Fansbook Platform

Complete Rebuild Plan

Modern Social Creator-Fan Platform

React + Node.js + TypeScript + PostgreSQL

Mobile-First Responsive Design (Figma-Driven)

Tailwind CSS v3.4 + shadcn/ui + Socket.IO

Version 2.1 | February 2026 | UPDATED

Design: Figma | Functionality: This Document

4 New Sections Added • Performance Budgets • Testing Strategy • Anti-Mess Rules • Risk Resolution

Table of Contents

Section 1: Context & Overview

The current Fansbook platform is a **Laravel 8.83 monolith** with 597+ routes, 66 controllers, and 301 Blade templates. The codebase has become messy and extremely hard to maintain. We are rebuilding from scratch with a modern tech stack and a clean white Figma design. This is a fresh start with no data migration from the old system.

What is Fansbook?

Fansbook is a **creator-fan social platform** where content creators monetize through subscriptions, tips, pay-per-view posts, live streaming, and a marketplace. Fans follow creators, subscribe to premium tiers, chat privately, send tips, bid on marketplace items, and enjoy real-time interactions.

Key Business Features

- **Subscription-based content monetization** (multiple tiers per creator)
- **Pay-per-view (PPV)** individual posts with price lock
- **Tipping system** for posts and messages
- **Real-time private messaging** with media sharing
- **Instagram-style 24-hour disappearing stories**
- **Live streaming** via OBS (RTMP) with live chat overlay
- **1-on-1 video calls** between creators and fans (Vonage/OpenTok)
- **Marketplace** with auction bidding system
- **Gamification** with badges and leaderboards
- **Full admin panel** for platform management and moderation
- **Multi-language support** (i18n) with dark/light theme
- **Payment gateways:** CCBill (Phase 3) + MirexPay (Phase 10)
- **Mobile-first responsive design** following Figma mockups

Design Philosophy

The design follows a **clean, modern white aesthetic** from Figma mockups. Every page is designed mobile-first and scales up to desktop. The Figma file is the single source of truth for visual design — colors, spacing, typography, component styles. **This**

document is the single source of truth for functionality — what each page does, what data it shows, how interactions work.

> ### Two Sources of Truth
>
> **Figma** = How it looks (visual design, spacing, colors, components)
>
> **This Document** = How it works (functionality, data flow, interactions, API)

Section 2: Confirmed Tech Stack UPDATED

Layer	Technology
Frontend	React 18 + Vite + TypeScript (SPA)
UI / CSS	⚡ **Tailwind CSS v3.4** + shadcn/ui (changed from v4 — v4 has breaking changes with shadcn/ui)
State Management	Zustand (client) + TanStack Query v5 (server)
Routing	React Router v6
Forms	React Hook Form + Zod validation
Backend	Node.js 20 LTS + Express 4 + TypeScript
ORM	Prisma 5
Database	PostgreSQL 15+
Cache / Queue	⚡ **Redis 7 + BullMQ (added in Phase 4, not Phase 0)**
Real-time	Socket.IO v4
Auth	Custom JWT (access + refresh) + speakeasy (2FA)
Media Storage	AWS S3 (presigned URLs for direct upload)
Video Calls	Vonage / OpenTok SDK
Email	Nodemailer + AWS SES
i18n	react-i18next
Theme	Dark / Light via Tailwind + CSS variables
Error Monitoring	⚡ **Sentry (from Phase 0 — day 1 error tracking)**
Staging	Hetzner (Nginx + PM2)
Production	AWS (EC2/ECS + RDS + ElastiCache + S3 + CloudFront)

Layer	Technology
CI/CD	GitHub Actions

⚡ Tech Stack Changes from v2.0

- **Tailwind CSS v4 → v3.4:** v4 has breaking changes with shadcn/ui, v3.4 is battle-tested and fully compatible.
- **Redis deferred to Phase 4:** Not needed until real-time chat features, reduces Phase 0 complexity.
- **CCBill only in Phase 3, MirexPay deferred to Phase 10:** One gateway at a time reduces risk and integration complexity.
- **Sentry moved to Phase 0:** Error monitoring from day 1, not an afterthought.

