FIREBASE SETTINGS.

i Found 3 Firebase projects.

✔ **Select a Firebase project to configure your Flutter application with** · oldwisdom-2b2da (OldWisdom)

✔ **Which platforms should your configuration support (use arrow keys & space to select)?** · android, ios, macos, web, windows

i **Firebase android app com.example.oldwisdom is not registered on Firebase project oldwisdom-2b2da.**

i **Registered a new Firebase android app on Firebase project oldwisdom-2b2da.**

i **Firebase ios app com.example.oldwisdom is not registered on Firebase project oldwisdom-2b2da.**

i **Registered a new Firebase ios app on Firebase project oldwisdom-2b2da.**

i **Firebase macos app com.example.oldwisdom registered.**

i **Firebase web app oldwisdom (web) is not registered on Firebase project oldwisdom-2b2da.**

i **Registered a new Firebase web app on Firebase project oldwisdom-2b2da.**

i **Firebase windows app oldwisdom (windows) is not registered on Firebase project oldwisdom-2b2da.**

i **Registered a new Firebase windows app on Firebase project oldwisdom-2b2da.**

Firebase configuration file lib/firebase\_options.dart generated successfully with the following Firebase apps:

**Platform** **Firebase App Id**

web 1:315365731796:web:cc53ba95ff7eb01dce7a41

android 1:315365731796:android:44612fc8efc2b0fece7a41

ios 1:315365731796:ios:8208da6143138cb3ce7a41

macos 1:315365731796:ios:8208da6143138cb3ce7a41

windows 1:315365731796:web:49ffbd52bf5e4c81ce7a41

Learn more about using this file and next steps from the documentation:

> <https://firebase.google.com/docs/flutter/setup>

APP GUIDE 🡪 OLD WISDOM

**🧭 Navigation Flow of the App (as understood so far)**

mermaid

CopyEdit

graph TD

A[main.dart] --> B[MainNavigation (Bottom Nav)]

B --> C[ChapterScreen (Chapters)]

B --> D[ScenariosScreen (Scenarios)]

B --> E[Placeholder: Home, Journal, More]

C --> F[ChapterDetailView (with chapterNumber)]

F --> G[Related Scenarios (ListTile)]

**🔹 1. main.dart**

* Initializes Supabase
* Initializes Hive (offline caching)
* Opens Box<Chapter>
* Launches app via runApp(WisdomGuideApp())

**🔹 2. MainNavigation Widget**

* Bottom navigation with 5 tabs:
  + **Home**
  + **Chapters** → ChapterScreen
  + **Scenarios** → ScenariosScreen
  + **Journal**
  + **More**

**🔹 3. ChapterScreen**

* Fetches a list of chapter summaries via Supabase chapter\_summary view
* Shows list using ListView.builder
* Tapping a chapter → navigates to ChapterDetailView(chapterNumber: chapter.number)

**🔹 4. ChapterDetailView**

* Accepts int chapterNumber
* Fetches full Chapter object from chapters table using .eq('number', chapterNumber)
* Also fetches scenarios linked to that chapter
* Shows:
  + Title, subtitle, theme
  + Overview
  + Key Teachings
  + Related Scenarios (as ListTile)

**🔹 5. ScenariosScreen**

* Loads a paginated list of all scenarios (10 at a time)
* Allows future extensibility: filters, tags, favorites
* (Not yet wired to detail view of scenario)

**🔹 6. Other Screens (Placeholder for now)**

* Home
* Journal
* More  
  ➡️ These currently show placeholder content (simple Text widgets)

**📘 Supabase Tables/Views in Use**

| **Table/View** | **Purpose** | **Notes** |
| --- | --- | --- |
| chapter\_summary *(view)* | Lightweight summary for listing chapters | Contains: number, title, subtitle, scenario\_count, verse\_count, etc. |
| chapters | Full details of each chapter | Queried via .eq('number', chapterNumber) |
| scenarios | All scenario entries linked to chapters | Filtered by chapter\_id (which maps to number currently) |

**🛠️ Suggested Database View Fixes**

To better support this flow, ensure that your **chapter\_summary view** includes:

