Course Project Report

Subject: WDP301

– Quy Nhon, September 2025 –

Table of Contents

Record of Changes

Date	A* M, D	In charge	Change Description

MỤC LỤC

Full Name	Role	Email	Mobile
VangVH	Lecturer		
Hoàng Lê Quý An	Leader		
Lương Gia Khánh	Member		
Lê Anh Khôi	Member		
Nguyễn Đăng Nhân	Member		
Nguyễn Trần Quang Nhật	Member		

II. Requirement Specification

II.1 Problem description

Shoppers face fragmented catalogs across brand sites and marketplaces, inconsistent size/fit guidance, limited personalization, slow support for simple questions, checkout/payment friction, and disjointed order tracking; out-of-stock items rarely offer clear preorder/quote paths. Merchants juggle web plus Shopee/TikTok, struggle to keep inventory/pricing in sync, manually handle preorders/quotes and repetitive “Where is my order?” inquiries, and lack unified insights.

AIFShop solves this with one web & mobile platform: AI-assisted Q&A/recommendations/fit, a unified cart and checkout with integrated payments and shipping (live rates, labels, tracking), explicit preorder/quote workflows, order history with personalized promotions and easy reorder, and admin/vendor dashboards—with optional Shopee/TikTok consolidation into a single view to streamline operations and retention.

II.2 Major Features

- **FE-01: Register/Login (Email, Google, Facebook)** — secure auth; optional OTP; social sign-in.
- **FE-02: Profile & Address Book** — profile, default shipping address, preferences.
- **FE-03: Product Discovery** — browse/search/filter by category, size, color, material, price; sort by relevance/newest/price.
- **FE-04: Product Details & Variants** — images, description, size guide, availability per variant, base/sale price.
- **FE-05: Cart & Checkout** — add/edit cart, apply promo codes/vouchers, shipping rate quote, taxes; guest checkout supported.
- **FE-06: Payments** — Stripe/VNPay/MoMo integrations with webhook confirmation and downloadable receipts.
- **FE-07: Orders & Tracking** — order history, reorder, carrier tracking status & ETA notifications.
- **FE-08: Preorders & Quote Requests** — submit/manage preorder or quote for out-of-stock/coming-soon items.
- **FE-09: AI Assistant (LLM)** — product Q&A, fit guidance, recommendations, order status, personalized promotions.
- **FE-10: Admin/Vendor Console** — catalog & variant management, pricing & sales, inventory, orders/returns/preorders, shipping label & monitoring, user/vendor management, content approval/ban, chatbot configuration, basic revenue & inventory reports; optional Shopee/TikTok order & customer ingestion.

II.3 Context Diagram

The platform interacts with external services and internal components. Below is the context (with suitable technologies for this project).

External entities:

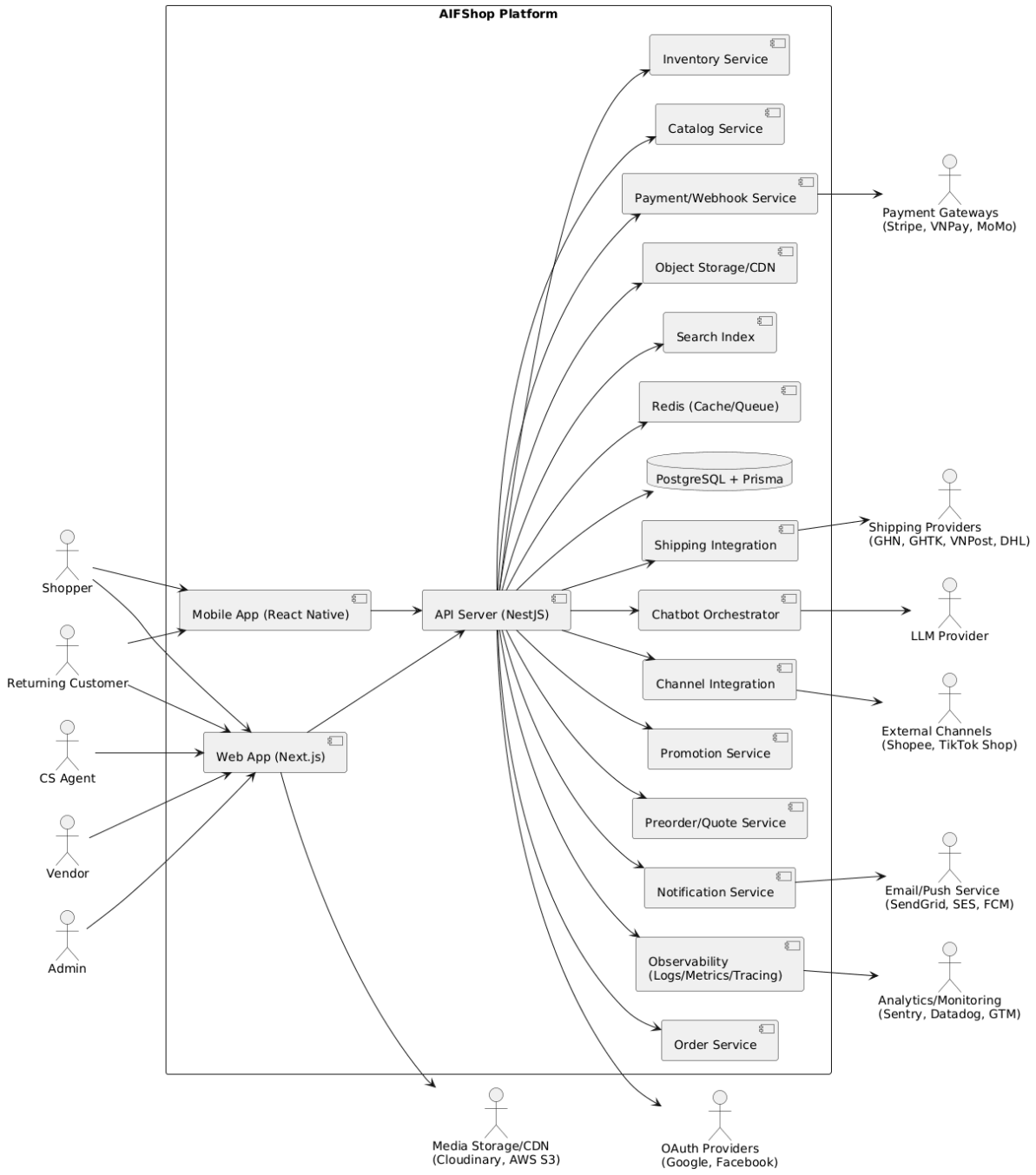
- **Shopper / Returning Customer / Vendor / Admin / (Optional) CS Agent:** Use web & mobile apps to browse, purchase, track orders, manage catalog, and moderate.
- **OAuth Providers (Google, Facebook):** Social login via OAuth 2.0/OpenID Connect.
- **Payment Gateways (Stripe, VNPay, MoMo):** Card/e-wallet/bank payments; signed webhooks.
- **Shipping Providers (e.g., GHN, GHTK, VNPost/DHL):** Rates, labels, tracking webhooks.
- **LLM Provider (e.g., OpenAI/Azure/Open-source):** Intent parsing, recommendations,

Q&A.

- **Email/Push Notification (e.g., SendGrid/SES, FCM/OneSignal):** Verification, order updates, promos.
- **Media Storage/CDN (Cloudinary or AWS S3 + CDN):** Product & banner images, optimization.
- **Analytics/Monitoring (e.g., Sentry/Datadog, GTM):** Error, performance, behavior tracking.
- **External Channels (Shopee, TikTok Shop):** Order & customer ingestion; optional product mapping.

Internal components:

- **Web App (Next.js 14 + React, Tailwind CSS):** Responsive UI, SSR/SEO, OAuth flows, calls backend APIs.
- **Mobile App (React Native / Expo):** iOS/Android app for browse, checkout, tracking, chat.
- **API Server (Node.js + NestJS, REST):** Business logic, validation, RBAC (JWT + Passport OAuth2), rate limiting.
- **Domain Services:** Catalog, Inventory, Order, Payment/Webhook, Shipping Integration, Preorder/Quote, Promotion, Chatbot Orchestrator, Channel Integration, Notification.
- **Database (PostgreSQL + Prisma ORM):** Users, profiles, stores, products, variants, inventory, carts, orders, payments, shipments, preorders, quotes, promotions, chat, channel orders, audit logs.
- **Cache/Queue (Redis):** Sessions/blacklist, rate limits, background jobs, webhook retries.
- **Search Index (Meilisearch/Elasticsearch – optional):** Fast full-text search and faceting.
- **Object Storage/CDN:** Image storage & delivery.
- **Observability:** Logs, metrics, traces; alerting on SLAs.



II.4 Nonfunction Requirements.

#	Feature	System Function	Description

1	**Performance**	System Performance	The system must handle up to 10,000 simultaneous users with a maximum response time of 3 seconds for any user interaction.
2	**Scalability**	Data Handling and System Load	The application must scale to accommodate growing numbers of users and data, ensuring consistent performance as the user base increases.
3	**Security**	User Authentication	Secure user registration, login, and password management using Firebase Authentication with data encryption.
4	**Platform Support**	Multi-Platform Compatibility	The app must support both Android and iOS devices using React Native for cross-platform compatibility.
5	**Data Integrity**	Data Synchronization	The system must ensure that data between Firebase Firestore and the app is consistent and synchronized in real-time.
6	**Reliability**	System Availability	The system should have 99% uptime, with automatic failover capabilities in case of hardware or software failure.
7	**Maintainability**	Code Structure and Updates	The code must be modular, easy to maintain, and allow for regular updates without disrupting user experience.
8	**Accessibility**	Accessibility Features	The app must meet WCAG (Web Content Accessibility Guidelines) standards, ensuring it is usable for people with disabilities (e.g., voice commands).
9	**Localization**	Language Support	The app should be localized for at least two languages (English and Vietnamese) to cater to a wider user base.

10	**Backup and Recovery**	Data Backup and Recovery	Regular backups should be taken, and the system must support data recovery in case of loss or corruption.

II.5 Functional requirements.

II.5.1 Actor

Actor	**Description**
Guest	● Browse public product listings ● View limited product details ● Sign up / sign in
Shopper / Returning Customer	● Register/Login via Email, Google, or Facebook ● Manage profile & shipping addresses ● Browse/search/filter fashion products (category, size, color, material, price) ● View product details & variants ● Add/remove items from cart ● Checkout & pay via Stripe/VNPay/MoMo ● Track order/shipping status ● Save/unsave favorites, reorder from history ● Submit preorder/quote requests ● Chat with AI assistant for Q&A, recommendations, promotions, order info ● Receive notifications (order updates, promotions)
Vendor / Store Owner	● Create/Edit/Publish/Unpublish products (name, images, description, size, color, price, stock) ● Set sale prices & manage preorders/quotes ● Track orders, payments, shipments ● Manage inventory ● View sales/revenue reports ● Respond to customer messages via system

Admin	• Approve/Hide/Delete products or stores • Manage users & vendors (ban/unban, assign/revoke roles) • Monitor orders, payments, and shipping statuses • Configure chatbot intents and guardrails • Review violation reports and maintain audit logs • Monitor platform health and content quality
External Actors (System)	• **OAuth Providers (Google/Facebook):** authenticate users and return basic profile data. • **Payment Gateways (Stripe/VNPay/MoMo):** process card/e-wallet/bank payments, send signed webhooks. • **Shipping Providers (GHN, GHTK, VNPost/DHL):** provide shipping rates, labels, and tracking webhooks. • **LLM Provider (OpenAI/Azure/Open-source):** parse intents, provide Q&A, recommendations. • **Email/Push Service (SendGrid/SES, FCM/OneSignal):** send verification, order updates, promotions, and notifications.

II.5.2 Use Cases

Use Case ID	Use Case Name	Actors Involved	Description
UC-01	**Register User**	Shopper, Vendor, Admin	Create a new account with email/password or social login (Google/Facebook); verify email/phone.
UC-02	**Login User**	Shopper, Vendor, Admin	Securely log in using credentials or social login to access personalized features.
UC-03	**Logout User**	Shopper, Vendor, Admin	End the session and clear auth tokens/cookies.
UC-04	**Manage Profile**	Shopper	View and update profile (name, avatar, shipping addresses, preferences).
UC-05	**Browse Events**	Guest,	View public product listings with pagination, filters (category, size,

		Shopper	color, material), and sorting.
UC-06	**View Product Details**	Shopper	View product details including images, description, price, stock availability, and variants.
UC-07	**Add to Cart**	Shopper	Add a product variant to the shopping cart for later checkout.
UC-08	**Checkout & Payment**	Shopper	Proceed to checkout, select shipping method, apply promo codes, and pay via Stripe/VNPay/MoMo.
UC-09	**Track Order**	Shopper	View real-time order/shipping status and estimated delivery.
UC-10	**Manage Products**	Vendor	Create, edit, publish/unpublish product listings with images, descriptions, prices, and stock.
UC-11	**Manage Orders**	Vendor	View and fulfill orders, handle preorders, manage cancellations/refunds.
UC-12	**Preorder/Quote Request**	Shopper, Vendor	Shopper submits preorder or quote request for out-of-stock/coming-soon items; vendor approves/rejects.
UC-13	**AI Chat Assistant**	Shopper	Interact with AI assistant for product Q&A, fit guidance, recommendations, order status, and promotions.
UC-14	**Admin Moderation**	Admin	Approve/ban stores or products, manage users/vendors, review violation reports, configure chatbot settings.

