Contents

UNIT 4 - Android Tasks, Services, and Testing

CHAPTER 10 –Testing Android Applications

Structure of Android Tasks and Services

10.1 Learning Outcomes

After the successful completion of this chapter, the student will be able to:

- Describe the levels/types of testing used in android application development
- Differentiate between unit, integration, system, and acceptance testing
- Describe various tools used to test an android application

10.2 Testing Android Apps

Any software development is incomplete without testing. In a software application development process, testing is one of its phases. Any software application developed is tested before it is handed over to the end user for ensuring that the application meets the desired quality standards and the requirements of the end users. Testing is done to find out if any error(s)/defects exist. The errors and defects identified are fixed and checked to deliver an application of quality to the users with proper/correct functionality. Testing ensures that the software is reliable, efficient, and user-friendly. It also helps to reduce the cost and time involved in fixing errors and defects that are found after the application has been released to the end users. To assure and maintain the quality of the software, the application is tested at various levels as shown in figure 10.1.

10.2.1 Levels of Testing

The different levels of testing are

- Unit Testing
- Integration Testing
- System Testing
- Acceptance Testing

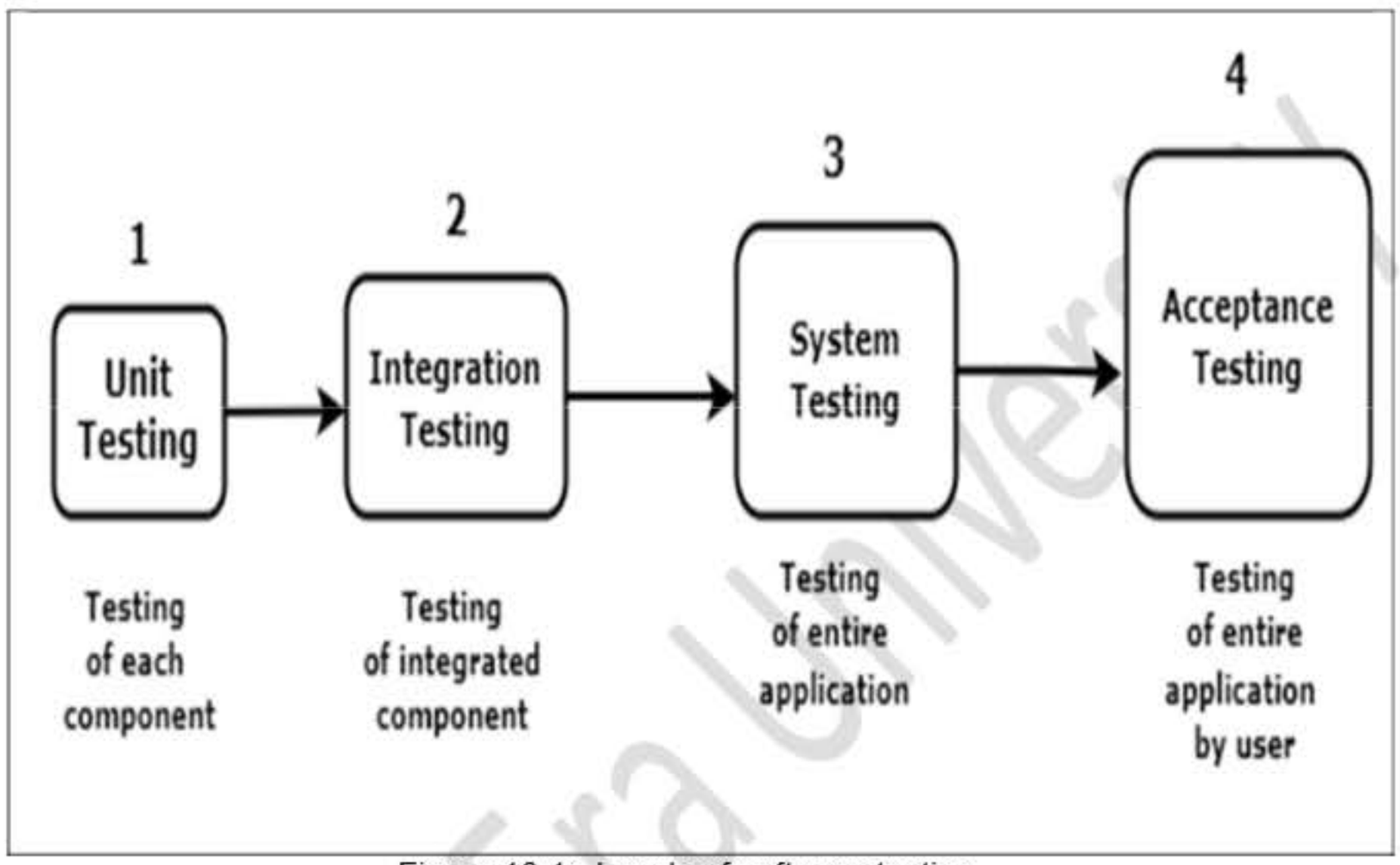

Figure 10.1: Levels of software testing

Unit Testing - Testing of small unit/program code is called unit testing. It is done at the class or method level to ensure that they are working as expected.

In the case of Android applications for e.g. consider a shopping application, unit testing can be done using the following test cases.

For e.g. Consider a login Activity that allows the user to choose either of the radio buttons

- **Create a New account**
- **Sign In**

Test Case 1: On Click of the 1st radio button **Create a New account,** and a registration form for new customers should appear as shown in figure 10.3. The **onClick()** event is tested to check whether the event handler code is calling the next appropriate activity as required. In this case, whether the Registration form for a New customer gets displayed.

The following code snippet shows the eventListener added to the radio button and on click, an Intent object is created and the next activity is called.

```
 // For new customer
RBnewlogin.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {   // Unit testing
// creating Intent object
Intent intent=new Intent(LoginActivity.this,LoginNewActivity.class);
startActivity(intent);    // starting activity for new customer
 }
});
```

Figure 10.2:Snippet of onClick() for unit testing

[Click of radio button for the new customer]