Section 3: Anti-Mess Architecture Rules

NEW

⚡ NEW SECTION — ADDED IN V2.1

The old Laravel codebase became unmaintainable due to lack of enforced standards. These rules are enforced via **ESLint + Husky pre-commit hooks from Phase 0**. No exceptions.

1 Max 200 Lines Per File

ESLint `max-lines` rule. If a file exceeds 200 lines, it must be split. No megacontrollers, no god-files. Forces modular, focused code.

2 No Business Logic in Controllers

Controllers ONLY parse request, call service, return response. All logic lives in services. ESLint custom rule enforces this pattern.

3 Thin Controller → Service → Repository

Mandatory architecture: Controller (HTTP layer) → Service (business logic) → Prisma queries (data layer). No skipping layers.

4 No any Type in TypeScript

tsconfig strict mode ON. ESLint `@typescript-eslint/no-explicit-any = error`. Every variable, parameter, return type must be typed.

5 Shared Zod Schemas = Single Source of Truth

All validation schemas live in `packages/shared`. Frontend and backend import the SAME schemas. No duplicate validation logic.

6 Max Cyclomatic Complexity of 10

ESLint `complexity` rule. Max 1 level of nesting for if/else. Forces clean, readable code with early returns.

7 Pre-Commit Hooks (Husky + lint-staged)

Every commit MUST pass: ESLint (zero errors), TypeScript type-check (`tsc --noEmit`), All tests (`vitest run`). If any fail, commit is blocked.

8 No Debug Code in Codebase

`console.log` / `console.debug` banned via ESLint (`no-console` rule). Use structured logger (winston/pino) with proper log levels. Debug code = instant PR rejection.

Enforcement Mechanism

- **Phase 0:** ESLint config + Husky setup + CI pipeline configured
- **Every PR:** Automated checks must pass before merge
- **Code Review:** Architecture violations = PR rejected
- **No exceptions, no "we'll fix it later"**

Section 4: Performance Budget NEW

⚡ NEW SECTION — ADDED IN V2.1

Built into CI/CD from Phase 0. These are **hard limits, not guidelines**. Violations block deployment.

Metric	Budget	Enforcement
Lighthouse Performance	**≥ 90**	Lighthouse CI in GitHub Actions
Lighthouse Accessibility	**≥ 95**	Lighthouse CI in GitHub Actions
Lighthouse Best Practices	**≥ 95**	Lighthouse CI in GitHub Actions
JS Bundle (initial, gzipped)	**≤ 200KB**	bundlesize in CI
Route Code Splitting	**Every route lazy loaded**	React.lazy + Suspense mandatory
Image Format	**WebP with responsive sizes**	sharp on server, next-gen formats
API Response (reads)	**≤ 200ms**	Sentry performance monitoring
API Response (writes)	**≤ 500ms**	Sentry performance monitoring

Additional Performance Measures

Database Performance

All queries must use indexes. N+1 detection via Prisma query logging. Indexes defined WITH the schema, not as an afterthought.

Virtual Scrolling

`react-virtual` for feed, comments, and any list exceeding 50 items. Prevents DOM bloat and keeps scroll performance smooth.

Image Optimization

`sharp` for server-side processing, WebP format, responsive `srcset` sizes. All uploaded images processed before storage.

CDN

All static assets served via CloudFront CDN. Geographic distribution for fast load times globally.

Caching

HTTP cache headers on static assets (1 year). API responses cached where appropriate. ETags for conditional requests.

Section 5: Testing Strategy NEW

⚡ NEW SECTION — ADDED IN V2.1

Testing was completely missing from the original plan. This section defines minimum testing requirements per phase.

