**✅ Section 23: ViewPager2 with Fragments & TabLayout – Master Notes**

**🔷 I. Key Concepts Taught**

| **Concept** | **Description** |
| --- | --- |
| **ViewPager2** | Modern widget for horizontal/vertical swiping through fragments or views. |
| **Fragments** | Reusable UI components displayed inside ViewPager2 pages. |
| **FragmentStateAdapter** | Adapter that manages fragment lifecycles efficiently for ViewPager2. |
| **TabLayout** | Material component providing tab-based navigation UI. |
| **TabLayoutMediator** | Connects TabLayout and ViewPager2 to sync swipes and tab selections. |

**🔷 II. Project Setup: Step-by-Step**

**📁 Step 1: Create a New Project**

* Project Name: ViewPagerApp
* Template: Empty Views Activity
* Language: Java

**📦 Step 2: Add Required Dependencies**

In build.gradle (app):

gradle

CopyEdit

dependencies {

implementation 'androidx.viewpager2:viewpager2:1.0.0'

implementation 'com.google.android.material:material:1.10.0' // or latest

}

✅ Sync the Gradle after adding dependencies.

**📐 Step 3: Design activity\_main.xml Layout**

xml

CopyEdit

<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<com.google.android.material.tabs.TabLayout

android:id="@+id/tabLayout"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toTopOf="parent"/>

<androidx.viewpager2.widget.ViewPager2

android:id="@+id/viewPager2"

android:layout\_width="match\_parent"

android:layout\_height="0dp"

app:layout\_constraintTop\_toBottomOf="@id/tabLayout"

app:layout\_constraintBottom\_toBottomOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>

💡 0dp height + constraints allow ViewPager2 to fill remaining space.

**🔷 III. Create Fragments**

Create 3 fragments: FragmentOne, FragmentTwo, FragmentThree

**✅ Example: FragmentOne.java**

java

CopyEdit

public class FragmentOne extends Fragment {

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

return inflater.inflate(R.layout.fragment\_one, container, false); // Inflate UI

}

}

**fragment\_one.xml**

xml

CopyEdit

<FrameLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="#FFCDD2">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Fragment One"

android:textSize="24sp"

android:layout\_gravity="center"/>

</FrameLayout>

🔁 Repeat similar structure for FragmentTwo and FragmentThree with different colors/text.

**🔷 IV. Create ViewPager Adapter**

**✅ MyViewPagerAdapter.java**

java

CopyEdit

public class MyViewPagerAdapter extends FragmentStateAdapter {

private final ArrayList<Fragment> fragmentList = new ArrayList<>();

public MyViewPagerAdapter(@NonNull FragmentManager fragmentManager, @NonNull Lifecycle lifecycle) {

super(fragmentManager, lifecycle);

}

public MyViewPagerAdapter(@NonNull FragmentActivity activity) {

super(activity);

}

public void addFragment(Fragment fragment) {

fragmentList.add(fragment);

}

@NonNull

@Override

public Fragment createFragment(int position) {

return fragmentList.get(position); // Returns fragment by position

}

@Override

public int getItemCount() {

return fragmentList.size(); // Total fragments

}

}

🔄 Supports both FragmentManager + Lifecycle or just FragmentActivity.

**🔷 V. Final Integration in MainActivity.java**

java

CopyEdit

public class MainActivity extends AppCompatActivity {

ViewPager2 viewPager2;

TabLayout tabLayout;

MyViewPagerAdapter adapter;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Initialize views

viewPager2 = findViewById(R.id.viewPager2);

tabLayout = findViewById(R.id.tabLayout);

// Initialize adapter

adapter = new MyViewPagerAdapter(getSupportFragmentManager(), getLifecycle());

// Add fragments to adapter

adapter.addFragment(new FragmentOne());

adapter.addFragment(new FragmentTwo());

adapter.addFragment(new FragmentThree());

// Set adapter to ViewPager2

viewPager2.setAdapter(adapter);

viewPager2.setOrientation(ViewPager2.ORIENTATION\_HORIZONTAL); // Optional

// Attach TabLayout with ViewPager2

new TabLayoutMediator(tabLayout, viewPager2, (tab, position) -> {

tab.setText("Fragment " + (position + 1));

