



DOP: / /2023

DOS: / /2023

Experiment No: 01

Title: To install and configure Android Studio / Genymotion and Implement simple Android apk.

Theory:

- **What is Android:**

Android is a software package and linux based operating system for mobile devices such as tablet computers and smartphones. It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

The goal of android project is to create a successful real-world product that improves the mobile experience for end users. There are many code names of android such as Lollipop, KitKat, Jelly Bean, Ice cream Sandwich, Froyo, Eclair, Donut etc .

Features of Android:

The important features of android are given below:

- 1) It is open-source.
- 2) Anyone can customize the Android Platform.
- 3) There are a lot of mobile applications that can be chosen by the consumer.
- 4) It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

How to Install Android Studio?

Step 1 - System Requirements

You will be delighted, to know that you can start your Android application development on either of the following operating systems–

- Microsoft® Windows® (32 or 64-bit).
- Mac® OS X® 10.8.5 or higher, up to 10.9 (Mavericks)
- GNOME or KDE desktop

Second point is that all the required tools to develop Android applications are open source and can be downloaded from the Web. Following is the list of software's you will need before you start your Android application programming.

- Java JDK5 or later version
- Java Runtime Environment (JRE) 6
- Android Studio

Step 2 - Setup Android Studio:

Android Studio

Android Studio is the official integrated development environment for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems.

Installation

Step 1: Head over to <https://developer.android.com/studio#downloads> link to get the Android Studio executable or zip file.

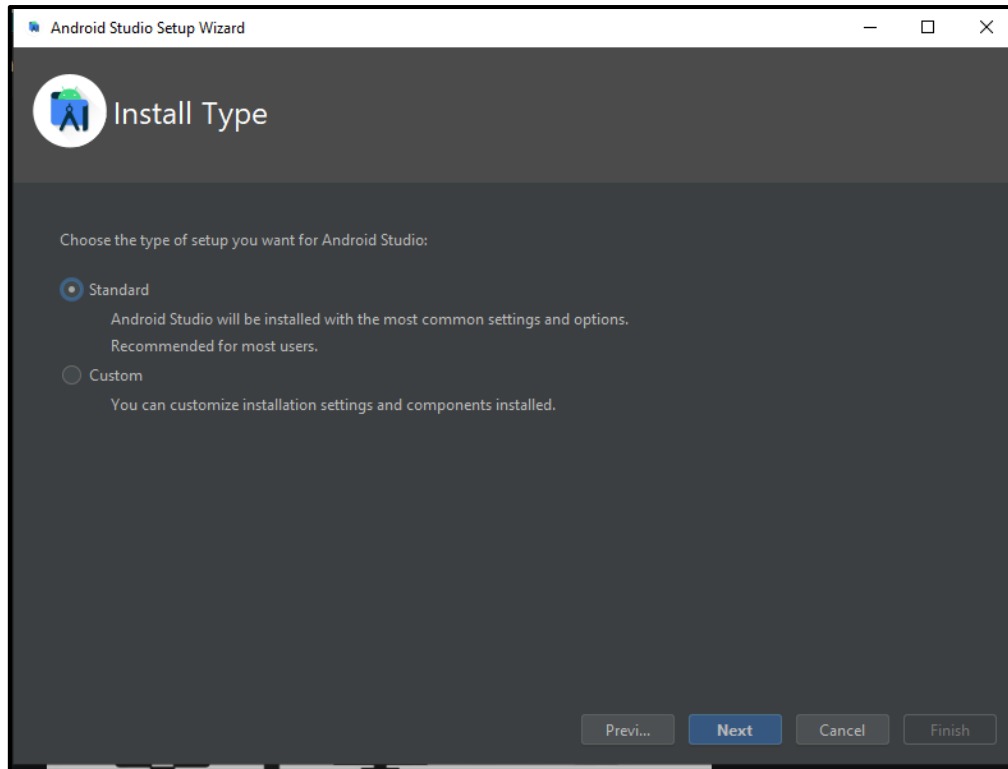
Step 2: Click on the Download Android Studio Button.



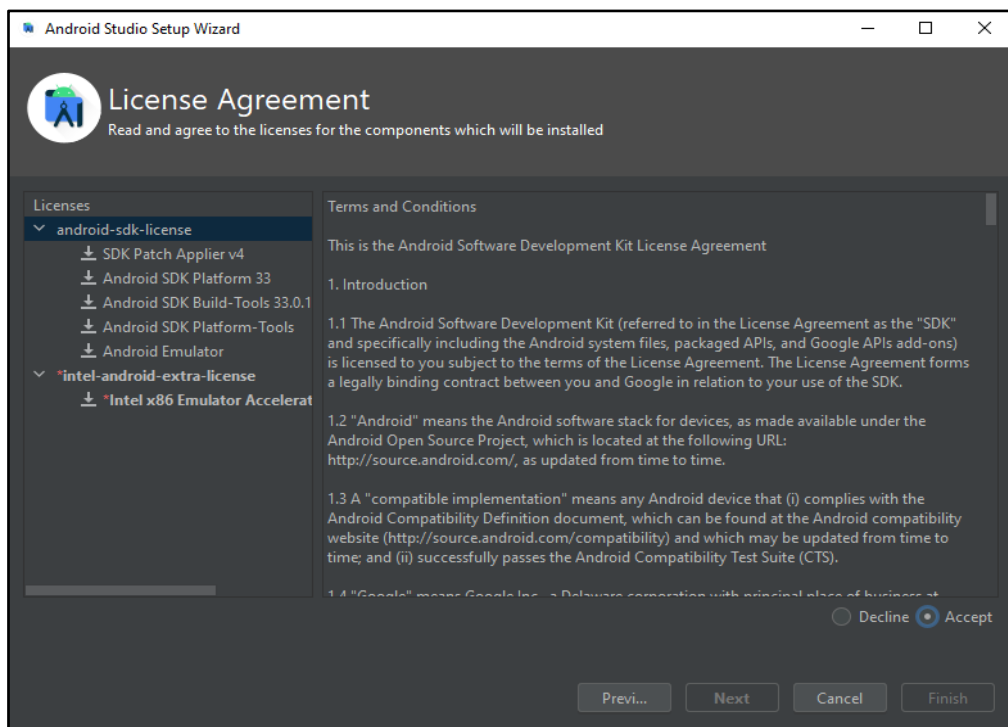
Step 3: After the downloading has finished, open the file from downloads and run it.

Step 4: Once "Finish" is clicked, it will ask whether the previous settings need to be imported [if the android studio had been installed earlier], or not. It is better to choose the 'Don't import Settings' option.

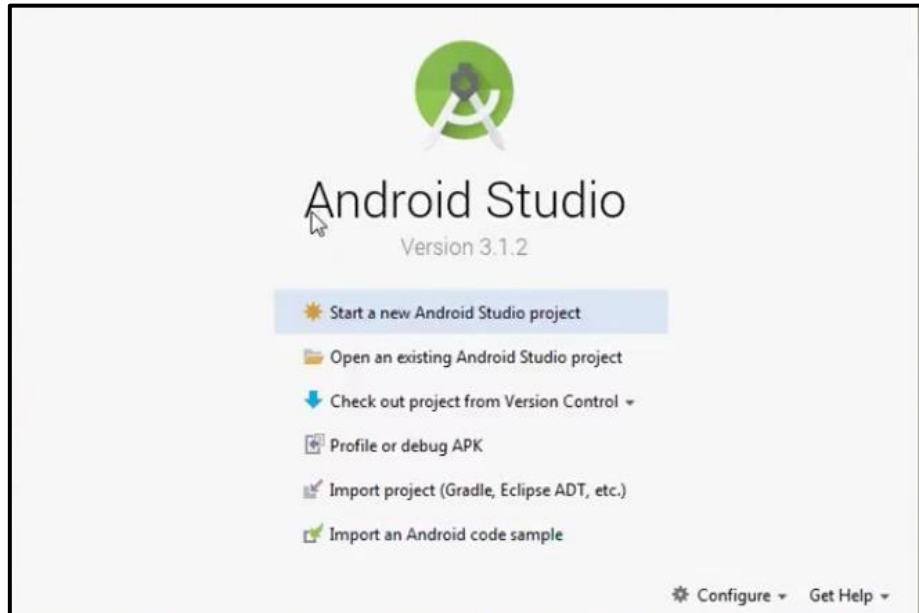
Step 5: After it has found the SDK components, it will redirect to the Welcome dialog box.



Step 6: Now it is time to download the SDK components.



Step 7: Click on Start a new Android Studio project to build a new app.



Conclusion: -

Thus, we to install and configure Android Studio / Genymotion and Implement simple Android apk.



DOP: / /2023

DOS: / /2023

Experiment No: 2.

Title: Android Security Case Studies.

Theory:

What Is Android Studio?

Android Studio is an integrated development environment (IDE) created by Google to help developers create Android applications. It provides a wide range of features that make it easier to write, test, and deploy applications to Android devices. Android Studio supports all the major mobile SDKs such as the Android SDK, the Android NDK, and the Google Play Services SDK. It also includes several writing and testing tools for things such as unit testing and version control. It can also be used to build and publish Android applications to the Google Play Store. Additionally, Android Studio includes a graphical user interface for managing the emulator, creating activity layouts, and creating customized UI components.

The Android Concepts:

We first describe Android's main concepts, before presenting the challenges related to analyzing the platform with respect to security. Note that there does not exist a comprehensive document on Android's security concepts. The information is scattered throughout the Android developer's website. Android Components: An Android application consists of different parts, called components, having, according to its task, one of four basic component types. Activities are the presentation layer of an application, allowing a user to interact with the application. Services represent background processes without a user interface. Content providers are data stores that allow developers to share databases across application boundaries. Finally, broadcast receivers are components that receive and react to broadcast messages, for example, the Android OS itself sends such a broadcast message if the battery is low. Each component of an application runs as a separate task, making an Android device to a large distributed system, even if all processes are running on the same device.

How Android To Users:

Android provides security to users through a variety of mechanisms, including: Application Sandboxing: Each Android application runs in its own sandbox, isolated from other applications and the system, which helps to prevent unauthorized access to data and system resources. Permissions: Android uses a permission-based model to allow or deny access to specific features and data on a device. Users are prompted to grant or deny permissions when installing an application. Security Updates: Android releases security updates on a regular basis to patch vulnerabilities and protect against known security threats. Google Play Protect: It scans apps and checks the device for potentially harmful apps in real-time. Encryption: Android devices can be encrypted to protect data stored on the device. Built-in malware protection: Google Play store has built-in malware protection to protect users from downloading malicious apps.



Main 4 Security Techniques Used By Developer:

1. Protect Your App's Transport Layer:

One of the first things an attacker will look for when targeting an Android app is to see if they can intercept any of the data passing between it and your server's backend. By eavesdropping on those communications, they can tell an awful lot about your app. And if you're really unlucky, they might even be able to use the data to figure out how to impersonate your app and gain inappropriate access to server-side data.