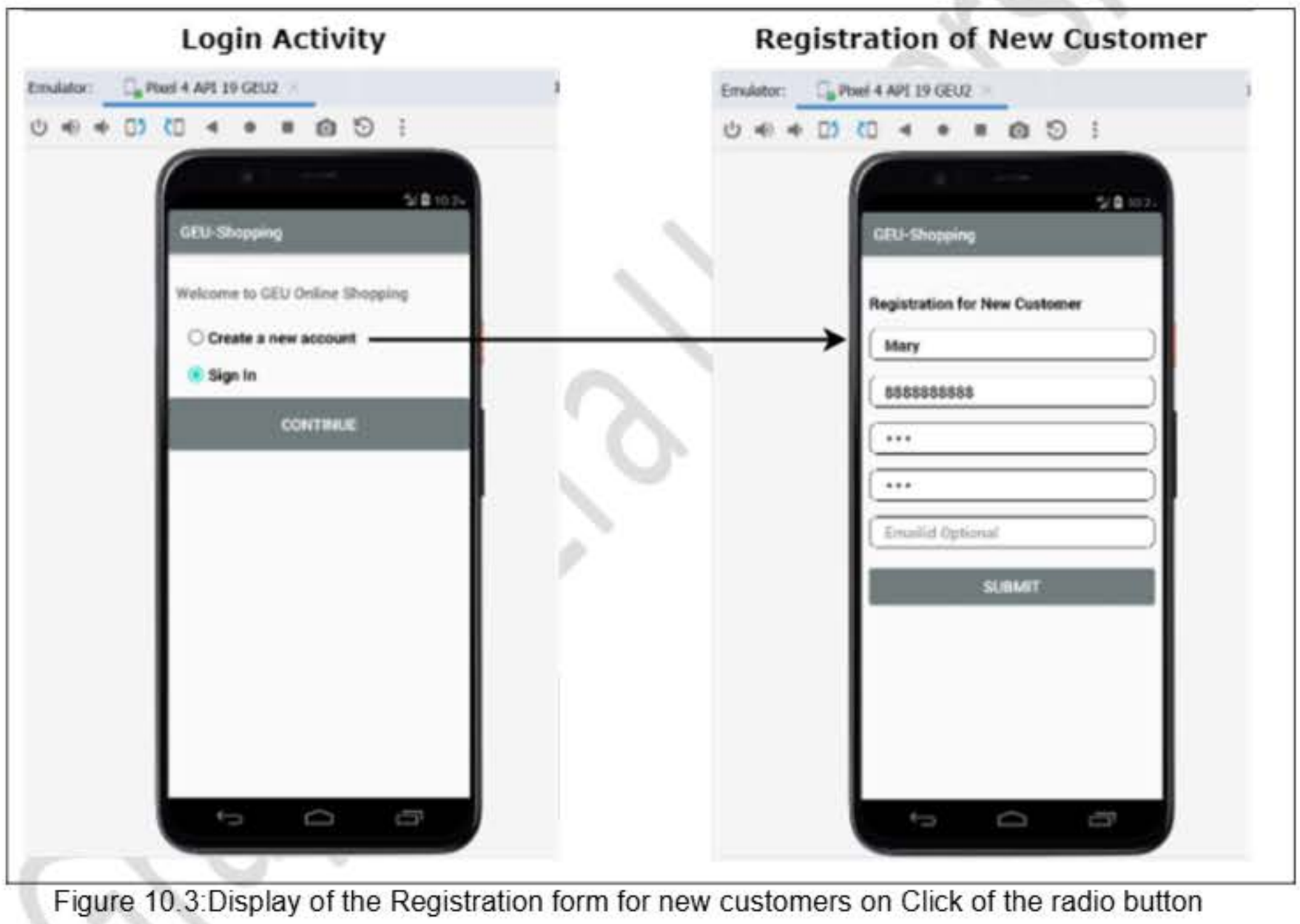

Figure 10.3:Display of the Registration form for new customers on Click of the radio button

Test Case 2: On Click of the 2nd radio button **Sign In,** a registration form for the registered customer should appear as shown in figure 10.5. The **onClick()** event is tested to check whether the eventhandler code is calling the next appropriate activity as required. In this case, whether the Registration form for a Registered customer gets displayed.

The following code snippet shows the eventListener added to the radio button and onClick, an Intent object is created and the next activity is called.

```
    // For registered customer
RBregdusr.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {    // Unit testing
   // creating Intent object
   Intent intent=new Intent(LoginActivity.this,LoginRegdActivity.class);
   startActivity(intent);   // starting activity for registered customer
   }
});
```

Figure 10.4:Snippet of onClick() for unit testing

[Click of radio button for the registered customer]

Figure 10.5:Display of the Registration form for registered customers on Click of the radio button

Test Case 3: Testing a Login process

Unit testing involves creating a mock user and verifying whether the user credentials are validated by the login method.

Test Case 4: Testing of the total price of the items added to the cart.

Unit testing involves creating a mock user, adding items to the cart, and verifying whether the total price of the items added to the cart is calculated correctly.

Integration Testing: After testing the small modules, these modules are integrated and then tested to check whether the code gets executed properly.

Test Case 1: For e.g. on the Click of 1st radio button **Create a New account,** a registration form for the new customer should appear, and on Click of 2nd radio button **Sign In,** a registration form for the registered customer should appear as shown in figure 10.6.

