نکته بسیار مهم: snapshotFlow یک State کامپوزی را به Flow تبدیل می کند.
یعنی یک mutableStateOf را به StateFlow تبدیل می کند

مشکلات عدم استفاده از snapshotFlow در Jetpack Compose

در حالی که `snapshotFlow` یک ابزار مفید برای ایجاد جریان داده از منابع داده ایستا در Jetpack Compose است، عدم استفاده از آن در برخی موارد می‌تواند منجر به **مشکلات** و **رفتار غیرمنتظره** در برنامه شما شود.

مشکلات رایج:

- **محاسبات غیرضروری :**اگر از `snapshotFlow` استفاده نکنید، ممکن است **محاسبات غیرضروری** برای **منابع داده ایستا** انجام شود. این امر می‌تواند **عملکرد برنامه** را به خصوص در دستگاه‌های ضعیف‌تر **کاهش دهد.**
- **تغییرات غیرمنتظره :**اگر از `snapshotFlow` برای **منابع داده پویا** استفاده کنید، ممکن است **تغییرات غیرمنتظره‌ای** در جریان داده مشاهده کنید. این امر می‌تواند منجر به **رفتار غیرقابل پیش‌بینی** در برنامه شما و **اشکالات** شود.
- **نشت حافظه :**اگر از `snapshotFlow` به درستی استفاده نکنید، ممکن است **حافظه نشت** کند. این امر می‌تواند منجر به **کاهش عملکرد** و **مشکلات ثبات** در برنامه شما شود.

مثال:

```
val detailsSheetState =
rememberViraBottomSheetState(skipPartiallyExpanded = false)
LaunchedEffect(detailsSheetState.currentValue) {
    snapshotFlow { detailsSheetState.currentValue }
        .collect { currentValue ->
            bottomSheetCurrentValue = currentValue
        }
}
```

تحلیل کد به صورت بخش به بخش:

1.تعریف حالت شیت پایین:(BottomSheet)

- val detailsSheetState = rememberViraBottomSheetState(skipPartiallyExpanded = false):
  - این خط یک حالت برای کنترل شیت پایین با استفاده از کتابخانه Vira تعریف می‌کند.
  - skipPartiallyExpanded = false به این معنی است که شیت می‌تواند در حالت نیمه باز نیز قرار گیرد.

2.اجرای یک اثر جانبی (Side Effect) با:LaunchedEffect

- LaunchedEffect(detailsSheetState.currentValue):

- این بلوک یک اثر جانبی را اجرا می‌کند که به تغییرات در detailsSheetState.currentValueحساس است .هر زمان که مقدار این حالت تغییر کند، کد داخل این بلوک اجرا می‌شود.

3.استفاده از:snapshotFlow

- snapshotFlow { detailsSheetState.currentValue }:
  - snapshotFlowیک تابع در Compose است که یک جریان (Flow) ایجاد می‌کند .این جریان هر زمان که مقدار منبع (در اینجا detailsSheetState.currentValue)تغییر کند، مقدار جدید را منتشر می‌کند.
  - به عبارت دیگر، snapshotFlowتغییرات در یک مقدار قابل مشاهده را به یک جریان تبدیل می‌کند.

4.جمع‌آوری مقادیر از جریان:

- .collect { currentValue -> ... }:
  - این بخش باعث می‌شود که هر زمان مقدار جدیدی از جریان snapshotFlowمنتشر شود، یک lambda اجرا شود.
  - در داخل این lambda ، مقدار جدید در متغیر currentValueقرار می‌گیرد.

5.به‌روزرسانی مقدار جهانی:

- bottomSheetCurrentValue = currentValue:
  - در نهایت، مقدار به‌روز شده از جریان در یک متغیر جهانی به نام bottomSheetCurrentValueذخیره می‌شود.

خلاصه عملکرد کد:

این قطعه کد به طور مداوم تغییرات در حالت شیت پایین را رصد می‌کند و هر زمان که حالت تغییر کرد، مقدار جدید را در یک متغیر جهانی ذخیره می‌کند .این کار به شما اجازه می‌دهد که در جاهای دیگر برنامه‌تان به این مقدار دسترسی داشته باشید و بر اساس آن اقدام کنید.

نتیجه:

`snapshotFlow`یک ابزار قدرتمند برای **ایجاد جریان داده کارآمد و قابل پیش‌بینی** از منابع داده ایستا در Jetpack Composeاست. استفاده از `snapshotFlow`به جای محاسبات مستقیم می‌تواند به **بهبود عملکرد، کاهش نشت حافظه و جلوگیری از اشکالات** کمک کند.

نکته :در حالی که `snapshotFlow`مزایای زیادی دارد، مهم است که از آن به درستی و فقط برای منابع داده ایستا استفاده کنید.