# Testing (Complete Guide)

# Testing (Complete Guide)

This application is covered by two complementary test layers:

1. \*\*Repository unit tests\*\* (JVM) — validate data & business logic quickly and deterministically.   
2. \*\*Jetpack Compose UI tests\*\* (instrumented) — validate screen behavior and user flows on a device/emulator.

Everything below gives you the exact environment, step-by-step processes, CI setup, how to view results, common pitfalls, and how we fixed them.

---

## Environment (Project Baseline)

- \*\*Gradle wrapper:\*\* 8.11.1   
- \*\*Android Gradle Plugin (AGP):\*\* 8.5.2   
- \*\*Kotlin:\*\* 2.0.20   
- \*\*Java:\*\* 17   
- \*\*Compose BOM:\*\* 2024.06.00   
- \*\*compileSdk / targetSdk:\*\* 34   
- \*\*minSdk:\*\* 24

> Recommended local IDE: \*\*Android Studio Koala (2024.1.1+)\*\* with JDK 17.

---

## Local Setup (One-time)

1) Install:  
- Android Studio Koala (or newer)  
- Android SDK Platform \*\*34\*\* and Build-Tools \*\*34.x\*\*  
- JDK \*\*17\*\* (the AS embedded JDK is fine)

2) Open project in Android Studio and \*\*Sync Gradle\*\*.

3) Create an emulator (for UI tests):  
- Device: \*\*Pixel 4a\*\* (or any x86\_64)  
- System image: \*\*Android 14 (API 34)\*\*

---

## Dependencies (Verify)

In `app/build.gradle.kts`, confirm:

```kotlin  
android {  
 defaultConfig {  
 testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
 }  
 testOptions { animationsDisabled = true } // avoids flakiness  
}

dependencies {  
 // Unit tests (JVM)  
 testImplementation("junit:junit:4.13.2")  
 testImplementation("org.jetbrains.kotlinx:kotlinx-coroutines-test:1.8.1")  
 testImplementation("app.cash.turbine:turbine:1.1.0")

// Compose UI tests (instrumented)  
 androidTestImplementation(platform("androidx.compose:compose-bom:2024.06.00"))  
 androidTestImplementation("androidx.compose.ui:ui-test-junit4")  
 debugImplementation("androidx.compose.ui:ui-test-manifest")

// AndroidX test helpers  
 androidTestImplementation("androidx.test.ext:junit:1.1.5")  
 androidTestImplementation("androidx.test:rules:1.5.0")  
}  
```

---

## Repository Layout for Tests

```  
app/  
 src/  
 test/ # JVM unit tests (repositories / pure Kotlin logic)  
 java/com/example/communitypolls/...  
 androidTest/ # Instrumented Compose UI tests  
 java/com/example/communitypolls/ui/  
 SignInScreenTest.kt  
 SignUpScreenTest.kt  
 PollListScreenTest.kt  
 PollEditorScreenTest.kt  
```

---

## How to Run Tests

### Android Studio  
- \*\*Unit tests:\*\* Right-click `app/src/test` → \*\*Run\*\*   
- \*\*UI tests:\*\* Start an emulator → Right-click `app/src/androidTest` → \*\*Run\*\*

### Command line  
```bash  
# Unit tests (fast, JVM)  
./gradlew test

# UI tests (need an emulator or device)  
./gradlew connectedAndroidTest

# Run a single UI test class  
./gradlew connectedAndroidTest -Pandroid.testInstrumentationRunnerArguments.class=com.example.communitypolls.ui.SignInScreenTest

# Run a single UI test method  
./gradlew connectedAndroidTest -Pandroid.testInstrumentationRunnerArguments.class=com.example.communitypolls.ui.SignInScreenTest#signin\_callsOnSubmit\_withEnteredCredentials  
```

### Where to Find Reports (HTML)  
- \*\*Unit:\*\* `app/build/reports/tests/testDebugUnitTest/index.html`   
- \*\*UI:\*\* `app/build/reports/androidTests/connected/index.html`

\*(These are generated on each run; don’t commit them.)\*

---

## What Each Test Covers

### Repository Unit Tests (JVM)  
- Happy & error paths for repository methods  
- Mapping/merging across sources  
- Coroutine & Flow behavior (first emission / subsequent updates)  
- Side effects (e.g., DAO upserts, cache invalidation)

