Navigation3 in Compose

Aayush Gupta

Dependencies

```toml
[versions]
adaptive-navigation3 = "1.3.0-alpha03"
kotlin = "2.2.20"
lifecycle-navigation3 = "1.0.0-alpha04"
navigation3 = "1.0.0-beta01"
serialization = "1.9.0"

[libraries]
androidx-adaptive-navigation3 = { module = "androidx.compose.material3.adaptive:adaptive-navigation3", version.ref = "adaptive-navigation3" }
androidx-lifecycle-navigation3 = { module = "androidx.lifecycle:lifecycle-viewmodel-navigation3", version.ref = "lifecycle-navigation3" }
androidx-navigation3-runtime = { module = "androidx.navigation3:navigation3-runtime", version.ref = "navigation3" }
androidx-navigation3-ui = { module = "androidx.navigation3:navigation3-ui", version.ref = "navigation3" }
kotlinx-serialization-json = { module = "org.jetbrains.kotlinx:kotlinx-serialization-json", version.ref = "serialization" }

[plugins]
jetbrains-kotlin-serialization = { id = "org.jetbrains.kotlin.plugin.serialization", version.ref = "kotlin" }
```

Non-Adaptive

```kotlin
import android.os.Parcelable
import androidx.navigation3.runtime.NavKey
import com.aurora.store.data.model.PermissionType
import kotlinx.parcelize.Parcelize
import kotlinx.serialization.Serializable

/**
 * Destinations for navigation in compose
 */
@Parcelize
@Serializable
sealed class Screen : NavKey, Parcelable {

    @Serializable
    data class AppDetails(val packageName: String) : Screen()

    @Serializable
    data object Search : Screen()

    @Serializable
    data object About : Screen()
}
```

```kotlin
@Composable
fun NavDisplay(startDestination: NavKey) {
    val backstack = rememberNavBackStack(startDestination)

    val activity = LocalActivity.current
    fun onNavigateUp() {
        if (backstack.size == 1) activity?.finish() else backstack.removeLastOrNull()
    }

    NavDisplay(
        backStack = backstack,
        entryDecorators = listOf(
            rememberSaveableStateHolderNavEntryDecorator(),
            rememberViewModelStoreNavEntryDecorator()
        ),
        entryProvider = entryProvider {
            entry<Screen.AppDetails> { screen ->
                AppDetailsScreen(
                    packageName = screen.packageName,
                    onNavigateUp = ::onNavigateUp,
                    onNavigateToAppDetails = { packageName ->
                        backstack.add(Screen.AppDetails(packageName))
                    }
                )
            }
        }
    )
}
```

