Technical Requirements – A New Morning

Challenge is accessible on Memberspot: https://pl-coding.mymemberspot.io/library/jx3b7Qik9ip5qpNI8IF2/2BiP9k9ZAdvPhLG5wf7P/BwLxTncB77r7ewNdhz08/details

🎭 Scenario

Imagine a small app that helps you start your day with a clean slate. You open it in the morning, tap a single button, and that's it - yesterday is gone, today begins. No extra steps, no way back, just a fresh start and a short confirmation that everything worked as expected. This challenge focuses on making sure that this simple flow behaves consistently, even when the app is recreated or restarted.

👉 GitHub Repository

The implementation for this mini-challenge is provided in a GitHub repository: https://github.com/PL-Coding-GmbH/Campus-SpringValidation/tree/challenge/new-morning

The repository contains multiple branches. Each branch corresponds to a **separate mini-challenge** in this series.

Clone the repository, switch to the branch for the current challenge, and work with the existing code. Your task is to focus on writing tests - the implementation itself should not be modified.

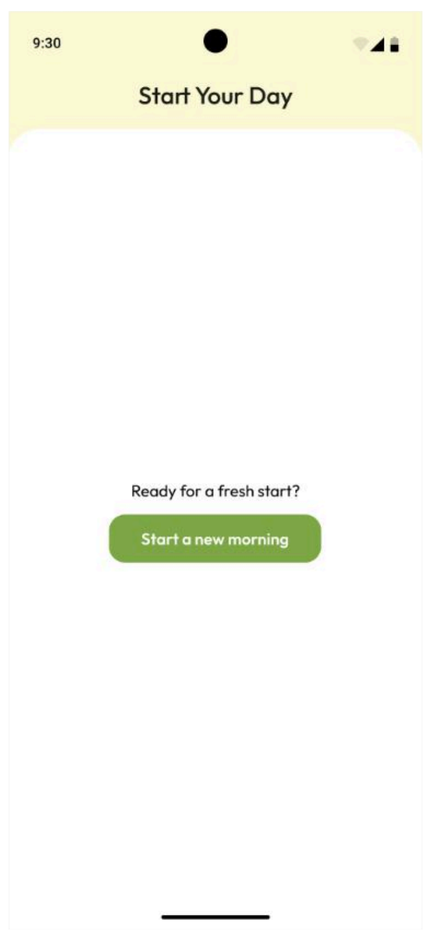
🎯 Feature Goal

The goal of this challenge is to validate a reset-style navigation flow across lifecycle events. You must ensure correct screen transitions, proper back stack clearing, correct handling of one-time UI events, and accurate start destination based on persisted state.

⚙️ App Behavior Overview

The app consists of a simple reset flow with two screens that represent starting a new day.

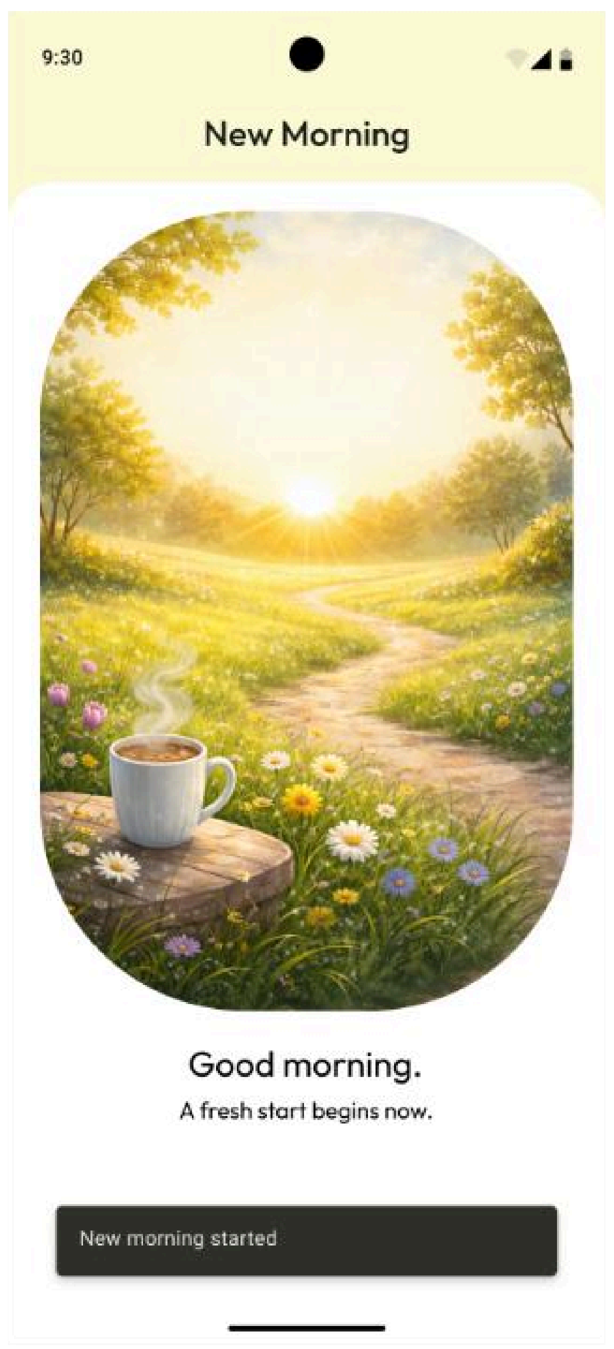
Start Screen



- This is the initial screen shown when the app is launched.
- The screen displays:
  - the title *"Start Your Day"*;
  - a short prompt encouraging the user to begin;
  - a primary action button *"Start a new morning"*.
- No additional actions or states are available on this screen.