2. Make Authentication Bulletproof:

Besides your app's data streams, the next most common attack vector to eliminate is any weakness in its authentication methods. The trouble with doing that is your users. By that, I mean that you need to make your app's authentication process as secure as you can without making it so onerous that your users will revolt (and if I had a dollar for every time a client asked me if 2FA was really necessary, I'd be retired by now). Regardless of how many times you have to answer the question, though, 2FA is both necessary and worth implementing. On top of that, you also need to pay attention to how you handle things like key exchanges. At a minimum, you should be using AES encryption to keep those transactions secure.

3. Guard Against Code Injection :

The next thing you should worry about is the public-facing parts of your app. Because most apps are interactive, they will provide users the ability to input data in one form or another. This can be through text-input fields like forms, or through direct data uploads for exchanging things like documents and pictures. And every time you add a user input feature, you should take great pains to make sure that nobody can use them against you. The first way to address this is to use proper input validation. If your app expects specific text in a field, make sure it won't accept anything else. You can do this by adding a pre-built text validation module or by building your own (if you have the time and inclination). If you plan to let a user upload pictures or other specific files, you must include an ability for the app to scan the uploaded file to make sure it is what it claims to be.

4. Minimize and Secure Client-Side Storage :

Last but not least, you should strive to build your app with the smallest local data footprint that's feasible to get the job done. That's because any data you store on a client device is outside of your control and is therefore vulnerable to external threats. If a user's device gets compromised by any attack that yields access to the data stored on it, you may inadvertently give the attacker a roadmap to steal information from your app.



Android Study:

We conducted our security assessment along with a security expert, the third author, to understand the implementation of some aspects of the Android security mechanism with the help of the Bauhaus tool-suite.

Issues: Starting from the documentation, we planned to focus on several aspects related to the implementation of permission checking. The process of permission enforcement consists of different steps, which we aimed to understand. After reading the security documentation, available at the Android website, we picked the Bluetooth API for further investigation. Other functionality of the platform could have been analysed in a similar way. The Bluetooth API documentation states that an application needs at least the BLUETOOTH permission to use the Bluetooth device. If a program wants to administer the Bluetooth device, it needs the BLUETOOTH_ADMIN permission in addition. It is explicitly noted that one needs the former permission, to use the latter. During the security assessment we wanted to answer several questions the security expert had. We will discuss these in the following: Question 1 Where are permissions enforced within the Bluetooth API (enforcement points)? Question 2 Which permissions are enforced within the Bluetooth API (access control policy)? Question 3 Can we check whether an application needs the BLUETOOTH permission, in order to use the BLUETOOTH_ADMIN permission?

Conclusion: -

Thus we have studied about android, there various security and the history of the android security.

DOP: / /2023

DOS: / /2023

Experiment No:3

Title: Modeling Threats in android using STRIDE

Theory:

- **Modeling Threats:**

Threats can be anything that can **take advantage of a vulnerability to breach security** and negatively alter, erase, harm objects or objects of interest. Threat Modelling can be done at any stage of development but if done at the beginning it will help in the early determination of threats that can be dealt with properly.

The **purpose of Threat modelling** is to identify, communicate, and understand threats and mitigation to the organisation's stakeholders as early as possible. Documentation from this process provides system analysts and defenders with a complete analysis of probable attacker's profiles, the most likely attack vectors, and the assets most desired by the attacker.

Threat modelling helps to achieve the following:

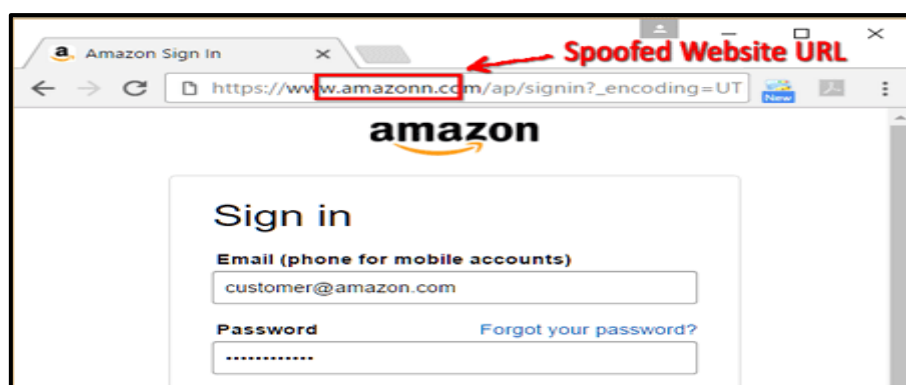
- Defines security of application
- Identifies and investigates potential threats and vulnerabilities
- Results in finding architecture bugs earlier.

STRIDE:

STRIDE is a methodology developed by **Microsoft for threat modelling**. It provides a mnemonic for security threats in six categories:

- **Spoofing Identity:**

Identify spoofing occurs when the hacker pretends to be another person, assuming the identity and information in that identity to commit fraud. A very common example of this threat is when an email is sent from a false email address, appearing to be someone else (also called a phishing attack). Typically, these emails request sensitive data. A vulnerable or unaware recipient provides the requested data and the hacker is then easily able to assume the new identity.





- **Tampering With Data:**

Data tampering occurs when data or information is changed without authorization. Ways that a bad actor can execute tampering could be through changing a configuration file to gain system control, inserting a malicious file, or deleting/modifying a log file.

- **Repudiation Threats:**

Repudiation threats happen when a bad actor performs an illegal or malicious operation in a system and then denies their involvement with the attack. In these attacks, the system lacks the ability to actually trace the malicious activity to identify a hacker.

Repudiation attacks are relatively easy to execute on e-mail systems, as very few systems check outbound mail for validity. Most of these attacks begin as access attacks.

- **Information Disclosure:**

Information disclosure is also known as information leakage. It happens when an application or website unintentionally reveals data to unauthorized users. This type of threat can affect the process, data flow and data storage in an application. Some examples of information disclosure include unintentional access to source code files via temporary backups, unnecessary exposure of sensitive information such as credit card numbers, and revealing database information in error messages.

- **Denial of Service (DoS):**

Occurs when an adversary uses illegitimate means to assume a trust level than he currently has with different privileges. Denial of Service (DoS) attacks restrict an authorized user from accessing resources that they should be able to access. This affects the process, data flow and data storage in an application. DoS attacks are getting bigger and more frequent, with an estimated 12.5 million DoS weapons detected in 2020.

- **Elevation of Privileges:**

Through the elevation of privileges, an authorized or unauthorized user in the system can gain access to other information that they are not authorized to see. An example of this attack could be as simple as a missed authorization check, or even elevation through data tampering where the attacker modifies the disk or memory to execute non-authorized commands.



- **Example Business Case**

The example business case, for our STRIDE threat modeling example, consists of:

Company and industry: Health care insurance provider, within the health care and insurance industry.

Object in the scope of threat modeling: The company is in the process of developing a new web application called 'MyHealth'. It will allow customers to view their medical data, billing, appointments with healthcare providers, and potential deals that customers can purchase.

Scoping limitations: Currently, the business processes related to the MyHealth application are not in scope.

Tech stack of the application: The MyHealth web application will be cloud-based, hosted in Azure, and will have a JavaScript framework front-end.

Team: Various DevOps teams are involved in the project, as well as business driving the transformation to provide as many health insurance services through the newly developed MyHealth application.

- **First Step: Gather Background Information:**

The first step in the STRIDE threat modeling example is to gather as much background information as possible (which sounds obvious but is still worth mentioning).

Potential information that may help in later STRIDE threat modeling steps:

Architectural diagrams and overviews of the proposed application design: Such architectural diagrams can provide threat modelers and team members with an understanding of key components within the design, technology used, communication flows, etc.

Requirements of the solution: Requirements often contain the business and IT requirements that explain what the application must be capable of doing. The amount and quality of requirements may vary in quality from project to project. There may even be some security requirements that can help with the later STRIDE threat modeling steps.

Project documentation: Project documentation will explain how the application will be built, who is involved, how long the project will take, etc. Pay attention to the inclusion of security as part of the project. This will indicate whether security is an integral part of the project (and thus resulting application), or whether it is a 'forgotten' quality. Team overview and governance: Understanding the key players in a project is very important. Pay attention to the security team members involved (i.e., security champions, security department representatives, etc.).



- **Second Step: Create a Data Flow Diagram (DFD):**

Because we're applying the component-based STRIDE threat modeling method, we need a good understanding of the key components involved in the project, and how they interact with each other.

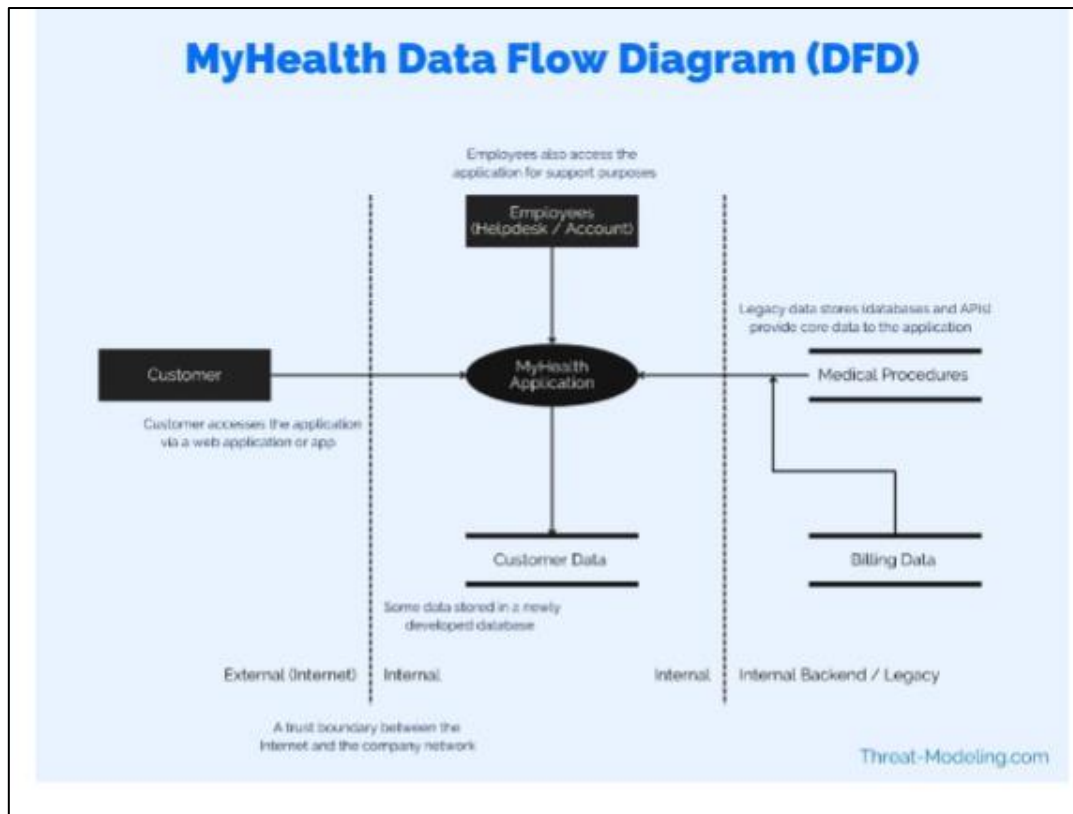
To achieve this, I'll create a Data Flow Diagram (DFD), which is the second step in the STRIDE threat modeling example. It will include only the key components, the key interactors (e.g., users), data stores, and the communication flows between them.