II.5.2.1 Use case descriptions

UC-1: Register User

UC ID and Name:	UC-1 - Register User

Created By:		Date Created:	
Primary Actor:	Shopper, Vendor, Admin	Secondary Actors:	OAuth Providers (Google/Facebook), Email/Push Service
Trigger:	User selects “Sign up”		
Description:	Create account via email/password or social login; verify email/phone.		
Preconditions:	User not authenticated.		
Postconditions:	Account created, verification sent.		
Normal Flow:	1. Open Sign up 2. Choose email or social 3. Provide data/consent 4. System creates account & role 5. Send verification/auto-login		
Alternative Flows:	● A1: Social login returns profile → linked ● A2: Email sign-up → verify later		
Exceptions:	Duplicate email; OAuth failure; weak password		
Priority:	High		
Frequency of Use:	High (Daily usage expected)		
Business Rules:	Unique email; password policy; role required		

UC-2: Login User

UC ID and Name:	UC-02 - Login User		
Created By:		Date Created:	
Primary Actor:	Shopper, Vendor, Admin	Secondary Actors:	OAuth Providers
Trigger:	User clicks **Log in**		
Description:	Log in securely using credentials or social login.		
Preconditions:	Account exists		
Postconditions:	Session created; token issued.		
Normal Flow:	Enter credentials → verify → issue token → redirect.		
Alternative Flows:	OAuth login → auto profile link.		
Exceptions:	Invalid credentials; locked account; OAuth failure.		
Priority:	High		
Frequency of Use:	High		
Business Rules:	Token expiry; password policy; account status		

UC-3: Logout User

UC ID and Name:	UC-3: Logout User		
Created By:		Date Created:	
Primary Actor:	Shopper, Vendor, Admin	Secondary Actors:	N/A
Trigger:	Click **Log out**.		
Description:	End session and clear tokens.		
Preconditions:	User logged in.		
Postconditions:	Session invalidated.		
Normal Flow:	Click logout → token blacklisted → redirect to homepage.		
Alternative Flows:	None		
Exceptions:	Session expired already		
Priority:	High		
Frequency of Use:	Medium		
Business Rules:	Session must be cleared from cache		

UC-4: Manage Profile

UC ID and Name:	UC-4: Manage Profile		
Created By:		Date Created:	
Primary Actor:	Shopper	Secondary Actors:	Media Storage
Trigger:	User selects “Profile”		
Description:	View/update name, avatar, shipping addresses, preferences.		
Preconditions:	User logged in		
Postconditions:	Profile updated in DB.		
Normal Flow:	Open profile → edit fields → save → confirmation.		
Alternative Flows:	Partial update only.		
Exceptions:	Invalid data format; image upload fails.		
Priority:	Medium		
Frequency of Use:	Medium		
Business Rules:	Address must be valid; phone format required		

UC-5: Browse Products

UC ID and Name:	UC-5: Browse		
Created By:		Date Created:	
Primary Actor:	Guest, Shopper	Secondary Actors:	N/A
Trigger:	User selects “Browse”		
Description:	View product catalog with filters (category, size, color, material).		
Preconditions:	Products available.		
Postconditions:	Products displayed.		
Normal Flow:	Open catalog → apply filter → view paginated list.		
Alternative Flows:	Search text + filters.		
Exceptions:	No products found.		
Priority:	High		
Frequency of Use:	High		
Business Rules:	Pagination default 20 items/page		

UC-6: View Products Details

UC ID and Name:	UC-6: Search/Filter Events		
Created By:		Date Created:	
Primary Actor:	Shopper	Secondary Actors:	Search Index (optional)
Trigger:	User clicks product		
Description:	View images, description, price, stock, variants.		
Preconditions:	Product exists.		
Postconditions:	Details displayed.		
Normal Flow:	Click product → fetch details → render page.		
Alternative Flows:	Related products suggested.		
Exceptions:	Product discontinued.		
Priority:	High		
Frequency of Use:	High		
Business Rules:	Show real-time stock status		

UC-7: Add To Cart

UC ID and Name:	UC-7: Add To Cart		
Created By:		Date Created:	
Primary Actor:	Shopper	Secondary Actors:	Maps, Media Storage
Trigger:	User clicks “Add to cart”		
Description:	Add product variants to cart.		
Preconditions:	User browsing products.		
Postconditions:	Cart updated.		
Normal Flow:	Select size/color → add to cart → confirm.		
Alternative Flows:	Guest cart stored with session.		
Exceptions:	Out of stock.		
Priority:	High		
Frequency of Use:	High		
Business Rules:	Quantity must not exceed available stock		

UC-8: Check Out And PayMent

UC ID and Name:	UC-8: Check Out And Payment		
Created By:		Date Created:	
Primary Actor:	Shopper	Secondary Actors:	Payment Gateway
Trigger:	User clicks “Checkout”		
Description:	Complete order with shipping + payment.		
Preconditions:	Cart not empty.		
Postconditions:	Order created; payment confirmed.		
Normal Flow:	1. Review cart 2. Enter address & shipping method 3. Choose payment method 4. Redirect to gateway / in-app pay 5. Webhook confirms payment		
Alternative Flows:	Payment retries.		
Exceptions:	Payment failed; stock unavailable.		
Priority:	High		
Frequency of	High		

Use:	
Business Rules:	Inventory reserved until payment outcome

UC-9: Track Order

UC ID and Name:	UC-9: Track Order		
Created By:		Date Created:	
Primary Actor:	Shopper	Secondary Actors:	Shipping Provider
Trigger:	User selects “Track Order”		
Description:	View shipping status and ETA.		
Preconditions:	Order exists.		
Postconditions:	Status displayed.		
Normal Flow:	Request order → fetch shipping info → show timeline.		
Alternative Flows:	Chatbot answers “Where is my order?”		
Exceptions:	Carrier delay; tracking unavailable.		
Priority:	Medium		
Frequency of Use:	Medium		

Business Rules:	Display last scan timestamp

UC-10: Manage Products

UC ID and Name:	UC-10: Manage Products		
Created By:		Date Created:	
Primary Actor:	Vendor	Secondary Actors:	N/A
Trigger:	Vendor opens dashboard		
Description:	Create/edit/publish products with variants.		
Preconditions:	Vendor authenticated.		
Postconditions:	Product updated in catalog.		
Normal Flow:	Add product info → save → publish.		
Alternative Flows:	Unpublish instead of delete.		
Exceptions:	Invalid data; duplicate SKU.		
Priority:	Medium		
Frequency of Use:	Medium		

Business Rules:	SKU unique; images required

UC-11: Manage Orders

UC ID and Name:	UC-11: Manage Orders		
Created By:		Date Created:	
Primary Actor:	Vendor	Secondary Actors:	N/A
Trigger:	Vendor views order list		
Description:	View/fulfill orders; handle returns/refunds.		
Preconditions:	Orders exist.		
Postconditions:	Order status updated.		
Normal Flow:	Select order → update status (Packed, Shipped, Cancelled).		
Alternative Flows:	Auto-update via webhook.		
Exceptions:	Refund failure.		
Priority:	High		
Frequency of Use:	High		

Business Rules:	Status transitions follow order state machine

UC-12: Preorder/Quote Request

UC ID and Name:	UC-12: Create Forum Post		
Created By:		Date Created:	
Primary Actor:	Shopper, Vendor	Secondary Actors:	Media Storage
Trigger:	User selects “Preorder” or “Request Quote”		
Description:	Shopper requests preorder/quote; vendor approves/rejects.		
Preconditions:	Product flagged as preorderable.		
Postconditions:	Preorder or quote created.		
Normal Flow:	Submit request → vendor notified → approve/reject.		
Alternative Flows:	Auto-approval.		
Exceptions:	Vendor rejects; ETA invalid.		
Priority:	Medium		
Frequency of Use:	Medium		

Business Rules:	Preorder SLA ≤ 24h for vendor response

UC-13: AI Chat Assistant

UC ID and Name:	UC-13: AI Chat Assistant		
Created By:		Date Created:	
Primary Actor:	Shopper	Secondary Actors:	N/A
Trigger:	User asks chatbot		
Description:	AI assistant answers Q&A, fit, recommendations, order status, promotions.		
Preconditions:	User logged in for personal data.		
Postconditions:	Response delivered.		
Normal Flow:	User types question → LLM parses → system fetches data → reply.		
Alternative Flows:	Handoff to human agent.		
Exceptions:	Low confidence; LLM timeout.		
Priority:	High		
Frequency of Use:	High		

Business Rules:	No PII leakage; responses logged for audit

UC-14: Admin Moderation

UC ID and Name:	UC-14: Report Content		
Created By:		Date Created:	
Primary Actor:	Admin	Secondary Actors:	Admin
Trigger:	Admin opens console		
Description:	Approve/ban stores/products; manage users; configure chatbot.		
Preconditions:	Admin authenticated.		
Postconditions:	Content/user state updated.		
Normal Flow:	View flagged items → take action (approve/ban).		
Alternative Flows:	Bulk actions.		
Exceptions:	Permission denied; system error.		
Priority:	Medium		
Frequency of Use:	Medium		

Business Rules:	Audit logs required for all admin actions

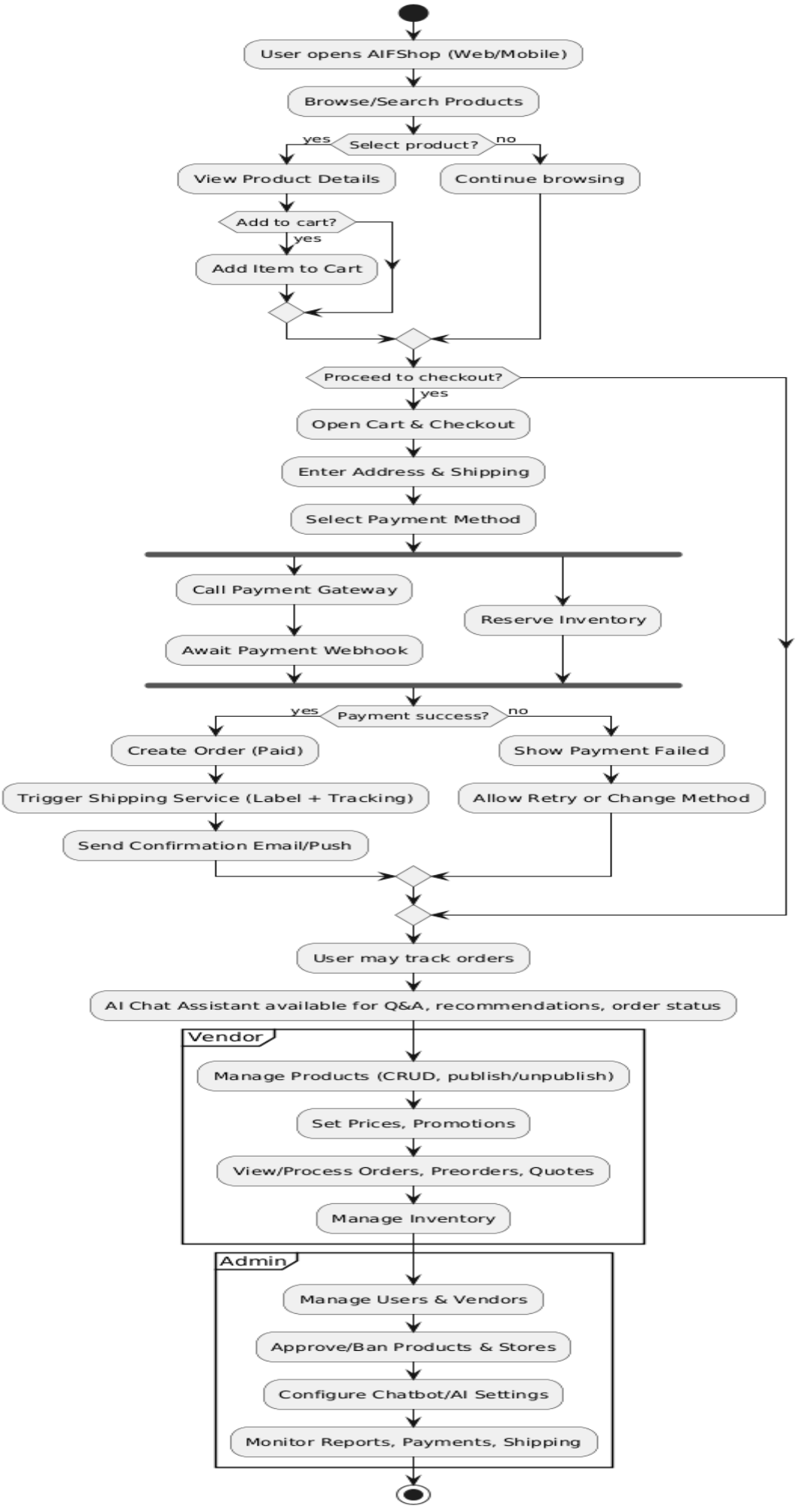
Business Rules

Provide the business rules those are applied only to the use case

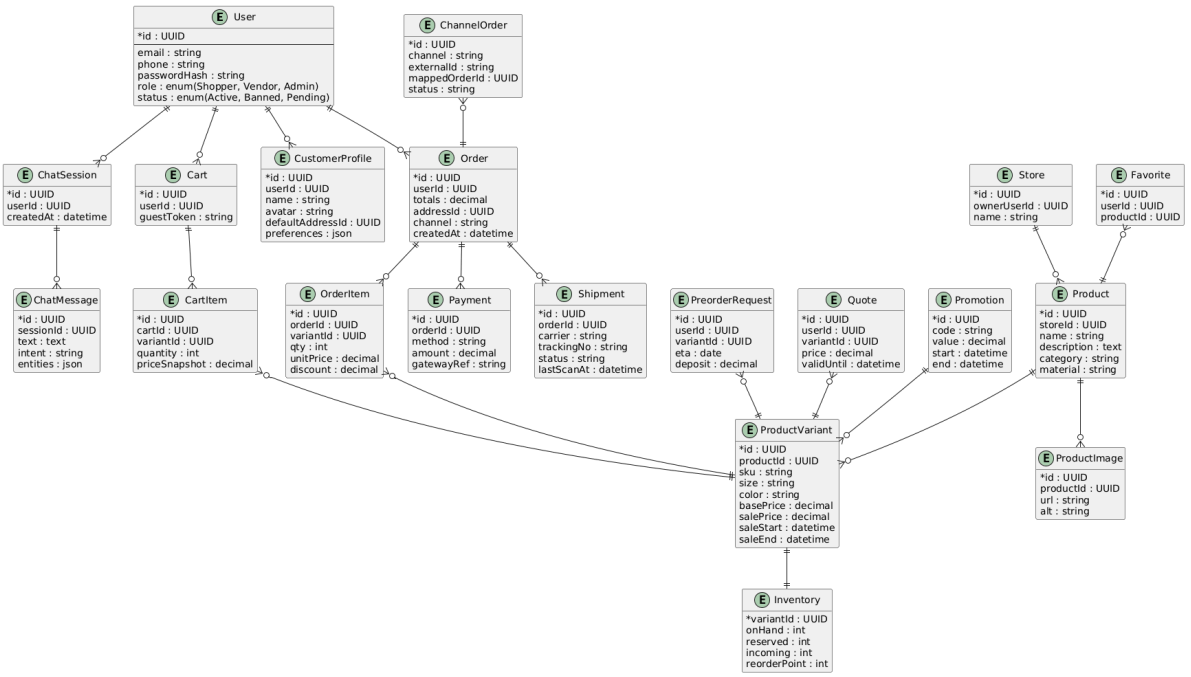
ID	Business Rule	Business Rule Description
FR1	**Password Hashing**	User passwords must be hashed with Argon2 or bcrypt with per-user salts before storage (MD5 is not allowed).
FR2	**Email/Phone Verification**	The system must send a verification email/SMS upon registration; accounts must be verified before purchasing or posting products.
FR3	**Role & RBAC**	Role is assigned at sign-up (Shopper/Vendor/Admin); Vendor has catalog and order management; Admin has moderation and system management only.
FR4	**Product Publication Workflow**	Products have states Draft → Published → Unpublished; only Published products are visible to shoppers.
FR5	**Inventory & Cart Capacity**	Cart quantity per variant must not exceed available stock; inventory is reserved during checkout and released if payment fails.
FR6	**Favorites Uniqueness**	Each user may favorite an event at most once; favorite counts reflect **unique** users and update on toggle.
FR7	**Content Moderation & Audit**	Product listings, reviews, and promotions must comply with policy; items may be auto-flagged/hidden after a threshold of reports; all moderator/admin actions are audit-logged.
FR8	**Data Privacy & Contact Sharing**	Shopper PII (phone, email, address) is hidden by default; Vendors can only view fields needed for fulfillment; all PII must follow applicable data privacy laws.

