## QLC‑3: Logging Mood Selections (TDD with Mocks)

**Objective**

Introduce a pluggable logging mechanism into the Mood Tag Selector app by defining a MoodLogger interface and injecting it into MoodAnalyticsManager. Use Test‑Driven Development to ensure that every mood the user selects is passed, in order, to the logger, keeping analytics logic decoupled from any particular output and fully verifiable via mocks.

**Scenario**

Your MoodAnalyticsManager must now record every mood the user selects. Rather than writing output directly, it should delegate logging to a separate component.

### Goals

1. **Define a MoodLogger interface**
   * Single responsibility: handle the act of recording or dispatching a log entry.
   * Method signature: takes one string parameter—the mood name.
2. **Wire the logger into MoodAnalyticsManager**
   * Add a constructor parameter of type MoodLogger.
   * Remove any direct logging calls (e.g. println, file writes) from the manager itself.
3. **Log every mood selection**
   * Whenever processMoods() is called, fetch the list of moods from the selector.  
     For each mood in that list, invoke the logger exactly once, in order.
4. **Verify via mock**
   * Use a Mockito‑Kotlin or MockK mock for MoodLogger.  
     Test interaction: ensure the mock’s log method is called with the right arguments, in sequence, and the correct number of times.
5. **Adhere to TDD**
   * **Red**: write the test first—it should fail because no logging exists yet.
   * **Green**: add only the code required to make it pass.
   * **Refactor**: clean up any duplication or clarify naming without changing behavior.

## QLC‑3.1: Timestamped Logging

**Objective:** Augment MoodLogger so that it timestamps each entry:

**Red (failing test):**

* Mock a fixed Clock to return a known Instant.
* Stub selector to return ["Happy"].
* Verify logger.log("Happy", fixedInstant).

**Green (implement):**

* Inject Clock into MoodAnalyticsManager.
* Call logger.log(mood, clock.instant()).

**Refactor:**

* Extract a BatchLogger decorator that adds timestamps around any existing MoodLogger.

## QLC‑3.2: Refactoring & Cleanup

Once QLC‑3.1 is green, carve out a refactor ticket:

1. **Extract Interface for Selector + Logger**
   * Move MoodLogger and MoodTagSelector into their own package or module.
2. **Reduce Duplication**
   * If you added multiple tests for empty / error / flag cases, consolidate common setup into a base test class.
3. **Rename Methods for Clarity**
   * Perhaps MoodAnalyticsManager.processMoods() → recordSelectedMoods() or captureSelections().
4. **Improve Test Names**
   * Make names more readable, maybe taking influence from BDD, “testPositiveRoute”, “testingMethodNameWithPositiveNumbers”
   * fun \when no moods selected, logger is not called`() { … }`
   * fun \logs each mood in selection order`() { … }`
5. **Tidy Up Working Folders**
   * Did we put everything in the right place?
   * Could we improve the working structure for maintainability?