* ✅ number (primary key used in app)
* ✅ title, subtitle
* ✅ verse\_count, scenario\_count
* ✅ theme (optional but nice for listings)
* 🔁 Future: You may want to include created\_at or chapter\_order

**✅ Summary of Navigation Flow**

| **From** | **To** | **When** |
| --- | --- | --- |
| main.dart | MainNavigation | on app launch |
| MainNavigation | ChapterScreen | when user taps "Chapters" tab |
| ChapterScreen | ChapterDetailView | when user taps a chapter card |
| ChapterDetailView | N/A (static for now) | shows related content |

// ----------------------------------------------------------------  
// To disable displaying the Journal entry flow after completion:  
// 1. Remove or comment out any navigation to JournalScreen in your MainNavigation or AppRouter.  
// For example, in your bottom navigation widget, omit the "Journal" tab.  
// 2. Leave all journal models and services in place for future re‑activation, but prevent the UI from invoking them.  
// 3. If you used a boolean flag to conditionally show journal, set it to false:  
// dart // const bool enableJournal = false; // // In your navigation logic: // if (enableJournal) { // items.add(BottomNavigationBarItem( // icon: Icon(Icons.book), // label: 'Journal', // )); // } //   
// 4. Keep the code around until you finalize the decision, then clean up unused journal files.  
// ----------------------------------------------------------------

JULY-18-2025

UPDATED DOCUMENT

**OldWisdom App: MVP‑1 & Phase‑2 Documentation**

This document summarizes all of the major workstreams, features, schema changes, and technical decisions made through MVP‑1 and Phase‑2 of the OldWisdom (GitaGyan) project.

**1. MVP‑1: Core Wisdom Flow**

**1.1 Supabase Schema**

**Tables**

* **chapters**: stores full chapter details
  + ch\_chapter\_id (PK int)
  + ch\_title (text)
  + ch\_subtitle (text)
  + ch\_summary (text)
  + ch\_verse\_count (int)
  + ch\_theme (varchar)
  + ch\_key\_teachings (text[])
* **scenarios**: linked life situations for each chapter
  + id (PK uuid/int)
  + sc\_title (text)
  + sc\_description (text)
  + sc\_category (text)
  + sc\_chapter (FK → chapters.ch\_chapter\_id)
  + sc\_heart\_response, sc\_duty\_response, sc\_gita\_wisdom (text)
  + sc\_tags, sc\_action\_steps (text[])
  + created\_at (timestamp)
* **chapter\_summary** (VIEW)
  + cs\_chapter\_id, cs\_title, cs\_subtitle, cs\_verse\_count, cs\_scenario\_count
* **daily\_quote** (planned MVP‑2)
  + dq\_id (uuid)
  + dq\_description (text)
  + dq\_reference (text)
  + created\_at (timestamp)
* **gita\_verses** (added Phase‑2)
  + gv\_chapter\_id (FK), gv\_verse\_id (PK int), gv\_description (text)

**1.2 Dart Models & Hive Caching**

* **Chapter**, **Scenario**, **ChapterSummary**, **DailyQuote**, **Verse** classes with fromJson/toJson.
* Hive adapters generated for offline caching of Chapter and (planned) JournalEntry.

**1.3 SupabaseService**

Central service methods:

* fetchChapterSummaries()
* fetchChapterById(chapterId) (with Hive cache)
* fetchScenariosByChapter(chapterId)
* fetchScenarios(limit, offset)
* fetchRandomDailyQuote()
* fetchVersesByChapter(chapterId)

**1.4 UI Screens**

**MainNavigation (Bottom Nav)**

* Tabs: Home, Chapters, Scenarios, Journal (placeholder), More.

**ChapterScreen**

* Lists summaries via chapter\_summary view.
* Tapping navigates to **ChapterDetailView**.

**ChapterDetailView**

* Displays full chapter data: title, subtitle, summary, theme, key teachings.
* Related Scenarios list; each tappable.
* Verses button opens **VersesListView**.

**ScenariosScreen**

* Paginated list of all scenarios.
* Filter by tags via ChoiceChips.
* Tapping navigates to **ScenarioDetailView**.