FR9	**Pricing & Discounts**	Sale price must always be less than or equal to base price; discounts require a start and end date; expired discounts auto-revert.
FR10	**Payment Confirmation**	Payment status is confirmed only via signed gateway webhooks; client-side success alone cannot update order status.

II.5.3 Activity diagram


User opens AIFShop (Web/Mobile)
Browse/Search Products
Select product?
yes
no
View Product Details
Continue browsing
Add to cart?
yes
Add Item to Cart
Proceed to checkout?
yes
Open Cart & Checkout
Enter Address & Shipping
Select Payment Method
Call Payment Gateway
Reserve Inventory
Await Payment Webhook
Payment success?
yes
no
Create Order (Paid)
Show Payment Failed
Trigger Shipping Service (Label + Tracking)
Allow Retry or Change Method
Send Confirmation Email/Push
User may track orders
AI Chat Assistant available for Q&A, recommendations, order status
Vendor
Manage Products (CRUD, publish/unpublish)
Set Prices, Promotions
View/Process Orders, Preorders, Quotes
Manage Inventory
Admin
Manage Users & Vendors
Approve/Ban Products & Stores
Configure Chatbot/AI Settings
Monitor Reports, Payments, Shipping

II.6 Entity Relationship Diagram



Entity Descriptions:

User

- **Description: Account entity controlling access and roles.**
- **Purpose: Authentication/authorization and ownership.**
- **Key fields: id, email (unique), phone?, password_hash, provider, provider_id?, role (Shopper/Vendor/Admin), status, created_at.**
- **Notes/Relations: 1–1 CustomerProfile; 1–N Store (if Vendor); 1–N Order, Cart, ChatSession; N–N via Favorite, Promotion usage; 1–N AuditLog.**

CustomerProfile

- **Description: Public-facing user profile linked to a user.**
- **Purpose: Stores display and preference information.**
- **Key fields: user_id (PK/FK), display_name, avatar_url?, default_address_id?, preferences (JSON).**
- **Notes: Exactly one profile per user; contains personalization preferences (size, style).**

Store

- **Description: A vendor’s store containing products.**
- **Purpose: Organize vendor catalog.**
- **Key fields: id, owner_user_id (FK), name, status, created_at.**
- **Notes: 1–N Product; belongs to Vendor role users.**

Product

- **Description: Item offered for sale in a store.**
- **Purpose: Core entity for discovery and purchase.**
- **Key fields: id, store_id, name, description, material?, category, created_at.**
- **Notes: 1–N ProductVariant, 1–N ProductImage; can be tagged via Promotion.**

ProductVariant

- **Description: Specific variant of a product (size, color).**
- **Purpose: Granularity for inventory and pricing.**
- **Key fields: id, product_id, sku (unique), size, color, base_price, sale_price?, sale_start?, sale_end?.**
- **Notes: 1–1 Inventory; referenced by CartItem, OrderItem, PreorderRequest, Quote.**

ProductImage

- **Description: Image of a product.**
- **Purpose: Rich visual display.**
- **Key fields: id, product_id, url, alt.**
- **Notes: 1–N per Product.**

Inventory

- **Description: Tracks stock availability for each variant.**
- **Purpose: Prevent overselling, support reordering.**
- **Key fields: variant_id (PK/FK), on_hand, reserved, incoming, reorder_point.**
- **Notes: Updated at checkout and fulfillment.**

0Cart

- **Description: Shopping basket for a user or guest.**
- **Purpose: Temporary collection of items before checkout.**
- **Key fields: id, user_id?, guest_token?, created_at.**

- **Notes: 1–N CartItem.**

CartItem

- **Description: Item in a shopping cart.**
- **Purpose: Capture chosen variant and quantity.**
- **Key fields: id, cart_id, variant_id, quantity, price_snapshot.**
- **Notes: Unique (cart_id, variant_id).**

Order

- **Description: Confirmed purchase record.**
- **Purpose: Traceable unit of payment and fulfillment.**
- **Key fields: id, user_id, status (New/Paid/Shipped/Delivered/Cancelled/Refunded), totals, payment_status, address_id, channel, created_at.**
- **Notes: 1–N OrderItem, Payment, Shipment.**

OrderItem

- **Description: Line item in an order.**
- **Purpose: Record purchased variants.**
- **Key fields: id, order_id, variant_id, qty, unit_price, discount.**
- **Notes: References ProductVariant.**

Payment

- **Description: Transaction with payment gateway.**
- **Purpose: Record amount and status.**
- **Key fields: id, order_id, method, amount, status (Pending/Success/Failed/Refunded), gateway_ref, created_at.**
- **Notes: Final payment state comes from gateway webhook.**

Shipment

- **Description: Delivery of an order.**
- **Purpose: Track logistics.**
- **Key fields: id, order_id, carrier, tracking_no, status, last_scan_at.**
- **Notes: Updates via shipping provider API.**

PreorderRequest

- **Description: Shopper request for upcoming/out-of-stock item.**
- **Purpose: Capture demand and manage approvals.**
- **Key fields: id, user_id, variant_id, status (Pending/Approved/Rejected), eta, deposit?.**
- **Notes: Linked to ProductVariant.**

Quote

- **Description: Price quotation for special request.**
- **Purpose: Handle negotiations or bulk interest.**
- **Key fields: id, user_id, variant_id, price, valid_until, status.**
- **Notes: Linked to ProductVariant.**

Promotion

- **Description: Discount or voucher.**
- **Purpose: Apply special pricing.**
- **Key fields: id, code (unique), discount_type (Percent/Fixed), value, start, end.**
- **Notes: Can apply to multiple ProductVariants.**

Favorite

- **Description: Shopper’s saved product.**
- **Purpose: Quick access, popularity counting.**
- **Key fields: user_id, product_id, created_at.**
- **Notes: Composite PK (user_id, product_id).**

ChatSession

- **Description: AI assistant interaction session.**
- **Purpose: Group messages and intents.**
- **Key fields: id, user_id, created_at.**
- **Notes: 1–N ChatMessage.**

ChatMessage

- **Description: Individual user/AI message.**
- **Purpose: Capture conversation flow.**
- **Key fields: id, session_id, role (User/AI/System), text, intent?, entities?, created_at.**
- **Notes: Used for intent parsing and auditing.**

ChannelOrder

- **Description: External order record (Shopee/TikTok).**
- **Purpose: Consolidate multi-channel sales.**
- **Key fields: id, channel, external_id, mapped_order_id, status.**
- **Notes: Links external order to internal Order.**

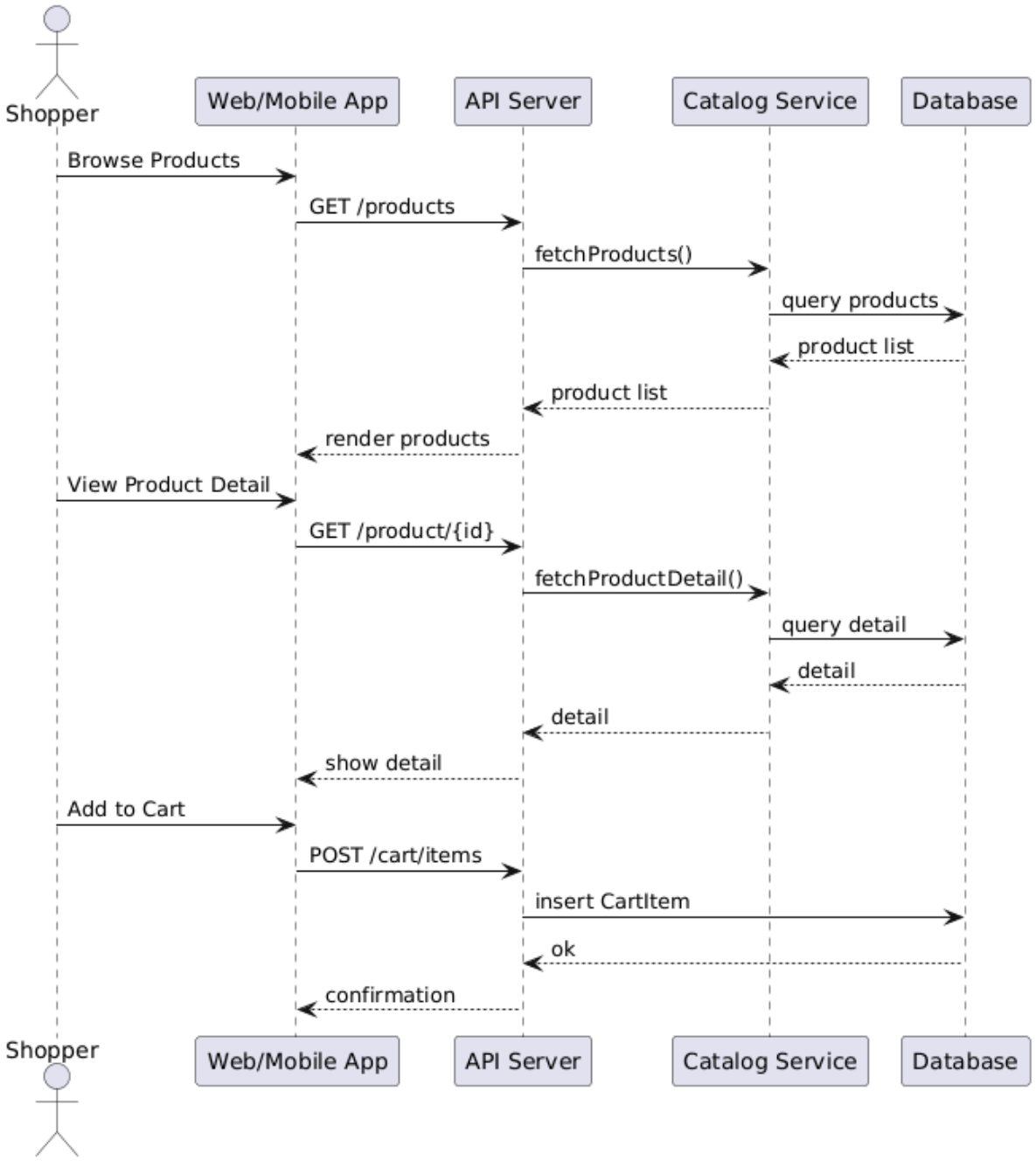
AuditLog

- **Description: Immutable record of sensitive actions.**
- **Purpose: Compliance and traceability.**
- **Key fields: id, actor_id, action, entity_type, entity_id?, before?, after?, created_at.**
- **Notes: Covers admin/vendor critical operations.**

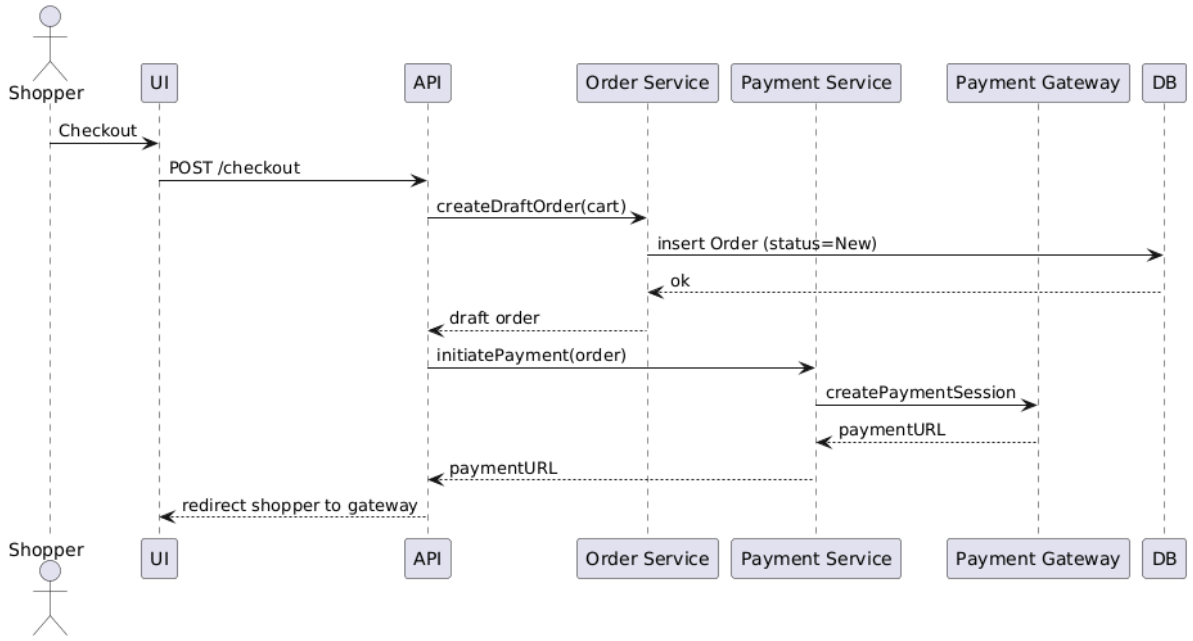
III. Analysis models.

