Menus

5.3 Menus

Android menus are excellent user interface to the end users that makes the user interface more interactive. Menus are used to provide the additional option to the user which is generally not visible in the user interface. It suggests a method to depict the functionality of the application without losing the valued space of the screen.

5.3.1 Types of Menus

Mainly, Menus can be categorised as Popup Menu, Option Menu and Context Menu. Their working is defined in the following sections.

5.3.1.1 Popup Menus

A Popup Menu is used to display a list of elements in a vertical list which is fixed to the view that calls it. It is a good idea to provide multiple functions related to a content as a list hidden behind the click.

To understand the implementation of Popup Menu, first create activity_main.xml file having a Button view with below mentioned code.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/LinearLayout1"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<Button
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Popup Menu"/>

</LinearLayout>
```

activity_main.xml

Now, create the popup_menu.xml XML file in the “menu” folder of “src” folder.

```
<?xml version="1.0"encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
        android:id="@+id/save"
        android:title="Save"/>
<item
        android:id="@+id/edit"
        android:title="Edit"/>
<item
        android:id="@+id/delete"
        android:title="Delete"/>
</menu>
```

popup_menu.xml

Write the following code in MainActivity.java file to show the popup menu to on the click of a button.

MainActivity.java

```
package com.example.testapplication;
 import androidx.appcompat.app.AppCompatActivity;
 import android.os.Bundle;
 import android.view.MenuItem;
 import android.view.View;
 import android.widget.Button;
 import android.widget.PopupMenu;
 import android.widget.Toast;
public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
 Button b1;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 b1 = findViewById(R.id.b1);
 b1.setOnClickListener(this);
 }
 @Override
 public void onClick(View v) {
 PopupMenu popup = new PopupMenu(this, b1);
 popup.getMenuInflater().inflate(R.menu.popup_menu, popup.getMenu());
popup.setOnMenuItemClickListener(new
 PopupMenu.OnMenuItemClickListener() {
 @Override
 public boolean onMenuItemClick(MenuItem item) {
Toast.makeText(MainActivity.this, item.getTitle(),
Toast.LENGTH_LONG).show();
 return true;
 }
 });
 popup.show();
 }
 }
```

Output of the above-mentioned code is shown in Figure 5.4.

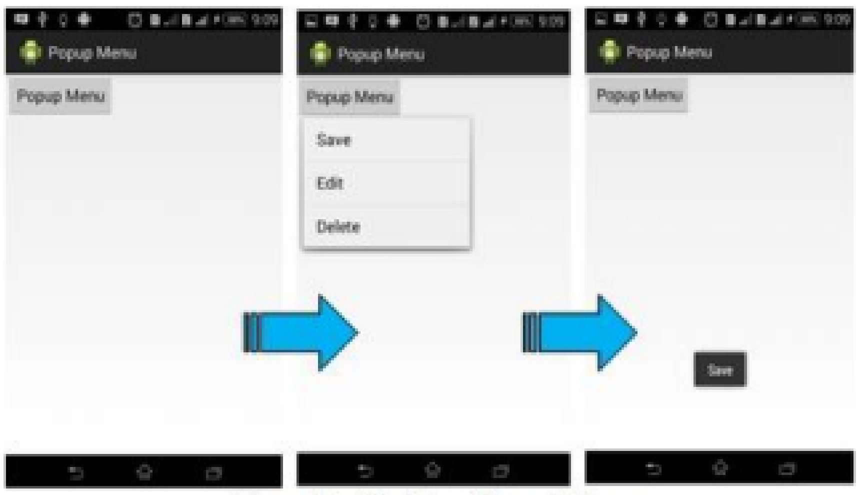

Figure 5.4: Working of Popup Menu

Explanation of the related code:

- **PopupMenu popup = new PopupMenu(MainActivity.this, b1)**: Creates a new PopupMenu instance, anchored to the button b1.
- **popup.getMenuInflater().inflate(R.menu.popup_menu, popup.getMenu())**: Inflates the menu resource (popup_menu.xml) into the popup menu. This means the items defined in popup_menu.xml will be added to the popup menu.
- **popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() { ... })**: Sets a listener for menu item clicks within the popup menu.
- **public boolean onMenuItemClick(MenuItem item)**: Overrides the onMenuItemClick method to define what happens when a menu item is clicked.
- **Toast.makeText(MainActivity.this, item.getTitle(), Toast.LENGTH_LONG).show()**: Displays a toast message showing the title of the clicked menu item.
- **return true**: Indicates that the menu item click was handled.

5.3.1.2 Option Menu

Option menu is one of the simplest and oldest menus in android where options like search, settings are displayed in the user interface. The options menu associated with an Activity gets displayed when the user hits the Menu button on the Android device when the associated Activity is the active (i.e. showing) Activity.

To show the option menu in the activity, change the theme of the application (or activity) in Manifest.xml to a theme which shows option menu (for exp - android:theme="@style/Theme.AppCompat.Light").

Add the following code in activity_main.xml file to add a TextView in the main Activity.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/LinearLayout1"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Select the Option Menu from the Action Bar."/>

</LinearLayout>
```