Architectural diagrams and overviews of the proposed application design help in creating a DFD.

The reason why I'm only including key components (versus including all components) is that including too much information can make the overall threat model too complicated and too difficult to understand. Participants in the threat modeling sessions will not be invested or engaged if they do not understand what they are threat modeling.

The Data Flow Diagram (DFD) should be made based on input from team members in an interactive threat modeling session. It is important to get input and cooperation from a wide range of team members, including technical/non-technical, developers/non-developers, testers, Product Owners, and any other people and roles with knowledge of the application.

The diagram below is an example of a Data Flow Diagram (DFD) based on the MyHealth example.



Customer: The end customer using the application. The customer is a health insurance recipient (paying a monthly health insurance fee, and receiving health insurance coverage).

Employees: Employees access the application as well as customers. Employees provide support to customers via a helpdesk and other channels. Employees can see more data than customers, including access to many different customers.

MyHealth Application: This is the main application in the scope of the threat modeling exercise. It is shown here as one icon. However, the application has many sub- components, backend, frontend, etc.

Customer Data: The new application will have a new database environment to store some of the relevant customer data.

Medical Procedures (and activities): Data provided to the customer includes medical procedures, activities, etc. This data is stored in a legacy database and system. This data will not be sent to the Customer Data database.

Billing Data: Billing data is managed in a separate system, but is shown via the MyHealth application (to the customer).

Note that the above Data Flow Diagram is not perfect, but for the purposes of keeping the STRIDE threat modeling example simple, it will do for now. Future posts will focus on how to tweak and improve on details included in a Data Flow Diagram.

- **Third Step: Perform Component Based STRIDE Threat Modeling:**

Now, as a result of creating a Data Flow Diagram, we have an understanding of the main components in the STRIDE threat modeling example. As a next step, I will perform STRIDE threat modeling against each component (known as component-based STRIDE threat modeling).

Two approaches for this step:

1. Think of potential threats per component, and assign a STRIDE threat type (i.e., assign Spoofing, or Tampering, etc.).
2. Per component, think of Spoofing threats specifically, then Tampering threats, then Repudiation threats, etc. So in order of STRIDE, go through the list and think of potential threats limited to the STRIDE threat type. I usually use both methods, or a combination of both.

For this example, I'll use method 2 – the method of strictly thinking per component, and walking through all STRIDE threat types, to determine which threats may apply.

Conclusion: -

Hence successfully performed Modeling Threat in Android using Stride.



DOP: / /2023

DOS: / /2023

Experiment No:4

Title: Building Android applications User interfaces using various Views and Layouts.

Theory:

View:

View is the basic building block of UI(User Interface) in android. View refers to the android.view.View class, which is the super class for all the GUI components like TextView, ImageView, Button etc.

View class extends Object class and implements Drawable.Callback, KeyEvent.Callback and AccessibilityEventSource.

View can be considered as a rectangle on the screen that shows some type of content. It can be an image, a piece of text, a button or anything that an android application can display. The rectangle here is actually invisible, but every view occupies a rectangle shape.

The question that might be bothering you would be , what can be the size of this rectangle?

The answer is either we can set it manually, by specifying the exact size(with proper units) or by using some predefined values. These predefined values are match_parent and wrap_content.

match_parent means it will occupy the complete space available on the display of the device. Whereas, wrap_content means it will occupy only that much space as required for its content to display.

XML syntax for creating a View

Now, as we have explained earlier as well, to draw anything in your android application, you will have to specify it in the design XML files. And to add functionality we will create Java files.

<ViewName

Attribute1=Value1

Attribute2=Value2

Attribute3=Value3

AttributeN=ValueN

/>



Most commonly used Android View classes

Here we have some of the most commonly used android View classes:

- TextView
- EditText
- Button
- ImageView
- ImageButton
- CheckBox
- RadioButton
- ListView
- GridView

ViewGroup:

A ViewGroup is a special view that can contain other views. The ViewGroup is the base class for Layouts in android, like LinearLayout, RelativeLayout, FrameLayout etc.

In other words, ViewGroup is generally used to define the layout in which views(widgets) will be set/arranged/listed on the android screen.

ViewGroups acts as an invisible container in which other Views and Layouts are placed. Yes, a layout can hold another layout in it, or in other words a ViewGroup can have another ViewGroup in it.

Most commonly used Android layout types

- LinearLayout
- RelativeLayout
- Web View
- Tabular layout
- ListView
- GridView

Input:

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#291D3E"
    android:orientation="horizontal"
    android:layout_weight="2"
    tools:context=".MainActivity"
    tools:ignore="MissingPrefix">

    <ImageView
        android:layout_width="224dp"
        android:layout_height="197dp"
        android:layout_marginTop="124dp"
        android:background="@drawable/img_1"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="MissingConstraints" />

    <EditText
        android:id="@+id/edittext1"
        android:layout_width="229dp"
        android:layout_height="50dp"
        android:hint="Entre Username"
        android:textColor="#ffff"
        android:inputType="text"
        android:textStyle="bold"
        android:layout_weight="1"
        android:textAlignment="center"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.483"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.499"
```

```
    <EditText
        android:id="@+id/edittext2"
        android:layout_width="229dp"
        android:layout_height="50dp"
        android:layout_below="@id/edittext1"
        android:layout_weight="2"
        android:hint="Password"
        android:textAlignment="center"
        android:textColor="#ffff"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.483"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.64" />

    <Button
        android:id="@+id/my_button"
        android:layout_width="230px"
        android:layout_height="100px"
        android:layout_marginBottom="96dp"
        android:onClick="click"
        android:text="Go"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="MissingConstraints" />

</androidx.constraintlayout.widget.ConstraintLayout>
```


activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity2">

    <TextView
        android:id="@+id/received_value_id"
        android:layout_width="300dp"
        android:layout_height="50dp"
        android:layout_marginStart="40dp"
        android:layout_marginLeft="40dp"
        android:layout_marginTop="100dp"
        android:textSize="40sp"
        android:textStyle="bold" />
</RelativeLayout>
```

java

```
package com.example.ui_view_layout;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {
    EditText username;
    EditText password;
    Button button;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        username=findViewById(R.id.edittext1);
        password=findViewById(R.id.edittext2);
        button=findViewById(R.id.my_button);

        button.setOnClickListener(view ->{
            String str= username.getText().toString();
            String strr= password.getText().toString();
            Intent intent=new Intent(getApplicationContext(),MainActivity2.class);
            intent.putExtra("message_key",str);
            intent.putExtra("message_key",strr);
            startActivity(intent);

        });
    }
}
```

```
package com.example.ui_view_layout;

import androidx.appcompat.app.AppCompatActivity;

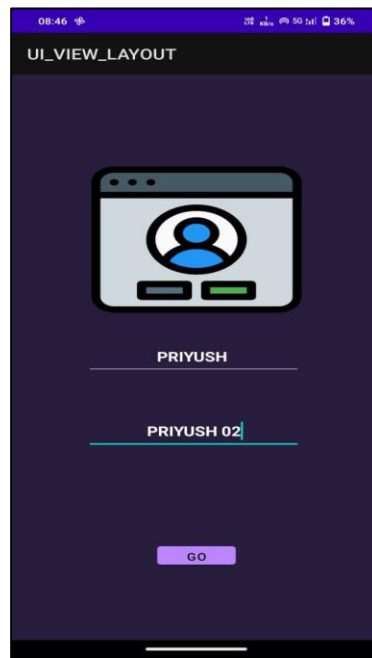
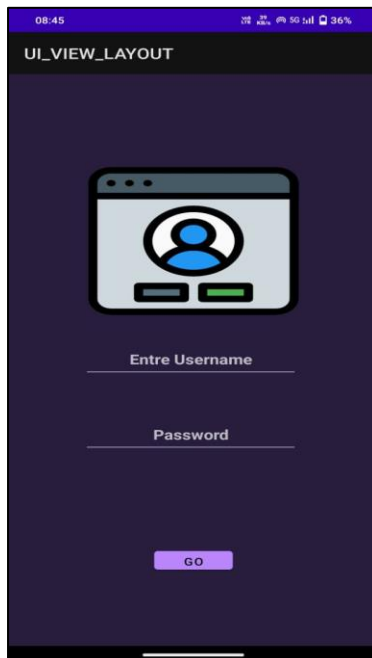
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity2 extends AppCompatActivity {

    TextView recerview_msg;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        recerview_msg=findViewById(R.id.received_value_id);
        Intent intent=getIntent();
        String str=intent.getStringExtra("message_key");
        String strr=intent.getStringExtra("message_key");
        recerview_msg.setText(str);
        recerview_msg.setText(strr);

    }
}
```

Output:



Conclusion: - Hence successfully performed Building Android applications User interfaces using various Views and Layouts.