Adaptive

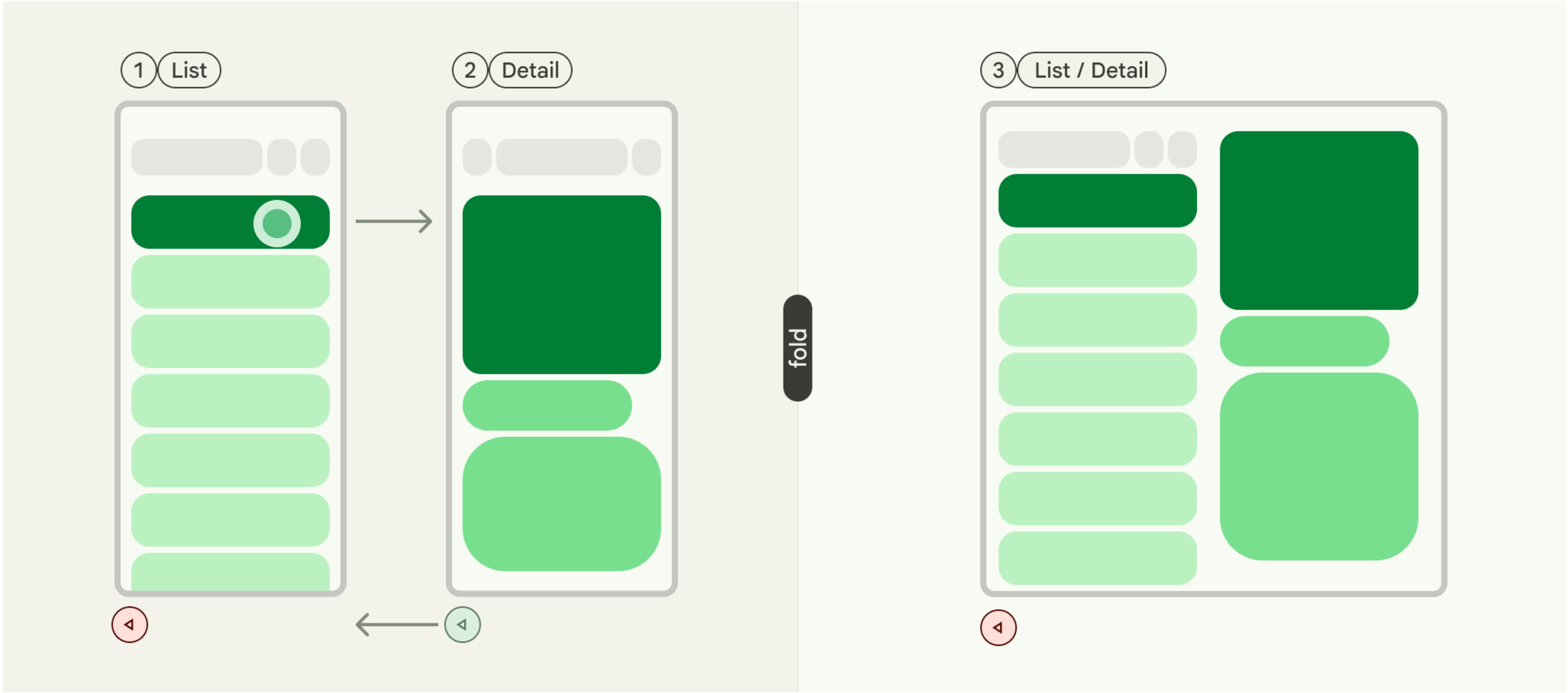
Adaptive

- Google provides 3 types of adaptive layouts for compose
- List-detail, detail-supporting, and navigation suite scaffold
- Handles their own navigation internally; a bit customisable

