

Experiment No. 7

Aim: To create GUI base application in Android.

Requirements: Compatible version of Android Studio, Gradle and Java.

Theory:

User Interface

Your app's user interface is everything that the user can see and interact with. Android provides a variety of pre-built UI components such as structured layout objects and UI controls that allow you to build the graphical user interface for your app. Android also provides other UI modules for special interfaces such as dialogs, notifications, and menus.

Layouts

A layout defines the structure for a user interface in your app, such as in an activity. All elements in the layout are built using a hierarchy of View and ViewGroup objects. A View usually draws something the user can see and interact with. Whereas a ViewGroup is an invisible container that defines the layout structure for View and other ViewGroup objects, as shown in figure 1.

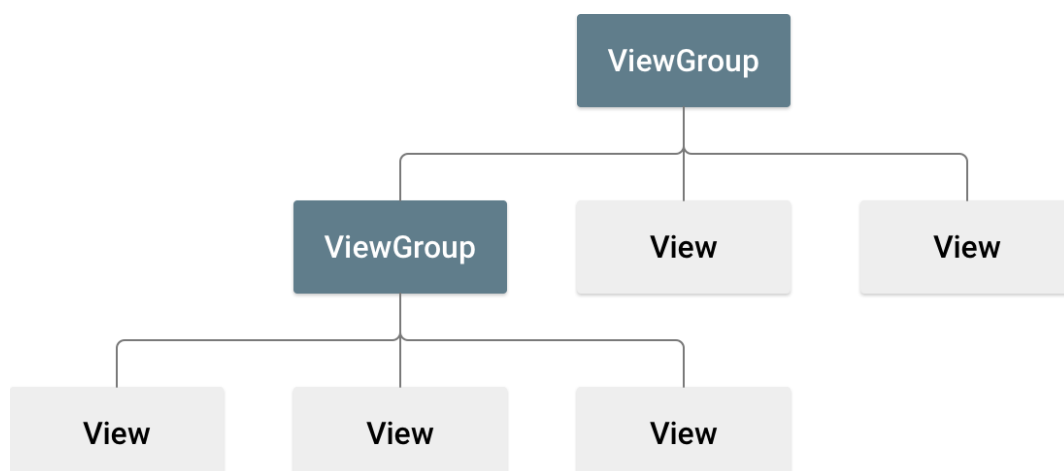


Figure 1. Illustration of a view hierarchy, which defines a UI layout

The View objects are usually called "widgets" and can be one of many subclasses, such as Button or TextView. The ViewGroup objects are usually called "layouts" can be one of many types that provide a different layout structure, such as LinearLayout or ConstraintLayout.

You can declare a layout in two ways:

- **Declare UI elements in XML.** Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.

You can also use Android Studio's Layout Editor to build your XML layout using a drag-and-drop interface.

- **Instantiate layout elements at runtime.** Your app can create View and ViewGroup objects (and manipulate their properties) programmatically.

Declaring your UI in XML allows you to separate the presentation of your app from the code that controls its behavior. Using XML files also makes it easy to provide different layouts for different screen sizes and orientations (discussed further in Supporting Different Screen Sizes).

The Android framework gives you the flexibility to use either or both of these methods to build your app's UI. For example, you can declare your app's default layouts in XML, and then modify the layout at runtime.

