



SILVER OAK UNIVERSITY
COLLEGE OF TECHNOLOGY



DEGREE ENGINEERING
MOBILE APPLICATION DEVELOPMENT
(1010043367)
6th SEMESTER

Laboratory Manual

Prepared By: Dr. Jay Dave

Reference no: SOU_1010043367_LM_2024_1



DEPARTMENT OF COMPUTER ENGINEERING

VISION

- ❖ To create competent professionals in the field of Computer Engineering and promote research with a motive to serve as a valuable resource for the IT industry and society.

MISSION

- ❖ To produce technically competent and ethically sound Computer Engineering professionals by imparting quality education, training, hands on experience and value based education.
- ❖ To inculcate ethical attitude, sense of responsibility towards society and leadership ability required for a responsible professional computer engineer.
- ❖ To pursue creative research, adapt to rapidly changing technologies and promote self learning approaches in Computer Engineering and across disciplines to serve the dynamic needs of industry, government and society.

Program Educational Objectives (PEO):

PEO1: To provide fundamental knowledge of science and engineering for an IT professional and to equip them with proficiency of mathematical foundations and algorithmic principles and inculcate competent problem-solving ability.

PEO2: To implant ability in creativity & design of IT systems and transmit knowledge and skills to analyze, design, test and implement various software applications.

PEO3: To exhibit leadership capability, triggering social and economical commitment and inculcate community services.

PEO4: To inculcate professional-social ethics, teamwork in students and acquaint them with requisite technical and managerial skills to attain a successful career.

PROGRAM OUTCOMES (POs)

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PREFACE

MOBILE APPLICATION DEVELOPMENT PRACTICAL BOOK

DEPARTMENT OF CE

Before introducing Android Programming, we should understand that Android is an open source and Linux-based operating system for mobile devices such as smart phones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Android is a complete set of software for mobile devices such as tablet computers, notebooks, smart phones, electronic book readers, set-top boxes etc. It contains a **Linux-based Operating System, middleware** and **key mobile applications**. It can be thought of as a mobile operating system. But it is not limited to mobile only. It is currently used in various devices such as mobiles, tablets, televisions etc. It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc. To learn Android Studio, you must have the basic knowledge of Java programming language. 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.

We acknowledge the authors and publishers of all the books which we have consulted while developing this Practical book. Hopefully this Practical Book will serve the purpose for which it has been developed.

1. Be prompt in arriving at the laboratory and always come well-prepared for the experiment.
2. Without Prior permission do not enter into the Laboratory.
3. While entering into the LAB students should wear their ID cards and arrange your shoes in the proper way.
4. Students should maintain silence inside the laboratory.
5. Every student should have his/her individual copy of the Database Management Systems Book.
6. Every student has to prepare the notebooks specifically reserved for the SE Practical work.
7. Every student has to necessarily bring his/her Database Management Systems Book.
8. Find the answers of all the questions mentioned under the section 'Find the Answers' at the end of each experiment in the Database Management Systems Practical Book.
9. Finally record the verified output along with the calculation and results in the Database Management Systems Notebook.
10. Do not forget to get the information of your next allotment (the experiment which is to be performed by you in the next laboratory session) before leaving the laboratory from the Technical Assistant.
11. Lab Manual records need to be submitted on or before date of submission.
12. The grades for the Database Management Systems Practical course work will be awarded based on your performance in the laboratory, regularity, obtaining of experiments results, lab quiz, and regular viva voice and end-term examination.
13. Students are restricted to not to use mobile phones inside the laboratory.
14. After completing the laboratory exercise make sure to switch off all the amenities properly.
15. Lab can be used in free time/lunch hours by the students who need to use the systems should take prior permission from the lab in-charge



TABLE OF CONTENT

Sr. No	Experiment Title	Page No		Date of Start	Date of Completi on	Sign	Mar ks (out of 10)	CO	PO
		From	To						
1	Android studio installation and Create a “Hello World” application. That will display “Hello World” in the middle of the screen using the Text View Widget in the red color.	9	11					1	1
2	Design Registration activity using necessary UI Components: Implement using the following layouts: 1. Linear Layout 2. Relative Layout 3. Constraint Layout	12	19					2	3
3	Create Activities & implement the following: a) Implicit intent b) Explicit Intent c) StartActivityForResult	20	28					2	1
4	Create Android applications that demonstrate the Android activity life cycle and concept of fragments.	29	40					2	3
5	Create Android applications that demonstrate the concept of the menu with navigation.	41	44					2	3
6	Create an Android app to display student details in Listview (using Adapter class).	45	49					2	3



7	Create an Android application that stores user Login credentials using Shared Preferences.	50	57					2	3
8	Create a registration application to store data using the Room database.	58	82					3	5
9	To implement parsing JSON data using the Retrofit library in Android application.	83	90					4	1
10	Develop an application for working with view animation.	91	94					5	1
11	Write a program in Dart that finds simple interest. Formula= $(p * t * r) / 100$.	95	97					5	3
12	Working with Widgets in Flutter (Text, padding, margin, and alignment).	98	101					5	3
13	Create a stateless widget and a stateful widget and use them in your main application.	102	105					5	3
14	Create an app with different basic layouts, such as Column, Row, and Container.	106	109					5	1
15	Create forms for user input with validation.	110	113					5	3



PRACTICAL – 1

DATE: _____

AIM: Android studio installation and Create a “Hello World” application. That will display “Hello World” in the middle of the screen using the Text View Widget in the red color.

Theory :

Prerequisites for this android project

Java: First of all you need to have the knowledge of Java Programming. Java programming plays a very important role as we will develop the app code in Java

XML: XML is another important part of our Android application. It will be used for the development of the user interface for the application

Android Studio: Android Studio is the backbone of our application, as we will develop our app using android studio. Android Virtual Device (AVD) is also shipped with android studio that will be helpful in testing whether the applications are working or not.

Prerequisites for this android project.

Java: First of all you need to have the knowledge of Java Programming. Java programming plays a very important role as we will develop the app code in Java

XML: XML is another important part of our android application. It will be used for the development of the user interface for the application

Android Studio: Android Studio is the backbone of our application, as we will develop our app using android studio. Android virtual device is also shipped with android studio that will be helpful in testing whether the applications are working or not

Code:

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

    <TextView
        android:id="@+id/helloWorldTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
android:text="Hello World"  
android:textColor="#FF0000"  
android:textSize="24sp"  
android:layout_centerInParent="true" />
```

```
</RelativeLayout>
```

```
// MainActivity.java
```

```
package com.example.helloworldapp;
```

```
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;
```

```
public class MainActivity extends AppCompatActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

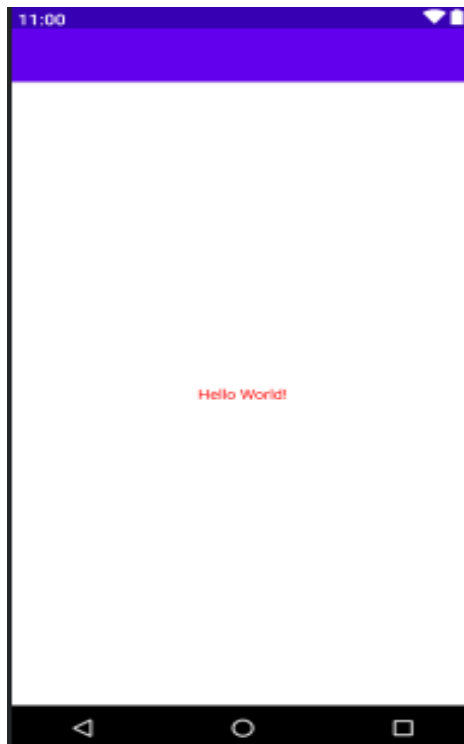
```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
```

```
    }
```

```
}
```

Output:





Post Practical Questions

1. Which of the following virtual machine is used by the Android operating system?
 - a. JVM
 - b. Dalvik Virtual Machine
 - c. Simple Virtual Machine
 - d. None of the above
2. Android is based on which of the following language?
 - a. Java
 - b. C++
 - c. C
 - d. .Net
3. Android was developed by.
 - a. Microsoft
 - b. Apple
 - c. Google
 - d. IBM
4. Does android support other languages than java?
 - a. Yes
 - b. No
 - c. May be
 - d. Can't say

Conclusion:

In conclusion we will learn use of the Text View Widget.

References:

1. <https://www.geeksforgeeks.org/working-with-the-textview-in-android/>
2. https://www.tutorialspoint.com/android/android_textview_control.htm

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-2

DATE: _____

AIM:- Design Registration activity using necessary UI Components: Implement using the following layouts:

- a. Linear Layout**
- b. Relative Layout**
- c. Constraint Layout**

Theory:

1.Linear Layout :

Linear layout is a simple layout used in android for layout designing. In the Linear layout all the elements are displayed in linear fashion means all the childs/elements of a linear layout are displayed according to its orientation. The value for orientation property can be either horizontal or vertical.

1.Types Of Linear Layout Orientation

There are two types of linear layout orientation:

1. Vertical
2. Horizontal

As the name specified these two orientations are used to arrange their child one after the other, in a line, either vertically or horizontally. Let's describe these in detail.

2. Gravity: The gravity attribute is an optional attribute which is used to control the alignment of the layout like left, right, center, top, bottom etc.

3. layout_weight: The layout_weight attribute specify each child control's relative importance within the parent linear layout.

4. weightSum: weightSum is the sum up of all the child attributes weight. This attribute is required if we define the weight property of the childs.

2.Relative Layout :

Theory :

Android RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent.

Sr.No.	Attribute & Description
1	android:layout_above Positions the bottom edge of this view above the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name"
2	android:layout_alignBottom Makes the bottom edge of this view match the bottom edge of the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".
3	android:layout_alignLeft Makes the left edge of this view match the left edge of the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".
4	android:layout_alignParentBottom If true, makes the bottom edge of this view match the bottom edge of the parent. Must be a boolean value, either "true" or "false".
5	android:layout_alignParentEnd If true, makes the end edge of this view match the end edge of the parent. Must be a boolean value, either "true" or "false".
6	android:layout_alignParentLeft If true, makes the left edge of this view match the left edge of the parent. Must be a boolean value, either "true" or "false".
7	android:layout_alignParentRight If true, makes the right edge of this view match the right edge of the parent. Must be a boolean value, either "true" or "false".
8	android:layout_alignParentStart If true, makes the start edge of this view match the start edge of the parent. Must be a boolean value, either "true" or "false".
9	android:layout_alignParentTop

	If true, makes the top edge of this view match the top edge of the parent. Must be a boolean value, either "true" or "false".
10	android:layout_alignRight Makes the right edge of this view match the right edge of the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".
11	android:layout_alignStart Makes the start edge of this view match the start edge of the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".
12	android:layout_alignTop Makes the top edge of this view match the top edge of the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".
13	android:layout_below Positions the top edge of this view below the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".
14	android:layout_centerHorizontal If true, centers this child horizontally within its parent. Must be a boolean value, either "true" or "false".
15	android:layout_centerInParent If true, centers this child horizontally and vertically within its parent. Must be a boolean value, either "true" or "false".
16	android:layout_centerVertical If true, centers this child vertically within its parent. Must be a boolean value, either "true" or "false".
17	android:layout_toEndOf Positions the start edge of this view to the end of the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".
18	android:layout_toLeftOf

	Positions the right edge of this view to the left of the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".
19	android:layout_toRightOf Positions the left edge of this view to the right of the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".
20	android:layout_toStartOf Positions the end edge of this view to the start of the given anchor view ID and must be a reference to another resource, in the form "@+[package:]type:name".

Practical Code:

```
<!-- activity_registration_linear.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Name"
        android:layout_margin="16dp"/>

    <EditText
        android:id="@+id/editTextEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:layout_margin="16dp"/>

    <Button
```



```
android:id="@+id/buttonRegister"  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="Register"  
android:layout_margin="16dp"/>
```

```
</LinearLayout>
```

```
<!-- activity_registration_relative.xml -->
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:padding="16dp">
```

```
<EditText  
    android:id="@+id/editTextName"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="Name"  
    android:layout_marginBottom="16dp"/>
```

```
<EditText  
    android:id="@+id/editTextEmail"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:hint="Email"  
    android:layout_below="@id/editTextName"  
    android:layout_marginBottom="16dp"/>
```

```
<Button  
    android:id="@+id/buttonRegister"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Register"  
    android:layout_below="@id/editTextEmail"/>
```




```
</RelativeLayout>
<!-- activity_registration_constraint.xml -->
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <EditText
        android:id="@+id/editTextName"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Name"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:layout_marginBottom="16dp"/>

    <EditText
        android:id="@+id/editTextEmail"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="Email"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@id/editTextName"
        android:layout_marginBottom="16dp"/>

    <Button
        android:id="@+id/buttonRegister"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Register"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
```



```
app:layout_constraintTop_toBottomOf="@id/editTextEmail"/>
```

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

Post Practical Questions

1. To display text which control you will use?
 - a. Edittext
 - b. **Textview**
 - c. Label
 - d. ViewText
2. In the end, it is closed by ?
 - a. >
 - b. ?>
 - c. !>
 - d. **/>**
3. `wrap` means it will occupy only that much space as required for its content to display.
 - a. **Wrap content**
 - b. Match content
 - c. Wrap parent
 - d. Match parent
4. Which of the following layout in android aligns all children either vertically or horizontally?
 - a. Relative
 - b. Table
 - c. Frame
 - d. **Linear**

References:

1. <https://www.androidcodefinder.com/blog/2018/06/05/login-screen-design-in-android-studio/>
2. <https://codedost.com/get-started-android/android-programs/android-program-design-login-screen-using-relative-layout/>
3. <https://developer.android.com/develop/ui/views/layout/declaring-layout>



**SILVER OAK
UNIVERSITY**
EDUCATION TO INNOVATION

Conclusion:

These layouts define a registration form with EditText fields for Name and Email, and a Button for registering. The layouts are implemented using Linear Layout, Relative Layout, and Constraint Layout respectively

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-3

DATE: _____

AIM: Create Activities & implement the following:

1. **Implicit intent**
2. **Explicit Intent**
3. **StartActivityForResult**

Theory:

Android Intent is the message that is passed between components such as activities, content providers, broadcast receivers, services etc.

