**Section 65: Navigation Drawer**.

**1. Key Concepts Covered**

1. **Navigation Drawer**
   * A side panel that slides in from the left (or right) to show app navigation options.
   * Common in apps with multiple top-level destinations (e.g., Gmail, YouTube).
2. **DrawerLayout**
   * Main container layout that hosts the **content** and the **drawer panel**.
3. **NavigationView**
   * Specialized view inside DrawerLayout to display navigation menu items.
   * Populated via **menu resource file** and optional **header layout**.
4. **Fragments**
   * Modular sections of UI, swapped dynamically without restarting Activity.
   * Used here for **Home**, **Message**, **Settings**, **Login**.
5. **ActionBarDrawerToggle**
   * Synchronizes the drawer state with the toolbar's “hamburger” icon.
6. **Menu Grouping**
   * Organizing menu items into sections for better navigation structure.
7. **Fragment Transactions**
   * Replacing the content area with the selected fragment smoothly.

**2. Step-by-Step Implementation**

**Step 1 — Create a New Project**

* Choose **Empty Activity**.
* Name: NavDrawerApp
* Language: **Kotlin**
* Finish.

**Step 2 — Layout Setup**

**activity\_main.xml**

<androidx.drawerlayout.widget.DrawerLayout

android:id="@+id/drawer\_layout"

android:layout\_width="match\_parent"

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

<!-- Main Content Area -->

<androidx.constraintlayout.widget.ConstraintLayout

android:layout\_width="match\_parent"

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

<FrameLayout

android:id="@+id/frame\_layout1"

android:layout\_width="match\_parent"

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

</androidx.constraintlayout.widget.ConstraintLayout>

<!-- Navigation Drawer Panel -->

<com.google.android.material.navigation.NavigationView

android:id="@+id/nav\_view"

android:layout\_width="wrap\_content"

android:layout\_height="match\_parent"

android:layout\_gravity="start"

app:headerLayout="@layout/nav\_header"

app:menu="@menu/nav\_menu"

android:fitsSystemWindows="true" />

</androidx.drawerlayout.widget.DrawerLayout>

**Notes:**

* DrawerLayout is the root container.
* FrameLayout hosts the current fragment.
* NavigationView contains the menu & header.

**Step 3 — Create Header Layout**

**nav\_header.xml**

<TextView

android:id="@+id/textViewHeader"

android:layout\_width="wrap\_content"

android:layout\_height="150dp"

android:gravity="center"

android:text="Navigation Header"

android:textSize="30sp"

android:textStyle="bold"

android:background="@color/purple\_700"

android:textColor="@android:color/white" />

**Step 4 — Create Menu Resource**

**res/menu/nav\_menu.xml**

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

<!-- Group 1 -->

<group android:checkableBehavior="single">

<item android:id="@+id/home"

android:title="Home Page" />

<item android:id="@+id/message"

android:title="Message Page" />

<item android:id="@+id/settings"

android:title="Settings Page" />

</group>

<!-- Group 2 -->

<item android:title="Profile">

<menu>

<group android:checkableBehavior="single">

<item android:id="@+id/login"

android:title="Login Page" />

</group>

</menu>

</item>

</menu>

**Key Points:**

* checkableBehavior="single" ensures only one item is selected at a time.
* Groups improve UX by separating categories.

**Step 5 — Create Fragments**

Example: **HomeFragment.kt**

class HomeFragment : Fragment() {

override fun onCreateView(

inflater: LayoutInflater, container: ViewGroup?,

savedInstanceState: Bundle?

): View? {

return inflater.inflate(R.layout.fragment\_home, container, false)

}

}

**fragment\_home.xml**

<TextView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:text="Home Fragment"

android:textSize="32sp"

android:gravity="center"

android:background="#2196F3"

android:textColor="@android:color/white" />

Repeat similarly for:

* MessageFragment
* SettingsFragment
* LoginFragment

**Step 6 — MainActivity Logic**

**MainActivity.kt**

class MainActivity : AppCompatActivity() {

private lateinit var toggle: ActionBarDrawerToggle

private lateinit var drawerLayout: DrawerLayout

private lateinit var navView: NavigationView

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

drawerLayout = findViewById(R.id.drawer\_layout)

navView = findViewById(R.id.nav\_view)

// Setup Drawer Toggle (Hamburger Icon)

toggle = ActionBarDrawerToggle(

this, drawerLayout,

R.string.open, R.string.close

)

drawerLayout.addDrawerListener(toggle)

toggle.syncState()

supportActionBar?.setDisplayHomeAsUpEnabled(true)

// Default Fragment

replaceFragment(HomeFragment(), "Home Page")

// Navigation Item Click Handling

navView.setNavigationItemSelectedListener {

when (it.itemId) {

R.id.home -> replaceFragment(HomeFragment(), it.title.toString())

R.id.message -> replaceFragment(MessageFragment(), it.title.toString())

R.id.settings -> replaceFragment(SettingsFragment(), it.title.toString())

R.id.login -> replaceFragment(LoginFragment(), it.title.toString())

}

drawerLayout.closeDrawers()

true

}

}

// Function to Replace Fragments

private fun replaceFragment(fragment: Fragment, title: String) {

supportFragmentManager.beginTransaction()

.replace(R.id.frame\_layout1, fragment)

.commit()

setTitle(title)

}

override fun onOptionsItemSelected(item: MenuItem): Boolean {

return if (toggle.onOptionsItemSelected(item)) true

else super.onOptionsItemSelected(item)

