of legitimate apps to extract data from them. It occurs in various places such as binary data stores, cookie stores, SQL databases, etc.

Unsafe Communication

The data transfer in mobile applications takes place in a typical client-server model. Such data transfer is done over the internet and device carrier network. There is a possibility of hackers exploiting this vulnerability to access sensitive information. This cyber-attack can be carried out through an insecure Wi-Fi network, exploiting the network via routers, proxy servers or malware-infected apps. This leads to stealing of data, Man in The Middle (MITM) attack, etc.

Inappropriate Authentication

The improper implementation of the authentication process leads to security threats in the mobile app. Due to poor authentication, the hacker tries to bypass the regular identification process and thus access the information using a false identity. Mobile app authentication is more vulnerable when compared to the traditional web application as it does not require online authentication all the time. This offline loophole can be used by hackers to access the app.

Lack of Data Encryption

Cryptographic technology is used to safely transfer data by encryption and decryption methods. The poor data encryption method can compromise the integrity of the information. Hackers can exploit this vulnerability to interpret, steal, or tamper with the original data. It holds a high threat to the security of confidential data during transfer.

Weak Coding Practice

It is a solid threat caused due to the volatile coding practice. It is the result of dis-orientation among the members of the coding group where each person follows a different coding procedure. It causes inconsistency in the out of coding program of the app. However, it is difficult to identify the coding errors, which requires both automated and manual coding review. If the hacker identifies such a loophole, it could cause storage leaks; buffer overflows, compromised default libraries, and more.

Reverse Engineering

The reverse engineering threat occurs in Android apps where JavaScript codes are written. The coding pattern of the android apps can be exposed by applying reverse engineering tools and techniques. The online binary tools are used for this purpose. The threat is not the exposure of the base code but how it will be to replicate the original app to steal user information. It does not affect the app in a single device, but all the devices that use such an application.

Mobile App Security Best Practices

There are various mobile application security practices that can be implemented in app designing and maintenance. Some of them are as follow.

Open-Source Code Assessment

Most mobile applications use open-source code or third-party libraries which contain reusable source codes. Even though such codes make it easy to develop and deploy mobile applications, they are readily available to anyone, which poses a threat to the android apps that are using them. The method of reverse engineering can be used to crack the code easily.

Moreover, in the app development process, the open-source codes from the third-party libraries are merged with the written code, and they go unnoticed during app testing. So, it is mandatory to run a thorough test on the open codes for any vulnerabilities before adding them to the app code.

Solid Usage of Cryptography

The cryptography was originally used to transfer the data without revealing the message to the third party. This can be done by encryption and decryption of data during transfer. Now, this method is used in network communication for safe communication and storage of data. Using a strong data encryption technique, application data such as source code, user info, login credentials, app storage can be secured from hackers. Once the data is encrypted, even if the hacker steals the data, it becomes difficult to interpret the original content.

Using Code Signing Certificate

It is very important to code sign the mobile application to protect it from cyber-attack and gain the trust of the user. A code signing certificate is a digital certificate containing the CA's digital signature (Certificate issuing Authority) along with the developer's identity. It makes sure that the code has not been interpreted or altered after signing the app. There are many 'cheap code signing certificates' available, which can sign the mobile application. It is both cost-effective and provides high security to the application.

Enhancing the Data Cache

The cached data contains information retrieved from the app to help in faster reopening, thus improving the app's performance. This data cache is generally stored unsafely in the user device. It makes it easy for the hacker to interpret the cache data and steal sensitive information. So, it is always safe to create a password to lock the application, making it difficult to access the cache data. It is also a good practice to clear the cache data frequently and log in using a secured network connection.

Safe Data Storage

Each application contains lots of data that cybercriminals can exploit to do malicious activities. It includes both users' as well as the app developer's information. Therefore, it is essential to store the app data safely where both the app and the device are up to date. The developer always recommends not to store the app data in a local store. Secured cloud storage can be used for this purpose. It is also important to have a layer of protection in the app to safeguard private information. It is a best practice to separate app data from user data.