- Developers are also free to build their own adaptive layouts

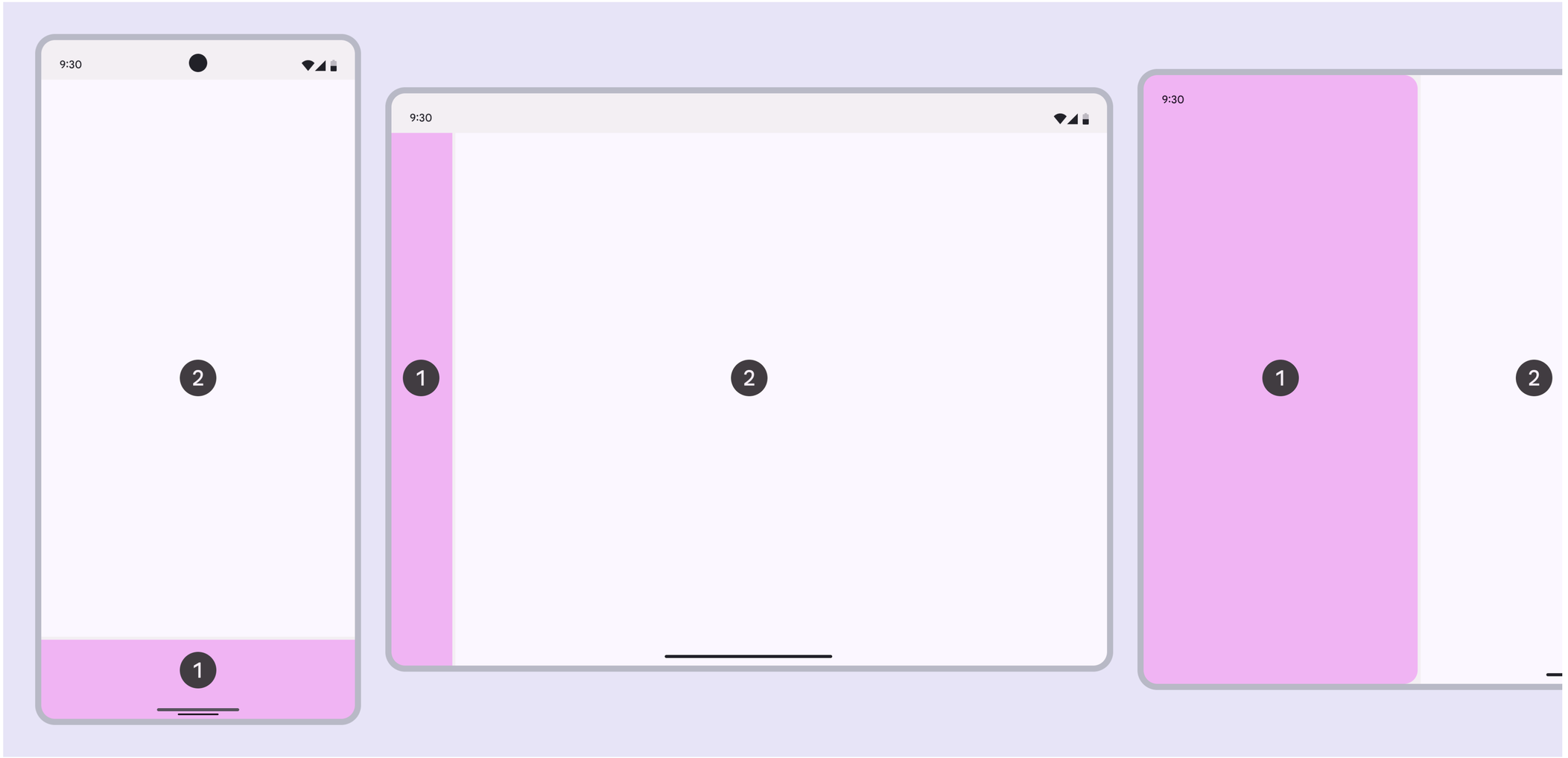
Adaptive

NavigableListDetailPaneScaffold



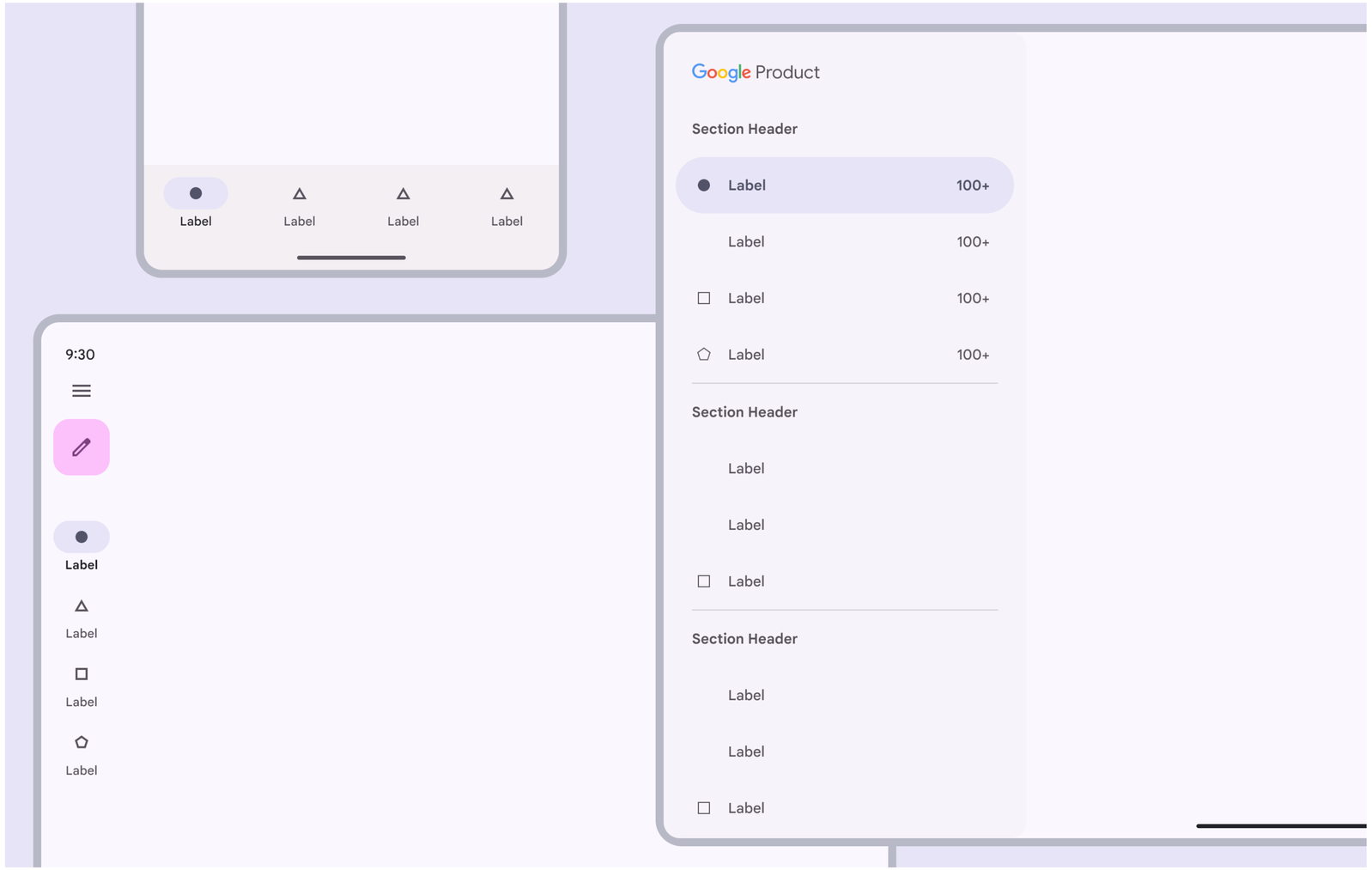
Adaptive

NavigableSupportingPaneScaffold



Adaptive

NavigationSuiteScaffold



Adaptive
Navigation3

- Navigation3 provides a new feature called Scene
- A Scene can be constructed by various nav entries
- Google provides existing scene strategies for their adaptive layouts

- Developers can also build their own custom scene strategies as required

```kotlin
import android.os.Parcelable
import androidx.navigation3.runtime.NavKey
import kotlinx.parcelize.Parcelize
import kotlinx.serialization.Serializable

/**
 * Extra destinations for app detail's screen
 *
 * All of these destinations require and show information related to an app and thus aren't part of
 * the main navigation display class.
 */
@Parcelize
@Serializable
sealed class ExtraScreen : NavKey, Parcelable {

    @Serializable
    data object More : ExtraScreen()

    @Serializable
    data class Screenshot(val index: Int) : ExtraScreen()

    @Serializable
    data object Review : ExtraScreen()
}
```

```kotlin
import androidx.compose.material3.adaptive.WindowAdaptiveInfo
import androidx.window.core.layout.WindowSizeClass

/**
 * Whether the device width is compact or not
 *
 */
val WindowAdaptiveInfo.isWindowCompact: Boolean
    get() = !windowSizeClass.isWidthAtLeastBreakpoint(WindowSizeClass.WIDTH_DP_MEDIUM_LOWER_BOUND)
```

```kotlin
@Composable
fun AppDetailsNavDisplay(
    packageName: String,
    onNavigateUp: () -> Unit,
    onNavigateToAppDetails: (packageName: String) -> Unit,
    viewModel: AppDetailsViewModel = hiltViewModel(key = packageName),