Code:

```
package com.trendyol.uicomponents
import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private val buttonRatingBar by lazy { findViewById<Button>(R.id.button_rating_bar) }
    private val buttonDialogs by lazy { findViewById<Button>(R.id.button_dialogs) }
```

```

    private val buttonImageslider by lazy { findViewById<Button>(R.id.button_imageslider)
}
    private val buttonPhoneNumber by lazy {
findViewById<Button>(R.id.button_phone_number) }
    private val buttonToolbar by lazy { findViewById<Button>(R.id.button_toolbar) }
    private val buttonSuggestionInputView by lazy {
findViewById<Button>(R.id.button_suggestion_input_view) }
    private val buttonCardInput by lazy { findViewById<Button>(R.id.button_card_input) }
    private val buttonQuantityPicker by lazy {
findViewById<Button>(R.id.button_quantity_picker) }
    private val buttonTimelineView by lazy {
findViewById<Button>(R.id.button_timeline_view) }
    private val buttonFitOptionMessageView by lazy {
findViewById<Button>(R.id.button_fit_option_message_view) }

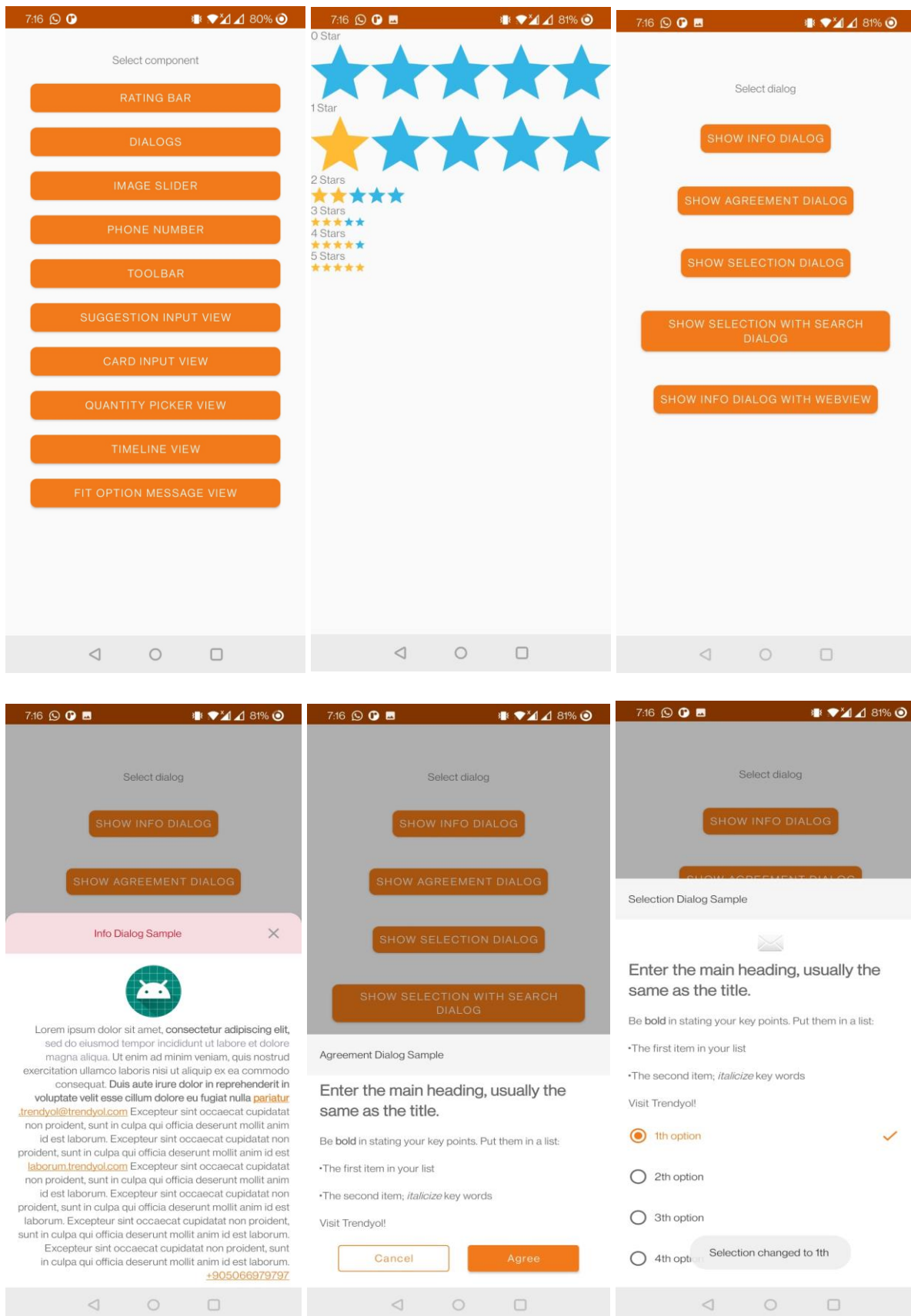
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

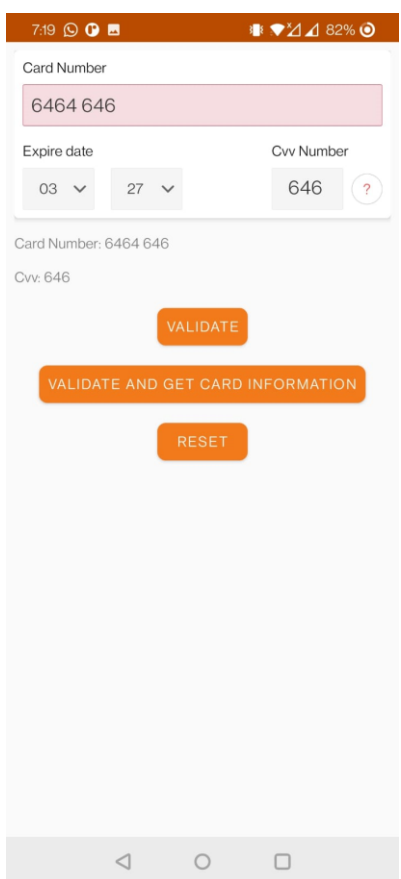
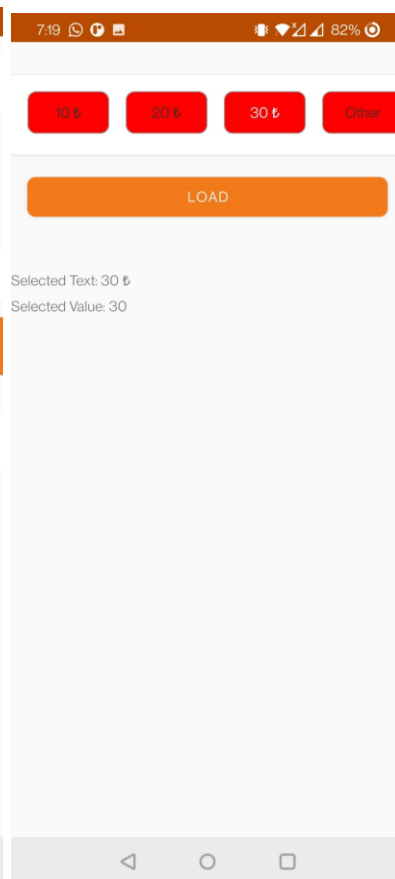
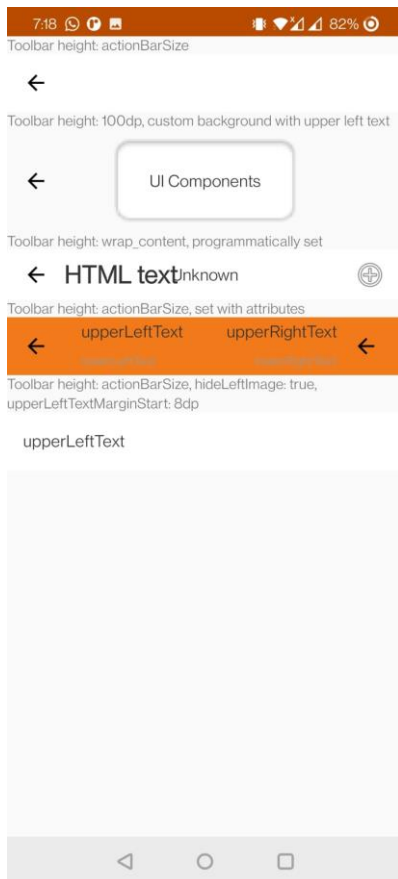
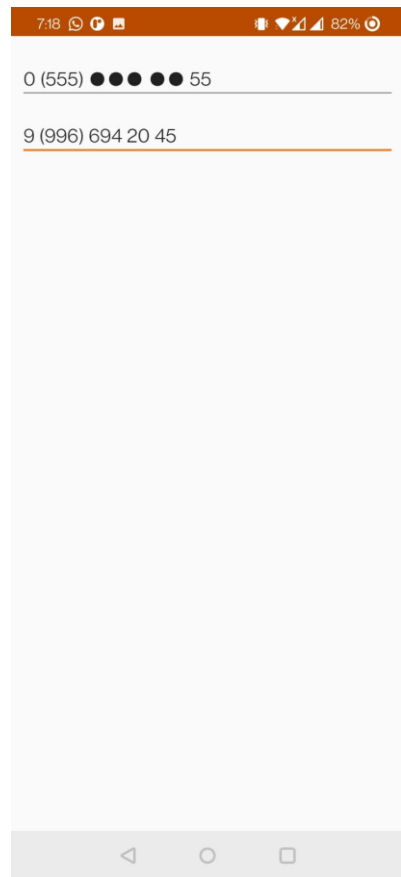
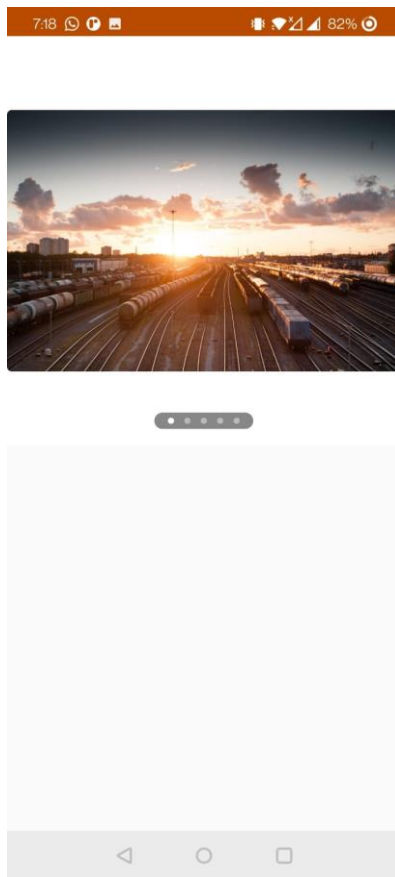
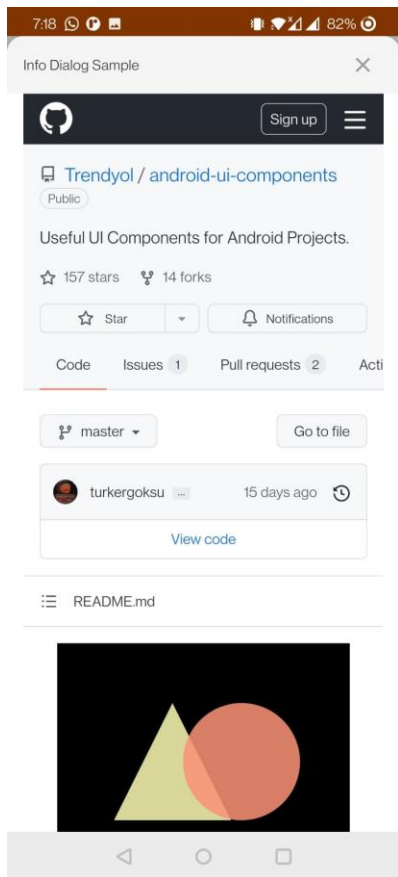
        buttonRatingBar.setOnClickListener {