}

}

**Strings (res/values/strings.xml)**

<string name="open">Open</string>

<string name="close">Close</string>

**3. Tools, Libraries, and APIs Used**

* **AndroidX DrawerLayout** (androidx.drawerlayout.widget.DrawerLayout)
* **Material Components** NavigationView (com.google.android.material.navigation.NavigationView)
* **Fragments API** (androidx.fragment.app.Fragment)
* **ActionBarDrawerToggle** for icon animation
* **Menu Resource System** for navigation items

**4. Best Practices (Expanded)**

1. **Follow Material Design Guidelines**
   * Google’s Material Design defines recommended drawer width, icon size, padding, and animation behavior.
   * Recommended drawer width = min(screen width - 56dp, 320dp) for phones.
   * Icons should be **24dp** in size with **72dp padding** for touch targets.
   * Use materialComponents theme to get built-in ripple and elevation styles.  
     **Why:** Ensures consistent UX across apps and devices.  
     **Example:**
2. <com.google.android.material.navigation.NavigationView
3. style="@style/Widget.MaterialComponents.NavigationView"
4. android:layout\_width="wrap\_content"
5. android:layout\_height="match\_parent" />
6. **Reserve Navigation Drawer for Top-Level Destinations**
   * Drawer is meant for *main sections* of your app — avoid adding deep links or rarely-used pages.  
     **Why:** Reduces clutter, improves discoverability of important pages.  
     **Tip:** Secondary actions (like “Help”) should go in the bottom group or overflow menu.
7. **Close Drawer After Selection**
   * Always call drawerLayout.closeDrawers() after handling a click.  
     **Why:** Keeps UX smooth and prevents accidental multiple taps.  
     **Example:**
8. navView.setNavigationItemSelectedListener {
9. replaceFragment(HomeFragment(), "Home")
10. drawerLayout.closeDrawers()
11. true
12. }
13. **Use Fragments Instead of Multiple Activities**
    * Load content in the same Activity via fragments to keep the drawer persistent.  
      **Why:**
      + Avoids reloading the drawer menu on every screen change.
      + Improves performance and memory usage.  
        **Example:** Your *Vote Chain* app can switch between **DashboardFragment**, **ResultsFragment**, and **SettingsFragment** without destroying the navigation drawer.
14. **Maintain Consistent IDs Between XML and Code**
    * Ensure itemId in nav\_menu.xml exactly matches the ID in when (it.itemId) block.  
      **Why:** Avoids navigation bugs and unhandled clicks.  
      **Example:**
15. <item android:id="@+id/settings" android:title="Settings" />
16. when (it.itemId) {
17. R.id.settings -> openSettings()
18. }
19. **Limit Drawer Items for Clarity**
    * Ideally 3–7 items. Group them logically (e.g., “Profile” group, “App” group).
    * More items → slower user decision-making (**Hick’s Law**).
20. **Use Icons with Titles for Accessibility**
    * Icons help recognition; text helps comprehension.
    * Ensure content descriptions for screen readers.  
      **Example:**
21. <item
22. android:id="@+id/home"
23. android:title="Home"
24. android:icon="@drawable/ic\_home"
25. android:contentDescription="Home Page" />

**Part B — Important But Not Covered (Expanded)**

1. **Navigation Component Integration**
   * Instead of manually replacing fragments, use **Jetpack Navigation Component**:
     + Add NavHostFragment in XML as the content area.
     + Create a nav\_graph.xml for all destinations.
     + Link NavigationView to NavController with NavigationUI.setupWithNavController().  
       **Benefits:**
     + Automatic back stack management.
     + Safe Args plugin for type-safe data passing between destinations.  
       **Example:**
2. val navController = findNavController(R.id.nav\_host\_fragment)
3. navView.setupWithNavController(navController)
4. **Responsive Drawer (Right-Side / End Drawer)**
   * Use android:layout\_gravity="end" for right-hand drawers.
   * Can hold filters, advanced options (e.g., filter election results in *Vote Chain*).
   * Lock/unlock programmatically with:
5. drawerLayout.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_LOCKED\_CLOSED, GravityCompat.END)
6. **Handling Back Stack Properly**
   * If you want back navigation between fragments:
7. supportFragmentManager.beginTransaction()
8. .replace(R.id.frame\_layout1, fragment)
9. .addToBackStack(null)
10. .commit()
    * Without addToBackStack(), back button will exit the app.
11. **RTL (Right-to-Left) Language Support**
    * For Arabic/Hebrew users, swap drawer position:
12. android:layout\_gravity="start"
13. android:fitsSystemWindows="true"
    * start automatically adapts to LTR or RTL based on locale.
14. **Custom Header with Profile Image & Dynamic Data**
    * Inflate custom header view:
15. val headerView = navView.getHeaderView(0)
16. val userName = headerView.findViewById<TextView>(R.id.user\_name)
17. userName.text = "John Doe"
    * Use **Glide** or **Coil** for profile pictures:
18. Glide.with(this).load(userImageUrl).into(profileImageView)
19. **Theming Drawer Items**
    * Use **color selectors** for checked/unchecked states:
20. <selector xmlns:android="http://schemas.android.com/apk/res/android">
21. <item android:color="@color/selected" android:state\_checked="true"/>
22. <item android:color="@color/unselected"/>
23. </selector>
    * Apply via app:itemTextColor and app:itemIconTint.
24. **Drawer Lock Modes for Context-Specific Behavior**
    * Prevent drawer opening in certain screens (e.g., during voting submission).
25. drawerLayout.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_LOCKED\_CLOSED)
    * Unlock later:
26. drawerLayout.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_UNLOCKED)