It is generally used with the `startActivity()` method to invoke activity, broadcast receivers etc.

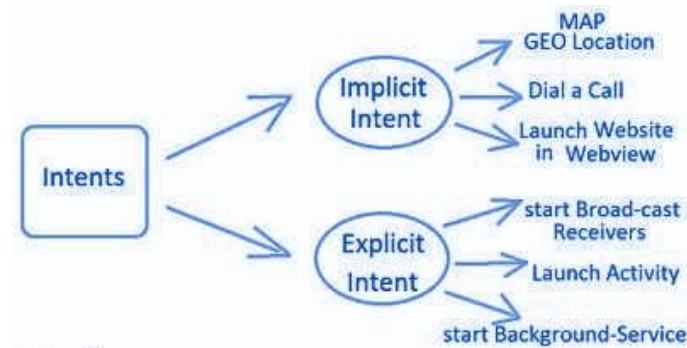
The **dictionary meaning** of intent is *intention or purpose*. So, it can be described as the intention to do action.

Android intents are mainly used to:

- Start the service
- Launch an activity
- Display a web page
- Display a list of contacts
- Broadcast a message
- Dial a phone call etc.

Types of Android Intents :

There are two types of intents in android: **Implicit** and **Explicit**.



1) Implicit Intent:

Implicit Intent doesn't specify the component. In such case, intent provides information of available components provided by the system that is to be invoked

For example, you may write the following code to view the webpage:

```
Intent intent=new Intent(Intent.ACTION_VIEW);  
intent.setData(Uri.parse("webpage url"));  
startActivity(intent);
```

2) Explicit Intent:

Explicit Intent specifies the component. In such case, intent provides the external class to be invoked.

Android Explicit intent specifies the component to be invoked from activity. In other words, we can call another activity in android by explicit intent.

We can also pass the information from one activity to another using explicit intent.

Android calling one activity from another activity example:

```
Intent i = new Intent(getApplicationContext(), ActivityTwo.class);  
startActivity(i);
```

Practical Code:

```
<!-- activity_registration_linear.xml -->

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:orientation="vertical">

    <EditText

        android:id="@+id/editTextName"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:hint="Name"/>

    <EditText

        android:id="@+id/editTextEmail"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:hint="Email"/>

    <Button

        android:id="@+id/buttonRegister"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"

        android:text="Register"/>
```



</LinearLayout>

<!-- activity_registration_relative.xml -->

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="match_parent"

android:layout_height="match_parent">

<EditText

android:id="@+id/editTextName"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:hint="Name"

android:layout_marginTop="50dp"/>

<EditText

android:id="@+id/editTextEmail"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:hint="Email"

android:layout_below="@id/editTextName"

android:layout_marginTop="20dp"/>

<Button

android:id="@+id/buttonRegister"

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:text="Register"

android:layout_below="@id/editTextEmail"



```
        android:layout_marginTop="20dp"/>
</RelativeLayout>
<!-- activity_registration_constraint.xml -->
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent">
<EditText
    android:id="@+id/editTextName"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Name"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:layout_marginTop="50dp"/>
<EditText
    android:id="@+id/editTextEmail"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Email"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
```



```
app:layout_constraintTop_toBottomOf="@id/editTextName"
```

```
android:layout_marginTop="20dp"/>
```

```
<Button
```

```
    android:id="@+id/buttonRegister"
```

```
    android:layout_width="0dp"
```

```
    android:layout_height="wrap_content"
```

```
    android:text="Register"
```

```
    app:layout_constraintStart_toStartOf="parent"
```

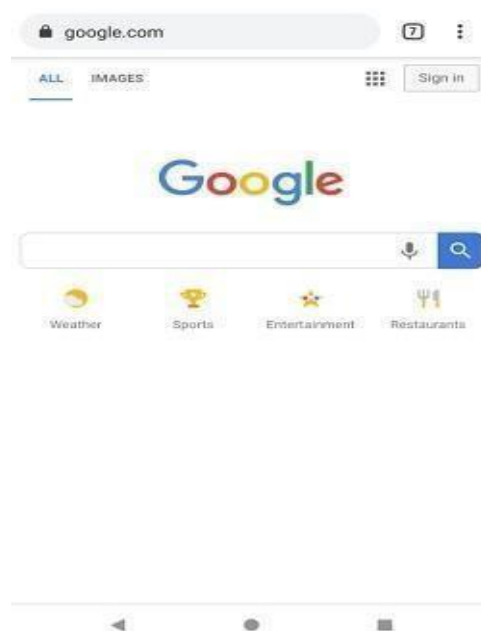
```
    app:layout_constraintEnd_toEndOf="parent"
```

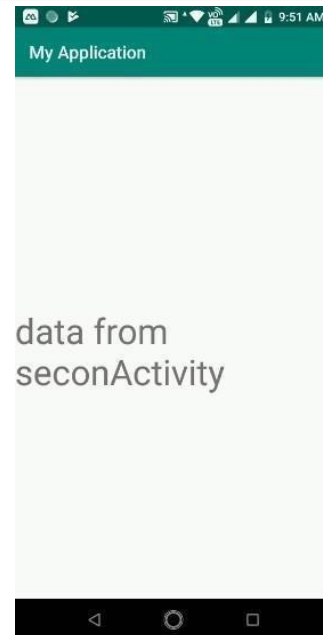
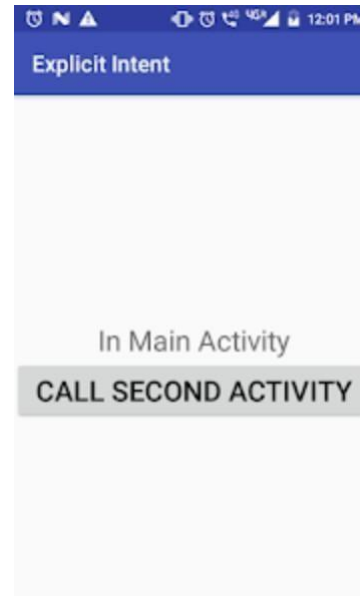
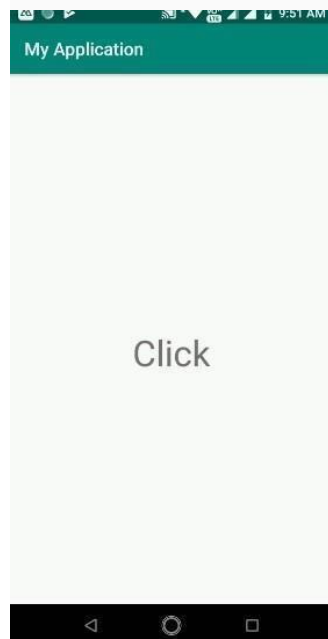
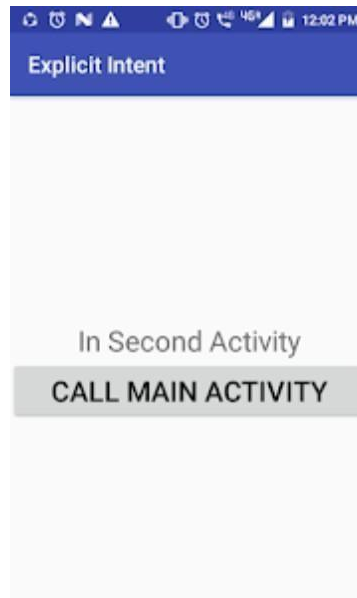
```
    app:layout_constraintTop_toBottomOf="@id/editTextEmail"
```

```
    android:layout_marginTop="20dp"/>
```

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

Output:







Post Practical Questions

1. How can we stop the services in android?
 - a. **Stopself() and stopservice()**
 - b. Finish()
 - c. System.exit()
 - d. None of the above
2. What is the use of a content provider in android?
 - a. Storing data in database
 - b. **Sharing data between application**
 - c. Sending data from one app to another app
 - d. None
3. An Android _____ is an abstract description of an operation to be performed. It can be used with startActivity to launch an Activity.
 - a. Filters
 - b. **Intent**
 - c. Service
 - d. Broadcast receiver
4. Which of the following actions will display the phone dialer with the given number filled in?
 - a. ACTION_VIEW tel:123
 - b. ACTION_SET
 - c. **ACTION_DIAL tel:123**
 - d. None

References:

1. <https://www.geeksforgeeks.org/android-implicit-and-explicit-intents-with-examples/>
2. <https://www.tutorialspoint.com/what-is-an-intent-in-android#:~:text=An%20intent%20is%20to%20perform,Implicit%20Intents%20and%20Explicit%20Intents.>
3. <https://abhiandroid.com/programming/intent-in-android>



**SILVER OAK
UNIVERSITY**
EDUCATION TO INNOVATION

Conclusion:

By understanding and implementing these concepts, developers can create versatile and interactive Android applications that seamlessly navigate between different components and handle various user interactions. Intents provide a flexible mechanism for achieving diverse functionalities within an Android app, enhancing its usability and user experience.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-4

DATE: _____

AIM: Create Android applications that demonstrate the Android activity life cycle and concept of fragments.

Theory :

Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

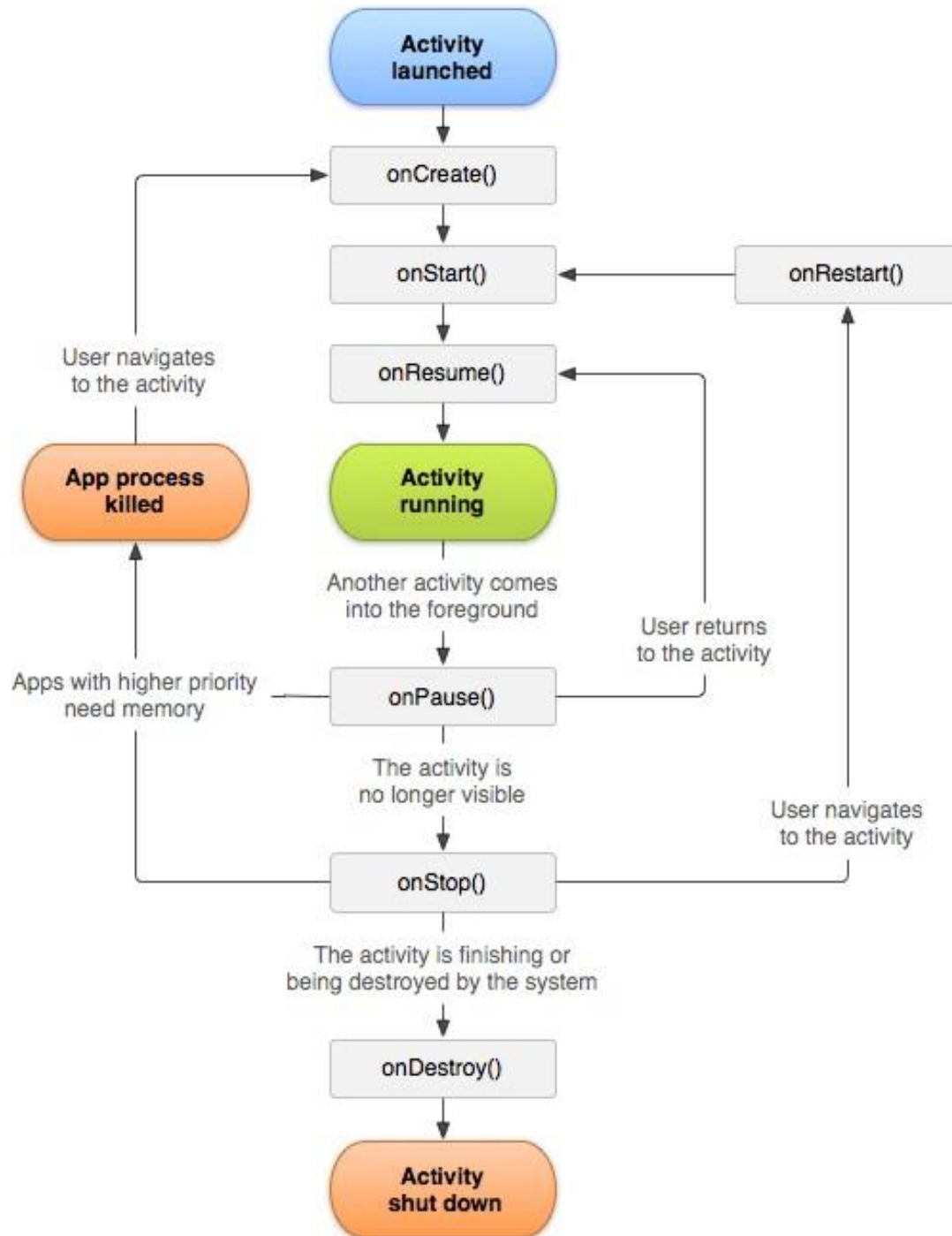
An activity is the single screen in android. It is like a window or frame of Java.

By the help of activity, you can place all your UI components or widgets in a single screen.

The 7 lifecycle method of Activity describes how activity will behave at different states.

Android Activity Lifecycle methods:

Method	Description
onCreate()	called when activity is first created.
onStart()	called when activity is becoming visible to the user.
onResume()	called when activity will start interacting with the user.
onPause()	called when activity is not visible to the user.
onStop()	called when activity is no longer visible to the user.
onRestart()	called after your activity is stopped, prior to start.
onDestroy()	called before the activity is destroyed.



Fragment Theory :

Android Fragment is the part of activity, it is also known as sub-activity. There can be more than one fragment in an activity. Fragments represent multiple screen inside one activity.

Android fragment lifecycle is affected by activity lifecycle because fragments are included in activity.

Each fragment has its own life cycle methods that is affected by activity life cycle because fragments are embedded in activity.

The `FragmentManager` class is responsible to make interaction between fragment objects.

Android Fragment Lifecycle

The lifecycle of android fragment is like the activity lifecycle. There are 12 lifecycle methods for fragment.