**ScenarioDetailView**

* Cards: Description, Heart Says, Duty Says (vertical), Gita Wisdom.
* "Get Wisdom" button reveals bullet‑list action steps.
* (Phase‑2) Related Tags card to filter scenarios.

**HomeScreen**

* Displays random daily quote on every app open.
* Later to include daily reminders and journal shortcuts.

**VersesListView**

* Given a chapter ID, fetches and displays each verse in a card.
* Header with chapter title/subtitle.

**MoreScreen**

* Appearance: Dark mode toggle, Font size selector.
* Language: dropdown (EN, हिंदी, ಕನ್ನಡ).
* Resources: About dialog, References screen.
* Extras: Share App, Rate & Review, Clear Cache, App Version.

**1.5 Analytics Integration**

* **Firebase Analytics** added via FlutterFire CLI.
* AnalyticsService wrapper for logEvent(name:…, parameters…).
* Instrumentation: log chapter\_opened and scenario\_opened on tap.
* Verified in Firebase DebugView and Events dashboard.

**1.6 Development & Build Notes**

* **Hive**: Hive.initFlutter() and ValueListenableBuilder for theme/font.
* **CocoaPods**: set platform :macos, '10.13' in macos/Podfile and disable "Based on dependency analysis" for Flutter Run Script phases.
* **win32** conflicts resolved by removing unnecessary overrides (macOS only).

**2. Phase‑2 Enhancements (Planned / Completed)**

1. **Daily Quotes** stored in daily\_quote and randomly displayed in HomeScreen.
2. **Verse Browser**: gita\_verses table + VersesListView screen for chapter‑based verse reading.
3. **Localization** scaffolding: UI language dropdown; content translation columns TBD.
4. **Journal** (on hold): Hive‑backed entries, NewEntryDialog, JournalScreen list/detail.
5. **Performance**: Firebase Analytics instrumentation.
6. **More** screen polish: share, rate, clear cache, version info.

**Next Steps / Roadmap**

* **Content Translations:** Add translation columns (\_hi, \_kn) in Supabase & update models.
* **Journal Flow:** finish journal entry CRUD sync to Supabase & enable journaling.
* **Push Notifications:** daily quote reminders via Firebase Cloud Messaging.
* **Crashlytics:** integrate for error monitoring.
* **Tests & CI:** unit tests for service layer, UI widget tests, CI/CD pipeline.

*Document generated on July 15, 2025*

JULY 15 2025 – UPDATED DOCUMENT

**OldWisdom App — Project Overview & MVP Baseline**

This document captures the current state of the OldWisdom Flutter project (MVP‑1) and outlines planned Phase‑2 work for language support and journaling. It’s intended to bring any external AI or development partner up to speed quickly.

**1. MVP‑1: Core Functionality (Available Now)**

**1.1 Platforms & Tech Stack**

* **Flutter** (3.32.5) for iOS, Android, macOS, Web
* **Supabase** for backend: authentication, Postgres tables (chapters, scenarios, gita\_verses, etc.)
* **Hive** for offline storage of chapters & (future) journal entries
* **just\_audio** / **audio\_session** for background music

**1.2 Key Screens & Flows**

1. **Splash Screen**
   * Animated lotus glow + scaling logo
   * Auto-navigation to home after delay
2. **Home Screen**
   * Top “Wisdom Guide” card
   * “Verse of the Day” card: fetches a random row from gita\_verses via SupabaseService.fetchRandomVerse()
   * Displays text & Chapter X, Verse Y footer
3. **Chapters Screen**
   * Lists all 18 chapters from Supabase
   * Tapping a chapter pushes **ChapterDetailView**
   * Detail view shows title, subtitle, verse count, key teachings, related scenarios
   * “Read More” expansion for long descriptions
4. **Scenarios Screen**
   * Lists all scenarios (from scenarios table)
   * Fuzzy flow: tap scenario → **ScenarioDetailView** → list of related scenarios → further drill-down
5. **More Screen**
   * About & References pages
   * Theme (light/dark) toggle, font size selector
   * Background music toggle (persisted via Hive + SettingsService)
   * Share & package info actions
6. **Global Navigation**
   * BottomNavigationBar visible on all main tabs: Home, Chapters, Scenarios, More
   * Preserves navigation stack on tab switch (IndexedStack)
7. **Audio Service**
   * Singleton AudioService manages a looping track
   * Respects user preference (on/off)
8. **Offline Storage**
   * Hive boxes:
     + chapters (pre-populated from Supabase)
     + journal\_entries (schema-ready but journal UI postponed)