\*\*Pattern\*\*  
```kotlin  
@Test  
fun fetchPolls\_readsCache\_thenUpdatesRemote() = runTest {  
 val dao = FakePollDao(initial = listOf(/\*...\*/))  
 val api = FakePollApi(response = listOf(/\*...\*/))  
 val repo = PollRepository(dao, api)

val polls = repo.getPolls()  
 assertEquals(/\* expected \*/, polls)  
 assertTrue(dao.wasUpsertCalled)  
}  
```

### Compose UI Tests (Instrumented)

- \*\*SignInScreenTest\*\*  
 - Inputs Email/Password, taps \*\*Continue\*\* → `onSubmit(email, password)` called  
 - Loading disables button; error text renders

- \*\*SignUpScreenTest\*\*  
 - Inputs Email/Password/Display Name, taps \*\*Create account\*\*  
 - Loading disables button; error text renders

- \*\*PollListScreenTest\*\*  
 - Loading message  
 - Error + \*\*Refresh\*\* → `onRetry`  
 - List renders; item tap → `onPollClick`  
 - Admin actions (\*\*Edit/Delete\*\*) visible when enabled; callbacks fire

- \*\*PollEditorScreenTest\*\*  
 - Title & Description updates (resilient selectors)  
 - Option \*\*Text\*\* updates (option \*\*ID\*\* asserted only if editable)  
 - \*\*Add option\*\*, \*\*Active\*\* toggle, \*\*close preset\*\* selection (e.g., “24h”)  
 - \*\*Save\*\* shows “Saving…” and is disabled while loading

\*\*Selector Techniques Used\*\*  
- Wildcard import for stability:  
 ```kotlin  
 import androidx.compose.ui.test.\*  
 import androidx.compose.ui.test.junit4.createComposeRule  
 ```  
- `useUnmergedTree = true` for `TextField` queries (often required in Compose)  
- Label-based field matcher (descendant/sibling/self):  
 ```kotlin  
 fun fieldLabeled(label: String) =  
 hasSetTextAction() and (  
 hasAnyDescendant(hasText(label, ignoreCase = true, substring = true)) or  
 hasAnySibling(hasText(label, ignoreCase = true, substring = true)) or  
 hasText(label, ignoreCase = true, substring = true)  
 )  
 ```  
- Clickable container matcher (Button/Chip/etc.) that contains a text label  
- `performTextReplacement("…")` for consistent text input across versions  
- Prefer \*\*“button disabled while loading”\*\* over progress bar semantics (less brittle)

---

## Full Step-by-Step Process (Repeatable)

1. \*\*Run unit tests first\*\*  
 ```bash  
 ./gradlew test  
 ```  
 - Fix logic failures quickly (no emulator needed).

2. \*\*Run UI tests with emulator\*\*  
 ```bash  
 ./gradlew connectedAndroidTest  
 ```  
 - If selectors fail, use `useUnmergedTree = true` for `TextField`s and semantics-based matchers.

3. \*\*Open reports\*\*  
 - Unit: `app/build/reports/tests/testDebugUnitTest/index.html`  
 - UI: `app/build/reports/androidTests/connected/index.html`

4. \*\*Commit with tests\*\*  
 ```bash  
 git add .  
 git commit -m "feat/tests: add/adjust tests for <feature>"  
 git push  
 ```

