Competency Based Learning Material (CBLM)

Android Mobile Application Development

Level-4

Module: Implementing Background Service and Application Deployment

Code: CBLM-OU-ICT-AMAD-06-L4-V1

National Skills Development Authority
Prime Minister's Office
Government of the People's Republic of Bangladesh

Copyright

National Skills Development Authority
Prime Minister's Office
Level: 10-11, Biniyog Bhaban,
E-6 / B, Agargaon, Sher-E-Bangla Nagar Dhaka-1207, Bangladesh.
Email: ec@nsda.gov.bd
Website: www.nsda.gov.bd.
National Skills Portal: http:\\skillsportal.gov.bd

This Competency Based Learning Materials (CBLM) on "Implementing Background Service and Application Deployment" under the Android Mobile Application Development, Level-4" qualification is developed based on the national competency standard approved by National Skills Development Authority (NSDA)

This document is to be used as a key reference point by the competency-based learning materials developers, teachers/trainers/assessors as a base on which to build instructional activities.

National Skills Development Authority (NSDA) is the owner of this document. Other interested parties must obtain written permission from NSDA for reproduction of information in any manner, in whole or in part, of this Competency Standard, in English or other language.

It serves as the document for providing training consistent with the requirements of industry in order to meet the qualification of individuals who graduated through the established standard via competency-based assessment for a relevant job.

This document has been developed by NSDA with the assistance of related specialist/trainer /related employee

Public and private institutions may use the information contained in this CBLM for activities benefitting Bangladesh.

Approved by __ th Authority Meeting of NSDA Held on ----------

How to use this Competency Based Learning Material (CBLM)

The module, Implementing Background Service and Application Deployment contains training materials and activities for you to complete. These activities may be completed as part of structured classroom activities or you may be required you to work at your own pace. These activities will ask you to complete associated learning and practice activities in order to gain knowledge and skills you need to achieve the learning outcomes.

1. Review the **Learning Activity** page to understand the sequence of learning activities you will undergo. This page will serve as your road map towards the achievement of competence.

2. Read the **Information Sheets.** This will give you an understanding of the jobs or tasks you are going to learn how to do. Once you have finished reading the **Information Sheets** complete the questions in the **Self-Check.**

3. **Self-**Checks are found after each **Information Sheet**. **Self-Checks** are designed to help you know how you are progressing. If you are unable to answer the questions in the **Self-Check** you will need to re-read the relevant **Information Sheet**. Once you have completed all the questions check your answers by reading the relevant **Answer Keys** found at the end of this module.

4. Next move on to the **Job Sheets. Job Sheets** provide detailed information about *how to do the job* you are being trained in. Some **Job Sheets** will also have a series of **Activity Sheets**. These sheets have been designed to introduce you to the job step by step. This is where you will apply the new knowledge you gained by reading the Information Sheets. This is your opportunity to practice the job. You may need to practice the job or activity several times before you become competent.

5. Specification **sheets**, specifying the details of the job to be performed will be provided where appropriate.

6. A review of competency is provided on the last page to help remind if all the required assessment criteria have been met. This record is for your own information and guidance and is not an official record of competency

When working though this Module always be aware of your safety and the safety of others in the training room. Should you require assistance or clarification please consult your trainer or facilitator.

When you have satisfactorily completed all the Jobs and/or Activities outlined in this module, an assessment event will be scheduled to assess if you have achieved competency in the specified learning outcomes. You will then be ready to move onto the next Unit of Competency or Module

Table of Contents

Module Content

Unit of Competency	**Implement Background Service and Application Deployment**
Unit Code	**OU-ICT-AMAD-06-L4-V1**
Module Title	**Implementing Background Service and Application Deployment**
Module Descriptor	This module covers the knowledge, skills and attitudes required to implement Background Service and Application Deployment It includes the task of implementing broadcast receivers, starting a service, performing Test Driven Development (TDD) and introducing application signing and deployment.
Nominal Hours	**40** Hours
Lerning Outcome	After completing the practice of the module, the trainees will be able to perform the following jobs: 1. Implement broadcast receivers 2. Start a service 3. Perform Test Driven Development (TDD) 4. Introduce application signing and deployment

Assessment Criteria

1. Event receiving is explained.
2. An event is received.
3. A service is started using broadcast receiver.
4. Lifecycle of services is explained.
5. Different types of services are implemented.
6. Notification using service is generated.
7. Music playing as a background service is made.
8. TDD is explained.
9. Unit testing is performed.
10. UI testing is performed.
11. Key store file to make signed APK is generated.
12. Google play console dashboard is used.
13. Application signing and deployment is introduced.

Learning Outcome 1: Implement Broadcast Receivers

Assessment Criteria	1. Event receiving is explained. 2. An event is received. 3. A service is started using broadcast receiver.
Conditions and Resources	• Actual workplace or training environment • CBLM • Handouts • Job related tools, equipment, and materials • Multimedia Projector • Paper, Pen, Pencil, and Eraser • Internet Facilities • Whiteboard and Marker
Contents	1. Receiving techniques of an event 2. Starting a service by using broadcast receiver 3. Job scheduler 4. Work manager
Activities/job/Task	1. Implement Broadcast Receivers in Android App with following activity: a. Create a New Android Project: b. Define Custom Broadcast Actions c. Implement BroadcastReceiver for New Message d. Register and Unregister Receiver Dynamically e. Send Broadcast for New Message f. Implement BroadcastReceiver for Connectivity Change g. Add Permissions to the Manifest
Training Methods	• Blended • Discussion • Presentation • Demonstration • Guided Practice • Individual Practice • Project Work • Problem Solving • Brainstorming
Assessment Methods	Assessment methods may include but not limited to • Written Test • Demonstration • Oral Questioning

Learning Experience 1: Implement Broadcast Receivers

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Activities	Recourses/Special Instructions
1. Trainee will ask the instructor about the learning materials	1. Instructor will provide the learning materials Implement Broadcast Receivers
2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Implement Broadcast Receivers"	2. Read Information sheet 1: Implement Broadcast Receivers 3. Answer Self-check 1: Implement Broadcast Receivers 4. Check your answer with Answer key 1: Implement Broadcast Receivers.
3. Read the Job/Task Sheet and Specification Sheet and perform job/Task	5. Job/Task Sheet and Specification Sheet **Job Sheet 1.1:** Implement Broadcast Receivers in Android App with following activity: h. Create a New Android Project: i. Define Custom Broadcast Actions j. Implement BroadcastReceiver for New Message k. Register and Unregister Receiver Dynamically l. Send Broadcast for New Message m. Implement BroadcastReceiver for Connectivity Change n. Add Permissions to the Manifest **Specification Sheet 1.1:** Implement Broadcast Receivers in Android App with following activity:

Information Sheet 1: Implement Broadcast Receivers

Learning Objective: After completion of this information sheet, the learners will be able to explain, define and interpret the following contents

1.1 Receiving techniques of an event
1.2 Starting a service by using broadcast receiver
1.3 Job scheduler
1.4 Work manager

1.1 Receiving techniques of an event

In Android app development, "receiving techniques" typically refer to how your app can receive data or notifications from external sources, such as user inputs, system broadcasts, or network requests. Here are some common receiving techniques and how they are implemented:

Broadcast Receivers

Broadcast receivers allow your app to listen for system-wide broadcast announcements. For example, an app can listen for when the device battery is low or when a picture is captured.

- Static Broadcast Receivers: Declared in the AndroidManifest.xml.
- Dynamic Broadcast Receivers: Registered in the code (usually in an Activity or Service).

```
// Static Receiver in AndroidManifest.xml
<receiver android:name=".MyBroadcastReceiver">
   <intent-filter>
      <action android:name="android.intent.action.BOOT_COMPLETED" />
   </intent-filter>
</receiver>

// Dynamic Receiver in Code
IntentFilter filter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
MyBroadcastReceiver receiver = new MyBroadcastReceiver();
registerReceiver(receiver, filter);
```

Intents and Intent Filters

Intents are messaging objects used to request an action from another app component. Intent filters specify the types of intents that a component can respond to.

```
// Sending an Intent
Intent intent = new Intent(this, SecondActivity.class);
```

```
intent.putExtra("key", "value");
startActivity(intent);

// Receiving an Intent in SecondActivity
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);

    Intent intent = getIntent();
    String value = intent.getStringExtra("key");
}
```

Content Providers

Content providers manage access to a structured set of data. They encapsulate data and provide mechanisms for defining data security. They are the standard interface that connects data in one process with code running in another process.

```
// Querying a Content Provider
Cursor cursor = getContentResolver().query(
    ContactsContract.Contacts.CONTENT_URI,
    null,
    null,
    null,
    null
);
if (cursor != null && cursor.moveToFirst()) {
    String contactName =
cursor.getString(cursor.getColumnIndex(ContactsContract.Contacts.DISPLAY_NAM
E));
    cursor.close();
}
```

*Services**

Services are components that run in the background to perform long-running operations. They can communicate with other components via binding.

```
```java
// Starting a Service
Intent serviceIntent = new Intent(this, MyService.class);
startService(serviceIntent);

// Service Code
```
```

```
public class MyService extends Service {
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // Handle the service task here
        return START_STICKY;
    }

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

Network Requests

Using libraries such as Retrofit or Volley, you can make network requests to fetch or send data over the internet.

```
// Using Retrofit for a Network Request
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.example.com/")
    .addConverterFactory(GsonConverterFactory.create())
    .build();

ApiService apiService = retrofit.create(ApiService.class);
Call<List<User>> call = apiService.getUsers();
call.enqueue(new Callback<List<User>>() {
    @Override
    public void onResponse(Call<List<User>> call, Response<List<User>> response) {
        if (response.isSuccessful()) {
            List<User> users = response.body();
            // Process the list of users
        }
    }

    @Override
    public void onFailure(Call<List<User>> call, Throwable t) {
        // Handle the error
    }
});
```

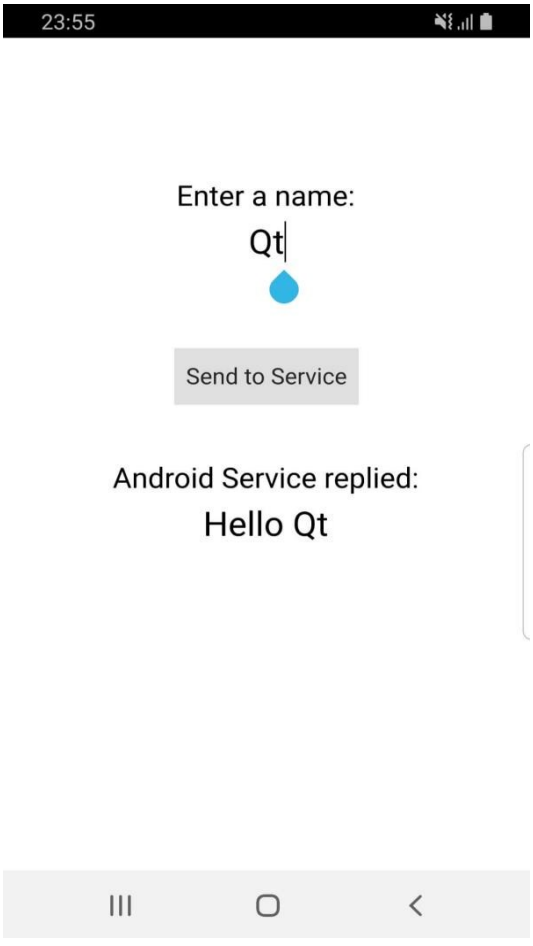
Push Notifications

Using Firebase Cloud Messaging (FCM), you can receive push notifications even when your app is not running.

```
// Receiving a Push Notification
public class MyFirebaseMessagingService extends FirebaseMessagingService {
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        // Handle the received message
        if (remoteMessage.getNotification() != null) {
            String message = remoteMessage.getNotification().getBody();
            // Display the notification or take appropriate action
        }
    }
}
```

These techniques enable your app to effectively receive and handle data from various sources, enhancing the interactivity and functionality of your Android application.

1.2 Starting a service by using broadcast receiver



When clicking the Send to Service button, the name entered in the QML view, Qt, in this case, is sent to the Android service. Then, the service replies back with a message Hello Qt which is printed in the QML view.

Running the Example

To run the example from Qt Creator, open the **Welcome** mode and select the example from **Examples**. For more information, visit Building and Running an Example.

Create the Service

When running the app's process, you can extend either QtService or Service. Extending QtService allows Qt to load all the necessary libraries to load Qt components correctly and call native methods on Android. However, here the service is running in the same process, and with the BroadcastReceiver you don't need native calls to exchange messages with Qt, so extending either class works.

Start by creating the Java service class. This is a normal Android Service that receives a name from QML and replies back with Hello <name>:

package org.qtproject.example.qtandroidservice;

```
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.os.IBinder;
import org.qtproject.qt5.android.bindings.QtService;
import android.content.IntentFilter;
public class QtAndroidService extends QtService
{
    private static final String TAG = "QtAndroidService";
    @Override
    public void onCreate() {
        super.onCreate();
        Log.i(TAG, "Creating Service");
    }
    @Override
    public void onDestroy() {
        super.onDestroy();
        Log.i(TAG, "Destroying Service");
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        int ret = super.onStartCommand(intent, flags, startId);

        String name = new String(intent.getByteArrayExtra("name"));
        Intent sendToUiIntent = new Intent();
```

```
        sendToUiIntent.setAction(ActivityUtils.BROADCAST_NAME_ACTION);
        sendToUiIntent.putExtra("name", name);
        Log.i(TAG, "Service sending broadcast");
        sendBroadcast(sendToUiIntent);

        return ret;
    }

    @Override
    public IBinder onBind(Intent intent) {
        return super.onBind(intent);
    }
}
```

In the owerwritten method onStartCommand(), the service receives a name from the calling intent, then sends a broadcast to the BroadcastReceiver, which in turn will call the native method sendToQt(String message). For more information on managing native calls in Qt, see Calling QML/C++ Functions from Java Code.

Since the service is run on a separate .so lib file, you must create a sub-project for the service process which uses QAndroidService. Start with a .pro file as follows:

```
TEMPLATE = lib
TARGET = service
CONFIG += dll
QT += core androidextras

SOURCES += \
    service_main.cpp
```

Then, create the file service_main.cpp:

```
#include <QDebug>
#include <QAndroidService>

int main(int argc, char *argv[])
{
    qWarning() << "Service starting with BroadcastReceiver from separate .so file";
    QAndroidService app(argc, argv);

    return app.exec();
}
```

Manage the AndroidManifest.xml File

To use the service, it must be declared in the AndroidManifest.xml file

```
<service android:process=":qt_service" android:name=".QtAndroidService">
    <meta-data android:name="android.app.lib_name" android:value="service"/>
    <meta-data android:name="android.app.qt_sources_resource_id"
android:resource="@array/qt_sources"/>
    <meta-data android:name="android.app.repository" android:value="default"/>
    <meta-data android:name="android.app.qt_libs_resource_id"
android:resource="@array/qt_libs"/>
    <meta-data android:name="android.app.bundled_libs_resource_id"
android:resource="@array/bundled_libs"/>
    <!-- Deploy Qt libs as part of package -->
    <meta-data android:name="android.app.bundle_local_qt_libs" android:value="--
%%BUNDLE_LOCAL_QT_LIBS%% --"/>
    <!-- Run with local libs -->
    <meta-data android:name="android.app.use_local_qt_libs" android:value="--
%%USE_LOCAL_QT_LIBS%% --"/>
    <meta-data android:name="android.app.libs_prefix"
android:value="/data/local/tmp/qt/"/>
    <meta-data android:name="android.app.load_local_libs_resource_id"
android:resource="@array/load_local_libs"/>
    <meta-data android:name="android.app.load_local_jars" android:value="--
%%INSERT_LOCAL_JARS%% --"/>
    <meta-data android:name="android.app.static_init_classes" android:value="--
%%INSERT_INIT_CLASSES%% --"/>
    <!-- Run with local libs -->
    <!-- Background running -->
    <meta-data android:name="android.app.background_running"
android:value="true"/>
    <!-- Background running -->
  </service>
```

Start the Service

Take the following steps to set up and start the service:

a. Register the native method
b. Create the BroadcastReceiver in a custom Java class:

```
package org.qtproject.example.qtandroidservice;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.content.BroadcastReceiver;
import android.content.IntentFilter;
public class ActivityUtils {
 private static native void sendToQt(String message);
 private static final String TAG = "ActivityUtils";
   public static final String BROADCAST_NAME_ACTION =
"org.qtproject.example.qtandroidservice.broadcast.name";
   public void registerServiceBroadcastReceiver(Context context) {
      IntentFilter intentFilter = new IntentFilter();
      intentFilter.addAction(BROADCAST_NAME_ACTION);
      context.registerReceiver(serviceMessageReceiver, intentFilter);
      Log.i(TAG, "Registered broadcast receiver");
   }
   private BroadcastReceiver serviceMessageReceiver = new BroadcastReceiver() {
      @Override
      public void onReceive(Context context, Intent intent) {
         Log.i(TAG, "In OnReceive broadcast receiver");
         if (BROADCAST_NAME_ACTION.equals(intent.getAction())) {
            String name = intent.getStringExtra("name");
            Log.i(TAG, "Service received name: " + name);
            String message = "Hello " + name;
            sendToQt(message);
            Log.i(TAG, "Service sent back message: " + message);
         }
      }
   };
}
```

c. *Register the BroadcastReceiver*

```
void QtAndroidService::registerBroadcastReceiver()
{
 QAndroidJniEnvironment env;
   jclass javaClass =
env.findClass("org/qtproject/example/qtandroidservice/ActivityUtils");
   QAndroidJniObject classObject(javaClass);

   classObject.callMethod<void>("registerServiceBroadcastReceiver",
```

```
                    "(Landroid/content/Context;)V",
                    QtAndroid::androidContext().object());
}
```

d. Call the startService() method, as follows

```
void QtAndroidService::sendToService(const QString &name)
{
    QAndroidIntent serviceIntent(QtAndroid::androidActivity().object(),
                    "org/qtproject/example/qtandroidservice/QtAndroidService");
    serviceIntent.putExtra("name", name.toUtf8());
    QAndroidJniObject result = QtAndroid::androidActivity().callObjectMethod(
            "startService",
            "(Landroid/content/Intent;)Landroid/content/ComponentName;",
            serviceIntent.handle().object());
}
```

This function is used to start the Service. If the service is already running, it will only send the names without starting a new service instance.

1.3 Job scheduler

In Android, the JobScheduler is a system component that allows you to schedule tasks to run at specific times or under specific conditions. It's a more efficient and reliable way to perform tasks in the background compared to traditional approaches like services or broadcasts.

What are Jobs?

In Android, a Job is a unit of work that needs to be executed. Jobs are scheduled by the JobScheduler, which is responsible for managing the execution of these jobs.

Key features of JobScheduler:

a. Scheduled execution: Jobs can be scheduled to run at specific times, intervals, or under specific conditions (e.g., when the device is charging).
b. Background execution: Jobs are executed in the background, without interrupting the user or affecting the user experience.
c. Persistence: Jobs are persisted even if the device is restarted or the app is closed.
d. Prioritization: Jobs can be prioritized based on their importance, ensuring that critical tasks are executed first.

To use JobScheduler, you'll need to:

Step 1: Create a JobService

Create a new class that extends JobService. This service will handle the execution of your job.

For example:

```
public class MyJobService extends JobService {
    @Override
    public boolean onStartJob(JobParameters jobParameters) {
        // Perform your task here
        return true;
    }

    @Override
    public boolean onStopJob(JobParameters jobParameters) {
        // Clean up any resources used by the job
        return true;
    }
}
```

Step 2: Schedule a Job

Use the JobScheduler API to schedule your job. You can schedule a job using one of the following methods:

- schedule() : Schedules a one-time job to run at a specific time or interval.
- schedule() : Schedules a recurring job to run at a specific interval.
- enqueue() : Enqueues a job for execution, allowing the system to schedule it based on availability.

For example:

```
JobScheduler js = (JobScheduler)
context.getSystemService(Context.JOB_SCHEDULER_SERVICE);
JobInfo jobInfo = new JobInfo.Builder(JOB_ID, MyJobService.class)
        .setRequiredNetworkType(JobInfo.NETWORK_TYPE_UNMETERED)
        .setRequiresCharging(true)
        .build();
js.schedule(jobInfo);
```

Step 3: Handle Job Completion

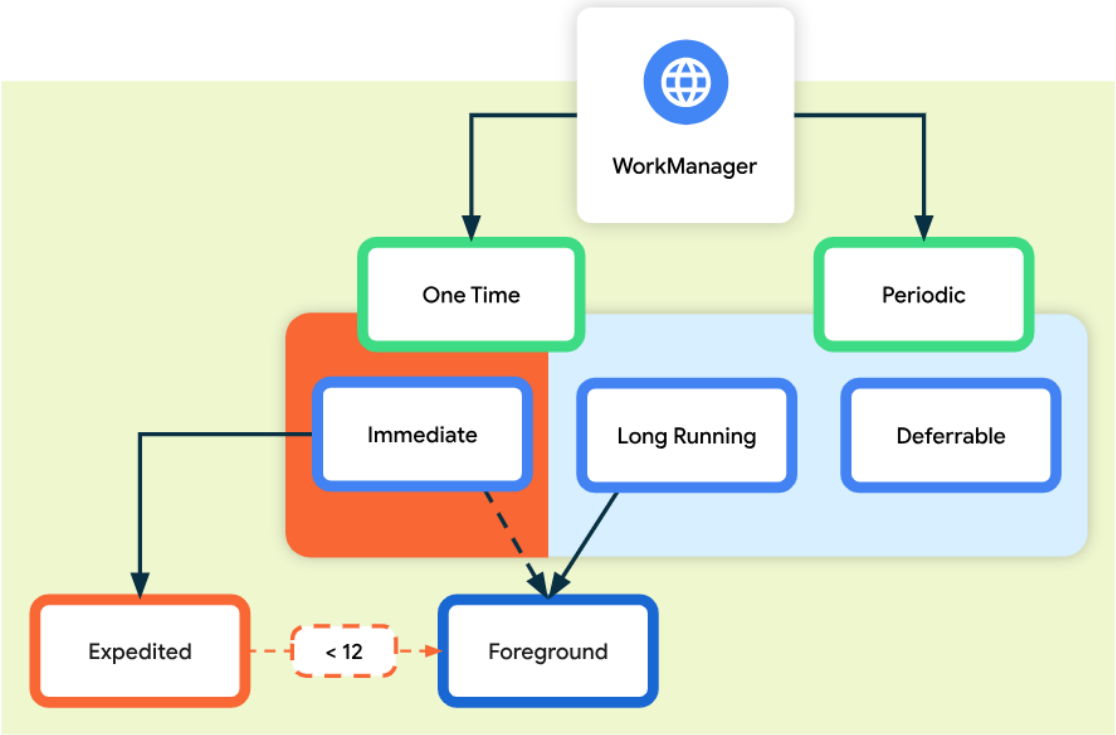
In your MyJobService class, override the onStartJob() and onStopJob() methods to handle the execution and completion of your job.

For example:

```
@Override
public boolean onStartJob(JobParameters jobParameters) {
    // Perform your task here
    return true;
}
@Override
public boolean onStopJob(JobParameters jobParameters) {
    // Clean up any resources used by the job
    return true;}
```

1.4 Work manager

Work is persistent when it remains scheduled through app restarts and system reboots. WorkManager is the recommended solution for persistent work. Because most background processing is best accomplished through persistent work, WorkManager is therefore also the primary recommended API for background processing in general.



Types of persistent work

WorkManager handles three types of persistent work:

- **Immediate**: Tasks that must begin immediately and complete soon. May be expedited.
- **Long Running**: Tasks which might run for longer, potentially longer than 10 minutes.
- **Deferrable**: Scheduled tasks that start at a later time and can run periodically.

Key Features of WorkManager

- **Guaranteed Execution**: Ensures that work is completed even if the app exits or the device restarts.
- **Constraints**: Allows you to specify conditions under which the work should run, such as network availability, charging status, etc.
- **Chaining Work**: Supports chaining of tasks, where tasks can run sequentially or in parallel.
- **Periodic Work**: Allows scheduling of recurring tasks.

Self-Check - 1: Implement Broadcast Receivers

1. What is a Broadcast Receiver in Android?
   Answer:
2. How do you declare a Broadcast Receiver in the AndroidManifest.xml?
   Answer:
3. hat is the purpose of the `onReceive()` method in a Broadcast Receiver?
   Answer:
4. Can Broadcast Receivers start an activity or service?
   Answer:
5. What is the difference between a static and a dynamic Broadcast Receiver?**
   Answer:
6. How can you register a dynamic Broadcast Receiver?
   Answer:
7. How can you unregister a dynamic Broadcast Receiver?
   Answer:
8. What is an example of a system broadcast that an app can listen for?
   Answer:
9. How do you start a service from a Broadcast Receiver?
   Answer:
10. What is JobScheduler in Android?
    Answer:
11. How do you schedule a job using JobScheduler?
    Answer:
12. What is WorkManager in Android?
    Answer:
13. What are the key advantages of using WorkManager over JobScheduler?
    Answer:
14. How do you define a work task using WorkManager?
    Answer:
15. What is the difference between `OneTimeWorkRequest` and `PeriodicWorkRequest` in WorkManager?
    Answer:

Answer Key - 1: Implement Broadcast Receivers

1. What is a Broadcast Receiver in Android?

   Answer: A Broadcast Receiver is a component that responds to broadcast messages from other applications or the system.

2. How do you declare a Broadcast Receiver in the AndroidManifest.xml?

   Answer: You declare it using the `<receiver>` tag in the AndroidManifest.xml file.

3. hat is the purpose of the `onReceive()` method in a Broadcast Receiver?

   Answer: The `onReceive()` method is called when the Broadcast Receiver receives an event or broadcast message.

4. Can Broadcast Receivers start an activity or service?

   Answer: Yes, Broadcast Receivers can start an activity or service by using an Intent.

5. What is the difference between a static and a dynamic Broadcast Receiver?**

   Answer: Static Broadcast Receivers are declared in the AndroidManifest.xml, while dynamic Broadcast Receivers are registered and unregistered in code (at runtime).

6. How can you register a dynamic Broadcast Receiver?

   Answer: By calling the `registerReceiver()` method in the activity or service.

7. How can you unregister a dynamic Broadcast Receiver?

   Answer: By calling the `unregisterReceiver()` method in the activity or service.

8. What is an example of a system broadcast that an app can listen for?

   Answer: An example is `android.intent.action.BOOT_COMPLETED`, which is broadcasted when the device finishes booting.

9. How do you start a service from a Broadcast Receiver?

   Answer: By calling `context.startService()` or `context.startForegroundService()` from the `onReceive()` method.

10. What is JobScheduler in Android?

    Answer: JobScheduler is an API that allows scheduling of jobs to run at a later time or under specific conditions, suitable for long-running tasks.

11. How do you schedule a job using JobScheduler?

    Answer: By creating a `JobInfo` object and using `JobScheduler.schedule(JobInfo job)`.

12. What is WorkManager in Android?

    Answer: WorkManager is an API for scheduling deferrable, asynchronous tasks that need guaranteed execution.

13. What are the key advantages of using WorkManager over JobScheduler?

    Answer: WorkManager supports constraints, chained tasks, and provides backward compatibility.

14. How do you define a work task using WorkManager?

    Answer: By creating a `Worker` class and overriding the `doWork()` method.

15. What is the difference between `OneTimeWorkRequest` and `PeriodicWorkRequest` in WorkManager?

    Answer: OneTimeWorkRequest` schedules a task to run once, while `PeriodicWorkRequest` schedules a task to run periodically.

Job Sheet-1.1: Implement Broadcast Receivers in Android App

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Collect the resources from your trainer as per the job requirement.
5. Create a New Android Project:
6. Define Custom Broadcast Actions
7. Implement BroadcastReceiver for New Message
8. Register and Unregister Receiver Dynamically
9. Send Broadcast for New Message
10. Implement BroadcastReceiver for Connectivity Change
11. Add Permissions to the Manifest
12. Run the App and Show your work to the trainer.
13. Clean your work place as per standard procedure.
14. Turn off the computer and clean your workplace.

Specification Sheet-1.1: Implement Broadcast Receivers in Android App

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Learning Outcome 2: Start a service

Assessment Criteria	1. Lifecycle of services is explained. 2. Types of services are implemented. 3. Notification using service is generated. 4. Music playing as a background service is made.
Conditions and Resources	• Actual workplace or training environment • CBLM • Handouts • Job related tools, equipment, and materials • Multimedia Projector • Paper, Pen, Pencil, and Eraser • Internet Facilities • Whiteboard and Marker
Contents	2.1 Lifecycle of services a. On start command b. On bind c. On create d. On destroy e. Bind service f. Stop self g. Stop service 2.2 Types of services 2.3 Notification using services
Activities/job/Task	1. Start a Background Service in Android with following activity: a. Create a New Android Project b. Design a Music Player UI c. Implement a Background Service d. Handle Music Playback in the Service e. Start the Service from the Activity
Training Methods	• Blended • Discussion • Presentation • Demonstration • Guided Practice • Individual Practice • Project Work • Problem Solving • Brainstorming
Assessment Methods	Assessment methods may include but not limited to • Written Test • Demonstration • Oral Questioning

Learning Experience 2: Start a service

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Activities	Recourses/Special Instructions
1. Trainee will ask the instructor about the learning materials	1. Instructor will provide the learning materials Implement Broadcast Receivers
2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Start a service"	2. Read Information sheet 2: Start a service Answer Self-check 2: Start a service Check your answer with Answer key 2: Start a service
3. Read the Job/Task Sheet and Specification Sheet and perform job/Task	5. Job/Task Sheet and Specification Sheet **Job Sheet 2.1:** Start a Background Service in Android with following activity: a. Create a New Android Project b. Design a Music Player UI c. Implement a Background Service d. Handle Music Playback in the Service e. Start the Service from the Activity **Specification Sheet 2.1:** Start a Background Service in Android with following activity

Information Sheet 2: Start a service

Learning Objective: After completion of this information sheet, the learners will be able to explain, define and interpret the following contents

2.1 Lifecycle of services
2.2 Types of services
2.3 Notification using services

2.1 Lifecycle of services

Android services life-cycle can have two forms of services and they follow two paths, that are:

- Started Service
- Bounded Service

Let us see these services and their approach.

Started Service

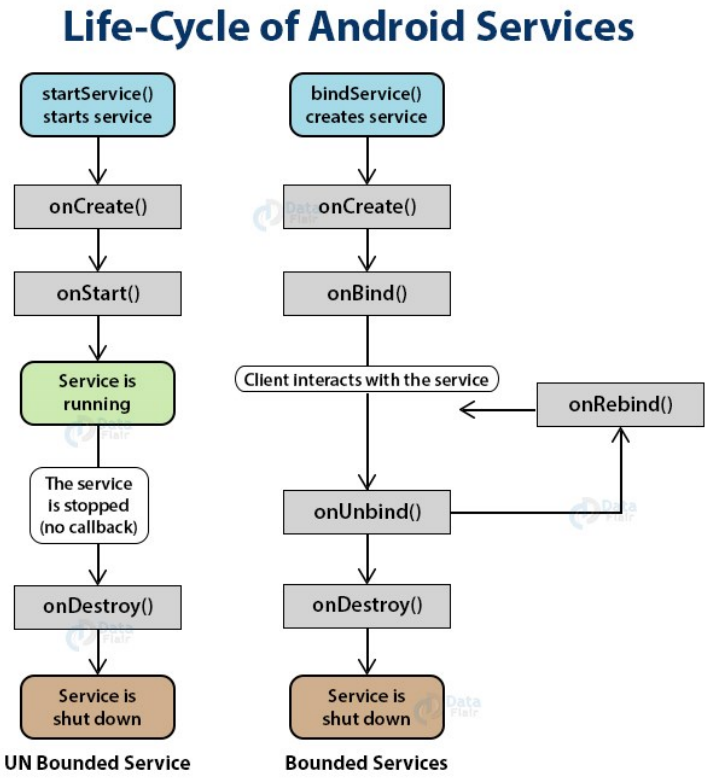
A service becomes started only when an application component calls startService(). It performs a single operation and doesn't return any result to the caller. Once this service starts, it runs in the background even if the component that created it destroys. This service can be stopped only in one of the two cases:

- By using the stopService() method.
- By stopping itself using the stopSelf() method.

Bound Service

A service is bound only if an application component binds to it using bindService(). It gives a client-server relation that lets the components interact with the service. The components can send requests to services and get results.

This service runs in the background as long as another application is bound to it. Or it can be unbound according to our requirement by using the unbindService() method.



IntentService()

There's an additional service class, that extends Service class, IntentService Class. It is a base class for services to handle asynchronous requests. It enables running an operation on a single background. It executes long-running programs without affecting any user's interface interaction. Intent services run and execute in the background and terminate themself as soon as they are executed completely.

Certain important features of Intent are :

- It queues up the upcoming request and executes them one by one.
- Once the queue is empty it stops itself, without the user's intervention in its lifecycle.
- It does proper thread management by handling the requests on a separate thread.

Methods of Android Services

The service base class defines certain callback methods to perform operations on applications. When we talk about Android services it becomes quite obvious that these services will do some operations and they'll be used. The following are a few important methods of Android services

- onStartCommand()
- onBind()
- onCreate()
- onUnbind()
- onDestroy()
- onRebind()

Fundamentals of Android Services

A user-defined service can be created through a normal class which is extending the **class Service**. Further, to carry out the operations of service on applications, there are certain callback methods which are needed to be **overridden**. The following are some of the important methods of Android Services:

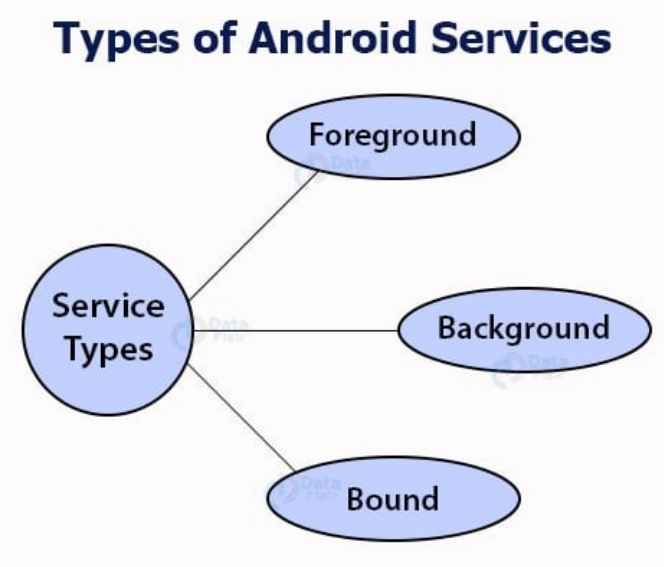
Methods	Description
onStartCommand()	The Android service calls this method when a component(eg: activity) requests to start a service using startService(). Once the service is started, it can be stopped explicitly using stopService() or stopSelf() methods.
onBind()	This method is mandatory to implement in android service and is invoked whenever an application component calls the bindService() method in order to bind itself with a service. User-interface is also provided to communicate with the service effectively by returning an IBinder object. If the binding of service is not required then the method must return null.
onUnbind()	The Android system invokes this method when all the clients get disconnected from a particular service interface.
onRebind()	Once all clients are disconnected from the particular interface of service and there is a need to connect the service with new clients, the system calls this method.
onCreate()	Whenever a service is created either using onStartCommand() or onBind(), the android system calls this method. This method is necessary to perform a one-time-set-up.
onDestroy()	When a service is no longer in use, the system invokes this method just before the service destroys as a final clean up call. Services must implement this method in order to clean up resources like registered listeners, threads, receivers, etc.

2.2 Types of services

Services in Android are a special component that facilitates an application to run in the background in order to perform long-running operation tasks. The prime aim of a service is to ensure that the application remains active in the background so that the user can operate multiple applications at the same time. A user-interface is not desirable for android services as it is designed to operate long-running processes without any user intervention. A service can run continuously in the background even if the application is closed or the user switches to another application. Further, application components can bind itself to service to carry out inter-process communication(IPC). There is a major difference between android services and threads, one must not be confused between the two. Thread is a feature provided by the Operating system to allow the user to perform operations in the background. While service is an android component that performs a long-running operation about which the user might not be aware of as it does not have UI.

Types of Android Services

When we talk about services, they can be of three types as shown in the figure below:



Foreground Services

Foreground services are those services that are visible to the users. The users can interact with them at ease and track what's happening. These services continue to run even when users are using other applications.

The perfect example of this is Music Player and Downloading.

Background Services

These services run in the background, such that the user can't see or access them. These are the tasks that don't need the user to know them.

Syncing and Storing data can be the best example.

Bound Services

Bound service runs as long as some other application component is bound to it. Many components can bind to one service at a time, but once they all unbind, the service will destroy.

To bind an application component to the service, bindService() is used.

2.3 Notification using services

Android Notification provides short, timely information about the action happened in the application, even it is not running. The notification displays the icon, title and some amount of the content text.

Set Android Notification Properties

The properties of Android notification are set using **NotificationCompat.Builder** object. Some of the notification properties are mention below:

- **setSmallIcon()**: It sets the icon of notification.
- **setContentTitle()**: It is used to set the title of notification.
- **setContentText()**: It is used to set the text message.
- **setAutoCancel()**: It sets the cancelable property of notification.
- **setPriority()**: It sets the priority of notification.

Android Notification Example

In this example, we will create a notification message which will launch another activity after clicking on it.