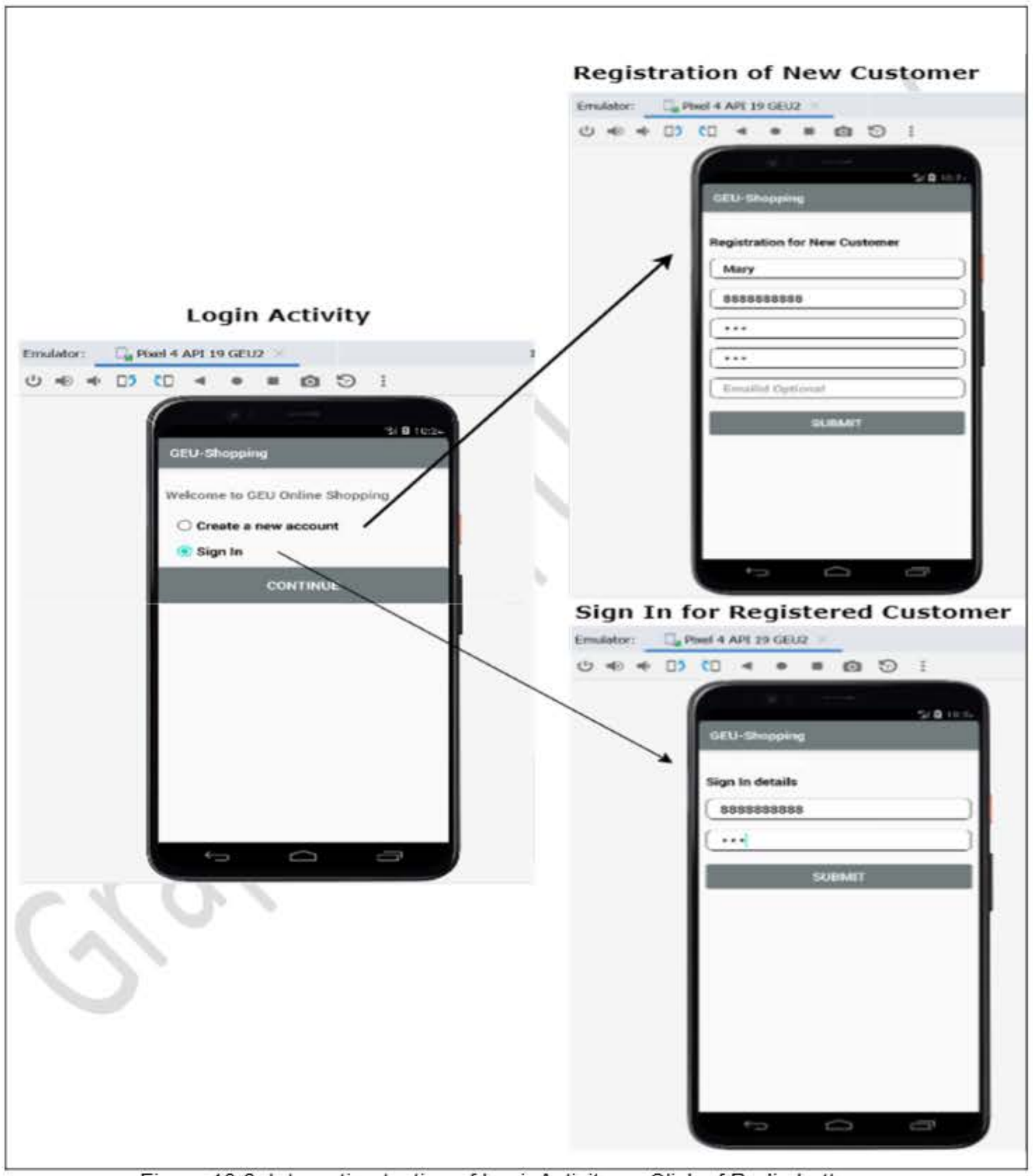

Figure 10.6: Integration testing of LoginActivity on Click of Radio button

The eventListener and onClick() event handlers for the 1st, as well as 2nd radio buttons, are tested and checked whether the appropriate activities get displayed.

```
// For new customer
RBnewlogin.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {   // Unit testing
// creating Intent object
Intent intent=new
                Intent(LoginActivity.this,LoginNewActivity.class);
startActivity(intent);     // starting another activity
}
});

// For registered customer
RBregdusr.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {    // Unit testing
// creating Intent object
Intent intent=new
          Intent(LoginActivity.this,LoginRegdActivity.class);
 startActivity(intent);   // starting another activity
 }
});
}
```

Figure 10.7:Event handler and EventListener in LoginActivity on Click of Radio buttons

Test Case 2:

Integration testing involves checking whether the functionality is working properly after the integration of modules like connecting to the database, adding data to the database, retrieving the data from the database, and displaying it to the user as per the query.

Overall integration testing is done to test the interaction between the different components or services of a software application. By testing the integration between these components, developers/testers can catch bugs that might not have been caught by unit testing alone and ensure that the system as a whole is functioning correctly.

System Testing: The entire software application is tested to ensure that all the required features incorporated in the application are working/responsive and meet the specifications of the end users. System testing is done after integration testing. System testing is done by the developers/testers before publishing the application in the market. This helps to minimize and fix the bugs and issues arising after the application has been released and ensures that the software is of high quality, reliable, and meets the user's needs. It also helps to improve the user experience, increase customer satisfaction and maintain the software's reputation in the market.

For example, considering that the Android Online Shopping application is to be tested. A few tests that can be performed are:

User interface testing – Check whether all buttons, action bar icons, image clicks, scrolling, links, etc. are working properly to ensure that the user can navigate with ease.

Functional testing – Check whether the selected items can be added/removed to/from the cart and whether the bill amount is displayed correctly.

A new customer is able to register by providing the details and later login/SignIn using the login credentials.

Acceptance Testing: This is the last level of software testing done by the end user before accepting the end product/android application. Testing is usually done by two or more end users to check whether all the specifications given at the time of design have been met. The software is tested by using real-world scenarios and provides feedback on any issues or areas for improvement.

However, acceptance testing is not the last level of testing in the software development life cycle. After user acceptance testing, there may still be additional levels of testing such as regression testing, performance testing, and security testing depending on the complexity and requirements of the software.

For example, considering that the Android Online Shopping application is to be tested. A few tests that can be performed by the users apart from the tests carried out during system testing are:

User Interface testing - Check whether the product details like name, image, price, etc are displayed correctly.

Functional Testing – Check whether the product image can be zoomed, the quantity of the product can be increased/decreased, and the bill amount gets updated accordingly.

An admin user is able to store the product images in the database and display them in the view correctly.

10.2.2 Types of Testing

There are various other types of testing done based on the environments and the type of android application developed before releasing to the users.

Functional Testing - The application is tested to check whether it meets the user's requirements. Functional testing involves testing each function or feature of the application individually, as well as testing the application as a whole. It can be performed manually or through automated testing tools and it can be executed at different stages of the software development life cycle, including unit testing, integration testing, and system testing.

User-Interface Testing - The application is tested on various devices for proper alignment of images, text visibility, scrolling of information wherever applicable, size of elements, etc. User interface testing focuses on the usability and accessibility of an application's interface. This testing ensures that the application's interface is visually appealing, easy to use, and accessible to all users.

Regression Testing - The application is retested after making some modifications/updation to the code, adding new features, and fixing bugs to check whether the new functionality as well as the previous functionality is working properly/correctly, and no new issues or bugs have been introduced.

The main objective of regression testing is to verify that the changes made to the software have not had any adverse impact on the previously working features and that the software still meets the desired specifications and requirements. It ensures that the software continues to function as intended.

For example: Suppose the online shopping application has been developed and released to the market with the payment feature as cash on delivery. However, the development team then decides to add some new features such as adding a new payment method and improving search functionality.

So before releasing the new version, the development team needs to perform regression testing to ensure that the existing functionalities of the application are still working correctly after the changes have been made. For example, the previous features like the login process, adding items to the cart, cash on delivery, etc. are functioning correctly along with the new features which are added.

Performance Testing - The primary goal of performance testing is to evaluate the system's ability to handle a specific workload or user traffic when a large number of users are using the application simultaneously.

For e.g. If the application under test is using a database, then it is verified whether data is retrieved quickly and efficiently without any issues. The performance tester can measure the

number of requests per second that the system can handle and ensure that it meets the desired performance criteria.

In performance testing, various metrics are tested, including response time, throughput, resource utilization, and scalability. These metrics help to determine the system's efficiency, stability, and scalability.

Compatibility Testing – The application to be tested is run using various devices and API levels to check for the proper functioning of the application on all the devices used.

In compatibility testing, an application is tested on a variety of devices, operating systems, web browsers, and other software and hardware configurations to verify that it functions correctly and consistently across different platforms.

For example, if an application is designed to work on an Android device, compatibility testing would involve testing the application on various devices with different screen sizes, resolutions, and versions of the Android operating system, including different API levels. The compatibility tester would ensure that the application works correctly on all devices and does not crash or produce unexpected behavior on any particular device.

Interruption Testing – The application is tested for its functionality in an interrupted state.

