**SRM Institute of Science and Technology**

**Department of Computer Applications**

**Delhi – Meerut Road, Sikri Kalan, Ghaziabad, Uttar Pradesh – 201204**

**Circular – 2020-21**

**MCA 2nd semester**
**ANDROID APPLICATIONS DEVELOPMENT (PCA20D04J)**

**Lab Manual**

# Lab 1: Login Page Creation with Toast Message

## Title

Android Lab 1: Login Page with Toast Message

## Aim

To create a simple Android login page with input fields for username and password, and display a `Toast` message indicating the login status upon a button click.

## Procedure

1. **Create a New Android Project:**
    - Open Android Studio and start a new project.
    - Select "Empty Activity" template.
    - Configure your project with a suitable name (e.g., `LoginPageApp`), package name, and language (Java/Kotlin).
2. **Design the Layout (`activity_main.xml`):**
    - Open `activity_main.xml` (or your main activity's layout file) in the `res/layout` directory.
    - Use a `ConstraintLayout` or `LinearLayout` as the root.
    - Add two `EditText` widgets: one for username (set `android:hint="Username"`) and one for password (set `android:hint="Password"` and `android:inputType="textPassword"`). Assign unique IDs (e.g., `usernameEditText`, `passwordEditText`).
    - Add a `Button` widget (set `android:text="Login"`) and assign an ID (e.g., `loginButton`).
    - Arrange these elements appropriately using constraints or layout weights.
3. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - Open `MainActivity.java` (or `MainActivity.kt`).
    - In the `onCreate()` method, get references to the `EditText` and `Button` views using `findViewById()`.
    - Set an `OnClickListener` for the login button.
    - Inside the `onClick` method:
        - Retrieve the text from the username and password `EditText` fields.
        - Convert the retrieved text to `String`.
        - Implement a simple login validation (e.g., if username is "admin" and password is "password").

- Based on the validation, display an appropriate `Toast` message (e.g., "Login Successful" or "Invalid Credentials").

## Source Code

*activity_main.xml* **(Layout)**

```
<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:padding="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/loginTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="User Login"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"/>

    <EditText
        android:id="@+id/usernameEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/loginTitle"
        android:layout_marginTop="40dp"
        android:hint="Username"
        android:inputType="text" />

    <EditText
        android:id="@+id/passwordEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/usernameEditText"
        android:layout_marginTop="20dp"
        android:hint="Password"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/loginButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/passwordEditText"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:text="Login" />

</RelativeLayout>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.loginpageapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
```

```
    private EditText usernameEditText;
    private EditText passwordEditText;
    private Button loginButton;

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

        // Get references to the views
        usernameEditText = findViewById(R.id.usernameEditText);
        passwordEditText = findViewById(R.id.passwordEditText);
        loginButton = findViewById(R.id.loginButton);

        // Set an OnClickListener for the login button
        loginButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Get the text from EditText fields
                String username = usernameEditText.getText().toString();
                String password = passwordEditText.getText().toString();

                // Simple validation
                if (username.equals("admin") && password.equals("password")) {
                    Toast.makeText(MainActivity.this, "Login Successful!",
Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(MainActivity.this, "Invalid Credentials",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

## Input

- **Scenario 1 (Valid):**
  - Username: `admin`
  - Password: `password`
- **Scenario 2 (Invalid):**
  - Username: `user123`
  - Password: `wrongpass`

## Expected Output

- **Scenario 1 (Valid):** A short `Toast` message appearing at the bottom of the screen saying "Login Successful!".
- **Scenario 2 (Invalid):** A short `Toast` message appearing at the bottom of the screen saying "Invalid Credentials".

# Lab 2: Student Registration Form with Toast Message

## Title

Android Lab 2: Student Registration Form with Toast Message

## Aim

To develop an Android application that allows users to fill out a student registration form and displays a `Toast` message upon successful submission.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Use a `ScrollView` as the root to ensure all fields are visible on smaller screens.
    - Inside the `ScrollView`, use a `LinearLayout` with `android:orientation="vertical"`.
    - Add multiple `EditText` widgets for student details such as:
        - Name (`android:hint="Student Name"`)
        - Roll Number (`android:hint="Roll Number"`, `android:inputType="number"`)
        - Course (`android:hint="Course"`)
        - Email (`android:hint="Email"`, `android:inputType="textEmailAddress"`)
        - Phone Number (`android:hint="Phone Number"`, `android:inputType="phone"`)
    - Add a `Button` widget (set `android:text="Register"`) and assign an ID (e.g., `registerButton`).
    - Assign unique IDs to all `EditText` fields.
3. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get references to all `EditText` and `Button` views.
    - Set an `OnClickListener` for the register button.
    - Inside the `onClick` method:
        - Retrieve the text from all `EditText` fields.
        - Perform basic validation (e.g., check if fields are empty).
        - If all fields are valid, display a `Toast` message like "Student Registered Successfully!".
        - Optionally, clear the `EditText` fields after successful registration.

## Source Code

*`activity_main.xml` (Layout)*

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">

        <TextView
            android:id="@+id/registrationTitle"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Student Registration"
            android:textSize="24sp"
            android:textStyle="bold"
            android:layout_gravity="center_horizontal"
            android:layout_marginBottom="30dp"/>

        <EditText
            android:id="@+id/nameEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Student Name"
            android:inputType="textPersonName"
            android:layout_marginBottom="15dp"/>

        <EditText
            android:id="@+id/rollNumberEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Roll Number"
            android:inputType="number"
            android:layout_marginBottom="15dp"/>

        <EditText
            android:id="@+id/courseEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Course"
            android:inputType="text"
            android:layout_marginBottom="15dp"/>

        <EditText
            android:id="@+id/emailEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Email"
            android:inputType="textEmailAddress"
            android:layout_marginBottom="15dp"/>

        <EditText
            android:id="@+id/phoneEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Phone Number"
            android:inputType="phone"
            android:layout_marginBottom="30dp"/>

        <Button
            android:id="@+id/registerButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Register"
            android:layout_gravity="center_horizontal"/>

    </LinearLayout>
</ScrollView>
```

*MainActivity.java* ***(Logic - Java)***

```
// MainActivity.java
```

```
package com.example.studentregistrationapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private EditText nameEditText, rollNumberEditText, courseEditText,
emailEditText, phoneEditText;
    private Button registerButton;

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

        // Get references to the views
        nameEditText = findViewById(R.id.nameEditText);
        rollNumberEditText = findViewById(R.id.rollNumberEditText);
        courseEditText = findViewById(R.id.courseEditText);
        emailEditText = findViewById(R.id.emailEditText);
        phoneEditText = findViewById(R.id.phoneEditText);
        registerButton = findViewById(R.id.registerButton);

        // Set an OnClickListener for the register button
        registerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Get the text from EditText fields
                String name = nameEditText.getText().toString().trim();
                String rollNumber =
rollNumberEditText.getText().toString().trim();
                String course = courseEditText.getText().toString().trim();
                String email = emailEditText.getText().toString().trim();
                String phone = phoneEditText.getText().toString().trim();

                // Basic validation: Check if any field is empty
                if (name.isEmpty() || rollNumber.isEmpty() || course.isEmpty()
|| email.isEmpty() || phone.isEmpty()) {
                    Toast.makeText(MainActivity.this, "Please fill all fields!",
Toast.LENGTH_SHORT).show();
                } else {
                    // All fields are filled, display success message
                    String message = "Student Registered Successfully!\n" +
                                     "Name: " + name + "\n" +
                                     "Roll No: " + rollNumber + "\n" +
                                     "Course: " + course + "\n" +
                                     "Email: " + email + "\n" +
                                     "Phone: " + phone;
                    Toast.makeText(MainActivity.this, message,
Toast.LENGTH_LONG).show();

                    // Optionally, clear the fields after registration
                    nameEditText.setText("");
                    rollNumberEditText.setText("");
                    courseEditText.setText("");
                    emailEditText.setText("");
                    phoneEditText.setText("");
                }
            }
        });
    }
}
```

## Input

User fills in the following details:

- Student Name: `Alice Smith`
- Roll Number: `12345`
- Course: `Computer Science`
- Email: `alice.smith@example.com`
- Phone Number: `9876543210` Then clicks the "Register" button.

## Expected Output

A long `Toast` message appearing at the bottom of the screen with details similar to:

```
Student Registered Successfully!
Name: Alice Smith
Roll No: 12345
Course: Computer Science
Email: alice.smith@example.com
Phone: 9876543210
```

# Lab 3: Implement Explicit Intent

## Title

Android Lab 3: Implementing Explicit Intent

## Aim

To understand and implement Explicit Intents to navigate from one Android Activity to another within the same application.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Create a Second Activity:**
    - In Android Studio, right-click on your package name in the Project pane (`app/java/your.package.name`).
    - Select `New` -> `Activity` -> `Empty Activity`.
    - Name it `SecondActivity` (or any other descriptive name) and click Finish. This will create `SecondActivity.java/.kt` and `activity_second.xml`.
3. **Design Layouts:**
    - **`activity_main.xml`:** Add a `TextView` (e.g., "Main Activity") and a `Button` (e.g., "Go to Second Activity"). Assign an ID to the button (e.g., `goToSecondActivityButton`).
    - **`activity_second.xml`:** Add a `TextView` (e.g., "Second Activity") to confirm navigation.
4. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `MainActivity.java`, get a reference to the `goToSecondActivityButton`.
    - Set an `OnClickListener` for this button.
    - Inside the `onClick` method:
        - Create an `Intent` object, explicitly specifying the current activity and the target activity: `Intent intent = new Intent(MainActivity.this, SecondActivity.class);`
        - Start the new activity using `startActivity(intent);`.

