Automated Testing

Part 3: Navigation Tests

What To Test

- Navigation between two screens
  - Trigger action(s) in the source screen
  - Confirm the destination screen is “visible”

Libraries Setup

```
dependencies {
    // libraries needed by androidTest files
    androidTestImplementation(“androidx.navigation:navigation-testing:2.8.6”)

    // import assertXXXX helper functions
    androidTestImplementation(kotlin(“test”))
}
```


Isolate Navigation into Own Screen

```
YourAppTheme {
    val vc = rememberNavController()
    NavHost(navController = vc) {
        composable<DestinationOne> {
            ScreenOne(____)
        }
        composable<DestinationTwo> {
            ScreenTwo(____)
        }
    }
}
```

```
YourAppTheme {
    val vc = rememberNavController()
    AppNavigationScreen(navCtrl = nc)
}

@Composable
fun AppNavigationScreen(navCtrl: NavHostController) {
    NavHost(navController = navCtrl) {
        composable<DestinationOne> {
            ScreenOne(____)
        }
        composable<DestinationTwo> {
            ScreenTwo(____)
        }
    }
}
```

Test Class Setup

```
@RunWith(AndroidJUnit4::class)
class NavigationTest {
   @get:Rule val composeTestRule = createComposeRule()
   private lateinit var navController: TestNavHostController
   private lateinit var fakeViewModel: AppViewModel
   @Before
   fun setup() {
       composeTestRule.setContent {
           navController = TestNavHostController(LocalContext.current)
           navController.navigatorProvider.addNavigator(ComposeNavigator())
           fakeViewModel = mockk<AppViewModel>()
           AppNavigationScreen(navCtrl = navController, viewModel = fakeViewModel)
       }
   }
}
```


Verify “Landing” Screen

```
@RunWith(AndroidJUnit4::class)
class NavigationTest {
    @Before
    fun setup() { /* details hidden */ }

    @Test
    fun landingScreenIsPurchaseScreen() {
        assertTrue {
            navController.currentDestination?.hasRoute<Route.Purchase>() == true
        }
    }

}
```

Verify Navigation to Destination

```
@RunWith(AndroidJUnit4::class)
class NavigationTest {
    @Before
    fun setup() { /* details hidden */ }

    @Test fun settingButtonTakesToSettingScreen() {
        composeTestRule.apply {
            onNodeWithTag("settingsbutton").performClick()
        }
        assertTrue {
           navController.currentDestination?.hasRoute<Route.Preference>() == true
        }
    }
}
```


Verify: “Confirm” Navigate Back to “Origin”

```
@Test
fun tappingConfirmInSettingScreenShouldNavigateBackToPurchaseScreen() {
    composeTestRule.runOnUiThread {
        navController.navigate(Route.Preference)
    }
    every {
        fakeViewModel.setColor(any<Float>(), any<Float>(), any<Float>())
    } just runs
    composeTestRule.apply {
        onNodeWithText(“Confirm”).performClick()
    }
    assertTrue {
        navController.currentDestination?.hasRoute<Route.Purchase>() == true
    }
}
```

Verify: “Cancel” Navigate Back to “Origin”

```
@Test
fun tappingCancelInSettingScreenShouldNavigateBackToPurchaseScreen() {
    composeTestRule.runOnUiThread {
        navController.navigate(Route.Preference)
    }
    // every {
    //     fakeViewModel.setColor(any<Float>(), any<Float>(), any<Float>())
    // } just runs
    composeTestRule.apply {
        onNodeWithText(“Cancel”).performClick()
    }
    assertTrue {
        navController.currentDestination?.hasRoute<Route.Purchase>() == true
    }
}
```