🛠️ Claude Code Task Flow for Odyseya MVP1 with Best Practices

Phase 1: Project Setup & Core Infrastructure

Initialize Flutter project:

Use latest stable Flutter version

Set up platform-specific configurations for iOS and Android

Follow platform UI/UX guidelines where relevant

Version control and backups:

Create Git repository with clear branching strategy (e.g., main, develop, feature branches)

Use .gitignore properly

Setup remote repo (GitHub/GitLab) for cloud backup

CI/CD pipeline:

Configure automated builds, tests, and linting via GitHub Actions or similar

Include code formatting checks and static analysis (e.g., Dart analyze)

Automate deployment to TestFlight and Android Beta

Firebase integration:

Initialize Firebase with security best practices (Firestore rules, storage rules)

Use environment config files (e.g., .env) to separate dev/stage/prod

Setup Firebase Authentication and enable social logins securely

Dependency management:

Pin package versions in pubspec.yaml

Regularly audit for vulnerabilities and deprecated packages

Phase 2: Mood Entry & Voice Recording

UI/UX implementation:

Use responsive layouts supporting multiple screen sizes

Follow accessibility guidelines (contrast, touch targets, screen reader labels)

Use theme and palette consistently with design tokens

Voice recording & transcription:

Manage permissions gracefully (camera, mic)

Handle recording interruptions and errors robustly

Use asynchronous calls and isolate heavy processing (transcription) off main UI thread

State management:

Use scalable state management pattern (e.g., Provider, Riverpod, or Bloc)

Separate UI from business logic and data layer

Local caching & offline support:

Cache recent journal entries locally for instant access and offline fallback

Phase 3: AI Analysis & Feedback

API integration:

Handle API keys securely (don’t hardcode in app)

Implement retry and error handling strategies for AI calls

Respect rate limits and optimize requests

User privacy:

Clearly communicate data usage and obtain consent for AI processing

Anonymize or encrypt sensitive data

Phase 4: Journaling History & Calendar

Efficient data querying:

Use paginated queries to Firestore

Index Firestore collections for performance

UI performance:

Lazy load calendar entries

Use optimized widgets to avoid rebuilds

Phase 5: Reminder System

Notifications:

Use local and push notifications with user control over preferences

Follow platform guidelines for notification presentation

Battery & resource optimization:

Minimize background tasks

Use efficient scheduling APIs

Phase 6: Payments & Paywall

In-app purchase compliance:

Follow Apple and Google policies strictly

Test purchase flows thoroughly in sandbox environments

Subscription management:

Handle edge cases like renewals, expirations, and refunds gracefully

Keep subscription state in sync with backend

Phase 7: Testing & QA

Automated testing:

Write unit tests for core logic

Widget tests for UI components

Integration tests for key user flows

End-to-end tests simulating real user interactions

Manual QA:

Test on multiple devices and OS versions

Accessibility audits

Performance profiling and optimization

Phase 8: Deployment & Monitoring

Build optimization:

Minify and obfuscate release builds

Use app bundle formats to reduce download size

Error reporting & analytics:

Integrate Crashlytics or Sentry

Monitor user engagement and retention metrics

User feedback loop:

Implement in-app feedback forms or surveys

Analyze reviews and crash reports for continuous improvement

Would you like me to generate starter Claude Code prompts or sample config files for any of these best practice steps?