Fresh Start Screen

- Tapping "Start a new morning" navigates the user to the **Fresh Start Screen**.
- The screen displays:
  - a large illustrative image;
  - the headline *"Good morning."*;
  - the subheadline *"A fresh start begins now."*.
- After navigation, a confirmation Snackbar with the message *"New morning started."* is shown.



Navigation & Lifecycle Behavior

- After navigating to the Fresh Start Screen:
  - the Start Screen is removed from the back stack;
  - pressing the system Back button **does not** return to the Start Screen.
- The Snackbar is a **one-time event** and should not be shown again after screen recreation.
- If the Activity is relaunched after completing the navigation:
  - the app opens **directly** on the Fresh Start Screen;
  - the Snackbar is **not shown** again.

🧪 Validation Tests Requirements

This challenge includes two types of tests:

- **Instrumentation UI tests** — to verify navigation flow, back stack behavior, and one-time UI events.
- **ViewModel unit tests** — to verify start destination logic based on persisted state.

Each test must use the exact test name specified below and clearly verify the expected behavior.

1️⃣ Initial State - Start Screen (UI Test)

Test name:

- startScreen_isDisplayed_onAppLaunch

Verify:

- when the app is launched:
  - the screen title *"Start Your Day"* is displayed;
  - the text *"Ready for a fresh start?"* is displayed;
  - the primary button *"Start a new morning"* is visible and enabled;

2️⃣ Navigation to Fresh Start Screen (UI Test)

Test name:

- navigatesToFreshStartScreen_afterReset

Action:

- Tap "*Start a new morning*".

Verify:

- navigation to the Fresh Start Screen **occurs**;
- the headline *“Good morning.”* is **displayed**;
- the subhead line *“A fresh start begins now.”* is **displayed**;
- the main illustrative image is **visible**;
- a Snackbar with the message *“New morning started.”* is **shown** after navigation.

3️⃣ Back Stack Is Cleared After Reset (UI Test)

Test name:

- backStack_isCleared_afterReset

Setup:

- Navigate to the Fresh Start Screen.

Action:

- Press the system Back button.

Verify:

- The Start Screen is not **displayed**.
- The Fresh Start Screen is not **displayed**.

4️⃣ Snackbar Is Not Repeated After Screen Recreation (UI Test)

Test name:

- snackbar_isNotShown_again_afterRecreation

Setup:

- Navigate to the Fresh Start Screen and ensure the Snackbar has been shown once.

Action:

- Recreate the screen (e.g., simulate configuration change).

Verify:

- The Fresh Start Screen is still **displayed**.
- The headline and subheadline remain **visible**.
- The Snackbar is **not shown again**.

5️⃣ App Starts From Start Screen When No State Is Saved (Unit Test)

Test name:

- startDestination_isStartScreen_whenNoStateSaved

Setup:

- Ensure no persisted state exists.

Action:

- Create a new ViewModel instance (cold start simulation).

Verify:

- The computed start destination is Start Screen.

6️⃣ App Starts From Fresh Start Screen When Reset Was Completed (Unit Test)

Test name:

- startDestination_isFreshStartScreen_whenMorningAlreadyStarted

Setup:

- Persist state indicating that the morning was already started.

Action:

- Create a new ViewModel instance (app restart simulation).

Verify:

- The computed start destination is Fresh Start Screen.

ℹ️ Notes

- Tests must be written at the appropriate layer:
  - UI and navigation behavior → UI / instrumentation tests.
  - Start destination and state restoration logic → ViewModel unit tests.
- Test names must be used **exactly as specified**.
- Snackbar is treated as a **one-time UI event**, not as part of persistent UI state.
- Back stack behavior must be verified **through navigation behavior**, not by inspecting internal navigation state.

🤔 What's Allowed?

- Standard Android / Jetpack libraries.
- Any testing approach that verifies validation logic.

⚠️ What's not important

- Writing additional tests beyond those specified in the requirements.
- Covering extra edge cases not mentioned in the challenge.

🔗 Useful Links for This Challenge

- Test your Compose layout
- Compose testing common patterns
- Testing in Jetpack Compose Codelab
- Testing APIs
- Fundamentals of testing Android apps
- Testing Basics
- The Ultimate Guide to Android Testing

🏆 Submission & Rewards

- Successfully submitting this challenge via the **/submit-challenge** command on Discord grants you **200 XP**.
- Your submission must include:
  a. A **Gist link** with your implementation.
  b. A **screenshot** of the test run results showing:
    - which tests passed,
    - which tests failed,
    - and the names of the executed tests.

💡 **Note:**

Some tests in this challenge are **expected to fail** due to intentionally incorrect behavior in the app implementation.

Your goal is to write correct tests, not to make all tests pass.

📩 How to Submit a Mini-Challenge

- In any Discord channel, type **/submit-challenge**.
- Attach your screen recording demonstrating the implementation according to the challenge requirements.
- Supported formats: MP4, MOV, AVI, MKV, WEBM, PNG, JPEG, JPG, GIF.
- The total file size must **not exceed** 50 MB.

- If additional materials are required (e.g. screenshots), attach up to 4 additional image files in the command pop-up before submitting.
- Press **Enter** to send the files.
- In the bot flow, select **Mini-Challenge**.
- Choose the month this mini-challenge belongs to (each month includes five mini-challenges).
- Select the exact **challenge name** you are submitting.
- Submit challenge.