Authentication and Authorization Techniques

The authentication and authorization process forms the two strong pillars of mobile app security. Both are equally important to secure the application from cyber-attack. The authentication process ensures that the users provide required information such as login credentials to open and access the data in the app. It is essential to have multi-factor authentication to prevent data theft. It includes user id, password, PIN, OTP, etc.

It is also essential to give limited authorization according to the requirements. Giving high-level authorization such as administrator to the normal user could result in data theft and tampering with the entire app. The authorization should always take place on the server-side to verify the role and permissions of the authenticated users.

Data Wipe and Device Locking

This feature is mostly used in the application, containing confidential data such as personal, financial, health information, etc. It is a security layer where the remote data is wiped after several unsuccessful logins attempts from the user side, and the application is locked. It also mandates the users not to use a sequential number instead of capital letters, special characters, alphabets, numbers, etc.

The enterprises use three types of data wiping solutions: factory reset wipe, full device wipe, and enterprise device wipe.

Counteract Reverse Engineering

The method of reverse engineering used by the hackers to tamper with the function of the application causes a serious threat to mobile app security. By gaining access to the app’s source code, the hacker can circumvent the authentication process, fake the location, and steal the app data. Enforcing the run time security is paramount to counteract reverse engineering. This prevents the hacker from modifying the internal functions of the app by changing the code structures to affect the application behaviour.

4 Ways to Secure Your Mobile Apps

Securing your mobile application is essential in today’s digital age. Whether you’re developing a new app or already have one in use, take the necessary steps to ensure that your application is secure and protected against malicious attacks. To help you get started, here are the top 4 tips that can help you secure your mobile applications.

Data Security: Securing your mobile app ensures that your customer’s data remains safe and secure. Without proper security measures in place, malicious actors may be able to gain access to customer information or even use the app to attack other systems. Businesses must use data security measures such as encryption to maximize data security of both businesses and app users.

User Authentication: User authentication helps protect against unauthorized users attempting to access your application. This can be done by implementing strong passwords and two-factor authentication. By requiring app users to use 2-Factor authentication, businesses can prevent malicious actors from gaining access to sensitive data or performing other malicious activities.

Secure APIs: Most mobile apps rely on Application Programming Interfaces (APIs) for communication between the app and its backend services. These APIs should be properly secured to ensure that only authenticated users are allowed access. Implementing encryption, token authentication, and other security measures can help protect against potential threats from malicious actors.

Reduce Malware Threats: Malware can be a serious threat to your mobile app. By implementing proper security measures such as in-built sandboxes, antivirus software and intrusion detection systems, you can reduce the risk of malware entering your applications.

With the increasing use of mobile devices, mobile applications are becoming an integral part of our daily lives. However when it comes to mobile app security, many businesses throughout the world are still struggling to secure their business applications.

Unsecure mobile apps can be vulnerable to a wide range of malicious attacks, leaving users at risk of losing their personal information and data. That’s why as a business it’s essential to secure your mobile apps to protect your users and your business.

CHAPTER 4 Introduction to Android Development

Defination of Android studio, Android SDK, emulator

Local Storage SQLite

SQLite is a database engine written in the C programming language. It is not a standalone app rather, it is a library that software developers embed in their apps. As such, it belongs to the family of embedded databases. It is the most widely deployed database engine, as it is used by several of the top web browsers, operating systems, mobile phones, and other embedded systems.

SQLite was designed to allow the program to be operated without installing a database management system or requiring a database administrator. Unlike client–server database management systems, the SQLite engine has no standalone processes with which the application program communicates. Instead, a linker integrates the SQLite library statically or dynamically into an application program which uses SQLite's functionality through simple function calls, reducing latency in database operations.

Android SQLite is a very lightweight database which comes with Android OS. Android SQLite combines a clean SQL interface with a very small memory footprint and decent speed. For Android, SQLite is “baked into” the Android runtime, so every Android application can create its own SQLite databases.