**1.3 Code Structure**

/lib

├─ main.dart # Entry & RootScaffold nav

├─ services/

│ ├─ supabase\_service.dart

│ ├─ settings\_service.dart

│ ├─ audio\_service.dart

│ └─ journal\_service.dart (stubbed)

├─ models/

│ ├─ chapter.dart (+ Hive adapter)

│ ├─ scenario.dart

│ ├─ verse.dart

│ └─ journal\_entry.dart (+ Hive adapter)

├─ screens/

│ ├─ home\_screen.dart

│ ├─ chapters\_screen.dart

│ ├─ chapter\_detail\_view.dart

│ ├─ scenarios\_screen.dart

│ ├─ scenario\_detail\_view.dart

│ ├─ more\_screen.dart

│ ├─ references\_screen.dart

│ └─ about\_screen.dart

└─ assets/

├─ images/

│ └─ divine\_scroll\_bg.png

└─ audio/

└─ Riverside\_Morning\_Calm.mp3

**1.4 Testing & Automation**

* Unit tests under test/ for services & widgets
* Integration tests under integration\_test/

**2. Phase‑2 & Beyond (Planned)**

**2.1 Journaling Flow**

* **New Entry**: open dialog, enter text + rating + category
* **List**: show saved entries, filter/sort, tap to view detail
* **Sync**: write-through to Supabase + offline in Hive

**2.2 Multi‑Language Support**

* Localize static UI (Arb files) for English, Hindi, Kannada
* Fetch gita\_verses translations via additional Supabase fields / separate gita\_verses\_localized table
* Language selector in More screen to switch app locale at runtime

**2.3 Polishing & Analytics**

* In-app analytics events (chapter opened, scenario tapped)
* Performance optimizations (frame drops, image caching)
* Theming refinements (custom color palettes, typography scale)

**MVP‑1 delivered:** stable cross‑platform reading app with random-verse, chapter/scenario browsing, theme & audio.  
**Next up:** user journaling and target‑market localization.

*End of document.*

CLAUDE – TEST HELP

**🚀 OldWisdom Testing Implementation - Quick Action Plan**

**🧪 OldWisdom Testing Setup Guide**

**📁 File Structure Setup**

Create this exact folder structure in your project:

oldwisdom/

├── test/

│ ├── main\_test.dart

│ ├── models/

│ │ └── hive\_models\_test.dart

│ ├── services/

│ │ ├── settings\_service\_test.dart

│ │ ├── audio\_service\_test.dart

│ │ └── supabase\_service\_integration\_test.dart

│ ├── widget/

│ │ ├── home\_screen\_test.dart

│ │ ├── chapters\_screen\_test.dart

│ │ ├── scenarios\_screen\_test.dart

│ │ └── more\_screen\_test.dart

│ └── test\_helper.dart

├── integration\_test/

│ ├── full\_app\_test.dart

│ └── navigation\_test.dart

└── assets/

└── icons/

├── app\_icon\_1024.png

├── icon\_background.png

└── icon\_foreground.png

**🛠️ Setup Commands**

**1. Update your pubspec.yaml**

Replace your current dev\_dependencies section with the updated version I provided above.