Coverage Requirements

Layer	Minimum Coverage
Services (business logic)	**70%**
Controllers (HTTP layer)	**50%**
Shared Schemas (Zod)	**80%**
React Components (critical)	**50%**

Testing Per Phase

Phase	Testing Requirement
Phase 0	Vitest + React Testing Library setup, CI pipeline running tests, example tests
Phase 1	Unit tests for auth services (register, login, OTP, 2FA), integration tests for auth API endpoints
Phase 2	Unit tests for feed/post services, component tests for PostCard/PostComposer
Phase 3	Payment webhook tests with mocks, wallet transaction tests, CCBill sandbox integration tests
Phase 4	Socket.IO integration tests for chat, message delivery tests
Phase 5+	E2E tests for critical user flows (Playwright), notification delivery tests

Testing Tools

Type	Tool	Purpose
Unit / Integration	**Vitest**	Fast, Vite-native test runner
Component	**React Testing Library**	DOM-based component testing
E2E	**Playwright**	End-to-end browser tests (added Phase 5+)
API	**Supertest**	HTTP assertion library for Express
Mocking	**vitest mocks + MSW**	Mock Service Worker for API mocking

Section 6: Responsive Design Strategy

Breakpoints

Name	Width	Usage
`sm`	640px	Large phones landscape
`md`	768px	Tablets portrait
`lg`	1024px	Tablets landscape / small laptops
`xl`	1280px	Desktop
`2xl`	1536px	Large desktop

Layout Behavior

Mobile (<768px)

- Single column layout
- Bottom navigation bar (5 icons: Home, Explore, Create, Notifications, Profile)
- No sidebar. Hamburger menu for additional nav
- Full-width content. Stacked cards
- Touch-friendly tap targets (min 44px)
- Pull-to-refresh
- Swipe gestures for navigation

Tablet (768–1023px)

- Collapsible sidebar (icon-only mode)
- Top navbar
- Content area takes remaining width
- Cards in 2-column grid where applicable
- Bottom nav hidden, sidebar icons used instead

Desktop (1024px+)

- Full sidebar with labels expanded (left side)
- Top navbar
- Main content in center column (max-width constrained)
- Optional right sidebar for suggestions/trending
- Cards can be in multi-column grid
- Hover states active

Global Responsive Components

Navbar (Top Bar)

- **Desktop:** Logo left, search bar center (expandable), notification bell + message icon + theme toggle + user avatar dropdown right.
- **Tablet:** Logo left, search icon (expands on click), icons right (condensed).
- **Mobile:** Logo left, search icon + notification bell right. Other actions in hamburger menu or bottom nav.
- **[M] Hamburger menu** slides in from right with: Messages, Settings, Theme toggle, Logout.

Sidebar (Left Navigation)

- **Desktop:** Fixed left sidebar, 250px wide. Nav links with icons + labels. "Create Post" CTA button at bottom. Creator section (Dashboard, Analytics, Earnings) shown if user is creator.
- **Tablet:** Collapsed sidebar, 60px wide, icons only. Hover/click to expand temporarily.
- **Mobile:** No sidebar. Navigation via bottom nav bar + hamburger menu.

Bottom Navigation (Mobile Only)

- Fixed at bottom, visible on mobile only (<768px)
- 5 icons: Home, Explore, Create Post (+), Notifications (bell with badge), Profile (avatar)
- Active icon highlighted with primary color
- Create button opens post composer as fullscreen modal on mobile

- **[M]** Bottom nav hides when scrolling down, shows when scrolling up (auto-hide behavior)

Section 7: User Roles & Permissions

Role: FAN (Default)

Every new user starts as a Fan. Full list of permissions:

- Browse public posts
- Follow / unfollow users
- Subscribe to tiers
- View subscriber-only content
- Purchase PPV posts
- Like / comment on posts
- Bookmark posts
- Send tips ($1 minimum)
- Send / receive messages
- View stories
- Watch live streams
- Join video calls
- Browse marketplace / bid
- Manage wallet
- Update profile
- Settings & 2FA
- Block / unblock users
- Report content
- Search users / content

Role: CREATOR

All Fan permissions plus:

- Create posts (5,000 chars + 10 media)
- Set post visibility (public/subscribers/tier)
- Set PPV price on posts
- Pin / unpin posts
- Edit / delete own posts
- Manage tiers (up to 5 tiers, $4.99–$99.99)
- Post stories
- Go live (stream)
- Initiate video calls
- Creator Dashboard
- Analytics page
- Earnings page
- Request withdrawals ($20 minimum)
- Create marketplace listings
- Earn badges
- Leaderboards
- Verification request

Role: ADMIN

Full platform management:

- Admin Panel access
- Dashboard metrics
- User management (view/edit/ban/delete)
- Content moderation
- Review reports
- Payment oversight
- Withdrawal processing
- Platform settings
- Badge management
- Analytics dashboard
- Send announcements
- Audit log access

Section 8: Auth Pages Map

/login

Login Page. Email + password form. "Remember me" checkbox. Links: "Forgot password?", "Don't have an account? Register". Social login buttons (future). On success: redirect to home feed. If 2FA enabled: redirect to /2fa/verify. Mobile: fullscreen form, keyboard-aware scroll.