The goal of interrupted testing is to ensure that the system can handle and recover from various interruptions such as network failures, power failures or low battery issues, and other unexpected events like incoming calls or SMS that may occur during the normal operation of the application.

For e.g. The execution of an android app is tested for its proper and efficient working when the user is using the android app and is being interrupted by an incoming call or SMS, in this case, the app runs in the background and is later resumed to its state before the interruption. The interruption could distract the user from completing tasks like registration of a new customer, logging in process, adding an item to the cart, placing an order, etc. Therefore during interrupted testing the tester might simulate such interruptions to ensure that the application can handle them and recover from them appropriately under various conditions

Installation Testing – The application is checked for smooth installation without any errors during installation.

For example, installation testing for online shopping applications on Android devices would involve the application can be installed properly on different Android versions such as Android 9, Android 10, and Android 11. It should also be tested on different Android devices with varying screen sizes and resolutions to ensure it can adapt to different environments,

10.3 Android Testing Frameworks

Many Android Testing Frameworks are available in the market that can be used for testing android applications. A few are listed here

- **Junit** – It provides an instrumental test runner which can be used to run tests on Android devices or emulators and generate reports on the performance and behavior of the application. It generates XML reports which can be used to analyze the results of test runs and identify any issues or errors in the code.
- **Appium** – It is an open-source framework that supports automated testing on real devices and emulators. It works for both Android OS and IOS. It supports many languages like JavaScript, Java, PHP, Ruby, Python, etc.
- **Detox** – It is a powerful mobile testing framework that can be used to test Android applications. It is JavaScript mobile testing framework used for testing an android application.
- **TestProject -** It is a free test automation platform that can be used for testing mobile applications. It is integrated with the browser and allows execution on any remote system which are registered with the platform. It is a cloud-based platform that provides a range of testing capabilities including test recording, test execution, and result analysis. This makes it easy for teams to collaborate and share test results, even if they are located in different geographic locations. TestProject also provides a range of pre-built test automation plugins, which can be used to speed up the testing process and improve the accuracy and reliability of test results.
- **UI Automator -** It is a GUI tool used for testing a user interface of an android application. It is used to inspect the hierarchy of the layouts and also the properties of the user-interface elements used.

10.4 Self-Assessment Questions

Q1. Describe the levels of testing and android application in brief. [8 marks, L2]

Q2. Define Unit testing. Explain unit testing with a suitable example.[6 marks, L2]

Q3. Define Integration testing. Explain integration testing with a suitable example. [6 marks, L2]

Q4. Explain the types of testing used for testing an android application.[6 marks, L2]

Q5. Explain the tools available for testing an android application.[5 marks, L2]

Q6. Why is testing done? It is necessary to test an application before release. Justify your answer. [8 marks, L2]

10.5 Self-Assessment Activities

Note: Use an appropriate emulator for the execution of an android application.

A1. Various android applications have been created as part of the activities in the previous chapters. Using any one of them discuss and demonstrate various types of testing that have been used in the application with suitable test cases.
[Unit testing, Integration testing, System testing, Functional testing, etc.]

10.6 Multiple-Choice Questions

1. The different levels of testing an android application are ____. [1 mark, L1]
   a) Unit testing
   b) Integration testing
   c) System testing and Acceptance testing
   d) All of the above

2. Testing of a small/single unit is known as ____. [1 mark, L1]
   a) Unit testing
   b) Integration testing
   c) System testing
   d) Functional Testing

3. Testing of more than one unit together is known as ____. [1 mark, L1]
   a) Unit testing
   b) Integration testing
   c) System testing
   d) Functional Testing

4. Unit testing is done by ____. [1 mark, L1]
   a) Users
   b) Developers
   c) Designers
   d) None of the above

5. System Testing is done by ____. [1 mark, L1]
   a) Developers/Testers

b) Users
c) Designers
d) None of the above

6. Acceptance testing is done by _____. [1 mark, L1]
a) Developers
b) Designers
c) Testers
d) End Users

7. Functional testing is done to check whether _____. [1 mark, L1]
a) it meets the user's requirements
b) it meets the developer's requirement
c) it meets the tester's requirement
d) all of the above

8. The application is run using various devices and API levels in ____. [1 mark, L1]
a) Unit testing
b) Integration testing
c) Compatibility testing
d) None of the above

9. _____ is not an automated tool used for testing an android application.[1 mark, L1]
a) JUnit
b) TestProject
c) Appium
d) Ruby

10. Regression testing is done when ____. [1 mark, L1]
a) The software application is modified/updated
b) The software application is not modified/updated
c) The network issues exist
d) None of the above

10.7 Key Answers to multiple-choice questions

1. The different levels of testing an android application are **Unit testing, Integration testing, System testing, and Acceptance testing.** [d]
2. Testing of a small/single unit is known as **Unit testing**.[a]
3. Testing of more than one unit together is known as **Integration testing**.[b]
4. Unit testing is done by **Developers**.[b]
5. System Testing is done by **Developers/Testers**.[a]
6. Acceptance testing is done by **end users**.[d]
7. Functional testing is done to check whether **it meets the user's requirements**.[a]
8. The application is run using various devices and API levels in **Compatibility testing**.[c]
9. **Ruby** is not an automated tool used for testing an android application.[d]
10. Regression testing is done when **the software application is modified/updated**.[a]

10.8 Summary

Any software development is incomplete without testing. In a software application development process, testing is one of its phases. Any software application developed is tested before it is handed over to the end user. Testing is done to find out if any error(s) exists. The errors identified are fixed and checked to deliver an application of quality to the users. To assure and maintain the quality of the software, the application is tested at various levels. The different levels of testing are - Unit Testing, Integration Testing, System Testing, and Acceptance Testing.

Unit Testing - Testing of small unit/program code is called unit testing.

Integration Testing: After testing the small modules, these modules are integrated and then tested to check whether the code gets executed properly.

System Testing: The full software application as a whole is checked to ensure that all the required features incorporated in the application are working/responsive and meet the specifications of the end users.

Acceptance Testing: This is the last level of software testing done by the end-user before accepting the end product/android application.

There are various other types of testing done based on the environments and the type of android application developed before releasing to the users. They are **- Functional Testing, User-Interface Testing, Regression Testing, Performance Testing**, **Compatibility Testing**, **Interruption Testing, Installation Testing, etc.**

Many Android Testing Frameworks are available in the market that can be used for testing android applications. Some of them are **Junit, Appium, Detox, TestProject, and UI Automator, etc**.

10.9 Keywords

- Unit testing
- Integration testing
- System testing
- Acceptance testing
- Regression testing

10.10 Recommended Resources for Further Reading

Text Book(s)

1. Pradeep Kothari, "*Android Application Development*" Black-Book, Dreamtech Press,2015
2. Wei-Meng Lee, "*Beginning Android Application Development*", Wiley 2011.
3. Dawn Griffiths and David Griffiths, "*Head First Android Development*", O'Reilly, 2017.