activity_main.xml

Now create option_menu.xml file in the "menu" folder under "res" folder. It has three menu items in it.

```
<?xml version="1.0"encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
android:id="@+id/save"
android:title="Save"/>
<item
android:id="@+id/edit"
android:title="Edit"/>
<item
android:id="@+id/delete"
android:title="Delete"/>
</menu>
```

option_menu.xml

Now use the following code to implement the Option menu in MainAcitvity.java file.

```
package com.example.testapplication;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity {
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 }
 @Override
 public boolean onCreateOptionsMenu(Menu menu) {
 getMenuInflater().inflate(R.menu.option_menu, menu);  return
super.onCreateOptionsMenu(menu);
 }
 @Override
 public boolean onOptionsItemSelected(MenuItem item) {
```

```
 int id = item.getItemId();
 if(id==R.id.edit)
 Toast.makeText(MainActivity.this, "Edit",
Toast.LENGTH_LONG).show();
 if(id==R.id.save)
 Toast.makeText(MainActivity.this, "Save",
Toast.LENGTH_LONG).show();
 if(id==R.id.delete)
 Toast.makeText(MainActivity.this, "Delete",
Toast.LENGTH_LONG).show();
 return super.onOptionsItemSelected(item);
 }
}
```

MainActivity.java

Output of the above-mentioned code is shown in Figure 5.5.

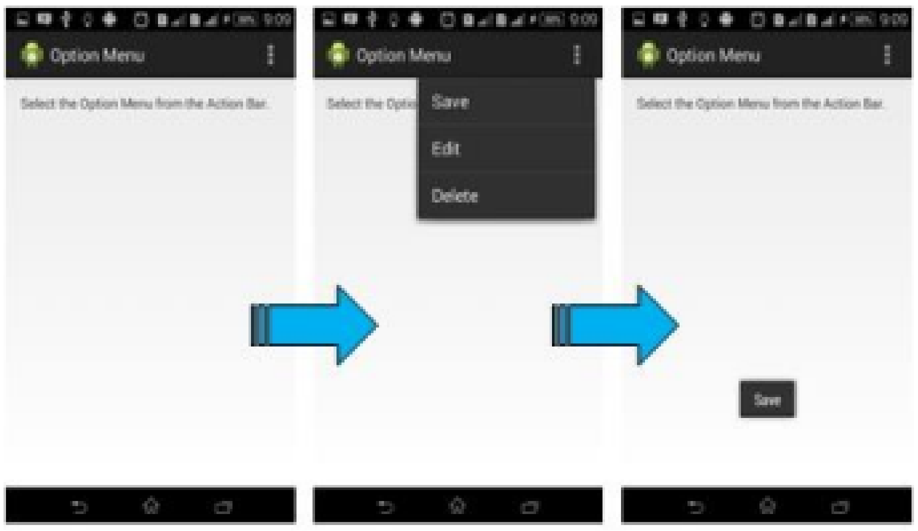

Figure 5.5: Working of Option Menu

Explanation of the related code:

onCreateOptionsMenu Method:

- **onCreateOptionsMenu(Menu menu):** This method is called to initialize the contents of the activity's options menu.
- **getMenuInflater().inflate(R.menu.option_menu, menu):** Uses the MenuInflater to inflate the menu resource (option_menu.xml) into the menu object. This means the items defined in option_menu.xml will be added to the options menu.
- **return super.onCreateOptionsMenu(menu):** Calls the superclass method to handle the menu creation.

onOptionsItemSelected Method:

- **onOptionsItemSelected(MenuItem item):** This method is called when an item in the options menu is selected.
- **int id = item.getItemId():** Retrieves the ID of the selected menu item.
- **if(id==R.id.edit), if(id==R.id.save), if(id==R.id.delete):** Checks which menu item was selected and displays a corresponding toast message.
- **Toast.makeText(MainActivity.this, "Edit/Save/Delete", Toast.LENGTH_LONG).show():**

Displays a toast message indicating the selected action.

- **return super.onOptionsItemSelected(item):** Calls the superclass method to handle the menu item selection.