III.1.1.Sequence Diagram

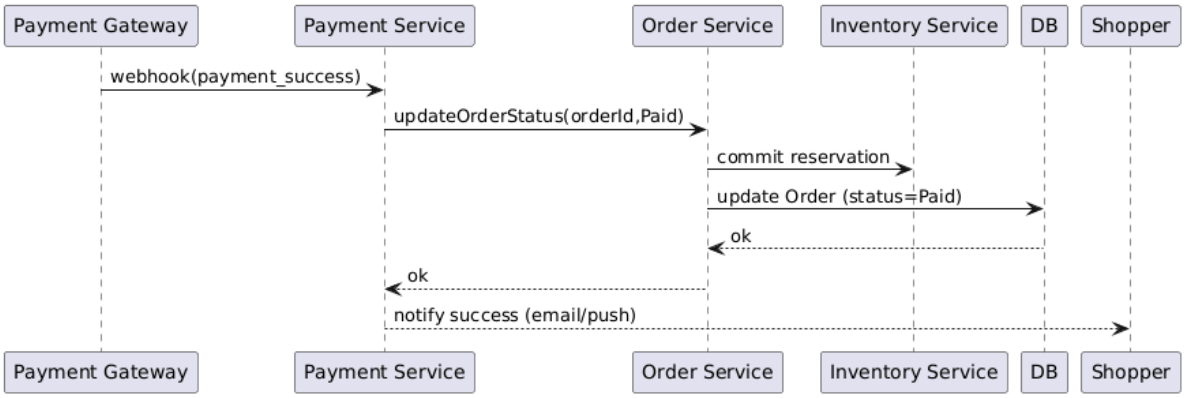
1) Browse & Add to Cart



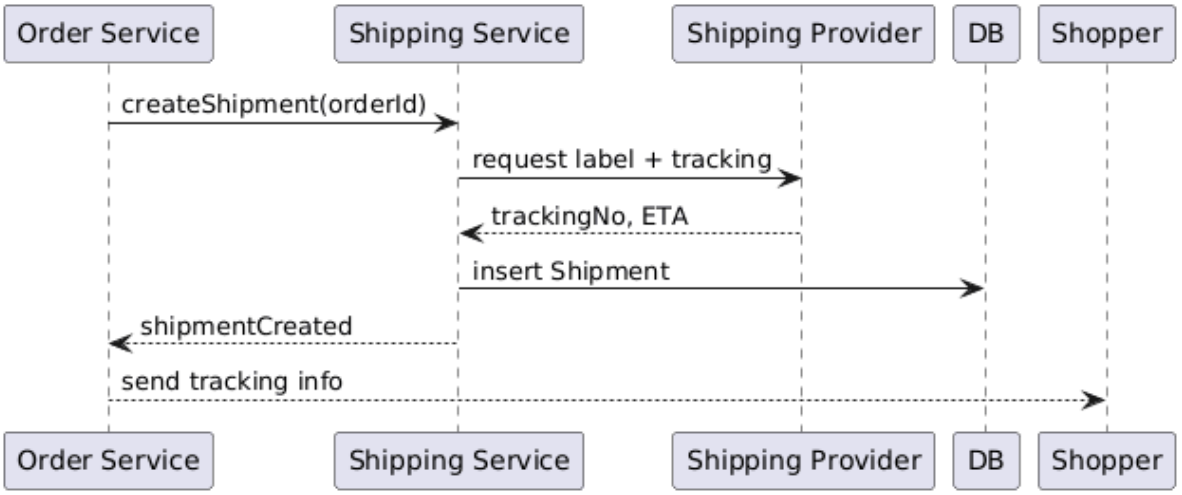
2) Checkout & Payment



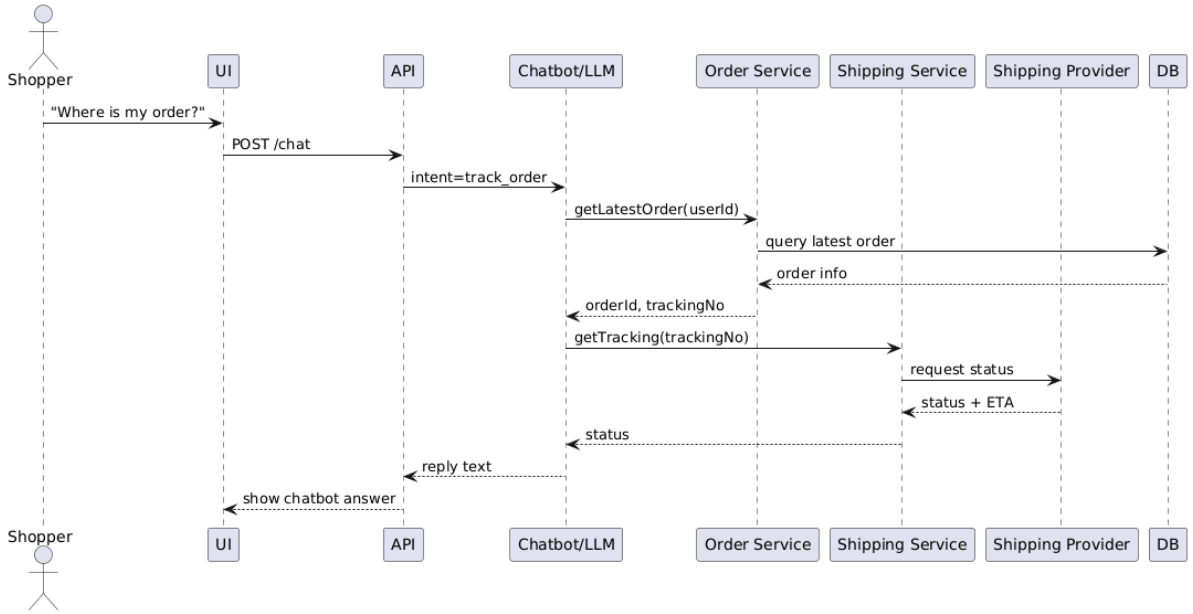
3) Payment Confirmation



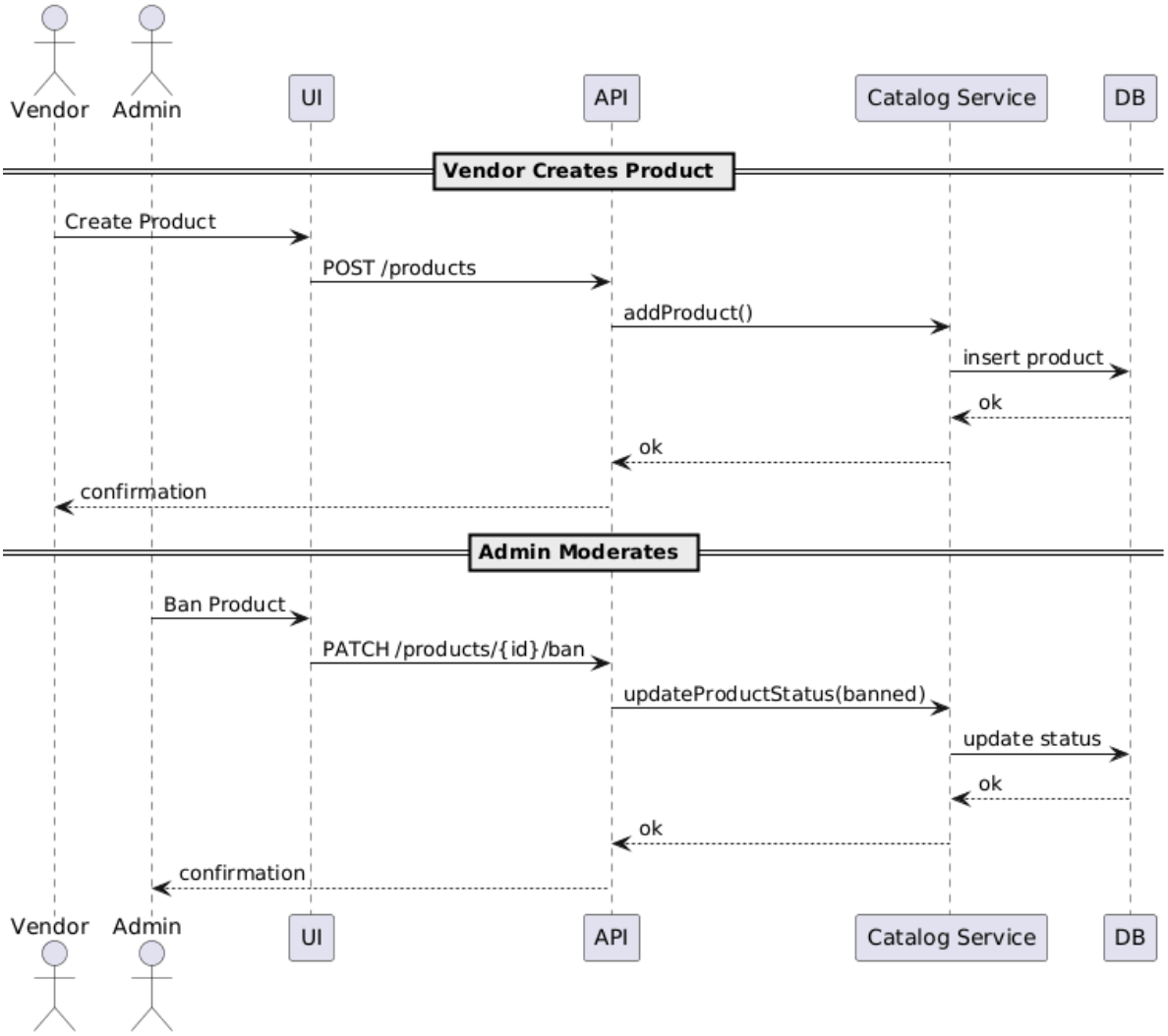
4) Shipping



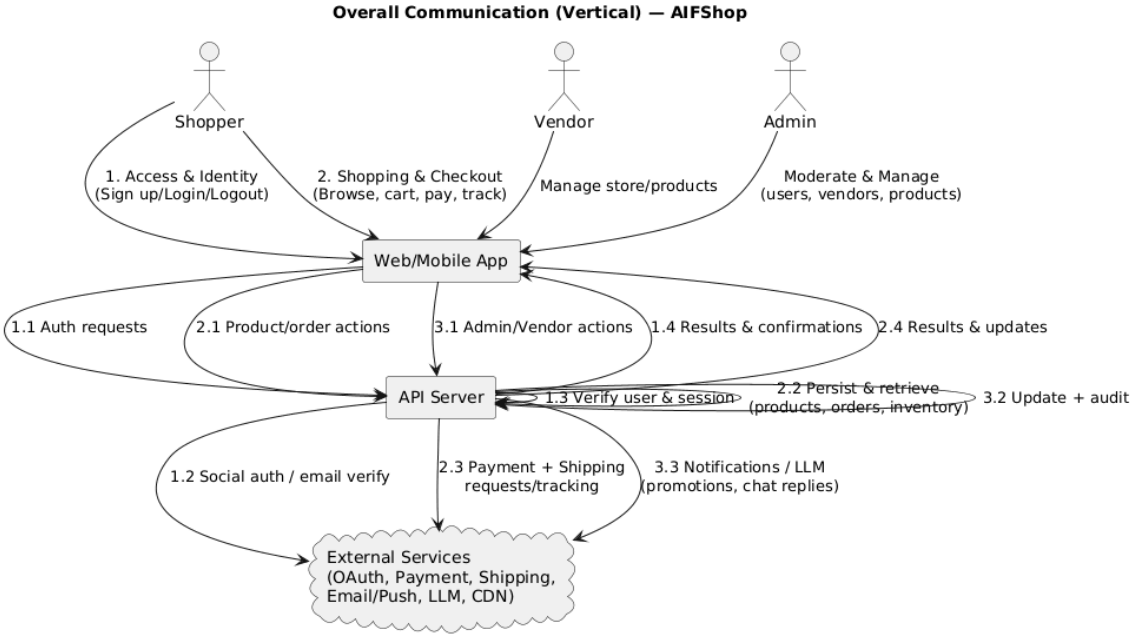
5) Order Tracking via AI Chat



6) Vendor & Admin



III.1.2. Communication Diagram



Explanation:

Actors:
Shopper (buyers), Vendor (sellers), and Admin (system moderators).

Core Components:
Web/Mobile App (frontend UI) → API Server (backend gateway) → Database (system source of truth).
External Services include OAuth (Google/Facebook), Payment (Stripe/VNPay/MoMo), Shipping APIs (GHN/GHTK/DHL), Email/Push Notifications, LLM (AI Chatbot), and CDN (images).