There are some primary classes related to Fragment's are:

- 1. FragmentActivity:** The base class for all activities using compatibility based Fragment (and loader) features.
- 2. Fragment:** The base class for all Fragment definitions
- 3. FragmentManager:** The class for interacting with Fragment objects inside an activity
- 4. FragmentTransaction:** The class for performing an atomic set of Fragment operations such as Replace or Add a Fragment.

Add a fragment via XML:

```
<!-- res/layout/example_activity.xml -->
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container_view"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:name="com.example.ExampleFragment" />
```

Add a fragment programmatically:

```
public class FirstFragment extends Fragment {  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
        Bundle savedInstanceState) {  
        // Inflate the layout for this fragment  
        return inflater.inflate(R.layout.fragment_first, container, false);  
    }  
}
```

Practical Code:

```
package com.example.activitylifecycle;  
  
import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
import android.util.Log;  
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Log.d("Lifecycle", "onCreate() called");  
    }  
    @Override  
    protected void onStart() {  
        super.onStart();  
        Log.d("Lifecycle", "onStart() called");  
    }  
  
    @Override  
    protected void onResume() {  
        super.onResume();  
        Log.d("Lifecycle", "onResume() called");  
    }  
    @Override  
    protected void onPause() {  
        super.onPause();  
        Log.d("Lifecycle", "onPause() called");  
    }  
    @Override  
    protected void onStop() {  
        super.onStop();  
    }  
}
```



```

    Log.d("Lifecycle", "onStop() called");
}
@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d("Lifecycle", "onDestroy() called");
}
}
package com.example.fragmentsdemo;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
<!-- fragment_demo.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:id="@+id/textViewFragment"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="This is a fragment"
        android:textSize="24sp"
        android:layout_gravity="center"/>
</LinearLayout>
activity_main.xml

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="codedost.fragment.MainActivity">
    <fragment
        android:layout_width="match_
        parent"
        android:layout_height="0dp"

```

```
        android:id="@+id/fr1"
        android:layout_weight="1"
        android:background="#0f0"
        android:name="codedost.fragment.first"></fragment>
    <FrameLayout
        android:layout_width="match_parent"
        " android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@color/colorA
        ccent" android:id="@+id/fr2"
        android:name="codedost.fragment.second"></FrameLayout>
</LinearLayout>
```

first_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:background="#0f0"
    android:layout_height="match_parent">
    <EditText
        android:layout_width="match_p
        arent"
        android:layout_height="wrap_c
        ontent" android:ems="10"
        android:textColorHint="#fff"
        android:hint="Enter text"
        android:id="@+id/edittext"></E
        ditText>
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        " android:text="Send"
        android:layout_below="@+id/edittext"
        android:layout_alignParentStart="true
        " />
</RelativeLayout>
```

second_fragment.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <TextView android:layout_width="match_parent" android:layout_height="wrap_content"
```



```
android:layout_centerHorizontal="true" android:layout_centerVertical="true"
android:textSize="27sp"
android:hint="You will receive the text here!" android:textColorHint="#fff"
android:textColor="#fff" android:gravity="center" android:id="@+id/rtxt"/>
</RelativeLayout>
```

MainActivity.java

```
package codeodst.fragment;
import android.app.FragmentManager; import android.app.FragmentTransaction;
import android.support.v7.app.AppCompatActivity; import android.os.Bundle;
public class MainActivity extends AppCompatActivity implements
    first.OnFragmentInteractionListener { @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        placeFragmentDynamically();
    }

    private void
        placeFragmentDynamically(){
        FragmentManager fm =
            getFragmentManager();
        FragmentTransaction ft =
            fm.beginTransaction();
        ft.add(R.id.fr2,new second());
        ft.commit();
    }

    @Override
    public void onFragmentInteraction(String msg)
    {
        second s =
            (second)getFragmentManager().findFragmentById(R.id.fr2)
            ; s.receiveinfo(msg);
    }
}
```

first.java

```
package codedost.fragment;
import android.content.Context;
import android.os.Bundle;
import android.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
```



```
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
public class first extends Fragment {
    EditText mMessage;
    Button msubmitbtn;
    private OnFragmentInteractionListener mListener;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
        container, Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.first_fragment,
            container, false); mMessage = (EditText)
            view.findViewById(R.id.edittext); msubmitbtn = (Button)
            view.findViewById(R.id.button1);
            msubmitbtn.setOnClickListener(new
            View.OnClickListener() {

                @Override
                public void onClick(View v) {
                    String mSend =
                    mMessage.getText().toString();
                    if(mSend.isEmpty())
                    {
                        Toast.makeText(getContext(),"You have not sent anything",Toast.LENGTH_SHORT).show();
                    }
                    else
                        onButtonClick(mSend);
                }
            });

        return view;//return view
    }

    public void onButtonClick(String msg)
    {
        if(mListener!=null)
        {
```



```
        mListener.onFragmentInteraction(msg);
    }
}

@Override
public void onAttach(Context context)
{
    super.onAttach(context);
    try {
        mListener=(OnFragmentInteractionListener)context ;
    }
    catch (ClassCastException e){
    }
}

public interface OnFragmentInteractionListener
{
    void onFragmentInteraction(String msg);
}
}
```

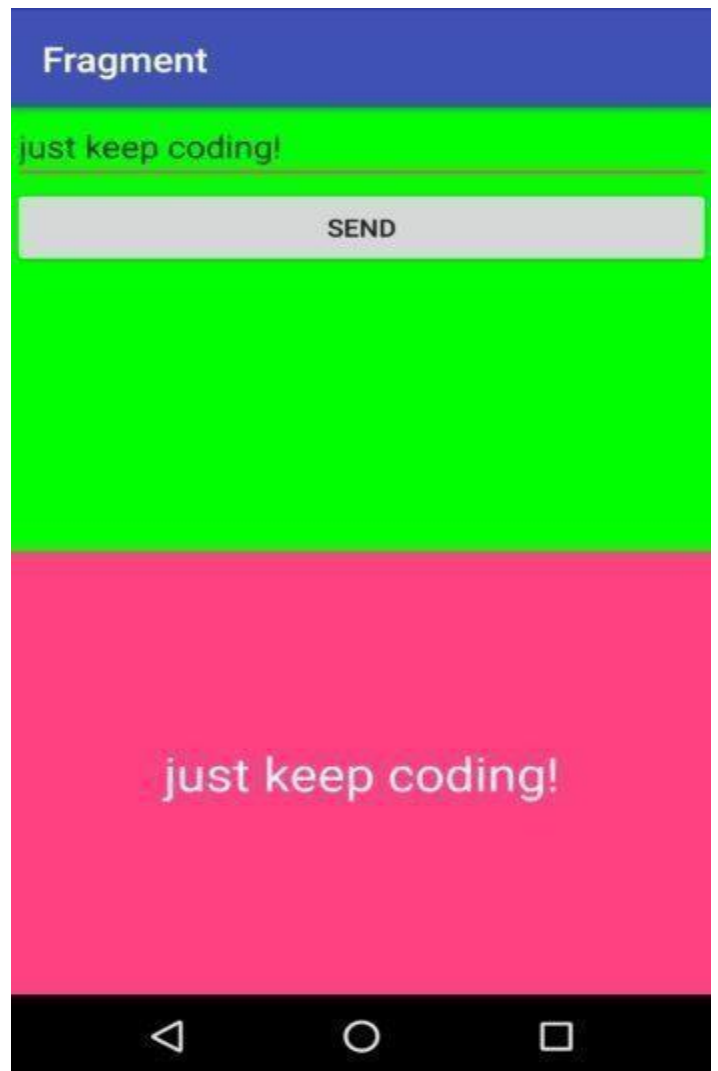
second.java

```
package
codedost.fragment; import
android.os.Bundle; import
android.app.Fragment;
import
android.view.LayoutInflater;
import android.view.View;
import
android.view.ViewGroup;
import
android.widget.TextView;

public class second extends
    Fragment { private TextView
    mReceivedmsg;
    @Override
    public View onCreateView(LayoutInflater inflater , ViewGroup container , Bundle savedInstanceState)
    {
        View view = inflater.inflate(R.layout.second_fragment,container,false);
        mReceivedmsg= (TextView) view.findViewById(R.id.rtxt);
        return view;
    }
    public void receivedinfo(String txt)
```

```
{  
    mReceivedmsg.setText(txt);  
}  
}
```

Output:





Post Practical Questions

1. What is an Activity in Android?
 - a. Android class
 - b. Android package
 - c. A single screen in application with supporting java code
 - d. None of the above
2. How can we kill an activity in android?
 - a. Finish()
 - b. finishActivity(int requestCode)
 - c. both a and b
 - d. neither a nor b
3. Which of the following is not an activity lifecycle callback method?
 - a. onClick() method
 - b. onCreate() method
 - c. onStart() method
 - d. onBackPressed() method
4. Which of the following is contained in the src folder?
 - a. XML
 - b. Java source code
 - c. Manifest
 - d. None of the above

References:

1. <https://www.geeksforgeeks.org/introduction-to-activities-in-android/>
2. <https://developer.android.com/reference/android/app/Activity>
3. <https://www.geeksforgeeks.org/introduction-fragments-android/>
4. <https://www.tutorialspoint.com/fragment-tutorial-with-example-in-android-studio>



**SILVER OAK
UNIVERSITY**
EDUCATION TO INNOVATION

Conclusion:

In conclusion, understanding the Android activity lifecycle and the concept of fragments are essential for Android developers to build robust and user-friendly applications.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-5

DATE: _____

AIM: Create Android applications that demonstrate the concept of the menu with navigation.

Theory:

There are three types of menus in Android: **Popup**, **Contextual** and **Options**.

Android Option Menu : **Android Option Menus** are the primary menus of android. They can be used for settings, search, delete items etc.

Here, we are going to see two examples of option menus. First, the simple option menus and second, options menus with images.

Here, we are inflating the menu by calling the **inflate()** method of **MenuInflater** class. To perform event handling on menu items, you need to override **onOptionsItemSelected()** method of Activity class.

Each menu must have an XML file related to it which defines its layout. These are the tags associated with the menu option:

<menu> - This is the container element for your menu (similar to **LinearLayout**)

<item> - This denotes an item and is nested inside of the menu tag. Be aware that an item element can hold a **<menu>** element to represent a submenu

<group> - This is used to signify a certain property or feature to a couple of menu items (I.E. state/visibility)

Each menu item has various attributes associated with it.

- **id** - This is a unique identifier for the item in the menu. You can use this to see exactly which item the user clicked
- **icon** - If you want to show an icon associated with that menu item
- **title** - Text that will be shown in the menu for that item
- **showAsAction** - This attribute should only be used when using a menu in an activity that uses an application bar (or as it is also referred to, the action bar). It controls when and how this item should appear as an action in the application bar. There are five values: **always**, **never**, **ifRoom**, **withText**, and **collapseActionView**.

Practical Code:

XML Code:-

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto">

<item
android:id="@+id/mob" android:title="Mobile" app:showAsAction="never"/>
</item>
```



```
android:id="@+id/lp" android:title="Laptop" app:showAsAction="never"/>
```

Activity Code:

```
package codedost.optionsmenu;
```

```
import android.support.v7.app.AppCompatActivity; import  
android.os.Bundle; import android.view.Menu;  
import android.view.MenuInflater; import android.view.MenuItem; import  
android.widget.Toast; protected void onCreate(Bundle savedInstanceState) {  
super.onCreate(savedInstanceState); setContentView(R.layout.activity_main);  
}
```

```
@Override
```

```
public boolean onCreateOptionsMenu(Menu menu) {
```

```
MenuInflater menuInflater=getMenuInflater(); menuInflater.inflate(R.menu.option_menu,menu);  
//setTitle("SELECT A PRODUCT"); return true;  
}
```

```
@Override
```

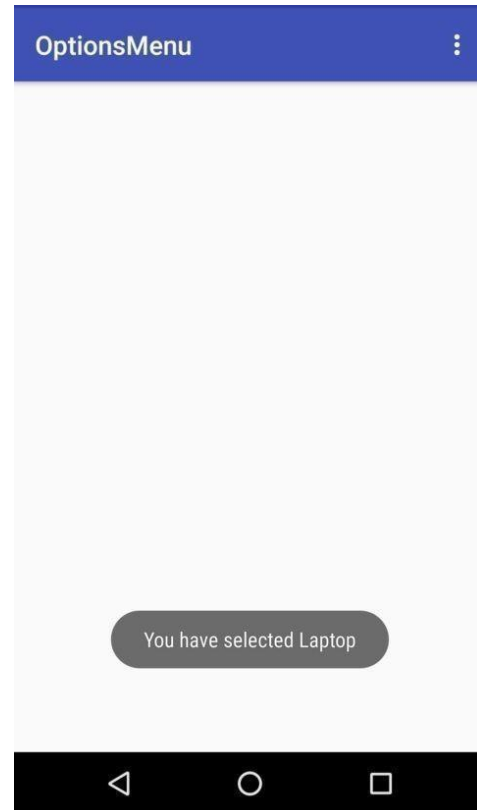
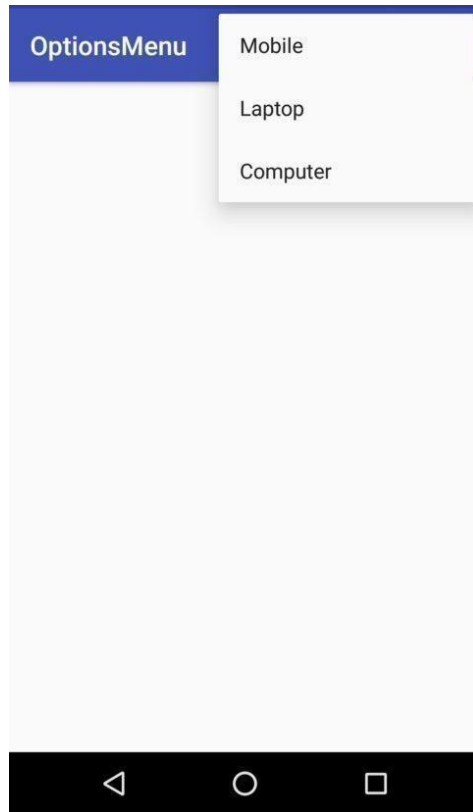
```
public boolean onOptionsItemSelected(MenuItem item) {
```