/register

Registration Page. Display name, email, password, confirm password. Password strength indicator (real-time). Username auto-generated from display name (editable). Terms & Conditions checkbox. On success: send OTP email, redirect to /verify-email. Mobile: stepped form for smaller screens, auto-focus next field.

/verify-email

Email Verification Page. 6-digit OTP input (auto-focus, auto-advance). Countdown timer (5 minutes). "Resend OTP" button (after timer). Shows masked email. On success: redirect to /onboarding. Mobile: large OTP boxes, numpad keyboard.

/forgot-password

Forgot Password Page. Email input. "Send Reset Link" button. Success: show confirmation message. Link to /login. Rate limited (3 attempts per hour). Mobile: simple centered form.

/reset-password

Reset Password Page. Accessed via email link with token. New password + confirm password. Password strength indicator. Token validation on page load. On success: redirect to /login with success message. Mobile: fullscreen form.

/2fa/verify

Two-Factor Authentication Page. 6-digit TOTP code input. "Use backup code" link (switches to backup code input). "Lost access?" help link. Auto-submit on 6th digit. On success: redirect to home feed. Mobile: large code boxes, numpad keyboard.

/onboarding

Onboarding Wizard (4 Steps).

- **Step 1:** Upload avatar + cover photo (crop/resize). Camera capture on mobile.
- **Step 2:** Bio text (500 chars), location (optional), website URL (optional).
- **Step 3:** Select interests (tag chips) from predefined categories. Min 3 interests.
- **Step 4:** Suggested creators to follow (based on interests). Follow buttons. "Skip" option.

Progress bar at top. "Skip" and "Next" buttons. "Back" button after step 1. On complete: redirect to home feed. Mobile: fullscreen steps, swipe between steps.

Section 9: Fan Pages Map

/ (Home Feed)

Main Feed Page. Stories row at top (horizontal scrollable avatars with colored rings). Infinite scroll feed of PostCards from followed creators. Each PostCard: avatar, name, time, text, media (images/video), like/comment/tip/bookmark buttons, comment preview. "New posts" toast when fresh content available (click to scroll to top). Feed algorithm: chronological from followed users + pinned posts. Empty state for new users: "Follow creators to see their posts" with suggestions. Mobile: full-bleed cards, pull-to-refresh, bottom sheet for post actions. [M] Swipe left on post to bookmark, swipe right to like.

/explore

Explore/Discover Page. Search bar at top. Filter tabs: All, Creators, Posts, Hashtags. Masonry grid of content (images/videos). Trending hashtags section. Suggested creators carousel. Category filters (chips). Infinite scroll. Mobile: 2-column masonry, filter chips horizontally scrollable, search expands fullscreen on tap.

/profile/:username

User Profile Page. Cover photo (full-width banner). Avatar (overlapping cover). Display name + @username + verification badge. Bio text. Stats: posts count, followers, following. Action buttons: Follow/Unfollow, Message, Tip, Subscribe. Subscription tiers section (if creator): tier cards with name, price, benefits, subscribe button. Posts tab (grid or list toggle). Media tab (grid of images/videos). Liked tab (own profile only). Mobile: stacked layout, sticky action buttons, tab content scrollable. [M] Pull-down cover photo to see full image.

/post/:id

Single Post Page. Full post with all media displayed. Like/comment/tip/bookmark/share buttons. Comment section: threaded comments, infinite scroll, reply to comments. Related posts from same creator. PPV: if locked, shows blurred preview + price + unlock button. Mobile: fullscreen media viewer, bottom sheet for comments, pinch-to-zoom on images.

