**Implicit Intent:**

val btn=findViewById<Button>(R.id.*button*)  
 btn.setOnClickListener()**{** *intent*=Intent(Intent.*ACTION\_VIEW*)  
 *intent*.setData(Uri.parse("https://www.google.com")  
 )  
 startActivity(*intent*)  
  
 }

**Explicit Intent:**

val btn=findViewById<Button>(R.id.*button*)  
 btn.setOnClickListener()**{** val intent=Intent(this,MainActivity2::class.*java*)  
 startActivity(intent)  
 **}**

**Toast Message:**

val btn=findViewById<Button>(R.id.*button*)  
 btn.setOnClickListener(){  
 Toast.makeText(this,"Niru is an idiot",Toast.*LENGTH\_SHORT*).show()  
 }

**Menu:**

**menu.xml**

<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android">  
 <item  
 android:title="opt 1"/>  
 <item  
 android:title="opt 2"/>  
 <item  
 android:title="opt 3"/>  
  
</menu>

**Option menu:**

override fun onCreateOptionsMenu(menu: Menu?): Boolean{  
 *menuInflater*.inflate(R.menu.*menu*,menu)  
 return true  
}

override fun onOptionsItemSelected(item:MenuItem):Boolean {  
  
 return when (item.*itemId*) {  
 R.id.*o1* -> {  
 Toast.makeText(this, "Edit", Toast.*LENGTH\_SHORT*).show()  
 true  
 }  
  
 R.id.*o2* -> {  
 Toast.makeText(this, "Delete", Toast.*LENGTH\_SHORT*).show()  
 true  
 }  
  
 R.id.*o3* -> {  
 Toast.makeText(this, "Add", Toast.*LENGTH\_SHORT*).show()  
 true  
 }  
  
 else -> false  
 }  
}

**POPUP MENU**

val button = findViewById<Button>(R.id.button)

button.setOnClickListener {

val popupMenu = PopupMenu(this, button)

popupMenu.menuInflater.inflate(R.menu.*popup\_menu*, popupMenu.menu)

popupMenu.setOnMenuItemClickListener **{** item **->**

when (item.itemId) {

R.id.option\_edit -> {

Toast.makeText(this, "Edit Clicked", Toast.LENGTH\_SHORT).show()

true

}

R.id.option\_delete -> {

Toast.makeText(this, "Delete Clicked", Toast.LENGTH\_SHORT).show()

true

}

else -> false

}

**}**

popupMenu.show()

}

**CONTEXT MENU**

private lateinit var textView: TextView

textView = findViewById(R.id.text1)

registerForContextMenu(textView)

override fun onCreateContextMenu(menu: ContextMenu, v: View, menuInfo: ContextMenu.ContextMenuInfo?) {

super.onCreateContextMenu(menu, v, menuInfo)

menu.setHeaderTitle("Choose a color")

menu.add(0, v.id, 0, "Yellow")

menu.add(0, v.id, 0, "Gray")

menu.add(0, v.id, 0, "Cyan")

}

override fun onContextItemSelected(item: MenuItem): Boolean {

when (item.title) {

"Yellow" -> textView.setTextColor(Color.YELLOW)

"Gray" -> textView.setTextColor(Color.GRAY)

"Cyan" -> textView.setTextColor(Color.CYAN)

}

return true

}

**GEOCODER**  
val textView: TextView = findViewById(R.id.*details*)

val latitude = 9.9129 // Example: San Francisco

val longitude = 78.1477

val address=getAddressFromLatLong(latitude,longitude)

textView.*text*=address ?: "no location found"

private fun getAddressFromLatLong(latitude:Double,longitude:Double):String?{

val geocoder = Geocoder(this, Locale.getDefault())

val addressList= geocoder.getFromLocation(latitude, longitude, 1)

if (addressList != null && addressList.*isNotEmpty*()) {

val address= addressList[0]

return "${address.getAddressLine(0)}${address.*locality*}, ${address.*countryName*}"