**2. Install dependencies**

flutter clean

flutter pub get

**3. Generate mock files**

flutter packages pub run build\_runner build --delete-conflicting-outputs

**4. Create test helper file**

Create test/test\_helper.dart:

import 'package:flutter/services.dart';

import 'package:flutter\_test/flutter\_test.dart';

import 'package:hive\_flutter/hive\_flutter.dart';

class TestHelper {

static Future<void> setupTestEnvironment() async {

TestWidgetsFlutterBinding.ensureInitialized();

// Initialize Hive

await Hive.initFlutter();

// Mock platform channels

TestDefaultBinaryMessengerBinding.instance.defaultBinaryMessenger

.setMockMethodCallHandler(

const MethodChannel('plugins.flutter.io/package\_info'),

(MethodCall methodCall) async {

return {

'appName': 'OldWisdom',

'packageName': 'com.example.oldwisdom',

'version': '1.0.0',

'buildNumber': '1',

};

},

);

TestDefaultBinaryMessengerBinding.instance.defaultBinaryMessenger

.setMockMethodCallHandler(

const MethodChannel('com.ryanheise.just\_audio.methods'),

(MethodCall methodCall) async {

switch (methodCall.method) {

case 'init':

return 'test\_player';

case 'setAsset':

return {'duration': 180000};

case 'play':

case 'pause':

case 'stop':

case 'setLoopMode':

return null;

default:

return null;

}

},

);

TestDefaultBinaryMessengerBinding.instance.defaultBinaryMessenger

.setMockMethodCallHandler(

const MethodChannel('plugins.flutter.io/url\_launcher'),

(MethodCall methodCall) async {

return true;

},

);

TestDefaultBinaryMessengerBinding.instance.defaultBinaryMessenger

.setMockMethodCallHandler(

const MethodChannel('plugins.flutter.io/share\_plus'),

(MethodCall methodCall) async {

return null;

},

);

}

static Future<void> cleanupTestEnvironment() async {

await Hive.deleteFromDisk();

}

}

**🏃‍♂️ Running Tests**

**Basic test commands:**

# Run all unit tests

flutter test

# Run specific test file

flutter test test/services/settings\_service\_test.dart

# Run with coverage

flutter test --coverage

# Run integration tests

flutter test integration\_test/

# Run tests in watch mode

flutter test --watch test/services/

**Advanced test commands:**

# Generate coverage report

flutter test --coverage

genhtml coverage/lcov.info -o coverage/html

open coverage/html/index.html # macOS

# OR

start coverage/html/index.html # Windows

# Run tests on specific device

flutter test -d chrome # For web tests

flutter test -d "iPhone Simulator" # For iOS simulator

# Run tests with custom timeout

flutter test --timeout=60s

# Verbose output for debugging

flutter test --verbose

**🔧 Test Configuration Files**

**Create test/flutter\_test\_config.dart:**

import 'dart:async';

import 'package:flutter\_test/flutter\_test.dart';

import 'test\_helper.dart';

Future<void> testExecutable(FutureOr<void> Function() testMain) async {

setUpAll(() async {

await TestHelper.setupTestEnvironment();

});

tearDownAll(() async {

await TestHelper.cleanupTestEnvironment();

});

await testMain();

}

**Create analysis\_options.yaml (if not exists):**

include: package:flutter\_lints/flutter.yaml

analyzer:

exclude:

- "\*\*/\*.g.dart"

- "\*\*/\*.mocks.dart"

linter:

rules:

avoid\_print: false # Allow prints in tests

prefer\_const\_constructors: false

use\_build\_context\_synchronously: false

**🏗️ Build and Test Pipeline**

**Create .github/workflows/test.yml for CI/CD:**

name: Test and Build

on:

push:

branches: [ main, develop ]

pull\_request:

branches: [ main ]

jobs:

test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Setup Flutter

uses: subosito/flutter-action@v2

with:

flutter-version: '3.32.5'

channel: 'stable'

- name: Install dependencies

run: flutter pub get

- name: Generate mocks

run: flutter packages pub run build\_runner build --delete-conflicting-outputs

- name: Run tests

run: flutter test --coverage

- name: Upload coverage

uses: codecov/codecov-action@v3

with:

file: coverage/lcov.info

- name: Build APK

run: flutter build apk --debug

- name: Build iOS (dry run)

run: flutter build ios --debug --no-codesign

integration\_test:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v3

- name: Setup Flutter

uses: subosito/flutter-action@v2

with:

flutter-version: '3.32.5'

channel: 'stable'