Opening and Closing Android SQLite Database Connection

first open the database connection by calling getWritableDatabase() method as shown below:

```
public DBManager open() throws SQLException {

       dbHelper = new DatabaseHelper(context);

       database = dbHelper.getWritableDatabase();

       return this;

   }
```

To close a database connection the following method is invoked.

```
 public void close() {

       dbHelper.close();

   }
```

Inserting new Record into Android SQLite database table

```
public void insert(String name, String desc) {

       ContentValues contentValue = new ContentValues();

       contentValue.put(DatabaseHelper.SUBJECT, name);

       contentValue.put(DatabaseHelper.DESC, desc);

       database.insert(DatabaseHelper.TABLE_NAME, null, contentValue);

   }
```

Updating Record in Android SQLite database table

The following snippet shows how to update a single record.

```
public int update(long _id, String name, String desc) {

       ContentValues contentValues = new ContentValues();

       contentValues.put(DatabaseHelper.SUBJECT, name);

       contentValues.put(DatabaseHelper.DESC, desc);

       int i = database.update(DatabaseHelper.TABLE_NAME, contentValues, DatabaseHelper._ID + " = " +
_id, null);

       return i;

   }
```

Android SQLite - Deleting a Record

We just need to pass the id of the record to be deleted as shown below.

```
public void delete(long _id) {

       database.delete(DatabaseHelper.TABLE_NAME, DatabaseHelper._ID + "=" + _id, null);

   }
```

Realm is an open source object database management system, initially for mobile operating systems (Android/iOS) but also available for platforms such as Xamarin, React Native, and others, including desktop applications (Windows6). It is licensed under the Apache License.

Firebase is a set of backend cloud computing services and application development platforms provided by Google. It hosts databases, services, authentication, and integration for a variety of applications, including Android, iOS, JavaScript, Node.js, Java, Unity, PHP, and C++

Firebase provides detailed documentation and cross-platform app development SDKs, to help you build and ship apps for iOS, Android, the web, Flutter, Unity, and C++. Integrating it into your app is easy.

Firebase might be an excellent choice for a rapid prototype, a mobile app, or a project with real-time features.

A push notification is a short message that appears as a pop-up on your desktop browser, mobile home screen, or in your device notification center from a mobile app. Push notifications are typically opt-in alerts that display text and rich media, like images or buttons, which enable a user to take a specific action.

These small, pop-up messages are sent to users devices by a mobile app and can be viewed from the device lock screen when an app isn't currently in use. Unlike other communication channels like email, push notifications are designed to be viewed in real-time and often trigger immediate engagement. They can be used to convey reminders, updates, promotions, and more.

CHAPTER 5

Testing and Debugging Mobile Application

5.1 Tools and techniques for testing and debugging mobile applications:

-Crash Reporting Tools

One of the most useful tools for debugging mobile applications is a crash reporting tool. A crash reporting tool is a service that collects and analyzes crash data from your mobile application and sends you detailed reports on the frequency, severity, and location of the crashes. Crash reporting tools can help you identify the most common and critical bugs in your mobile application, as well as the devices, operating systems, and app versions that are affected by them. Some examples of crash reporting tools are Firebase Crashlytics, Sentry, and Bugsnag.

-Debugging Tools

A debugging tool is a software that allows you to run your mobile application on a simulator or a real device and inspect its behavior, state, and performance. Debugging tools can help you set breakpoints, step through code, watch variables, evaluate expressions, and monitor network traffic. Some examples of debugging tools are Xcode for iOS, Android Studio for Android, and Visual Studio Code for cross-platform development.

- Testing Tools

A testing tool is a software that enables you to automate the testing of your mobile application and verify its functionality, usability, and quality. Testing tools can help you simulate user interactions, check user interface elements, measure performance, and generate test reports. Some examples of testing tools are Espresso for Android, XCUITest for iOS, and Appium for cross-platform development.

-Espresso Testing Framework is an open-source framework created by Google for Android that allows QA to write tests on the user interface. It has a simple API, which easily adapts to the testing needs, and eliminates the complexity of managing different threads.