```
package example.javatpoint.com.androidnotification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.support.v4.app.NotificationCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
 public class MainActivity extends AppCompatActivity {
    Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
            addNotification();
        }
      });
  }
    private void addNotification() {
      NotificationCompat.Builder builder =
            new NotificationCompat.Builder(this)
                  .setSmallIcon(R.drawable.messageicon) //set icon for notification
                  .setContentTitle("Notifications Example") //set title of notification
                  .setContentText("This is a notification message")//this is notification me
ssage
                  .setAutoCancel(true) // makes auto cancel of notification
                  .setPriority(NotificationCompat.PRIORITY_DEFAULT); //set priority
of notification
        Intent notificationIntent = new Intent(this, NotificationView.class);
      notificationIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
      //notification message will get at NotificationView
      notificationIntent.putExtra("message", "This is a notification message");
       PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, notificationInte
nt,
            PendingIntent.FLAG_UPDATE_CURRENT);
      builder.setContentIntent(pendingIntent);
       // Add as notification
      NotificationManager manager = (NotificationManager) getSystemService(Conte
xt.NOTIFICATION_SERVICE);
      manager.notify(0, builder.build());
  }
}
```

Self-Check - 2: Start a service

1. What is a service in software development?

   Answer:
2. What are the main states in the lifecycle of a service in Android?

   Answer:
3. How is a service started in Android?

   Answer:
4. What method is called when a service is created in Android?

   Answer:
5. What happens when `stopService()` is called?

   Answer:
6. What is a foreground service?

   Answer:
7. What is a background service?

   Answer:
8. What is a bound service?

   Answer:
9. What is a remote service?

   Answer:
10. What is an intent service?

    Answer:
11. How do you create a notification in Android?

    Answer:
12. What is a notification channel in Android?

    Answer:
13. How do you set a notification channel for a notification?

    Answer:
14. What is a foreground notification?

    Answer:
15. How can you update a notification in Android?

    Answer:

Answer Key - 2: Start a service

1. What is a service in software development?
   Answer: A service is a reusable component that performs a specific task, often accessible over a network. In mobile and web applications, services can run in the background to perform long-running operations without user interaction.

2. What are the main states in the lifecycle of a service in Android?
   Answer: The main states are: `Started`, `Bound`, and `Destroyed`.

3. How is a service started in Android?
   Answer: A service is started in Android using `startService()` or `startForegroundService()`.

4. What method is called when a service is created in Android?
   Answer: The `onCreate()` method is called when a service is created.

5. What happens when `stopService()` is called?
   Answer: When `stopService()` is called, the service is stopped, and the `onDestroy()` method is called.

6. What is a foreground service?
   Answer: A foreground service performs operations noticeable to the user and must display a notification. It is less likely to be killed by the system.

7. What is a background service?
   Answer: A background service performs operations that the user is not directly aware of. It has a higher chance of being killed by the system to free resources.

8. What is a bound service?
   Answer: A bound service allows components (such as activities) to bind to it and interact with it through a communication interface.

9. What is a remote service?
   Answer: A remote service runs in a separate process and can be accessed by other applications. It requires inter-process communication (IPC).

10. What is an intent service?
    Answer: An intent service is a subclass of `Service` that handles asynchronous requests (expressed as Intents) on demand. It automatically stops itself when it has completed handling all its requests.

11. How do you create a notification in Android?
    Answer: You create a notification in Android using the `NotificationCompat.Builder` class.

12. What is a notification channel in Android?
    Answer: A notification channel is used to group notifications into manageable categories on devices running Android 8.0 (API level 26) and higher.

13. How do you set a notification channel for a notification?
    Answer: You set a notification channel by using the `setChannelId()` method on the `NotificationCompat.Builder`.

14. What is a foreground notification?
    Answer: A foreground notification is a persistent notification that is required by a foreground service to keep the user informed of its ongoing operation.

15. How can you update a notification in Android?
    Answer: You can update a notification by issuing a new notification with the same notification ID using the `NotificationManager`

Job Sheet-2.1: Start a Background Service in Android

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Collect the resources from your trainer as per the job requirement.
5. Create a New Android Project
6. Design a Music Player UI
7. Implement a Background Service
8. Handle Music Playback in the Service
9. Start the Service from the Activity
10. Run the App and Show your work to the trainer.
11. Clean your work place as per standard procedure.
12. Turn off the computer and clean your workplace.

Specification Sheet-2.1: Start a Background Service in Android

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Learning Outcome 3: Perform Test Driven Development (TDD)

Assessment Criteria	1. TDD is explained. 2. Unit testing is performed. 3. UI testing is performed.
Conditions and Resources	• Actual workplace or training environment • CBLM • Handouts • Job related tools, equipment, and materials • Multimedia Projector • Paper, Pen, Pencil, and Eraser • Internet Facilities • Whiteboard and Marker
Contents	1. TDD 2. Unit testing 3. UI testing
Activities/job/Task	2. Implement Broadcast Receivers in Android App with following activity: a. Create a New Android Project: b. Define Custom Broadcast Actions c. Implement BroadcastReceiver for New Message d. Register and Unregister Receiver Dynamically e. Send Broadcast for New Message f. Implement BroadcastReceiver for Connectivity Change g. Add Permissions to the Manifest
Training Methods	• Blended • Discussion • Presentation • Demonstration • Guided Practice • Individual Practice • Project Work • Problem Solving • Brainstorming
Assessment Methods	Assessment methods may include but not limited to • Written Test • Demonstration • Oral Questioning

Learning Experience 3: Perform Test Driven Development (TDD)

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Activities	Recourses/Special Instructions
4. Trainee will ask the instructor about the learning materials	1. Instructor will provide the learning materials Perform Test Driven Development (TDD)
5. Read the Information sheet and complete the Self Checks & Check answer sheets on "Perform Test Driven Development (TDD)"	2. Read Information sheet 3: Perform Test Driven Development (TDD) 3. Answer Self-check 3: Perform Test Driven Development (TDD) 4. Check your answer with Answer key 3: Perform Test Driven Development (TDD).
6. Read the Job/Task Sheet and Specification Sheet and perform job/Task	5. Job/Task Sheet and Specification Sheet **Job Sheet 3.1:** Perform unit and UI Testing **Specification Sheet 3.1:** Perform unit and UI Testing

Information Sheet 3: Perform Test Driven Development (TDD)

Learning Objective: After completion of this information sheet, the learners will be able to explain, define and interpret the following contents

3.1 TDD is explained.
3.2 Unit testing is performed.
3.3 UI testing is performed.

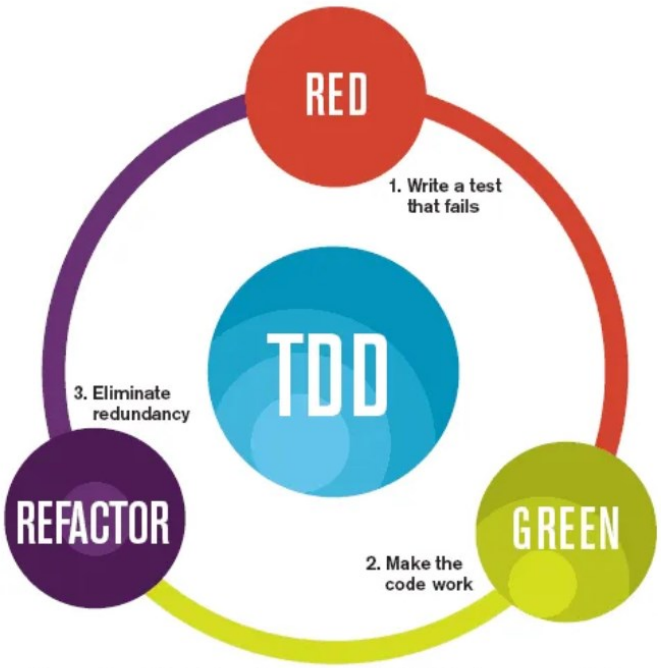
3.1 TDD is explained

Prior to writing the actual code, Test-Driven Development (TDD) emphasizes the production of unit test cases. It iteratively combines development, the creation of unit tests, and refactoring.

The origins of the TDD methodology are the Agile manifesto and Extreme programming. As its name implies, software development is driven by testing. In addition, it is an approach for structuring code that enables developers and testers to produce code that is both efficient and resilient over time.

Based on their initial comprehension, engineers begin writing small test cases for each feature using TDD. This method aims to only modify or develop new code if tests fail. This avoids multiple scripts for testing.

TDD can be represented by the Red-Green-Refactor Cycle



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

It comprises three essential steps:

- Develop a test that will not pass (Red)
- Develop code that can pass a test (Green)
- Refactorize your code to attain high code quality (Refactor)

So what exactly does it mean? Before writing any code for a new project, you should be able to write a test that fails, then write the code necessary for the test to pass, then rewrite the code if necessary and begin the cycle again with another test.

Obviously, it is not always required to test every component of your application, especially when developing with Flutter; for example you will rarely need to test your entire UI and verify that each AppBar is displayed correctly,. Nevertheless, it may be beneficial to unit test some API calls if your application is consuming data from an external service or to do database-related tests if your application makes extensive use of the database. TDD will ultimately improve the stability and quality of your code greatly, especially if you maintain or contribute open-source code.

TDD important in Android

a. **Easier code maintenance:** With TDD, developers write code that is more readable, manageable, and maintainable. Also, it requires less effort to concentrate on smaller, more digestible code bits. When transferring a project to a different individual or group, it is advantageous to have clean code.
b. **Allows for Modular Design:** The focus is placed on a particular feature at a time until the test has been passed. These iterations make finding bugs and reusing code in a project simple. In addition, solution architecture is improved by adherence to certain design principles.