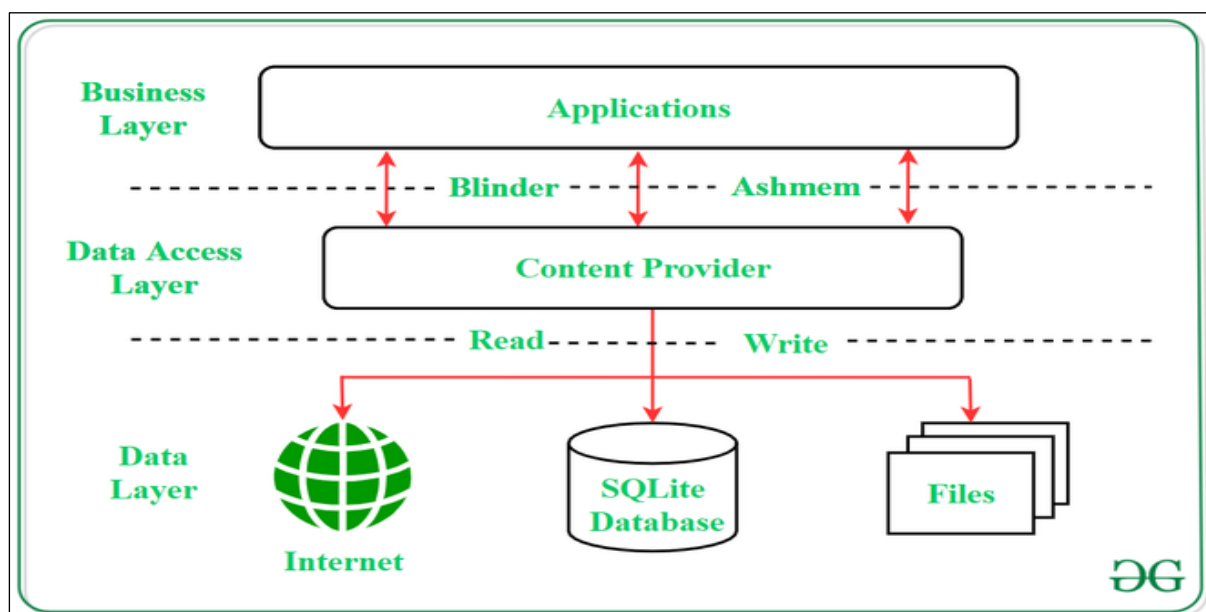
Experiment No:5

Title: Developing Android applications using Receivers and Content Providers.

Theory:

Content Providers:

Content Providers are a very important component that serves the purpose of a relational database to store the data of applications. The role of the content provider in the android system is like a central repository in which data of the applications are stored, and it facilitates other applications to securely access and modifies that data based on the user requirements. Android system allows the content provider to store the application data in several ways. Users can manage to store the application data like images, audio, videos, and personal contact information by storing them in SQLite Database, in files, or even on a network. In order to share the data, content providers have certain permissions that are used to grant or restrict the rights to other applications to interfere with the data.



Content URI:

Content URI (Uniform Resource Identifier) is the key concept of Content providers. To access the data from a content provider, URI is used as a query string.

Structure of a Content URI: content://authority/optionalPath/optionalID

Operations in Content Provider:

Four fundamental operations are possible in Content Provider namely Create, Read, Update, and Delete. These operations are often termed as CRUD operations.

- Create: Operation to create data in a content provider.
- Read: Used to fetch data from a content provider.
- Update: To modify existing data.
- Delete: To remove existing data from the storage.

Input:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#2EAF43"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Content provider"
        android:textSize="25sp"
        android:textStyle="bold"
        tools:ignore="MissingConstraints"
        tools:layout_editor_absoluteX="126dp"
        tools:layout_editor_absoluteY="99dp" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="133dp"
        android:layout_height="34dp"
        android:layout_marginEnd="32dp"
        android:hint="Name"
        android:text=""
        android:textAlignment="center"
        android:textSize="25sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.516"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.286" />
```

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="140dp"
    android:layout_height="34dp"
    android:layout_marginEnd="144dp"
    android:hint="Contact no"
    android:maxLength="13"
    android:text=""
    android:textAlignment="center"
    android:textSize="25sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.433" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.592" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Javafile:

```
package com.example.content_provider;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import android.Manifest;
import android.annotation.SuppressLint;
import android.content.ContentResolver;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.util.Log;
import android.util.Range;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    Button btn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn = findViewById(R.id.button);
        btn.setOnClickListener(this);
    }
    @Override
    public void onClick(View view) {
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_CONTACTS) != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(new String[]{Manifest.permission.READ_CONTACTS}, 100);
        } else {
            Intent openContactApp = new Intent(Intent.ACTION_PICK, ContactsContract.CommonDataKinds.Phone.CONTENT_URI);
            startActivityForResult(openContactApp, 123);
        }
    }
    @Override
```

```

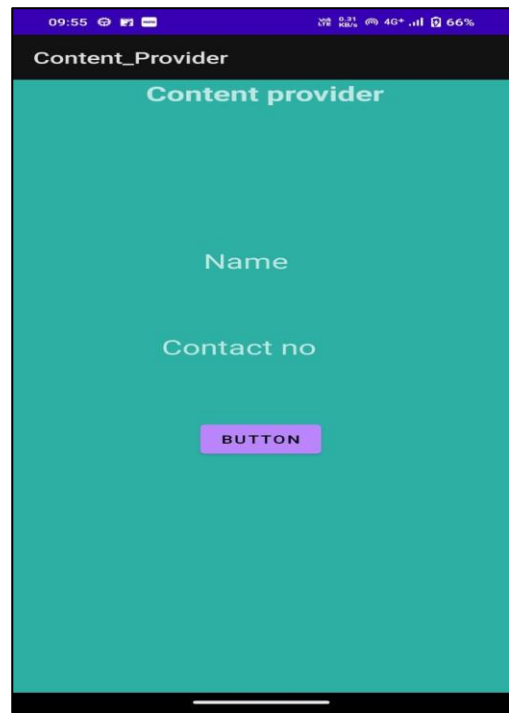
    @Override
    public void onClick(View view) {
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_CONTACTS) != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(new String[]{Manifest.permission.READ_CONTACTS}, 100);
        } else {
            Intent openContactApp = new Intent(Intent.ACTION_PICK, ContactsContract.CommonDataKinds.Phone.CONTENT_URI);
            startActivityForResult(openContactApp, 123);
        }
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode==123&& resultCode == RESULT_OK) {
            String[] columns = new String[]{
                ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
                ContactsContract.CommonDataKinds.Phone.NUMBER
            };
            ContentResolver resolver = getContentResolver();
            Uri dataUri = data.getData();
            Cursor cursor = resolver.query(dataUri, columns, null, null, null);
            if (cursor.moveToFirst()) {
                @SuppressLint("Range") String name = cursor.getString(cursor.getColumnIndex(columns[0]));
                Log.i("name", "" + name);

                @SuppressLint("Range") String contact = cursor.getString(cursor.getColumnIndex(columns[1]));
                Log.i("contact", "" + contact);

                TextView tvname = findViewById(R.id.textView);
                tvname.setText(name);
                TextView tvcontact = findViewById(R.id.textView2);
                tvcontact.setText(contact);
            } else {
                Toast.makeText(this, "Unable to load contact", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(this, "Contact not selected", Toast.LENGTH_SHORT).show();
        }
    }
}
}
```



**Jawahar Education Society's Annasaheb Chudaman Patil College of
Engineering, Kharghar, Navi Mumbai**



Conclusion: - Hence successfully performed Developing Android applications using Receivers and Content Providers.

DOP: / /2023

DOS: / /2023

Experiment No:6

Title: Developing user interactive Database applications (Using SQLite or other) in Android.

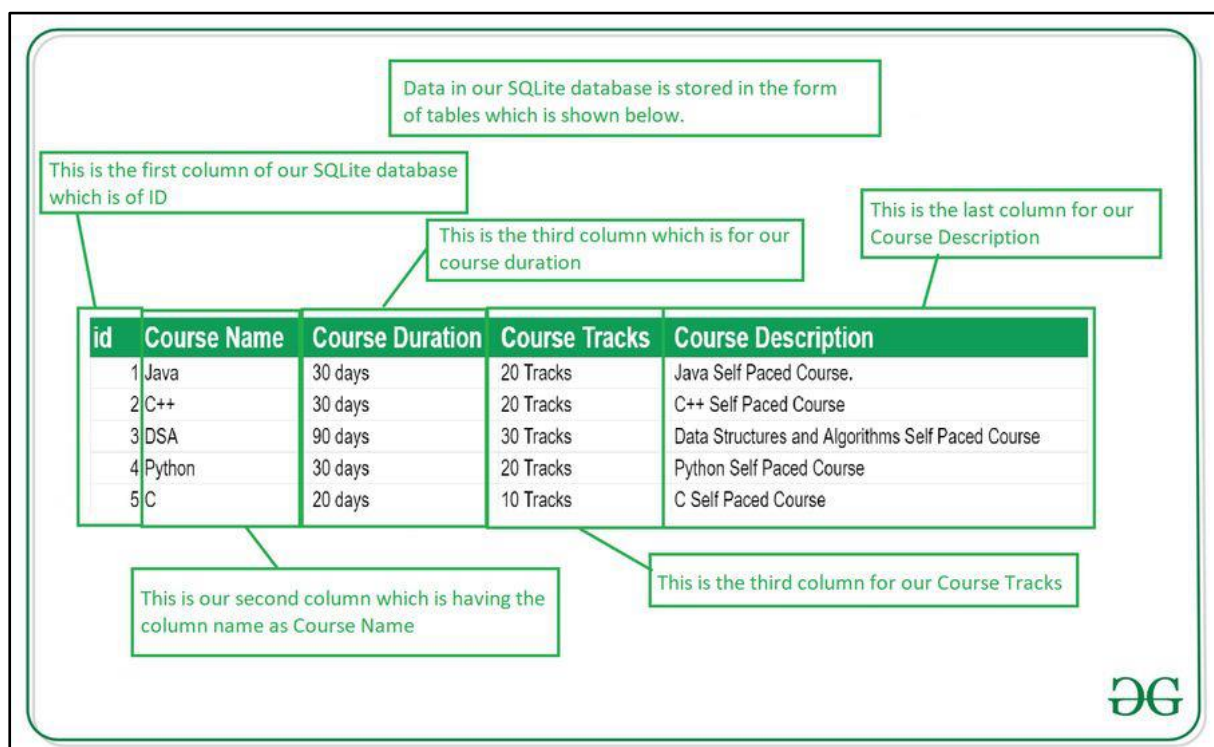
Theory:

What is SQLite Database?

SQLite Database is an open-source database provided in Android which is used to store data inside the user's device in the form of a Text file. We can perform so many operations on this data such as adding new data, updating, reading, and deleting this data. SQLite is an offline database that is locally stored in the user's device and we do not have to create any connection to connect to this database.

How Data is Being Stored in the SQLite Database?

Data is stored in the SQLite database in the form of tables. When we stored this data in our SQLite database it is arranged in the form of tables that are similar to that of an excel sheet. Below is the representation of our SQLite database which we are storing in our SQLite database.



Data in our SQLite database is stored in the form of tables which is shown below.

This is the first column of our SQLite database which is of ID


This is the third column which is for our course duration

This is the last column for our Course Description

id	Course Name	Course Duration	Course Tracks	Course Description
1	Java	30 days	20 Tracks	Java Self Paced Course.
2	C++	30 days	20 Tracks	C++ Self Paced Course
3	DSA	90 days	30 Tracks	Data Structures and Algorithms Self Paced Course
4	Python	30 days	20 Tracks	Python Self Paced Course
5	C	20 days	10 Tracks	C Self Paced Course

This is our second column which is having the column name as Course Name

This is the third column for our Course Tracks



Important Methods in SQLite Database:

Method	Description
getColumnNames()	This method is used to get the Array of column names of our SQLite table.
getCount()	This method will return the number of rows in the cursor.
isClosed()	This method returns a Boolean value when our cursor is closed.
getColumnCount()	This method returns the total number of columns present in our table.
getColumnName(int columnIndex)	This method will return the name of the column when we passed the index of our column in it.
getColumnIndex(String columnName)	This method will return the index of our column from the name of the column.
getPosition()	This method will return the current position of our cursor in our table.

Input:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_above="@+id/idEdtCourseName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="SQLite"
        android:textSize="100px"/>

    <EditText
        android:id="@+id/idEdtCourseName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:hint="Enter course Name" />

    <!--edit text to enter course duration-->
    <EditText
        android:id="@+id/idEdtCourseDuration"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:hint="Enter Course Duration"
        android:layout_below="@+id/idEdtCourseName" />
```



```

<!--edit text to display course tracks-->
<EditText
    android:id="@+id/idEdtCourseTracks"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/idEdtCourseDuration"
    android:layout_marginStart="10dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="10dp"
    android:layout_marginBottom="10dp"
    android:hint="Enter Course Tracks" />

<!--edit text for course description-->
<EditText
    android:id="@+id/idEdtCourseDescription"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:hint="Enter Course Description"
    android:layout_below="@+id/idEdtCourseTracks" />

<!--button for adding new course-->
<Button
    android:id="@+id/idBtnAddCourse"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginStart="10dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="10dp"
    android:layout_marginBottom="10dp"
    android:text="Add Course"
    android:textAllCaps="false"
    android:layout_below="@+id/idEdtCourseTracks" />
</RelativeLayout>

