# Final Project Report: Momentum

CS 501: Mobile Application Development

Mussie Abraham, Labeeb Alam, Afzal Khan, Rashfiqur Rahman

May 8, 2025

## Introduction

Momentum is a multi-platform Android application designed to help users build and maintain positive habits through goal setting, progress tracking, and motivational reinforcement. Built with Kotlin and Jetpack Compose, the app features adaptive layouts for phones and WearOS devices, integrates with the ZenQuotes API for daily inspiration, and utilizes Room Database for local data persistence. This report details the app's features, architectural decisions, technical challenges, and user feedback gathered during testing.

## App Features and Design Decisions

The application's core functionality revolves around habit management with a focus on user engagement. Users can create and edit customizable habits with frequency settings, track daily completions through a color-coded calendar interface, and receive motivational quotes via the ZenQuotes API. The home screen displays active habits with completion toggle functionality, while the history section provides visual progress tracking through weekly and monthly calendar views.

Key design decisions prioritized usability across different form factors. For phones, the interface adapts dynamically between portrait and landscape orientations. Portrait mode uses a bottom navigation bar while landscape switches to a side navigation rail for better use of screen space. On WearOS, the interface was simplified to show critical information at a glance, with swipe gestures for navigation. The UI follows Material Design 3 guidelines with customizable dark/light themes, ensuring accessibility through proper contrast ratios and scalable typography.

Integrating the step counter sensor enhances habit tracking by enabling users to connect physical activity with their daily goals. Step data is presented through clean, minimal visualizations on both phone and WearOS interfaces, contributing to a seamless multi-device experience. The use of Room Database over Firebase was a deliberate choice to prioritize reliable offline access and low-latency local reads/writes, both critical for responsive habit management.

## Architecture and Technology Stack

The application follows a layered MVVM (Model-View-ViewModel) architecture that cleanly separates concerns between data management, business logic, and presentation layers. At the data layer, Room Database provides local persistence with two primary tables - one for habit definitions and another for completion records. The repository pattern abstracts database operations, exposing suspend functions for CRUD operations while handling thread management through Kotlin coroutines.

The domain layer centers around the HabitViewModel, which maintains application state and handles business logic. ViewModel instances survive configuration changes, ensuring stable data across screen rotations. StateFlow is used extensively to propagate changes to the UI layer, enabling reactive updates whenever habits are modified. The ZenQuotes API integration demonstrates clean separation of concerns by isolating network operations in a dedicated suspend function with proper error handling.

For the presentation layer, Jetpack Compose enables declarative UI development with efficient recomposition. The app dynamically adapts layouts based on screen size and orientation through conditional logic checking the isLandscape flag. Complex UI components like the calendar view were built as reusable composables with customizable parameters. Theme management is handled through a custom ThemePreference class that persists user selections across sessions.

The multi-device support is implemented through shared ViewModel logic between phone and WearOS modules, with platform-specific composables handling device-specific presentations. The TYPE_STEP_COUNTER provides unified step counting across devices, with data synchronization managed through the shared repository pattern.

**Technical Challenges and Solutions**

Several significant challenges emerged during development, each requiring careful consideration to maintain app stability and performance. The first major hurdle involved implementing the adaptive layout system that could seamlessly transition between portrait, landscape, and watch interfaces. The solution involved creating a centralized isLandscape state variable derived from the current device configuration, which then conditionally renders either a bottom navigation bar or side navigation rail. This approach kept the navigation patterns familiar while optimizing for each form factor.

Database synchronization between devices presented another complex challenge. Initially, race conditions occurred when both devices attempted to update habit completion status simultaneously. This was resolved by implementing a mutex lock in the repository layer during write operations and using Room's built-in conflict resolution strategies. This solution ensured data consistency while maintaining responsiveness across devices.

Integrating the step counter required careful handling of runtime permissions and sensor availability across devices. On WearOS, persistent tracking raised battery optimization challenges, which were addressed by implementing a foreground service to ensure continuous updates. In contrast, the phone version employs a lightweight, periodic polling model that strikes a balance between power efficiency and data accuracy.

Performance optimization was particularly challenging for the calendar view, which needed to render completion indicators for up to 365 days while remaining responsive. The solution involved implementing lazy loading for calendar cells and using *derivedStateOf* to minimize recomposition when scrolling through months. The quote fetching operation was also moved to a coroutine launched during splash screen display to prevent UI jank.

Another technical hurdle involved ensuring user-uploaded habit icons persisted across app restarts. Initial implementations passed temporary URIs that became invalid after the session ended. This was resolved by copying selected images from the content resolver to the app's internal storage using a unique file name and referencing this local path in the database. Each custom icon remained accessible and correctly displayed on subsequent app launches.

## User Testing and Feedback

We conducted user testing with a few of our friends to gather feedback on usability, clarity, and design throughout development. Their insights helped shape the final product and directly influenced multiple feature iterations.

Mohammed Faizan highlighted the ease of use, stating, “It’s very accessible on the phone, and I would use it to keep track of my gym and class schedule.” He also suggested adding a streak counter next to each habit. Isabel H. commented, “The calendar view is really cool!”, and Rahat expressed excitement with, “This app is sick bro!!!” Additional participants echoed similar sentiments, praising the app’s simplicity, responsiveness, and clean design.

The visual calendar was especially well-received for helping users immediately understand their progress. Based on feedback that the habit creation form felt overwhelming, we streamlined the input fields and added image previews for custom icons. On the WearOS side, large touch targets and haptic feedback improved navigation for smaller screens.

To validate the newly added streak tracking feature, we implemented unit tests that simulate different completion patterns. These tests confirmed expected behavior, such as counting 3-day streaks correctly or resetting the count if a day is missed. This ensured the feature's accuracy before release.

Other features, such as the calendar and habit toggling UI, underwent manual testing during each sprint, ensuring they worked consistently under different device orientations, themes, and interaction scenarios.

Accessibility testing revealed improvements were needed in dark mode contrast and screen reader support. We resolved these issues by adjusting theme color tokens and adding content descriptions to all interactive elements.

## Conclusion and Future Enhancements

The Momentum Habit Tracker successfully delivers a polished, multi-platform experience that can provide genuine user value. The combination of adaptive design, motivational features, and robust data management creates an engaging tool for habit formation. The technical implementation demonstrates mastery of modern Android development practices, including Jetpack Compose, MVVM architecture, and coroutine-based async operations.

Future development will focus on four key areas: implementing login authentication to ensure secure access, expanding social features to enable friendly competition, integrating with Google Fit for more comprehensive health tracking, and adding cloud synchronization through Firebase to enable cross-device usage. Additional UX improvements include voice-controlled habit logging and predictive habit suggestions based on usage patterns. These enhancements will build upon the solid foundation established in this initial release.

The complete source code is available on GitHub, with detailed documentation in the README file.