Contents

[Description 1](#_Toc222219626)

[1. Product Vision & Scope 1](#_Toc222219627)

[2. Users, Roles, Permissions, Admin Control 1](#_Toc222219628)

[3. Functional Requirements (MVP) 2](#_Toc222219629)

[4. Non-Functional Requirements 2](#_Toc222219630)

[5. Architecture & Project Structure 3](#_Toc222219631)

[6. Database & Data Modeling 3](#_Toc222219632)

[7. Authentication & Authorization 4](#_Toc222219633)

[8. Storage, State Management, Caching 4](#_Toc222219634)

[9. DevOps & Deployment 4](#_Toc222219635)

[10. Testing, Observability, Performance 4](#_Toc222219636)

[11. API, Integrations, Feature Flags 4](#_Toc222219637)

[12. Payments & Notifications 4](#_Toc222219638)

[13. Analytics, Compliance, Localization 5](#_Toc222219639)

[14. Governance & Maintenance 5](#_Toc222219640)

[15. Edge Cases, Abuse, Scaling, Failure 5](#_Toc222219641)

[16. AI/ML Components 5](#_Toc222219642)

[17. Documentation & Knowledge Base 5](#_Toc222219643)

[Phases 5](#_Toc222219644)

[Phase 0 – Setup & Core Infrastructure 5](#_Toc222219645)

[Phase 1 – User Management & Authentication 6](#_Toc222219646)

[Phase 2 – Product & Catalog Management 6](#_Toc222219647)

[Phase 3 – Order & Checkout Flow 7](#_Toc222219648)

[Phase 4 – Security & Edge Case Handling 7](#_Toc222219649)

[Phase 5 – Notifications & Analytics 7](#_Toc222219650)

[Phase 6 – Admin & Platform Control 8](#_Toc222219651)

[Phase 7 – Testing & Observability 8](#_Toc222219652)

[Granular implementation checklist 8](#_Toc222219653)

[1. User Management & Authentication 8](#_Toc222219654)

[2. Product & Catalog Management 9](#_Toc222219655)

[3. Order & Checkout Flow 10](#_Toc222219656)

[4. Security & Edge Cases 10](#_Toc222219657)

[5. Notifications & Analytics 11](#_Toc222219658)

[6. Admin & Platform Control 11](#_Toc222219659)

[7. Testing & Observability 12](#_Toc222219660)

[8. Deployment & DevOps 12](#_Toc222219661)

[Sprints 15](#_Toc222219662)

[Sprint 1: Project Setup & Core Infrastructure 15](#_Toc222219663)

[Sprint 2: Authentication & Users 15](#_Toc222219664)

[Sprint 3: Product Management 15](#_Toc222219665)

[Sprint 4: Orders & Checkout 16](#_Toc222219666)

[Sprint 5: Admin & Platform Control 16](#_Toc222219667)

[Sprint 6: Security, Middleware & Notifications 16](#_Toc222219668)

[Sprint 7: Observability, Analytics & Testing 17](#_Toc222219669)

[Sprint 8: Final Polishing & Deployment 17](#_Toc222219670)

[Pages 18](#_Toc222219671)

[Buyer/Public Pages 18](#_Toc222219672)

[Seller Pages 19](#_Toc222219673)

[Admin Pages 19](#_Toc222219674)

[System Pages 20](#_Toc222219675)

# Description

## 1. Product Vision & Scope

**Core Problem**

* Buyers in Ethiopia face fragmented discovery of products on social media, inconsistent pricing, and risky payments.
* Small vendors lack technical expertise to create secure online storefronts with payment integrations.
* Admins need a secure, auditable platform to manage sellers and transactions.

**Solution**

* A desktop-first, responsive web marketplace connecting verified buyers and sellers.
* Secure payments via Chapa, inventory management, and admin moderation.
* Focused MVP: discovery, purchase, fulfillment tracking, seller management, and platform security.

**Target Users**

* Buyers: browse, view, and purchase products securely.
* Sellers: manage product catalog, inventory, orders, and bank info.
* Admins: verify sellers, moderate listings, monitor orders, manage security.

**MVP Features**

* Buyer: search & browse, product details, Buy Now + Chapa payment, order history, profile management.
* Seller: product CRUD, image uploads (Cloudinary), inventory tracking, order fulfillment, bank linking.
* Admin: manual seller verification, suspend/ban users, product moderation, global order monitoring, API/security management.

**Out-of-Scope / Deferred**

* Mobile native apps, social features, in-app messaging, AI recommendations, internationalization, automated payouts, advanced analytics, discount engines.

**Success Metrics**

* MVP adoption: 500–1000 users, 20 orders/day, page load <2s, API response <500ms.
* Admin oversight: zero security breaches, verified seller approvals.