            startActivity(Intent(this, RatingBarActivity::class.java))
        }
        buttonDialogs.setOnClickListener {
            startActivity(Intent(this, DialogsActivity::class.java))
        }
        buttonImageslider.setOnClickListener {
            startActivity(Intent(this, ImageSliderActivity::class.java))
        }
        buttonPhoneNumber.setOnClickListener {
            startActivity(Intent(this, PhoneNumberActivity::class.java))
        }
        buttonToolbar.setOnClickListener {
            startActivity(Intent(this, ToolbarActivity::class.java))
        }
        buttonSuggestionInputView.setOnClickListener {
            startActivity(Intent(this, SuggestionInputViewActivity::class.java))
        }
        buttonCardInput.setOnClickListener {
            startActivity(Intent(this, CardInputViewActivity::class.java))
        }
        buttonQuantityPicker.setOnClickListener {
            startActivity(Intent(this, QuantityPickerViewActivity::class.java))
        }
        buttonTimelineView.setOnClickListener {
            startActivity(Intent(this, TimelineViewActivity::class.java))
        }
        buttonFitOptionMessageView.setOnClickListener {
            startActivity(Intent(this, FitOptionMessageViewActivity::class.java))
        }
    }
}

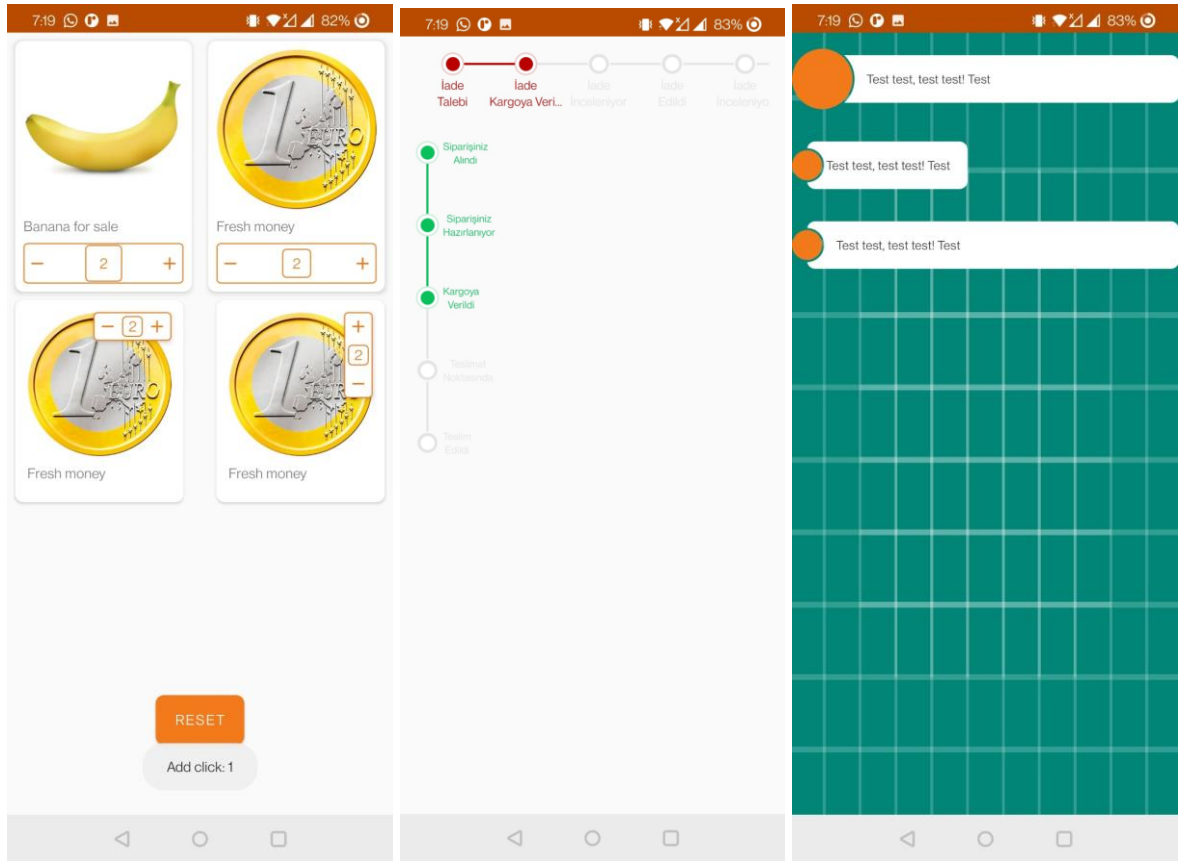
```

}

Output:







Conclusion: We have successfully implemented GUI app in android.