}

else

return "Address Not Found"

}

override fun onCreateContextMenu(menu: ContextMenu, v: View,menuInfo:ContextMenu.ContextMenuInfo?){

super.onCreateContextMenu(menu,v,menuInfo)

menu.setHeaderTitle("Context Menu")

menu.add(0,v.*id*,0,"Yellow")

menu.add(0,v.*id*,0,"Green")

menu.add(0,v.*id*,0,"Red")

}

override fun onContextItemSelected(item: MenuItem):Boolean{

when(item.*title*){

"Yellow"->layout.setBackgroundColor(android.graphics.Color.*YELLOW*)

"Green"->layout.setBackgroundColor(android.graphics.Color.*GREEN*)

"Red"->layout.setBackgroundColor(android.graphics.Color.*RED*)

}

return true

}

}

**Progress Bar**

private lateinit var showProgressBtn: Button  
private lateinit var progressBar: ProgressBar  
private val handler = Handler(Looper.getMainLooper())

showProgressBtn = findViewById(R.id.*button*)  
 progressBar = findViewById(R.id.*progressBar*)  
  
 showProgressBtn.setOnClickListener **{** incrementProgress()  
 **}**}

private fun incrementProgress() {  
 Thread **{** for (i in 1..100) {  
 Thread.sleep(50)  
 handler.post **{** progressBar.*progress* = i  
 **}** }  
 **}**.start()

**DatePicker**

val dateButton = findViewById<Button>(R.id.*btn2*)

dateButton.setOnClickListener **{**

// Get current date

val calendar = Calendar.getInstance()

val year = calendar.get(Calendar.*YEAR*)

val month = calendar.get(Calendar.*MONTH*)

val day = calendar.get(Calendar.*DAY\_OF\_MONTH*)

// Create DatePickerDialog

val datePickerDialog = DatePickerDialog(

this,

**{** \_, selectedYear, selectedMonth, selectedDay **->**

val selectedDate = "$selectedDay/${selectedMonth + 1}/$selectedYear"

dateButton.*text* = selectedDate // Show it on the button

**}**,

year, month, day

)

datePickerDialog.show()

**}**

**TimePicker**

val timeButton = findViewById<Button>(R.id.timeButton)

timeButton.setOnClickListener {

val calendar = Calendar.getInstance()

val hour = calendar.get(Calendar.HOUR\_OF\_DAY)

val minute = calendar.get(Calendar.MINUTE)

val timePickerDialog = TimePickerDialog(

this,

{ \_, selectedHour, selectedMinute ->

val selectedTime = String.format("%02d:%02d", selectedHour, selectedMinute)

timeButton.text = selectedTime // Show selected time on the button

},

hour, minute, true // 'true' = 24-hour format. Set false for 12-hour format

)

timePickerDialog.show()

}

**AlertDialog**

val btn=findViewById<Button>(R.id.*btn2*)

btn.setOnClickListener**{**

val alert= AlertDialog.Builder(this)

alert.setTitle("alert")

.setMessage("sure to book?")

.setPositiveButton("yes")**{**dialog,\_**->**

Toast.makeText(this,"you successfully booked",Toast.*LENGTH\_SHORT*).show()

dialog.dismiss()**}**

.setNegativeButton("no")**{**dialog,\_**->**dialog.dismiss()**}**

.show()

**}**

**SharedPref**

class MainActivity : AppCompatActivity() {

private lateinit var sharedPref: SharedPreferences

private lateinit var keyEditText: EditText

private lateinit var valueEditText: EditText

private lateinit var resultTextView: TextView

private lateinit var addUpdateButton: Button

private lateinit var viewButton: Button

private lateinit var deleteButton: Button

private lateinit var clearAllButton: Button

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

sharedPref = getSharedPreferences("MyPrefs", Context.MODE\_PRIVATE)

keyEditText = findViewById(R.id.keyEditText)

valueEditText = findViewById(R.id.valueEditText)

resultTextView = findViewById(R.id.resultTextView)

addUpdateButton = findViewById(R.id.addUpdateButton)

viewButton = findViewById(R.id.viewButton)

deleteButton = findViewById(R.id.deleteButton)

clearAllButton = findViewById(R.id.clearAllButton)

addUpdateButton.setOnClickListener {

val key = keyEditText.text.toString().trim()

val value = valueEditText.text.toString().trim()

if (key.isNotEmpty() && value.isNotEmpty()) {

sharedPref.edit().putString(key, value).apply()