-XCUITest is a test automation framework used for UI testing of mobile apps and web applications on iOS devices such as iPads and iPhones. It is part of Apple's testing framework. XCUITest was released in 2015 as part of the broader XCTest system built into XCode, Apple's IDE

-Appium is a mobile automation testing tool which is used to automate mobile applications on different mobile operating systems such as Android and iOS. It is an open source tool which can be used to automate the testing of different mobile device applications such as native, WAP and hybrid application.

- Logging Tools

A logging tool is a software that records and displays the events and messages that occur in your mobile application during its execution. Logging tools can help you trace the flow of your code, identify errors and warnings, and debug complex issues. Some examples of logging tools are Logcat for Android, NSLog for iOS, and Console for cross-platform development.

-Profiling Tools.

A profiling tool is a software that measures and analyzes the performance and resource consumption of your mobile application. Profiling tools can help you identify bottlenecks, memory leaks, CPU usage, battery drain, and network latency. Some examples of profiling tools are Android Profiler, Instruments for iOS, and Chrome DevTools for cross-platform development.

-Feedback tools

A feedback tool is a software that enables you to collect and manage user feedback on your mobile application. Feedback tools can help you understand user needs, preferences, and pain points, as well as discover bugs and issues that users encounter. Some examples of feedback tools are Instabug, UserTesting, and Appcues.

5.2 Unit Testing in Mobile app development

Unit testing is the process of testing small isolated portions of a software application called units. Unit testing is a method of testing small pieces of a software application without relying on a third-party system. Any component that interacts with an external database, files, or the network cannot be checked as part of unit testing. Unit tests primarily test isolated components during the product's early development phase.

During an application's development phase, developers typically perform unit testing. When a piece of code is written to carry out a specific function, there must be a means to check and prove that the implementation is correct. Hence, tests are required.

Unit Testing Techniques

The process of unit testing can be carried out with the help of three main testing techniques:

Structural Technique

Structural testing is a sort of testing that examines the structure of a program. It's often referred to as White Box or Glass Box testing. Because this type of testing necessitates a thorough understanding of the code, developers typically perform it. It concerns more about how the system accomplishes it than its functioning. It expands the scope of the tests.

Functional Testing Technique

Functional testing is a sort of testing that aims to determine whether each application feature functions in accordance with the program requirements. The result of each function is compared to the relevant requirement to see if it meets the end user's expectations. The testing is carried out by supplying sample inputs, recording the resulting outputs, and ensuring that the actual outputs match the expected outputs.

Error based Technique

The person who designed the code must be involved in the process as they are the ones who know the code end-to-end. The following are a few examples of error-based techniques:

Historical Test Data: This technique uses historical data from previous executions of the test case to determine the priority of each test case.

Mutation Testing: This involves changing some statements in your source code and seeing if your test code can discover the mistakes.

Fault seeding techniques: This can be used to introduce known problems into the code and test them until all of them are identified.

5.3 Alpha Testing and Beta testing in mobile applications

Alpha testing is an important step in software testing. This process aims to identify and eliminate errors or bugs before the product is released. An internal QA team performs this testing in a controlled lab environment to ensure software quality before it goes into production. Alpha testing employs white and black box testing, which checks the system's internal structure or design and ensures input and output functionality. This testing is required for identifying and fixing any issues before the product is released, ensuring the end-users have a seamless experience.

Alpha testing is performed after the product is tested and ready for release. It is performed by people who are a part of the organization, like software engineers or quality assurance professionals.

Beta testing is a user acceptance test where a group of users evaluates a nearly finished application to see how well it functions in a real-world setting. This application development stage seeks to present your application to a group of actual users and collect input for enhancement. The application has completed beta testing and is almost ready for public release.

Customer validation is used in beta testing to reduce the likelihood that the application will fail. Although beta testing can be expensive and time-consuming, it is an excellent approach to fixing technical problems with your new product.

The advantage of the beta testing app is that we can understand user perspectives before making the product available to the public. Furthermore, you receive incredibly accurate performance findings because this testing is conducted outside a controlled testing environment.

Chapter 6: Deployment of mobile Application

6.1 Understanding Bundling of Mobile application.

An Android App Bundle is a publishing format that includes all your app's compiled code and resources, and defers APK generation and signing to Google Play.