```
switch (item.getItemId())
```

```
{
```

```
case R.id.mob:
```

```
Toast.makeText(this,"You have selected Mobile",Toast.LENGTH_SHORT).show(); return true;
```

Output:**Post Practical Questions**

1. __ is a piece of an activity which enables more modular activity design.
 - a. **Fragment**
 - b. Sub activity
 - c. Intents
 - d. Filters
2. Which method is called once the fragment gets visible?
 - a. **onStart**
 - b. onPause
 - c. onResume
 - d. onStop



3. Which method is called Fragment going to be stopped?
 - a. onDestroyView
 - b. onPause
 - c. onResume
 - d. **onStop**
4. Which method Fragment becomes active?
 - a. onPause
 - b. **onResume**
 - c. onStart
 - d. onCreate
5. Fragments having a special list view is called as?
 - a. **List**
 - b. View
 - c. Frame
 - d. Special

References:

1. <https://developer.android.com/develop/ui/views/components/menus>

Conclusion:

In conclusion, we will learn the concept of the menu with navigation.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-6

DATE: _____

AIM: Create an Android app to display student details in Listview (using Adapter class).

Practical Code:

// Student.java

```
public class Student {  
    private String name;  
    private String rollNumber;  
  
    public Student(String name, String rollNumber) {  
        this.name = name;  
        this.rollNumber = rollNumber;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public String getRollNumber() {  
        return rollNumber;  
    }  
}
```

// StudentAdapter.java

```
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ArrayAdapter;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;  
  
import java.util.List;  
  
public class StudentAdapter extends ArrayAdapter<Student> {  
  
    public StudentAdapter(Context context, List<Student> students) {  
        super(context, 0, students);  
    }  
}
```



```
@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent)
{
    if (convertView == null) {
        convertView = LayoutInflater.from(getContext()).inflate(R.layout.list_item_student, parent,
false);
    }

    Student student = getItem(position);

    TextView nameTextView = convertView.findViewById(R.id.textViewName);
    TextView rollNumberTextView = convertView.findViewById(R.id.textViewRoll);

    if (student != null) {
        nameTextView.setText(student.getName());
        rollNumberTextView.setText(student.getRollNumber());
    }

    return convertView;
}
}
```

```
<!-- list_item_student.xml -->
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/textViewName"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/textViewRoll"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="16sp" />

</LinearLayout>
```



```
// MainActivity.java
import android.os.Bundle;
import android.widget.ListView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

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

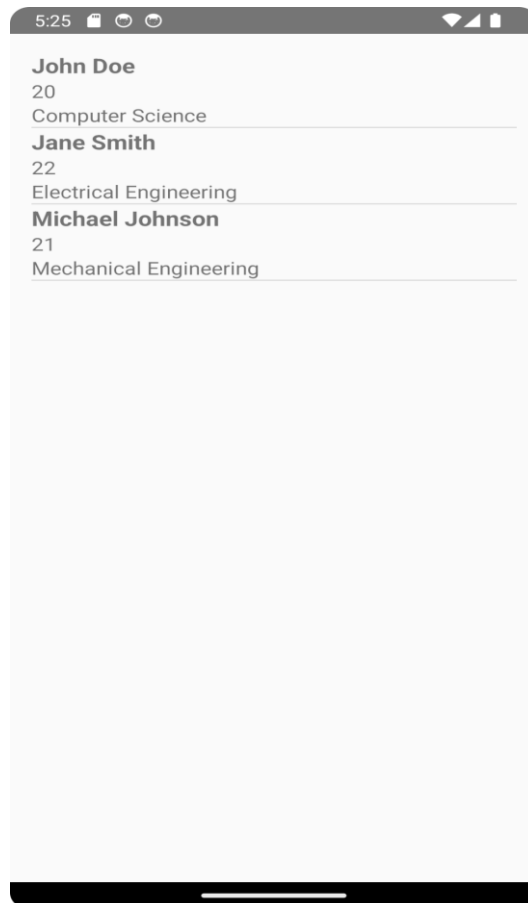
        List<Student> students = new ArrayList<>();
        students.add(new Student("John Doe", "101"));
        students.add(new Student("Jane Smith", "102"));
        students.add(new Student("Alice Johnson", "103"));
        students.add(new Student("Bob Brown", "104"));

        StudentAdapter adapter = new StudentAdapter(this, students);

        ListView listView = findViewById(R.id.listView);
        listView.setAdapter(adapter);
    }
}
```



Output:





Post Practical Questions

1. How can we stop the services in android?
 - a. **Stopservice() and stopservice()**
 - b. Finish()
 - c. System.exit()
 - d. None of the above
2. What is the use of a content provider in android?
 - a. Storing data in database
 - b. **Sharing data between application**
 - c. Sending data from one app to another app
 - d. None
3. An Android _____ is an abstract description of an operation to be performed. It can be used with startActivity to launch an Activity.
 - a. Filters
 - b. **Intent**
 - c. Service
 - d. Broadcast receiver
4. Which of the following actions will display the phone dialer with the given number filled in?
 - a. ACTION_VIEW tel:123
 - b. ACTION_SET
 - c. **ACTION_DIAL tel:123**
 - d. None

References:

1. <https://www.geeksforgeeks.org/android-listview-in-java-with-example/>

Conclusion:

In conclusion we will learn listview with adapter class and show students details.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-7

DATE: _____

AIM: Create an Android application that stores user Login credentials using Shared Preferences.

Practical Code:

Step 1: Create a New Project

Step 2: Add the below strings in your strings.xml file

```
<resources>
    <string name="app_name">sharedPreferences</string>
    <!--string for login button-->
    <string name="login">Login</string>
    <!--string for edittext hint in password-->
    <string name="enter_password">Enter password</string>
    <!--string for edittext hint in email-->
    <string name="enter_youe_email">Enter your Email</string>
    <!--string for logout button-->
    <string name="logout">Logout</string>
</resources>
```

Step 3: Working with the activity_main.xml file

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

    <!--EditText for getting user email address-->
    <!--input type is set to email-->
    <EditText
        android:id="@+id/idEdtEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="10dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="10dp"
        android:hint="@string/enter_youe_email"
        android:importantForAutofill="no"
        android:inputType="textEmailAddress" />

    <!--EditText for getting user password-->
```



```
<!--input type is set to password-->
<EditText
    android:id="@+id/idEdtPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/idEdtEmail"
    android:layout_marginStart="10dp"
    android:layout_marginTop="30dp"
    android:layout_marginEnd="10dp"
    android:hint="@string/enter_password"
    android:importantForAutofill="no"
    android:inputType="textPassword" />

<!--button to continue to login-->
<Button
    android:id="@+id/idBtnLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/idEdtPassword"
    android:layout_marginStart="10dp"
    android:layout_marginTop="30dp"
    android:layout_marginEnd="10dp"
    android:text="@string/login" />

</RelativeLayout>
```

Step 4: Create a new Activity for the Home Screen

Step 5: Working with the MainActivity.kt file

package com.example.sharedpreferences

```
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.text.TextUtils
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    // creating constant keys for shared preferences.
    companion object {
        const val SHARED_PREFS = "shared_prefs"
        const val EMAIL_KEY = "email_key"
```



```
const val PASSWORD_KEY = "password_key"
}

// variable for shared preferences.
private lateinit var sharedPreferences: SharedPreferences
private var email: String? = null
private var password: String? = null

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    // Initializing EditTexts and our Button
    val emailEdt = findViewById<EditText>(R.id.idEdtEmail)
    val passwordEdt = findViewById<EditText>(R.id.idEdtPassword)
    val loginBtn = findViewById<Button>(R.id.idBtnLogin)

    // getting the data which is stored in shared preferences.
    sharedPreferences = getSharedPreferences(SHARED_PREFS,
Context.MODE_PRIVATE)

    // in shared prefs inside get string method
    // we are passing key value as EMAIL_KEY and
    // default value is
    // set to null if not present.
    email = sharedPreferences.getString("EMAIL_KEY", null)
    password = sharedPreferences.getString("PASSWORD_KEY", null)

    // calling on click listener for login button.
    loginBtn.setOnClickListener {
        // to check if the user fields are empty or not.
        if (TextUtils.isEmpty(emailEdt.text.toString()) &&
TextUtils.isEmpty(passwordEdt.text.toString())) {
            // this method will call when email and password fields are
            empty.
            Toast.makeText(this@MainActivity, "Please Enter Email and
            Password", Toast.LENGTH_SHORT).show()
        } else {
            val editor = sharedPreferences.edit()

            // below two lines will put values for
            // email and password in shared preferences.
            editor.putString(EMAIL_KEY, emailEdt.text.toString())
            editor.putString(PASSWORD_KEY,
passwordEdt.text.toString())
        }
    }
}
```



```
// to save our data with key and value.
editor.apply()

// starting new activity.
val i = Intent(this@MainActivity, HomeActivity::class.java)
startActivity(i)
finish()

    }
}

override fun onStart() {
    super.onStart()
    if (email != null && password != null) {
        val i = Intent(this@MainActivity, HomeActivity::class.java)
        startActivity(i)
    }
}
```

Step 6: Now we will work on our Home Screen

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

    <!--TextView for displaying
    user's email address-->
    <TextView
        android:id="@+id/idTVWelcome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:padding="5dp"
        android:textAlignment="center"
        android:textSize="20sp" />

    <!--button for logging out of the app-->
    <Button
        android:id="@+id/idBtnLogout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/idTVWelcome"
```



```
android:layout_marginStart="20dp"
android:layout_marginTop="20dp"
android:layout_marginEnd="20dp"
android:text="@string/logout" />
```

</RelativeLayout>

HomeActivity.kt

```
package com.example.sharedpreferences
```

```
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
```

```
class HomeActivity : AppCompatActivity() {
```

```
    // creating constant keys for shared preferences.
```

```
    companion object {
```

```
        const val SHARED_PREFS = "shared_prefs"
```

```
        const val EMAIL_KEY = "email_key"
```

```
        const val PASSWORD_KEY = "password_key"
```

```
    }
```

```
    // variable for shared preferences.
```

```
    private lateinit var sharedPreferences: SharedPreferences
```

```
    private var email: String? = null
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
```

```
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_home)
```

```
        // initializing our shared preferences.
```

```
        sharedPreferences = getSharedPreferences(SHARED_PREFS,
Context.MODE_PRIVATE)
```

```
        // getting data from shared prefs and
```

```
        // storing it in our string variable.
```

```
        email = sharedPreferences.getString(EMAIL_KEY, null)
```

```
        // initializing our textview and button.
```

```
        val welcomeTV = findViewById<TextView>(R.id.idTVWelcome)
```

```
        welcomeTV.text = "Welcome $email"
```



```
val logoutBtn = findViewById<Button>(R.id.idBtnLogout)
logoutBtn.setOnClickListener {
    // calling method to edit values in shared prefs.
    val editor = sharedPreferences.edit()

    // below line will clear
    // the data in shared prefs.
    editor.clear()

    // below line will apply empty
    // data to shared prefs.
    editor.apply()

    // starting mainactivity after
    // clearing values in shared preferences.
    val i = Intent(this@HomeActivity, MainActivity::class.java)
    startActivity(i)
    finish()
}
}
```



**SILVER OAK
UNIVERSITY**
EDUCATION TO INNOVATION

Output:

Login

Welcome avi@gmail.com

Logout



Post Practical Question

1. Which is the container element for your menu?
 - a. **<menu>**
 - b. <item>
 - c. <group>
 - d. <icon>
2. This denotes an item and is nested inside of the menu tag.
 - a. **<menu>**
 - b. <item>
 - c. <group>
 - d. <icon>
3. This is used to signify a certain property or feature to a couple of menu items
 - a. <menu>
 - b. **<item>**
 - c. <group>
 - d. <icon>
4. This is a unique identifier for the item in the menu.
 - a. **Id**
 - b. Icon
 - c. Title
 - d. Text

References:

1. <https://developer.android.com/reference/android/content/SharedPreferences>

Conclusion:

In conclusion when the user enters their username and password and clicks the login button, the credentials are stored using SharedPreferences. On subsequent app launches, the stored credentials are retrieved and automatically filled into the EditText fields.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-8

DATE: _____

AIM: Create a Registration application to store data using the Room database.

Practical Code:

Step 1: Create a New Project

Step 2: Adding dependency for using Room in build.gradle files

// add below dependency for using room.

implementation 'androidx.room:room-runtime:2.2.5'

annotationProcessor 'androidx.room:room-compiler:2.2.5'

// add below dependency for using lifecycle extensions for room.

implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'

annotationProcessor 'androidx.lifecycle:lifecycle-compiler:2.2.0'

Step 3: Working with the activity_main.xml file

```
<?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"
```

```
    android:orientation="vertical"
```

```
    tools:context=".MainActivity">
```

```
<!--recycler view to display our data-->
```

```
<androidx.recyclerview.widget.RecyclerView
```

```
    android:id="@+id/idRVCourses"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent" />
```

```
<!--fab to add new courses-->
```

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
```

```
    android:id="@+id/idFABAdd"
```



```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentEnd="true"
android:layout_alignParentBottom="true"
android:layout_marginStart="18dp"
android:layout_marginTop="18dp"
android:layout_marginEnd="18dp"
android:layout_marginBottom="18dp"
android:src="@android:drawable/ic_input_add"
app:tint="@color/white" />
```