Toast.makeText(this, "Saved!", Toast.LENGTH\_SHORT).show()

} else {

Toast.makeText(this, "Enter key and value", Toast.LENGTH\_SHORT).show()

}

}

viewButton.setOnClickListener {

val allPrefs = sharedPref.all

val data = StringBuilder()

for ((k, v) in allPrefs) {

data.append("$k : $v\n")

}

resultTextView.text = if (data.isNotEmpty()) data.toString() else "No data found"

}

deleteButton.setOnClickListener {

val key = keyEditText.text.toString().trim()

if (key.isNotEmpty() && sharedPref.contains(key)) {

sharedPref.edit().remove(key).apply()

Toast.makeText(this, "Deleted!", Toast.LENGTH\_SHORT).show()

} else {

Toast.makeText(this, "Key not found", Toast.LENGTH\_SHORT).show()}}

clearAllButton.setOnClickListener {

sharedPref.edit().clear().apply()

Toast.makeText(this, "All data cleared", Toast.LENGTH\_SHORT).show()}}}

**FusedLocationProvider**

private lateinit var locationProvider: FusedLocationProviderClient

private lateinit var viewBtn: Button

locationProvider = LocationServices.getFusedLocationProviderClient(this)

viewBtn = findViewById(R.id.btn4)

private fun getLastLocation() {

if (ActivityCompat.checkSelfPermission(

this,

Manifest.permission.*ACCESS\_FINE\_LOCATION*

) != PackageManager.*PERMISSION\_GRANTED* && ActivityCompat.checkSelfPermission(

this,

Manifest.permission.*ACCESS\_COARSE\_LOCATION*

) != PackageManager.*PERMISSION\_GRANTED*

) {

// Request location permission

ActivityCompat.requestPermissions(

this,

*arrayOf*(Manifest.permission.*ACCESS\_FINE\_LOCATION*),

100

)

return

}

// Now that we have permission, get the location

locationProvider.*lastLocation*

.addOnSuccessListener **{** location **->**

if (location != null) {

val lat = location.*latitude*

val lon = location.*longitude*

val detailsView = findViewById<TextView>(R.id.*details*)

detailsView.*text* = "Latitude: $lat\nLongitude: $lon"

} else {

Toast.makeText(this, "Location not available", Toast.*LENGTH\_SHORT*).show()

}

**}**

.addOnFailureListener **{**

Toast.makeText(this, "Failed to get location", Toast.*LENGTH\_SHORT*).show()

**}**

}

SQLite

import android.content.ContentValues  
import android.database.Cursor  
import android.database.sqlite.SQLiteDatabase  
import android.os.Bundle  
import android.widget.\*  
import androidx.activity.enableEdgeToEdge  
import androidx.appcompat.app.AppCompatActivity  
  
class MainActivity : AppCompatActivity() {  
  
 private lateinit var dbHelper: DatabaseHelper  
 private lateinit var database: SQLiteDatabase  
  
 private lateinit var nameInput: EditText  
 private lateinit var ageInput: EditText  
 private lateinit var addButton: Button  
 private lateinit var updateButton: Button  
 private lateinit var deleteButton: Button  
 private lateinit var viewButton: Button  
 private lateinit var resultText: TextView  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *enableEdgeToEdge*()  
 setContentView(R.layout.*activity\_main*)  
  
