ANDROID APP DEVELOPMENT 1 - ITE-5333-0NA

LAB 9 - CASE STUDY

NAME : Linga Sai Srilaxmi

ID : N01653528

1. **What are the key differences between explicit and implicit intents?**

ANS :

**Explicit Intents:**

* Used to start a specific activity within the same app.
* Requires specifying the class name of the target component.
* Used for navigation within the app, like moving from one screen to another.

**Implicit Intents:**

* Used to request an action that any compatible app can handle.
* No specific component is mentioned; the system finds the appropriate app.
* Commonly used for actions like opening a web link, sending an email, or sharing data.

2**. How would you handle a scenario where an application is unavailable to handle an implicit intent**?

ANS :

* If no app is available to handle an implicit intent, the intent will fail, possibly crashing the app.
* To prevent this, check if an app can handle the intent before starting it:

val intent = Intent(Intent.ACTION\_VIEW, Uri.parse("https://www.google.com"))

if (intent.resolveActivity(packageManager) != null) {

startActivity(intent)

} else {

Toast.makeText(this, "No app found to handle this action", Toast.LENGTH\_SHORT).show()

}

* This way, the app doesn’t crash, and the user is informed that no suitable app is installed.

**3. Describe the process of securing an intent to prevent unauthorized data access.**

ANS : **Avoid Sending Sensitive Data via Implicit Intents**

* Any app that can handle the intent might access the data, so avoid sending personal or sensitive information through implicit intents.

**Use Permissions for Secure Data Sharing**

* When sharing files or data between apps, use FileProvider and grant only necessary permissions.

**Restrict Exported Components**

* In the AndroidManifest.xml, ensure components are not unnecessarily exposed:

<activity android:name=".MyActivity" android:exported="false"/>

* This prevents unauthorized apps from accessing certain activities or services.

**4. When should you prefer Parcelable over Serializable, and why?**

ANS :

**Parcelable is better for Android because:**

* It is **faster and more efficient** than Serializable.
* It does not use reflection, making it optimized for Android’s memory management.
* It reduces the amount of code needed when using @Parcelize.

**Serializable is slower because:**

* It uses reflection, which consumes more CPU and memory.
* It is easier to implement but can slow down the app when transferring large objects.

@Parcelize

data class User(val name: String, val age: Int) : Parcelable

* This makes it easier to send objects between activities using intents.

**5. How can you ensure large data transfers do not compromise application  
performance or security**?

ANS :

**Avoid Passing Large Data Directly in Intents**

* If a large object (like a high-resolution image or a big list) is passed in an intent, it can slow down the app or even cause crashes.

**Use Storage Instead:**

* Store large data in a **database, shared preferences, or files** instead of sending it through an intent.
* Instead of passing actual data, pass only a reference (like a file URI or database ID).

**6. How can implicit intents be used to interact with other apps in a secure way?**

ANS :

* **Use Intent.createChooser()** to let the user pick an app:

val intent = Intent(Intent.ACTION\_SEND)

intent.putExtra(Intent.EXTRA\_TEXT, "Hello!")

intent.type = "text/plain"

startActivity(Intent.createChooser(intent, "Share via"))

* **Use FileProvider** instead of direct file paths to protect sensitive data.
* **Limit permissions** when sharing data, only allowing read or write when necessary.

**7. What are the best practices for ensuring data integrity during inter-app  
communication?**

ANS :

**Validate Incoming Data**

* Always check for **null values and correct data types** before using received data to prevent crashes.

**Use Intent Filters Wisely**

* Only declare intent filters for actions that your app truly supports.

**Secure Data Sharing**

* If sensitive data needs to be shared, use explicit intents instead of implicit ones.

**Use Encryption When Necessary**

* If the data being shared is sensitive, encrypt it before sending and decrypt it upon receiving.
* Example of checking null values:
* val receivedData = intent.getStringExtra("key") ?: "Default Value"