```

Java file:

```

package com.example.sqlite;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    private EditText courseNameEdt, courseTracksEdt, courseDurationEdt, courseDescriptionEdt;
    private Button addCourseBtn;
    private DBHelper dbHelper;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        courseNameEdt = findViewById(R.id.idEdtCourseName);
        courseTracksEdt = findViewById(R.id.idEdtCourseTracks);
        courseDurationEdt = findViewById(R.id.idEdtCourseDuration);
        courseDescriptionEdt = findViewById(R.id.idEdtCourseDescription);
        addCourseBtn = findViewById(R.id.idBtnAddCourse);
        dbHelper = new DBHelper(MainActivity.this);

        // below line is to add on click listener for our add course button.
        addCourseBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                // below line is to get data from all edit text fields.
                String courseName = courseNameEdt.getText().toString();
                String courseTracks = courseTracksEdt.getText().toString();
                String courseDuration = courseDurationEdt.getText().toString();
                String courseDescription = courseDescriptionEdt.getText().toString();

                // validating if the text fields are empty or not.
                if (courseName.isEmpty() && courseTracks.isEmpty() && courseDuration.isEmpty() && courseDescription.isEmpty()) {
                    Toast.makeText(MainActivity.this, "Please enter all the data..", Toast.LENGTH_SHORT).show();
                    return;
                }
            }
        });
    }
}

```



Jawahar Education Society's Annasaheb Chudaman Patil College of Engineering, Kharghar, Navi Mumbai

```
dbHandler.addNewCourse(courseName, courseDuration, courseDescription, courseTracks);

// after adding the data we are displaying a toast message.
Toast.makeText(MainActivity.this, "Course has been added.", Toast.LENGTH_SHORT).show();
courseNameEdt.setText("");
courseDurationEdt.setText("");
courseTracksEdt.setText("");
courseDescriptionEdt.setText("");

    }
}
}
```

DBHandler.java:

```
package com.example.sqlite;

import android.content.ContentValues;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHandler extends SQLiteOpenHelper {

    // creating a constant variables for our database.
    // below variable is for our database name.
    private static final String DB_NAME = "coursedb";

    // below int is our database version
    private static final int DB_VERSION = 1;

    // below variable is for our table name.
    private static final String TABLE_NAME = "mycourses";

    // below variable is for our id column.
    private static final String ID_COL = "id";

    // below variable is for our course name column
    private static final String NAME_COL = "name";

    // below variable id for our course duration column.
    private static final String DURATION_COL = "duration";

    // below variable for our course description column.
    private static final String DESCRIPTION_COL = "description";

    // below variable is for our course tracks column.
    private static final String TRACKS_COL = "tracks";

    // creating a constructor for our database handler.
    public DBHandler(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
    }

    // below method is for creating a database by running a sqlite query
    @Override
```

```
// below method is for creating a database by running a sqlite query
@Override
public void onCreate(SQLiteDatabase db) {
    // on below line we are creating
    // an sqlite query and we are
    // setting our column names
    // along with their data types.
    String query = "CREATE TABLE " + TABLE_NAME + " ("
        + ID_COL + " INTEGER PRIMARY KEY AUTOINCREMENT, "
        + NAME_COL + " TEXT,"
        + DURATION_COL + " TEXT,"
        + DESCRIPTION_COL + " TEXT,"
        + TRACKS_COL + " TEXT)";

    // at last we are calling a exec sql
    // method to execute above sql query
    db.execSQL(query);
}

// this method is use to add new course to our sqlite database.
public void addNewCourse(String courseName, String courseDuration, String courseDescription, String courseTracks) {

    // on below line we are creating a variable for
    // our sqlite database and calling writable method
    // as we are writing data in our database.
    SQLiteDatabase db = this.getWritableDatabase();

    // on below line we are creating a
    // variable for content values.
    ContentValues values = new ContentValues();

    // on below line we are passing all values
    // along with its key and value pair.
    values.put(NAME_COL, courseName);
    values.put(DURATION_COL, courseDuration);
    values.put(DESCRIPTION_COL, courseDescription);
    values.put(TRACKS_COL, courseTracks);

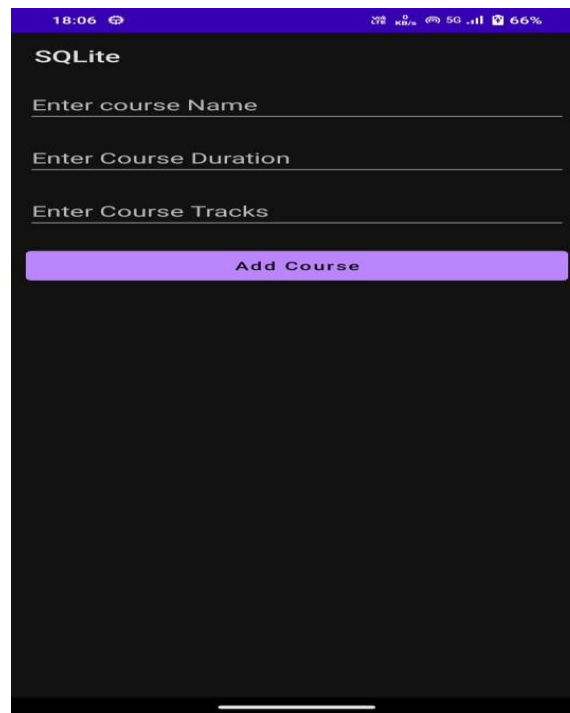
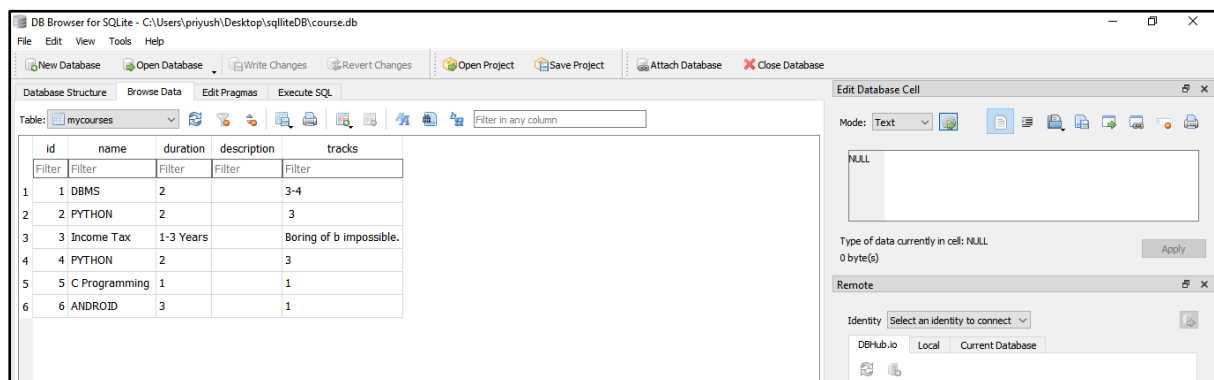
    // after adding all values we are passing
    // content values to our table.
    long insert = db.insert(TABLE_NAME, null, values);

    // at last we are closing our
    // database after adding database.
```

```
// at last we are closing our
// database after adding database.
db.close();
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // this method is called to check if the table exists already.
    db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
    onCreate(db);
}
}
```

Output:

	id	name	duration	description	tracks
1	1	DBMS	2		3-4
2	2	PYTHON	2		3
3	3	Income Tax	1-3 Years	Boring of b impossible.	
4	4	PYTHON	2		3
5	5	C Programming	1		1
6	6	ANDROID	3		1

Conclusion: - Hence successfully performed Developing user interactive Database applications (Using SQLite or other) in Android.



DOP: / /2023

DOS: / /2023

Experiment No:6

Title: Deploying and Publishing Android application.

Theory:

When the application development process is completed, it's high time to introduce your special product to the world. First publishing may be thrilling and tricky. Don't worry. Today we will guide you through the release process on the Google Play Store, which is undoubtedly the leader in the number of published apps and users.

Step 1: Create a Google Developer account:

This is something you can do at the beginning of the app development process. Without registering a Google Developer Account, you can't publish your app on Google Play.

You can use any of your current Google accounts or create another one to sign up for a Google Developer Account. It doesn't matter whether it's a private or corporate account. You may easily transfer your app to another one in the future.

The creation process includes signing the Google Play Developer distribution agreement, adding some personal information, and paying a one-time registration fee of \$25.



Step 2: Add a Merchant Account:

If you plan to sell paid apps or in-app purchases, you have to create a Google Merchant Account. There you can manage app sales and your monthly payouts, as well as analyze sales reports.

Step 3: Prepare the Documents:

Paperwork always requires much effort, especially when it comes to any kind of legal documents. Based on our experience, we highly recommend starting to prepare the End User License Agreement (EULA) and Privacy Policy in advance.

You can take the documents from similar apps as references and create your own based on them, or ask a lawyer to make everything from scratch.

EULA is an agreement between you as an owner and a user of your product. In brief, it contains:

- What the users can do with the app, and what they aren't allowed to do
- Licensing fees
- Intellectual property information, etc.

Step 4: Study Google Developer Policies:

We guess you already made up your product concept. Now it's time to make sure that every feature you will implement in the app is aligned with the Google Developer Policies. These documents explain how mobile apps need to be developed, updated, and promoted to support the store's high-quality standards.