</RelativeLayout>

Step 4: Creating a modal class for storing our data

```
import androidx.room.Entity;
```

```
import androidx.room.PrimaryKey;
```

```
// below line is for setting table name.
```

```
@Entity(tableName = "course_table")
```

```
public class CourseModal {
```

```
    // below line is to auto increment
```

```
    // id for each course.
```

```
    @PrimaryKey(autoGenerate = true)
```

```
    // variable for our id.
```

```
    private int id;
```

```
    // below line is a variable
```

```
    // for course name.
```

```
    private String courseName;
```

```
    // below line is use for
```

```
    // course description.
```

```
    private String courseDescription;
```



```
// below line is use
// for course duration.
private String courseDuration;

// below line we are creating constructor class.
// inside constructor class we are not passing
// our id because it is incrementing automatically
public CourseModal(String courseName, String courseDescription, String
courseDuration) {
    this.courseName = courseName;
    this.courseDescription = courseDescription;
    this.courseDuration = courseDuration;
}

// on below line we are creating
// getter and setter methods.
public String getCourseName() {
    return courseName;
}

public void setCourseName(String courseName) {
    this.courseName = courseName;
}

public String getCourseDescription() {
    return courseDescription;
}

public void setCourseDescription(String courseDescription) {
    this.courseDescription = courseDescription;
}

public String getCourseDuration() {
    return courseDuration;
}
```



```
public void setCourseDuration(String courseDuration) {
    this.courseDuration = courseDuration;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}
}
```

Step 5: Creating a Dao interface for our database

```
import androidx.lifecycle.LiveData;
import androidx.room.Delete;
import androidx.room.Insert;
import androidx.room.Query;
import androidx.room.Update;

import java.util.List;

// Adding annotation
// to our Dao class
@androidx.room.Dao
public interface Dao {

    // below method is use to
    // add data to database.
    @Insert
    void insert(CourseModal model);

    // below method is use to update
    // the data in our database.
    @Update
    void update(CourseModal model);
}
```



```
// below line is use to delete a
// specific course in our database.
@Delete
void delete(CourseModal model);

// on below line we are making query to
// delete all courses from our database.
@Query("DELETE FROM course_table")
void deleteAllCourses();

// below line is to read all the courses from our database.
// in this we are ordering our courses in ascending order
// with our course name.
@Query("SELECT * FROM course_table ORDER BY courseName ASC")
LiveData<List<CourseModal>> getAllCourses();
}
```

Step 6: Creating a database class

```
import android.content.Context;
import android.os.AsyncTask;

import androidx.annotation.NonNull;
import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;
import androidx.sqlite.db.SupportSQLiteDatabase;

// adding annotation for our database entities and db version.
@Database(entities = {CourseModal.class}, version = 1)
public abstract class CourseDatabase extends RoomDatabase {

    // below line is to create instance
    // for our database class.
    private static CourseDatabase instance;
```



```
// below line is to create
// abstract variable for dao.
public abstract Dao Dao();

// on below line we are getting instance for our database.
public static synchronized CourseDatabase getInstance(Context context) {
    // below line is to check if
    // the instance is null or not.
    if (instance == null) {
        // if the instance is null we
        // are creating a new instance
        instance =

            // for creating a instance for our database
            // we are creating a database builder and passing
            // our database class with our database name.
            Room.databaseBuilder(context.getApplicationContext(),
                CourseDatabase.class, "course_database")
                // below line is use to add fall back to
                // destructive migration to our database.
                .fallbackToDestructiveMigration()
                // below line is to add callback
                // to our database.
                .addCallback(roomCallback)
                // below line is to
                // build our database.
                .build();
    }
    // after creating an instance
    // we are returning our instance
    return instance;
}

// below line is to create a callback for our room database.
private static RoomDatabase.Callback roomCallback = new RoomDatabase.Callback() {
    @Override
    public void onCreate(@NonNull SupportSQLiteDatabase db) {
```



```
super.onCreate(db);
// this method is called when database is created
// and below line is to populate our data.
new PopulateDbAsyncTask(instance).execute();
    }
};

// we are creating an async task class to perform task in background.
private static class PopulateDbAsyncTask extends AsyncTask<Void, Void, Void> {
    PopulateDbAsyncTask(CourseDatabase instance) {
        Dao dao = instance.Dao();
    }
    @Override
    protected Void doInBackground(Void... voids) {
        return null;
    }
}
```

Step 7: Create a new java class for our Repository

```
import android.app.Application;
import android.os.AsyncTask;

import androidx.lifecycle.LiveData;

import java.util.List;

public class CourseRepository {

    // below line is the create a variable
    // for dao and list for all courses.
    private Dao dao;
    private LiveData<List<CourseModal>> allCourses;

    // creating a constructor for our variables
    // and passing the variables to it.
```




```
public CourseRepository(Application application) {
    CourseDatabase database = CourseDatabase.getInstance(application);
    dao = database.Dao();
    allCourses = dao.getAllCourses();
}

// creating a method to insert the data to our database.
public void insert(CourseModal model) {
    new InsertCourseAsyncTask(dao).execute(model);
}

// creating a method to update data in database.
public void update(CourseModal model) {
    new UpdateCourseAsyncTask(dao).execute(model);
}

// creating a method to delete the data in our database.
public void delete(CourseModal model) {
    new DeleteCourseAsyncTask(dao).execute(model);
}

// below is the method to delete all the courses.
public void deleteAllCourses() {
    new DeleteAllCoursesAsyncTask(dao).execute();
}

// below method is to read all the courses.
public LiveData<List<CourseModal>> getAllCourses() {
    return allCourses;
}

// we are creating a async task method to insert new course.
private static class InsertCourseAsyncTask extends AsyncTask<CourseModal, Void,
Void> {
    private Dao dao;
```



```
private InsertCourseAsyncTask(Dao dao) {
    this.dao = dao;
}

@Override
protected Void doInBackground(CourseModal... model) {
    // below line is use to insert our modal in dao.
    dao.insert(model[0]);
    return null;
}

// we are creating a async task method to update our course.
private static class UpdateCourseAsyncTask extends AsyncTask<CourseModal, Void,
Void> {
    private Dao dao;

    private UpdateCourseAsyncTask(Dao dao) {
        this.dao = dao;
    }

    @Override
    protected Void doInBackground(CourseModal... models) {
        // below line is use to update
        // our modal in dao.
        dao.update(models[0]);
        return null;
    }
}

// we are creating a async task method to delete course.
private static class DeleteCourseAsyncTask extends AsyncTask<CourseModal, Void,
Void> {
    private Dao dao;

    private DeleteCourseAsyncTask(Dao dao) {
```



```
        this.dao = dao;
    }

    @Override
    protected Void doInBackground(CourseModal... models) {
        // below line is use to delete
        // our course modal in dao.
        dao.delete(models[0]);
        return null;
    }
}

// we are creating a async task method to delete all courses.
private static class DeleteAllCoursesAsyncTask extends AsyncTask<Void, Void, Void> {
    private Dao dao;
    private DeleteAllCoursesAsyncTask(Dao dao) {
        this.dao = dao;
    }
    @Override
    protected Void doInBackground(Void... voids) {
        // on below line calling method
        // to delete all courses.
        dao.deleteAllCourses();
        return null;
    }
}
}
```

Step 8: Creating a class for our Repository

```
import android.app.Application;

import androidx.annotation.NonNull;
import androidx.lifecycle.AndroidViewModel;
import androidx.lifecycle.LiveData;

import java.util.List;
```



```
public class ViewModal extends AndroidViewModel {

    // creating a new variable for course repository.
    private CourseRepository repository;

    // below line is to create a variable for live
    // data where all the courses are present.
    private LiveData<List<CourseModal>> allCourses;

    // constructor for our view modal.
    public ViewModal(@NonNull Application application) {
        super(application);
        repository = new CourseRepository(application);
        allCourses = repository.getAllCourses();
    }

    // below method is use to insert the data to our repository.
    public void insert(CourseModal model) {
        repository.insert(model);
    }

    // below line is to update data in our repository.
    public void update(CourseModal model) {
        repository.update(model);
    }

    // below line is to delete the data in our repository.
    public void delete(CourseModal model) {
        repository.delete(model);
    }

    // below method is to delete all the courses in our list.
    public void deleteAllCourses() {
        repository.deleteAllCourses();
    }
}
```

```
// below method is to get all the courses in our list.  
public LiveData<List<CourseModal>> getAllCourses() {  
    return allCourses;  
}  
}
```

Step 9: Creating a layout file for each item of RecyclerView

```
<?xml version="1.0" encoding="utf-8"?>  
<androidx.cardview.widget.CardView  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:app="http://schemas.android.com/apk/res-auto"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="5dp"  
    android:elevation="8dp"  
    app:cardCornerRadius="8dp">  
  
    <LinearLayout  
        android:id="@+id/idLLCourse"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
        android:layout_margin="5dp"  
        android:orientation="vertical">  
  
        <!--text view for our course name-->  
        <TextView  
            android:id="@+id/idTVCourseName"  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content"  
            android:padding="8dp"  
            android:text="Course Name"  
            android:textColor="@color/black"  
            android:textSize="15sp" />  
  
        <!--text view for our course duration-->
```



```
<TextView
    android:id="@+id/idTVCourseDuration"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp"
    android:text="Course Duration"
    android:textColor="@color/black"
    android:textSize="15sp" />

<!--text view for our course description-->
<TextView
    android:id="@+id/idTVCourseDescription"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp"
    android:text="Course Description"
    android:textColor="@color/black"
    android:textSize="15sp" />

</LinearLayout>

</androidx.cardview.widget.CardView>
```

Step 10: Creating a RecyclerView Adapter class to set data for each item of RecyclerView

```
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.DiffUtil;
import androidx.recyclerview.widget.ListAdapter;
import androidx.recyclerview.widget.RecyclerView;

public class CourseRVAdapter extends ListAdapter<CourseModal,
CourseRVAdapter.ViewHolder> {
```



```
// creating a variable for on item click listener.
private OnItemClickListener listener;

// creating a constructor class for our adapter class.
CourseRVAdapter() {
    super(DIFF_CALLBACK);
}

// creating a call back for item of recycler view.
private static final DiffUtil.ItemCallback<CourseModal> DIFF_CALLBACK = new
DiffUtil.ItemCallback<CourseModal>() {
    @Override
    public boolean areItemsTheSame(CourseModal oldItem, CourseModal newItem)
    {
        return oldItem.getId() == newItem.getId();
    }

    @Override
    public boolean areContentsTheSame(CourseModal oldItem, CourseModal
newItem) {
        // below line is to check the course name, description and course duration.
        return oldItem.getCourseName().equals(newItem.getCourseName()) &&
oldItem.getCourseDescription().equals(newItem.getCourseDescription()) &&
oldItem.getCourseDuration().equals(newItem.getCourseDuration());
    }
};

@NonNull
@Override
public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    // below line is use to inflate our layout
    // file for each item of our recycler view.
    View item = LayoutInflater.from(parent.getContext())
        .inflate(R.layout.course_rv_item, parent, false);
    return new ViewHolder(item);
}
```



```
}
```

@Override

```
public void onBindViewHolder(@NonNull ViewHolder holder, int position) {  
    // below line of code is use to set data to  
    // each item of our recycler view.  
    CourseModal model = getCourseAt(position);  
    holder.courseNameTV.setText(model.getCourseName());  
    holder.courseDescTV.setText(model.getCourseDescription());  
    holder.courseDurationTV.setText(model.getCourseDuration());  
}
```

// creating a method to get course modal for a specific position.

```
public CourseModal getCourseAt(int position) {  
    return getItem(position);  
}
```

```
public class ViewHolder extends RecyclerView.ViewHolder {
```

// view holder class to create a variable for each view.

```
TextView courseNameTV, courseDescTV, courseDurationTV;
```

```
ViewHolder(@NonNull View itemView) {
```

```
    super(itemView);
```

// initializing each view of our recycler view.

```
courseNameTV = itemView.findViewById(R.id.idTVCourseName);
```

```
courseDescTV = itemView.findViewById(R.id.idTVCourseDescription);
```

```
courseDurationTV = itemView.findViewById(R.id.idTVCourseDuration);
```

// adding on click listener for each item of recycler view.

```
itemView.setOnClickListener(new View.OnClickListener() {
```

@Override

```
public void onClick(View v) {
```

// inside on click listener we are passing

// position to our item of recycler view.

```
int position = getAdapterPosition();
```

```
if (listener != null && position !=
```



```
RecyclerView.NO_POSITION) {  
    listener.onClick(getItem(position));  
}  
}  
});  
}  
}
```

```
public interface OnItemClickListener {  
    void onClick(CourseModal model);  
}  
public void setOnItemClickListener(OnItemClickListener listener) {  
    this.listener = listener;  
}  
}
```

Step 11: Creating a new Activity for Adding and Updating our Course

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical"  
    tools:context=".NewCourseActivity">  
  
    <!--edit text for our course name-->  
    <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 for our 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" />
```

```
<!--edit text for course description-->
```

```
<EditText
```

```
    android:id="@+id/idEdtCourseDuration"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="10dp"  
    android:hint="Course Duration" />
```

```
<!--button for saving data to room database-->
```

```
<Button
```

```
    android:id="@+id/idBtnSaveCourse"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="10dp"  
    android:padding="5dp"  
    android:text="Save your course"  
    android:textAllCaps="false" />
```

```
</LinearLayout>
```

Step 12: Working with the NewCourseActivity.java file

```
import android.content.Intent;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.widget.Button;
```

```
import android.widget.EditText;
```

```
import android.widget.Toast;
```

```
import androidx.appcompat.app.AppCompatActivity;
```



```
public class NewCourseActivity extends AppCompatActivity {

    // creating a variables for our button and edittext.
    private EditText courseNameEdt, courseDescEdt, courseDurationEdt;
    private Button courseBtn;

    // creating a constant string variable for our
    // course name, description and duration.
    public static final String EXTRA_ID =
"com.gtappdevelopers.gfgroomdatabase.EXTRA_ID";
    public static final String EXTRA_COURSE_NAME =
"com.gtappdevelopers.gfgroomdatabase.EXTRA_COURSE_NAME";
    public static final String EXTRA_DESCRIPTION =
"com.gtappdevelopers.gfgroomdatabase.EXTRA_COURSE_DESCRIPTION";
    public static final String EXTRA_DURATION =
"com.gtappdevelopers.gfgroomdatabase.EXTRA_COURSE_DURATION";