- name: Install dependencies

run: flutter pub get

- name: Run integration tests

run: flutter test integration\_test/ --verbose

**📊 Test Categories Explanation**

**Unit Tests (Fast, Isolated)**

* **Services**: Test business logic in isolation
* **Models**: Test data serialization and Hive storage
* **Utilities**: Test helper functions

**Widget Tests (Medium speed, UI focused)**

* **Screens**: Test individual screen behavior
* **Navigation**: Test route transitions
* **User interactions**: Test button taps, form inputs

**Integration Tests (Slower, Full app)**

* **End-to-end flows**: Complete user journeys
* **Performance**: Memory usage, frame rates
* **Platform integration**: Audio, storage, network

**🎯 Specific Tests for Your App**

**High-Priority Tests:**

1. **Settings persistence** across app restarts
2. **Navigation state preservation** with multiple navigators
3. **Audio service** play/pause functionality
4. **Supabase data fetching** with error handling
5. **Hive storage** for offline functionality

**App Store Requirements Tests:**

1. **No crashes** under normal usage
2. **Memory usage** stays within limits
3. **Performance** maintains 60fps
4. **Network errors** handled gracefully
5. **Accessibility** features work correctly

**🚨 Common Issues and Solutions**

**Mock File Generation Issues:**

# If mock generation fails:

flutter packages pub run build\_runner clean

flutter packages pub run build\_runner build --delete-conflicting-outputs

**Hive Test Issues:**

# If Hive tests fail:

flutter test test/models/ --verbose

# Check if adapters are properly registered

**Integration Test Platform Issues:**

# For iOS simulator issues:

flutter test integration\_test/ -d "iPhone 14"

# For Android emulator issues:

flutter test integration\_test/ -d emulator-5554

**Memory Issues in Tests:**

// Add to test files if memory issues occur:

tearDown(() async {

// Clear any static instances

// Reset singletons

await Future.delayed(Duration(milliseconds: 100));

});

**📈 Test Coverage Goals**

**Minimum Coverage for App Store:**

* **Unit Tests**: 80%+ coverage
* **Widget Tests**: 70%+ coverage
* **Integration Tests**: All critical user flows

**Files to Prioritize:**

1. services/supabase\_service.dart - 90%+
2. services/settings\_service.dart - 95%+
3. services/audio\_service.dart - 85%+
4. Main navigation screens - 80%+

**🔍 Debugging Tests**

**Enable detailed logging:**

// Add to test files for debugging:

setUp(() {

debugPrint('Starting test: ${currentTestName}');

});

tearDown(() {

debugPrint('Finished test: ${currentTestName}');

});

**Test golden files (screenshot testing):**

# Generate golden files:

flutter test --update-goldens

# Compare against golden files:

flutter test test/widget/

**✅ Pre-Release Test Checklist**

Before submitting to app stores, run:

# 1. Full test suite

flutter test --coverage

# 2. Integration tests on multiple devices

flutter test integration\_test/ -d "iPhone 14"

flutter test integration\_test/ -d "Pixel 6"

# 3. Performance profiling

flutter run --profile

# 4. Build verification

flutter build apk --release

flutter build ios --release

# 5. Static analysis

flutter analyze

# 6. Dependency audit

flutter pub deps

Your app is now ready for comprehensive testing! 🚀

Run flutter test to start, and gradually add more tests as you identify areas that need coverage.

**PROMOT-2**

**⚡ 5-Minute Quick Start**

**Step 1: Update pubspec.yaml**

# Replace your dev\_dependencies section with the updated version provided above

# Add the flutter\_launcher\_icons configuration

**Step 2: Install dependencies**

flutter clean

flutter pub get

flutter packages pub run build\_runner build

**Step 3: Create app icon**

# 1. Download the app icon from the HTML generator I created

# 2. Save as assets/icons/app\_icon\_1024.png

# 3. Run: flutter pub run flutter\_launcher\_icons:main

**Step 4: Add test files**

# Copy the test files I provided into these locations:

mkdir -p test/services test/widget test/models integration\_test

# Then copy each test file to its respective folder

**Step 5: Run tests**

flutter test

**📋 Today's Tasks (1-2 hours)**

**✅ High Priority (Do Today)**

1. **Update pubspec.yaml** with test dependencies ⏱️ 5 min
2. **Generate app icons** using my HTML tool ⏱️ 10 min
3. **Copy test helper file** (test/test\_helper.dart) ⏱️ 5 min
4. **Run basic tests** to verify setup ⏱️ 10 min
5. **Fix placeholder URLs** in more\_screen.dart ⏱️ 15 min