Step 5: Technical Requirements:

You went through the development process, endless testing, and bug fixing, and finally, the "X-day" comes. Before moving on to the upload process, you need to check the following things:

- Unique Bundle ID

The package name should be suitable over the life of your application. You cannot change it after the distribution. You can set the package name in the application's manifest file.

- Signed App Release With a Signing Certificate

Every application should be digitally signed with a developer's certificate. The certificate is used to identify the author of an app and can't be generated again.

- The App Size



Jawahar Education Society's Annasaheb Chudaman Patil College of Engineering, Kharghar, Navi Mumbai

Google set the limit size of the uploaded file: 100MB for Android 2.3 and higher (API level 9-10, 14 and higher) and 50MB for lower Android versions.

If your app exceeds this limit, you can always switch to APK Expansion Files.

- The File Format

Two possible release formats are accepted by Google: app bundle and .apk. However, .aab is the preferred one. To use this format, you need to enroll in app signing by Google Play.

Step 6: Creating the App on the Google Console:

Now you have the file that is ready for uploading. It's time to get to the fun part. Let's create a new app in your Developer Account:

- Reach to All applications tab in the menu
- Now select Create Application
- Choose the app's default language from the drop-down menu
- Add a brief app description (you can change it later)
- Tap on Create

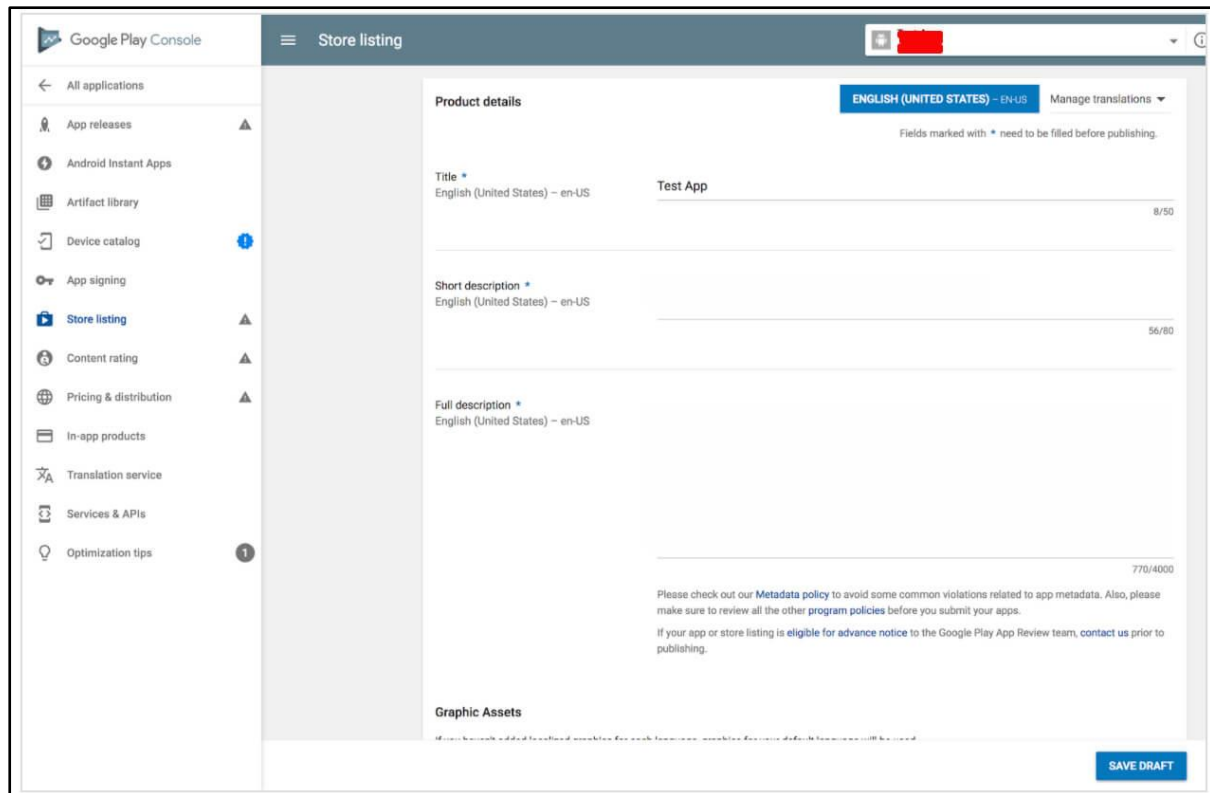
Step 7: Store Listing:

First, let's prepare the Store listing. It contains the most important information useful for app store optimization (ASO) and gives the users more details about your app before downloading. The mandatory sections are marked with *.

You may need some designer and copywriter efforts, so it's better to start preparing the following materials in advance.

- Product description

It contains a title of your app (up to 50 symbols), a brief description (up to 80 symbols), and a full description (up to 4000 symbols). Control yourself and do not overdo the keywords.



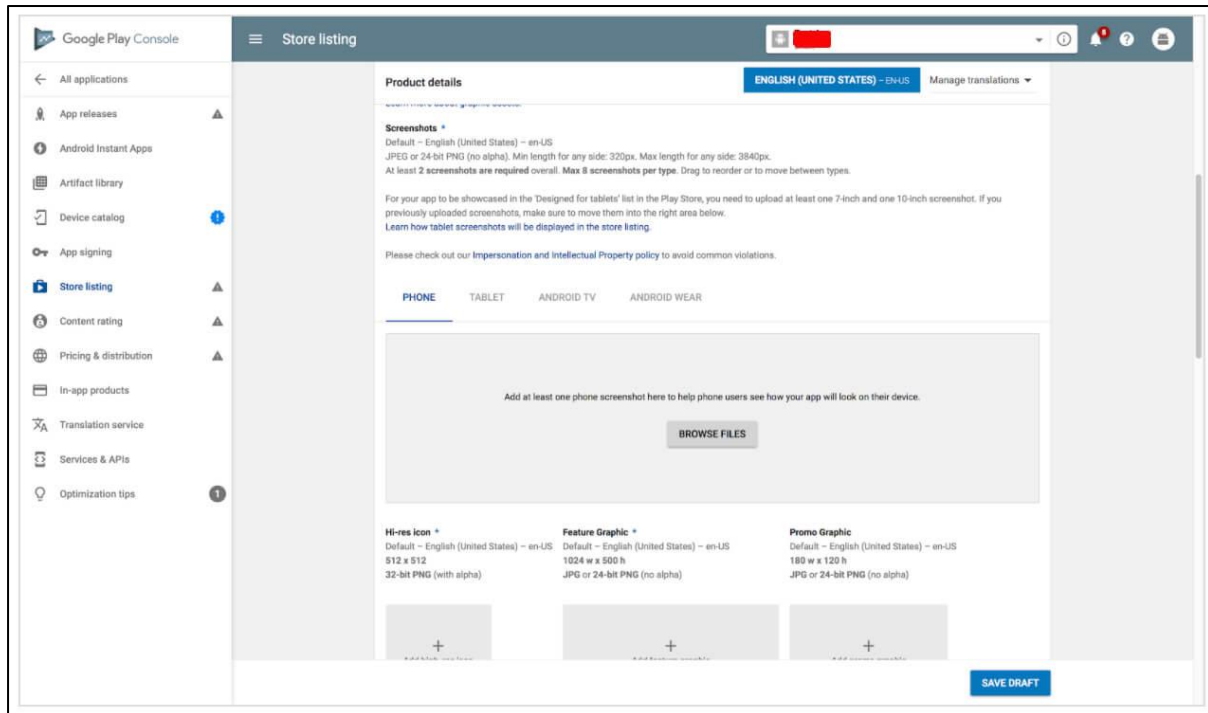
Store listing

Screenshots

You may add from 2 to 8 screenshots. Choose the ones that show the app functionality and value the most.

The requirements are the following:

- JPEG or 24-bit PNG (no alpha)
- from 320px to 3840 px
- the ratio of the long side to the short side should not be more than 2:1



Store listing - Product details

Icon

The requirements are the following:

- 512px by 512px
- 32-bit PNG (with alpha)
- Maximum file size: 1024KB

Feature graphic

It is an optional marketing tool displayed in various places on Google Play, for example, on the homepage.

The requirements are the following:

- JPEG or 24-bit PNG (no alpha)
- 1024px x 500px
- Promo video

If you have any promo video, you may add a link to your YouTube channel. This video will be shown before the screenshots on the app's page.

- Tags

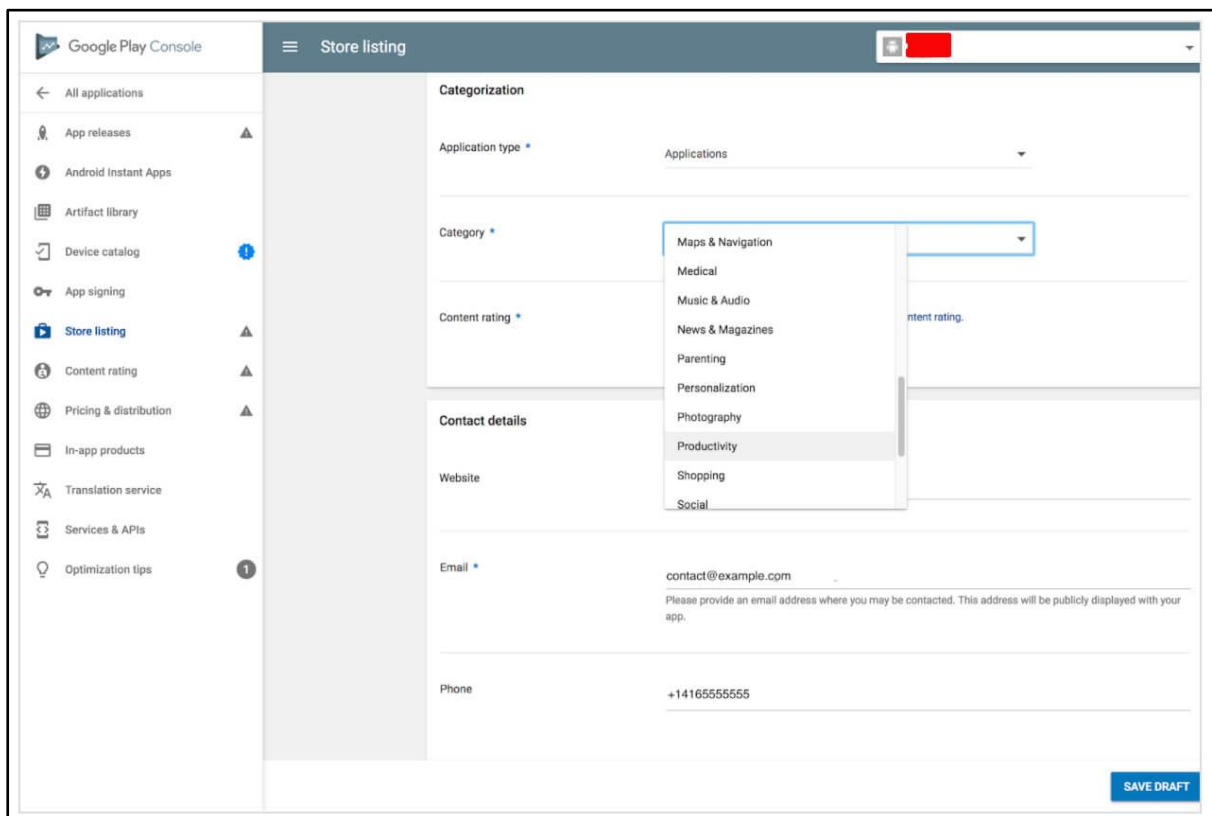
You may choose from the list the most relevant to your app keywords for better ASO. There is no possibility to add any custom tags.