    windowAdaptiveInfo: WindowAdaptiveInfo = currentWindowAdaptiveInfo()
) {
    val startDestinations = listOfNotNull<NavKey>(
        Screen.AppDetails(packageName),
        if (windowAdaptiveInfo.isWindowCompact) ExtraScreen.More else null
    )
    val backstack = rememberNavBackStack(*startDestinations.toTypedArray())
    val supportingPaneSceneStrategy = rememberSupportingPaneSceneStrategy<NavKey>()

    fun onRequestNavigateUp() {
        if (!backstack.all { it in startDestinations }) {
            backstack.removeLastOrNull()
        } else {
            onNavigateUp()
        }
    }

    NavDisplay(
      // More logic here in next slide
    )
}
```

```
NavDisplay(
  backStack = backstack,
  entryDecorators = listOf(
      rememberSaveableStateHolderNavEntryDecorator(),
      rememberViewModelStoreNavEntryDecorator()
  ),
  sceneStrategy = supportingPaneSceneStrategy,
  entryProvider = entryProvider {
      entry<Screen.AppDetails>(metadata = SupportingPaneSceneStrategy.mainPane()) {
          AppDetailsScreen(
              packageName = packageName,
              onNavigateUp = ::onRequestNavigateUp,
              onNavigateToAppDetails = onNavigateToAppDetails,
              onNavigateToExtra = { screen -> backstack.add(screen) }
          )
      }

      entry<ExtraScreen.More>(metadata = SupportingPaneSceneStrategy.supportingPane()) {
          MoreScreen(
              packageName = packageName,
              onNavigateUp = ::onRequestNavigateUp,
              onNavigateToAppDetails = onNavigateToAppDetails
          )
      }

      entry<ExtraScreen.Review>(metadata = SupportingPaneSceneStrategy.extraPane()) {
          ReviewScreen(
              packageName = packageName,
              onNavigateUp = ::onRequestNavigateUp
          )
      }
  }
)
```