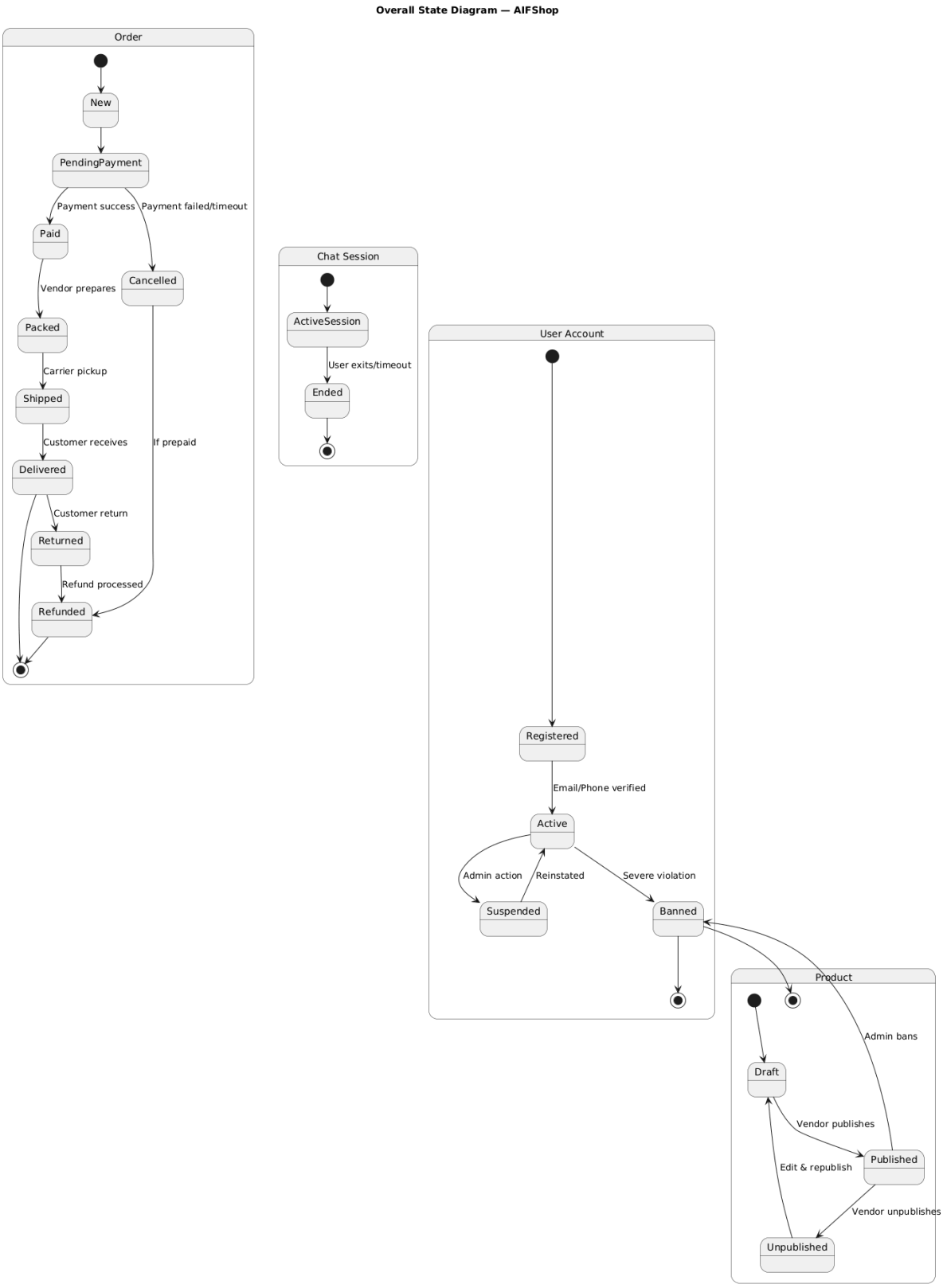
Main Flows:

- **Access & Identity:** Shopper/Vendor signs up/logs in through the Web/Mobile App. The API handles authentication, connects to OAuth/Email services, and records users/sessions in the DB.
- **Shopping & Checkout:** Shopper browses/filters products, adds to cart, makes payments, and tracks orders. The Web calls the API; the API reads/writes DB (products, orders, inventory) and integrates with Payment/Shipping; results return to the user.
- **Admin & Management:** Vendors manage products/orders; Admin moderates users and content via Web → API. The API updates the DB, keeps audit logs, and triggers notifications/promotions via External Services.

Key Idea:

The Web/Mobile App is the single entry point; the API Server orchestrates all business logic and integrations; the Database stores authoritative data; External Services are invoked on demand.

III.2 State Diagram



System (top level):

Operational – platform runs normally.
Maintenance – during deployment/migrations; returns to Operational when done.

Degraded – external service outages (OAuth, Payment, Shipping, Email/LLM); recovers to Operational.

User Session (under Operational):

Guest → Authenticated on login/sign-up.

Authenticated → Guest on logout.

Authenticated → Suspended if banned; Suspended → Authenticated if reinstated.

Product Lifecycle:

Draft → Published (vendor publishes).

Published → Unpublished (vendor action) or → Banned (admin).

Unpublished ↔ Draft (edit/republish).

Order Lifecycle:

New → PendingPayment.

PendingPayment → Paid (success) or → Cancelled (failure/timeout).

Paid → Packed → Shipped → Delivered.

Delivered → Returned → Refunded.

Cancelled → Refunded (if prepaid).

Preorder/Quote:

Requested → Approved/Rejected by vendor.

Approved → Fulfilled into Order.

Content Moderation:

Visible ↔ Pending (flagged/review).

Pending → Hidden (violation).

Visible → Hidden (admin hide/threshold).

Hidden ↔ Visible (restore).

Hidden → Deleted (permanent removal).

IV. Design specification

IV.1 Integrated Communication Diagrams

1) Objective

Describe how actors (Shopper/Vendor/Admin) interact with the core components (Web/Mobile App, API Server, Database) and external services (OAuth, Payment, Shipping, Email/Push, AI/LLM, CDN) in a single integrated diagram. The goal is to provide

a quick overview of the entire communication flow—without splitting into functional diagrams.

2) Boundary & Scope

- **In scope:** Synchronous request/response (HTTP/JSON), external service calls, DB updates, notifications.
- **Out of scope:** Detailed API logic, infrastructure-level retry/backoff, advanced batching or queue orchestration.

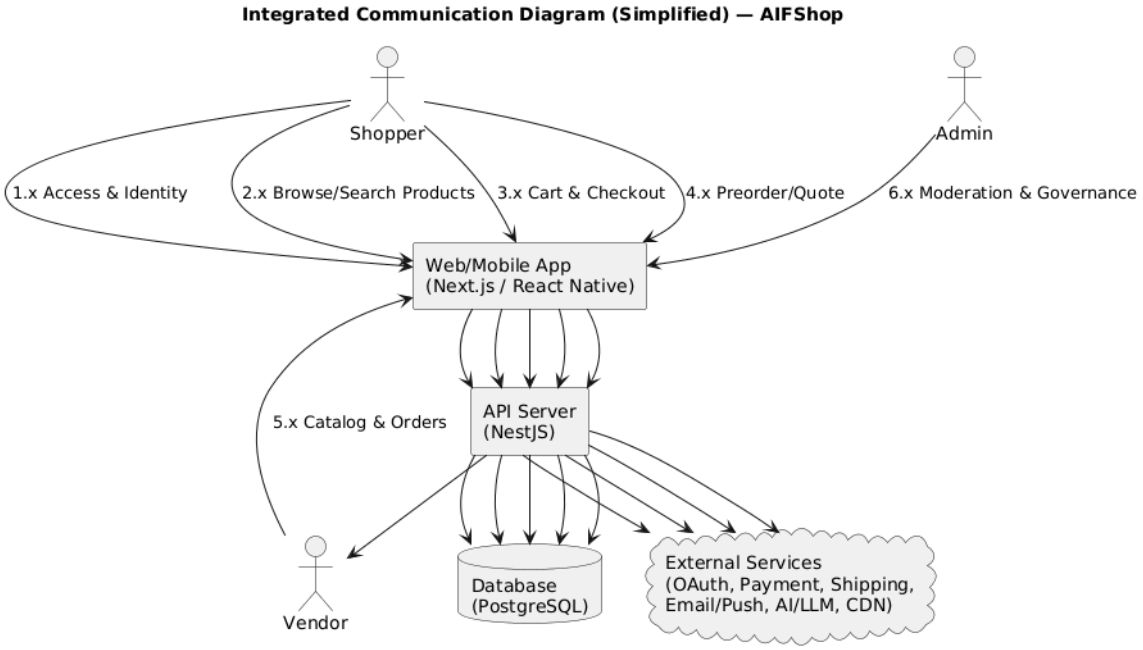
3) Actors & Components

- **Actors:** Shopper (customer), Vendor (store owner), Admin (moderator/operator).
- **Core Components:** Web/Mobile App (Next.js / React Native), API Server (NestJS), Database (PostgreSQL).
- **External Services:**
    - OAuth (Google/Facebook)
    - Payment Gateways (Stripe, VNPay, MoMo)
    - Shipping Providers (GHN/GHTK/VNPost/DHL)
    - Email/Push Notifications (SendGrid/SES, FCM/OneSignal)
    - LLM/AI Chatbot (OpenAI/Azure or open-source model)
    - Media CDN (Cloudinary/AWS S3)

4) Message Numbering Convention

- **1.x** Access & Identity (authentication, login, session)
- **2.x** Product browsing & search
- **3.x** Cart & Checkout (orders, payments, shipping)
- **4.x** Preorder & Quote requests
- **5.x** Vendor catalog & order management
- **6.x** Admin moderation & platform governance

5) Integrated Communication Diagrams



Explain:

Actors:

- **Shopper: End users browsing products, adding to cart, checking out, or making preorders.**
- **Vendor: Store owners managing products, inventory, and orders.**
- **Admin: Moderators who manage users, products, and enforce policies.**

Core Components:

- **Web/Mobile App (Next.js / React Native): The single entry point for all users.**
- **API Server (NestJS): Central gateway handling business logic, coordinating flows, and connecting to external services.**
- **Database (PostgreSQL): Authoritative system of record for users, products, orders, payments, and logs.**

  External Services (bundled): OAuth (login), Payment Gateways, Shipping APIs, Email/Push, AI/LLM Chatbot, Media CDN.

Main Flows:

1. **Access & Identity (1.x): Shopper/Vendor/Admin log in or register; API verifies via OAuth/Email and updates DB.**
2. **Product & Checkout (2.x–3.x): Shopper browses/searches products, adds to cart, checks out; API fetches DB, connects to Payment and Shipping, and confirms orders.**
3. **Preorder & Quote (4.x): Shopper submits requests for out-of-stock items; API records in DB and notifies Vendor for approval.**

4. **Vendor Management (5.x): Vendor manages catalog, pricing, and order fulfillment; API stores updates and triggers external notifications.**
5. **Admin Moderation (6.x): Admin reviews users/products, bans or restores items; API updates DB, audit logs, and may notify users.**

IV.2 System High-Level Design

1) Overview

AIFShop is an AI-powered online fashion shopping platform where shoppers can browse, purchase, preorder, and track fashion products, while vendors manage catalogs and orders, and admins moderate the system. The architecture includes a Web/Mobile App (Next.js/React Native), API Server (NestJS), PostgreSQL (Prisma ORM), with Redis (cache, rate-limit, jobs), Payment/Shipping APIs, Email/Push Notifications, AI/LLM Chatbot, and Media CDN. Authentication supports OAuth 2.0 (Google/Facebook) and JWT. The design prioritizes **security (RBAC, audit logs), scalability (horizontal scale), and high availability**.

2) Components

Frontend (Web/Mobile App)

- **Frameworks:** Next.js 14 + React (Web), React Native (Mobile), Tailwind CSS (SSR/SSG for SEO on product listings).
- **Main UI features:**
    - Browse/search/filter products; view details & variants.
    - Register/Login/Logout (Email, Google, Facebook); profile management.
    - Cart management, checkout, payment, and order tracking.
    - Preorder and quote requests.
    - Favorites & reorder from history.
    - Vendor dashboard: create/edit/publish/unpublish products, manage orders, inventory, promotions.
    - Admin dashboard: moderate products, users, vendors, configure AI/chatbot.
- **State & navigation:** Next Router; Zustand/Context; SWR/React Query for client-side caching.

Backend (API Server)

- **Tech stack:** Node.js + NestJS (REST), Passport (JWT + OAuth2), class-validator, Helmet.
- **Responsibilities:**
    - Authentication/OAuth, RBAC (Guest/Shopper/Vendor/Admin), rate limiting.
    - Product lifecycle (Draft/Published/Unpublished/Banned).
    - Cart, orders, payments, shipments, preorder & quote workflows.

    - Favorites, promotions, and personalized recommendations.
    - AI Chatbot orchestration (LLM intents: search, fit, order status, promotions).
    - Moderation queue, audit logs, email/in-app notifications.
- **API examples:**

    - **Auth:** `POST /auth/signup`, `POST /auth/login`, `GET /auth/oauth/callback`
    - **User/Profile:** `GET/PUT /me`, `GET /me/orders`
    - **Products:** `GET /products?filters`, `GET /products/:id`
    - **Vendor:** `POST /products`, `PATCH /products/:id`, `PATCH /products/:id/status`
    - **Cart & Checkout:** `POST /cart/items`, `POST /checkout`
    - **Orders:** `GET /orders/:id`, `PATCH /orders/:id/status`
    - **Preorder/Quote:** `POST /preorders`, `POST /quotes`
    - **Favorites:** `POST /products/:id/favorite/toggle`
    - **Admin:** `GET /moderation/reports`, `POST /moderation/action`, `POST /admin/users/:id/action`

Database (PostgreSQL via Prisma ORM)

- **Core models:** User, CustomerProfile, Store, Product, ProductVariant, ProductImage, Inventory, Cart, CartItem, Order, OrderItem, Payment, Shipment, PreorderRequest, Quote, Promotion, Favorite, ChatSession, ChatMessage, ChannelOrder, AuditLog.
- **Constraints & integrity:**

    - Unique: email, SKU, (user_id, product_id) for Favorites, (user_id, variant_id) for CartItem/Preorder.
    - Referential constraints between products, variants, inventory, and orders.
    - FSM (finite state machine) for Order lifecycle (New, Paid, Shipped, Delivered, Returned, Refunded).
    - Audit logs for all vendor/admin critical changes.

Cache/Queue (Redis)

- **Uses:** rate limiting, session blacklist/token revocation, counters (views, favorites), background jobs (notifications, emails), webhook retries with backoff.