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

        // initializing our variables for each view.
        courseNameEdt = findViewById(R.id.idEdtCourseName);
        courseDescEdt = findViewById(R.id.idEdtCourseDescription);
        courseDurationEdt = findViewById(R.id.idEdtCourseDuration);
        courseBtn = findViewById(R.id.idBtnSaveCourse);

        // below line is to get intent as we
        // are getting data via an intent.
        Intent intent = getIntent();
        if (intent.hasExtra(EXTRA_ID)) {
            // if we get id for our data then we are
            // setting values to our edit text fields.

            courseNameEdt.setText(intent.getStringExtra(EXTRA_COURSE_NAME));
            courseDescEdt.setText(intent.getStringExtra(EXTRA_DESCRIPTION));
```



```
        courseDurationEdt.setText(intent.getStringExtra(EXTRA_DURATION));
    }
    // adding on click listener for our save button.
    courseBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            // getting text value from edittext and validating if
            // the text fields are empty or not.
            String courseName = courseNameEdt.getText().toString();
            String courseDesc = courseDescEdt.getText().toString();
            String courseDuration = courseDurationEdt.getText().toString();
            if (courseName.isEmpty() || courseDesc.isEmpty() ||
courseDuration.isEmpty()) {
                Toast.makeText(NewCourseActivity.this, "Please enter the
valid course details.", Toast.LENGTH_SHORT).show();
                return;
            }
            // calling a method to save our course.
            saveCourse(courseName, courseDesc, courseDuration);
        }
    });
}

private void saveCourse(String courseName, String courseDescription, String
courseDuration) {
    // inside this method we are passing
    // all the data via an intent.
    Intent data = new Intent();

    // in below line we are passing all our course detail.
    data.putExtra(EXTRA_COURSE_NAME, courseName);
    data.putExtra(EXTRA_DESCRIPTION, courseDescription);
    data.putExtra(EXTRA_DURATION, courseDuration);
    int id = getIntent().getIntExtra(EXTRA_ID, -1);
    if (id != -1) {
        // in below line we are passing our id.
        data.putExtra(EXTRA_ID, id);
    }
}
```



```
}

// at last we are setting result as data.
setResult(RESULT_OK, data);

// displaying a toast message after adding the data
Toast.makeText(this, "Course has been saved to Room Database. ",
Toast.LENGTH_SHORT).show();
}
}
```

Step 13: Working with the MainActivity.java file

```
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.lifecycle.Observer;
import androidx.lifecycle.ViewModelProviders;
import androidx.recyclerview.widget.ItemTouchHelper;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.List;

public class MainActivity extends AppCompatActivity {

    // creating a variables for our recycler view.
    private RecyclerView coursesRV;
    private static final int ADD_COURSE_REQUEST = 1;
    private static final int EDIT_COURSE_REQUEST = 2;
```



```
private ViewModal viewmodal;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
    // initializing our variable for our recycler view and fab.
```

```
    coursesRV = findViewById(R.id.idRVCourses);
```

```
    FloatingActionButton fab = findViewById(R.id.idFABAdd);
```

```
    // adding on click listener for floating action button.
```

```
    fab.setOnClickListener(new View.OnClickListener() {
```

```
        @Override
```

```
        public void onClick(View v) {
```

```
            // starting a new activity for adding a new course
```

```
            // and passing a constant value in it.
```

```
            Intent intent = new Intent(MainActivity.this,  
NewCourseActivity.class);
```

```
            startActivityForResult(intent, ADD_COURSE_REQUEST);
```

```
        }
```

```
    });
```

```
    // setting layout manager to our adapter class.
```

```
    coursesRV.setLayoutManager(new LinearLayoutManager(this));
```

```
    coursesRV.setHasFixedSize(true);
```

```
    // initializing adapter for recycler view.
```

```
    final CourseRVAdapter adapter = new CourseRVAdapter();
```

```
    // setting adapter class for recycler view.
```

```
    coursesRV.setAdapter(adapter);
```

```
    // passing a data from view modal.
```

```
    viewmodal = ViewModelProviders.of(this).get(ViewModal.class);
```



```
// below line is use to get all the courses from view modal.
viewmodal.getAllCourses().observe(this, new Observer<List<CourseModal>>() {
    @Override
    public void onChanged(List<CourseModal> models) {
        // when the data is changed in our models we are
        // adding that list to our adapter class.
        adapter.submitList(models);
    }
});

// below method is use to add swipe to delete method for item of recycler view.
new ItemTouchHelper(new ItemTouchHelper.SimpleCallback(0,
ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
    @Override
    public boolean onMove(@NonNull RecyclerView recyclerView,
@NonNull RecyclerView.ViewHolder viewHolder, @NonNull RecyclerView.ViewHolder
target) {
        return false;
    }

    @Override
    public void onSwiped(@NonNull RecyclerView.ViewHolder viewHolder,
int direction) {
        // on recycler view item swiped then we are deleting the item of
        our recycler view.

        viewmodal.delete(adapter.getCourseAt(viewHolder.getAdapterPosition()));
        Toast.makeText(MainActivity.this, "Course deleted",
        Toast.LENGTH_SHORT).show();
    }
}).

// below line is use to attach this to recycler view.
attachToRecyclerView(coursesRV);

// below line is use to set item click listener for our item of recycler view.
adapter.setOnItemClickListener(new CourseRVAdapter.OnItemClickListener() {
    @Override
    public void onItemClick(CourseModal model) {
        // after clicking on item of recycler view
        // we are opening a new activity and passing
```



```
// a data to our activity.
Intent intent = new Intent(MainActivity.this,
NewCourseActivity.class);
intent.putExtra(NewCourseActivity.EXTRA_ID, model.getId());
intent.putExtra(NewCourseActivity.EXTRA_COURSE_NAME,
model.getCourseName());
intent.putExtra(NewCourseActivity.EXTRA_DESCRIPTION,
model.getCourseDescription());
intent.putExtra(NewCourseActivity.EXTRA_DURATION,
model.getCourseDuration());

// below line is to start a new activity and
// adding a edit course constant.
startActivityForResult(intent, EDIT_COURSE_REQUEST);
    }
});
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == ADD_COURSE_REQUEST && resultCode ==
RESULT_OK) {
        String courseName =
data.getStringExtra(NewCourseActivity.EXTRA_COURSE_NAME);
        String courseDescription =
data.getStringExtra(NewCourseActivity.EXTRA_DESCRIPTION);
        String courseDuration =
data.getStringExtra(NewCourseActivity.EXTRA_DURATION);
        CourseModal model = new CourseModal(courseName,
courseDescription, courseDuration);
        viewmodal.insert(model);
        Toast.makeText(this, "Course saved", Toast.LENGTH_SHORT).show();
    } else if (requestCode == EDIT_COURSE_REQUEST && resultCode ==
RESULT_OK) {
        int id = data.getIntExtra(NewCourseActivity.EXTRA_ID, -1);
        if (id == -1) {
```

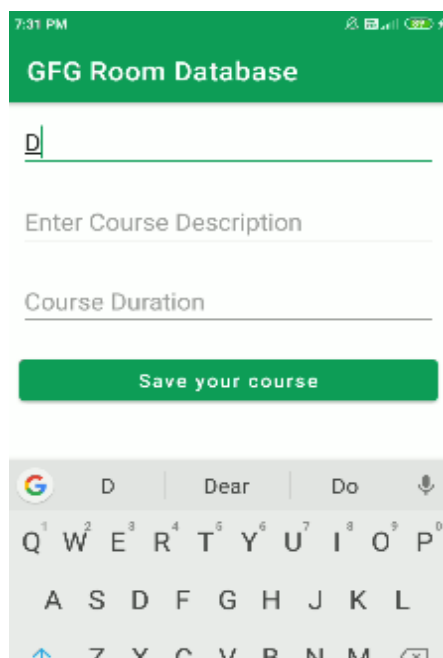


```

        Toast.makeText(this, "Course can't be updated",
Toast.LENGTH_SHORT).show();
        return;
    }
    String courseName =
data.getStringExtra(NewCourseActivity.EXTRA_COURSE_NAME);
    String courseDesc =
data.getStringExtra(NewCourseActivity.EXTRA_DESCRIPTION);
    String courseDuration =
data.getStringExtra(NewCourseActivity.EXTRA_DURATION);
    CourseModal model = new CourseModal(courseName, courseDesc,
courseDuration);
    model.setId(id);
    viewmodal.update(model);
    Toast.makeText(this, "Course updated", Toast.LENGTH_SHORT).show();
} else {
    Toast.makeText(this, "Course not saved",
Toast.LENGTH_SHORT).show();
}
}
}

```

Output:





MCQ Questions

1. You need to use _____ class to store elements in spinner.
 - a. Item Adapter
 - b. ArrayAdapter**
 - c. Listadapter
 - d. Listitemadapter

2. Spinner doesn't have a multiple choice.
 - a. True**
 - b. False

References:

1. <https://www.javatpoint.com/android-internal-storage-example>
2. <https://developer.android.com/training/data-storage>

Conclusion:

In conclusion You have now created an Android application that allows users to register and store their data using the Room database. When users register, their username and email are saved to the Room database.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-9

DATE: _____

AIM: To implement parsing JSON data using the Retrofit library in Android application.

The Retrofit library is a type-safe REST client for Android, Java, and Kotlin, developed by Square. With the help of the Retrofit library, we can have access to a powerful framework that helps us in authenticating and interacting with APIs and sending network requests with OkHttp. With the help of this library, downloading JSON or XML data from a web API becomes easy. In a Retrofit library, once the data is downloaded, it is parsed into a Plain Old Java Object (POJO) which must be defined for each “resource” in the response. Retrofit is an easy and fast library to retrieve and upload data via a REST-based web service.

Retrofit manages the process of receiving, sending, and creating HTTP requests and responses. It resolves issues before sending an error and crashing the application. It pools connections to reduce latency. It is used to cache responses to avoid sending duplicate requests.

CLASSES USED IN RETROFIT

Model Class: A model class contains the objects to be obtained from the JSON file.

Retrofit Instance: This is a Java class. It is used to send requests to an API.

Interface Class: This is a Java class. It is used to define endpoints.

Practical Code:

Step 1: Create a New Project

Step 2: Add the below dependency in your build.gradle file

// below dependency for using retrofit.

implementation 'com.squareup.retrofit2:retrofit:2.9.0'

implementation 'com.squareup.retrofit2:converter-gson:2.5.0'

// below dependency for using picasso image loading library

implementation 'com.squareup.picasso:picasso:2.71828'

Step 3: Adding permissions to the internet in the AndroidManifest.xml file

`<!--permissions for INTERNET-->`

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

Step 4: Working with the activity_main.xml file

```
<?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"
    android:layout_gravity="center"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">
```

```
    <androidx.cardview.widget.CardView
        android:id="@+id/idCVCourse"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:elevation="10dp"
        android:visibility="gone"
        app:cardCornerRadius="8dp">
```

```
        <LinearLayout
```

```
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
```

```
            <ImageView
```

```
                android:id="@+id/idIVCourse"
                android:layout_width="match_parent"
                android:layout_height="300dp"
                android:layout_margin="5dp" />
```

```
            <TextView
```

```
                android:id="@+id/idTVCourseName"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_margin="5dp"
                android:padding="5dp"
                android:text="Course Name "
                android:textColor="@color/black"
                android:textSize="18sp"
                android:textStyle="bold" />
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:orientation="horizontal"
    android:weightSum="2">

    <TextView
        android:id="@+id/idTVBatch"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:padding="5dp"
        android:text="Batch"
        android:textColor="@color/black" />

    <TextView
        android:id="@+id/idTVTracks"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:padding="5dp"
        android:text="Tracks"
        android:textColor="@color/black" />

</LinearLayout>

</LinearLayout>
```

```
</androidx.cardview.widget.CardView>
```

```
<ProgressBar
    android:id="@+id/idLoadingPB"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:visibility="visible" />
```

```
</RelativeLayout>
```

Step 5: Creating a modal class for storing our data

```
public class RecyclerData {
    // string variables for our data
    // make sure that the variable name
    // must be similar to that of key value
    // which we are getting from our json file.
```



```
private String courseName;
private String courseimg;
private String courseMode;
private String courseTracks;

public String getCourseName() {
    return courseName;
}

public void setCourseName(String courseName) {
    this.courseName = courseName;
}

public String getCourseimg() {
    return courseimg;
}

public void setCourseimg(String courseimg) {
    this.courseimg = courseimg;
}

public String getCourseMode() {
    return courseMode;
}

public void setCourseMode(String courseMode) {
    this.courseMode = courseMode;
}

public String getCourseTracks() {
    return courseTracks;
}

public void setCourseTracks(String courseTracks) {
    this.courseTracks = courseTracks;
}

public RecyclerData(String courseName, String courseimg, String courseMode,
String courseTracks) {
    this.courseName = courseName;
    this.courseimg = courseimg;
    this.courseMode = courseMode;
    this.courseTracks = courseTracks;
}
}
```



Step 6: Creating an Interface class for our API Call

```
import retrofit2.Call;
import retrofit2.http.GET;

public interface RetrofitAPI {

    // as we are making get request
    // so we are displaying GET as annotation.
    // and inside we are passing
    // last parameter for our url.
    @GET("63OH")

    // as we are calling data from array
    // so we are calling it with json object
    // and naming that method as getCourse();
    Call<RecyclerView> getCourse();
}
```

Step 7: Working with the MainActivity.java file

```
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import com.squareup.picasso.Picasso;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class MainActivity extends AppCompatActivity {
    // creating variables for our textview,
    // imageview, cardview and progressbar.
    private TextView courseNameTV, courseTracksTV, courseBatchTV;
    private ImageView courseIV;
    private ProgressBar loadingPB;
    private CardView courseCV;

    @Override
```



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

    // initializing our variables.
    loadingPB = findViewById(R.id.idLoadingPB);
    courseCV = findViewById(R.id.idCVCourse);
    courseNameTV = findViewById(R.id.idTVCourseName);
    courseTracksTV = findViewById(R.id.idTVTracks);
    courseBatchTV = findViewById(R.id.idTVBatch);
    courseIV = findViewById(R.id.idIVCourse);
    getCourse();
}

private void getCourse() {

    // on below line we are creating a retrofit
    // builder and passing our base url
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("https://jsonkeeper.com/b/")
        // on below line we are calling add Converter
        // factory as GSON converter factory.
        .addConverterFactory(GsonConverterFactory.create())
        // at last we are building our retrofit builder.
        .build();

    // below line is to create an instance for our retrofit api class.
    RetrofitAPI retrofitAPI = retrofit.create(RetrofitAPI.class);
    Call<RecyclerData> call = retrofitAPI.getCourse();
    call.enqueue(new Callback<RecyclerData>() {
        @Override
        public void onResponse(Call<RecyclerData> call, Response
<RecyclerData> response) {
            if (response.isSuccessful()) {
                // inside the on response method.
                // we are hiding our progress bar.
                loadingPB.setVisibility(View.GONE);
                // in below line we are making our card
                // view visible after we get all the data.
                courseCV.setVisibility(View.VISIBLE);
                RecyclerData modal = response.body();
                // after extracting all the data we are
                // setting that data to all our views.
                courseNameTV.setText(modal.getCourseName());
                courseTracksTV.setText(modal.getCourseTracks());
                courseBatchTV.setText(modal.getCourseMode());
                // we are using picasso to load the image from url.
```



```

        Picasso.get().load(modal.getCourseimg()).into(courseIV);
    }

    @Override
    public void onFailure(Call<RecyclerData> call, Throwable t) {
        // displaying an error message in toast
        Toast.makeText(MainActivity.this, "Fail to get the
data..", Toast.LENGTH_SHORT).show();
    }
});
}
}

```

Output:





Short Questions- Answers

1. What is Fragment?

In Android, the fragment is the part of Activity which represents a portion of User Interface(UI) on the screen. It is the modular section of the android activity that is very helpful in creating UI designs that are flexible in nature and auto-adjustable based on the device screen size.