Issues

ThreePaneScaffold automatically requests focus on SearchBar within list pane

+1 1 Hotlists Mark as Duplicate

Comments (2) Dependencies (0) Duplicates (0) Blocking (0) Resources (0) Insights

Bug P3 + Add Hotlist

STATUS UPDATE No update yet.

DESCRIPTION ov...@gmail.com created issue #1 Oct 16, 2025 09:34AM

When using a NavDisplay with the ListDetailSceneStrategy, it seems that ThreePaneScaffold will requestFocus on any focusable item within the child hierarchy. In this situation, with a simple search bar, the text field gains focus automatically which is unexpected.

The provided sample project contains a copy of the the listdetail example from Nav3-recipe with adjustments to add a search bar in the list pane.

Component used: Navigation3 NavDisplay with ListDetailSceneStrategy and SearchBar (material3)

Version used:

- adaptive-navigation3-android:1.3.0-alpha01
- androidx.navigation3:navigation3-*:1.0.0-alpha11

Devices/Android versions reproduced on: Pixel 9 Pro Fold API 36.0

- Sample Project: c.f. *Navigation3Issue.zip*
- A screenrecord: c.f. *focus_searchbar_listpane_nav3_recording.gif* (Please note that once the app is launched, there is no other input happening, the search bar gains focus on its own)

focus_searchbar_listpane_nav3_recording.gif

21 MB View Download

Navigation3Issue.zip

113 KB Download

Reporter	ov...@gmail.com
Type	Bug
Priority	P3
Severity	S2
Status	New
Access	Default access View
Expanded Access	
Assignee	
Verifier	
Collaborators	
CC	an...@google.com ov...@gmail.com
Code Changes	--
Pending Code Changes	--
AOSP ID	--
ReleaseRadar	--
Estimate	--
Estimate	--

Support shared ViewModels between entries on the back stack

+1 27 Hotlists (1) Mark as Duplicate

Comments (5) Dependencies (0) Duplicates (0) Blocking (0) Resources (1) Insights

Assigned Feature Request P2 + Add Hotlist

STATUS UPDATE No update yet.