References

- https://www.javatpoint.com/android-tutorial
- https://www.tutorialspoint.com/android/index.htm
- https://developer.android.com/guide/components/fundamentals

--*--

Contents

UNIT 5 - Publishing Android Applications

CHAPTER 11 – Publishing Android Applications

Structure of Publishing Android Applications

11.1 Learning Outcomes

After the successful completion of this chapter, the student will be able to:

- Describe the phases of publishing an android application
- Describe versioning used in preparing an android application for release
- Generate a self-signed certificate and digitally sign an android application for release
- Explain and release an android application to the users

11.2 Publishing Android Application

Once an android application is developed and tested, it can be made available to the users. The final phase of android application development is Publishing as shown in figure 11.1. Users can use the application on their devices only after publishing. There are various ways by which the app can be released/made available to users. To release an application to the users, it is necessary to create a .apk file and distribute it to them so that the user can install and run the application on an android device. The Source code (.java file) written in JAVA is converted by the JAVA compiler (javac) into a .class file. The Dex compiler then converts the generated .class file into a .dex file which in turn is later converted into a .apk file by the Android Asset Packaging Tool (AAPT).

If the application is to be distributed on the google play store, the signing process has to be completed. Signing means adding additional private key information to the .apk file.

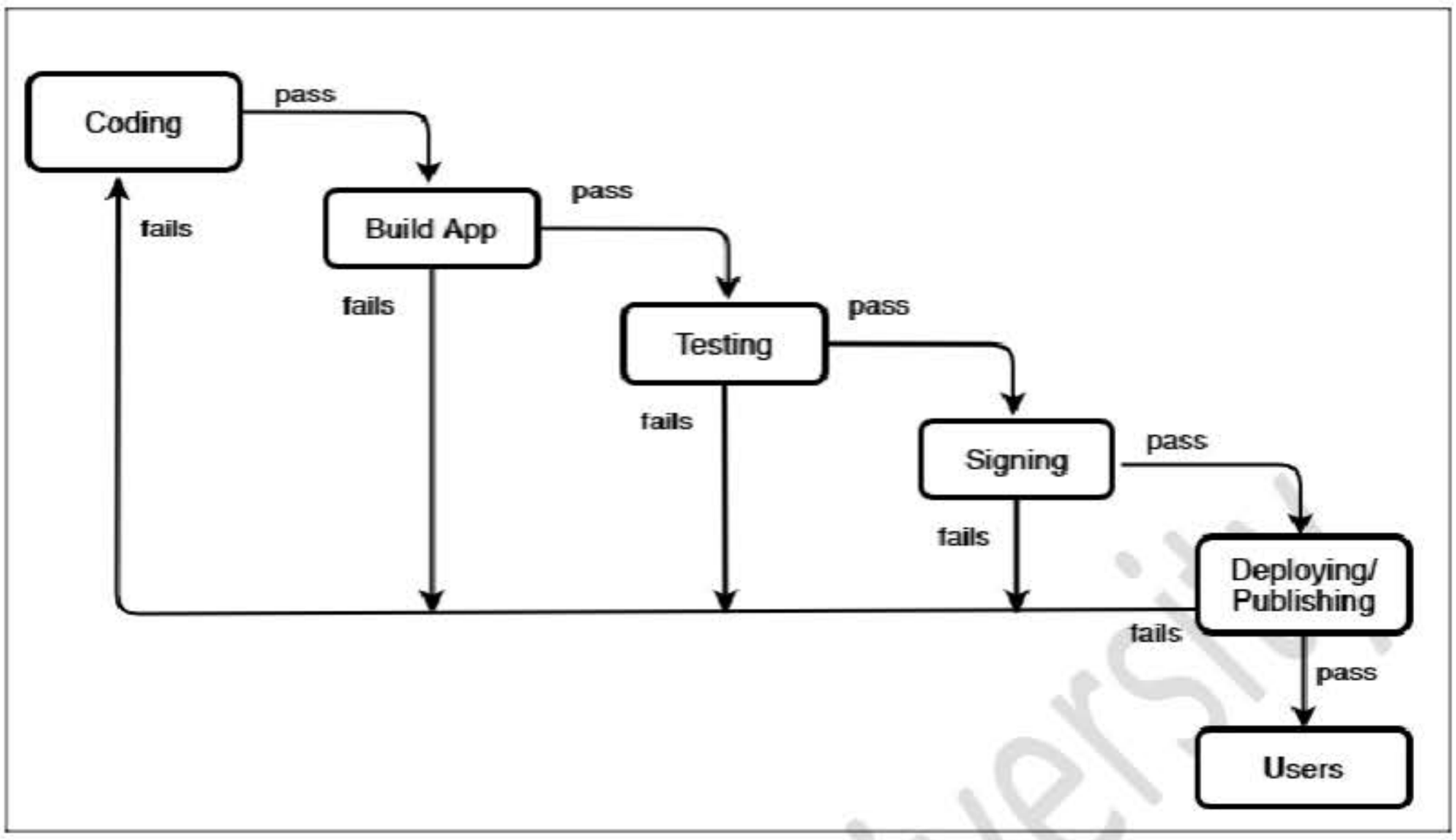

Figure 11.1: Android application development and deployment phases

Process of Publishing an android application

The process of publishing consists of 2 phases :