## 2. Users, Roles, Permissions, Admin Control

**Roles**

* Buyer, Seller, Admin (superuser)
* RBAC with ownership validation (buyers see only own orders, sellers manage own products, admins override).

**Admin Setup**

* Initial superuser seeded via CLI script.
* Admin dashboard is restricted, requires JWT verification.
* Audit logs for all sensitive admin actions.

**Permissions**

* Buyer: read/search products, checkout, manage profile.
* Seller: CRUD own products, manage orders & bank info, view inventory.
* Admin: verify sellers, suspend users, moderate products, global monitoring, security management.

## 3. Functional Requirements (MVP)

**Buyer**

* Search & browse products (keyword + category)
* Product details view (name, price, stock, images, seller info)
* Checkout via Chapa (direct Buy Now)
* Order history & tracking (Pending/Delivered/Cancelled)
* Account management (name/email/password)

**Seller**

* Product CRUD with Cloudinary image upload
* Simple inventory decrement per sale
* Order status updates (Pending → Delivered)
* Link single bank account

**Admin**

* Manual seller verification
* Suspend/ban users
* Moderate product listings
* Global order monitoring & security logs

**Edge Cases / Failure Handling**

* Race conditions on stock: atomic decrement & 409 error if sold out
* Concurrent edits: optimistic concurrency with versioning
* Admin actions on deleted users: 404 + audit log entry

## 4. Non-Functional Requirements

**Performance**

* Page load <2s, API response <500ms, CRUD writes <800ms
* Concurrent users: 50–100 initial, scaling triggers >500

**Availability**

* Uptime 99.5%, downtime ≤1 hour/month

**Security**

* HTTPS, JWT + Refresh, Helmet, CORS, CSRF, XSS prevention, input validation
* Rate limiting: 3-layer strategy (global, auth, payments)

**Scalability**

* Initial: single instance with 50–100 concurrent users
* Future: auto-scaling for bursts

**Usability**

* Desktop-first, responsive
* Checkout 2–3 steps (Product → Confirm → Payment)

## 5. Architecture & Project Structure

**Type:** Modular monolith, future-ready for microservices.

**Backend**

* Node.js + Express
* PostgreSQL + JSONB for hybrid relational/flexible schema
* Drizzle ORM
* Folder structure: modules, controllers, services, models, routes, middleware, utils, config, jobs, tests

**Frontend**

* React 18 + Vite
* CSR + React Router v6
* Tailwind + Atomic Component pattern
* Zustand for global state, React Query for server state
* Folder structure: assets, components, features, hooks, layouts, pages, services, state, styles, themes, utils, tests

**Repo & Branching**

* Monorepo (frontend + backend)
* Git flow: main, dev, feature/\* branches
* CI/CD with pre-commit hooks, linting, formatting

## 6. Database & Data Modeling

**Core Tables**

* Users (UUID PK, role, verified, JSONB metadata)
* Products (FK owner, stock, status, JSONB attributes)
* Orders (FK buyer/seller, payment\_status, items\_snapshot JSONB)
* Categories, AuditLogs

**Concurrency & Transactions**

* Optimistic locking for edits
* Atomic inventory decrement for purchases
* Strong consistency for payments

**Indexes & Queries**

* Index on product name, category, seller\_id
* JSONB indexes for flexible filtering

## 7. Authentication & Authorization

* JWT + Refresh tokens (access in memory, refresh in HttpOnly cookie)
* Email verification via Nodemailer
* Password hashing with bcrypt/Argon2
* MFA deferred, token expiry enforced

## 8. Storage, State Management, Caching

* Cloudinary for image storage
* CDN caching for product images
* Cart & session handled via Zustand + React Query
* Redis caching deferred for MVP, but ready for search caching

## 9. DevOps & Deployment

* Backend: Railway, frontend: Vercel
* CI/CD: GitHub Actions with linting, testing, build & deploy
* Environment separation: dev, staging, prod
* Secrets managed via environment variables + rotation

## 10. Testing, Observability, Performance

* Unit & integration tests (Jest + React Testing Library)
* Sentry for error monitoring
* Logs structured, retained for 30 days
* Load testing deferred, MVP target metrics tracked via GA4

## 11. API, Integrations, Feature Flags

* REST API
* Chapa payments, Cloudinary, EmailJS
* Feature flags deferred for future experimentation
* Real-time not in MVP

## 12. Payments & Notifications

* One-time payments via Chapa, ETB only
* Email notifications via Nodemailer
* Optional SMS via Twilio / InfoSMS deferred