- Localization

If your app supports several languages, mention all of them and add translations of your app's information. It's highly recommended to include localized screenshots and images.

- Application type and categorization

First, through the drop-down menu, select the application type: game or app. Then pick the category that your app fits into. You can also add a section to rate your content after uploading APK to Google Play store.



The screenshot shows the Google Play Console interface for the 'Store listing' of an application. The left sidebar contains navigation options like 'All applications', 'App releases', 'Android Instant Apps', 'Artifact library', 'Device catalog', 'App signing', 'Store listing' (selected), 'Content rating', 'Pricing & distribution', 'In-app products', 'Translation service', 'Services & APIs', and 'Optimization tips'. The main content area is titled 'Store listing' and includes a language selector (set to English). The 'Categorization' section has a dropdown for 'Application type' (set to 'Applications') and a dropdown for 'Category' (with a list of categories including 'Maps & Navigation', 'Medical', 'Music & Audio', 'News & Magazines', 'Parenting', 'Personalization', 'Photography', 'Productivity', 'Shopping', and 'Social'). Below this is the 'Content rating' section. The 'Contact details' section includes fields for 'Website', 'Email' (with a note: 'Please provide an email address where you may be contacted. This address will be publicly displayed with your app.'), and 'Phone'. A 'SAVE DRAFT' button is located at the bottom right.

- Contact details

Here you should provide the support service contacts. By filling the website URL, email, and phone, you make it easier for the users to contact you if necessary.

- Privacy Policy



Jawahar Education Society's Annasaheb Chudaman Patil College of Engineering, Kharghar, Navi Mumbai

Google requires you to add a link to the Privacy Policy that we discussed above.

While editing the Store Listing, you can take a break at any moment, click Save Draft, and complete this stage later.

Step 8: Content Rating:

In order not to be marked as an Unrated App (that may lead to app removal), pass a rating questionnaire. You can easily find this section on the left-side menu.

The information provided in the questionnaire must be accurate. Any misrepresentation of your app's content might lead to suspension or removal of the Google Play account.

- Click on Save Questionnaire once you complete the survey
- Click on Calculate Rating
- In the end, click on Apply Rating to confirm the rating and move forward with the pricing & distribution plan

Step 9: Pricing the Application

In the Pricing and distribution section, you need to fill the following information:

- Whether your app is free or paid
- Where the app will be available (just choose the countries from the list)
- Whether your app will be available only on the specific devices
- Whether the app has sensitive content and is not suitable for children under the age of 13
- Whether your app contains ads

Remember that you can change your paid app to a free one later, but you cannot do the vice versa. If you decide later that you want to distribute the app for money, you'll have to create another app.



Step 10: Upload APK and Send for Review

Finally, you are ready to upload your Android app file. That's the most exciting moment ever.

Let's go to the App Releases section on the left panel. Here you will find three options for publishing the app: Production, Beta and Alpha tracks.

We highly recommend starting with Alpha or Beta versions. In this case, after passing the review process, your app will not be available to everyone on the Google Play store.

The Alpha version assumes closed testing and is available only to those who you invite as testers. The Beta version means that anyone can join your testing program and send feedback to you.

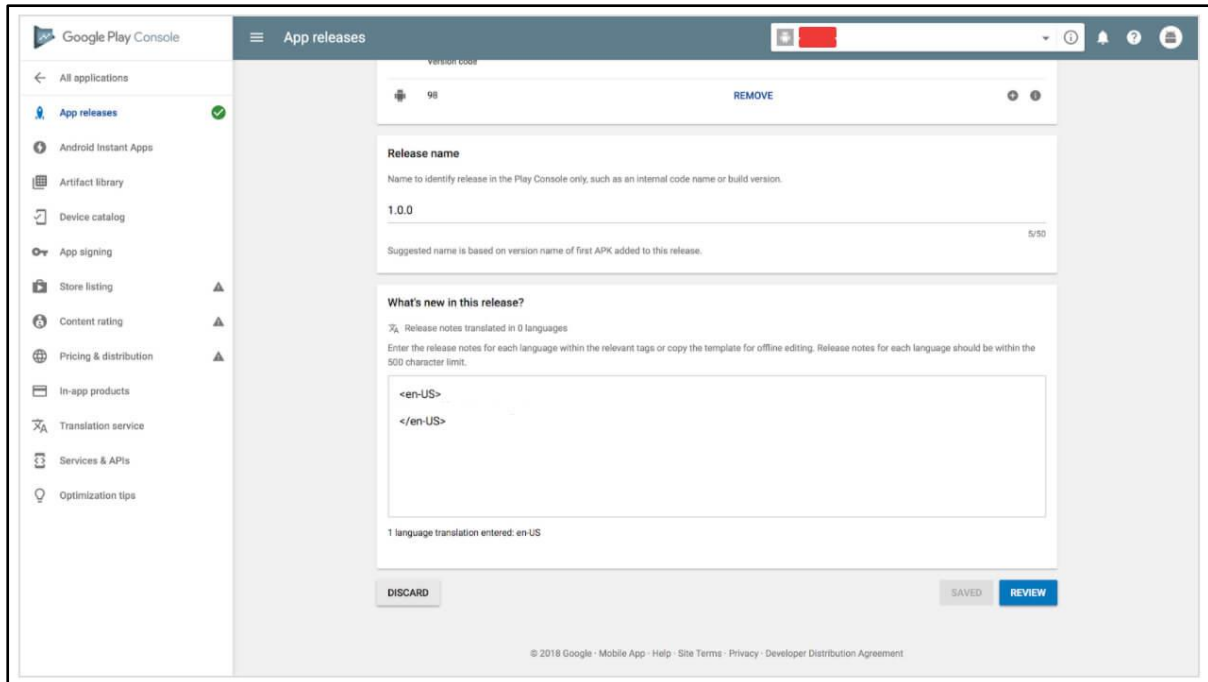
Pre-release testing allows you to gather people's opinions, test your app in a broader audience, and fix issues before making the app public.

Note that if you decide later to change the Alpha or Beta version to Production type, it will take time to go through another review round.

Once you choose the type of release, follow the steps:

- Choose Manage (Production/Beta/Alpha)
- Click on Edit Release
- Upload an APK or app bundle

The release name will be added automatically. For the first time, you may delete the text from the What's new in this release field.



The screenshot shows the Google Play Console interface for managing app releases. The left sidebar contains navigation options: All applications, App releases (selected), Android Instant Apps, Artifact library, Device catalog, App signing, Store listing, Content rating, Pricing & distribution, In-app products, Translation service, Services & APIs, and Optimization tips. The main content area is titled 'App releases' and shows a version code '98' with a 'REMOVE' button. Below this, the 'Release name' section has a text input field containing '1.0.0' and a character count '5/50'. The 'What's new in this release?' section has a text area with placeholder text '<en-US>' and '</en-US>'. At the bottom, there are buttons for 'DISCARD', 'SAVED', and 'REVIEW'. The footer of the console shows '© 2018 Google · Mobile App · Help · Site Terms · Privacy · Developer Distribution Agreement'.

Click on Review to confirm the changes and send your app to the review by pressing Start rollout to production.

Conclusion: - Hence successfully performed Deploying and Publishing Android application.

DOP: / /2023

DOS: / /2023

Experiment No:7



Title: Reversing Android applications (APKs) APKTOOL, dex2jar and JD-GUI.

Theory:

Reverse engineering

Reverse engineering an Android application typically involves using specialized tools to decompile the applications compiled code and resources into a human-readable form. As we go through this blog post, we will discuss the various available tools and examples of how they can be used to find hardcoded data and potentially find static application vulnerabilities.

- Understand how a particular UI in an App is constructed
- reading AndroidManifest.xml - permissions, activities, intents etc in the App
- native libraries and images used in that App
- obfuscated code (android SDK, by default, uses ProGuard tool which shrinks, optimizes, and obfuscates your code by removing unused code and renaming classes, fields, and methods with semantically obscure names.

Required Tools:

Download the followings first.

- Dex2jar from <http://code.google.com/p/dex2jar/>
- JD-GUI from <http://java.decompiler.free.fr/?q=jdgui>
- ApkTool from <http://code.google.com/p/android-apktool/>

Using ApkTool

- to extract AndroidManifest.xml and everything in res folder(layout xml files, images, htmls used on webview etc..)

Run the following command :

```
>apktool.bat d sampleApp.apk
```

It also extracts the .smali file of all .class files, but which is difficult to read.

You can achieve this by using zip utility like 7-zip.

Using dex2jar

- to generate .jar file from .apk file, we need JD-GUI to view the source code from this .jar.