1. Preparing the application for release
2. Releasing the application to the users

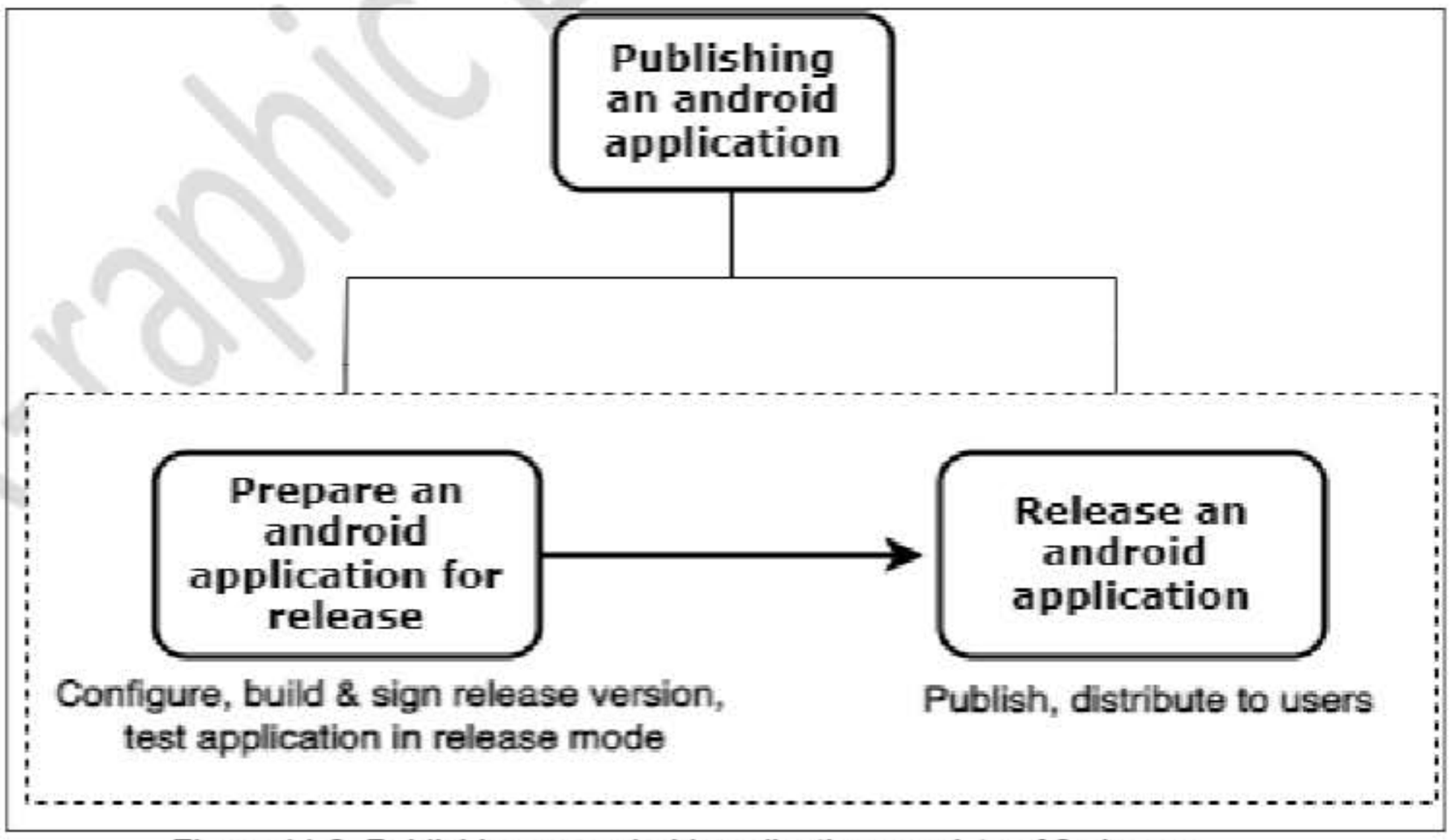

Figure 11.2: Publishing an android application consists of 2 phases

11.2.1 Preparing the application for release

The following tasks have to be performed before the application is released

1. Specify a suitable name and icon for the application
2. Provide proper versioning for the application
3. Remove unwanted resources, garbage code, and any logging calls used
4. Configure the application with the required permissions in the AndroidManifest.xml file
5. Update all the application resources like graphics and multimedia files
6. Prepare remote servers used by the application if any
7. Generate a self-signed certificate, digitally sign the application and build an apk
8. Test the release version

1. Specify a suitable name and icon for the application

Create and use an icon for the application, so that the users can identify and launch the application easily. It helps to manage the application once it is distributed. Choose a suitable name for the application before distribution because the name cannot be changed once it is distributed. These changes have to be made in the AndroidManifest.xml file by setting the values of android:icon and android:label attributes in the <application> element

```
<application>
        android:icon="@mipmap/xxx"          // specify an icon
        android:label="@string/app_name"    // application name
            . . . .
            . . . .
</application>
```

2. Provide proper versioning for the application

Versioning: It plays an important role in the application development life cycle.

- It helps users to identify which version of the release is installed on their device and upgrade if new release versions are available
- It helps to determine the compatibility and dependencies with other applications
- The version information can be used even by publishing services to perform the tasks depending upon the release version

There are 2 attributes used to set the application version

- versionCode
- versionName

These values are set in the AndroidManifest.xml file directly or in the gradle build file which is then merged during the process of build in the manifest file.

versionCode: It is a positive integer value used only by the android system for the internal purpose and is not shown/known to the users. The higher value indicates the latest version. This number is used by the android system to prevent users from installing apk with a lower number than the existing one on the device.
For e.g. android:versionCode = "1"

versionName: It is a string value given in the format "<major>.<minor>.<point>".
For e.g. android:versionName = "1.1.0"
where the major value is 1, the minor value is 1, and the point value is 0. The value of each of these is incremented by 1 based on the changes made.
<major> value is changed when a new feature/module is added
<minor> value is changed when some small features are added which were rolled out during development
<point> value is changed when some bugs are fixed and were not noticed earlier.
versionName represents the application release version.

3. Remove unwanted resources, garbage code, and any logging calls used

Any log calls used in the application should be removed. The debuggable attribute in the AndroidManifest.xml file should be set to false android:debuggable = "false". Any unwanted code, display messages, and resources used for testing flow and checking the proper functioning of the application should be removed.

4. Configure the application with the required permissions in the AndroidManifest.xml file

The permissions in the AndroidManifest.xml file should be verified.

- To send SMS through the application check whether the permission is set

```
<uses-permission android:name="android.permission.SEND_SMS"/>
```

- To read/write data stored on external storage verify

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

- The API levels used in the application are set by using attributes in <uses-sdk> element
  - android:minSdkVersion – It is used to specify the minimum level of API on which the android application can run. The default value is 1.
  - android:maxSdkVersion – It is used to specify the maximum level of API on which the android application can run. It is an optional attribute.
  - android:targetSdkVersion – It is used to specify the level of API on which the android application is designed to run.

```
<uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="19"
    / >
```

5. Update all the application resources like graphics and multimedia files

   Update the required graphics and image/icon files in the drawable folder and multimedia files such as audio, and video used by the application

6. Prepare remote servers used by the application if any

   If the android application is using any database stored on remote servers, check the connectivity and ensure the availability of the network so that the application works properly without any network issues.

7. Generate a self-signed certificate, digitally sign the application and build an apk

   Any android application has to be digitally signed by the developer before it is released in the market. All android applications require a digital certificate and a private key of that certificate to install/upload the application in Google play. Google does not appoint any authority to sign the certificates, they are self-signed by the developers.

The certificate is used to identify the developer and also share data with other applications that are signed with the same certificate. The same key is used by the developers to publish different or multiple applications. New versions/releases of the applications also use the same certificate for release in the market. Users on the other hand have to fully install the upgraded version if a different certificate is used. Android applications that are not signed are rejected by the emulator or any device.

The signing of an android application is done in 2 modes

- Debug
- Release

Signing in debug mode: In the android studio, any application that is developed is automatically signed in debug mode when the project is run from the IDE. By default, a debug certificate/Keystore with an unknown password and a private key with an unknown password is automatically generated by the android SDK. So whenever the developer makes the changes and runs the project on the emulator or even a real device that is connected using USB to the development machine, it does not ask for a password. However, an android application signed in debug mode cannot be distributed to the users.