## 13. Analytics, Compliance, Localization

* GA4 for analytics
* GDPR-lite: email-only user info, deletion on request
* English interface only, i18n libraries in place for future localization

## 14. Governance & Maintenance

* Git flow + code review required
* Modular architecture to minimize technical debt
* Versioned API contracts, ADRs, changelogs

## 15. Edge Cases, Abuse, Scaling, Failure

* CAPTCHA + rate-limiting + heuristic detection for bots
* Account lockout + email notifications for brute-force attempts
* RPO/RTO: daily backups + tested recovery
* Auto-scaling triggers for peak load

## 16. AI/ML Components

* MVP: none
* Future: API-based models with versioning, cost monitoring, and input/output validation

## 17. Documentation & Knowledge Base

* Full SRS/PRD
* User stories, technical architecture, DB schema
* API contracts, ADRs
* Internal onboarding + changelogs

Phases

## Phase 0 – Setup & Core Infrastructure

**Backend**

* Initialize Node.js + Express project
* Setup PostgreSQL database with Drizzle ORM
* Configure environment variables for dev/staging/prod
* Implement CLI script to seed initial admin account (superuser)
* Configure logging (Sentry + structured logs)
* Setup middleware: JWT auth, rate limiting, Helmet, CORS, error handling

**Frontend**

* Initialize React + Vite project
* Tailwind CSS + Atomic component setup
* Routing: React Router v6
* State management: Zustand + React Query for server state
* Environment config for API URLs and feature flags

**DevOps / CI/CD**

* Git repo structure (monorepo)
* Setup GitHub Actions for linting, testing, build, deploy
* Configure Railway hosting for backend, Vercel for frontend
* Define backup strategy for database

## Phase 1 – User Management & Authentication

**Backend Tasks**

* User model: Buyers, Sellers, Admins
* JWT + Refresh token authentication
* Email verification via Nodemailer + Gmail SMTP
* Password hashing (Argon2 / bcrypt)
* Role-based authorization (RBAC) middleware
* Audit logs for admin actions

**Frontend Tasks**

* Login / Signup pages with validation (Zod)
* Email verification flow
* Profile management (update name/email/password)
* Protected routes by role

## Phase 2 – Product & Catalog Management

**Backend**

* Product model with JSONB attributes
* Category model (parent\_id for sub-categories)
* Product CRUD endpoints (with ownership validation)
* Optimistic concurrency control via version column
* Stock decrement logic on purchase (atomic)
* Cloudinary integration for image upload
* Product moderation endpoints for Admin

**Frontend**

* Product listing page (search + category filter)
* Product detail page (images, price, stock, seller info)
* Seller dashboard: Add/Edit/Delete products, view stock
* Admin dashboard: Approve/Reject products, global product view

## Phase 3 – Order & Checkout Flow

**Backend**

* Orders table with items\_snapshot JSONB
* Order CRUD endpoints
* Checkout endpoint integrating Chapa payments
* Payment verification webhook handling
* Order status updates (Pending → Delivered)
* Error handling for sold-out products (409 Conflict)

**Frontend**

* Buy Now flow (2–3 steps: Product → Confirm → Payment)
* Order history page for Buyer
* Seller order dashboard to update status
* Admin global order monitoring page

## Phase 4 – Security & Edge Case Handling

**Backend**

* Rate limiting: multi-layer (global, auth, payments)
* Input validation: Zod + sanitization
* XSS, CSRF protection
* Concurrency handling for edits & purchases
* Logging failed operations for audit

**Frontend**

* Display error messages for conflicts or invalid actions
* Form validation with immediate user feedback
* Loading spinners for Chapa / Cloudinary API calls

## Phase 5 – Notifications & Analytics

**Backend**

* Email notifications: order confirmation, stock alerts
* Optional SMS notifications (Twilio/InfoSMS deferred)

**Frontend**

* Toast notifications for actions (success, error)
* Order updates via API polling

**Analytics**

* GA4 integration for page views, conversions, product clicks

## Phase 6 – Admin & Platform Control

**Backend**

* Admin endpoints for seller verification, user suspension/ban
* Global API health endpoint
* Security configuration endpoints (rate limits, headers)

**Frontend**

* Admin dashboard: verification queue, user management, product moderation
* Audit log display with filtering
* Restricted React interface, JWT-protected

## Phase 7 – Testing & Observability

**Backend**

* Unit tests (Jest) for models, services, controllers
* Integration tests for APIs
* Mock Chapa & Cloudinary during testing
* Sentry for error monitoring

**Frontend**

* Component/unit tests (React Testing Library)
* End-to-end test flow for Buy Now checkout, product CRUD, login