Search Index (optional: Meilisearch/Elasticsearch)

- **Purpose:** Full-text search and faceted filtering (by category, color, size, material, price).
- **Sync:** Index updated when products are created/updated/published/unpublished.

Media Storage & CDN

- **Solutions:** Cloudinary or AWS S3 + CDN.
- **Purpose:** Product images, promotional banners; resizing/optimization; URLs stored in DB.

Payment & Shipping Integration

- **Payment Gateways:** Stripe, VNPay, MoMo for card/e-wallet/bank payments.
- **Shipping APIs:** GHN, GHTK, VNPost, DHL for rates, labels, and real-time tracking.
- **Integration:** Webhooks for confirmation, updates, and error handling.

Notifications (Email/In-App/Push)

- **Email:** SendGrid or AWS SES for account verification, order confirmations, shipment updates, promotions.
- **Push/In-App:** Via FCM/OneSignal for real-time order status, promotions, and chatbot updates.
- **Triggers:** order events (created, paid, shipped, delivered), preorder/quote decisions, moderation actions.

AI/LLM Chatbot

- **Provider:** OpenAI/Azure or self-hosted LLM.
- **Functions:** product Q&A, fit/recommendations, order status, personalized promotions.
- **Integration:** API orchestrator routes intents to DB, Payment, Shipping, or Vendor modules.

Security & Compliance

- **Standards:** HTTPS, JWT (HttpOnly cookie or bearer), CORS, Helmet, input validation.
- **RBAC:** Shopper/Vendor/Admin with strict separation of endpoints.
- **Mitigations:** CSRF (if cookie), XSS protection, SQL injection prevention (Prisma), rate limiting, captcha for sensitive actions.

- **Privacy:** Minimal PII storage; shoppers control data sharing with vendors; support data export/deletion.

Observability & Operations

- **Logging & APM:** Sentry/Datadog/OpenTelemetry; centralized structured JSON logs.
- **Metrics:** error rates, latency, payment success ratio, webhook retries, job backlog.
- **Backup:** daily DB backups; RPO ≤ 24h, RTO ≤ 4h.
- **Deployment:** CI/CD (build/test/lint/scan), multi-environment (Dev/Staging/Prod); horizontal scaling for API/Search; CDN for static/media assets.

---

3) System Architecture

3.1 Architecture Drivers

- **Core Functional Requirements:**
  Product browsing & search, cart & checkout, payments, shipping, favorites & reorder, preorder/quote workflows, vendor catalog & order management, AI chatbot assistant, and admin moderation.
- **Non-Functional Requirements:**
  Security (OAuth2 + JWT, RBAC, audit logs), performance (paging, caching), horizontal scalability, high availability, observability & ease of operations.
- **Constraints:**
  Web-first (SEO for product listings), Mobile support (React Native), leverage external services for OAuth, Payments, Shipping, Email/Push, AI/LLM, Media CDN, Search Index.

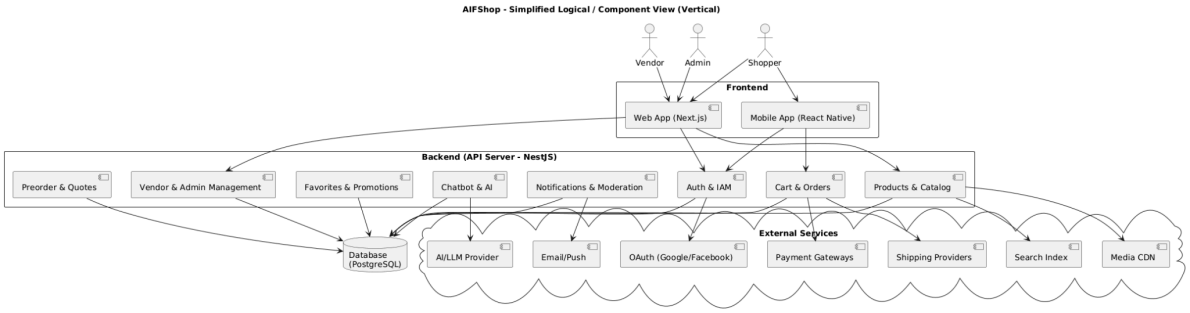
3.2 Style & Bounded Modules

- **Style: Modular Monolith (NestJS) with background workers for async jobs.**
- **Bounded Modules (responsibilities & owned DB tables):**

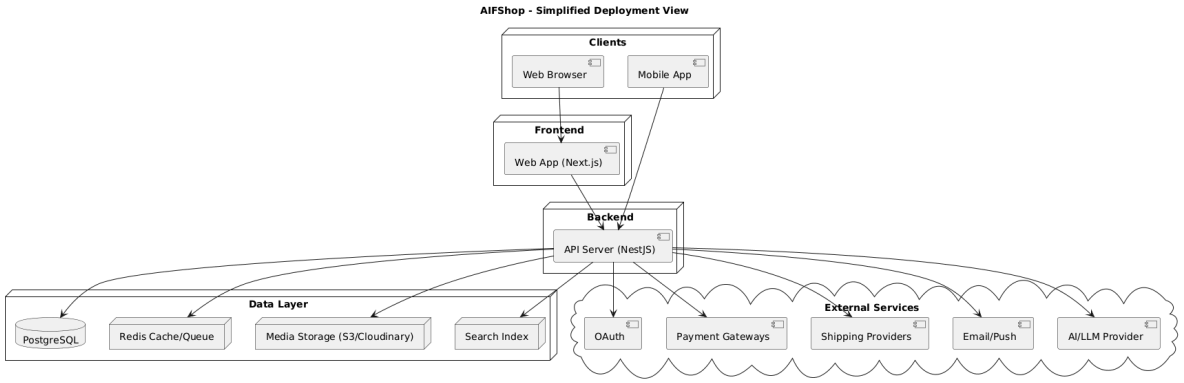
1. **Auth & IAM**
   - **Responsibilities: Sign-up/login, OAuth2, JWT, account lock/unlock, RBAC.**
   - **Tables:** `User`, `CustomerProfile`
2. **Products**

    - **Responsibilities: CRUD products, variants, images, product lifecycle (Draft/Published/Unpublished/Banned).**
    - **Tables: `Store`, `Product`, `ProductVariant`, `ProductImage`, `Inventory`**
3. **Cart & Orders**
    - **Responsibilities: Manage cart, checkout, order lifecycle (New → Paid → Shipped → Delivered/Returned/Refunded).**
    - **Tables: `Cart`, `CartItem`, `Order`, `OrderItem`, `Payment`, `Shipment`**
4. **Favorites & Promotions**
    - **Responsibilities: Add/remove favorites, count popularity, manage discount codes and campaigns.**
    - **Tables: `Favorite`, `Promotion`**
5. **Preorders & Quotes**
    - **Responsibilities: Handle preorder requests, vendor approval/rejection, quote pricing.**
    - **Tables: `PreorderRequest`, `Quote`**
6. **Chatbot & AI Assistant**
    - **Responsibilities: AI-driven Q&A, recommendations, order tracking, promotions.**
    - **Tables: `ChatSession`, `ChatMessage`**
7. **Vendor Management**
    - **Responsibilities: Vendor dashboards, catalog updates, order fulfillment.**
    - **Tables: `Store`, `Product` (shared with Products module)**
8. **Moderation & Audit**
    - **Responsibilities: Admin moderation of products, vendors, and users; logging all sensitive actions.**
    - **Tables: `AuditLog`**
9. **Notifications**
    - **Responsibilities: Email & in-app notifications, push messages, queued delivery.**
    - **Tables: `Notification`**
10. **Channel Integration (optional)**
    - **Responsibilities: Import/export orders & products from external channels (Shopee, TikTok Shop).**
    - **Tables: `ChannelOrder`**

3.3 Logical / Component View



3.4 Deployment View



3.5 Data Architecture

- **Database: PostgreSQL with Prisma ORM; UUID primary keys; unique constraints for composite keys such as `(user_id, product_id)` in Favorites and `(user_id, variant_id)` in Cart/Preorders.**
- **Suggested Indexes:**
  - `products(status, category)` **for browsing.**
  - `orders(user_id, created_at)` **for history.**
  - `favorites(user_id, product_id)` **for quick lookups.**
  - `cart_items(cart_id, product_id)` **for fast cart operations.**
  - `preorders(user_id, product_id)` **for preorder handling.**
- **Transactions: Checkout and stock allocation use `SELECT ... FOR UPDATE` within transactions to avoid overselling.**

- **Media: Only URLs stored in DB; actual content in CDN (Cloudinary/S3).**
- **Audit: `AuditLog` records all admin/vendor critical actions and state changes (append-only).**

3.6 Integration

- **OAuth2: Google/Facebook login (authorization code exchange).**
- **Payment & Shipping: Stripe/VNPay/MoMo for payments; GHN/GHTK/DHL/VNPost APIs for rates, labels, and tracking (via signed webhooks).**
- **Email/Notifications: SES/SendGrid for email; FCM/OneSignal for push/in-app; background workers handle retries/backoff.**
- **Search: Meilisearch/Elasticsearch (optional) for full-text and faceted search; reindexed by background jobs.**
- **AI/LLM: OpenAI/Azure API (or self-hosted) for product Q&A, recommendations, order tracking.**
- **Redis: For rate limiting (bucket/token), job queue (emails, notifications, reindex).**

3.7 Security Architecture

- **Authentication: JWT (HttpOnly cookie or Bearer), refresh token rotation, logout/blacklist support.**
- **RBAC: Roles: Guest, Shopper, Vendor, Admin; enforced with route guards.**
- **App Security: HTTPS, CORS, Helmet, input validation, protections against XSS, SQL injection, CSRF; captcha for sensitive operations (e.g., login, payments).**
- **Privacy: Minimal PII; opt-in sharing of contact data with vendors; user rights for export/deletion of personal data.**
- **Audit: All admin/vendor approvals, bans, and product/order changes logged immutably (append-only).**

3.8 Observability & Operations

- **Logging/Tracing: Structured JSON logs; OpenTelemetry for tracing; error monitoring via Sentry/Datadog.**
- **Metrics/Alerts: API latency (p95), error rates (5xx), OAuth/Payment/Email failure rates, queue backlog.**
- **Backup/DR: Daily DB snapshots; RPO ≤ 24h, RTO ≤ 4h.**
- **CI/CD: Pipeline includes build → test → lint → security scan → deploy (blue/green or rolling); configuration managed via environment variables and secrets vault.**

3.9 Scalability & Performance

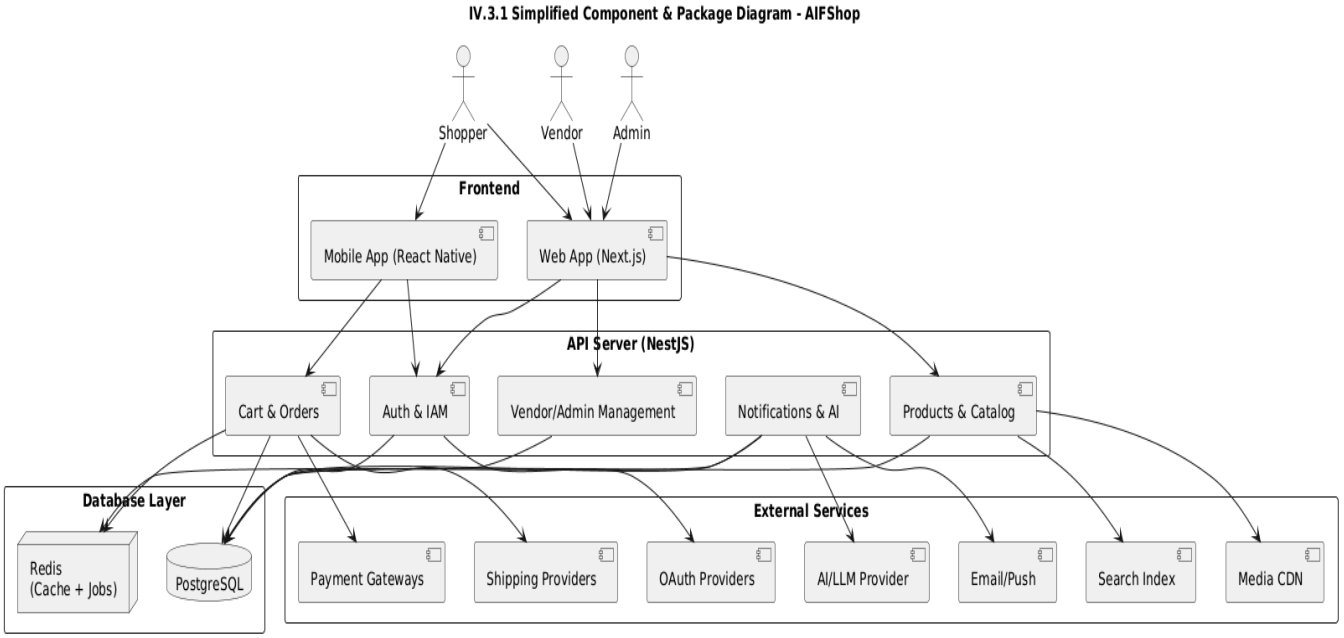
- **Horizontal Scaling: API server pods and worker pods; DB with optional read-replicas; optional search cluster.**
- **Performance Optimizations: Pagination on lists, CDN image compression, caching short lists (e.g., top products), pre-aggregated statistics for vendor dashboards.**
- **Graceful Degradation: If Search/Payment/Shipping/CDN is down → fallback to DB/basic flows; queue email/push for later.**

3.10 Risks & Mitigation

- **Overselling/Overbooking: Use DB row-level locks and transactions to ensure stock integrity.**
- **Spam/Abuse: Rate limiting, captcha, moderation queue, auto-flagging based on thresholds.**
- **Vendor Lock-In: Use adapter pattern for Email/Payment/Search/CDN integrations to swap providers if needed.**
- **Traffic Spikes: Autoscaling pods, job queues to smooth load peaks, CDN for media delivery.**

IV.3 Component and Package Diagram

IV.3.1 Component Diagram



- **Actors:**
    - **Shopper (Guest/Registered):** browse/search products, add to cart, place orders, mark favorites, request preorders/quotes, chat with AI assistant.
    - **Vendor:** create and manage products, inventory, pricing, promotions, and fulfill orders.
    - **Admin:** moderate products/vendors/users, configure AI/chatbot, manage platform policies.