## Source Code

*`activity_main.xml` (Layout for Main Activity)*

```
<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:padding="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/mainActivityTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Main Activity"
        android:textSize="28sp"
        android:textStyle="bold"
        android:layout_centerHorizontal="true"
```

```
        android:layout_marginTop="100dp"/>

    <Button
        android:id="@+id/goToSecondActivityButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go to Second Activity"
        android:layout_below="@id/mainActivityTitle"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"/>

</RelativeLayout>
```

*activity_second.xml* **(Layout for Second Activity)**

```
<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:padding="16dp"
    tools:context=".SecondActivity">

    <TextView
        android:id="@+id/secondActivityTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to Second Activity!"
        android:textSize="28sp"
        android:textStyle="bold"
        android:layout_centerInParent="true"/>

</RelativeLayout>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.explicitintentapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private Button goToSecondActivityButton;

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

        goToSecondActivityButton = findViewById(R.id.goToSecondActivityButton);

        goToSecondActivityButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Create an Explicit Intent to start SecondActivity
                Intent intent = new Intent(MainActivity.this,
SecondActivity.class);
                startActivity(intent); // Start the new activity
            }
        });
    }
}
```

*SecondActivity.java* **(Logic - Java)**

```
// SecondActivity.java
package com.example.explicitintentapp; // Replace with your package name

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

public class SecondActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        // No specific logic needed here for this lab, just displaying the
layout
    }
}
```

## Input

User taps the "Go to Second Activity" button on the main screen.

## Expected Output

The screen transitions from "Main Activity" to "Welcome to Second Activity!".

# Lab 4: Implement Implicit Intent

## Title

Android Lab 4: Implementing Implicit Intent

## Aim

To understand and implement Implicit Intents to request an action from the Android system, allowing the user to choose an application to handle the request (e.g., open a web page, make a call, send an email).

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Add several `Button` widgets, each for a different implicit intent action:
        - "Open Web Page" (e.g., `openWebButton`)
        - "Make a Call" (e.g., `makeCallButton`)
        - "Send Email" (e.g., `sendEmailButton`)
    - You might also add `EditText` fields for URL, phone number, or email recipient/subject if you want to allow user input. For simplicity, we'll hardcode values in the example.
3. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get references to all `Button` views.
    - Set `OnClickListener` for each button.
    - Inside each `onClick` method:
        - **For "Open Web Page":**
            - Create an `Intent` with `Intent.ACTION_VIEW`.
            - Set the data URI using `Uri.parse("http://www.google.com")`.
            - Start the activity: `startActivity(intent);`
        - **For "Make a Call":**
            - Create an `Intent` with `Intent.ACTION_DIAL`.
            - Set the data URI using `Uri.parse("tel:1234567890")`.
            - Start the activity: `startActivity(intent);` (Note: `ACTION_CALL` requires `CALL_PHONE` permission and directly initiates a call; `ACTION_DIAL` opens the dialer with the number pre-filled).
        - **For "Send Email":**
            - Create an `Intent` with `Intent.ACTION_SENDTO`.
            - Set the data URI using `Uri.parse("mailto:recipient@example.com")`.
            - Add extra data for subject and body using `intent.putExtra(Intent.EXTRA_SUBJECT, "Subject Here")` and `intent.putExtra(Intent.EXTRA_TEXT, "Email body here")`.
            - Start the activity: `startActivity(intent);`
4. **Add Permissions (if necessary) in `AndroidManifest.xml`:**
    - For `ACTION_DIAL`, no special permission is typically needed as it opens the dialer. For `ACTION_CALL`, you would need `<uses-permission android:name="android.permission.CALL_PHONE" />`.

## Source Code

*activity_main.xml* **(Layout)**

```
<LinearLayout 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"
    android:padding="16dp"
    android:gravity="center"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/openWebButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open Web Page"
        android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/makeCallButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Make a Call"
        android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/sendEmailButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send Email"
        android:layout_marginBottom="20dp"/>

</LinearLayout>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.implicitintentapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private Button openWebButton, makeCallButton, sendEmailButton;

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

        openWebButton = findViewById(R.id.openWebButton);
        makeCallButton = findViewById(R.id.makeCallButton);
        sendEmailButton = findViewById(R.id.sendEmailButton);

        // Open Web Page Button
        openWebButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String url = "http://www.google.com";
                Intent intent = new Intent(Intent.ACTION_VIEW);
```

```
                intent.setData(Uri.parse(url));
                startActivity(intent);
            }
        });

        // Make a Call Button
        makeCallButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String phoneNumber = "tel:1234567890";
                Intent intent = new Intent(Intent.ACTION_DIAL); // Opens dialer
with number pre-filled
                intent.setData(Uri.parse(phoneNumber));
                startActivity(intent);
            }
        });

        // Send Email Button
        sendEmailButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String recipient = "recipient@example.com";
                String subject = "Hello from Android App";
                String body = "This is a test email sent from my Android
application using an implicit intent.";

                Intent intent = new Intent(Intent.ACTION_SENDTO);
                intent.setData(Uri.parse("mailto:")); // Only email apps should
handle this
                intent.putExtra(Intent.EXTRA_EMAIL, new String[]{recipient});
                intent.putExtra(Intent.EXTRA_SUBJECT, subject);
                intent.putExtra(Intent.EXTRA_TEXT, body);

                // Check if there's an app to handle this intent
                if (intent.resolveActivity(getPackageManager()) != null) {
                    startActivity(Intent.createChooser(intent, "Send Email
Using..."));
                } else {
                    Toast.makeText(MainActivity.this, "No email client
installed.", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

## Input

User taps on any of the three buttons: "Open Web Page", "Make a Call", or "Send Email".

## Expected Output

- **"Open Web Page":** The device's default web browser opens and navigates to `http://www.google.com`.
- **"Make a Call":** The device's phone dialer opens with "1234567890" pre-filled.
- **"Send Email":** A chooser dialog appears, allowing the user to select an email client. The selected email client opens with the recipient, subject, and body pre-filled. If no email client is installed, a Toast message "No email client installed." appears.

# Lab 5: Implement Time Picker

## Title

Android Lab 5: Implementing Time Picker

## Aim

To integrate a `TimePicker` widget into an Android application, allowing users to select a specific time, and then display the selected time.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Add a `TextView` to display the selected time (e.g., `selectedTimeTextView`).
    - Add a `Button` (e.g., `pickTimeButton`) that, when clicked, will launch the `TimePickerDialog`.
3. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get references to the `TextView` and `Button`.
    - Set an `OnClickListener` for the `pickTimeButton`.
    - Inside the `onClick` method:
        - Get the current hour and minute to set as the default for the `TimePickerDialog`.
        - Create a new `TimePickerDialog` instance.
        - Pass the current context, a `TimePickerDialog.OnTimeSetListener`, and the initial hour and minute.
        - The `OnTimeSetListener` will be triggered when the user selects a time and clicks "OK". Inside this listener, retrieve the selected hour and minute.
        - Format the selected time (e.g., "HH:mm AM/PM") and set it to the `selectedTimeTextView`.
        - Call `timePickerDialog.show()` to display the dialog.

## Source Code

*activity_main.xml (Layout)*

```
<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:padding="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/selectedTimeTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="No time selected"
        android:textSize="22sp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="100dp"/>

    <Button
```

```
        android:id="@+id/pickTimeButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pick Time"
        android:layout_below="@id/selectedTimeTextView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"/>

</RelativeLayout>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.timepickerapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.TimePicker;
import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    private TextView selectedTimeTextView;
    private Button pickTimeButton;

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

        selectedTimeTextView = findViewById(R.id.selectedTimeTextView);
        pickTimeButton = findViewById(R.id.pickTimeButton);

        pickTimeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Get current time for default values
                final Calendar c = Calendar.getInstance();
                int hour = c.get(Calendar.HOUR_OF_DAY);
                int minute = c.get(Calendar.MINUTE);

                // Create a new TimePickerDialog
                TimePickerDialog timePickerDialog = new
TimePickerDialog(MainActivity.this,
                        new TimePickerDialog.OnTimeSetListener() {
                            @Override
                            public void onTimeSet(TimePicker view, int
hourOfDay, int minute) {
                                // Format the selected time
                                String amPm;
                                int displayHour = hourOfDay;
                                if (hourOfDay >= 12) {
                                    amPm = "PM";
                                    if (hourOfDay > 12) {
                                        displayHour = hourOfDay - 12;
                                    }
                                } else {
                                    amPm = "AM";
                                    if (hourOfDay == 0) {
                                        displayHour = 12; // 12 AM
                                    }
                                }
```