**Performance**

* Ensure page load <2s, API <500ms
* Stress tests for 50–100 concurrent users

**Phase 8 – Deployment & Monitoring**

* CI/CD pipelines for frontend & backend
* Deploy dev/staging/prod to Railway/Vercel
* Database backup & restore scripts
* Monitoring & alerting for downtime or failed payments
* Logging retention & audit policy (30 days)

# Granular implementation checklist

## 1. User Management & Authentication

**Backend Endpoints**

* POST /auth/signup → Buyer/Seller registration
* POST /auth/login → JWT + refresh token
* POST /auth/logout → Invalidate refresh token
* GET /auth/verify-email?token= → Email verification
* POST /auth/resend-verification → Resend token
* GET /users/me → Current user profile
* PUT /users/me → Update profile
* Admin-only:
  + GET /users → List all users
  + PATCH /users/:id/suspend → Suspend user
  + PATCH /users/:id/verify-seller → Verify seller

**Frontend Components / Pages**

* Signup page (Buyer / Seller)
* Login page
* Profile page (update name/email/password)
* Admin dashboard → User management section

**Order of Tasks**

1. DB models for users
2. JWT auth + refresh token
3. Email verification via Nodemailer
4. Admin seed script + restricted dashboard
5. Role-based access middleware

## 2. Product & Catalog Management

**Backend Endpoints**

* POST /products → Add product (Seller)
* PUT /products/:id → Edit product (Seller + Admin override)
* DELETE /products/:id → Delete product (Seller + Admin override)
* GET /products → Search + category filter
* GET /products/:id → Product details
* PATCH /products/:id/status → Admin approval
* GET /categories → Category listing

**Frontend Components / Pages**

* Product listing page (search + category filters)
* Product detail page (images, description, stock, seller info)
* Seller dashboard → Product CRUD
* Admin dashboard → Product moderation queue

**Task Order**

1. DB models for products & categories
2. CRUD endpoints with ownership checks + concurrency control
3. Cloudinary integration for image upload
4. Frontend listing + detail pages
5. Seller dashboard for product management
6. Admin moderation workflow

## 3. Order & Checkout Flow

**Backend Endpoints**

* POST /orders → Create order + Chapa payment init
* POST /orders/verify-payment → Chapa webhook
* GET /orders/history → Buyer orders
* GET /orders/seller → Seller orders
* PATCH /orders/:id/status → Seller updates (Pending → Delivered)
* Admin:
  + GET /orders → Global order monitoring

**Frontend Components / Pages**

* Buy Now checkout modal / page
* Order confirmation page
* Order history page (Buyer)
* Seller order dashboard
* Admin global orders

**Task Order**

1. Orders DB model + items\_snapshot JSONB
2. Checkout endpoint with atomic stock decrement
3. Chapa payment integration + webhook
4. Frontend checkout flow & order pages
5. Admin global order monitoring

## 4. Security & Edge Cases

**Backend**

* Rate limiting (Global, Auth, Payment layers)
* Input validation (Zod) for all endpoints
* XSS/CSRF/SQLi prevention
* Optimistic concurrency control (version column)
* Graceful 404/409 handling for race conditions

**Frontend**

* Conflict messages for sold-out items or concurrent edits
* Loading spinners during payment or uploads
* Form validation on all inputs

**Task Order**

1. Apply global middleware
2. Implement concurrency control & validation
3. Frontend error handling + feedback

## 5. Notifications & Analytics

**Backend**

* Nodemailer: Order confirmation, stock alerts
* Optional: SMS via Twilio/InfoSMS

**Frontend**

* Toast notifications for success/fail actions
* Order update messages

**Analytics**

* GA4: Track page views, product clicks, checkout conversion

**Task Order**

1. Email template creation
2. API integration for notifications
3. Frontend toast implementation
4. GA4 event tracking

## 6. Admin & Platform Control

**Backend**

* Seller verification queue
* User suspension/ban endpoints
* Product moderation endpoints
* Global API health endpoint
* Audit log recording

**Frontend**

* Admin dashboard with tabs:
  + User management
  + Product moderation
  + Order monitoring
  + Security logs

**Task Order**

1. Admin dashboard UI skeleton
2. Backend endpoints + RBAC
3. Audit logging integration
4. Connect dashboard tabs to endpoints

## 7. Testing & Observability

**Backend**

* Unit tests: Controllers, Services, DB Models
* Integration tests: Auth, Product, Order endpoints
* Mock external services: Chapa, Cloudinary
* Sentry setup for runtime error logging

**Frontend**

* Component/unit tests (React Testing Library)
* End-to-end checkout + product flow testing

**Task Order**

1. Setup Jest + React Testing Library
2. Write unit tests for models and services
3. Integration tests for API endpoints
4. E2E tests for critical flows