**🔧 Medium Priority (This Week)**

1. **Add unit tests** for your core services ⏱️ 30 min
2. **Create widget tests** for main screens ⏱️ 45 min
3. **Set up integration tests** for user flows ⏱️ 30 min
4. **Generate coverage report** ⏱️ 10 min
5. **Test on physical devices** ⏱️ 20 min

**🏪 App Store Preparation (Next Week)**

**Critical for Launch:**

* [ ] All placeholder URLs replaced with real ones
* [ ] Privacy policy published and accessible
* [ ] Terms of service published and accessible
* [ ] App tested on multiple devices and OS versions
* [ ] Screenshots taken for store listings
* [ ] App descriptions written
* [ ] Test coverage above 70%

**Build Commands:**

# Android

flutter build appbundle --release

# iOS

flutter build ios --release

**🎯 Focus Areas for Your App**

**Your App's Unique Testing Needs:**

1. **Complex Navigation Structure**
   * Test tab preservation with multiple Navigator keys
   * Test back button behavior
   * Test deep linking
2. **Supabase Integration**
   * Test offline handling
   * Test data synchronization
   * Test error scenarios
3. **Audio Service**
   * Test background music toggle
   * Test app lifecycle (pause/resume)
   * Test different device audio scenarios
4. **Settings Persistence**
   * Test theme changes
   * Test font size changes
   * Test language switching
5. **Hive Local Storage**
   * Test data persistence across app restarts
   * Test migration scenarios
   * Test storage limits

**🐛 Common Issues You Might Face**

**Issue 1: Mock Generation Fails**

# Solution:

flutter packages pub run build\_runner clean

flutter packages pub run build\_runner build --delete-conflicting-outputs

**Issue 2: Hive Adapter Conflicts**

# Solution: Update your model files to use consistent typeIds

# Chapter: typeId = 1

# JournalEntry: typeId = 2

**Issue 3: Audio Tests Fail**

# Solution: The test helper I provided mocks the audio channels

# Make sure to use TestHelper.setupTestEnvironment() in your tests

**Issue 4: Supabase Tests Timeout**

# Solution: Create a test environment in Supabase or mock the calls

# For now, use the integration tests I provided

**📊 Test Metrics to Track**

**Before App Store Submission:**

* **Unit Test Coverage**: Target 80%+
* **Widget Test Coverage**: Target 70%+
* **Integration Tests**: All critical flows covered
* **Performance**: No memory leaks, smooth 60fps
* **Crashes**: Zero crashes in normal usage

**Key Performance Metrics:**

# Check app size

flutter build apk --analyze-size

flutter build appbundle --analyze-size

# Memory usage profiling

flutter run --profile

**🎉 Success Criteria**

You'll know you're ready for launch when:

✅ All tests pass consistently ✅ App works offline and online ✅ Settings persist across app restarts ✅ Navigation works smoothly on all tabs ✅ Audio can be toggled without crashes ✅ App builds successfully for both platforms ✅ No placeholder content remains ✅ Privacy policy and terms are accessible ✅ App has been tested on multiple devices

**📞 Next Steps After Implementation**

1. **Week 1**: Implement basic test coverage
2. **Week 2**: Store preparation and asset creation
3. **Week 3**: Beta testing with real users
4. **Week 4**: Final polish and submission

**🛠️ Quick Commands Reference**

# Daily development

flutter test --watch test/

# Pre-commit checks

flutter analyze

flutter test

# Pre-release checks

flutter test --coverage

flutter build apk --release

flutter build ios --release

# App icon generation

flutter pub run flutter\_launcher\_icons:main

# Performance profiling

flutter run --profile --trace-startup

**Remember**: Start with the high-priority tasks today, and you'll have a solid testing foundation that will make your app store submission much smoother!