 dbHelper = DatabaseHelper(this)  
 database = dbHelper.*writableDatabase* nameInput = findViewById(R.id.*nameInput*)  
 ageInput = findViewById(R.id.*ageInput*)  
 addButton = findViewById(R.id.*addButton*)  
 updateButton = findViewById(R.id.*updateButton*)  
 deleteButton = findViewById(R.id.*deleteButton*)  
 viewButton = findViewById(R.id.*viewButton*)  
 resultText = findViewById(R.id.*resultText*)  
  
 addButton.setOnClickListener **{** addData() **}** updateButton.setOnClickListener **{** updateData() **}** deleteButton.setOnClickListener **{** deleteData() **}** viewButton.setOnClickListener **{** viewData() **}** }  
  
 private fun addData() {  
 val name = nameInput.*text*.toString()  
 val age = ageInput.*text*.toString().*toIntOrNull*()  
  
 if (name.*isNotEmpty*() && age != null) {  
 val values = ContentValues().*apply* **{** put("name", name)  
 put("age", age)  
 **}** database.insert("users", null, values)  
 Toast.makeText(this, "Data Inserted", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, "Please enter valid details", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 private fun updateData() {  
 val name = nameInput.*text*.toString()  
 val age = ageInput.*text*.toString().*toIntOrNull*()  
  
 if (name.*isNotEmpty*() && age != null) {  
 val values = ContentValues().*apply* **{** put("age", age)  
 **}** val rowsUpdated = database.update("users", values, "name=?", *arrayOf*(name))  
 if (rowsUpdated > 0) {  
 Toast.makeText(this, "Data Updated", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, "No record found", Toast.*LENGTH\_SHORT*).show()  
 }  
 } else {  
 Toast.makeText(this, "Please enter valid details", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 private fun deleteData() {  
 val name = nameInput.*text*.toString()  
  
 if (name.*isNotEmpty*()) {  
 val rowsDeleted = database.delete("users", "name=?", *arrayOf*(name))  
 if (rowsDeleted > 0) {  
 Toast.makeText(this, "Data Deleted", Toast.*LENGTH\_SHORT*).show()  
 } else {  
 Toast.makeText(this, "No record found", Toast.*LENGTH\_SHORT*).show()

}  
 } else {  
 Toast.makeText(this, "Please enter a name", Toast.*LENGTH\_SHORT*).show()  
 }  
 }  
  
 private fun viewData() {  
 val cursor: Cursor = database.rawQuery("SELECT \* FROM users", null)  
 val data = StringBuilder()  
  
 if (cursor.moveToFirst()) {  
 do {  
 val id = cursor.getInt(0)  
 val name = cursor.getString(1)  
 val age = cursor.getInt(2)  
 data.append("ID: $id, Name: $name, Age: $age\n")  
 } while (cursor.moveToNext())  
 } else {  
 data.append("No records found.")  
 }  
 cursor.close()  
  
 resultText.*text* = data.toString()  
 }  
}

DatabaseHelper.kt

import android.content.Context  
import android.database.sqlite.SQLiteDatabase  
import android.database.sqlite.SQLiteOpenHelper  
  
class DatabaseHelper(context: Context) :  
 SQLiteOpenHelper(context, DATABASE\_NAME, null, DATABASE\_VERSION) {  
  
 companion object {  
 private const val DATABASE\_NAME = "UserDatabase.db"  
 private const val DATABASE\_VERSION = 1  
 private const val TABLE\_USERS = "users"  
 private const val CREATE\_TABLE = """  
 CREATE TABLE $TABLE\_USERS (  
 id INTEGER PRIMARY KEY AUTOINCREMENT,  
 name TEXT,  
 age INTEGER  
 )  
 """  
 }  
  
 override fun onCreate(db: SQLiteDatabase) {  
 db.execSQL(CREATE\_TABLE)  
 }  
  
 override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {  
 db.execSQL("DROP TABLE IF EXISTS users")  
 onCreate(db)  
 }  
}