## 8. Deployment & DevOps

**Tasks**

* CI/CD for frontend & backend
* Railway deployment for backend, Vercel for frontend
* Environment variables setup (dev/staging/prod)
* Database backup & restore scripts
* Monitoring alerts (uptime, failed payments)

**Task Order**

1. Configure GitHub Actions
2. Deploy dev environment
3. Smoke test all endpoints & pages
4. Deploy prod with monitoring

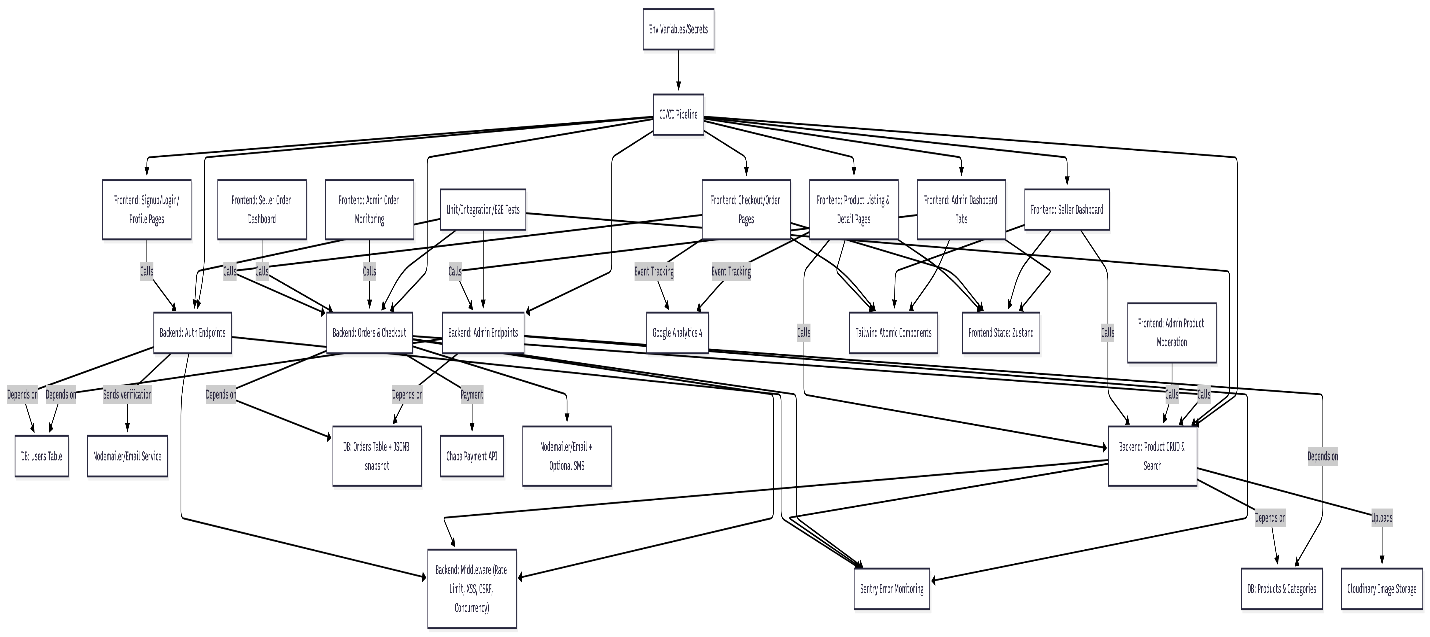


Figure 2: dependency Diagram of Shemsu MVP tasks

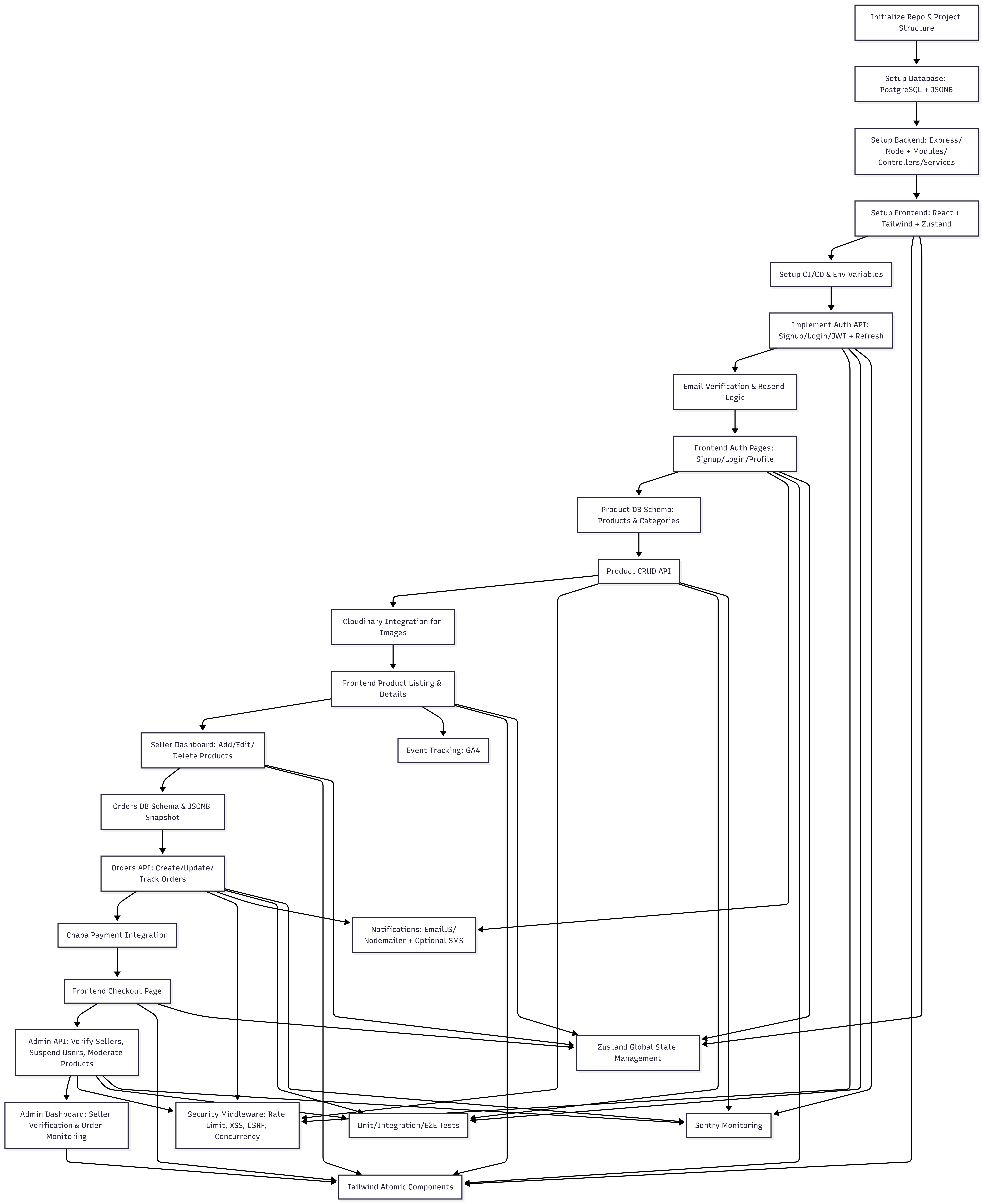


Figure 3: Shemsu MVP task critical path

# Sprints

## Sprint 1: Project Setup & Core Infrastructure

**Goals:** Initialize project, backend, frontend, and CI/CD. Ensure dev environment is fully operational.  
**Tasks:**

* Initialize Git repo and project structure (monorepo: backend + frontend)
* Setup PostgreSQL with JSONB support and create dev/test databases
* Configure Node.js/Express backend structure (Modules, Controllers, Services, Routes)
* Setup React frontend (Vite, Tailwind CSS, Zustand, React Router v6)
* Configure CI/CD (GitHub Actions or Railway/Vercel pipelines)
* Setup environment variables and secret management for dev & prod

**Deliverables:**

* Working dev environment with backend + frontend scaffold
* Database connected and ready for schema creation

## Sprint 2: Authentication & Users

**Goals:** Implement full user auth flows with secure tokens and email verification.  
**Tasks:**

* Implement backend auth API (signup/login, JWT + refresh tokens)
* Email verification flow with token links and resend logic (Nodemailer + Gmail SMTP)
* Password hashing (Argon2/Bcrypt) and MFA placeholder (future)
* Frontend: Signup/Login pages, profile management, session persistence
* Backend middleware for auth, role-based access (Buyer/Seller/Admin)
* Add basic audit logging for user creation and login attempts

**Deliverables:**

* Users can create accounts, verify emails, and log in
* JWT + refresh token system fully functional
* Role-based access enforcement in backend

## Sprint 3: Product Management

**Goals:** Enable sellers to manage products with images, categories, and inventory.  
**Tasks:**

* Create Products & Categories DB schema (including JSONB for attributes)
* Implement Product CRUD API endpoints
* Integrate Cloudinary for image uploads (with validation & cleanup)
* Implement optimistic concurrency/versioning for product edits
* Frontend pages: Seller dashboard, product listing, product details page
* Frontend product forms with validation (Zod)