/messages

Messages Page. Desktop: split layout — conversation list (left 350px) + active chat (right). Search conversations. Online status dots (green). Unread count badges. Last message preview + timestamp. Sort by recent. Mobile: conversation list only (fullscreen). Tap to navigate to /messages/:conversationId. [M] Swipe left to delete/archive conversation.

/messages/:conversationId

Chat Conversation (Mobile). Fullscreen chat view. Message bubbles (sent right/blue, received left/gray). Typing indicator. Online status in header. Media messages (images, videos). Tip-in-chat button. Emoji picker. Send media button (camera/gallery). Auto-scroll to bottom. Infinite scroll up for history. Mobile: keyboard-aware, auto-resize input, camera/gallery attachment. [M] Long-press message for options (copy, delete, reply).

/notifications

Notifications Page. Grouped notifications: Today, This Week, Earlier. Filter tabs: All, Likes, Comments, Follows, Subscriptions, Tips, System. Each notification: avatar, action text, timestamp, read/unread state. Click to navigate to relevant content. "Mark all as read" button. Mobile: swipe left to dismiss, pull-to-refresh, unread indicator on tab.

/settings/*

Settings Pages. Sidebar navigation (desktop) or stacked list (mobile).

- **/settings/profile:** Edit display name, username, bio, avatar, cover photo, location, website.
- **/settings/account:** Change email, change password, delete account.
- **/settings/privacy:** Profile visibility, message permissions, block list management.
- **/settings/notifications:** Toggle per-type: email and in-app separately for likes, comments, follows, messages, tips, subscriptions, live, system.
- **/settings/security:** Enable/disable 2FA (QR code setup + backup codes), active sessions list, logout all devices.
- **/settings/display:** Theme (dark/light/system), language selection, content preferences.
- **/settings/become-creator:** Apply to become creator. Agreement form. Verification steps.

Mobile: each section is a separate fullscreen page with back navigation.

/wallet

Wallet Page. Balance display (prominent). Add funds button. Transaction history (infinite scroll table). Filters: All, Deposits, Tips Sent, Tips Received, Subscriptions, PPV, Withdrawals. Each transaction: type icon, description, amount (+/-), date, status. Mobile: card-style transactions, filter chips, pull-to-refresh.

Section 10: Creator Pages Map

/creator/dashboard

Creator Dashboard. Overview metrics: total subscribers, total earnings (this month), new followers (this week), post count. Quick stats cards with trend arrows (up/down vs. last period). Recent activity feed (new subs, tips, comments). Quick actions: Create Post, Go Live, View Analytics. Charts: subscriber growth (line), earnings by day (bar). Mobile: stacked metric cards, swipe between charts, quick action FAB.

/creator/analytics

Creator Analytics. Date range picker (7d, 30d, 90d, custom). Engagement metrics: total views, likes, comments, shares. Top performing posts (table with metrics). Audience demographics: age groups, locations (if available). Post performance over time (chart). Best posting times heatmap. Follower growth chart. Mobile: simplified charts, scrollable tables, date range as bottom sheet.

/creator/earnings

Creator Earnings. Total balance (large display). Pending balance (processing). Available for withdrawal. Revenue breakdown: subscriptions, tips, PPV, marketplace. Earnings chart over time. Withdrawal history table (status: pending, processing, completed, failed). Request Withdrawal button (opens form: amount, payment method). Minimum withdrawal: $20. Mobile: card layout for balances, scroll table for history.

/creator/tiers

Manage Subscription Tiers. List of current tiers (up to 5 max). Each tier card: name, price (monthly), benefits list, subscriber count, edit/delete buttons. "Add New Tier" button. Tier form: name, price ($4.99–$99.99), description, benefits (add/remove), tier color/badge. Drag to reorder tiers. Preview how tier appears on profile. Mobile: fullscreen tier editor, stacked tier cards.

/creator/go-live

Go Live Setup Page. Stream title input. Stream description. Category selection. Thumbnail upload. Stream key display (copy button). RTMP URL display. "Start Stream" button (if using browser). Preview panel. Viewer settings: who can watch (all, subscribers, specific tier). Mobile: simplified setup form, stream key with tap-to-copy.