Signing in release mode :

To sign a project in release mode, a Keystore and private key are created.

Keystore: It is a binary file that consists of a set of private keys. This file has to be kept in a secure and safe folder.

Private/Upload key: It is used to identify the author/developer of the application

There are 2 ways for generating a signed apk for an android application in android studio

1. On the menu bar, select build and then use Generate signed APK wizard and provide the required information
2. Configure the project to automatically release the signed APK during the build process by providing the required information in the Open Module setting, an option available in the project browser on right click of the app

Standard command line tools are also available which are used to generate keys and sign an android application

For e.g. keytool is used to generate keys for an android application

jarsigner is used to sign an android application

zipalign is used to optimize the .apk file

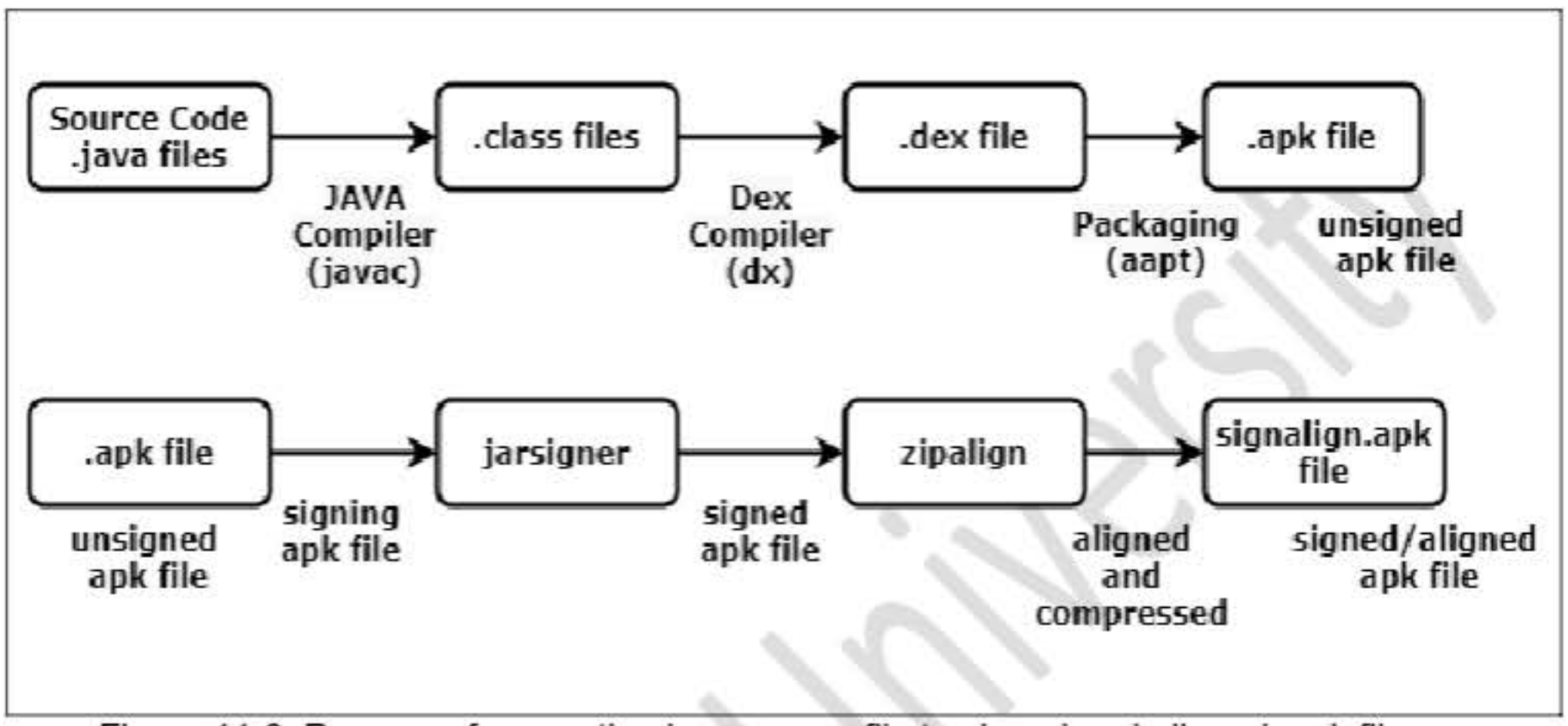

Figure 11.3: Process of converting java source file to signed and aligned .apk file

8. Test the release version

Test the release version of the application on real devices such as handsets, tablets, etc. to ensure that the user interface is user-friendly and working properly on all devices.

11.2.2 Releasing the android application to the users

Once the android application is prepared for release and tested it is ready to be distributed to the users

There are several ways by which the android application can be released.