**Deliverables:**

* Sellers can add, edit, delete products with images
* Product info displayed to buyers
* Versioning prevents overwrites from concurrent edits

## Sprint 4: Orders & Checkout

**Goals:** Allow buyers to purchase products via Chapa with order tracking.  
**Tasks:**

* Create Orders DB schema with JSONB snapshot of items
* Implement Orders API: create, update, track orders
* Integrate Chapa payment gateway
* Handle concurrency for stock decrements at purchase
* Frontend checkout page: direct “Buy Now” flow
* Order history page for buyers

**Deliverables:**

* Buyers can purchase products securely
* Orders update inventory atomically
* Buyers can track their orders

## Sprint 5: Admin & Platform Control

**Goals:** Enable admin verification, moderation, and platform monitoring.  
**Tasks:**

* Admin CLI script for initial superuser creation
* Admin API: verify sellers, suspend users, moderate products, monitor global orders
* Admin dashboard pages (React) with restricted access
* Audit logging of admin actions

**Deliverables:**

* Admin can approve sellers, ban users, moderate products
* Admin actions logged and traceable

## Sprint 6: Security, Middleware & Notifications

**Goals:** Harden platform and setup notifications.  
**Tasks:**

* Security middleware: rate limiting, XSS/CSRF prevention, helmet, concurrency checks
* Input validation everywhere (backend + frontend with Zod)
* Notifications: EmailJS for contact forms and order confirmations
* Optional SMS integration for Ethiopia (InfoSMS/Twilio)

**Deliverables:**

* Platform resistant to attacks
* Users notified of key events (order creation, verification)

## Sprint 7: Observability, Analytics & Testing

**Goals:** Ensure stability, error tracking, and performance monitoring.  
**Tasks:**

* Add Sentry for error tracking
* Setup Google Analytics 4 for event tracking
* Implement unit/integration/E2E tests for critical paths (auth, products, orders)
* Load & stress test backend APIs for MVP concurrency levels

**Deliverables:**

* Errors and anomalies tracked
* Analytics for product views and transactions
* Tests ensure core flows are reliable

## Sprint 8: Final Polishing & Deployment

**Goals:** Deploy MVP and ensure smooth launch.  
**Tasks:**

* Final styling polish with Tailwind + Atomic Components
* Frontend state management finalized (Zustand + React Query)
* Deploy backend (Railway) + frontend (Vercel with custom domain)
* Verify HTTPS, environment variables, secrets, and CI/CD rollback strategies
* QA: Manual testing of all flows

**Deliverables:**

* Fully functional Shemsu MVP live
* Buyers, sellers, and admin can perform their MVP tasks

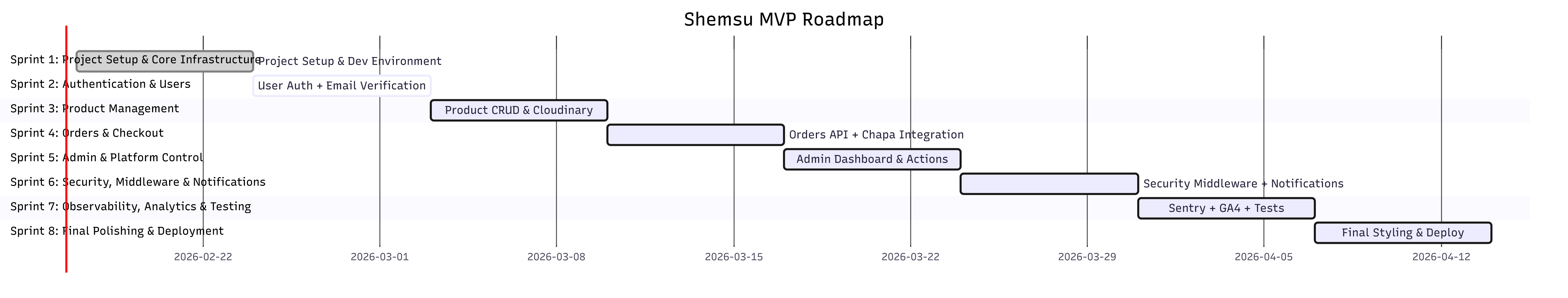


Figure: Full 8-sprint MVP roadmap

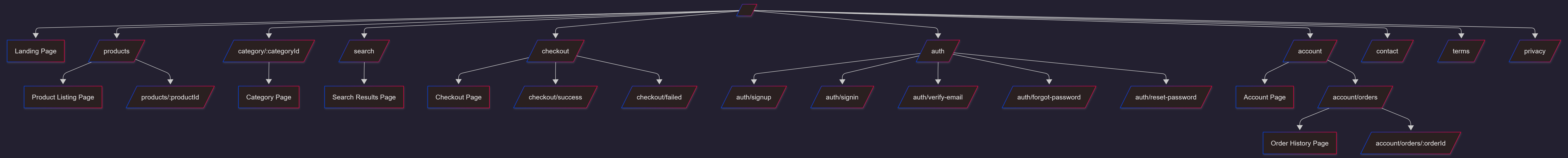
# Pages

## Public Pages

* Welcome Landing Page
* Sign Up Page
* Sign In Page
* Email Verification Page
* Forgot Password Page
* Reset Password Page

