SimpleLife: Project Technical Report

Developers: Danila Morozov, Jiancai Hou

Date: November 26, 2025

Repository: https://github.com/Danmoroz96/Simplelife_kotlin_public/tree/master

1. Executive Summary

SimpleLife is a modern Android application designed to help users take control of their everyday life by tracking expenses and habits. Built with **Kotlin** and **Jetpack Compose**, it demonstrates clean architecture and modern Android development practices.

2. Architecture & Design Pattern

The application follows the **MVVM (Model-View-ViewModel)** architectural pattern, ensuring a clean separation of concerns and testability.

- **UI Layer (Jetpack Compose):** A reactive UI that observes state changes. It contains no business logic.
- **Presentation Layer (ViewModel):** Holds the application state using StateFlow. It processes user actions (e.g., "Add Expense") and communicates with the Repository.
- **Data Layer (Repository):** The single source of truth. It manages data synchronization between:
    - **Local:** Room Database (SQLite) for offline persistence.
    - **Remote:** Firebase Realtime Database for cloud backup.

3. Technology Stack & Tools

- **Language:** Kotlin (v1.9.20).
- **User Interface:** Jetpack Compose (Material Design 3).
- **Local Storage:** Room Database (with KSP annotation processing).
- **Cloud Storage:** Firebase Realtime Database.
- **Concurrency:** Kotlin Coroutines & Flow for asynchronous operations.
- **Visualization:** MPAndroidChart (integrated via AndroidView interoperability).

- **Build System:** Gradle (Kotlin DSL).

4. Technical Challenges & Solutions

During development, three primary challenges were encountered and resolved:

- **Dependency Management:** Initial conflicts occurred between Kotlin versions and the Compose Compiler (e.g., version 1.5.1 vs 1.9.20).
    - *Solution:* Aligned the kotlinCompilerExtensionVersion in build.gradle.kts to match the project's Kotlin version.
- **Legacy Library Interoperability:** MPAndroidChart is an older View-based library, while the app uses Jetpack Compose.
    - *Solution:* implemented the AndroidView composable wrapper to bridge the gap, allowing the legacy chart to render and animate within the modern Compose UI.
- **Hybrid Data Sync:** Ensuring data saved to both local storage and the cloud simultaneously without blocking the main thread.
    - *Solution:* Utilized CoroutineScope(Dispatchers.IO) in the Repository to handle Firebase network calls asynchronously while Room handled local data immediately.

5. Key Lessons Learned

1. **Modern UI Paradigm:** Transitioning from XML to **Jetpack Compose** significantly reduced code and made UI state management (via collectAsState) intuitive.
2. **Scalable Architecture:** Using MVVM meant that adding the "Mood Tracker" feature at the end of the project was trivial; the logic layer required no changes to the existing Expense or Habit code.
3. **Build Configuration:** Understanding the relationship between Gradle plugins (KSP, Google Services) and the app module is critical for integrating third-party SDKs like Firebase.