DESCRIPTION pr...@gmail.com created issue #1 May 22, 2025 08:52PM

Component used: Navigation3

Version used:

Devices/Android versions reproduced on:

If this is a bug in the library, we would appreciate if you could attach:

- Sample project to trigger the issue.
- A screenrecord or screenshots showing the issue (if UI related). how to share a viewmodel with backstack entry

@Composable fun MainApp(modifier: Modifier = Modifier) {

KoinApplication(application = { modules(androidModule) }) { PbAdminTheme { val backstack = rememberNavController()

```
Scaffold {
  NavHost(
      modifier = modifier.fillMaxSize().padding(it),
      navController = backstack,
      startDestination = Home) {
        composable<Home> {
          val viewModel = koinViewModel<AppViewModel>()

          HomePage(
              modifier = modifier.fillMaxSize(),
              viewModel = viewModel,
              navController = backstack)
        }
```

Reporter pr...@gmail.com
Type Feature Request
Priority P2
Severity S2
Status Assigned
Access Default access View

Expanded Access
Assignee jb...@google.com
Verifier
Collaborators
CC an...@google.com pr...@gmail.com

Code Changes --
Pending Code Changes --
AOSP ID --
Estimate --
Found In --

If you don't use ViewModelStoreNavLocalProvider, ViewModels aren't scoped to back stack keys

+1 3 Hotlists (1) Mark as Duplicate

Comments (1) Dependencies (0) Duplicates (0) Blocking (0) Resources (0) Insights

Assigned Bug P3 + Add Hotlist

STATUS UPDATE No update yet.

DESCRIPTION do...@google.com created issue #1 Mar 22, 2025 06:43PM

Component used: Navigation3 Version used: Snapshot build 13151678

For `ViewModel`s created using `viewModel` to be scoped to keys in your back stack, you must supply `ViewModelStoreNavLocalProvider` to `SinglePaneNavDisplay.localProviders`. Failure to do this will result in `ViewModel`s being scoped to the nearest `ViewModelStoreOwner`, usually the app's `Activity`.

The problem is that it's easy to forget to do this, and everything may seem to work fine at runtime. It's only when you start testing for state destruction that you may realize you have a problem. Even once you've spotted the problem it may not be immediately obvious what's causing it e.g. "why is my screen's state restored even though I popped it previously?"

Feedback from P4, P5, P10

COMMENTS All comments Oldest first

jb...@google.com <jb...@google.com> May 7, 2025 11:05PM

Assigned to jb...@google.com.

Add a comment.

Reporter do...@google.com
Type Bug
Priority P3
Severity S2
Status Assigned
Access Default access View

Expanded Access
Assignee jb...@google.com
Verifier
Collaborators
CC an...@google.com
do...@google.com

Code Changes --
Pending Code Changes --
AOSP ID --
Estimate --
Found In --

Provide pane-level size information

+1 2 Hotlists (1) Mark as Duplicate

Comments (6) Dependencies (0) Duplicates (1) Blocking (0) Resources (2) Insights

Feature Request P2 + Add Hotlist

STATUS UPDATE No update yet.

DESCRIPTION ch...@google.com created issue #1 Apr 5, 2024 05:18AM

Jetcaster implements a differentiated UI depending on the `WindowWidthSizeClass`.

For `Compact`, a search bar is shown on the top app bar (see attached screenshot), and for `Expanded` mode, the top app bar contains both a search bar and tabs to toggle between "Library" and "Discover".

However, depending on the screen, the pane's width might be too small resulting in the search bar being squished. The screenshot attached shows the UI rendered on a medium tablet (pane width just fine) and a pixel fold (pane too narrow). In both of these devices, the `WindowWidthSizeClass` is `Expanded` and the pane scaffold (`SupportingPaneScaffold` in this case) does not provide sizing information so that I can show the `Compact` UI when the main pane is to narrow.

A possible solution here is to provide panel-level sizing information so that its contents can further be customized.

Thu Apr 04 2024 14:05:19 GMT-0700 (Pacific Daylight Time).png

1.2 MB View Download

medium tablet landscape.png

935 KB View Download

pixel-fold.png

673 KB View Download

Reporter ch...@google.com
Type Feature Request
Priority P2
Severity S2
Status New
Access Default access View

Expanded Access
Assignee
Verifier
Collaborators
CC
ch...@google.com
co...@google.com
ia...@google.com
va...@google.com

Code Changes --
Pending Code Changes --
AOSP ID --

Thank You!