Run the following command :

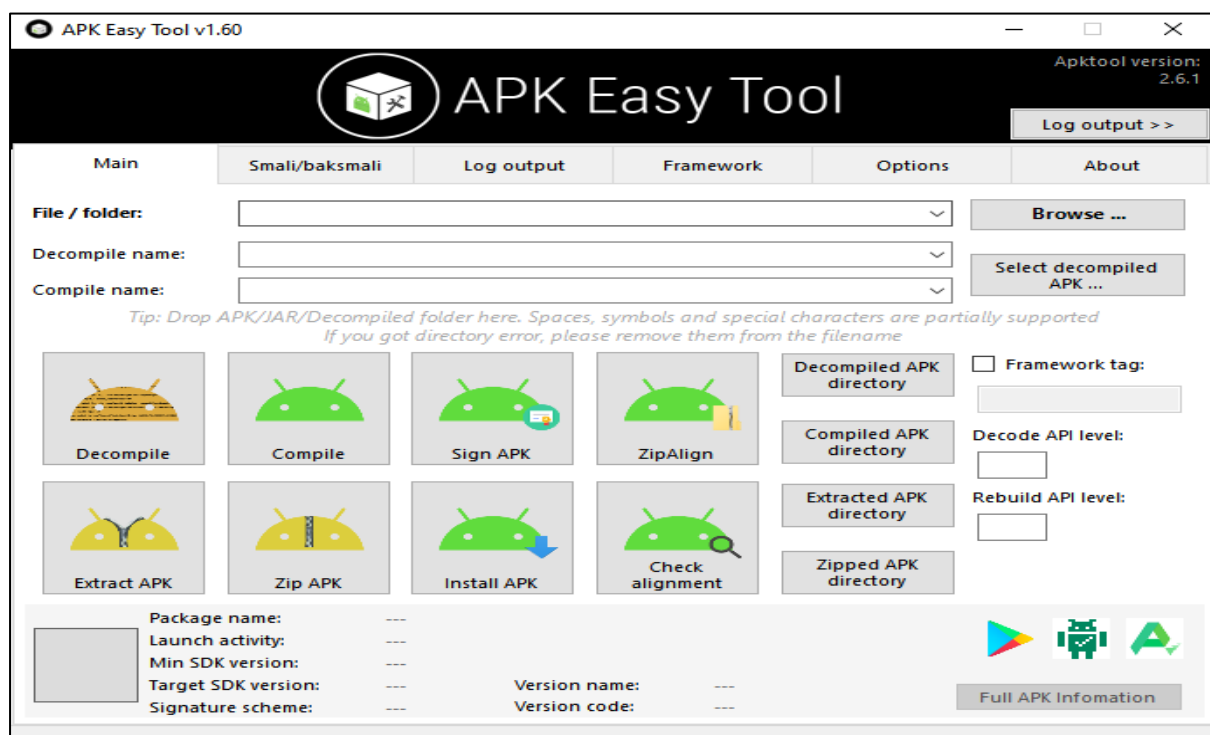
```
>dex2jar sampleApp.apk
```

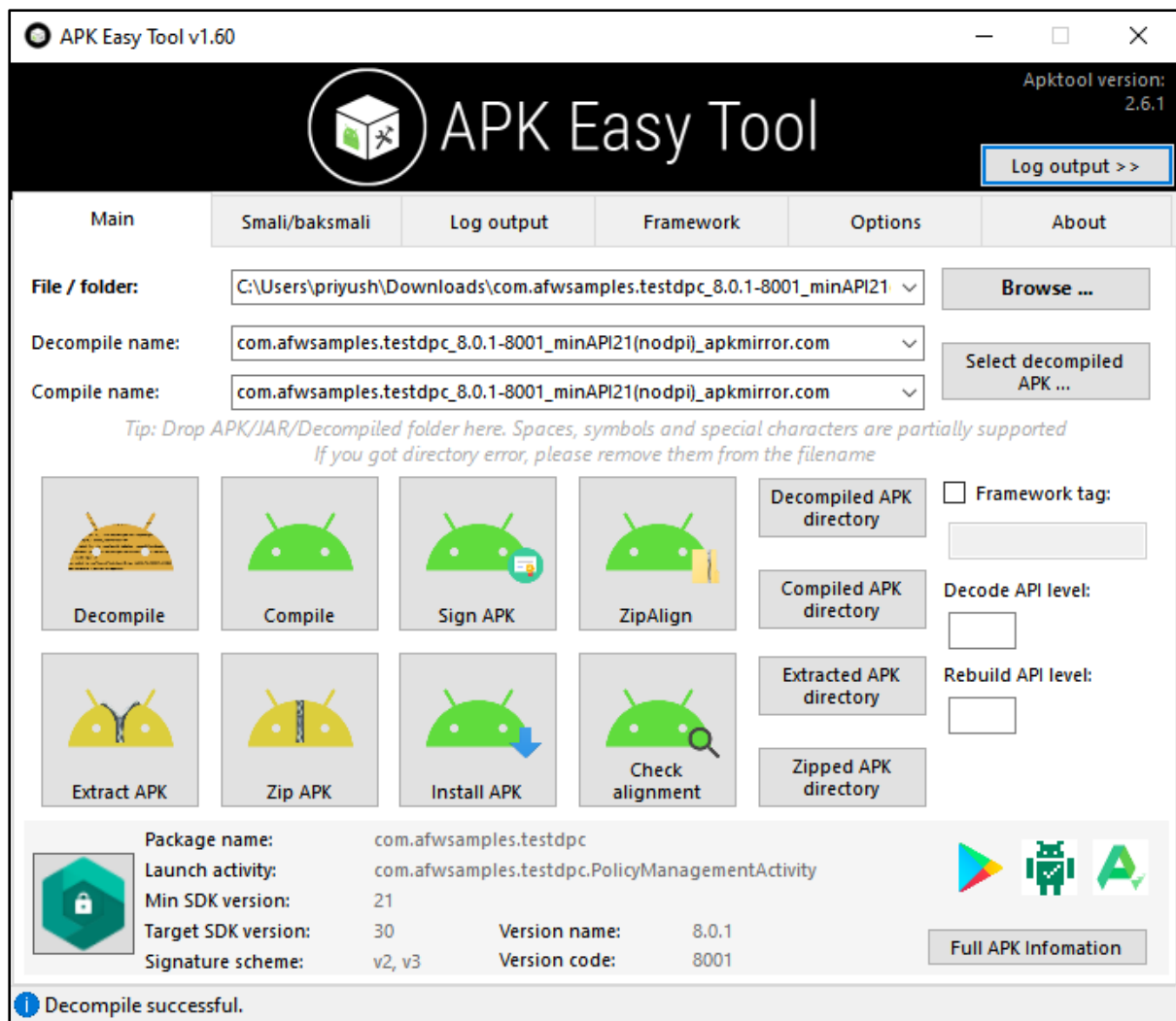
Decompiling .jar JD-GUI

it decompiles the .class files (obfuscated- in case of android app, but readable original code is obtained in case of other .jar file). i.e., we get .java back from the application.

Just Run the jd-gui.exe and File->Open to view java code from .jar or .class file.

Using ApkTool





Today (17)		
1-Decompiled APKs	06-04-2023 20:26	File folder
2-Recompiled APKs	06-04-2023 20:12	File folder
3-Extracted APKs	06-04-2023 20:12	File folder
4-Zipped APKs	06-04-2023 20:12	File folder
5-Baksmali	06-04-2023 20:12	File folder
6-Smali	06-04-2023 20:12	File folder
angular-1.8.2	06-04-2023 10:41	File folder
com	06-04-2023 17:58	File folder
layout	06-04-2023 17:57	File folder

Using dex2jar

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\priyush\Downloads\ex07\dex2jar-2.0>d2j-dex2jar -h
d2j-dex2jar -- convert dex to jar
usage: d2j-dex2jar [options] <file0> [file1 ... fileN]
options:
  -d,--debug-info           translate debug info
  -e,--exception-file <file> detail exception file, default is $current_dir/[fi
                             le-name]-error.zip
  -f,--force                force overwrite
  -h,--help                 Print this help message
  -n,--not-handle-exception not handle any exception thrown by dex2jar
  -nc,--no-code             output .jar file, default is $current_dir/[file-na
                             me]-dex2jar.jar
  -o,--output <out-jar-file>
  -os,--optimize-synchronized optimize-synchronized
  -p,--print-ir             print ir to System.out
  -r,--reuse-reg            reuse register while generate java .class file
  -s                        same with --topological-sort/-ts
  -ts,--topological-sort    sort block by topological, that will generate more
                             readable code, default enabled
version: reader-2.0, translator-2.0, ir-2.0
```

```
C:\Users\priyush\Downloads\ex07\dex2jar-2.0>d2j-dex2jar "com.afwsamples.testdpc_8.0.1-8001_minAPI21(nodpi)_apkmirror.com.apk"
dex2jar com.afwsamples.testdpc_8.0.1-8001_minAPI21(nodpi)_apkmirror.com.apk -> .\com.afwsamples.testdpc_8.0.1-8001_minAPI21(nodpi)_apkmirror.com-dex2jar.jar
C:\Users\priyush\Downloads\ex07\dex2jar-2.0>
```

Decompiling .jar JD-GUI



Conclusion: - Hence successfully performed Reversing Android applications (APKs) APKTOOL, dex2jar and JD-GUI.



DOP: / /2023

DOS: / /2023

Experiment No:8

Title: Reversing Android applications (APKs) APKTOOL, dex2jar and JD-GUI.

Theory:

What is DIVA?

DIVA (Damn insecure and vulnerable App) is an App intentionally designed to be insecure. We are releasing the Android version of Diva. We thought it would be a nice way to start the year by contributing something to the security community. The aim of the App is to teach developers/QA/security professionals, flaws that are generally present in the Apps due poor or insecure coding practices. If you are reading this, you want to either learn App pentesting or secure coding and I sincerely hope that DIVA solves your purpose. So, sit back and enjoy the ride.

Who can use Diva?

The idea originated, from a developer's perspective. The Android security training for developers becomes slightly boring with lot of theory and not much hands-on. SO, I created DIVA for our Android developer training. Diva gamifies secure development learning. With that said, it is an excellent learning tool for aspiring Android penetration testers and security professionals as it gives an insight into app vulnerabilities including the source code. To sum it up:

- Android App developers
- Android Penetration testers
- Security professionals
- Students

What is included in Diva?

I tried to put as much vulnerabilities as possible in a short period of time. I am sure I have missed out on some vulnerabilities. Please ping me if you know of a good vulnerability that can be included in Diva. It covers common vulnerabilities in Android apps ranging from insecure storage, input validation to access control issues. I have also included few vulnerabilities in native code, which makes it more interesting from the perspective of covering both Java and C vulnerabilities. Current Challenges include:

1. Insecure Logging
2. Hardcoding Issues – Part 1
3. Insecure Data Storage – Part 1
4. Insecure Data Storage – Part 2
5. Insecure Data Storage – Part 3
6. Insecure Data Storage – Part 4
7. Input Validation Issues – Part 1
8. Input Validation Issues – Part 2
9. Access Control Issues – Part 1



10. Access Control Issues – Part 2
11. Access Control Issues – Part 3
12. Hardcoding Issues – Part 2
13. Input Validation Issues – Part 3

How to compile Diva?

- Download the source
- Open the project in Android Studio
- For Native library – open command line
- `$ cd /app/src/main/jni`
- `$ make` (This needs to be done only once, unless you make changes to the native code – in which case run `\"make clean && make\"`)
- This will compile the native library and copy all the compiled versions in directory `jniLibs` which is required when building the app
- From the menu bar: Build->Make Project or Run->Run App

How to run Diva?

- Download the app
- On your phone settings. Go to security and check Unknown Sources checkbox. This allows you to install apps outside of play store. You don't need to do this if you are installing the app on an emulator.
- Connect your phone to the computer (make sure USB debugging is enabled on your phone) or run the emulator.
- `cd`
- `adb install`
- Start playing.