// Optional: tab.setIcon(R.drawable.ic\_tab);

}).attach();

}

}

**🔷 VI. Key Behaviors & Internal Flow**

| **Component** | **Purpose** |
| --- | --- |
| ViewPager2 | Swipes between fragments |
| FragmentStateAdapter | Supplies & manages fragment lifecycles |
| TabLayout | UI tabs for navigation |
| TabLayoutMediator | Binds tabs to pages and syncs swipe/tab interactions |
| FragmentManager | Manages fragment operations behind the scenes |

**🧠 Instructor's Key Insights**

* ✅ Prefer FragmentStateAdapter over older adapters (memory-efficient)
* ✅ Always connect TabLayout via TabLayoutMediator
* ✅ Never forget .attach() at the end
* ✅ Pass lifecycle-aware components (getSupportFragmentManager() + getLifecycle())

**💡 Best Practices & Architecture Tips**

| **Area** | **Practice** |
| --- | --- |
| **Adapter Design** | Add fragments via a list (not hardcoded) |
| **Lifecycle** | Use FragmentStateAdapter for better lifecycle and memory management |
| **Layout Constraints** | Always use 0dp + constraints for responsive UI |
| **Performance** | Set viewPager2.setOffscreenPageLimit(1) to limit memory usage |
| **Customization** | Use icons + text on tabs (tab.setText(), tab.setIcon()) |
| **Modular Fragments** | Avoid business logic inside fragments; use ViewModel for state |

**✅ Real-World Use Cases**

| **Feature** | **How ViewPager2 Helps** |
| --- | --- |
| Onboarding Screens | Swipe through intro fragments |
| Tab-Based UIs | Social apps (Home, Chat, Profile) |
| Multi-Step Forms | Step-by-step wizard forms |
| E-commerce Product Details | Tabs: Description, Reviews, Q&A |
| Messaging/Chat Screens | Conversations, Groups, Archive tabs |

**🔁 Part B: Important But Not Covered in This Section**

| **Topic** | **Why Learn It** |
| --- | --- |
| **Fragment Arguments** | Use Bundle to pass data to each fragment |
| **Shared ViewModel** | Share data between fragments using ViewModel |
| **Back Stack Handling** | Add swipe history or manage state when returning |
| **Page Transitions** | Add slide, zoom, fade animations using PageTransformer |
| **Nested Fragments** | Host ViewPager inside another fragment |
| **Compose Alternative** | Use Jetpack Compose's HorizontalPager, TabRow |
| **RTL Support** | Enable right-to-left swiping for Arabic/Hebrew layouts |
| **Safe Args + Navigation** | Combine ViewPager with Navigation Graph |
| **Testing** | Use Espresso + FragmentScenario for ViewPager2 UI tests |
| **Accessibility** | Provide content descriptions for tab titles |

**📚 Tools, Libraries, APIs Used**

| **Tool/API** | **Use** |
| --- | --- |
| androidx.viewpager2.widget.ViewPager2 | Swipe-based paging mechanism |
| FragmentStateAdapter | Fragment lifecycle-aware paging adapter |
| com.google.android.material.tabs.TabLayout | Material tab navigation |
| TabLayoutMediator | Links tabs with ViewPager2 |
| ConstraintLayout | Layout parent in activity\_main.xml |
| FragmentManager / Lifecycle | For adapter construction |

**✅ Final Checklist for ViewPager2 + TabLayout Projects**

✔ Use ViewPager2 for swipe navigation  
✔ Add FragmentStateAdapter to manage pages  
✔ Supply fragments dynamically using ArrayList  
✔ Attach with TabLayoutMediator  
✔ Dynamically set tab text/icon using lambda  
✔ Set layout constraints to make UI responsive  
✔ Separate business logic via ViewModel  
✔ Explore transitions & animations for visual polish  
✔ Use modern libraries (Material, AndroidX, ViewPager2)  
✔ Test navigation and swipes with UI test tools

✅ **Instructor’s Final Thought**  
“ViewPager2 with FragmentStateAdapter gives you performance, flexibility, and a modern swipe experience. Use it over ViewPager1. Always keep your fragments clean and lifecycle-aware.”