- **Core Platform:**
    - **Web App (Next.js + React):** User-facing interface; calls backend APIs; uploads images to CDN; handles search and order tracking.
    - **Mobile App (React Native):** iOS/Android client for shopping, checkout, and AI chat.
    - **API Server (NestJS):** contains the main business modules:
        - **API Gateway:** entry point for HTTP requests, routes to modules.

        - **Auth & IAM:** signup/login (email + OAuth), JWT, RBAC, account lock/unlock.
        - **Products & Catalog:** CRUD products, variants, images, lifecycle (Draft/Published/Unpublished/Banned), sync to search.
        - **Cart & Orders:** manage cart, checkout, order lifecycle (New/Paid/Shipped/Delivered/Returned/Refunded), stock locking (transaction).
        - **Favorites & Promotions:** toggle favorites, count popularity, apply discounts.
        - **Preorders & Quotes:** request and approve/reject preorders, handle special pricing.
        - **Chatbot & AI:** AI-driven recommendations, product Q&A, order tracking, promotions.
        - **Vendor Management:** vendor dashboards for catalog and orders.
        - **Admin Moderation:** content/user moderation, banning, auditing critical actions.
        - **Notifications:** trigger emails/in-app/push for order updates, promotions, moderation events.
    - **PostgreSQL (Prisma ORM):** authoritative data store (users, profiles, stores, products, inventory, carts, orders, payments, shipments, favorites, preorders, quotes, promotions, chats, audit logs).
    - **Redis:** cache, rate-limit, queue jobs; token blacklist; async processing for notifications and reindex.
    - **Job Worker:** background processing (emails, push notifications, reindexing, scheduled tasks).

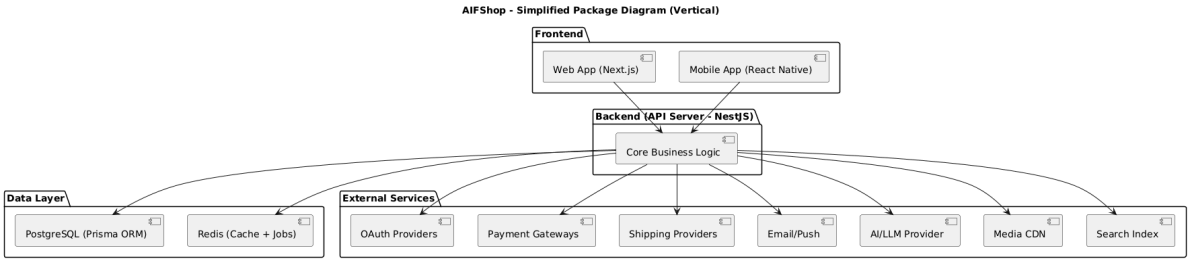
- **External Services:**
    - **OAuth (Google/Facebook):** social login.
    - **Payment Gateways (Stripe, VNPay, MoMo):** secure card/e-wallet/bank payments.

- **Shipping Providers (GHN, GHTK, VNPost, DHL):** real-time rates, labels, tracking.
- **Email/Push (SES/SendGrid, FCM/OneSignal):** verification, order updates, promos.
- **AI/LLM (OpenAI/Azure):** intent parsing, product Q&A, recommendations.
- **Media CDN (Cloudinary/S3):** store & optimize product and banner images.
- **Search Index (Meilisearch/Elastic):** fast full-text and faceted product search.

- **Main Flow:**
  Shopper/Vendor/Admin → Web/Mobile App → API Gateway → (business module) → Database/Redis → (when needed) External Services.
  Heavy/async tasks (emails, notifications, search indexing, webhook retries) are pushed to **Job Worker** via Redis.

- **Architectural Rationale:**
  Clear separation of **UI – API – Data/Infra**, modularized business logic, horizontally scalable (API & workers), secure (OAuth, JWT, RBAC, audit logs), performant (cache, search, CDN), and maintainable (external services integrated via adapters).

IV.3.2 . Package Diagram



Package descriptions

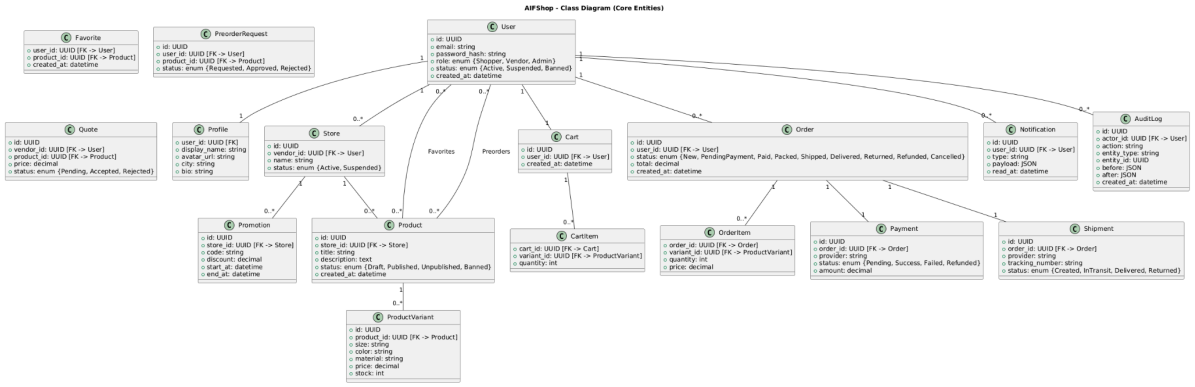
No	Package	Description

01	**Presentation Layer – Web/Mobile App (Next.js + React, React Native)**	***User interface for Shoppers, Vendors, and Admins.*** • ***Web (Next.js) for SEO-optimized product browsing.*** • ***Mobile App (React Native) for iOS/Android shopping and AI chat.*** • ***Calls REST APIs, uploads images directly to Media CDN, displays product details, recommendations, and order tracking.***
02	**Application Layer – API Server (NestJS)**	***Contains business modules:*** • ***Auth & IAM: signup/login (email + OAuth), JWT, RBAC, account lock/unlock.*** • ***Products & Catalog: CRUD products, variants, images; lifecycle (Draft/Published/Unpublished/Banned); sync to Search Index.*** • ***Cart & Orders: cart management, checkout, order lifecycle (New/Paid/Shipped/Delivered/Returned/Refunded); prevent overselling with transactions.*** • ***Favorites & Promotions: add/remove favorites, count popularity, apply discounts.*** • ***Preorders & Quotes: handle requests for out-of-stock items and vendor approval/rejection.*** • ***Chatbot & AI: AI-powered Q&A, recommendations, order tracking, promotions.*** • ***Vendor Management: vendor dashboards for catalog, pricing, orders.***

		• ***Admin Moderation: reports, approve/hide/ban products, user management, audit logs.*** • ***Notifications: email, in-app, and push notifications for order status, promotions, moderation actions.*** • ***Shared: DTOs, guards, pipes, utilities.*** *The Presentation layer **uses** this layer; it does not directly access the lower layers.*
03	**Domain Layer**	***Entities: domain models/objects (User, Profile, Store, Product, Variant, Cart Order, Payment, Shipment, Favorite, Preorder, Quote, Promotion, AuditLog).*** ***Services/Policies: pure business rules and policies, independent of technical frameworks.*** *All Application modules depend on this Domain layer for business logic.*
04	**Data / Infrastructure Layer**	• ***Database (PostgreSQL via Prisma ORM):*** ***Stores all core entities (users, profiles, stores, products, variants, carts, orders, payments, shipments, favorites, preorders, quotes, promotions, audit logs, chat sessions).*** • ***Redis (Cache + Jobs):*** ***Used for caching, rate limiting, session blacklist, background jobs (notifications, reindexing, webhook retries).*** • ***Job Worker:*** ***Asynchronous worker that processes queued jobs (emails, push notifications, search indexing, scheduled tasks).***

05	**External Services Layer**	• **OAuth Providers (Google/Facebook): for social login.** • **Payment Gateways (Stripe, VNPay, MoMo): for card/e-wallet/bank payments.** • **Shipping Providers (GHN, GHTK, DHL, VNPost): for rates, labels, and tracking.** • **Email/Push (SES/SendGrid, FCM/OneSignal): for account verification, order updates, promotions.** • **AI/LLM Provider (OpenAI/Azure): for chatbot intents and recommendations.** • **Media CDN (Cloudinary, AWS S3): for product and banner images.** • **Search Index (Meilisearch/Elastic): for full-text and faceted product search.**

IV.4 Class diagram



IV.5 Database Design

IV.1. Principles & Conventions

- **DBMS:** PostgreSQL 15+; **ORM:** Prisma.
- **Primary keys:** UUID (v4) for all tables; standard timestamp columns: `created_at`, `updated_at` (UTC).
- **Foreign keys:** `ON UPDATE CASCADE`, `ON DELETE RESTRICT` (or `SET NULL` selectively, e.g., for media).

- **Naming convention:** `snake_case`; plural table names (e.g., `users`, `orders`).
- **Currency:** `NUMERIC(12,2)`; **state values:** `ENUM` or `TEXT` + check constraint.
- **Soft delete:** discouraged; prefer `status` fields (e.g., `status = banned/unpublished`).
- **Security:** only **store URLs** for media; never store raw payment/PII data.

IV.2 Logical Schema (Key Tables & Columns)

Accounts & Profiles

- **users**
  `id PK, email UNIQUE, phone?, password_hash, role ENUM(shopper|vendor|admin), status ENUM(active|suspended|banned), provider?, provider_id?, created_at, updated_at`

- **customer_profiles**
  `user_id PK/FK→users, display_name, avatar_url?, default_address_id?, preferences JSONB, updated_at`

Stores & Products

- **stores**
  `id PK, owner_user_id FK→users, name, status ENUM(active|suspended), created_at, updated_at`

- **products**
  `id PK, store_id FK→stores, name, description TEXT, category, material?, status ENUM(draft|published|unpublished|banned), created_at, updated_at`

- **product_images**
  `id PK, product_id FK→products, url, alt?`

- **product_variants**
  `id PK, product_id FK→products, sku UNIQUE, size, color, base_price NUMERIC, sale_price?, sale_start?, sale_end?, created_at, updated_at`

- **inventory**
  `variant_id PK/FK→product_variants, on_hand INT, reserved INT, incoming INT, reorder_point INT`

Cart & Orders

- **carts**
  `id PK, user_id FK→users NULLABLE, guest_token UNIQUE?, created_at, updated_at`

- **cart_items**
  `id PK, cart_id FK→carts, variant_id FK→product_variants, quantity INT CHECK>0, price_snapshot NUMERIC, UNIQUE(cart_id, variant_id)`

- **orders**
  `id PK, user_id FK→users, status ENUM(new|pending_payment|paid|packed|shipped|delivered|returned|refunded|cancelled), payment_status ENUM(pending|paid|failed|refunded), subtotal NUMERIC, shipping_fee NUMERIC, discount NUMERIC, total NUMERIC, address_id?, channel TEXT DEFAULT 'aifshop', created_at, updated_at`

- **order_items**
  `id PK, order_id FK→orders, variant_id FK→product_variants, qty INT, unit_price NUMERIC, discount NUMERIC DEFAULT 0`

- **payments**
  `id PK, order_id FK→orders, method TEXT, amount NUMERIC, status ENUM(pending|success|failed|refunded), gateway_ref TEXT, created_at`

- **shipments**
  `id PK, order_id FK→orders, carrier TEXT, tracking_no TEXT, status TEXT, last_scan_at TIMESTAMP`

Interaction & Promotions

- **favorites**
  `user_id FK→users, product_id FK→products, created_at; PK(user_id, product_id)`

- **promotions**
  `id PK, code UNIQUE, discount_type ENUM(percent|fixed), value NUMERIC, start_at, end_at, created_at`

- **promotion_applications**
  `id PK, promotion_id FK→promotions, order_id FK→orders, amount_applied NUMERIC`

Preorder & Quote

- **preorder_requests**
  `id PK, user_id FK→users, variant_id FK→product_variants, status ENUM(pending|approved|rejected), eta DATE?, deposit NUMERIC?, created_at, updated_at`

- **quotes**
  `id PK, user_id FK→users, variant_id FK→product_variants, price NUMERIC, valid_until TIMESTAMP, status ENUM(pending|accepted|rejected|expired)`

AI/Chat & External Channels

- **chat_sessions**
  `id PK, user_id FK→users, created_at`

- **chat_messages**
  `id PK, session_id FK→chat_sessions, role ENUM(user|ai|system), text TEXT, intent?, entities JSONB?, created_at`

- **channel_orders (Shopee/TikTok)**
  `id PK, channel TEXT, external_id TEXT UNIQUE, mapped_order_id FK→orders, status TEXT, payload JSONB, created_at`

System & Logs

- **notifications**
  `id PK, user_id FK→users, type TEXT, payload JSONB, read_at?, created_at`

- **audit_logs**
  `id PK, actor_id FK→users, action TEXT, entity_type TEXT, entity_id UUID?, before JSONB?, after JSONB?, created_at`

IV.3 Constraints & Integrity

Unique/composite keys:

- `users.email` **UNIQUE,** `product_variants.sku` **UNIQUE**

- `favorites (user_id, product_id)` **UNIQUE**
- `cart_items (cart_id, variant_id)` **UNIQUE**

Checks & FSM: enforce state machines (e.g., order lifecycle), ensure `sale_price <= base_price`**.**

Transactions:

- **During checkout:** `SELECT ... FOR UPDATE` **on** `inventory.variant_id` **to avoid overselling.**
- **Payment webhook success: confirm order, adjust** `reserved → on_hand`**.**

IV.4 Suggested Indexes

- `products(status, category, created_at DESC)`
- `product_variants(product_id, sku)`
  `orders(user_id, created_at DESC)`, `orders(status, created_at)`
- `favorites(user_id, product_id)`
- `payments(order_id, status)`
  `shipments(order_id, tracking_no)`
- `chat_messages(session_id, created_at)`
- `audit_logs(created_at)`