c. **Facilitates Code Refactoring:** Refactoring is the process of optimising existing code in order to make it easier to implement. If the code for a modest update or improvement passes the preliminary tests, it can be refactored to acceptable standards. This is a necessary TDD process step.
d. **Reduces the dependency on code documentation:** The TDD methodology eliminates the need for time-consuming and detailed documentation. TDD entails a large number of simple unit tests that can function as documentation. Also, these unit tests illustrate how the code should operate.
e. **Decreases the necessity for debugging: When** there are fewer problems in the code, developers spend less time correcting them. In addition, errors are easier to detect, and developers are notified faster when anything breaks. This is one of the major benefits of the TDD process.

3.2 Unit testing

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.

Unit Testing Techniques

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

a. 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.

b. 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.

c. 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.

Best Practices in Unit Testing

Write tests during development, not after it

Unit tests are the first tests performed in the development cycle and are at the bottom of the testing pyramid. Hence, they perform best when they are run concurrently with development rather than afterward.

Setting up unit tests as soon as feasible encourages the production of clean code and the early detection of issues.

Writing tests at the end of development may result in untestable code; on the other hand, writing tests concurrently with production code allows us to evaluate both test and production code simultaneously, which helps developers understand the code better. It also makes the unit testing process more scalable and long-term.

In tests, don't use reasoning.
Using logical conditions and manual string concatenation in unit tests increases the likelihood of defects in your test suite. Instead of focusing on the implementation specifics, tests should focus on the intended result.

If you use conditions like if, while, switch, for, and so on, your tests will become less predictable and readable. If it appears that including logic in a test is unavoidable, divide it into two or more tests.

Avoid test interdependencies
Interdependencies between tests make them unstable and challenging to perform and debug. Test runners typically run numerous unit tests simultaneously without regard for order. To eliminate test interdependencies, each test case should have its own setup and takedown procedure.

3.3 UI testing

UI Testing, also known as GUI Testing is basically a mechanism meant to test the aspects of any software that a user will come into contact with. This usually means testing the visual elements to verify that they are functioning according to requirements – in terms of functionality and performance. UI testing ensures that UI functions are bug-free.

Websites comprise web elements created with CSS, JavaScript, and numerous other programming languages. UI testing performs tests and assertions of these elements to validate their efficacy. It is focused on examining visual and structural parts of the software i.e., parts the user would be concerned with, rather than the internal logic of the software.
UI Testing covers the gamut of visual indicators and graphic-based icons- toolbars, fonts, menus, text boxes, radio buttons, checkboxes, colors, and more.
Some of the features included in UI test suites include:

- Functionality
- Visual Design
- Responsiveness
- Performance
- Usability
- Accessibility
- Compliance

a. The Scope of UI Testing

Here are a few essential test cases that UI tests tend to verify:

- **Data type errors**: The test checks that only valid data can be entered for certain data fields such as dates, currency, etc.
- **Field widths**: The test checks that certain text fields do not allow the user to place inputs over a specific character limit.
- **Navigational elements**: The test checks that all navigational buttons on a page are working and that they redirect users to the right page.
- **Progress bars**: The test checks that when displaying pages or screens that take time to load completely, a progress bar appears to let the user know that the page is loading.
- **Type-ahead**: If the UI uses drop-down lists, type-ahead is required. In a drop-down menu with multiple options, the user should be able to find the right one by typing the first letter. Making the user go through a long list constitutes an unfavorable user experience.
- **Table scrolling**: If the website has data tables, and if the table extends into a second page, then the user should be able to scroll through all the data while keeping the headers visible and in place.
- **Error Logging**: This test checks that in case of a system error, the software records error details to a log file so that it can be reviewed later.
- **Menu Items**: The test checks that the software displays only the menu available in its particular geographical location (if that is applicable).
- **Working shortcuts**: If the software supports shortcuts, this test validates that each of them works as expected across multiple browsers, platforms, and devices.

b. Perform Manual UI Testing

In this case, a tester manually uses all the features of the website or app to check for any discrepancies. This makes sense when the software has a limited number of UI elements, which is usually the case in initial versions of a website or app. However, given the tech-savvy user base of our times, most expect software with rich, layered user interfaces with hundreds, perhaps thousands of UI elements that require verification.

Automated UI Testing

The advantages of automated testing are obvious. Tests are completed faster, which is a necessity in an industry where users expect top-notch software at lightning speed. Automated Selenium testing allows the software to be put through multiple test scenarios and for the same tests to be run repeatedly (with different variables, if necessary) quickly and correctly.

Additionally, automated tests are not prone to human error and exhaustion. As long as test scripts are written correctly and the right tools are in place, test results will be

accurate. Test automation frameworks are also often set up to automatically record results and share them with the team once tests are completed.

In the case of manual testing, the team has to not just complete the tests themselves but extract results, place them in reports, and share them with the right people themselves. Again, this becomes a demand on their time and effort. Automated Selenium testing is usually the best and most convenient option for UI testing. This is especially true of cross-browser testing.

c. Creating UI Test Scenarios

In order to perform comprehensive User Interface Testing, QA teams need to create a test plan that identifies the features of the app or website that must be tested. It also maps the resources available for testing so that bandwidth can be effectively used. By virtue of this data, the team can shape test scenarios, craft test cases, and write test scripts that address required issues.

Think of the test scenario as a document that defines how the application is likely to be used in the real world. For example, one scenario is that a user will successfully sign in with a valid username and password. For this step to be tested, the test script has to take into account the following possibilities:

- User enters a valid username and password
- User enters a valid username and invalid password
- User resets the password
- User tries to copy password from password field
- User tries to copy password to password field
- User clicks Help button

Self-Check - 3: Perform Test Driven Development (TDD)

1. What is TDD?

   Answer:
2. What are the main steps in TDD?

   Answer:
3. Why is TDD beneficial?

   Answer:
4. What does the 'red-green-refactor' cycle mean in TDD?

   Answer:
5. How does TDD influence code design?

   Answer:
6. What is unit testing?

   Answer:
7. Why are unit tests important?

   Answer:
8. What is mocking in unit testing?

   Answer:
9. What is the difference between unit tests and integration tests?

   Answer:
10. What is UI testing?

    Answer:
11. Why is UI testing important?

    Answer:
12. What tools are commonly used for UI testing?

    Answer:
13. What are some challenges of UI testing?

    Answer:
14. What is the difference between manual and automated UI testing?

    Answer

Answer Key - 3: Perform Test Driven Development (TDD)

1. What is TDD?

   Answer: TDD stands for Test-Driven Development, a software development process where tests are written before the code itself. It emphasizes writing a test, then producing the minimum amount of code to pass that test, and finally refactoring the code.

2. What are the main steps in TDD?

   Answer: The main steps in TDD are:
   - Write a failing test.
   - Write the minimum code to pass the test.
   - Refactor the code to improve it while ensuring that the tests still pass.

3. Why is TDD beneficial?

   Answer: TDD ensures that the code works as expected, helps to identify bugs early, promotes better design, and leads to more maintainable and cleaner code.

4. What does the 'red-green-refactor' cycle mean in TDD?

   Answer: 'Red' indicates writing a test that fails (because the feature isn't implemented yet), 'Green' means writing code to pass the test, and 'Refactor' is the step where the code is improved without changing its functionality.

5. How does TDD influence code design?

   Answer: TDD encourages writing only necessary code, leading to simpler, more modular designs. It also promotes decoupling and ensures each component is tested in isolation.

6. What is unit testing?

   Answer: Unit testing involves testing individual units or components of a software to ensure they work as intended. A unit is the smallest testable part of an application, like a function or method.

7. Why are unit tests important?

   Answer: Unit tests help catch bugs early, ensure that code changes don't break existing functionality, and facilitate code refactoring and maintenance.

8. What is mocking in unit testing?

   Answer: Mocking is the practice of replacing real objects with mock objects in unit tests. This allows for testing code in isolation by simulating the behavior of real objects.