```
                                String formattedTime = String.format("%02d:%02d
%s", displayHour, minute, amPm);
                                selectedTimeTextView.setText("Selected Time: " +
formattedTime);
                            }
                        }, hour, minute, false); // false for 12-hour format,
true for 24-hour format

                timePickerDialog.show(); // Show the dialog
            }
        });
    }
}
```

## Input

User taps the "Pick Time" button, then selects a time (e.g., 03:30 PM) from the `TimePickerDialog` and taps "OK".

## Expected Output

The `TextView` on the screen updates to display "Selected Time: 03:30 PM" (or the chosen time in the specified format).

# Lab 6: Implement Date Picker

## Title

Android Lab 6: Implementing Date Picker

## Aim

To integrate a `DatePicker` widget into an Android application, allowing users to select a specific date, and then display the selected date.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Add a `TextView` to display the selected date (e.g., `selectedDateTextView`).
    - Add a `Button` (e.g., `pickDateButton`) that, when clicked, will launch the `DatePickerDialog`.
3. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get references to the `TextView` and `Button`.
    - Set an `OnClickListener` for the `pickDateButton`.
    - Inside the `onClick` method:
        - Get the current year, month, and day to set as the default for the `DatePickerDialog`.
        - Create a new `DatePickerDialog` instance.
        - Pass the current context, a `DatePickerDialog.OnDateSetListener`, and the initial year, month, and day.
        - The `OnDateSetListener` will be triggered when the user selects a date and clicks "OK". Inside this listener, retrieve the selected year, month (which is 0-indexed), and day of month.
        - Format the selected date (e.g., "DD/MM/YYYY") and set it to the `selectedDateTextView`.
        - Call `datePickerDialog.show()` to display the dialog.

## Source Code

*`activity_main.xml` (Layout)*

```
<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:padding="16dp"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/selectedDateTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="No date selected"
        android:textSize="22sp"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="100dp"/>
```

```
    <Button
        android:id="@+id/pickDateButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Pick Date"
        android:layout_below="@id/selectedDateTextView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="50dp"/>

</RelativeLayout>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.datepickerapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.app.DatePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;
import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    private TextView selectedDateTextView;
    private Button pickDateButton;

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

        selectedDateTextView = findViewById(R.id.selectedDateTextView);
        pickDateButton = findViewById(R.id.pickDateButton);

        pickDateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Get current date for default values
                final Calendar c = Calendar.getInstance();
                int year = c.get(Calendar.YEAR);
                int month = c.get(Calendar.MONTH); // Month is 0-indexed
                int day = c.get(Calendar.DAY_OF_MONTH);

                // Create a new DatePickerDialog
                DatePickerDialog datePickerDialog = new
DatePickerDialog(MainActivity.this,
                        new DatePickerDialog.OnDateSetListener() {
                            @Override
                            public void onDateSet(DatePicker view, int year, int
monthOfYear, int dayOfMonth) {
                                // Format the selected date (monthOfYear is 0-
indexed, so add 1)
                                String formattedDate =
String.format("%02d/%02d/%d", dayOfMonth, (monthOfYear + 1), year);
                                selectedDateTextView.setText("Selected Date: " +
formattedDate);
                            }
                        }, year, month, day);

                datePickerDialog.show(); // Show the dialog
            }
        });
    }
```

```
}
```

## Input

User taps the "Pick Date" button, then selects a date (e.g., May 22, 2025) from the `DatePickerDialog` and taps "OK".

## Expected Output

The `TextView` on the screen updates to display "Selected Date: 22/05/2025" (or the chosen date in the specified format).

# Lab 7: Student Registration Form using List View

## Title

Android Lab 7: Student Registration Form with ListView

## Aim

To create a student registration form and dynamically display the registered student details in a `ListView`.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Use a `LinearLayout` with vertical orientation as the root.
    - Add `EditText` widgets for student details (Name, Roll Number, Course, etc.) similar to Lab 2.
    - Add a "Register" `Button`.
    - Add a `ListView` widget below the form elements. Assign an ID (e.g., `studentListView`).
3. **Create a Model Class (Optional but Recommended):**
    - Create a simple Java/Kotlin class (e.g., `Student.java`) to hold student data (name, roll number, course). This makes managing data easier.
4. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get references to all `EditText` fields, the "Register" `Button`, and the `ListView`.
    - Create an `ArrayList` to store `String` representations of student data (or `Student` objects if you created a model class).
    - Create an `ArrayAdapter` to bridge the `ArrayList` data to the `ListView`. Initialize it with `this`, `android.R.layout.simple_list_item_1` (for a simple text item layout), and your `ArrayList`.
    - Set the `ArrayAdapter` to the `studentListView`.
    - Set an `OnClickListener` for the "Register" button.
    - Inside the `onClick` method:
        - Retrieve data from `EditText` fields.
        - Perform basic validation.
        - If valid, create a `String` representation of the student (e.g., "Name: [Name], Roll: [Roll No]") or a `Student` object.
        - Add this `String`/`Student` object to your `ArrayList`.
        - Notify the `ArrayAdapter` that the data has changed using `adapter.notifyDataSetChanged()`.
        - Optionally, clear the `EditText` fields.

## Source Code

*`activity_main.xml` (Layout)*

```
<LinearLayout 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"
    android:padding="16dp"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Student Registration"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_gravity="center_horizontal"
        android:layout_marginBottom="20dp"/>

    <EditText
        android:id="@+id/nameEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Student Name"
        android:layout_marginBottom="10dp"/>

    <EditText
        android:id="@+id/rollNumberEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Roll Number"
        android:inputType="number"
        android:layout_marginBottom="10dp"/>

    <EditText
        android:id="@+id/courseEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Course"
        android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/registerButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Register Student"
        android:layout_gravity="center_horizontal"
        android:layout_marginBottom="30dp"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Registered Students:"
        android:textSize="18sp"
        android:textStyle="bold"
        android:layout_marginBottom="10dp"/>

    <ListView
        android:id="@+id/studentListView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#f0f0f0"
        android:padding="5dp"/>

</LinearLayout>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.studentlistviewapp; // Replace with your package name

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

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

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private EditText nameEditText, rollNumberEditText, courseEditText;
    private Button registerButton;
    private ListView studentListView;
    private ArrayList<String> studentList; // Stores student data as strings
    private ArrayAdapter<String> adapter; // Adapter to link data to ListView

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

        // Get references to views
        nameEditText = findViewById(R.id.nameEditText);
        rollNumberEditText = findViewById(R.id.rollNumberEditText);
        courseEditText = findViewById(R.id.courseEditText);
        registerButton = findViewById(R.id.registerButton);
        studentListView = findViewById(R.id.studentListView);

        // Initialize ArrayList and ArrayAdapter
        studentList = new ArrayList<>();
        adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1,
studentList);
        studentListView.setAdapter(adapter);

        // Set OnClickListener for the register button
        registerButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String name = nameEditText.getText().toString().trim();
                String rollNumber =
rollNumberEditText.getText().toString().trim();
                String course = courseEditText.getText().toString().trim();

                // Basic validation
                if (name.isEmpty() || rollNumber.isEmpty() || course.isEmpty())
{
                    Toast.makeText(MainActivity.this, "Please fill all fields!",
Toast.LENGTH_SHORT).show();
                } else {
                    // Create a string representation of the student
                    String studentInfo = "Name: " + name + ", Roll No: " +
rollNumber + ", Course: " + course;

                    // Add to ArrayList and notify adapter
                    studentList.add(studentInfo);
                    adapter.notifyDataSetChanged(); // Refresh the ListView

                    Toast.makeText(MainActivity.this, "Student Registered!",
Toast.LENGTH_SHORT).show();

                    // Clear input fields
                    nameEditText.setText("");
                    rollNumberEditText.setText("");
                    courseEditText.setText("");
                }
```

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

## Input

User fills in student details and taps "Register Student" multiple times:

1. Name: `John Doe`, Roll No: `101`, Course: `Physics`
2. Name: `Jane Smith`, Roll No: `102`, Course: `Chemistry`
3. Name: `Peter Jones`, Roll No: `103`, Course: `Math`

## Expected Output

The `ListView` will display the registered students, with each student's details as a separate item:

```
Name: John Doe, Roll No: 101, Course: Physics
Name: Jane Smith, Roll No: 102, Course: Chemistry
Name: Peter Jones, Roll No: 103, Course: Math
```

Additionally, a "Student Registered!" Toast message will appear after each successful registration.

# Lab 8: Implement Context Menu

## Title

Android Lab 8: Implementing Context Menu

## Aim

To implement a Context Menu that appears when a user performs a long-press on a specific `View` in an Android application, and handle the selection of menu items.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Add a `TextView` (e.g., `myTextView`) that will serve as the target for the context menu. Set some text like "Long press me for options".
    - Optionally, add another `TextView` to display which context menu item was clicked.
3. **Create a Menu Resource File:**
    - In the `res` directory, right-click on `res` -> `New` -> `Android Resource Directory`.
    - Select `menu` as the Resource type. Click OK.
    - Right-click on the newly created `menu` directory -> `New` -> `Menu Resource File`.
    - Name it `context_menu.xml` (or any descriptive name).
    - Define menu items within this file using `<item>` tags, each with an `android:id` and `android:title`.
4. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get a reference to the `myTextView`.
    - **Register the View for Context Menu:** Call `registerForContextMenu(myTextView);`.
    - **Override `onCreateContextMenu()`:** This method is called when the registered view is long-pressed.
        - Inside this method, inflate your `context_menu.xml` file into the `ContextMenu` object using `getMenuInflater().inflate(R.menu.context_menu, menu);`.
        - Set a header for the menu if desired.
    - **Override `onContextItemSelected()`:** This method is called when a context menu item is selected.
        - Use a `switch` statement on `item.getItemId()` to identify which menu item was clicked.
        - Perform the desired action for each item (e.g., display a `Toast` message with the item's title).
        - Return `true` to indicate that the event was handled.

## Source Code

*`activity_main.xml` (Layout)*

```
<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:padding="16dp"
```

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

    <TextView
        android:id="@+id/myTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Long press me for options"
        android:textSize="24sp"
        android:textStyle="bold"
        android:padding="20dp"
        android:background="#E0E0E0"
        android:layout_centerInParent="true"/>

    <TextView
        android:id="@+id/statusTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/myTextView"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="30dp"
        android:textSize="18sp"
        android:text="Status: No item selected"/>

</RelativeLayout>
```

*res/menu/context_menu.xml* **(Menu Resource)**

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/option_edit"
        android:title="Edit" />
    <item
        android:id="@+id/option_delete"
        android:title="Delete" />
    <item
        android:id="@+id/option_share"
        android:title="Share" />
</menu>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.contextmenuapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private TextView myTextView;
    private TextView statusTextView;

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

        myTextView = findViewById(R.id.myTextView);
        statusTextView = findViewById(R.id.statusTextView);

        // Register the TextView for a context menu
        registerForContextMenu(myTextView);
    }
```

```
    // Called when a context menu is being built for the registered view
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);
        // Inflate the menu resource file
        getMenuInflater().inflate(R.menu.context_menu, menu);
        menu.setHeaderTitle("Choose an option"); // Optional: Set a header for
the menu
    }

    // Called when a context menu item is selected
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        String selectedOption = "";
        int itemId = item.getItemId();

        if (itemId == R.id.option_edit) {
            selectedOption = "Edit";
        } else if (itemId == R.id.option_delete) {
            selectedOption = "Delete";
        } else if (itemId == R.id.option_share) {
            selectedOption = "Share";
        } else {
            return super.onContextItemSelected(item); // Handle unhandled items
        }

        Toast.makeText(this, selectedOption + " selected",
Toast.LENGTH_SHORT).show();
        statusTextView.setText("Status: " + selectedOption + " was clicked.");
        return true; // Indicate that the item has been handled
    }
}
```

## Input

User performs a long-press on the "Long press me for options" `TextView`. Then, the user taps on one of the context menu items (e.g., "Delete").

## Expected Output

1. Upon long-press: A floating context menu appears with "Edit", "Delete", and "Share" options, and a header "Choose an option".
2. Upon tapping "Delete": A `Toast` message "Delete selected" appears, and the "Status" `TextView` updates to "Status: Delete was clicked.".

# Lab 9: Implement Option Menu

## Title

Android Lab 9: Implementing Option Menu

## Aim

To implement an Options Menu (also known as the Action Bar menu) in an Android application, which typically appears at the top right of the screen, and handle the selection of menu items.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Add a `TextView` (e.g., `statusTextView`) to display which option menu item was clicked.
3. **Create a Menu Resource File:**
    - If you don't have one, create a `menu` directory in `res` (as in Lab 8).
    - Create a new Menu Resource File, e.g., `options_menu.xml`.
    - Define menu items within this file using `<item>` tags, each with an `android:id` and `android:title`. You can also add `android:icon` and `app:showAsAction` attributes.
4. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get a reference to the `statusTextView`.
    - **Override `onCreateOptionsMenu()`:** This method is called when the activity is first created to inflate the options menu.
        - Inflate your `options_menu.xml` file into the `Menu` object using `getMenuInflater().inflate(R.menu.options_menu, menu);`.
        - Return `true` to display the menu.
    - **Override `onOptionsItemSelected()`:** This method is called when an options menu item is selected.
        - Use a `switch` statement on `item.getItemId()` to identify which menu item was clicked.
        - Perform the desired action for each item (e.g., display a `Toast` message with the item's title).
        - Return `true` to indicate that the event was handled.

## Source Code

*`activity_main.xml` (Layout)*

```
<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:padding="16dp"
    tools:context=".MainActivity">

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

```
        android:text="Status: No option selected"
        android:textSize="22sp"
        android:layout_centerInParent="true"/>

</RelativeLayout>
```

*res/menu/options_menu.xml* **(Menu Resource)**

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_settings"
        android:title="Settings"
        android:icon="@android:drawable/ic_menu_preferences"
        app:showAsAction="never" /> <item
        android:id="@+id/action_search"
        android:title="Search"
        android:icon="@android:drawable/ic_menu_search"
        app:showAsAction="ifRoom" /> <item
        android:id="@+id/action_about"
        android:title="About"
        app:showAsAction="never" />
</menu>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.optionmenuapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private TextView statusTextView;

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

        statusTextView = findViewById(R.id.statusTextView);
    }

    // Called to inflate the options menu
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.options_menu, menu);
        return true; // Return true to display the menu
    }

    // Called when an options menu item is selected
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        String selectedOption = "";
        int itemId = item.getItemId();

        if (itemId == R.id.action_settings) {
            selectedOption = "Settings";
        } else if (itemId == R.id.action_search) {
            selectedOption = "Search";
        } else if (itemId == R.id.action_about) {
            selectedOption = "About";
```

```
        } else {
            return super.onOptionsItemSelected(item); // Handle unhandled items
        }

        Toast.makeText(this, selectedOption + " selected",
Toast.LENGTH_SHORT).show();
        statusTextView.setText("Status: " + selectedOption + " was clicked.");
        return true; // Indicate that the item has been handled
    }
}
```

## Input

User taps the three-dot menu icon (overflow menu) or the search icon in the Action Bar. Then, the user taps on one of the menu items (e.g., "Settings").

## Expected Output

1. Upon tapping the menu icon: The options menu drops down (or displays in the action bar) with "Settings", "Search", and "About" options.
2. Upon tapping "Settings": A `Toast` message "Settings selected" appears, and the `statusTextView` updates to "Status: Settings was clicked.".

# Lab 10: Shared Preferences

## Title

Android Lab 10: Implementing Shared Preferences

## Aim

To demonstrate how to store and retrieve simple key-value pairs of data persistently using Android's `SharedPreferences`.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Add an `EditText` (e.g., `dataEditText`) for the user to input text.
    - Add two `Button` widgets: "Save Data" (e.g., `saveButton`) and "Load Data" (e.g., `loadButton`).
    - Add a `TextView` (e.g., `displayTextView`) to show the loaded data.
3. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get references to the `EditText`, `Buttons`, and `TextView`.
    - **Initialize `SharedPreferences`:** Get an instance of `SharedPreferences` using `getSharedPreferences("MyPrefs", MODE_PRIVATE);`. "MyPrefs" is the name of your preference file, and `MODE_PRIVATE` means only your app can access it.
    - **"Save Data" Button Listener:**
        - Retrieve the text from `dataEditText`.
        - Get a `SharedPreferences.Editor` object: `SharedPreferences.Editor editor = sharedPreferences.edit();`
        - Put the data using `editor.putString("my_key", data);` (where "my_key" is a unique identifier).
        - Apply the changes: `editor.apply();` (asynchronous) or `editor.commit();` (synchronous). `apply()` is generally preferred.
        - Display a `Toast` message confirming save.
    - **"Load Data" Button Listener:**
        - Retrieve the data using `sharedPreferences.getString("my_key", "Default Value");`. The second argument is a default value if the key is not found.
        - Set the retrieved data to `displayTextView`.
        - Display a `Toast` message confirming load.
    - **Initial Load (Optional):** You can also load the data when the activity is created (in `onCreate()`) to display any previously saved data immediately.

## Source Code

*`activity_main.xml` (Layout)*

```
<LinearLayout 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"
    android:padding="16dp"
```

```
    android:gravity="center_horizontal"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Shared Preferences Demo"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_marginBottom="30dp"/>

    <EditText
        android:id="@+id/dataEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Enter data to save"
        android:layout_marginBottom="20dp"/>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginBottom="30dp">

        <Button
            android:id="@+id/saveButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Save Data"
            android:layout_marginEnd="10dp"/>

        <Button
            android:id="@+id/loadButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Load Data"
            android:layout_marginStart="10dp"/>
    </LinearLayout>

    <TextView
        android:id="@+id/displayTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Loaded Data: (None)"
        android:textSize="20sp"
        android:textStyle="italic"/>