/stories/create

Story Creator. Upload image/video (camera capture on mobile). Text overlay tool. Drawing tool. Sticker/emoji overlay. Filters (basic color filters). Preview before posting. "Post Story" button. Story expires in 24 hours. Mobile: fullscreen camera, swipe filters, tap to add text, pinch to resize stickers.

Section 11: Admin Panel Pages Map

/admin/dashboard

Admin Dashboard. Platform overview metrics: total users, active users (DAU/MAU), total creators, total revenue, pending withdrawals, open reports. Charts: user growth, revenue over time, content posted per day. Recent activity feed. System health indicators. Mobile: stacked metric cards, simplified charts.

/admin/users

User Management. Searchable, filterable user table. Columns: avatar, name, email, role, status, joined date, posts count, subscribers count. Filters: role (fan/creator/admin), status (active/suspended/banned), date range. Actions: view profile, edit, suspend, ban, delete, change role. Bulk actions with checkbox selection. Pagination. Mobile: card layout instead of table, filter as bottom sheet.

/admin/users/:id

User Detail. Full user profile info. Account status with action buttons. Login history. Content posted (recent). Subscription info. Transaction history. Reports filed against this user. Notes field for admin comments. Action buttons: suspend, ban, verify, reset password, impersonate. Mobile: tabbed sections.

/admin/content

Content Moderation. Grid/list view of flagged content. Filters: type (post/comment/message), status (pending/reviewed/removed), report reason. Each item: preview, reporter info, report reason, timestamp. Actions: approve,

remove, warn user, ban user. Quick review mode (one-by-one with keyboard shortcuts). Mobile: card-based review queue.

/admin/reports

Reports Management. Table of user reports. Columns: reporter, reported user/content, reason, status, date. Filters: status (open/investigating/resolved/dismissed), reason category. Detail view with context (reported content + conversation thread). Actions: investigate, resolve, dismiss, take action on reported user. Mobile: card layout, swipe actions.

/admin/payments

Payment Oversight. Transaction log with filters. Columns: transaction ID, user, type, amount, status, gateway, date. Filters: type (subscription/tip/PPV/withdrawal), status, date range, gateway. Summary stats: total revenue, platform fees, pending payouts. Export to CSV. Mobile: simplified table, horizontal scroll.

/admin/withdrawals

Withdrawal Processing. Queue of pending withdrawals. Columns: creator, amount, payment method, requested date, status. Actions: approve, reject (with reason), mark as processed. Batch approval. Filter by status. Mobile: card layout with action buttons.

/admin/settings

Platform Settings. General: site name, description, maintenance mode toggle. Payments: commission percentage, minimum withdrawal, payment gateway config. Content: max upload size, allowed file types, auto-moderation rules. Email: SMTP config, email templates. Security: rate limiting, IP blocking. Mobile: section-based form, collapsible panels.

/admin/badges, /admin/analytics, /admin/announcements, /admin/audit-log

- **/admin/badges:** Manage badges (create/edit/delete). Badge criteria configuration. Assign badges manually. Preview badge appearance.
- **/admin/analytics:** Platform-wide analytics dashboard. Revenue analytics, user analytics, content analytics. Exportable reports.
- **/admin/announcements:** Create/manage platform announcements. Target audience (all users, creators only, specific roles). Schedule announcements. Banner display settings.
- **/admin/audit-log:** Full audit trail. Admin actions log. Filterable by admin user, action type, date range. Immutable records.

Section 12: Shared Pages Map

/search

Global Search. Full-page search with large search input. Real-time results as you type (debounced 300ms). Tabs: All, Users, Posts, Hashtags. Recent searches (saved locally). Trending searches. Result cards with relevance highlighting. Mobile: fullscreen search overlay, keyboard auto-focus, recent/trending suggestions.

/hashtag/:tag

Hashtag Page. Hashtag name as header with post count. Grid of posts containing this hashtag. Sort: recent, popular. Infinite scroll. Follow hashtag button (for feed). Related hashtags. Mobile: 2-column grid, pull-to-refresh.

/live/:sessionId

Live Stream Viewer. Video player (HLS, full-width). Live chat overlay (right sidebar on desktop, bottom sheet on mobile). Viewer count. Like/react animations. Tip button (sends animated tip notification). Creator info header. "Stream ended" state with replay option (if recorded). Mobile: fullscreen video, chat slides up from bottom, landscape support.