9. What is the difference between unit tests and integration tests?

   Answer: Unit tests focus on individual components in isolation, while integration tests verify the interactions and integration between multiple components or systems.

10. What is UI testing?

    Answer: UI testing involves testing the graphical user interface of an application to ensure it behaves as expected from the user's perspective.

11. Why is UI testing important?

    Answer: UI testing is important because it ensures the application is user-friendly, accessible, and functions correctly from the user's viewpoint.

12. What tools are commonly used for UI testing?

    Answer: Common tools for UI testing include Selenium, Cypress, and TestComplete.

13. What are some challenges of UI testing?

    Answer: Challenges of UI testing include handling dynamic content, cross-browser compatibility, dealing with asynchronous events, and maintaining tests as the UI evolves.

14. What is the difference between manual and automated UI testing?

    Answer: Manual UI testing involves a human tester executing test cases, while automated UI testing uses scripts and tools to perform the tests, offering faster and more consistent results.

Job Sheet-3.1: Perform unit and UI Testing

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Collect the resources from your trainer as per the job requirement.
5. **Create a new test package**
    a. In the Android project, create a new package com.example.myfirstapp.test under the src/test directory.
    b. Create a new Kotlin file MainActivityTest.kt inside the package.
6. **Write unit tests**
    a. Add the following code to MainActivityTest.kt: Run the App and Show your work to the trainer.
7. Run unit tests
    a. Run the unit tests by executing ./gradlew test in the terminal.
    b. The tests should pass if everything is working correctly.
    c. UI Testing for Android App (Kotlin)
8. **Create a new test package**
    a. In the Android project, create a new package com.example.myfirstapp.ui.test under the src/androidTest directory.
    b. Create a new Kotlin file MainActivityUITest.kt inside the package.
9. **Write UI tests**
    a. Add the following code to MainActivityUITest.kt:
10. **Run UI tests**
    a. Run the UI tests by executing ./gradlew connectedAndroidTests in the terminal.
    b. The tests should pass if everything is working correctly.
11. **Create a new test file**
    a. Create a new file called test_server.py in the same directory as your backend server code.
12. **Write unit tests**
    a. Add the following code to test_server.py:
13. **Run unit tests**
    a. Run the unit tests by executing python test_server.py in your terminal.
    b. The tests should pass if everything is working correctly.
14. Clean your work place as per standard procedure.
15. Turn off the computer and clean your workplace.

Specification Sheet-3.1: Perform unit and UI Testing

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Learning Outcome 4: Introduce application signing and deployment

Assessment Criteria	1. Key store file to make signed APK is generated. 2. Google play console dashboard is used. 3. Application signing and deployment is introduced.
Conditions and Resources	• Actual workplace or training environment • CBLM • Handouts • Job related tools, equipment, and materials • Multimedia Projector • Paper, Pen, Pencil, and Eraser • Internet Facilities • Whiteboard and Marker
Contents	1. Key store file to make signed APK 2. Google play console dashboard 3. Application signing and deployment
Activities/job/Task	
Training Methods	• Blended • Discussion • Presentation • Demonstration • Guided Practice • Individual Practice • Project Work • Problem Solving • Brainstorming
Assessment Methods	Assessment methods may include but not limited to • Written Test • Demonstration • Oral Questioning

Learning Experience 4: Introduce application signing and deployment

In order to achieve the objectives stated in this learning guide, you must perform the learning steps below. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Activities	Recourses/Special Instructions
1. Trainee will ask the instructor about the learning materials	1. Instructor will provide the learning materials Introduce application signing and deployment
2. Read the Information sheet and complete the Self Checks & Check answer sheets on "Introduce application signing and deployment"	2. Read Information sheet 4: Introduce application signing and deployment 3. Answer Self-check 4: Introduce application signing and deployment 4. Check your answer with Answer key 4: Introduce application signing and deployment.
3. Read the Job/Task Sheet and Specification Sheet and perform job/Task	5. Job/Task Sheet and Specification Sheet **Job Sheet 4.1:** **Specification Sheet 4.1:**

Information Sheet 4: Introduce application signing and deployment

Learning Objective: After completion of this information sheet, the learners will be able to explain, define and interpret the following contents

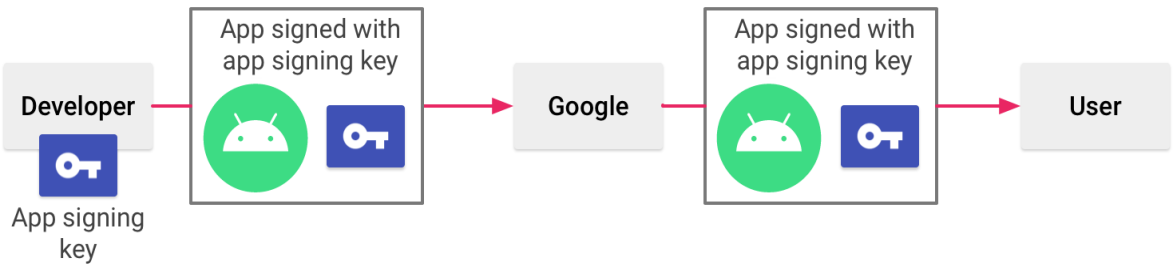
4.1 Key store file to make signed APK
4.2 Google play console dashboard
4.3 Application signing and deployment

4.1 Key store file to make signed APK

Signed APK generates a key and this key can be used to release versions of the app, so it is important to save this key which will be used when the next version of the app is released. The Android system needs that all installed applications be digitally signed with a certificate whose private key is owned by the application's developer. The Android system applies the certificate as a means of recognizing the author of an application and establishing trust relations between applications. The essential points to understand about signing Android applications are:

- When developers are ready to publish the android application for end-users, they are required to sign it with a suitable private key. They can't publish an application that is signed with the debug key generated by the SDK tools.
- Applications can only be installed when they are signed. Android does not allow unsigned applications to get installed.
- Developers can apply self-signed certificates to sign the application. No certificate authority is required
- To test and debug the application, the build tools sign the application with a special debug key that is created by the Android SDK build tools.
- The system will test a signer certificate's expiration date only at install time. If an application's signer license expires after the application is installed, the application will continue to operate normally.

Importance of Signed APK



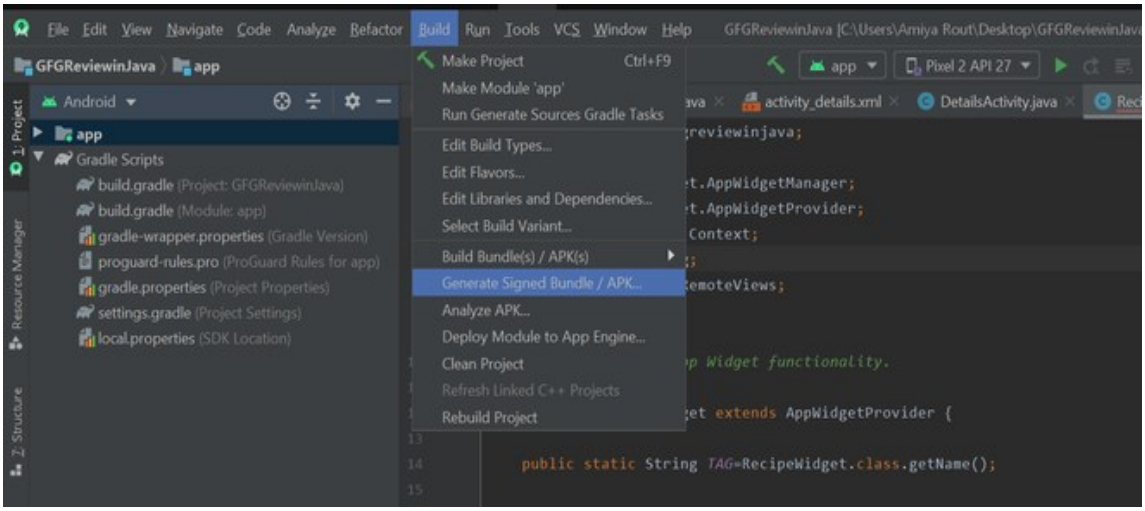
Application Modularity: The applications signed by the same process are recognized as a single application and are allowed to run on the same process. This allows the developers to make the application in modules and so users can update each module independently.

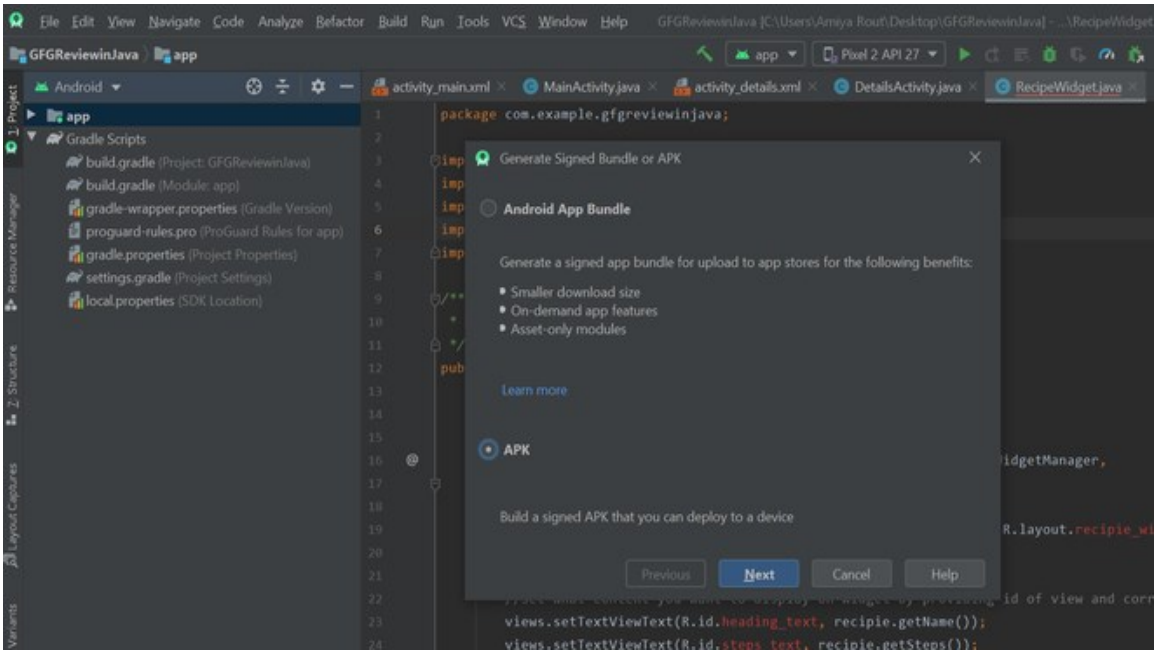
Application Upgrade: To update the application, the updates must be signed with the same certificates. When the system is installing an update to an application, it relates the certificate in the new version with those in the actual version. If the certificates match correctly, including both the certificate data and order, then the system releases the update. If we sign the new version without using matching certificates, we need to attach a different package name to the application and in this case, the user installs the new version as a completely new application.

Code/Data Sharing Through Permissions: To allow the applications to use different resources, Android system executes signature-based permissions enforcement so that the application can exhibit functionality to another application which is signed with a specified certificate. By signing various applications with the same certificate and working with signature-based permissions checks, your applications can yield code and data in a secure manner.

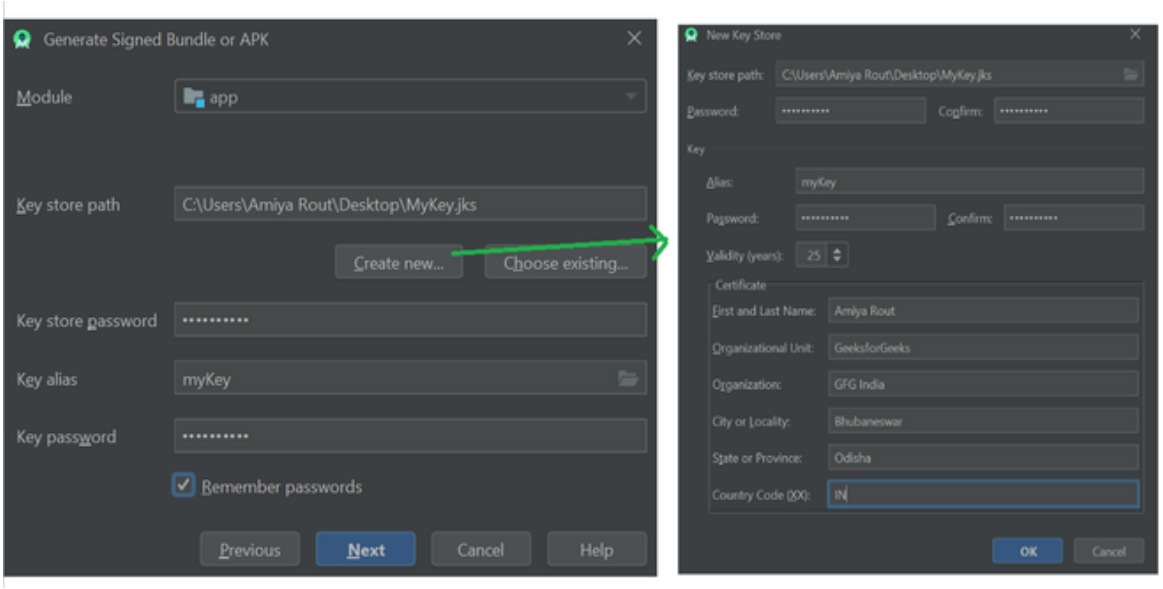
Generating Signed APK in Android Studio

Step 1: Go to Build -> Generate Signed Bundle or APK, a pop up will arise. Choose APK in the pop-up and click on Next.

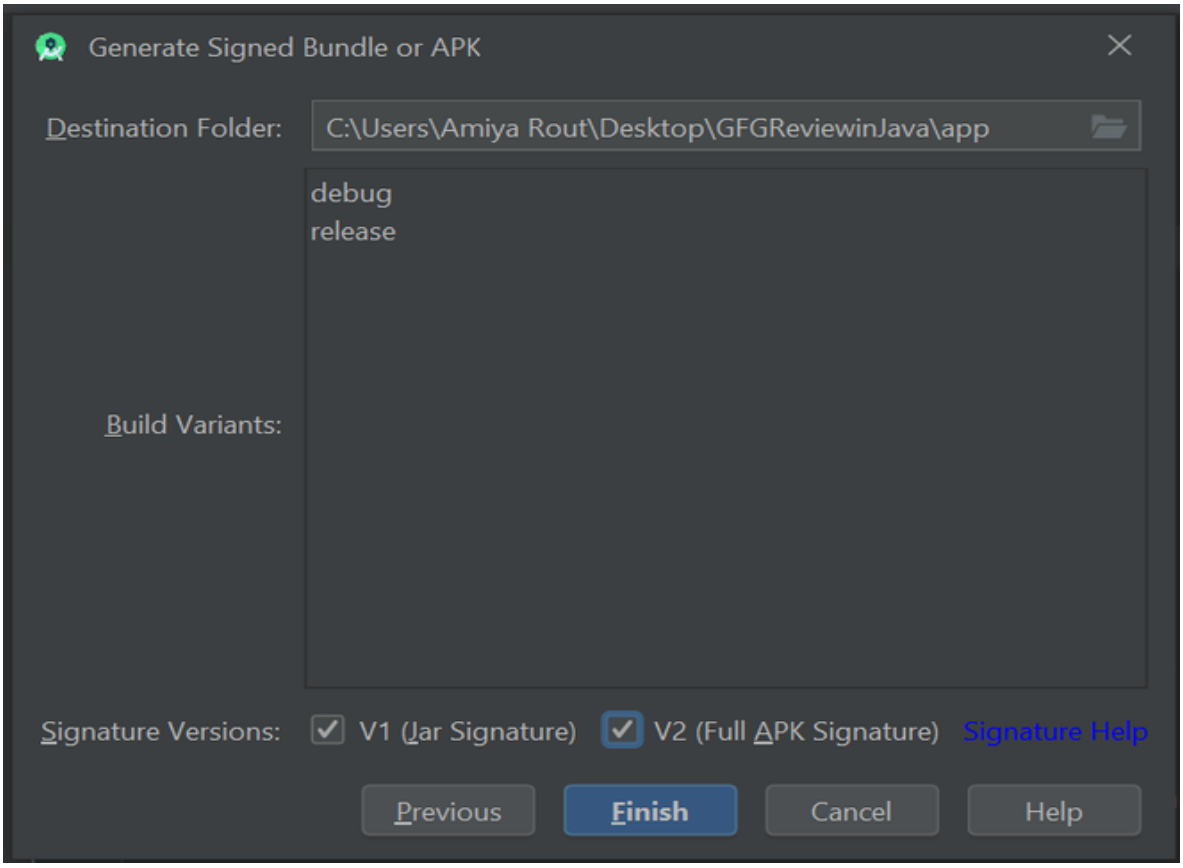




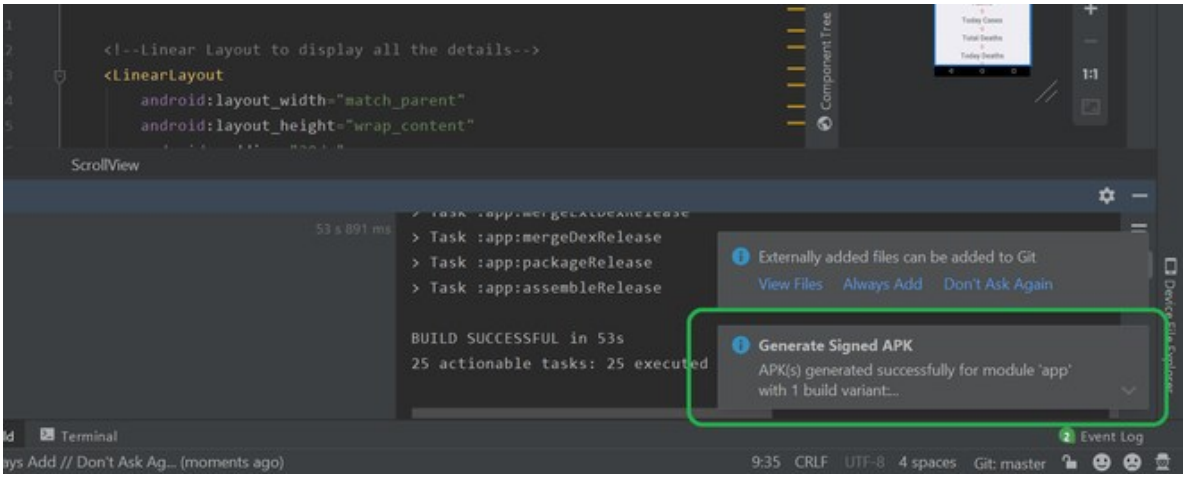
Step 2: After completing step 1, if you already have a key, make sure you just refer to that and you can release the version of that app but in case if it is the first time it is recommended to create a new key. After creating a new key click on OK and then click on Next**.**



Step 3: After clicking Next in the next pop-up make sure to choose release as Build Variants and check the two Signature Versions as well. Then click on Finish.



Step 4: After successfully completed these steps you can locate your signed apk at app -> release -> app-release as your apk file.



Summer-time is here and so is the time to skill-up! More than 5,000 learners have now completed their journey from **basics of DSA to advanced level development programs** such as Full-Stack, Backend Development, Data Science.

Manage your own signing key

If you choose not to opt in to Play App Signing (only for apps created before August 2021), you can manage your own app signing key and keystore. Keep in mind, you are responsible for securing the key and the keystore. Additionally, your app will not be able to support Android App Bundles, Play Feature Delivery and Play Asset Delivery.

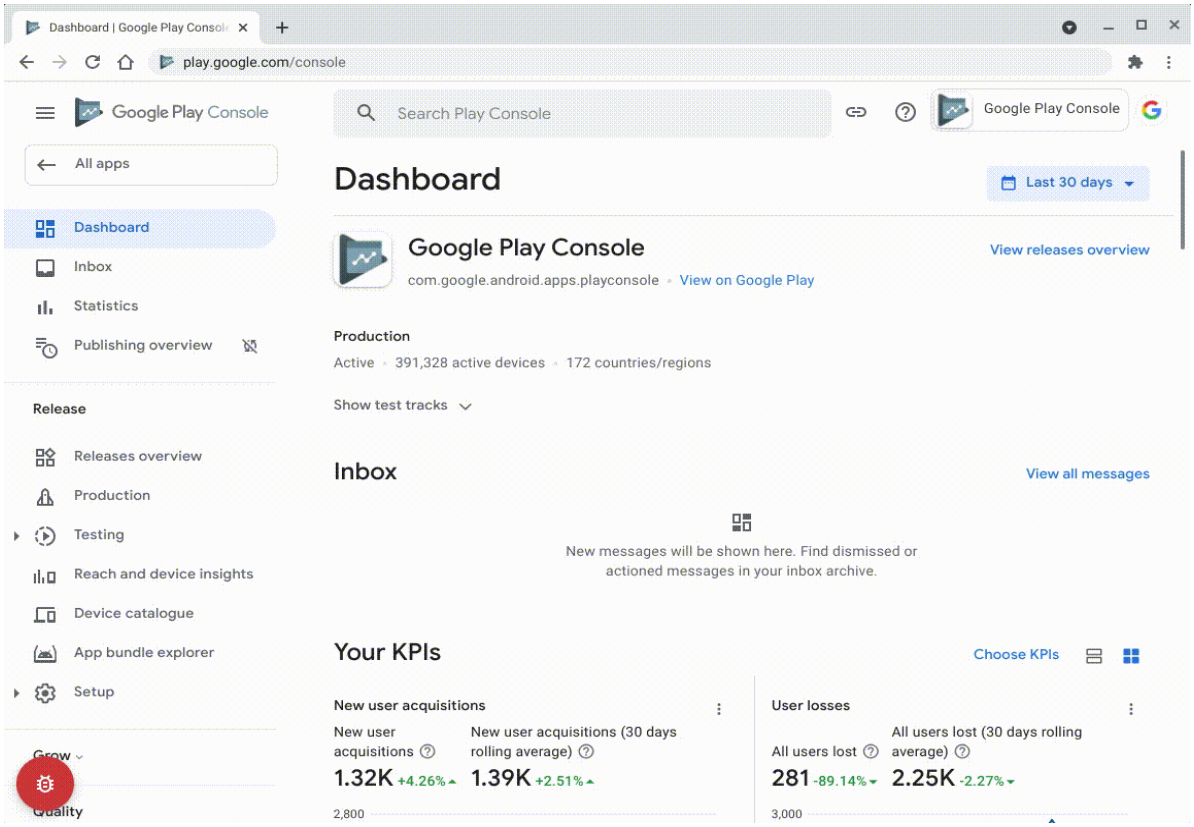
When you are ready to create your own key and keystore, make sure you first choose a strong password for your keystore and a separate strong password for each private key stored in the keystore. You must keep your keystore in a safe and secure place. If you lose access to your app signing key or your key is compromised, Google cannot retrieve the app signing key for you, and you will not be able to release new versions of your app to users as updates to the original app. For more information, see Keep your key secure, below.

4.2 Google play console dashboard

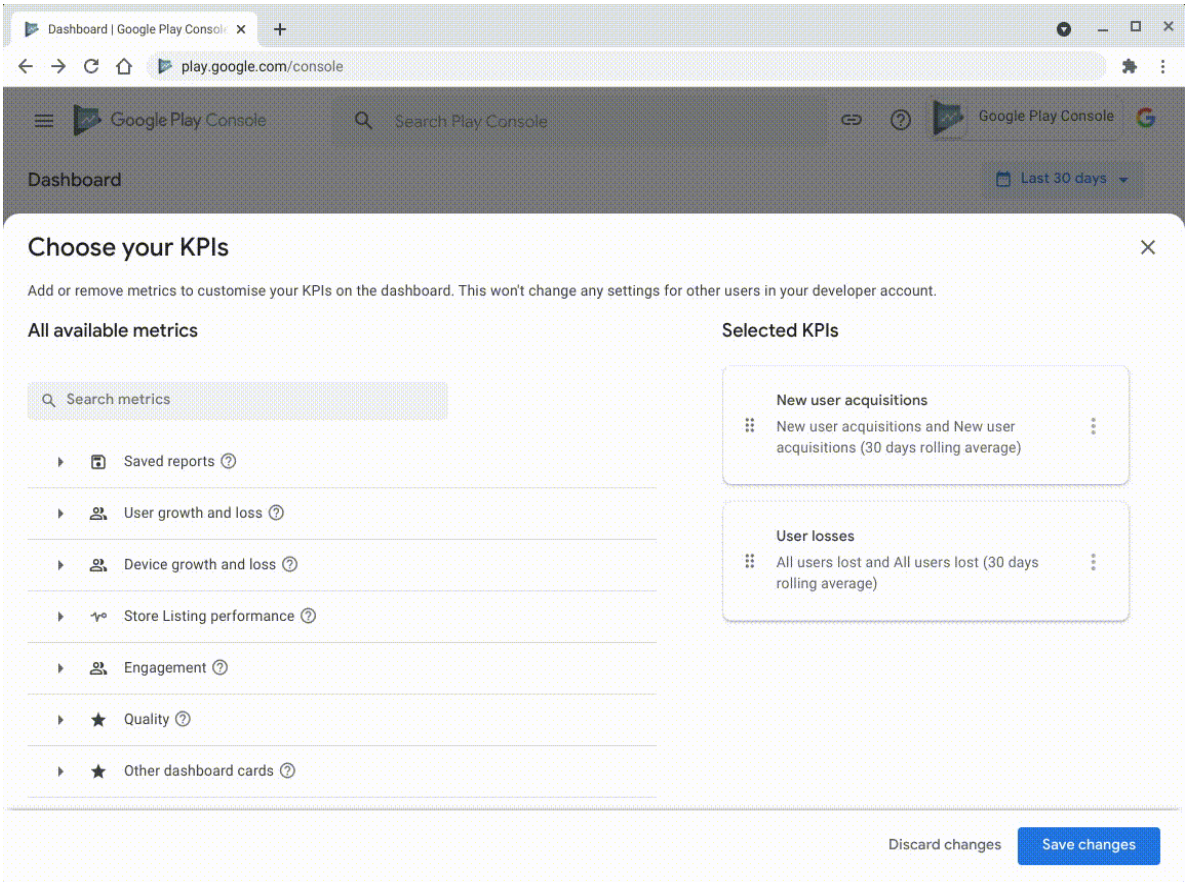
Google Play Console metrics can help you understand your app's performance across growth and acquisition, engagement and monetization, quality, and churn. But with dozens of metrics — and thousands of variations — we know not every metric is relevant to every person. One of the challenges you've shared with us is that it can sometimes be difficult to find exactly the metrics that you need for your personal job role, and to access them quickly and regularly once you have found them.