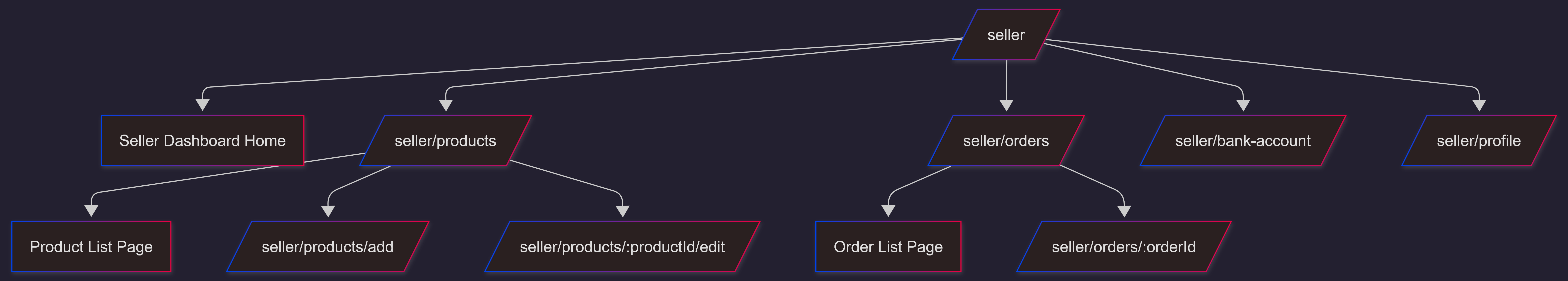
## Buyer/Public Pages

* Buyer Home Page
* Product Listing Page
* Category Page
* Search Results Page
* Product Details Page
* Checkout Page
* Payment Success Page
* Payment Failed Page
* Account Settings Page
* Logout Page
* Order History Page
* Order Details Page
* Contact Us Page
* Terms of Service Page
* Privacy Policy Page



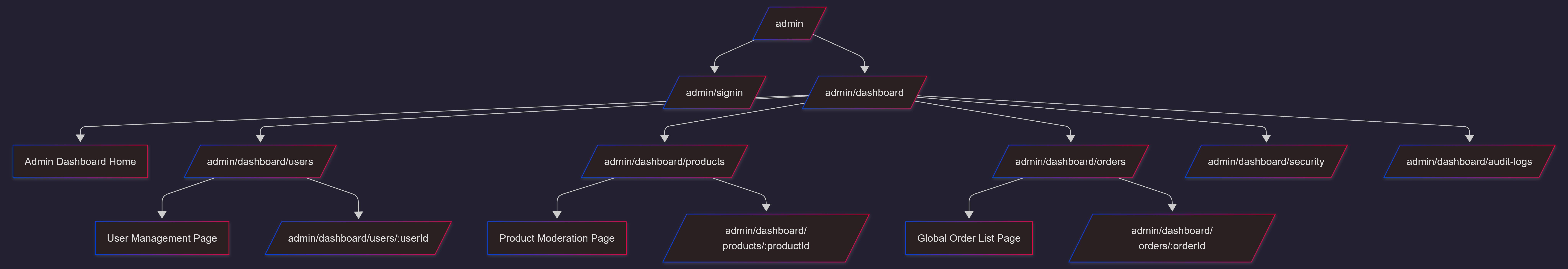
## Seller Pages

* Seller Dashboard Home
* Product List Page
* Add Product Page
* Edit Product Page
* Order List Page
* Seller Order Details Page
* Bank Account Page
* Seller Profile Page



## Admin Pages

* Admin Sign In Page
* Admin Dashboard Home
* Admin Dashboard Settings Page
* User Management Page
* Admin User Details Page
* Product Moderation Page
* Seller Verification Queue Page
* Admin Product Details Page
* Global Order List Page
* Admin Order Details Page
* Admin Reports & Analytics Page
* Admin Seller Payouts Report
* Security & Middleware Page
* Audit Logs Page



## System Pages

* 403 Page
* 404 Page
* 500 Page