</LinearLayout>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.sharedpreferencesapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private EditText dataEditText;
    private Button saveButton, loadButton;
```

```
    private TextView displayTextView;
    private SharedPreferences sharedPreferences;

    // Define a key for your preference
    private static final String PREF_KEY_DATA = "my_saved_data";
    private static final String PREF_NAME = "MyAppData"; // Name of the
preference file

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

        // Get references to views
        dataEditText = findViewById(R.id.dataEditText);
        saveButton = findViewById(R.id.saveButton);
        loadButton = findViewById(R.id.loadButton);
        displayTextView = findViewById(R.id.displayTextView);

        // Initialize SharedPreferences
        // MODE_PRIVATE means only this app can access the preferences
        sharedPreferences = getSharedPreferences(PREF_NAME, MODE_PRIVATE);

        // Set OnClickListener for Save Button
        saveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String dataToSave = dataEditText.getText().toString();
                SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString(PREF_KEY_DATA, dataToSave);
                editor.apply(); // Apply changes asynchronously
                Toast.makeText(MainActivity.this, "Data Saved!",
Toast.LENGTH_SHORT).show();
                dataEditText.setText(""); // Clear input field
            }
        });

        // Set OnClickListener for Load Button
        loadButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Get the string data, provide a default value if key not found
                String loadedData = sharedPreferences.getString(PREF_KEY_DATA,
"No data found");
                displayTextView.setText("Loaded Data: " + loadedData);
                Toast.makeText(MainActivity.this, "Data Loaded!",
Toast.LENGTH_SHORT).show();
            }
        });

        // Optionally, load data when the activity starts
        loadInitialData();
    }

    private void loadInitialData() {
        String loadedData = sharedPreferences.getString(PREF_KEY_DATA,
"(None)");
        displayTextView.setText("Loaded Data: " + loadedData);
    }
}
```

## Input

1. User types "Hello Shared Preferences!" into the `EditText`.
2. User taps "Save Data".

3. User closes and reopens the app (or taps "Load Data").

## Expected Output

1. After tapping "Save Data": A `Toast` message "Data Saved!" appears. The `EditText` clears.
2. Upon app restart or tapping "Load Data": The `displayTextView` updates to "Loaded Data: Hello Shared Preferences!". A `Toast` message "Data Loaded!" appears.

# Lab 11: SQLite Database (Part 1 - Basic Operations)

## Title

Android Lab 11: Implementing SQLite Database (Part 1 - Basic Operations)

## Aim

To perform basic CRUD (Create, Read, Update, Delete) operations on a local SQLite database within an Android application.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Add `EditText` fields for input (e.g., `idEditText`, `nameEditText`, `emailEditText`).
    - Add `Button` widgets for CRUD operations: "Add Data", "View Data", "Update Data", "Delete Data".
    - Add a `TextView` or `ListView` to display messages or retrieved data. For simplicity, we'll use a `TextView` for messages and `Toast` for data.
3. **Create a Database Helper Class (`DatabaseHelper.java`):**
    - Create a new Java/Kotlin class (e.g., `DatabaseHelper`) that extends `SQLiteOpenHelper`.
    - **Constructor:** Define the database name and version.
    - **`onCreate(SQLiteDatabase db)`:** This method is called when the database is created for the first time. Here, execute SQL to create your table (e.g., `CREATE TABLE students (ID INTEGER PRIMARY KEY AUTOINCREMENT, NAME TEXT, EMAIL TEXT);`).
    - **`onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)`:** This method is called when the database needs to be upgraded (e.g., table schema changes). Typically, you drop the old table and recreate it.
    - **Implement CRUD Methods:**
        - `insertData(String name, String email)`: Gets a writable database, creates `ContentValues`, puts data, and calls `db.insert()`.
        - `getAllData()`: Gets a readable database, executes a `SELECT *` query using `db.rawQuery()` or `db.query()`, and returns a `Cursor`.
        - `updateData(String id, String name, String email)`: Gets a writable database, creates `ContentValues`, and calls `db.update()` with a `WHERE` clause.
        - `deleteData(String id)`: Gets a writable database and calls `db.delete()` with a `WHERE` clause.
4. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get references to all UI elements.
    - Create an instance of `DatabaseHelper`.
    - Set `OnClickListener` for each CRUD button.
    - Inside each `onClick` method:
        - **"Add Data":** Get data from `EditText` fields and call `dbHelper.insertData()`. Display a `Toast` message.
        - **"View Data":** Call `dbHelper.getAllData()`. Iterate through the `Cursor` to build a string of all data. Display this string in a `TextView` or `Toast`.

- **"Update Data":** Get ID, name, email from `EditText` fields and call `dbHelper.updateData()`. Display a `Toast`.
- **"Delete Data":** Get ID from `idEditText` and call `dbHelper.deleteData()`. Display a `Toast`.

## Source Code

*activity_main.xml* ***(Layout)***

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp"
        android:gravity="center_horizontal">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="SQLite Database Operations"
            android:textSize="24sp"
            android:textStyle="bold"
            android:layout_marginBottom="20dp"/>

        <EditText
            android:id="@+id/idEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="ID (for Update/Delete)"
            android:inputType="number"
            android:layout_marginBottom="10dp"/>

        <EditText
            android:id="@+id/nameEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Name"
            android:layout_marginBottom="10dp"/>

        <EditText
            android:id="@+id/emailEditText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Email"
            android:inputType="textEmailAddress"
            android:layout_marginBottom="20dp"/>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:gravity="center_horizontal"
            android:layout_marginBottom="10dp">
            <Button
                android:id="@+id/addButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Add Data"
                android:layout_marginEnd="10dp"/>
```

```
            <Button
                android:id="@+id/viewButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="View Data"/>
        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:gravity="center_horizontal"
            android:layout_marginBottom="20dp">
            <Button
                android:id="@+id/updateButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Update Data"
                android:layout_marginEnd="10dp"/>
            <Button
                android:id="@+id/deleteButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Delete Data"/>
        </LinearLayout>

        <TextView
            android:id="@+id/dataDisplayTextView"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Database Output:"
            android:textSize="16sp"
            android:textStyle="italic"
            android:padding="10dp"
            android:background="#f0f0f0"
            android:minHeight="100dp"/>

    </LinearLayout>
</ScrollView>
```

