Development Specification: Supplement Tracker App

1. Overview

This document outlines the development-ready specification for a cross-platform mobile app designed to help users track supplements taken and symptoms experienced throughout the day. The application supports user-defined favorites, search and correlation features, and stores data securely using Firebase Firestore with offline capabilities.

2. Target Platforms

- Android (mobile)
- iOS (mobile)

3. Technology Stack

- Flutter (for cross-platform UI)
- Firebase (Authentication, Firestore, Analytics, Cloud Functions if needed)
- Firestore Local Caching (offline support)
- AI-assisted development with Copilot, Gemini, or ChatGPT for code scaffolding

4. Functional Requirements

- User registration and login with Firebase Auth
- Daily, weekly, and monthly calendar view
- Log supplements with optional dosage and notes
- Log symptoms with optional severity and notes
- Define favorite supplement/symptom groups and assign custom colors
- Search symptoms
- Correlate recent supplements with symptom occurrences
- View analytics (bar/line graphs) over time
- Offline data entry with sync on reconnect

5. Data Models

User:
  - id
  - displayName

- email
- preferences
- createdAt

Supplement:
- id
- name
- dosage
- note
- createdBy
- createdAt

Symptom:
- id
- name
- severityScale
- createdBy
- createdAt

FavoriteGroup:
- id
- type
- name
- colorHex
- items
- createdBy
- createdAt

LogEntry:
- id
- type
- userId
- itemId
- favoriteGroupId
- date
- time
- notes

- severity
- createdAt

CorrelationQuery:
- id
- userId
- symptomId
- timeFrameDays
- result
- createdAt

6. Firestore Data Model Diagram

7. Firestore Security Rules

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {

    match /users/{userId} {
      allow read, write: if request.auth.uid == userId;
    }

    match /supplements/{supplementId} {
      allow read, write: if request.auth.uid == resource.data.createdBy;
    }

    match /logEntries/{logId} {
      allow read, write: if request.auth.uid == resource.data.userId;
    }
```

```
    match /favoriteGroups/{groupId} {
      allow read, write: if request.auth.uid == resource.data.createdBy;
    }

    match /symptoms/{symptomId} {
      allow read: if true;
      allow write: if request.auth.uid == resource.data.createdBy;
    }

    match /correlationQueries/{queryId} {
      allow read, write: if request.auth.uid == resource.data.userId;
    }
  }
}
```

8. Data Sync Strategy

- Local cache provided by Firestore will enable offline logging.
- Writes will be queued and synced when the device reconnects to the internet.
- Firestore real-time listeners can be used to update the calendar and analytics UI reactively.

9. AI-Assisted Development

Developers can use tools like GitHub Copilot or ChatGPT to scaffold Flutter UI widgets, Firestore integration code, and Firebase Auth workflows. The requirements in this document can be pasted into those tools to bootstrap project files.