---

## CI: GitHub Actions (Automated Runs + Artifacts)

Create `.github/workflows/android-tests.yml`:

```yaml  
name: Android Tests

on:  
 push:  
 branches: [ "\*\*" ]  
 pull\_request:  
 branches: [ "\*\*" ]  
 workflow\_dispatch:

concurrency:  
 group: android-tests-${{ github.ref }}  
 cancel-in-progress: true

jobs:  
 unit-tests:  
 runs-on: ubuntu-latest  
 timeout-minutes: 20  
 steps:  
 - uses: actions/checkout@v4  
 - uses: gradle/wrapper-validation-action@v2  
 - uses: actions/setup-java@v4  
 with:  
 distribution: temurin  
 java-version: 17  
 cache: gradle  
 - name: Unit tests  
 run: ./gradlew test --stacktrace --no-daemon  
 - name: Upload unit test report  
 if: always()  
 uses: actions/upload-artifact@v4  
 with:  
 name: unit-test-report  
 path: app/build/reports/tests/testDebugUnitTest

ui-tests:  
 runs-on: ubuntu-latest  
 needs: unit-tests  
 timeout-minutes: 45  
 steps:  
 - uses: actions/checkout@v4  
 - uses: actions/setup-java@v4  
 with:  
 distribution: temurin  
 java-version: 17  
 cache: gradle  
 - uses: android-actions/setup-android@v3  
 - name: Run instrumented UI tests (API 34)  
 uses: reactivecircus/android-emulator-runner@v2  
 with:  
 api-level: 34  
 arch: x86\_64  
 profile: Pixel 4a  
 emulator-options: -no-snapshot -no-window -gpu swiftshader\_indirect  
 script: ./gradlew connectedDebugAndroidTest --stacktrace --no-daemon  
 - name: Upload UI test report  
 if: always()  
 uses: actions/upload-artifact@v4  
 with:  
 name: ui-test-report  
 path: app/build/reports/androidTests/connected  
```

\*\*Badge (optional)\*\*  
```markdown  
![Android Tests](https://github.com/<org>/<repo>/actions/workflows/android-tests.yml/badge.svg)  
```

---

## Results & History

- \*\*Attempt #1 (initial run):\*\* Several UI tests failed due to selector issues and Compose version semantics. Key problems we observed and fixed:  
 - Text fields only matched in the \*\*unmerged\*\* semantics tree → fixed with `useUnmergedTree = true`.  
 - Progress bar matchers (`hasProgressBarRangeInfo`) varied by version → replaced with \*\*“button disabled while loading”\*\* assertion.  
 - Button text nested in child `Text` → matched the \*\*clickable container\*\* instead of the inner label.  
 - Option IDs sometimes read-only → assert ID changes \*\*only if\*\* the callback fires; always assert \*\*option text\*\* changes.

- \*\*Attempt #2 (final run):\*\* All tests passed locally and under CI.  
 - Command: `./gradlew test connectedAndroidTest`  
 - \*\*Outcome:\*\* ✅ \*\*All tests successful\*\*  
 - Reports:  
 - Unit: `app/build/reports/tests/testDebugUnitTest/index.html`  
 - UI: `app/build/reports/androidTests/connected/index.html`

---

## Troubleshooting (Fast Fixes)

- \*\*`onNode` / `onAllNodes` unresolved:\*\*   
 Use wildcard import: `import androidx.compose.ui.test.\*`

- \*\*“Are you missing `useUnmergedTree`?” error:\*\*   
 Add `useUnmergedTree = true` to `onNode` / `onAllNodes` for `TextField`s.

- \*\*Progress semantics mismatch:\*\*   
 Don’t assert progress bars; assert \*\*disabled buttons\*\* while loading (or add `testTag("loading")` to the indicator).

- \*\*Typing doesn’t update state:\*\*   
 Use `performTextReplacement("…")` and semantics-driven selectors (labels & siblings). As a last resort, iterate over `hasSetTextAction()` nodes and stop when the captured callback value changes.

- \*\*Option IDs not changing:\*\*   
 IDs may be read-only. Only assert ID changes if the callback fired; always assert \*\*option text\*\* changes.

---

## Adding New Tests (Recipes)

\*\*Repository\*\*  
1. Create fakes for DAO/API/clock.  
2. `runTest { … }` and call the repository method.  
3. Assert return \*\*and\*\* side effects (DAO writes, cache).

\*\*UI\*\*  
1. Capture callback state (`var clicked = false`).  
2. `setContent { Screen(state, onClick = { clicked = true }) }`  
3. Use label matchers + `useUnmergedTree = true`.  
4. Interact & assert visible state and callback values.

---

## Conventions

- Test names: `action\_expectedResult\_condition`  
- Semantics-first queries; avoid fragile indexes  
- Keep tests deterministic (no live network or real storage)  
- If you change a label (e.g., “Continue” → “Sign in”), update the matcher set once