*DatabaseHelper.java* **(Helper Class)**

```
// DatabaseHelper.java
package com.example.sqlitedemoapp; // Replace with your package name

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

public class DatabaseHelper extends SQLiteOpenHelper {

    // Database Name and Version
    private static final String DATABASE_NAME = "Student.db";
    private static final int DATABASE_VERSION = 1;

    // Table Name and Columns
    public static final String TABLE_NAME = "students_table";
    public static final String COL_1 = "ID";
    public static final String COL_2 = "NAME";
    public static final String COL_3 = "EMAIL";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
```

```
    @Override
    public void onCreate(SQLiteDatabase db) {
        // SQL to create the table
        String CREATE_TABLE = "CREATE TABLE " + TABLE_NAME + " (" +
                COL_1 + " INTEGER PRIMARY KEY AUTOINCREMENT," +
                COL_2 + " TEXT," +
                COL_3 + " TEXT" +
                ")";
        db.execSQL(CREATE_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // Drop older table if it exists
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        // Create tables again
        onCreate(db);
    }

    // Method to insert data
    public boolean insertData(String name, String email) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COL_2, name);
        contentValues.put(COL_3, email);
        long result = db.insert(TABLE_NAME, null, contentValues);
        return result != -1; // Returns true if data inserted successfully,
false otherwise
    }

    // Method to get all data
    public Cursor getAllData() {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res = db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
        return res;
    }

    // Method to update data
    public boolean updateData(String id, String name, String email) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COL_1, id);
        contentValues.put(COL_2, name);
        contentValues.put(COL_3, email);
        int result = db.update(TABLE_NAME, contentValues, "ID = ?", new
String[]{id});
        return result > 0; // Returns true if updated successfully
    }

    // Method to delete data
    public int deleteData(String id) {
        SQLiteDatabase db = this.getWritableDatabase();
        int result = db.delete(TABLE_NAME, "ID = ?", new String[]{id});
        return result; // Returns number of rows deleted
    }
}
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.sqlitedemoapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private DatabaseHelper myDb;
    private EditText idEditText, nameEditText, emailEditText;
    private Button addButton, viewButton, updateButton, deleteButton;
    private TextView dataDisplayTextView;

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

        myDb = new DatabaseHelper(this); // Initialize database helper

        // Get references to UI elements
        idEditText = findViewById(R.id.idEditText);
        nameEditText = findViewById(R.id.nameEditText);
        emailEditText = findViewById(R.id.emailEditText);
        addButton = findViewById(R.id.addButton);
        viewButton = findViewById(R.id.viewButton);
        updateButton = findViewById(R.id.updateButton);
        deleteButton = findViewById(R.id.deleteButton);
        dataDisplayTextView = findViewById(R.id.dataDisplayTextView);

        // Set up button listeners
        AddData();
        ViewAll();
        UpdateData();
        DeleteData();
    }

    public void AddData() {
        addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                boolean isInserted =
myDb.insertData(nameEditText.getText().toString(),

emailEditText.getText().toString());
                if (isInserted) {
                    Toast.makeText(MainActivity.this, "Data Inserted",
Toast.LENGTH_SHORT).show();
                    nameEditText.setText("");
                    emailEditText.setText("");
                } else {
                    Toast.makeText(MainActivity.this, "Data Not Inserted",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    public void ViewAll() {
        viewButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Cursor res = myDb.getAllData();
                if (res.getCount() == 0) {
                    // Show message
                    dataDisplayTextView.setText("No Data Found.");
                    Toast.makeText(MainActivity.this, "No Data Found",
Toast.LENGTH_SHORT).show();
                    return;
```

```
                }

                StringBuilder buffer = new StringBuilder();
                while (res.moveToNext()) {
                    buffer.append("ID: ").append(res.getString(0)).append("\n");
                    buffer.append("Name:
").append(res.getString(1)).append("\n");
                    buffer.append("Email:
").append(res.getString(2)).append("\n\n");
                }
                dataDisplayTextView.setText(buffer.toString());
                Toast.makeText(MainActivity.this, "Data Displayed",
Toast.LENGTH_SHORT).show();
                res.close(); // Close the cursor
            }
        });
    }

    public void UpdateData() {
        updateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                boolean isUpdate =
myDb.updateData(idEditText.getText().toString(),

nameEditText.getText().toString(),

emailEditText.getText().toString());
                if (isUpdate) {
                    Toast.makeText(MainActivity.this, "Data Updated",
Toast.LENGTH_SHORT).show();
                } else {
                    Toast.makeText(MainActivity.this, "Data Not Updated",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }

    public void DeleteData() {
        deleteButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int deletedRows =
myDb.deleteData(idEditText.getText().toString());
                if (deletedRows > 0) {
                    Toast.makeText(MainActivity.this, "Data Deleted",
Toast.LENGTH_SHORT).show();
                    idEditText.setText("");
                } else {
                    Toast.makeText(MainActivity.this, "Data Not Deleted",
Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

## Input

### Scenario 1: Add Data

1. Name: `Alice`, Email: `alice@example.com` -> Click "Add Data"
2. Name: `Bob`, Email: `bob@example.com` -> Click "Add Data"

**Scenario 2: View Data**

1. Click "View Data"

**Scenario 3: Update Data (assuming ID 1 is Alice)**

1. ID: `1`, Name: `Alice Wonderland`, Email: `alice.w@example.com` -> Click "Update Data"

**Scenario 4: Delete Data (assuming ID 2 is Bob)**

1. ID: `2` -> Click "Delete Data"

**Scenario 5: View Data again**

1. Click "View Data"

## Expected Output

**Scenario 1: Add Data**

- `Toast`: "Data Inserted" (twice)

**Scenario 2: View Data**

- `dataDisplayTextView`:
- `ID: 1`
- `Name: Alice`
- `Email: alice@example.com`
-
- `ID: 2`
- `Name: Bob`
- `Email: bob@example.com`

- `Toast`: "Data Displayed"

**Scenario 3: Update Data**

- `Toast`: "Data Updated"

**Scenario 4: Delete Data**

- `Toast`: "Data Deleted"

**Scenario 5: View Data again**

- `dataDisplayTextView`:
- `ID: 1`
- `Name: Alice Wonderland`
- `Email: alice.w@example.com`

- `Toast`: "Data Displayed"

# Lab 12: SQLite Database (Part 2 - Data Display and Search)

## Title

Android Lab 12: Implementing SQLite Database (Part 2 - Data Display and Search)

## Aim

To enhance the SQLite database application by displaying retrieved data in a `ListView` and implementing a search functionality to filter the displayed data.

## Procedure

1. **Continue from Lab 11 Project:** Use the project from Lab 11.
2. **Modify Layout (`activity_main.xml`):**
    - Keep the `EditText` fields for adding/updating data.
    - Replace the `dataDisplayTextView` with a `ListView` (e.g., `dataListView`).
    - Add an `EditText` for search input (e.g., `searchEditText`).
    - Add a "Search" `Button` (e.g., `searchButton`).
3. **Modify `DatabaseHelper.java`:**
    - Add a new method `getFilteredData(String query)` that takes a search string and returns a `Cursor` with matching records (e.g., using `LIKE` operator in the `WHERE` clause).
4. **Modify `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get references to the new `ListView`, `searchEditText`, and `searchButton`.
    - **Data Structures:**
        - Maintain an `ArrayList<String>` (e.g., `dataList`) to store the formatted string representation of database records.
        - Use an `ArrayAdapter<String>` (e.g., `adapter`) to link `dataList` to `dataListView`.
    - **Initial Data Load:** When the activity starts, fetch all data from the database using `myDb.getAllData()` and populate `dataList` and `adapter`.
    - **"Add Data" / "Update Data" Logic:** After adding or updating data, re-fetch all data and call `adapter.notifyDataSetChanged()` to refresh the `ListView`.
    - **"Search" Button Listener:**
        - Get the search query from `searchEditText`.
        - Call `myDb.getFilteredData(query)`.
        - Clear the `dataList`.
        - Iterate through the returned `Cursor` and add matching records to `dataList`.
        - Call `adapter.notifyDataSetChanged()`.
        - If the search query is empty, display all data.
    - **Optional: `TextWatcher` for Live Search:** Instead of a button, you can attach a `TextWatcher` to `searchEditText` to perform live filtering as the user types.

## Source Code

*`activity_main.xml` (Layout)*

```
<LinearLayout 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"
    android:padding="16dp"
    android:gravity="center_horizontal"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SQLite Data Display & Search"
        android:textSize="24sp"
        android:textStyle="bold"
        android:layout_marginBottom="20dp"/>

    <EditText
        android:id="@+id/nameEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Name"
        android:layout_marginBottom="10dp"/>

    <EditText
        android:id="@+id/emailEditText"
        android:layout_width="match_content"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/addButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Add Data"
        android:layout_marginBottom="30dp"/>

    <EditText
        android:id="@+id/searchEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Search by Name or Email"
        android:layout_marginBottom="10dp"/>

    <Button
        android:id="@+id/searchButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Search"
        android:layout_marginBottom="20dp"/>

    <ListView
        android:id="@+id/dataListView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="#f0f0f0"
        android:padding="5dp"/>

</LinearLayout>
```

*DatabaseHelper.java* **(Modified Helper Class)**

```
// DatabaseHelper.java (Modified from Lab 11)
package com.example.sqlitedemoapp; // Replace with your package name

import android.content.ContentValues;
```

```
import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DatabaseHelper extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "Student.db";
    private static final int DATABASE_VERSION = 1;

    public static final String TABLE_NAME = "students_table";
    public static final String COL_1 = "ID";
    public static final String COL_2 = "NAME";
    public static final String COL_3 = "EMAIL";

    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String CREATE_TABLE = "CREATE TABLE " + TABLE_NAME + " (" +
                COL_1 + " INTEGER PRIMARY KEY AUTOINCREMENT," +
                COL_2 + " TEXT," +
                COL_3 + " TEXT" +
                ")";
        db.execSQL(CREATE_TABLE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }

    public boolean insertData(String name, String email) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(COL_2, name);
        contentValues.put(COL_3, email);
        long result = db.insert(TABLE_NAME, null, contentValues);
        return result != -1;
    }

    public Cursor getAllData() {
        SQLiteDatabase db = this.getReadableDatabase();
        Cursor res = db.rawQuery("SELECT * FROM " + TABLE_NAME, null);
        return res;
    }

    // New method to get filtered data
    public Cursor getFilteredData(String query) {
        SQLiteDatabase db = this.getReadableDatabase();
        // Use LIKE operator for partial matching, % acts as a wildcard
        String selection = COL_2 + " LIKE ? OR " + COL_3 + " LIKE ?";
        String[] selectionArgs = new String[]{"%" + query + "%", "%" + query +
"%"};
        Cursor res = db.query(TABLE_NAME, null, selection, selectionArgs, null,
null, null);
        return res;
    }
}
```

*MainActivity.java* ***(Logic - Java)***

```
// MainActivity.java (Modified from Lab 11)
package com.example.sqlitedemoapp; // Replace with your package name
```

```
import androidx.appcompat.app.AppCompatActivity;
import android.database.Cursor;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    private DatabaseHelper myDb;
    private EditText nameEditText, emailEditText, searchEditText;
    private Button addButton, searchButton;
    private ListView dataListView;
    private ArrayList<String> dataList;
    private ArrayAdapter<String> adapter;

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

        myDb = new DatabaseHelper(this);

        // Get references to UI elements
        nameEditText = findViewById(R.id.nameEditText);
        emailEditText = findViewById(R.id.emailEditText);
        addButton = findViewById(R.id.addButton);
        searchEditText = findViewById(R.id.searchEditText);
        searchButton = findViewById(R.id.searchButton);
        dataListView = findViewById(R.id.dataListView);

        // Initialize ArrayList and ArrayAdapter for ListView
        dataList = new ArrayList<>();
        adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1,
dataList);
        dataListView.setAdapter(adapter);