IV.5 DDL (PostgreSQL – simplified)

```sql
CREATE TABLE users (
  id UUID PRIMARY KEY,
  email CITEXT UNIQUE NOT NULL,
  phone TEXT,
  password_hash TEXT,
  role TEXT CHECK (role IN ('shopper','vendor','admin')) NOT NULL,
  status TEXT CHECK (status IN ('active','suspended','banned')) NOT NULL DEFAULT 'active',
  provider TEXT, provider_id TEXT,
  created_at TIMESTAMPTZ DEFAULT now(), updated_at TIMESTAMPTZ DEFAULT now()
);

CREATE TABLE products (
  id UUID PRIMARY KEY,
  store_id UUID NOT NULL REFERENCES stores(id),
  name TEXT NOT NULL,
  description TEXT,
  category TEXT,
  material TEXT,
  status TEXT CHECK (status IN ('draft','published','unpublished','banned')) NOT NULL DEFAULT 'd
  created_at TIMESTAMPTZ DEFAULT now(), updated_at TIMESTAMPTZ DEFAULT now()
);

CREATE TABLE product_variants (
  id UUID PRIMARY KEY,
  product_id UUID NOT NULL REFERENCES products(id) ON DELETE CASCADE,
  sku TEXT UNIQUE NOT NULL,
  size TEXT, color TEXT, material TEXT,
  base_price NUMERIC(12,2) NOT NULL,
  sale_price NUMERIC(12,2),
  sale_start TIMESTAMPTZ, sale_end TIMESTAMPTZ,
  created_at TIMESTAMPTZ DEFAULT now(), updated_at TIMESTAMPTZ DEFAULT now()
);

CREATE TABLE inventory (
  variant_id UUID PRIMARY KEY REFERENCES product_variants(id) ON DELETE CASCADE,
  on_hand INT NOT NULL DEFAULT 0,
  reserved INT NOT NULL DEFAULT 0,
  incoming INT NOT NULL DEFAULT 0,
  reorder_point INT NOT NULL DEFAULT 0
);
```

IV.6 Data Lifecycle & Retention

- **Orders: kept ≥ 5 years; logs/audit: ≥ 1 year.**
- **User data deletion: upon request (GDPR-like) → anonymize or mark deleted in reference tables.**
- **Backups: daily snapshots; regular recovery drills.**

V.7 Data Security

- **Encryption: at-rest (disk/backup) + in-transit (TLS).**
- **DB roles: separate app/read-only/admin.**
- **Payment: only store gateway_ref, never PAN/card.**
- **Auditing: append-only `audit_logs` for admin/vendor actions.**

V.8 Index & CDN Integration

- **Search Index: sync `products`/`variants` (name, category, attributes, price, status).**
- **CDN: only store `url`/`public_id` in `product_images`; orphan cleanup jobs.**

V.9 Scalability

- **Partitioning: optional by `created_at` for large `orders`/`audit_logs`.**
- **Read replicas: for reporting & AI recommendation pipelines.**
- **Batch jobs: re-indexing, popularity metrics, bestseller recalculation.**

V. Implementation

V.1 Chosen Architecture Overview

Architecture Style
AIFShop applies a **Layered Architecture + Modular Monolith** approach (modularized by business domains), with the ability to evolve into microservices as traffic grows. The design emphasizes **security** (OAuth2/JWT, RBAC, audit), **performance** (SSR/SSG, caching, job queues), and **horizontal scalability**.

Layers and Main Components

- **Presentation (Web & Mobile Apps):**
    - **Web:** React + Vite, Tailwind CSS (SSR/SSG for SEO on product listings).

    - **Mobile:** React Native for iOS/Android shopping and AI chat.
    - **Functions:** product browsing/search/filter, signup/login (Email/Google/Facebook), profile management, cart & checkout, order history, favorites & reorders, preorder/quote requests, vendor dashboards, and admin moderation tools.

- **Application (API – NestJS, REST):**
    - **Modules:**
        - **Auth & IAM:** OAuth2, JWT, RBAC, account management.
        - **Products & Catalog:** CRUD + product lifecycle (Draft/Published/Unpublished/Banned).
        - **Cart & Orders:** add/remove items, checkout, order lifecycle (New/Paid/Shipped/Delivered/Returned/Refunded), stock reservation logic.
        - **Favorites & Promotions:** favorite toggling, popularity counting, discounts.
        - **Preorders & Quotes:** request/approve/reject workflows.
        - **Chatbot & AI:** LLM-powered Q&A, recommendations, order status queries, promotions.
        - **Vendor Management:** vendor dashboards for catalogs, pricing, and order management.
        - **Admin Moderation:** reporting, approve/hide/ban, audit logging.
        - **Notifications:** email, in-app, push delivery via async workers.
    - Endpoints protected by guards/policies, rate limiting, and input validation.
- **Domain (Entities/Services):**
    - Pure business rules independent of infrastructure.
    - Examples: enforce product/order state machines, prevent overselling with transactions, preorder approval policies, moderation thresholds with auto-flagging.
- **Infrastructure:**

- **Database:** PostgreSQL + Prisma ORM (authoritative ACID data store).
- **Redis:** caching, rate limiting, job queue.
- **Workers:** async jobs for notifications, search indexing, and scheduled tasks.
- **External integrations:**
  - OAuth (Google/Facebook)
  - Payments (Stripe, VNPay, MoMo)
  - Shipping (GHN, GHTK, DHL, VNPost)
  - Email/Push (SES/SendGrid, FCM/OneSignal)
  - AI/LLM (OpenAI/Azure)
  - Media CDN (Cloudinary/S3)
  - Search Index (Meilisearch/Elastic)
- **Observability:** Logging/Tracing (Sentry, Datadog, OpenTelemetry), metrics, and alerts.

Why This Architecture

- **Maintainability & Fast Development:**
  Clear layer separation + domain-based modularization enables independent evolution (e.g., changing UI doesn't affect Domain/DB). Boundaries across modules (Auth, Products, Orders, AI, etc.) shorten feature delivery cycles.

- **Web-first & SEO:**
  Next.js (SSR/SSG) ensures SEO-friendly product catalogs, fast loading, and shareable links—critical for e-commerce discovery.

- **Scalability & Cost Efficiency:**
  Modular Monolith is simple to operate in early stages, but modules (e.g., Notifications, Search) can be extracted into services as load increases. Redis + background workers smooth peak loads (emails, indexing, analytics).

- **Security & Compliance:**
  OAuth2/JWT, RBAC, audit logs; minimal PII collection, opt-in data sharing with vendors; strict DB constraints (FK/unique) + ACID transactions to guarantee

consistency (e.g., no overselling stock).

- **Performance & Reliability:**
  Pagination, caching, image CDN, optional search index for fast queries; fallback to DB queries in degraded mode if Search/Payment/Shipping/CDN services fail.

- **Testability & CI/CD:**
  Clear separation (UI – API – Domain – Infra) supports unit/integration testing. CI/CD pipeline (build, test, lint, scan, deploy) ensures quality and fast release cycles.

V.2. Mapping to Project Structure

Below is the actual folder/package structure for the **AIFShop project**, mapped to the chosen architecture:

```
AIFShop/
├── README.md              # Documentation
├── .env.example           # Environment variables
├── docker-compose.yml     # PostgreSQL, Redis, Mailhog (dev)
├── infra/                 # Infra manifests (prod/staging)
│   └── k8s/               # Kubernetes manifests
├── .github-actions/       # CI/CD pipelines (build/test/lint)

# ================================
# Frontend – Presentation Layer
# ================================
├── apps/web/              # Web App (Next.js + React)
│   ├── next.config.js     # Next.js config
│   ├── public/            # Static assets
│   └── src/
│       ├── app/           # App Router (SSR/SSG)
│       │   ├── (catalog)/ # Product catalog & detail pages
│       │   ├── (auth)/    # Login / Signup (Email, OAuth)
│       │   ├── (profile)/ # Customer profile & settings
│       │   ├── (orders)/  # Order history, reorders
│       │   ├── (cart)/    # Shopping cart
│       │   ├── (vendor)/  # Vendor dashboard (products, sales)
│       │   └── (admin)/   # Admin moderation & reports
│       ├── components/    # UI components (forms, cards, lists)
│       ├── hooks/         # React hooks
│       ├── lib/           # API client, utils, helpers
│       ├── services/      # REST client for backend APIs
│       ├── store/         # Zustand/Context for global state
│       ├── styles/        # Tailwind/SCSS styles
```

```
│       └── index.tsx      # Entry point

# ==============================
# Backend – Application + Domain + Infrastructure
# ==============================
├── apps/api/              # API Server (NestJS)
│   ├── nest-cli.json
│   ├── tsconfig.json
│   └── src/
│       ├── main.ts        # Bootstrap NestJS
│       ├── app.module.ts  # Root module
│       ├── config/        # Env config, CORS, rate-limit, helmet
│       ├── common/        # Guards, interceptors, pipes, DTOs
│       ├── database/      # PrismaService, migrations, seed
│       ├── integrations/  # OAuth, payments, shipping, maps
│       ├── infra/         # Worker, Redis, queues
│       ├── modules/       # Business modules
│       │   ├── auth/      # Auth, JWT, RBAC
│       │   ├── users/     # User, profile
│       │   ├── products/  # Catalog, variants, tags
│       │   ├── cart/      # Cart management
│       │   ├── orders/    # Orders, states, payments, shipments
│       │   ├── favorites/ # Favorite toggle
│       │   ├── preorders/ # Preorder & quotes
│       │   ├── promotions/# Discounts & vouchers
│       │   ├── chatbot/   # AI assistant (LLM integration)
│       │   ├── vendors/   # Vendor management
│       │   ├── admin/     # Moderation & audit
│       │   └── notifications/ # Email, push, templates
│       └── domain/        # Entities, services, policies

# ==============================
# Workers – Background Tasks
# ==============================
├── worker/
│   ├── main.ts            # Worker entry point
│   ├── queues/            # Queue consumers (email, reindex, webhooks)
│   └── jobs/              # Job definitions (async tasks)

# ==============================
# Shared – Cross-cutting
# ==============================
├── packages/shared/
│   ├── dto/               # Shared DTOs, schema validation
│   ├── constants/         # Constants, enums
│   ├── utils/             # Shared helpers
│   └── api-client/        # Typed REST client (for web & worker)
```

Mapping to Architecture

1. **Presentation Layer (Web & Mobile Apps)**
    - **Path:** `/aifshop/apps/web/src`, `/aifshop/apps/mobile/`
    - **Inside:**
        - **Web (Next.js):** `app/` **(catalog, cart, orders, profile, vendor, admin),** `components/`, `services/`, `store/`, `lib/`.
        - **Mobile (React Native):** `app/` **(catalog, cart, checkout),** `components/`, `hooks/`, `store/`.
    - **Responsibilities:**
      Render product catalog & details, manage cart, checkout, profile, order history, favorites, preorder/quote requests; provide vendor dashboards and admin moderation tools. Call backend APIs (REST) for authentication (Email/Google/Facebook), CRUD products, checkout, order tracking, and AI chat support.
    - **Mapped Architecture: Presentation Layer.**
2. **Application Layer (Backend API)**
    - **Path:** `/aifshop/apps/api/src/modules/`
    - **Inside:** `auth/`, `users/`, `products/`, `cart/`, `orders/`, `favorites/`, `preorders/`, `promotions/`, `chatbot/`, `vendors/`, `admin/`, `notifications/`; **shared utilities in** `common/` **(guards/DTO/pipes) and** `config/`.
    - **Responsibilities:**
      Handle all core business logic: authentication & RBAC, product lifecycle, cart & order processing (including payment & shipping orchestration), promotion & favorite toggling, preorder/quote workflows, chatbot requests, vendor management, moderation, and notifications. Coordinate with Domain & Infrastructure layers and return responses.
      Mapped Architecture: Application/Business Logic Layer.
3. **Domain Layer**
    - **Path:** `/aifshop/apps/api/src/domain/`

- **Responsibilities:**
  Encapsulate pure business entities, policies, and services: product & order finite state machines, overselling prevention policies, preorder validation rules, moderation thresholds (auto-flagging). Independent from infrastructure.
- **Mapped Architecture: Domain Layer.**

4. **Data Access & Infrastructure Layer**
   - **Paths:**
     - **Database/ORM: `/aifshop/apps/api/prisma/` (schema.prisma, migrations), `/aifshop/apps/api/src/database/`**
     - **Cache/Queue: `/aifshop/apps/api/src/infra/redis/`, `/aifshop/apps/worker/` (job processing)**
   - **Responsibilities:**
     - **PostgreSQL + Prisma ORM: users, profiles, stores, products, variants, carts, orders, payments, shipments, favorites, preorders, quotes, promotions, chat logs, audit logs.**
     - **Redis: cache, rate limiting, job queue (emails, push notifications, reindexing).**
   - **Mapped Architecture: Data Access & Infrastructure Layer.**
5. **External Services Layer**
   - **Path: `/aifshop/apps/api/src/integrations/` (`oauth/`, `payment/`, `shipping/`, `email/`, `search/`, `storage/`, `maps/`, `llm/`).**
   - **Responsibilities:**
     Integrate with Google/Facebook OAuth, payment gateways (Stripe, VNPay, MoMo), shipping providers (GHN, GHTK, VNPost, DHL), email/push services (SES/SendGrid, FCM/OneSignal), search engines (Meilisearch/Elastic), AI/LLM providers (OpenAI/Azure), media storage/CDN (Cloudinary/S3), and maps/geocoding (Mapbox/Google).
   - **Mapped Architecture: External Services Layer.**
6. **Background Worker (Asynchronous)**
   - **Path: `/aifshop/apps/worker/`**
   - **Responsibilities:**
     Process asynchronous jobs: send emails & push notifications, sync product data with search index, handle payment/shipping webhooks,

run scheduled tasks. Ensures heavy tasks do not block synchronous API requests.

- **Mapped Architecture: Worker/Async Processing Layer.**
