**Section 12 - Layouts (Lucky Number Application)**

**Key Concepts Taught**

1. **Activity Navigation**
   * Explicit Intents: Switching between activities within the same app.
   * Implicit Intents: Sharing data with external apps (e.g., Gmail, WhatsApp).
2. **Data Passing**
   * Sending data: putExtra() with key-value pairs in intents.
   * Receiving data: getIntent().getStringExtra() (or other type-specific methods).
3. **Layout Design**
   * Using ConstraintLayout with constraints for UI elements.
   * Customizing views (backgrounds, text colors, margins).
4. **Random Number Generation**
   * Using Java's Random class to generate pseudo-random numbers.
5. **Sharing Data**
   * Intent.ACTION\_SEND for sharing plain text via implicit intents.
   * Pre-populating subject/body using Intent.EXTRA\_SUBJECT and Intent.EXTRA\_TEXT.

**Step-by-Step Implementation**

**1. Project Setup & Layouts**  
**Tools**: Android Studio, ConstraintLayout.  
**Files**:

* activity\_main.xml (MainActivity)
* activity\_second.xml (SecondActivity)

**Steps**:  
**MainActivity Layout**:

xml

*<!-- Background -->*

<androidx.constraintlayout.widget.ConstraintLayout

android:background="@drawable/background\_number\_one">

*<!-- Welcome Text -->*

<TextView

android:text="Welcome to Lucky Number"

android:textSize="32sp"

android:textColor="#FFFFFF"/>

*<!-- Name Input -->*

<EditText

android:id="@+id/edit\_text\_name"

android:hint="Please enter your name"

android:textColor="#FFFFFF"

android:textColorHint="#FFFFFF"

android:layout\_marginHorizontal="16dp"/>

*<!-- Button -->*

<Button

android:id="@+id/btn\_wish"

android:text="Wish me luck"/>

*<!-- Dice Image -->*

<ImageView

android:id="@+id/image\_dice"

android:layout\_width="200dp"

android:layout\_height="140dp"

android:src="@drawable/dice\_image"/>

</androidx.constraintlayout.widget.ConstraintLayout>

**SecondActivity Layout**:

xml

*<!-- Background -->*

<androidx.constraintlayout.widget.ConstraintLayout

android:background="@drawable/background\_number\_two">

*<!-- Lucky Number Display -->*

<TextView

android:id="@+id/lucky\_number\_text"

android:textColor="#FFFFFF"/>

*<!-- Share Button -->*

<Button

android:id="@+id/share\_btn"

android:text="Share"/>

</androidx.constraintlayout.widget.ConstraintLayout>

**2. Passing Data to SecondActivity (Explicit Intent)**  
**APIs**: Intent, putExtra(), startActivity().  
**Code (MainActivity.java)**:

java

Button btn = findViewById(R.id.btn\_wish);

EditText editText = findViewById(R.id.edit\_text\_name);

btn.setOnClickListener(v -> {

String userName = editText.getText().toString();

*// Create explicit intent*

Intent i = new Intent(MainActivity.this, SecondActivity.class);

i.putExtra("name", userName); *// Key: "name", Value: userName*

startActivity(i);

});

**3. Receiving Data & Generating Random Number**  
**APIs**: getIntent(), Random class.  
**Code (SecondActivity.java)**:

java

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_second);

*// Get data from intent*

Intent i = getIntent();

String userName = i.getStringExtra("name"); *// Key must match "name"*

*// Generate random number (0-1000)*

int randomNum = generateRandomNumber();

TextView luckyNumberText = findViewById(R.id.lucky\_number\_text);

luckyNumberText.setText("Your lucky number is: " + randomNum);