        // Load all data initially
        loadAllDataIntoListView();

        // Set up Add Button listener
        addButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                boolean isInserted =
myDb.insertData(nameEditText.getText().toString(),

emailEditText.getText().toString());
                if (isInserted) {
                    Toast.makeText(MainActivity.this, "Data Inserted",
Toast.LENGTH_SHORT).show();
                    nameEditText.setText("");
                    emailEditText.setText("");
                    loadAllDataIntoListView(); // Refresh ListView after adding
                } else {
                    Toast.makeText(MainActivity.this, "Data Not Inserted",
Toast.LENGTH_SHORT).show();
                }
            }
```

```
        });

        // Set up Search Button listener
        searchButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String query = searchEditText.getText().toString().trim();
                if (query.isEmpty()) {
                    loadAllDataIntoListView(); // If search is empty, show all
                    Toast.makeText(MainActivity.this, "Displaying all data",
Toast.LENGTH_SHORT).show();
                } else {
                    loadFilteredDataIntoListView(query);
                    Toast.makeText(MainActivity.this, "Searching for: " + query,
Toast.LENGTH_SHORT).show();
                }
            }
        });

        // Optional: Live search using TextWatcher
        searchEditText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count,
int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int
count) {
                String query = s.toString().trim();
                if (query.isEmpty()) {
                    loadAllDataIntoListView();
                } else {
                    loadFilteredDataIntoListView(query);
                }
            }

            @Override
            public void afterTextChanged(Editable s) {}
        });
    }

    // Helper method to load all data into ListView
    private void loadAllDataIntoListView() {
        dataList.clear(); // Clear existing data
        Cursor res = myDb.getAllData();
        if (res.getCount() == 0) {
            dataList.add("No Data Found.");
        } else {
            while (res.moveToNext()) {
                dataList.add("ID: " + res.getString(0) + ", Name: " +
res.getString(1) + ", Email: " + res.getString(2));
            }
        }
        adapter.notifyDataSetChanged(); // Notify adapter about data change
        res.close();
    }

    // Helper method to load filtered data into ListView
    private void loadFilteredDataIntoListView(String query) {
        dataList.clear();
        Cursor res = myDb.getFilteredData(query);
        if (res.getCount() == 0) {
            dataList.add("No matching data found for '" + query + "'.");
        } else {
            while (res.moveToNext()) {
                dataList.add("ID: " + res.getString(0) + ", Name: " +
res.getString(1) + ", Email: " + res.getString(2));
```

```
            }
        }
        adapter.notifyDataSetChanged();
        res.close();
    }
}
```

## Input

1. **Add Data:**
    - Name: `Alice`, Email: `alice@example.com` -> Click "Add Data"
    - Name: `Bob`, Email: `bob@example.com` -> Click "Add Data"
    - Name: `Charlie`, Email: `charlie@test.com` -> Click "Add Data"
2. **Search:**
    - Type `ali` into Search `EditText` -> Click "Search"
    - Type `test` into Search `EditText` -> Click "Search"
    - Clear Search `EditText` -> Click "Search"

## Expected Output

1. **After adding data:** The `ListView` will automatically update to show:

```
ID: 1, Name: Alice, Email: alice@example.com
ID: 2, Name: Bob, Email: bob@example.com
ID: 3, Name: Charlie, Email: charlie@test.com
```

5. **Search for `ali`:** `ListView` updates to:

```
ID: 1, Name: Alice, Email: alice@example.com
```

7. **Search for `test`:** `ListView` updates to:

```
ID: 3, Name: Charlie, Email: charlie@test.com
```

9. **Clear search and click "Search":** `ListView` reverts to showing all data:

```
 ID: 1, Name: Alice, Email: alice@example.com
 ID: 2, Name: Bob, Email: bob@example.com
 ID: 3, Name: Charlie, Email: charlie@test.com
```

# Lab 13: Simulate Paintbrush Applications

## Title

Android Lab 13: Simulating Paintbrush Applications

## Aim

To create a simple drawing application where users can draw lines on a `Canvas` by touching and dragging their finger on the screen.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Add a custom `View` (e.g., `DrawingView`) that will serve as the drawing area. Set its `layout_width` and `layout_height` to `match_parent`.
    - Optionally, add buttons for clear, change color, etc. For this basic lab, we'll focus on drawing.
3. **Create a Custom `DrawingView` Class:**
    - Create a new Java/Kotlin class (e.g., `DrawingView`) that extends `View`.
    - **Constructor:** Initialize `Paint` and `Path` objects. `Paint` defines the drawing style (color, stroke width), and `Path` stores the sequence of lines drawn.
    - **Override `onDraw(Canvas canvas)`:** This method is called to draw the content.
        - Draw the `Path` onto the `Canvas` using the `Paint` object: `canvas.drawPath(path, paint);`.
    - **Override `onTouchEvent(MotionEvent event)`:** This method handles touch input.
        - Use a `switch` statement on `event.getAction()` to differentiate between `ACTION_DOWN` (finger touches screen), `ACTION_MOVE` (finger drags), and `ACTION_UP` (finger lifts).
        - **`ACTION_DOWN`:** Move the `Path` to the initial touch coordinates: `path.moveTo(x, y);`.
        - **`ACTION_MOVE`:** Add a line segment to the `Path` from the previous point to the current point: `path.lineTo(x, y);`.
        - **`ACTION_UP`:** (Optional) Perform any cleanup or finalization.
        - After modifying the `Path` in `ACTION_DOWN` or `ACTION_MOVE`, call `invalidate();` to force the `View` to redraw itself, which in turn calls `onDraw()`.
        - Return `true` from `onTouchEvent` to indicate that you've consumed the touch event.

## Source Code

*`activity_main.xml` (Layout)*

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.example.paintbrushapp.DrawingView
```

```
        android:id="@+id/drawingView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FFFFFF"/> <Button
        android:id="@+id/clearButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clear Canvas"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="20dp"/>

</RelativeLayout>
```

*DrawingView.java* **(Custom View Class)**

```
// DrawingView.java
package com.example.paintbrushapp; // Replace with your package name

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;

public class DrawingView extends View {

    private Path drawPath; // The path that will be drawn
    private Paint drawPaint; // The paint style for the path
    private Paint canvasPaint; // Paint for the canvas background
    private Canvas drawCanvas; // The canvas where drawing occurs
    private android.graphics.Bitmap canvasBitmap; // Bitmap to hold the canvas
content

    public DrawingView(Context context, AttributeSet attrs) {
        super(context, attrs);
        setupDrawing();
    }

    private void setupDrawing() {
        drawPath = new Path();
        drawPaint = new Paint();

        // Set initial paint properties
        drawPaint.setColor(Color.BLACK); // Default drawing color
        drawPaint.setAntiAlias(true); // Smooth edges
        drawPaint.setStrokeWidth(10); // Stroke width
        drawPaint.setStyle(Paint.Style.STROKE); // Only stroke, no fill
        drawPaint.setStrokeJoin(Paint.Join.ROUND); // Round corners for lines
        drawPaint.setStrokeCap(Paint.Cap.ROUND); // Round ends of lines

        canvasPaint = new Paint(Paint.DITHER_FLAG);
    }

    // Called when the view is assigned a size
    @Override
    protected void onSizeChanged(int w, int h, int oldw, int oldh) {
        super.onSizeChanged(w, h, oldw, oldh);
        // Create a bitmap and canvas for drawing
        canvasBitmap = android.graphics.Bitmap.createBitmap(w, h,
android.graphics.Bitmap.Config.ARGB_8888);
        drawCanvas = new Canvas(canvasBitmap);
    }
```

```
    // Called to draw the view
    @Override
    protected void onDraw(Canvas canvas) {
        // Draw the cached bitmap first
        canvas.drawBitmap(canvasBitmap, 0, 0, canvasPaint);
        // Draw the current path being drawn
        canvas.drawPath(drawPath, drawPaint);
    }

    // Handle touch events
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        float touchX = event.getX();
        float touchY = event.getY();

        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                drawPath.moveTo(touchX, touchY); // Start a new path segment
                break;
            case MotionEvent.ACTION_MOVE:
                drawPath.lineTo(touchX, touchY); // Draw line to current point
                break;
            case MotionEvent.ACTION_UP:
                // When finger is lifted, draw the path onto the bitmap
                drawCanvas.drawPath(drawPath, drawPaint);
                drawPath.reset(); // Reset the path for the next stroke
                break;
            default:
                return false;
        }
        invalidate(); // Request a redraw of the view
        return true; // Indicate that the event was handled
    }

    // Method to clear the drawing
    public void clearDrawing() {
        drawCanvas.drawColor(Color.WHITE); // Fill canvas with white
        invalidate(); // Redraw the view
    }
}
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.paintbrushapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private DrawingView drawingView;
    private Button clearButton;

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

        drawingView = findViewById(R.id.drawingView);
        clearButton = findViewById(R.id.clearButton);

        clearButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                drawingView.clearDrawing(); // Call the clear method on the
custom view
            }
        });
    }
}
```

## Input

User touches the screen and drags their finger across the `DrawingView`.

