

Brightli System Architecture & Design

What Brightli Is

Brightli is an iOS mobile app with Android future plans for Android distribution. I built it around a local-first data strategy with optional cloud sync for premium users because privacy is becoming more important to users. Users keep complete control over personal data while premium subscribers get seamless multi-device experiences through platform-native encrypted cloud integration. The architecture prioritizes maintainability through modular design to allow seamless addition of features without creating unnecessary technical complexity.

Core Architecture Components

I structured the app around five main architectural pieces that each solve specific problems:

- Component-based state management with specialized containers for different domains
- Intelligent reflection engine with pattern recognition and on-device behavioral analytics
- Hybrid storage strategy combining local persistence with encrypted cloud sync
- Modular component design separating presentation, business logic, and data access
- Tiered feature architecture with subscription management and graceful downgrade paths

The component-based approach prevents the common iOS problem where state management becomes a tangled mess. Each state container manages its specific domain with explicit dependencies, creating predictable data flow patterns. This matters because the app needs responsive UI updates across multiple views when users modify entries or settings.

Reflection Engine Architecture

The reflection engine is the core differentiator within the app. It runs pattern recognition entirely on-device using a prompt database with adaptive selection algorithms. This creates personalized prompts based on user behavior without sending data anywhere. The system includes:

- Pattern recognition that analyzes entries locally
- Adaptive prompt selection based on detected patterns
- On-device behavioral analytics that never leave the phone
- Prompt database that updates through app releases

Processing everything locally solves the privacy problem while creating competitive differentiation. Cloud-dependent planner and data storage apps send intimate personal

data to external servers for analysis. Brightli processes everything on the device, guaranteeing behavioral insights remain strictly private.

Hybrid Storage Strategy

The hybrid storage approach uses local persistence as the primary layer with encrypted cloud synchronization for premium users. This decision stems from both privacy principles and practical user experience requirements. Local-first means the app maintains full functionality offline and users keep complete data sovereignty. Cloud sync presents a value-added enhancement that preserves privacy principles while addressing legitimate multi-device use cases.

The storage architecture includes:

- iOS native secure storage for local persistence
- CloudKit integration for premium users with end-to-end encryption
- Conflict resolution algorithms for multi-device sync
- Graceful degradation when cloud services are unavailable

This creates market differentiation from cloud-dependent competitors while establishing a privacy-focused brand identity. Users who never subscribe still get a fully functional app that respects their data sovereignty.

Modular Component Architecture

There is strict separation between presentation, business logic, and data access layers. This modularity means the UI can be modified without touching business rules, or changing data persistence without affecting the interface. Reusable UI components reduce code duplication and maintain visual consistency across the app.

Tiered Feature Architecture

The subscription system integrates with StoreKit for native iOS payment processing. Feature access controls were implemented throughout the app with graceful downgrade pathways when subscriptions expire. Subscription state validation happens across application lifecycle events to prevent unauthorized premium access.

Architecture components:

- StoreKit integration with receipt validation
- Server-side transaction verification preventing local manipulation
- Feature flags controlling premium functionality
- Graceful UI degradation preserving free tier functionality