/marketplace

Marketplace Browse. Grid of listings. Filters: category, price range, type (fixed/auction), status (active/sold). Search bar. Sort: newest, price low/high, ending soon. Each listing card: image, title, price/current bid, seller, time

remaining (for auctions). Mobile: 2-column grid, filter drawer slides from bottom, sticky search.

/marketplace/:id

Marketplace Listing Detail. Image gallery (swipeable). Title, description. Price or current bid + bid history. Seller info with link to profile. "Buy Now" or "Place Bid" button. Bid form (for auctions): amount input, minimum increment shown. Time remaining countdown. Similar listings. Mobile: fullscreen images, sticky bid/buy bar at bottom.

/marketplace/create

Create Marketplace Listing. Multi-step wizard: Step 1: images upload (up to 10). Step 2: title, description, category. Step 3: pricing (fixed price or auction with starting bid, reserve price, duration). Preview step. "Publish" button. Mobile: fullscreen steps, camera upload.

/leaderboard

Leaderboard Page. Rankings table: rank, avatar, name, score/metric. Filter tabs: Top Creators (by subscribers), Top Earners (by tips received), Most Active (by posts), Rising Stars (new creators with fast growth). Time period: weekly, monthly, all-time. Your rank highlighted. Mobile: card-style rankings, swipeable tabs.

/badges

All Badges Page. Grid of all available badges. Each badge: icon, name, description, criteria, rarity (common/rare/epic/legendary). Earned badges highlighted. Unearned badges grayed with progress indicator. Filter: earned/unearned, category. Mobile: 2-column grid, tap for detail modal.

/404

Not Found Page. Friendly 404 illustration. "Page not found" message. Search bar. Link to home. Suggested popular pages. Mobile: centered, simple layout.

/maintenance

Maintenance Mode Page. Displayed when admin enables maintenance mode. Maintenance illustration. Estimated return time. Contact email. Social media links. Mobile: centered, simple layout.

Section 13: Authentication & Onboarding Module

API Endpoints

Method	Endpoint	Description
POST	`/api/auth/register`	Register new user (email, password, displayName)
POST	`/api/auth/verify-email`	Verify email with 6-digit OTP code
POST	`/api/auth/resend-otp`	Resend OTP to email (rate limited)
POST	`/api/auth/login`	Login with email + password, returns JWT tokens
POST	`/api/auth/refresh`	Refresh access token using refresh token
POST	`/api/auth/logout`	Invalidate refresh token, clear cookies
POST	`/api/auth/forgot-password`	Send password reset email
POST	`/api/auth/reset-password`	Reset password with token
POST	`/api/auth/2fa/enable`	Generate 2FA secret + QR code
POST	`/api/auth/2fa/verify`	Verify TOTP code and enable 2FA
POST	`/api/auth/2fa/disable`	Disable 2FA (requires current TOTP)
POST	`/api/auth/2fa/validate`	Validate 2FA during login
GET	`/api/auth/me`	Get current authenticated user

Backend Logic

- **Password hashing:** bcrypt with 12 salt rounds
- **JWT:** Access token (15min), Refresh token (7 days, stored in DB + httpOnly cookie)

- **OTP:** 6-digit numeric, valid 5 minutes, single-use, stored hashed in DB
- **Rate limiting:** Login: 5 attempts per 15 min. OTP: 3 resends per hour. Password reset: 3 per hour
- **2FA:** speakeasy library, TOTP with 30-second window. Backup codes: 10 codes generated, hashed, single-use
- **Username generation:** From display name, check uniqueness, append random digits if needed
- **Onboarding:** Stored as step completion flags on user record. Can be skipped and resumed later

Frontend Components

- `LoginForm` — Email + password form with validation
- `RegisterForm` — Registration with password strength meter
- `OtpInput` — 6-digit auto-advance input boxes
- `TwoFactorSetup` — QR code display + verification
- `OnboardingWizard` — Multi-step wizard with progress
- `AvatarUpload` — Crop and resize image upload
- `ProtectedRoute` — HOC that checks auth state
- `useAuth` — Custom hook for auth state + actions
- `authStore` — Zustand store for user, tokens, loading state