Google Play uses your app bundle to generate and serve optimized APKs for each device configuration, so only the code and resources that are needed for a specific device are downloaded to run your app. You no longer have to build, sign, and manage multiple APKs to optimize support for different devices, and users get smaller, more-optimized downloads.

Advantages of Android App Bundling

- Reduces download size and disk allocation size when users are downloading apps.

- Reduce download size, disk allocation size, and installation time using a default uncompressed library (higher than Android 6.0) stored in APK instead of the user's device .

- Features and settings are provided when requested by users instead of when the app is on its installation process.

- Enables efficient building and launching management as it does not require building and publishing APKs.

6.2 Introduction to signing of mobile application

Application signing allows developers to identify the author of the application and to update their application without creating complicated interfaces and permissions. Every application that is run on the Android platform must be signed by the developer. Applications that attempt to install without being signed will be rejected by either Google Play or the package installer on the Android device.

Android requires that all apps be digitally signed with a certificate before they can be installed. Android uses this certificate to identify the author of an app, and the certificate does not need to be signed by a certificate authority. Android apps often use self-signed certificates. The app developer holds the certificate's private key.

The basics behind protecting your Android app is to use a generated certificate and digital "key" which provides a unique, encrypted, and reasonably un-hackable signature. This proves that the app came from you, not some other suspicious source.

6.3 Understanding the importance of app store guidelines and policies.

App store guideline are the way to provide a safe experience for users to get apps and a great opportunity for all developers to be successful. This is done by offering a highly curated App Store where every app is reviewed by experts and an editorial team.

Some of the app store guidelines are listed below

1.Before You Submit Make sure you:

-Test your app for crashes and bugs

-Ensure that all app information and metadata is complete and accurate

-Update your contact information in case App Review needs to reach you

2. Safety

When people install an app from the App Store, they want to feel confident that it's safe to do so—that the app doesn't contain upsetting or offensive content, won't damage their device, and isn't likely to cause physical harm from its use.

3. Performance

Your app must be a finished product. It must be free of any bugs and errors. Your app must pass all the testing required. So that, the user get the best performance that your app have to offer.

4. Business

There are many ways to monetize your app on the App Store.

****

6.4 Understanding the process of deployment

Application Deployment is one of the most important stages of the software development process as the strategy that is used to build, test, and deploy will directly impact how fast an application can respond to changes in constituent preferences or requirements, and most importantly—the quality of each change.

What is the Application Deployment Process?

The Application Deployment Process involves nine main steps:

1. Plan. This stage is what gets development and operations teams on the same page. This is where the deployment schedule is mapped out, the current infrastructure is assessed, and changes are made if necessary.

2. Build and Release Automation. Enabling automation is key to a successful app deployment process, as the possibility for human error needs to be decreased as much as possible. Developing script and hearty servers will help ease deployment across the network later.

3. Develop Continuous Integration / Continuous Delivery (CI/CD). Working to reduce the degree of change in each application update will help teams spot breakdowns in the future. When deployments are minimally impactful, your system will be able to carry out more frequent ones easier.

4.Create and Test Scripts. Identify environmental changes and differences by running test scripts on a backup copy of production before moving to the final release.

5.Identify Key Metrics. Make sure your team is clear on key performance indicators (KPIs) from one application to another. This step is simple: make sure KPIs are set, visibility is enabled, and any potential errors in the application set are troubleshot swiftly.

6.Test. Set up synthetic transaction tests, and make sure key items like log-in pages are working correctly. Go into your deployment with confidence.

7.Develop Deployment Tracking. Enable and implement tracking services to make sure ops teams can easily track when deployments occur (or when they’re scheduled to), and to immediately identify when errors occur and how to fix them.

8.Alert Users and Colleagues. This is an often-overlooked step, but when it applies, alert necessary constituents as to when an application is expected to deploy. This will help coordinate throughout the process, set expectations, and backtrack if there were any errors.

9.Monitor and Iterate. Once the application is deployed, it will be as equally as important to monitor deployment and correct as needed.

