TrueReminder: Technical Documentation

iOS App

- Swift and SwiftUI
- WidgetKit for home screen widgets
- UserNotifications framework with custom notification actions

Android App

- Flutter with Dart
- Native Android widgets
- Firebase Cloud Messaging for notifications

Backend

- Firebase Authentication for Google Sign In, Apple Sign In, and email login
- Cloud Firestore for real time database
- Firebase Cloud Functions for background tasks like friend code generation

Subscription Management

- RevenueCat SDK on both platforms

Architecture

The app follows a simple client side architecture. There is no custom backend server. All data flows through Firebase.

Firestore collections:

- users: stores user profile, friend codes, friend lists, and subscription status
- reminders: stores all reminders with userId field for ownership and sharedWith array for sharing

Real time sync works through Firestore snapshot listeners. When a reminder changes on one device, all other devices receive the update within seconds.

Notifications are scheduled locally on each device. When a reminder is created or updated, the app calculates the next trigger time and schedules a local notification. For recurring reminders, the app reschedules the next occurrence after each notification fires.

Friend sharing uses a 6 digit code system. Each user has a unique code stored in Firestore. Adding a friend queries Firestore for the matching code and creates a bidirectional friend relationship.

RevenueCat Implementation

RevenueCat handles all subscription logic. We use a single entitlement called "TrueReminder Pro" that unlocks all Pro features.

On app launch, we configure RevenueCat with our API key and set the user ID to match Firebase Auth UID. This ensures the same subscription status across iOS and Android for the same user.

When a user logs in, we call Purchases.logIn with their Firebase UID. RevenueCat automatically syncs their subscription status from either platform.

Subscription status is also written to Firestore under users/{userId}/subscription so Android can read iOS purchases without needing a separate RevenueCat setup on Android.

Pro features check revenueCat.isProUser before enabling custom intervals, custom snooze options, and custom groups.

Cross Platform Subscription Note

Currently the Android version does not support in app purchases because we have not yet obtained a Google Play Developer account. Users can purchase the Pro subscription on iOS and it will automatically sync to Android through Firestore. When a user logs into the same account on Android, the app reads the subscription status from Firestore and unlocks Pro features accordingly.