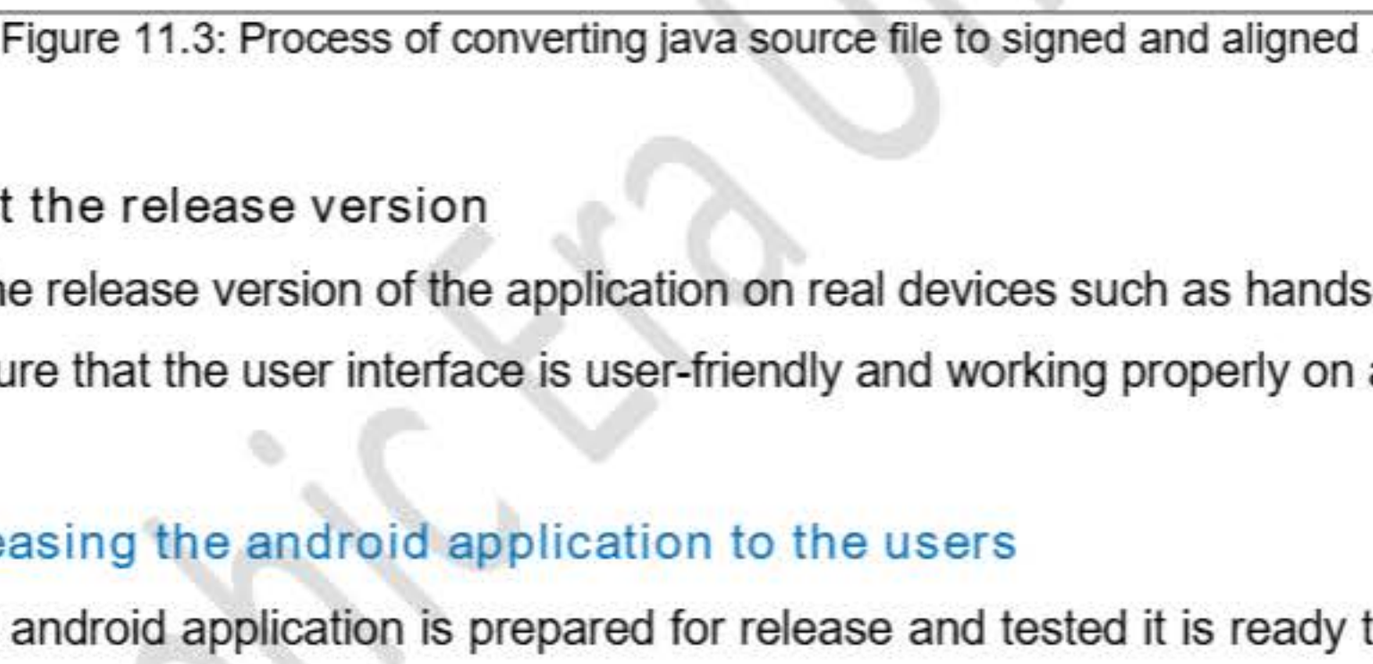

Figure 11.4: Different Ways of releasing android applications

- Email – It is one of the simplest and fastest ways to publish an android application. Email the .apk file to the user. But it is unsafe as anyone can pirate and use it.

- Download from a Website – Host the android application for release on a website so that the user can download and install it. But this is also unsafe as it's difficult to keep a track of the users who are downloading and using it.

- Bluetooth - .apk file can also be transferred to a mobile device by using Bluetooth by pairing it with the device. Only authorized users can be paired and allowed for file transfer.

- Google Play - It is an official marketplace for distributing android applications.
  Google Play supports publicity, selling, and distribution of android applications not only locally but throughout the world.
  To release an application in Google Play
  - Create a Google Developer Account by paying registration fees and signing the terms and agreement policies.
  - Product description, feedback questionnaire for content rating, and pricing information for the application are provided through the wizards before uploading the .apk files.
  - For uploading the app if the app size is more than 100 MB which is the limitation, APK expansion files are used.
  - The app can be published by either using alpha, beta, or production options.
  - Alpha version [closed testing] is available for testing only for a specific number of invited testers.
  - Any interested users, who have been given permission can test the Beta version and send feedback,
  - Production release is available to all users of the play store in the countries chosen.

11.3 Self-Assessment Questions

Q1. List and explain with a neat diagram the phases of publishing an android application. [8 marks, L2]

Q2. What are the tasks to be performed while preparing an android application for release? [8 marks, L2]

Q3. Explain the different ways of releasing an android application in brief.[8 marks, L2]

Q4. Explain how to generate a self-signed certificate, digitally sign the application and build an apk. [6 marks, L2]

Q5. Explain the signing of an android application in brief.[6 marks, L2]

Q6. Explain the elements which are to be updated while preparing an android application for release.[4 marks, L2]

Q7. Explain the versioning of an android application in brief.[6 marks, L2]

11.4 Self-Assessment Activities

Note: Use an appropriate emulator for the execution of an android application.

A1. Create an android application and display the message "My first android application for release".

- Run the application by using an emulator
- Run the application by using a real device
- Transfer the .apk file using Bluetooth to a real device and run the application.

11.5 Multiple-Choice Questions

1. The no of phases in publishing an android application is ___. [1 mark, L1]
   a) 2
   b) 3
   c) 4
   d) 8

2. To release an android application the file created is an ___ file. [1 mark, L1]
   a) exe
   b) dll
   c) apk
   d) None of the above

3. A suitable name and icon for an android application before release is set in the _____ file. [1 mark, L1]
   a) acitivity_main.xml
   b) AndroidManifest.xml
   c) MainActivity.java
   d) None of the above

4. The attributes used to set the version of an android application are ____. [1 mark, L1]
   a) appversionCode and appversionName
   b) appCode and appName
   c) versionCode and versionName
   d) None of the above

5. The format of the value used to set the versionName of an android application where "X" is an integer is ___. [1 mark, L1]
   a) "X"
   b) "X.X.X"
   c) "X.X"
   d) None of the above

6. The file that consists of a set of private keys used for releasing an android application is known as ____. [1 mark, L1]
   a) Keyfile
   b) KeyName
   c) KeyValues
   d) Keystore

7. To generate keys for an android application the tool used is ____. [1 mark, L1]
   a) keytool
   b) Keystore
   c) jarsigner
   d) zipalign

8. An android application signed in the _____ mode cannot be distributed to the users. [1 mark, L1]
   a) release
   b) debug
   c) private
   d) public

9. The attribute used to specify the level of API on which the android application is designed to run is _____. [1 mark, L1]
   a) android:minSdkVersion

b) android:maxSdkVersion
c) android:targetSdkVersion
d) None of the above

10. An android application cannot be released using _____.[1 mark, L1]
    a) email
    b) website
    c) Google Play
    d) Google Map

11.6 Key Answers to the multiple-choice question

1. The no of phases in publishing an android application is 2. [a]
2. To release an android application the file created is an apk file. [c]
3. A suitable name and icon for an android application before release is set in the AndroidManifest.xml file. [b]
4. The attributes used to set the version of an android application are versionCode and versionName.[c]
5. The format of the value used to set the versionName of an android application where "X" is an integer is "X.X.X". [b]
6. The file that consists of a set of private keys used for releasing an android application is known as Keystore. [d]
7. To generate keys for an android application the tool used is Keytool. [a]
8. An android application signed in the debug mode cannot be distributed to the users.[b]
9. The attribute used to specify the level of API on which the android application is designed to run is android:targetSdkVersion. [c]
10. An android application cannot be released using Google maps.[d]

11.7 Summary

Once an android application is developed and tested, it can be made available to the users. The final phase of android application development is Publishing. Users can use the application on their devices only after publishing. To release an application to the users, it is necessary to create a .apk file and distribute it to them so that the user can install and run the application on an android device.

The Source code (.java file) written in JAVA is converted by the JAVA compiler (javac) into a .class file. The Dex compiler then converts the generated .class file into a .dex file

which in turn is later converted into a .apk file by the Android Asset Packaging Tool (AAPT).

If the application is to be distributed on the google play store, the signing process has to be completed. Signing means adding additional private key information to the .apk file.

The process of publishing consists of 2 phases :

- Preparing the application for release
- Releasing the application to the users

11.8 Keywords

- Publishing
- Android Asset Packaging Tool (AAPT)
- Android Package Kit (APK)
- Keystore
- Keytool

11.9 Recommended Resources for Further Reading

Text Book(s)

1. Pradeep Kothari, "Android Application Development" Black-Book, Dreamtech Press,2015
2. Wei-Meng Lee, "Beginning Android Application Development", Wiley 2011.
3. Dawn Griffiths and David Griffiths, "Head First Android Development", O'Reilly, 2017.

References

- https://www.javatpoint.com/android-tutorial
- https://www.tutorialspoint.com/android/index.htm
- https://developer.android.com/guide/components/fundamentals

--*--