2. What do you mean by a Activity in android?

An activity is one screen of an app. In that way the activity is very similar to a window in the Windows operating system. The most specific block of the user interface is the activity. An Android app contains activities, meaning one or more screens. Examples: Login screen, sign up screen, and home screen.

3. What is a viewgroup in android?

A ViewGroup (which is itself a special type of view) provides the layout in which you can order the appearance and sequence of views. Examples of ViewGroups include LinearLayout and FrameLayout. A ViewGroup derives from the base class android.view.

References:

1. <https://www.digitalocean.com/community/tutorials/retrofit-android-example-tutorial>

Conclusion:

In conclusion we have now implemented parsing JSON data using the Retrofit library in your Android application. The application makes a network request to fetch user data from a REST API and displays the fetched data in the UI.

Marks out of 10	
Signature with Date of Completion	

PRACTICAL-10

DATE: _____

AIM: Develop an application for working with view animation.

View Animation can be used to add animation to a specific view to perform tweened animation on views. Tweened animation calculates animation information such as size, rotation, start point, and endpoint. These animations are slower and less flexible. An example of View animation can be used if we want to expand a specific layout in that place we can use View Animation. The example of View Animation can be seen in Expandable RecyclerView.

Methods	Description
startAnimation()	This method will start the animation.
clearAnimation()	This method will clear the animation running on a specific view.

Practical Code:

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

    <ImageView
        android:id="@+id/imageview"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="40dp"
        android:contentDescription="@string/app_name"
        android:src="@drawable/gfgimage" />
```

```
<LinearLayout
    android:id="@+id/linear1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/imageview"
    android:layout_marginTop="30dp"
    android:orientation="horizontal"
    android:weightSum="3">

    <!--To start the fading animation of the image-->
    <Button
        android:id="@+id/BTNfade"
        style="@style/TextAppearance.AppCompat.Widget.Button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:layout_weight="1"
        android:padding="3dp"
        android:text="@string/fade"
        android:textColor="@color/white" />

</LinearLayout>

<!--To stop the animation of the image-->
<Button
    android:id="@+id/BTNstop"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/linear2"
    android:layout_marginLeft="30dp"
    android:layout_marginTop="30dp"
    android:layout_marginRight="30dp"
    android:text="@string/stop_animation" />

</RelativeLayout>
```

```
// MainActivity.java
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.Button;
```

```
import android.widget.ImageView;

public class MainActivity extends AppCompatActivity {

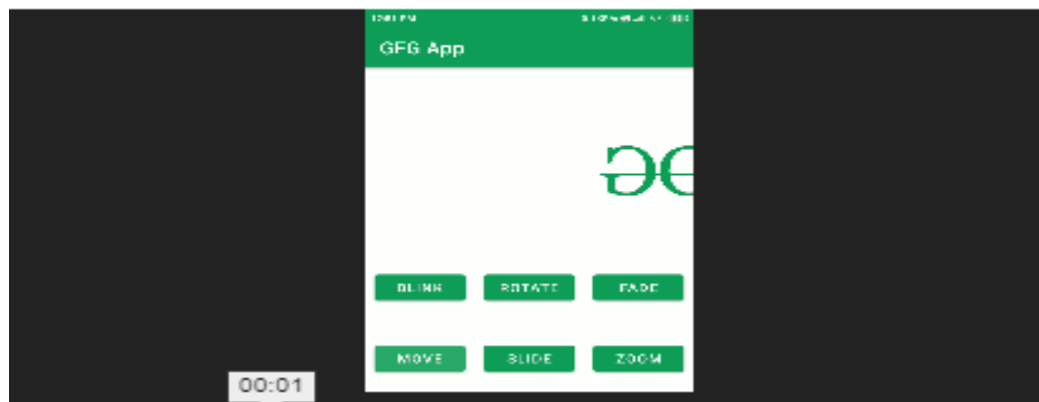
    ImageView imageView;
    Button fadeBTN;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        imageView = findViewById(R.id.imageview);
        fadeBTN = findViewById(R.id.BTNfade);

        fadeBTN.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // To add fade animation
                Animation animation = AnimationUtils.loadAnimation(getApplicationContext(),
R.anim.fade_animation);
                imageView.startAnimation(animation);
            }
        });
    }
}
```

Output:

Output:





Short Questions - Answers

1. What are the intent filters?

An intent filter declares the capabilities of its parent component: what an activity or service can do and what types of broadcasts a receiver can handle. It opens the component to receiving intents of the advertised type while filtering out those that aren't meaningful for the component.

2. Name the different data storage options available on Android platform.

1. Shared preferences
2. Internal storage
3. File storage
4. Network connection etc

3. What is the adapter in Android?

In Android, Adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to an Adapter view then view can takes the data from the adapter view and shows the data on different views like as ListView, GridView, Spinner etc.

4. What is 'bundle' in Android?

Bundle is a class in android which is used to pass data from one activity to another activity within an android application. We can pass data using key and value pairs using bundles. We can pass the data using the key and can use that same key to retrieve the data which is passed for that key

References:

1. <https://www.geeksforgeeks.org/animation-in-android-with-example/>
2. <https://developer.android.com/develop/ui/views/animations/overview>

Conclusion

In conclusion, we will see how animation works in action when you click the button.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-11

DATE: _____

AIM: Write a program in Dart that finds simple interest. Formula= $(p * t * r) / 100$.

Dart is used to build high-performance mobile or web applications. Some of its most significant applications include: It's the basic programming language for the Flutter framework—explained in further detail below—and is used to build scalable mobile applications.

Every app requires the top-level `main()` function, where execution starts. Functions that don't explicitly return a value have the `void` return type. To display text on the console, you can use the top-level `print()` function:

```
void main() {  
  print('Hello, World!');  
}
```

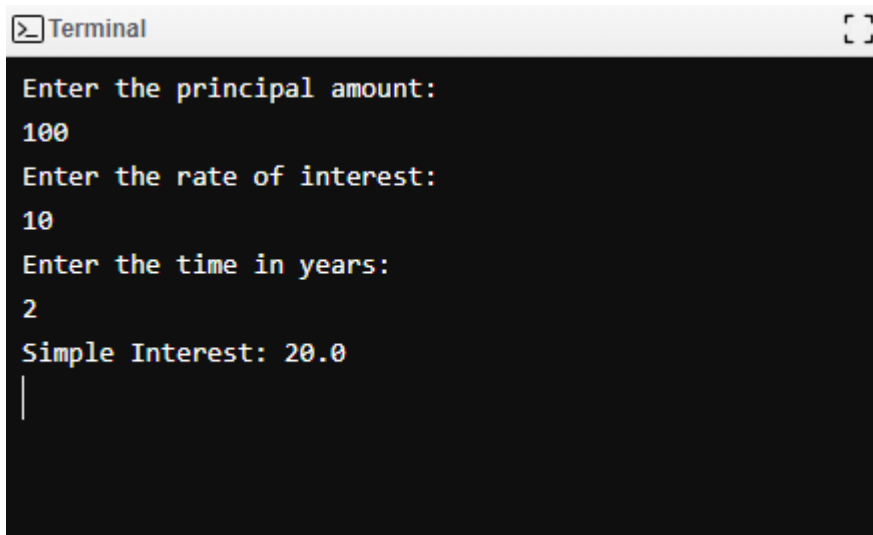
Here's an example of creating a variable and initializing it:

```
dart  
var name = 'Bob';
```

Practical Code:

```
import 'dart:io';  
  
void main() {  
  print("Enter the principal amount:");  
  double principal = double.parse(stdin.readLineSync());  
  
  print("Enter the rate of interest:");  
  double rate = double.parse(stdin.readLineSync());  
  
  print("Enter the time in years:");  
  double time = double.parse(stdin.readLineSync());  
  
  double simpleInterest = (principal * time * rate) / 100;  
  
  print("Simple Interest: $simpleInterest");  
}
```

Output:

A screenshot of a terminal window titled "Terminal" showing the execution of a program. The program prompts for the principal amount, rate of interest, and time in years, and then calculates the simple interest.

```
Terminal
Enter the principal amount:
100
Enter the rate of interest:
10
Enter the time in years:
2
Simple Interest: 20.0
|
```

Post Practical Questions

1. What is the Dart programming language?

Dart is a **client-optimized language for developing fast apps on any platform**. Its goal is to offer the most productive programming language for multi-platform .

2. What build modes are available in Flutter?

Flutter offers developers three modes: Debug Mode, Release Mode, and Profile Mode. However, Debug Mode introduces extra checks like assertions that are absent in the other modes, which can introduce jank or stalls in the app's performance.

3. Is Flutter an SDK?

Flutter is Google's free, open-source software development kit (SDK) for cross-platform mobile application development. Using a single platform-agnostic codebase, Flutter helps



developers build high-performance, scalable applications with attractive and functional user interfaces for Android or IOS.

4. In What technology is Flutter built?

Flutter uses the open-source programming language Dart, which was also developed by Google. Dart is optimized for building UIs, and many of Dart's strengths are used in Flutter.

References:

1. <https://www.simplilearn.com/flutter-interview-questions-article>
2. <https://flutter.dev/learn>

Conclusion:

In conclusion we prompt the user to enter the principal amount, rate of interest, and time in years. We read the input using `stdin.readLineSync()` and parse it into double values. We then calculate the simple interest using the formula $(p * t * r) / 100$. Finally, we print the calculated simple interest.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-12

DATE: _____

AIM: Working with Widgets in Flutter (Text, padding, margin, and alignment).

Understanding the Box Model

The box model in Flutter is a core concept that dictates how widgets are sized and positioned. Every widget in a Flutter app is surrounded by an invisible box that can be manipulated using padding and margin properties. The box model allows developers to control the space inside the widget, known as padding, and the space around the widget, referred to as margin.

For instance, when you want to add padding to a Text widget, you might wrap it in a Padding widget and specify the EdgeInsets:

```
Padding(  
  padding: const EdgeInsets.all(8.0),  
  child: Text('Hello World'),  
)
```

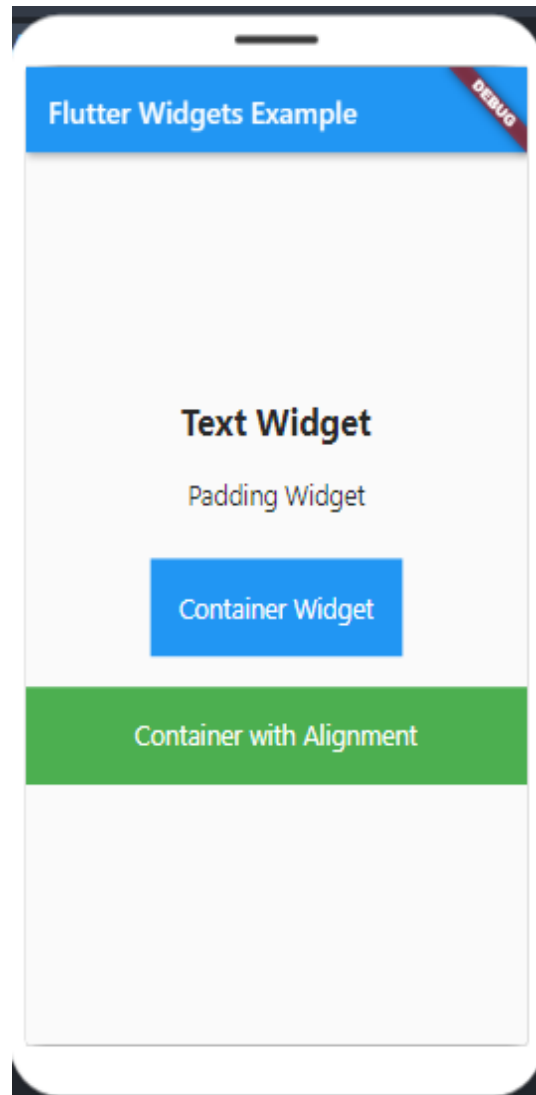
Practical Code:

```
import 'package:flutter/material.dart';  
  
void main() {  
  runApp(MyApp());  
}  
  
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      home: Scaffold(  
        appBar: AppBar(  
          title: Text('Widget Example'),  
        ),  
        body: Center(  
          child: Container(  
            margin: EdgeInsets.all(20.0),  
            padding: EdgeInsets.all(20.0),  
            alignment: Alignment.center,  
            decoration: BoxDecoration(  
              border: Border.all(color: Colors.blueAccent),  
            ),  
          ),  
        ),  
      ),  
    );  
  }  
}
```



99

Output:





Post Practical Questions

1. What is Flutter?
 - A. Flutter is an open-source backend development framework
 - B. Flutter is an open-source UI toolkit
 - C. **Flutter is an open-source programming language for cross-platform applications**
 - D. Flutter is a DBMS toolkit
2. How many types of widgets are there in Flutter?
 - A. **2**
 - B. 4
 - C. 6
 - D. 8+
3. What element is used as an identifier for components when programming in Flutter?
 - A. Widgets
 - B. **Keys**
 - C. Elements
 - D. Serial
4. What command would you use to compile your Flutter app in release mode?
 - A. Flutter --release
 - B. Flutter build --release
 - C. **Flutter run --release**
 - D. Flutter run \$release

References:

1. <https://www.javatpoint.com/flutter-dart-programming>

Conclusion:

In conclusion we create a simple UI with text displayed in the center of the screen, with padding, margin, and alignment applied.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-13

DATE: _____

AIM: Create a stateless widget and a stateful widget and use them in your main application.

StatefulWidget class:

State is information that (1) can be read synchronously when the widget is built and (2) might change during the lifetime of the widget. It is the responsibility of the widget implementer to ensure that the State is promptly notified when such state changes, using `State.setState`.

A stateful widget is a widget that describes part of the user interface by building a constellation of other widgets that describe the user interface more concretely. The building process continues recursively until the description of the user interface is fully concrete (e.g., consists entirely of `RenderObjectWidgets`, which describe concrete `RenderObjects`).

Practical Code:

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Stateless & Stateful Widgets Example'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              StatelessWidgetExample(),
              SizedBox(height: 20),
              StatefulWidgetExample(),
            ],
          ),
        ),
      ),
    );
  }
}
```

```
),  
);  
}  
}
```

```
class StatelessWidgetExample extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Container(  
      padding: EdgeInsets.all(10),  
      color: Colors.green,  
      child: Text(  
        'I am a StatelessWidget',  
        style: TextStyle(fontSize: 20, color: Colors.white),  
      ),  
    );  
  }  
}
```

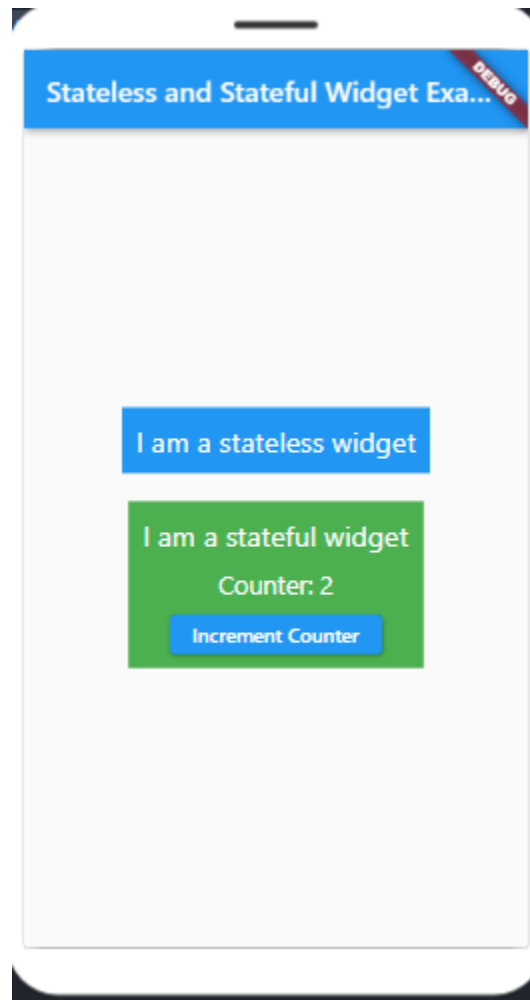
```
class StatefulWidgetExample extends StatefulWidget {  
  @override  
  _StatefulWidgetExampleState createState() => _StatefulWidgetExampleState();  
}
```

```
class _StatefulWidgetExampleState extends State<StatefulWidgetExample> {  
  bool _isToggled = false;  
  
  void toggleState() {  
    setState(() {  
      _isToggled = !_isToggled;  
    });  
  }  
}
```

```
@override  
Widget build(BuildContext context) {  
  return GestureDetector(  
    onTap: toggleState,  
    child: Container(  
      padding: EdgeInsets.all(10),  
      color: _isToggled ? Colors.blue : Colors.red,  
      child: Text(  
        _isToggled ? 'Stateful Widget: ON' : 'Stateful Widget: OFF',  
        style: TextStyle(fontSize: 20, color: Colors.white),  
      ),  
    ),  
  );  
};
```

```
}  
}
```

Output:



Short Answer - Questions

1. Which function will return the widgets attached to the screen as a root of the widget tree to be rendered on screen?
 - A. main()
 - B. runApp()**
 - C. container()
 - D. root()
2. What is the key configuration file used when building a Flutter project?
 - A. pubspec.yaml**
 - B. pubspec.xml



- C. config.html
- D. root.xml
- 3. Which component allows us to specify the distance between widgets on the screen?
 - A. SafeArea
 - B. **SizedBox**
 - C. table
 - D. AppBar
- 4. Which widget type allows you to modify its appearance dynamically according to user input?
 - A. **Stateful widget**
 - B. Stateless widget
- 5. What command would you run to verify your Flutter install and ensure your environment is set up correctly?
 - A. Flutter run
 - B. Flutter build
 - C. **Flutter doctor**
 - D. Flutter help

References:

1. <https://www.javatpoint.com/flutter-dart-programming>

Conclusion:

In conclusion, when we run this program we'll see both widgets displayed on the screen, demonstrating the use of both stateless and stateful widgets.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-14

DATE: _____

AIM: Create an app with different basic layouts, such as Column, Row, and Container.

Practical Code:

```
import 'package:flutter/material.dart';

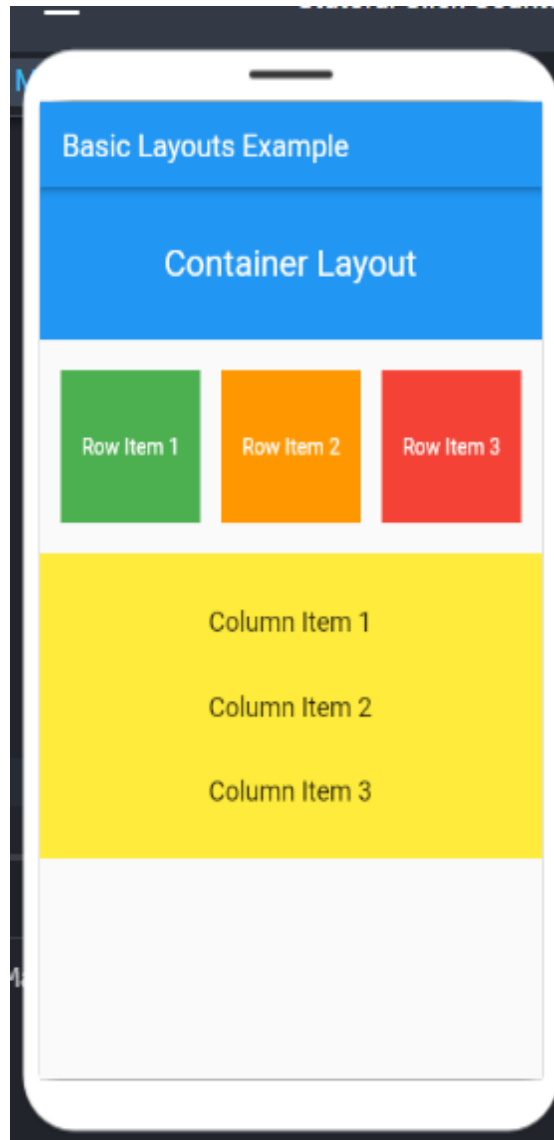
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Basic Layouts'),
        ),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              Text('Column Layout'),
              SizedBox(height: 20),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text('Row Layout'),
```



```
SizedBox(width: 20),
Container(
  width: 100,
  height: 100,
  color: Colors.blue,
  child: Center(
    child: Text('Container Layout'),
  ),
),
],
),
],
),
),
),
);
}
```

Output:





MCQ - Questions

1. What widget would you use for repeating content in Flutter?
 - A. ExpandedView
 - B. **ListView**
 - C. Stack
 - D. ArrayView
2. Does flutter support desktop application development?
 - A. **Yes**
 - B. No
3. Which of the following language is used to build flutter app?
 - A. Go
 - B. Java
 - C. **Dart**
 - D. React

References:

1. <https://www.javatpoint.com/flutter-dart-programming>
2. <https://www.linkedin.com/pulse/flutter-multiple-choice-questions-onlineinterviewquestions>

Conclusion:

In conclusion, when you run this app, you'll see both widgets displayed on the screen, demonstrating the use of both stateless and stateful widgets.

Marks out of 10	
Signature with Date of Completion	



PRACTICAL-15

DATE: _____

AIM: Create forms for user input with validation.

Practical Code:

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Form with Validation Example'),
        ),
        body: MyForm(),
      ),
    );
  }
}

class MyForm extends StatefulWidget {
  @override
  _MyFormState createState() => _MyFormState();
}

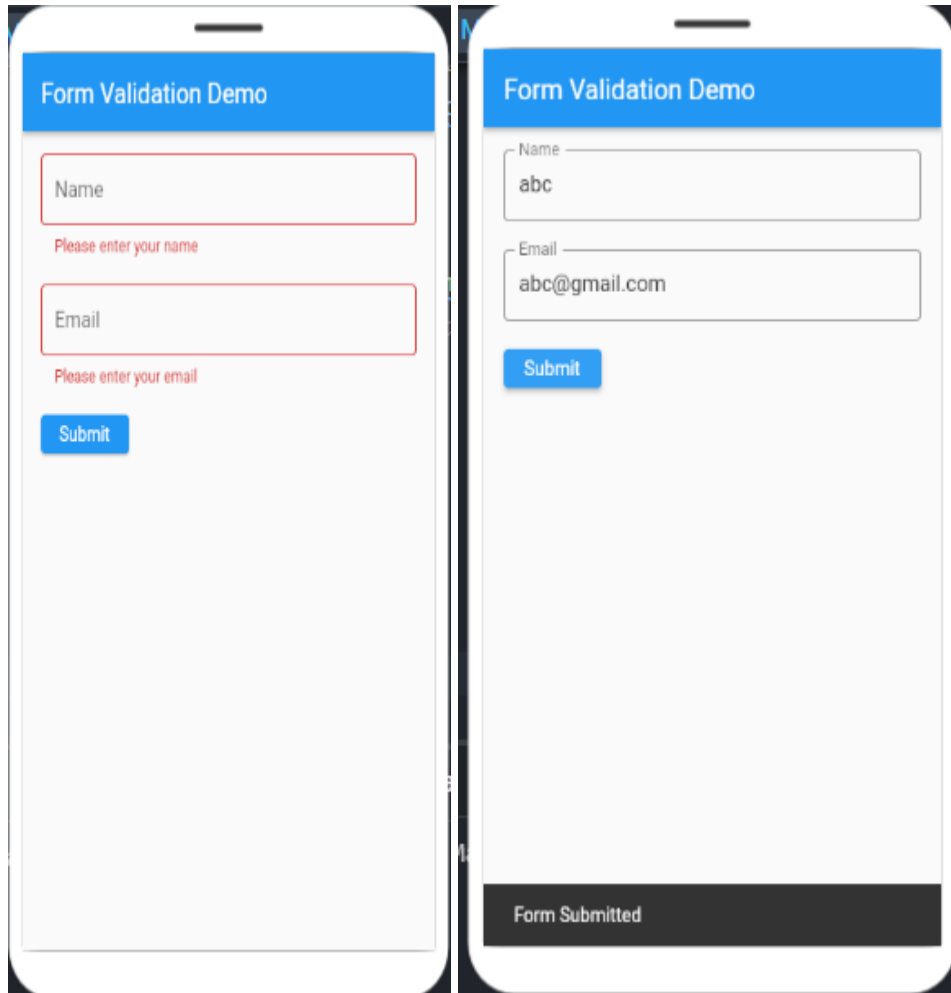
class _MyFormState extends State<MyForm> {
  final _formKey = GlobalKey<FormState>();
  String _name = "";
  String _email = "";

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Padding(
        padding: EdgeInsets.all(20),
```

111

```
);  
}  
}
```

Output:



The image displays two side-by-side mobile app screenshots of a 'Form Validation Demo'.

Left Screenshot: The app has a blue header with the title 'Form Validation Demo'. Below it, there are two input fields. The first field is labeled 'Name' and has a red border with the error message 'Please enter your name' below it. The second field is labeled 'Email' and also has a red border with the error message 'Please enter your email' below it. At the bottom, there is a blue 'Submit' button.

Right Screenshot: The app has the same blue header. The 'Name' field now contains the text 'abc' and the 'Email' field contains 'abc@gmail.com'. The error messages are gone. The blue 'Submit' button is still present. At the bottom of the screen, there is a black bar with the white text 'Form Submitted'.



Post Practical Questions

1. Everything is a widget in Flutter. True or False?
A. True
B. False
2. Which of the following is used to load images from the flutter project's assets?
A. Image
B. Image.file
C. Image.asset
D. Image.memory
3. The most important properties of the Image widget are
A. width, double
B. height, double
C. image, ImageProvider
D. all of the above

References:

1. <https://www.javatpoint.com/flutter-dart-programming>

Conclusion:

In conclusion, when the form is submitted, the captured form data is printed to the console. You can replace this with any action you want to perform with the form data.

Marks out of 10	
Signature with Date of Completion	