StoreFlow — A1.1 Architecture & System Design (Part 1)

1. Overview

This document outlines the foundational architecture of StoreFlow in high detail.
It focuses on global system structure, backend architecture, frontend architecture, tenancy isolation, environment configuration, and deployment models.

StoreFlow is a SaaS multi-tenant ecommerce storefront system supporting pickup, shipping, multi-store merchants, real-time operations, and loyalty. It is built to scale horizontally with clean domain separation and strict authorization controls.

Primary architectural goals:

• High tenant isolation

• Deterministic real-time order flow

• Predictable and safe multi-store scaling

• Efficient query performance under multi-tenant load

• Minimal operational overhead

• Cloud-friendly containerized or VM-based deployment

2. Backend Core Architecture

The backend is built using Laravel 10+, PHP 8.2. Its architecture is domain■modular,

with clear separation between domains: Products, Orders, Shipping, Loyalty, Stores, and Auth.

Key backend components:

• Controllers: thin, delegate to services

• Services: business logic orchestrators

• Actions: atomic single-purpose tasks

• Repositories: database access abstraction (optional but recommended)

• Policies: authorization layer

• Events: domain events emitted for real-time updates

• Listeners: handle asynchronous tasks

• Jobs: queued tasks for email sending, heavy processing, data sync

Laravel’s service container is used heavily to keep modules decoupled.

Domain boundaries:

• Auth Domain

• Merchant & Store Domain

• Product Catalog Domain

• Ordering Domain

• Fulfilment & Shipping Domain

• Customer Domain

• Loyalty Domain

• Notification Domain

3. Frontend Architecture (Dashboard)

The dashboard uses Vue 3 + Inertia.js to emulate a SPA while keeping the backend as the single source of truth. It avoids API bloat and reduces duplicate state management.

Core principles:

• Server-driven routing

• Minimal client-side state

• Modals over page transitions for rapid workflows

• Lazy-loading heavy sections such as Order History and Product Tables

• A single WebSocket connection per session

Component groups:

• Layout components

• Navigation components

• Table & list components

• Modal-based CRUD components

• Real-time order panels

• Form components with server■side validation

The dashboard must remain smooth under high order volume. Inertia partial reloads ensure only changed data is fetched, not full pages.

4. Frontend Architecture (Storefront)

The storefront is customer-facing and optimized for mobile. It has three themes: Classic, Modern, Minimal. Each theme is implemented as a standalone CSS bundle applied by selecting store.theme_key.

Principles:

• Zero JavaScript dependency except for cart and checkout logic

• Pre-rendered pages with dynamic Inertia or classical Blade

• High Lighthouse performance

• Accessibility compliance (WCAG AA)

Storefront responsibilities:

• Product browsing

• Category filtering

• Cart management

• Checkout (guest or account)

• Shipping quoting

• Order tracking real-time page

5. Tenancy Model

Tenancy isolation is a foundational requirement.

Hierarchy:

Merchant → Stores → Users → Orders/Products/etc

merchant_id is mandatory on:

• Stores

• Users

• Products

• Customers

• Orders

• Loyalty Accounts

• Audit Logs

store_id is mandatory on:

• Orders

• Shipping Zones

• Shipping Methods

• Shipping Rates

• Store-specific configurations

Enforcement:

• Middleware injects merchant_id and store_id into context

• All queries filtered by merchant_id (global scope or manual)

• Policies validate that current user belongs to merchant and store

• Prevents cross-tenant data leakage