That's why today, we're pleased to announce that **you can now customize and pin the precise metrics that matter to you in a personalized KPIs section at the top of your app dashboard**. These customizations are unique to you, so you can configure your KPIs however you want without affecting the rest of your team.

Getting started is easy. On the dashboard for any app, scroll down to the KPI section and select “choose KPIs.” You can either build your own or start with suggested KPIs for job specialties, such as Growth, Quality and Health, or Monetization.

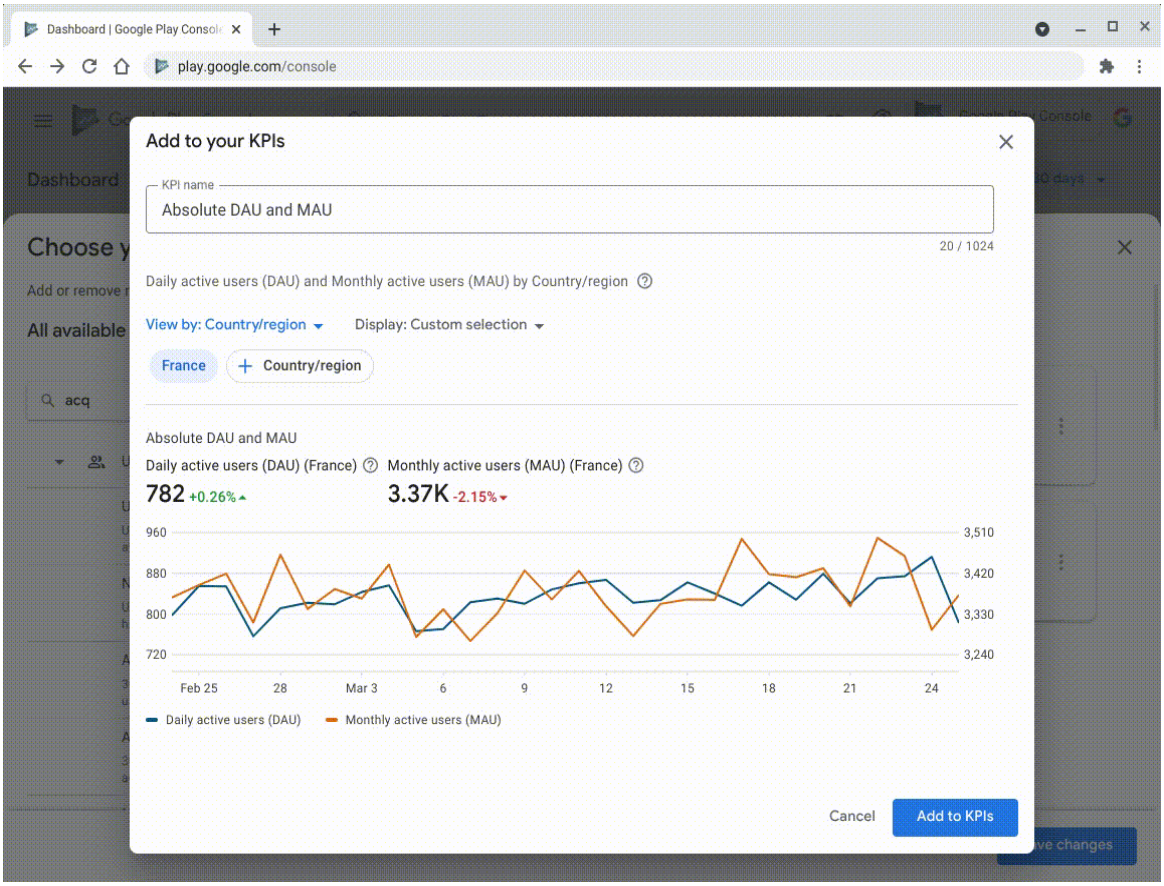


There is an extensive list of available metrics, including our new engagement data and peer comparisons. Search filters make it easy to find just what you want, and once selected you can edit the dimensions and filters to suit your exact needs. For instance, you could display Daily Active Users for your top-five languages; or if you are a country manager, only show revenue from a specific country or territory.



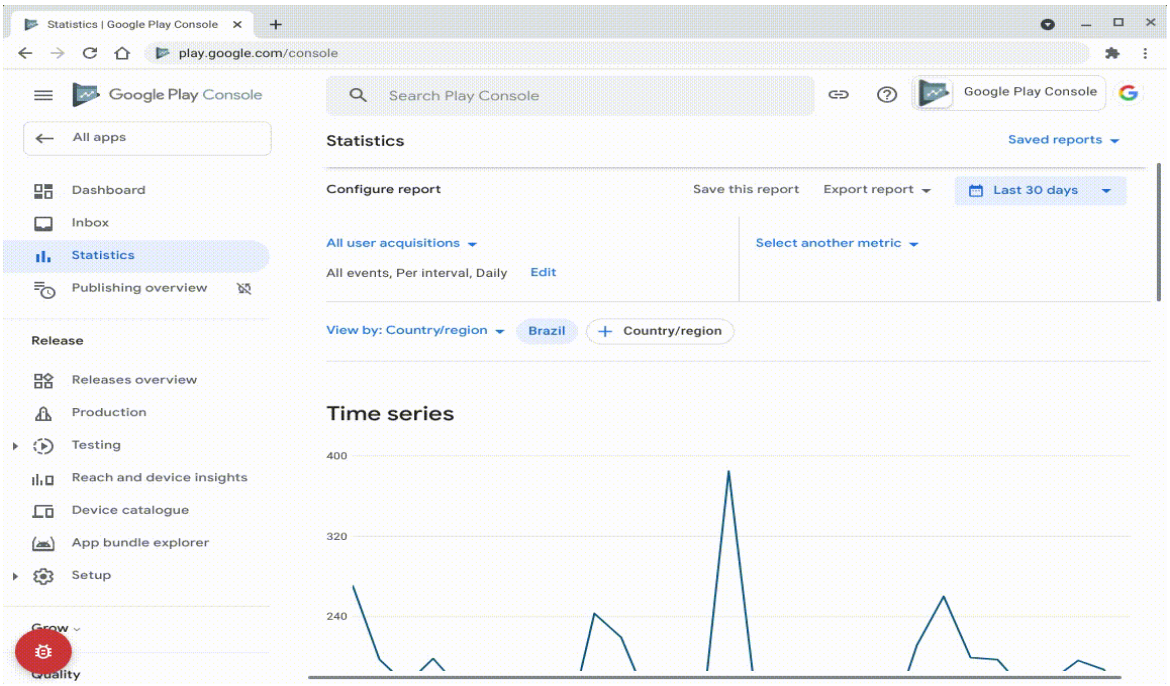
You can name any of your KPIs to make them easy to remember, and even include

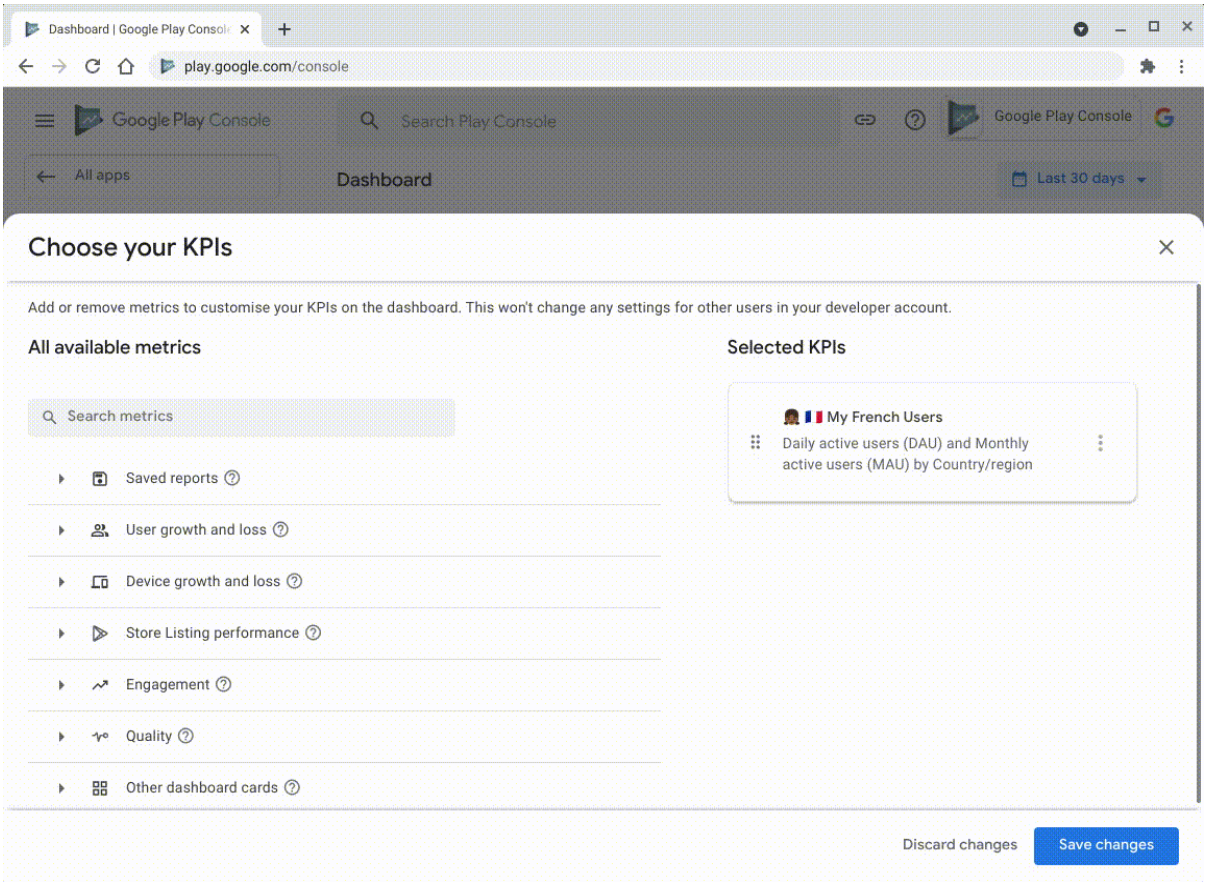
Once you've configured a list of KPIs that suit you, you can order them to control where they appear. This way, you can make sure that your most important metrics are always first to be seen.



You can include up to 20 KPIs, so your dashboard can be as robust or as streamlined as you want.

In addition to our suggested metrics, you can also pin any other dashboard card to your KPIs. For even finer control, you can even add any reports you've saved from the Statistics page. This allows you to create hyper-specific custom KPI cards from any of our core metrics.





There are as many ways to customize your KPIs as there are people using the console. Instead of seeing default KPIs, now everyone can have a customized dashboard showing exactly the metrics that matter to their individual workflows.

4.3 Application signing and deployment

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.

On Google Play, application signing bridges the trust Google has with the developer and the trust the developer has with their application. Developers know their application is provided, unmodified, to the Android device; and developers can be held accountable for behavior of their application.

On Android, application signing is the first step to placing an application in its Application Sandbox. The signed application certificate defines which user ID is associated with which application; different applications run under different user IDs. Application signing ensures that one application cannot access any other application except through well-defined IPC. When an application (APK file) is installed onto an Android device, the Package Manager verifies that the APK has been properly signed with the certificate included in that APK. If the certificate (or, more accurately, the public key in the certificate) matches the key used

to sign any other APK on the device, the new APK has the option to specify in the manifest that it will share a UID with the other similarly-signed APKs.

Applications can be signed by a third-party (OEM, operator, alternative market) or self-signed. Android provides code signing using self-signed certificates that developers can generate without external assistance or permission. Applications do not have to be signed by a central authority. Android currently does not perform CA verification for application certificates.

Applications are also able to declare security permissions at the Signature protection level, restricting access only to applications signed with the same key while maintaining distinct UIDs and Application Sandboxes. A closer relationship with a shared Application Sandbox is allowed using the shared UID feature where two or more applications signed with same developer key can declare a shared UID in their manifest.

APK signing schemes
Android supports three application signing schemes:

- v1 scheme: based on JAR signing
- v2 scheme: APK Signature Scheme v2, which was introduced in Android 7.0.
- v3 scheme: APK Signature Scheme v3, which was introduced in Android 9.

For maximum compatibility, sign applications with all schemes, first with v1, then v2, and then v3. Android 7.0+ and newer devices install apps signed with v2+ schemes more quickly than those signed only with v1 scheme. Older Android platforms ignore v2+ signatures and thus need apps to contain v1 signatures.
AR signing (v1 scheme)

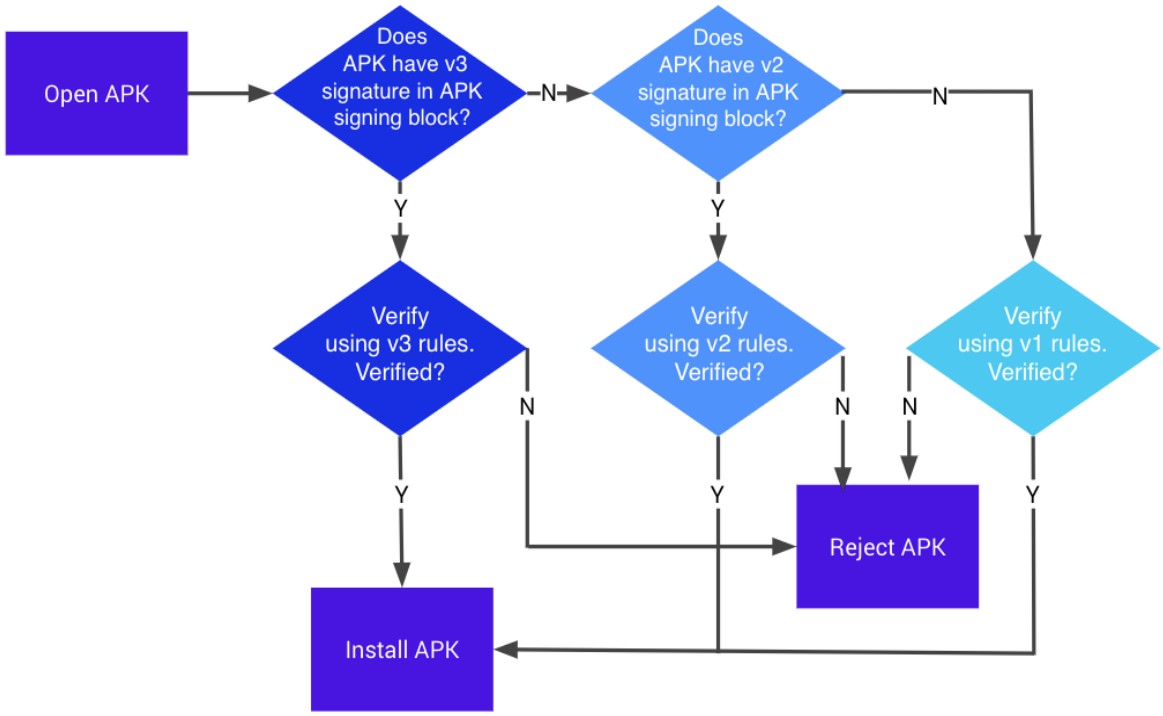
APK signing has been a part of Android from the beginning. It is based on signed JAR. For details on using this scheme, see the Android Studio documentation on Signing your app.

v1 signatures do not protect some parts of the APK, such as ZIP metadata. The APK verifier needs to process lots of untrusted (not yet verified) data structures and then discard data not covered by the signatures. This offers a sizeable attack surface. Moreover, the APK verifier must uncompress all compressed entries, consuming more time and memory. To address these issues, Android 7.0 introduced APK Signature Scheme v2.

APK Signature Scheme v2 & v3 (v2+ scheme)
Devices running Android 7.0 and later support APK signature scheme v2 (v2 scheme) and later. (v2 scheme was updated to v3 in Android 9 to include additional information in the signing block, but otherwise works the same.) The contents of the APK are hashed and signed, then the resulting APK Signing Block is inserted into the APK. For details on applying the v2+ scheme to an app, see APK Signature Scheme v2.

During validation, v2+ scheme treats the APK file as a blob and performs signature checking across the entire file. Any modification to the APK, including ZIP metadata modifications, invalidates the APK signature. This form of APK verification is substantially faster and enables detection of more classes of unauthorized modifications.

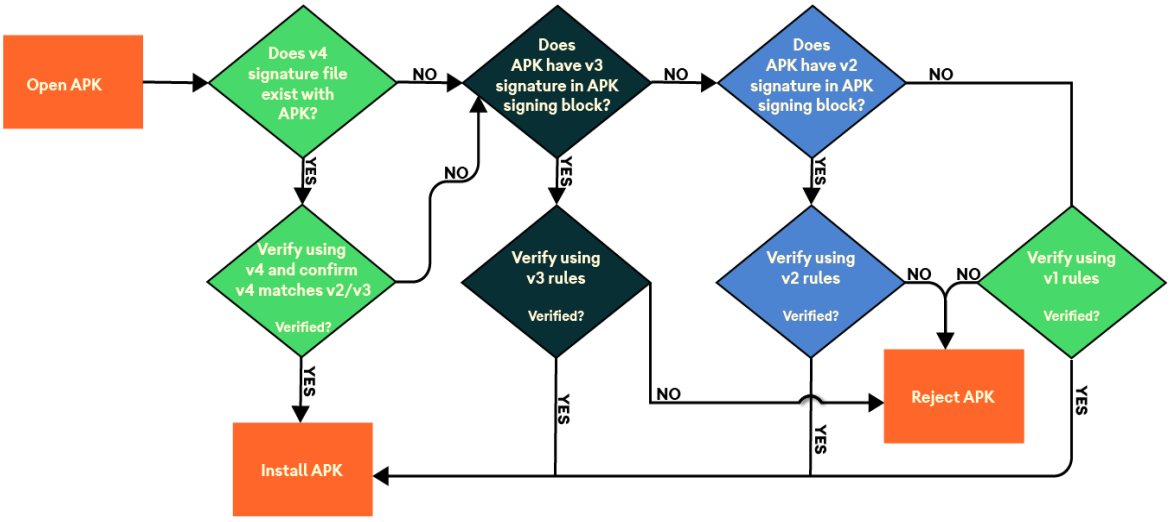
The new format is backwards compatible, so APKs signed with the new signature format can be installed on older Android devices (which simply ignore the extra data added to the APK), as long as these APKs are also v1-signed.



Whole-file hash of the APK is verified against the v2+ signature stored in the APK Signing Block. The hash covers everything except the APK Signing Block, which contains the v2+ signature. Any modification to the APK outside of the APK Signing Block invalidates the APK's v2+ signature. APKs with stripped v2+ signature are rejected as well, because their v1 signature specifies that the APK was v2-signed, which makes Android 7.0 and newer refuse to verify APKs using their v1 signatures.

APK Signature Scheme v4

Android 11 supports a streaming-compatible signing scheme with the APK Signature Scheme v4. The v4 signature is based on the Merkle hash tree calculated over all bytes of the APK. It follows the structure of the fs-verity hash tree exactly (for example, zero-padding the salt and zero-padding the last block). Android 11 stores the signature in a separate file, <apk name>.apk.idsigA v4 signature requires a complementary v2 or v3 signature.


Open APK
Does v4 signature file exist with APK?
NO
Does APK have v3 signature in APK signing block?
NO
Does APK have v2 signature in APK signing block?
NO
YES
YES
YES
NO
Verify using v4 and confirm v4 matches v2/v3
Verified?
Verify using v3 rules
Verified?
Verify using v2 rules
Verified?
NO
NO
Verify using v1 rules
Verified?
YES
YES
NO
YES
YES
Reject APK
Install APK

Self-Check - 4: Introduce application signing and deployment

1. What is the purpose of signing an APK?**

   Answer:
2. What is a keystore?**

   Answer:
3. What is a private key?**

   Answer:
4. What is a certificate?**

   Answer:
5. How many keystores can I have?**

   Answer:
6. Where do I store my keystore?**

   Answer:
7. Can I share my keystore with someone else?**

   Answer:
8. How do I generate a keystore?**

   Answer:
9. How do I export my keystore from Android Studio?**

   Answer:
10. Why do I need to upload my APK to the Google Play Console?**

    Answer:
11. How do I upload my APK to the Google Play Console?**

    Answer:
12. Can I upload multiple APKs at once?**

    Answer:
13. Can I roll back a previous version of my app?**

    Answer:
14. How do I track analytics for my app?**

    Answer:
15. How do I troubleshoot issues with my app on the Google Play Store?**

    Answer:

Answer Key - 4: Introduce application signing and deployment

1. What is the purpose of signing an APK?**
   Answer: To verify the identity of the developer and ensure the app is legitimate.

2. What is a keystore?**
   Answer: A keystore is a file that contains the private key and certificate used to sign an APK.

3. What is a private key?**
   Answer: A private key is a unique string used to sign an APK, ensuring its integrity and authenticity.

4. What is a certificate?**
   Answer: A certificate is a digital document that verifies the identity of the developer and associates it with the private key.

5. How many keystores can I have?**
   Answer: You can have multiple keystores, but each APK must be signed with a unique private key.

6. Where do I store my keystore?**
   Answer: You can store your keystore on your local machine or in a secure location, such as Google Cloud Storage.

7. Can I share my keystore with someone else?**
   Answer: No, it's recommended to keep your keystore secure and not share it with anyone, as it compromises the security of your app.

8. How do I generate a keystore?**
   Answer: You can generate a keystore using tools like KeyStore Explorer or Android Studio.

9. How do I export my keystore from Android Studio?**
   Answer: In Android Studio, go to "Build" > "Generate Signed Bundle/APK" and select "Export" to export your keystore.

10. Why do I need to upload my APK to the Google Play Console?**
    Answer: To distribute your app to the Google Play Store, you need to upload your signed APK to the console.

11. How do I upload my APK to the Google Play Console?**
    Answer: Log in to your Google Play Console account, go to "Release management" > "Production" > "Create release" and upload your signed APK.

12. Can I upload multiple APKs at once?**
    Answer: No, you can only upload one APK per release.

13. Can I roll back a previous version of my app?**