## Expected Output

As the user drags their finger, a black line (or whatever color/style is set in `drawPaint`) is drawn on the screen, following the path of their finger. Tapping the "Clear Canvas" button will erase all drawings.

# Lab 14: Draw an Object

## Title

Android Lab 14: Drawing a Specific Object on Canvas

## Aim

To programmatically draw a specific geometric object (e.g., a circle, rectangle, or custom shape) on an Android `Canvas` using `Paint` and `Path` objects.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Design the Layout (`activity_main.xml`):**
    - Add a custom `View` (e.g., `DrawingObjectView`) that will serve as the drawing area. Set its `layout_width` and `layout_height` to `match_parent`.
3. **Create a Custom `DrawingObjectView` Class:**
    - Create a new Java/Kotlin class (e.g., `DrawingObjectView`) that extends `View`.
    - **Constructor:** Initialize `Paint` objects for different drawing styles (e.g., one for fill, one for stroke).
    - **Override `onDraw(Canvas canvas)`:** This is where the drawing logic resides.
        - **Clear Canvas:** Optionally, fill the canvas with a background color: `canvas.drawColor(Color.WHITE);`.
        - **Draw a Circle:**
            - Set `paint.setColor()`, `paint.setStyle(Paint.Style.FILL)` or `Paint.Style.STROKE`.
            - Use `canvas.drawCircle(centerX, centerY, radius, paint);`.
        - **Draw a Rectangle:**
            - Set `paint.setColor()`, `paint.setStyle()`.
            - Use `canvas.drawRect(left, top, right, bottom, paint);`.
        - **Draw a Triangle (using Path):**
            - Create a `Path` object.
            - Use `path.moveTo(x1, y1);`, `path.lineTo(x2, y2);`, `path.lineTo(x3, y3);`, `path.close();`.
            - Set `paint.setColor()`, `paint.setStyle()`.
            - Use `canvas.drawPath(path, paint);`.
        - You can draw multiple objects with different `Paint` settings.

## Source Code

*`activity_main.xml` (Layout)*

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.example.drawobjectapp.DrawingObjectView
        android:id="@+id/drawingObjectView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
```

```
        android:background="#F0F0F0"/> </RelativeLayout>
```

*DrawingObjectView.java* **(Custom View Class)**

```
// DrawingObjectView.java
package com.example.drawobjectapp; // Replace with your package name

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.util.AttributeSet;
import android.view.View;

public class DrawingObjectView extends View {

    private Paint circlePaint;
    private Paint rectPaint;
    private Paint trianglePaint;
    private Path trianglePath;

    public DrawingObjectView(Context context, AttributeSet attrs) {
        super(context, attrs);
        setupPaints();
    }

    private void setupPaints() {
        // Paint for the circle
        circlePaint = new Paint();
        circlePaint.setColor(Color.BLUE);
        circlePaint.setStyle(Paint.Style.FILL); // Fill the circle
        circlePaint.setAntiAlias(true);

        // Paint for the rectangle
        rectPaint = new Paint();
        rectPaint.setColor(Color.RED);
        rectPaint.setStyle(Paint.Style.STROKE); // Only draw the outline
        rectPaint.setStrokeWidth(8);
        rectPaint.setAntiAlias(true);

        // Paint for the triangle
        trianglePaint = new Paint();
        trianglePaint.setColor(Color.GREEN);
        trianglePaint.setStyle(Paint.Style.FILL_AND_STROKE); // Fill and outline
        trianglePaint.setStrokeWidth(5);
        trianglePaint.setAntiAlias(true);

        // Path for the triangle
        trianglePath = new Path();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        // Get the width and height of the view
        int width = getWidth();
        int height = getHeight();

        // 1. Draw a Circle
        // Center of the view
        float centerX = width / 2f;
        float centerY = height / 4f;
        float radius = 100f; // pixels
        canvas.drawCircle(centerX, centerY, radius, circlePaint);
```

```
        // 2. Draw a Rectangle
        float rectLeft = centerX - 150;
        float rectTop = centerY + 100;
        float rectRight = centerX + 150;
        float rectBottom = centerY + 250;
        canvas.drawRect(rectLeft, rectTop, rectRight, rectBottom, rectPaint);

        // 3. Draw a Triangle (using Path)
        // Define triangle points
        float triX1 = centerX;
        float triY1 = centerY + 350; // Top point
        float triX2 = centerX - 120;
        float triY2 = centerY + 550; // Bottom-left point
        float triX3 = centerX + 120;
        float triY3 = centerY + 550; // Bottom-right point

        trianglePath.reset(); // Clear any previous path data
        trianglePath.moveTo(triX1, triY1);
        trianglePath.lineTo(triX2, triY2);
        trianglePath.lineTo(triX3, triY3);
        trianglePath.close(); // Connects the last point to the first point

        canvas.drawPath(trianglePath, trianglePaint);
    }
}
```

*MainActivity.java* ***(Logic - Java)***

```
// MainActivity.java
package com.example.drawobjectapp; // Replace with your package name

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);
        // No specific logic needed in MainActivity for this lab,
        // as drawing is handled entirely within DrawingObjectView.
    }
}
```

## Input

The application starts.

## Expected Output

The `DrawingObjectView` will display:

- A blue filled circle in the upper part of the screen.
- A red outlined rectangle below the circle.
- A green filled triangle with a green outline below the rectangle. All objects will be centered horizontally within the view.

# Lab 15: Implement Web View

## Title

Android Lab 15: Implementing WebView

## Aim

To integrate a `WebView` into an Android application to display web content (either a URL or a local HTML string) directly within the app.

## Procedure

1. **Create a New Android Project:** Follow steps similar to Lab 1.
2. **Add Internet Permission:**
    - Open `AndroidManifest.xml`.
    - Add the internet permission *outside* the `<application>` tag, but inside the `<manifest>` tag:
    - `<uses-permission android:name="android.permission.INTERNET" />`
3. **Design the Layout (`activity_main.xml`):**
    - Add a `WebView` widget. Set its `layout_width` and `layout_height` to `match_parent`. Assign an ID (e.g., `myWebView`).
    - Optionally, add an `EditText` for URL input and a "Load URL" button, or just hardcode a URL. For simplicity, we'll hardcode.
4. **Implement Logic in `MainActivity.java`/`MainActivity.kt`:**
    - In `onCreate()`, get a reference to the `WebView` using `findViewById()`.
    - **Enable JavaScript (if needed):** `myWebView.getSettings().setJavaScriptEnabled(true);`
    - **Set a `WebViewClient`:** This prevents the URL from opening in an external browser and keeps it within your app.
    - `myWebView.setWebViewClient(new WebViewClient());`
    - **Load Content:**
        - **Load a URL:** `myWebView.loadUrl("https://www.google.com");`
        - **Load HTML string:** `myWebView.loadData("<html><body><h1>Hello WebView!</h1><p>This is local HTML content.</p></body></html>", "text/html", "UTF-8");`
    - **Handle Back Button (Optional but Recommended):** Override `onBackPressed()` to allow the `WebView` to navigate back through its history if possible, rather than closing the activity.

```
@Override
public void onBackPressed() {
    if (myWebView.canGoBack()) {
        myWebView.goBack();
    } else {
        super.onBackPressed();
    }
}
```

## Source Code

*AndroidManifest.xml* **(Add Permission)**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.WebViewApp"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

*activity_main.xml* **(Layout)**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <WebView
        android:id="@+id/myWebView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</RelativeLayout>
```

*MainActivity.java* **(Logic - Java)**

```
// MainActivity.java
package com.example.webviewapp; // Replace with your package name

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class MainActivity extends AppCompatActivity {

    private WebView myWebView;

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

        myWebView = findViewById(R.id.myWebView);

        // Enable JavaScript (if the web content uses JavaScript)
```

```
        WebSettings webSettings = myWebView.getSettings();
        webSettings.setJavaScriptEnabled(true);

        // Set a WebViewClient to keep the URL loading within the app
        myWebView.setWebViewClient(new WebViewClient());

        // Load a URL
        myWebView.loadUrl("https://www.google.com");

        // Alternatively, load local HTML content:
        // String customHtml = "<html><body><h1>Hello from WebView!</h1>" +
        //                     "<p>This is some local HTML content displayed in
the app.</p>" +
        //                     "<a href='https://developer.android.com/'>Android
Dev Docs</a>" +
        //                     "</body></html>";
        // myWebView.loadData(customHtml, "text/html", "UTF-8");
    }

    // Override onBackPressed to allow WebView to navigate back in its history
    @Override
    public void onBackPressed() {
        if (myWebView.canGoBack()) {
            myWebView.goBack(); // Go back in WebView history
        } else {
            super.onBackPressed(); // Otherwise, let the system handle back
press (exit app)
        }
    }
}
```

## Input

The application starts.

## Expected Output

The `WebView` component fills the screen and displays the content of `https://www.google.com`. If you were to load local HTML, that content would be displayed instead. If you navigate to another page within the `WebView` and then press the device's back button, the `WebView` will navigate back to the previous page.