}

private int generateRandomNumber() {

Random random = new Random();

return random.nextInt(1000); *// Upper bound: 1000 (0-999)*

}

**4. Sharing Data via Implicit Intent**  
**APIs**: Intent.ACTION\_SEND, Intent.createChooser().  
**Code (SecondActivity.java)**:

java

Button shareBtn = findViewById(R.id.share\_btn);

shareBtn.setOnClickListener(v -> shareData(userName, randomNum));

private void shareData(String userName, int randomNum) {

Intent i = new Intent(Intent.ACTION\_SEND);

i.setType("text/plain"); *// MIME type for plain text*

*// Pre-populate subject and body*

i.putExtra(Intent.EXTRA\_SUBJECT, userName + " got lucky today!");

i.putExtra(Intent.EXTRA\_TEXT, "His lucky number is: " + randomNum);

*// Show app chooser dialog*

startActivity(Intent.createChooser(i, "Choose a platform"));

}

**Tools & APIs Used**

1. **Android APIs**:
   * Intent (Explicit/Implicit)
   * putExtra() / getStringExtra()
   * Intent.ACTION\_SEND, Intent.EXTRA\_SUBJECT, Intent.EXTRA\_TEXT
   * Random (Java)
   * ConstraintLayout
2. **Resources**:
   * Drawables: background\_number\_one, background\_number\_two, dice\_image.
   * Dimensional values (e.g., 32sp, 200dp).

**Best Practices & Alternatives**

1. **Data Passing**:
   * **Best Practice**: Use constants for intent keys (e.g., public static final String KEY\_NAME = "name";) to avoid typos.
   * **Alternative**: Use Bundle objects for passing multiple data pieces.
2. **Random Number Generation**:
   * **Best Practice**: For cryptographic security, use SecureRandom instead of Random.
   * **Alternative**: Use ThreadLocalRandom for better performance in multi-threaded environments.
3. **Implicit Intents**:
   * **Best Practice**: Always use Intent.createChooser() to force an app selector.
   * **Handle No Apps**: Check if any app can handle the intent:

java

if (i.resolveActivity(getPackageManager()) != null) {

startActivity(chooser);

} else {

Toast.makeText(this, "No app found!", Toast.LENGTH\_SHORT).show(); }

1. **Layout Design**:
   * Use android:layout\_margin for consistent spacing.
   * Extract hardcoded strings to strings.xml for localization.

**Part B: Important Topics Not Covered**

1. **Data Persistence**:
   * Saving data across app restarts using SharedPreferences or Room Database.
   * Why: The app loses the lucky number if the user rotates the screen/restarts the app.
2. **Lifecycle Awareness**:
   * Handling configuration changes (e.g., screen rotation) by overriding onSaveInstanceState().
   * Why: Random number regenerates on rotation, breaking UX.
3. **Input Validation**:
   * Checking for empty names in MainActivity:

java

if (userName.trim().isEmpty()) {

editText.setError("Name cannot be empty!");

return;

}

1. **Modern Navigation**:
   * Using Jetpack Navigation Component for type-safe activity/fragment navigation.
   * Why: Simplifies complex navigation graphs and reduces boilerplate.
2. **Sharing Rich Content**:
   * Sharing images/files using Intent.ACTION\_SEND with URI streams.
   * Why: The app only shares plain text; users might want to share the dice image.
3. **Testing**:
   * Unit testing Random logic with fixed seeds.
   * Espresso tests for UI interactions (e.g., button clicks, intent mocking).
4. **Accessibility**:
   * Adding contentDescription to ImageView for screen readers.
   * Why: The dice image is decorative but needs accessibility support.
   * Add android:contentDescription to ImageView for screen readers.

xml

CopyEdit

android:contentDescription="Dice icon"

1. **Dependency Injection**:
   * Using Dagger/Hilt to inject Random or other dependencies.
   * Why: Promotes testable and maintainable code.

**9 . Modern Architecture (MVVM)**

* Move business logic like generateRandomNumber() to ViewModel.
* Use **LiveData** or **StateFlow** to observe UI data.
* Benefit: Better testability and lifecycle awareness.

**10. Jetpack Compose (Modern UI Toolkit)**

* Equivalent app in Compose would use:
  + TextField for input
  + Button for events
  + remember and mutableStateOf for state
  + NavHost for navigation