    Answer: Yes, you can roll back a previous version of your app by creating a new release and uploading the previous APK.

14. How do I track analytics for my app?**
    Answer: You can track analytics for your app using tools like Google Analytics or Firebase Analytics within the Google Play Console.

15. How do I troubleshoot issues with my app on the Google Play Store?**
    Answer: Use the Google Play Console's "Crash reports" and "Issue reports" sections to troubleshoot issues with your app.

Job Sheet-4.1: Generating Signed APK in Android Studio

Working Procedure/ Steps:

1. Follow OSH and use Personal Protective Equipment (PPE).
2. Check Electricity & Internet Connections to your Computer.
3. Start the Computer.
4. Collect the resources from your trainer as per the job requirement.
5. Create a keystore
6. Open Android Studio and navigate to File
7. Click on Build, Execution, Deployment > Gradle > Android Studio.
8. Fill in the required information:
    a. Name: Your name
    b. Organization: Your organization
    c. Email: Your email
    d. Password: A strong password
    e. Confirm password: Retype your password
    f. Click OK to create the keystore.
9. Configure signing settings
10. Open your project's build.gradle file and add the following code:
11. Generate a signed APK
12. Open the terminal in Android Studio by navigating
13. Run the following command to generate a signed APK:
14. Generate a signed APK
15. Open the terminal in Android Studio by navigating
16. Run the following command to generate a signed APK:
17. Generate a signed APK
18. Open the terminal in Android Studio by navigating.
19. Run the following command to generate a signed APK:
20. Run the App and Show your work to the trainer.
21. Clean your work place as per standard procedure.
22. Turn off the computer and clean your workplace.

Specification Sheet-4.1: Generating Signed APK in Android Studio

Conditions for the job: Work must be carried out in a safe manner and according to relevant competency standards.

List of required PPE

S/N	Name of PPE	Specification	Unit	Required Quantity
01	Ergonomic Chair	Wood and foam	No	1
02	Eye protective glass	Metal and Glass	No	1
03	Mask	Surgical mask	1	1

List of required Software

S/N	Name of Materials	Specification	Unit	Required Quantity
01	IntelliJ IDEA	Latest version	…	1
02	Android Studio	Latest version	…	1
03	Java	JDK Latest version	…	1

List of required Tools & Equipment's

S/N	Name	Specification	Unit	Required Quantity
01	Personal Computer or Laptop	Minimum Core i5 7th gen Processor	No	1
04	Internet connection		…	1

Assessment Criteria	1. Encapsulation is described. 2. Language mechanism is explained for restricting access to object component. 3. Language construction is explained which facilitates bundling of data. 4. Association relationship is defined. 5. Association relationship between two classes is created. 6. Data and its functionality are encapsulated.
Conditions and Resources	1. Actual workplace or training environment 2. CBLM 3. Handouts 4. Job related tools, equipment, and materials 5. Multimedia Projector 6. Paper, Pen, Pencil, and Eraser 7. Internet Facilities 8. • Whiteboard and Marker Whiteboard and Marker
Contents	1. Encapsulation 2. Language mechanism 3. Language construction 4. Association relationship o One-to-one o One-to-many o Many-to-many 5. Data and its functionality
Activities/job/Task	1. Create two Kotlin classes, Student and Course. Establish an association relationship between them, where a student can be associated with multiple courses, and a course can have multiple students. 2. Design a Kotlin class representing a Bank Account. Encapsulate the account balance as a private field and provide public methods for deposit, withdrawal, and checking the balance. Ensure proper encapsulation and validation of transactions.
Training Methods	1. Blended 2. Discussion 3. Presentation 4. Demonstration 5. Guided Practice 6. Individual Practice 7. Project Work 8. Problem Solving

	9. Brainstorming
Assessment Methods	Assessment methods may include but not limited to 1. Written Test 2. Demonstration 3. Oral Questioning

Assessment Criteria	7. Encapsulation is described. 8. Language mechanism is explained for restricting access to object component. 9. Language construction is explained which facilitates bundling of data. 10. Association relationship is defined. 11. Association relationship between two classes is created. 12. Data and its functionality are encapsulated.
Conditions and Resources	9. Actual workplace or training environment 10. CBLM 11. Handouts 12. Job related tools, equipment, and materials 13. Multimedia Projector 14. Paper, Pen, Pencil, and Eraser 15. Internet Facilities 16. • Whiteboard and Marker Whiteboard and Marker
Contents	6. Encapsulation 7. Language mechanism 8. Language construction 9. Association relationship o One-to-one o One-to-many o Many-to-many 10. Data and its functionality
Activities/job/Task	3. Create two Kotlin classes, Student and Course. Establish an association relationship between them, where a student can be associated with multiple courses, and a course can have multiple students. 4. Design a Kotlin class representing a Bank Account. Encapsulate the account balance as a private field and provide public methods for deposit, withdrawal, and checking the balance. Ensure proper encapsulation and validation of transactions.
Training Methods	10. Blended 11. Discussion 12. Presentation 13. Demonstration 14. Guided Practice 15. Individual Practice 16. Project Work 17. Problem Solving

	18. Brainstorming
Assessment Methods	Assessment methods may include but not limited to 4. Written Test 5. Demonstration 6. Oral Questioning

Assessment Criteria	13. Encapsulation is described. 14. Language mechanism is explained for restricting access to object component. 15. Language construction is explained which facilitates bundling of data. 16. Association relationship is defined. 17. Association relationship between two classes is created. 18. Data and its functionality are encapsulated.
Conditions and Resources	17. Actual workplace or training environment 18. CBLM 19. Handouts 20. Job related tools, equipment, and materials 21. Multimedia Projector 22. Paper, Pen, Pencil, and Eraser 23. Internet Facilities 24. • Whiteboard and Marker Whiteboard and Marker
Contents	11. Encapsulation 12. Language mechanism 13. Language construction 14. Association relationship o One-to-one o One-to-many o Many-to-many 15. Data and its functionality
Activities/job/Task	5. Create two Kotlin classes, Student and Course. Establish an association relationship between them, where a student can be associated with multiple courses, and a course can have multiple students. 6. Design a Kotlin class representing a Bank Account. Encapsulate the account balance as a private field and provide public methods for deposit, withdrawal, and checking the balance. Ensure proper encapsulation and validation of transactions.
Training Methods	19. Blended 20. Discussion 21. Presentation 22. Demonstration 23. Guided Practice 24. Individual Practice 25. Project Work 26. Problem Solving

	27. Brainstorming
Assessment Methods	Assessment methods may include but not limited to 7. Written Test 8. Demonstration 9. Oral Questioning

Assessment Criteria	19. Encapsulation is described. 20. Language mechanism is explained for restricting access to object component. 21. Language construction is explained which facilitates bundling of data. 22. Association relationship is defined. 23. Association relationship between two classes is created. 24. Data and its functionality are encapsulated.
Conditions and Resources	25. Actual workplace or training environment 26. CBLM 27. Handouts 28. Job related tools, equipment, and materials 29. Multimedia Projector 30. Paper, Pen, Pencil, and Eraser 31. Internet Facilities 32. • Whiteboard and Marker Whiteboard and Marker
Contents	16. Encapsulation 17. Language mechanism 18. Language construction 19. Association relationship o One-to-one o One-to-many o Many-to-many 20. Data and its functionality
Activities/job/Task	7. Create two Kotlin classes, Student and Course. Establish an association relationship between them, where a student can be associated with multiple courses, and a course can have multiple students. 8. Design a Kotlin class representing a Bank Account. Encapsulate the account balance as a private field and provide public methods for deposit, withdrawal, and checking the balance. Ensure proper encapsulation and validation of transactions.
Training Methods	28. Blended 29. Discussion 30. Presentation 31. Demonstration 32. Guided Practice 33. Individual Practice 34. Project Work 35. Problem Solving

	36. Brainstorming
Assessment Methods	Assessment methods may include but not limited to 10. Written Test 11. Demonstration 12. Oral Questioning

Review of Competency

Below is yourself assessment rating for module "**Implementing Background Service and Application Deployment"**

Assessment of performance Criteria	Yes	No
Event receiving is explained.		
An event is received.		
A service is started using broadcast receiver.		
Lifecycle of services is explained.		
Different types of services are implemented.		
Notification using service is generated.		
Music playing as a background service is made.		
TDD is explained.		
Unit testing is performed.		
UI testing is performed.		
Key store file to make signed APK is generated.		
Google play console dashboard is used.		
Application signing and deployment is introduced.		

I now feel ready to undertake my formal competency assessment.

Signed:

Date:

Development of CBLM

The Competency based Learning Material (CBLM) of **"Implementing Background Service and Application Deployment" (Occupation: Android Mobile Application Development, Level-4)** for National Skills Certificate is developed by NSDA with the assistance of SIMEC System Ltd., ECF Consultancy & SIMEC Institute of Technology JV (Joint Venture Firm) in the month of July, 2024 under the contract number of package SD-9B dated 15th January 2024.

SL No.	Name & Address	Designation	Contact Number
1	Engr. Md. Zuwel Parves	Writer	01737-278906
2	Md. Abdul Al Hossain	Editor	01778-926438
3	Engr Md. Zuwel Parves	Co-Ordinator	01737-278906
4	Md. Abdur Razzaque	Reviewer	01713-304824

Reference

1. https://reflectoring.io/kotlin-design-patterns/
2. https://www.w3schools.com/kotlin/index.php
3. https://www.tutorialspoint.com/kotlin/index.htm
4. https://doc.qt.io/qt-5/qtandroidextras-services-servicebroadcast-example.html
5. https://www.browserstack.com/guide/tdd-in-android