5.3.1.3 Context Menu

Context menus are the types of menus that are attached to View components (widgets) in an activity. Context menus appears when the end user long presses any UI component on the screen. Because the menu hovers over the screen, it is also known as Float menu. Note that the menu items in context menu have text but not icons. Another characteristic that separates context menus from options menus is the action bars. Option menu resides into the action bar but context menu floats over the screen.

```
-----------------------------------------------------------------------------------------------------------------------
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
 xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <Button
 android:id="@+id/button1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Menu"
 android:layout_marginTop="100dp"
 android:layout_gravity="center"/>
</LinearLayout>
----------------------------------------------------------------------------
```

activity_main.xml

Now create context_menu.xml file in the “menu” folder under “res” folder. It has three menu items in it.

```
<?xml version="1.0"encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item
android:id="@+id/save"
android:title="Save"/>
<item
android:id="@+id/edit"
android:title="Edit"/>
<item
android:id="@+id/delete"
android:title="Delete"/>
</menu>
```

context_menu.xml

Use the following code to implement the Context Menu in MainActivity.java file.

```
----------------------------------------------------------------------------
package com.example.testapplication;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {
 Button b;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 b = findViewById(R.id.button1);
 registerForContextMenu(b);
 }
 @Override
 public void onCreateContextMenu(ContextMenu menu, View v,
ContextMenu.ContextMenuInfo menuInfo){
 menu.setHeaderTitle("Choose a color");
 getMenuInflater().inflate(R.menu.context_menu, menu);  }
 @Override
 public boolean onContextItemSelected(MenuItem item) {
 int id = item.getItemId();
 if(id==R.id.edit)
 Toast.makeText(MainActivity.this, "Edit",
Toast.LENGTH_LONG).show();
 if(id==R.id.save)
 Toast.makeText(MainActivity.this, "Save",
Toast.LENGTH_LONG).show();
 if(id==R.id.delete)
 Toast.makeText(MainActivity.this, "Delete",
Toast.LENGTH_LONG).show();
 return super.onContextItemSelected(item);
 }
}
----------------------------------------------------------------------------
```

MainActivity.java

When you run the above-mentioned code, the output of the code will be as shown in Figure 5.6.

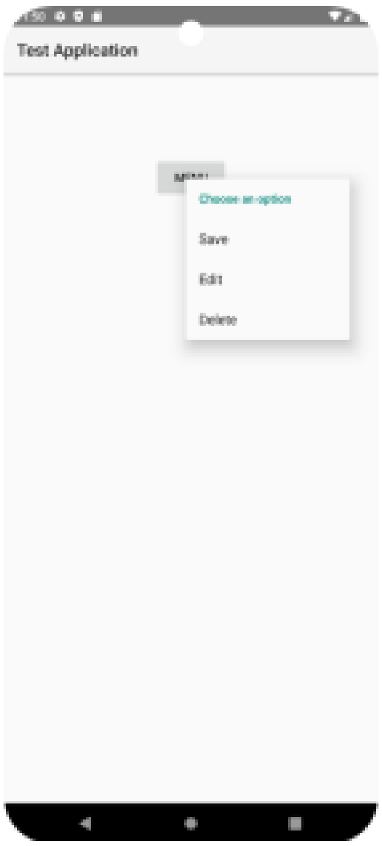

Figure 5.6. Context Menu will display on long-press of the button

Explanation of the related code:

- **registerForContextMenu(b)**: Registers the button b for a context menu. This means that a context menu will be shown when the user performs a long click on the button.

onCreateContextMenu Method:

- **public void onCreateContextMenu(ContextMenu menu, View v, ContextMenu.ContextMenuInfo menuInfo)**: Called when the context menu is being built.
- **menu.setHeaderTitle("Choose an option")**: Sets the header title of the context menu to "Choose an option".
- **getMenuInflater().inflate(R.menu.context_menu, menu)**: Uses the MenuInflater to inflate the menu resource (context_menu.xml) into the menu object. This means the items defined in context_menu.xml will be added to the context menu.

onContextItemSelected Method:

- **public boolean onContextItemSelected(MenuItem item)**: Called when a context menu item is selected.
- **int id = item.getItemId()**: Retrieves the ID of the selected menu item.
- **if(id == R.id.menu_red)**, **if(id == R.id.menu_green)**, **if(id == R.id.menu_blue)**, **if(id == R.id.menu_yellow)**: Checks which menu item was selected based on its ID (edit, save or delete).
- **Toast.makeText(MainActivity.this, "Edit/Save/Delete", Toast.LENGTH_LONG).show()**: Displays a toast message indicating the selected option.

- **return super.onContextItemSelected(item)**: Calls the superclass method to handle the menu item selection.