AbuByte POS – Architecture Overview

Production-Ready Windows Desktop POS System

PAGE 1 — SYSTEM ARCHITECTURE & CORE COMPONENTS

High-Level Architecture

Windows Desktop App (Flutter) ↔ Firebase Cloud Services

↓

Hive Local Database (Offline-First)

↓

Sync Queue → Background Sync → Firebase

Core Directory Structure

abubyte_pos/

```
├── lib/
│   ├── core/              # Foundation Services
│   ├── features/          # Business Feature Modules
│   ├── sync/              # Advanced Sync Engine
│   └── widgets/           # Reusable UI Components
├── firebase-admin-script/    # Admin User Management
├── windows/               # Windows Desktop Build
└── installer/             # Windows Installer
```

Core Services Layer

Essential Services (lib/core/services/)

- HiveService.dart   # Local database management
- FirebaseService.dart   # Cloud operations
- LogService.dart   # Audit logging
- ProviderInitService.dart # Dependency initialization

Key Configuration

// Firebase Configuration

- firebase_config.dart # Project setup

- api_keys.dart # External service keys

// Theme System

- app_theme.dart # Light/dark theme

- responsive_breakpoints.dart # Desktop UI adaptation

PAGE 2 — FEATURE MODULES & DATA FLOW

Feature Module Architecture

features/{feature_name}/

├── models/ # Data models (.dart + .g.dart)

├── provider/ # State management

├── screens/ # UI pages

├── services/ # Business logic

├── widgets/ # Feature-specific UI components

└── {feature}_FEATURES.md # Documentation

Key Feature Modules

1. Authentication & Security

// lib/features/auth/

- auth_provider.dart # User session management

- pin_service.dart # PIN lock for cashiers

- pin_lock_screen.dart # Secure access control

2. Sales & Transactions

// lib/features/sales/

- sales_provider.dart # Cart & transaction management

- promo_code_provider.dart # Discount management

- held_order_model.dart # Order suspension system

3. Product Management

// lib/features/products/

- product_provider.dart # Inventory management

- combo_bundle_provider.dart # Bundle products

- bulk_import_service.dart # CSV product imports

4. Sync Engine (Advanced)

// lib/sync/

- sync_provider.dart # Sync orchestration

- atomic_batch_service.dart # Batch operations

- connectivity_watcher.dart # Network monitoring

- sync_queue_service.dart # Operation queuing

Data Models & Hive Integration

```
@HiveType(typeId: 0)
class ProductModel {
  @HiveField(0) final String id;
  @HiveField(1) final String name;
  // ... other fields
}
```

- Hive adapters for efficient local storage
- Generated serialization (.g.dart) for fast performance

PAGE 3 — SYNC ARCHITECTURE & DEPLOYMENT

Advanced Sync System

Sync Queue Management

// lib/sync/services/sync_queue_service.dart

- Queue operations offline

- Batch processing (20 operations)

- Retry logic with exponential backoff

- Conflict resolution strategies

Atomic Batch Processing

// lib/sync/services/atomic_batch_service.dart

- Transactional operations

- All-or-nothing commits

- Ensures data consistency

Background Sync Services

// lib/sync/services/

- background_sync_service.dart

- connectivity_watcher.dart

- sync_garbage_collector.dart

- sync_alerting_service.dart

Windows Desktop Specifics

Build Configuration

// windows/runner/

- Native window management

- System tray integration

- File system access (receipts)

- Printing service integration

Admin User Setup

// firebase-admin-script/createAdminUser.js

- Creates admin users via REST API

- Sets up Firestore user docs

- Configures business profiles

- Role-based permissions

Firebase Project Structure

Firebase Project (Included)

```
├── Authentication/
│   └── Users (admin/cashier roles)
├── Firestore/
│   ├── businesses/{businessId}
│   ├── users/{userId}
│   ├── products/{productId}
│   ├── sales/{saleId}
│   └── settings/{businessId}
└── Storage/
    └── product-images/{imageId}
```

Deployment Architecture

Multi-Store Support

- Isolated Firestore documents per business
- Scoped user roles
- Independent sync queues

Offline-First Guarantees

- Hive boxes: products, sales, users, categories, settings, sync_queue, held_orders
- Immediate local operations
- Transparent online sync

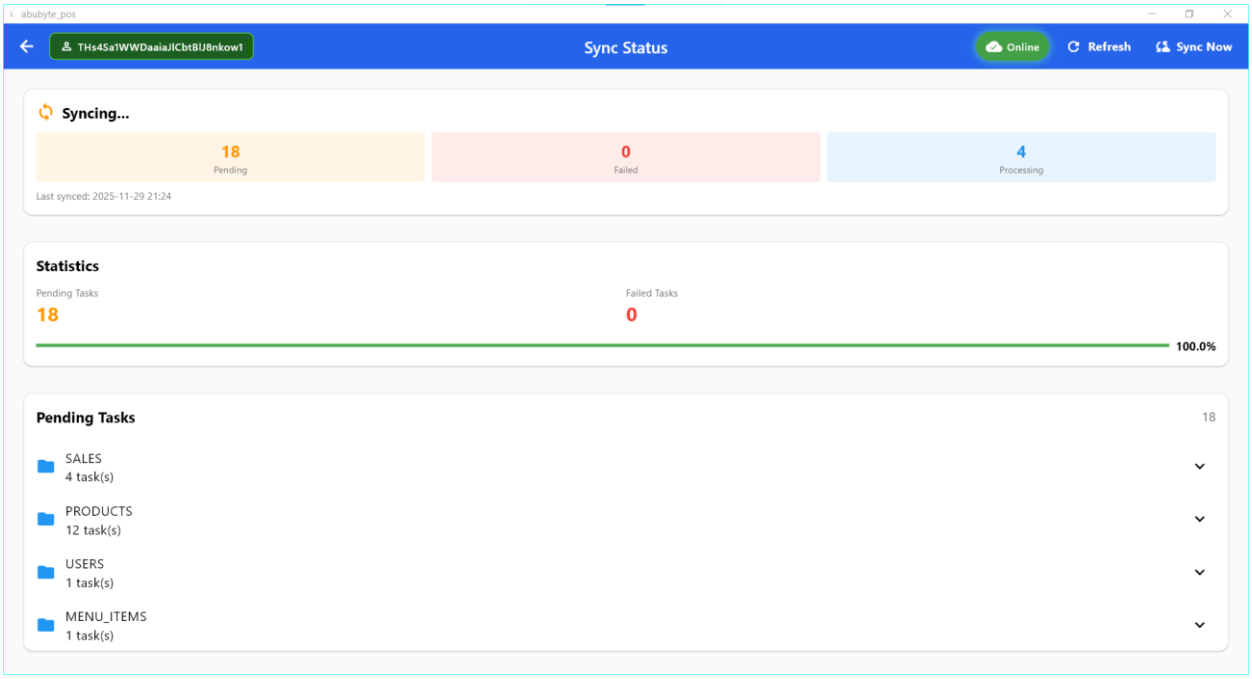
Monitoring & Maintenance

- **Sync Dashboard:** queue length, error reporting, real-time metrics
- **Audit Logging:** activity logs, user tracking, admin-only access

Scalability Features

- Batch processing & lazy loading
- Background sync for UI performance
- Modular feature architecture
- Multi-language & custom payment support ready

Sync Dashboard: queue length, error reporting, real-time metrics



"Real Business in Action: 18 transactions queued during offline operation,
now syncing automatically. Zero failures despite handling complex product updates,
user management, and sales data - proving enterprise reliability."

Technical Architecture — Enterprise Reliability & Offline-First Design 🚀