Chapter 7 Introduction to IOS development

Xcode is Apple's integrated development environment (IDE) for macOS, used to develop software for macOS, iOS, iPadOS etc. It was initially released in late 2003. It is available free of charge via the Mac App Store and the Apple Developer website.

Xcode supports source code for the programming languages: Swift, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez), and C, with a variety of programming models, including but not limited to Cocoa, Carbon, and Java.

Xcode includes the GUI tool Instruments, which runs atop a dynamic tracing framework, DTrace, created by Sun Microsystems and released as part of OpenSolaris.

What is Swift in iOS?

Swift is a robust and intuitive programming language created by Apple for building apps for iOS, Mac, Apple TV and Apple Watch. It's designed to give developers more freedom than ever. Swift is easy to use and open source

Swift is a general purpose programming language that employs modern programming-language theory concepts and strives to present a simple, yet powerful syntax. Swift incorporates innovations and conventions from various programming languages, with notable inspiration from Objective-C, which it replaced as the primary development language on Apple Platforms.

Swift was designed to be safe and friendly to new programmers while not sacrificing speed. By default Swift manages all memory automatically and ensures variables are always initialized before use.

What is an iOS emulator?

iOS app simulators mimic applications that run on an iOS device. These tools allow you to start and run iOS apps without an iPhone or iPad—accessing the app from your web browser, Mac, or Windows computer instead

Features of Xcode

Some of the key features and capabilities of Xcode are:

Integrated Development Environment (IDE): Xcode is a complete IDE that provides developers with a unified workflow for designing user interfaces, coding, testing, and debugging applications for Mac, iPhone, iPad, Apple TV and Apple Watch.

Programming Languages: Xcode includes the Swift programming language, which is safe, fast, and modern. It also supports C/C++/Objective-C compilers for legacy code or other use cases.

Simulator: This allows developers to test and prototype their apps in a simulated environment when a real device is not available. Simulator provides environments for iPhone, iPad, Apple Watch, and Apple TV devices with different settings, files, and operating system versions.

Instruments: This tool is used to profile and analyze apps, improve performance, and find memory problems. Instruments collects data and presents the results using different tools.

Create ML: Developers can create and train custom machine learning models for their app using this tool.

Reality Composer: This tool allows developers to construct 3D compositions and augmented reality (AR) experiences.

Build System: Developers can port their macOS apps to Apple Silicon and create a version of their macOS app that runs on both Apple silicon and Intel-based Mac computers. Developers can compile their code into a binary format and customize their project settings to build their code.

Swift Packages: Developers can create reusable code, organize it in a lightweight way, and share it across Xcode projects and with other developers.

User Interface Design: Xcode offers several tools to help developers design intuitive user interfaces, such as SwiftUI, a declarative framework that uses Swift code to describe interfaces, and Interface Builder, which builds UI views graphically using a library of controls and modifiers.

SwiftUI Design ToolsSwiftUI Design Tools

Debugging: Xcode allows developers to inspect their running app to isolate bugs, locate crashes, identify excess system resource usage, visualize memory bugs, and investigate view layout problems. They can set breakpoints to pause their running app and find the cause of bugs by watching variables change as they step through their source code in the debugger.

Analyze Traffic with Instruments: Xcode allows developers to measure HTTP-based network performance and usage of their apps.

Accessibility: Xcode supports designing and developing applications with accessibility features such as VoiceOver, AssistiveTouch, and Switch Control, enabling a wider range of users to enjoy your applications.

Diagnosing Issues with Entitlements: Developers can verify their app’s entitlements at every stage of development to track down errors during distribution.

Diagnosing memory, thread, and crash issues early: Developers can identify runtime crashes and undefined behaviors in their app during testing using Xcode’s sanitizer tools.

Diagnosing issues using crash reports and device logs: Developers can use crash reports and device logs to diagnose issues with their app.

Xcode Cloud: It is a new service from Apple that is currently in beta testing and is expected to be released later in 2023. It is a cloud-based continuous integration and delivery service that will integrate with Xcode, allowing developers to build, test, and deploy their applications in the cloud.