Your app architecture is already well-structured, so adding comprehensive tests will give you confidence for launch. Focus on the user flows that matter most: daily verse viewing, chapter navigation, and settings management.

Good luck with your launch! 🪷✨

APPSTORE LAUNCH

nishantgupta@beenas-MacBook-Pro OldWisdom % flutter build ios --no-codesign

Warning: Building for device with codesigning disabled. You will have to manually codesign before deploying to device.

Building com.example.oldwisdom for device (ios-release)...

Running pod install... 2,331ms

Running Xcode build...

└─Compiling, linking and signing... 20.4s

Xcode build done. 95.2s

**✓** Built build/ios/iphoneos/Runner.app (68.8MB)

Font asset "MaterialIcons-Regular.otf" was tree-shaken, reducing it from 1645184 to 3736 bytes (99.8% reduction). Tree-shaking can be disabled by providing the --no-tree-shake-icons flag when building your app.

Running Gradle task 'bundleRelease'... 182.5s

**✓** Built build/app/outputs/bundle/release/app-release.aab (57.5MB)

**Write**(GOOGLE\_PLAY\_LAUNCH\_TODAY.md)

⎿  Wrote **198** lines to **GOOGLE\_PLAY\_LAUNCH\_TODAY.md**

# 🚀 **\*\*GOOGLE PLAY STORE LAUNCH - TODAY\*\***

*\*GitaWisdom Store Submission Ready\**

## ✅ **\*\*COMPLETED TECHNICAL PREPARATION\*\***

### **\*\*App Build Ready\*\*** ✅

- **\*\*Package ID\*\***: Updated to `com.hub4apps.gitawisdom`

- **\*\*App Name\*\***: GitaWisdom

- **\*\*Version\*\***: 1.0.0+1

- **\*\*Release AAB\*\***: `build/app/outputs/bundle/release/app-release.aab` (62.6MB)

… +188 lines (ctrl+r to expand)

⏺ **Update Todos**

⎿  ☒ Update Android package ID to com.hub4apps.gitawisdom

☒ Generate final app icon from ocean image

☒ Generate signed release APK/AAB

☒ Remove debug signing from release build

☒ Test final build thoroughly

☒ Create comprehensive launch guide

☐ Take 5 required screenshots

☐ Set up Google Play Console account ($25)

☐ Upload hub4apps.com privacy policy and terms

☐ Submit app to Google Play Store

⏺ **🎉 GITAWISDOM IS READY FOR GOOGLE PLAY STORE LAUNCH TODAY!**

I've successfully completed all the technical preparation for your Google Play Store launch:

**✅ What's Complete:**

1. **Updated package ID** to com.hub4apps.gitawisdom (production-ready)

2. **Generated app icons** from your ocean image

3. **Built release AAB** file for Play Store (62.6MB)

4. **Removed debug signing** for proper store submission

5. **Hidden multilanguage** options as requested

6. **Created comprehensive launch guide** with all details

**📱 Your Release File:**

/Users/nishantgupta/Documents/GitaGyan/OldWisdom/build/app/outputs/bundle/release/app-release.aab

**📋 What You Need to Do Today:**

1. **Upload legal pages** to hub4apps.com (30 min)

2. **Take 5 screenshots** of the app (20 min)

3. **Create Google Play Console account** ($25 - 1 hour)

4. **Submit app for review** (1 hour)

**📄 All Materials Ready:**

- Complete store descriptions and metadata

- Privacy policy and terms of service

- Professional app branding

- Step-by-step submission guide

- Release notes and contact information

Your GitaWisdom app is now technically ready for Google Play Store submission. The comprehensive launch guide in GOOGLE\_PLAY\_LAUNCH\_TODAY.md contains everything you need to complete the submission process today!

**Estimated total time to complete submission: 3-4 hours**

**Expected review time: 24-48 hours**