Section 14: Feed & Content System

API Endpoints

Method	Endpoint	Description
GET	/api/feed	Get personalized feed (posts from followed users)
GET	/api/feed/explore	Get explore/discover feed (trending, suggested)
POST	/api/posts	Create new post (text + media)
GET	/api/posts/:id	Get single post with details
PUT	/api/posts/:id	Edit post (text only, within 24h)
DELETE	/api/posts/:id	Delete post (soft delete)
POST	/api/posts/:id/pin	Pin post to profile top
DELETE	/api/posts/:id/pin	Unpin post
POST	/api/posts/:id/like	Like a post
DELETE	/api/posts/:id/like	Unlike a post
POST	/api/posts/:id/bookmark	Bookmark a post
DELETE	/api/posts/:id/bookmark	Remove bookmark
GET	/api/posts/:id/comments	Get comments on post (paginated)
POST	/api/posts/:id/comments	Add comment to post
DELETE	/api/comments/:id	Delete comment (author or post owner)
POST	/api/upload/presign	Get presigned S3 URL for direct upload

Backend Logic

- **Post creation:** Max 5,000 chars text, up to 10 media files. Visibility: public, subscribers-only, specific-tier. PPV option with price ($1–$500).
- **Feed algorithm:** Chronological from followed users. Pinned posts shown first. Paginated with cursor-based pagination (20 per page).
- **Explore algorithm:** Trending posts (likes + comments in last 24h weighted). Suggested creators (based on follows-of-follows). Category-based discovery.
- **Media upload:** Presigned S3 URLs generated server-side. Client uploads directly to S3. Server validates completion. Image processing (resize, WebP conversion) via sharp.
- **Visibility enforcement:** Middleware checks subscription status before returning content. PPV content returns blurred thumbnail + price until purchased.
- **Soft delete:** Posts marked as deleted, not removed from DB. Hidden from feeds. Admin can see deleted content.

Frontend Components

- `PostComposer` — Rich text + media upload + visibility selector + PPV price
- `PostCard` — Full post display with actions (like, comment, tip, bookmark)
- `FeedList` — Infinite scroll container with virtual scrolling
- `MediaGallery` — Grid/carousel display of post media with lightbox
- `CommentSection` — Threaded comments with reply, load more
- `BlurredOverlay` — PPV/subscription lock overlay with unlock button
- `ExploreGrid` — Masonry layout for explore page
- `InfiniteScroll` — Reusable infinite scroll wrapper with loading states

Section 15: Social Graph

API Endpoints

Method	Endpoint	Description
POST	`/api/users/:id/follow`	Follow a user
DELETE	`/api/users/:id/follow`	Unfollow a user
GET	`/api/users/:id/followers`	Get user's followers list (paginated)
GET	`/api/users/:id/following`	Get user's following list (paginated)
POST	`/api/users/:id/block`	Block a user
DELETE	`/api/users/:id/block`	Unblock a user
GET	`/api/users/blocked`	Get blocked users list
POST	`/api/reports`	Report a user or content
GET	`/api/users/:username`	Get user profile by username
PUT	`/api/users/profile`	Update own profile
GET	`/api/users/suggestions`	Get suggested users to follow

Backend Logic

- **Follow:** Creates Follow record. Increments follower/following counts (denormalized). Triggers notification to followed user.
- **Block:** Creates Block record. Auto-unfollows in both directions. Blocked user cannot view profile, posts, or send messages. Existing conversations hidden.
- **Report:** Creates Report record with reason (spam, harassment, nudity, copyright, other). Admin notified. Reported content flagged for review.
- **Suggestions:** Based on follows-of-follows algorithm. Excludes blocked users. Weighted by engagement (more active creators ranked higher).
- **Profile:** Display name, username (unique, alphanumeric + underscore), bio (500 chars), avatar URL, cover URL, location, website.

Frontend Components

- `FollowButton` — Toggle follow/unfollow with optimistic UI update
- `UserCard` — Compact user display (avatar, name, bio snippet, follow button)
- `ProfileHeader` — Cover photo, avatar, stats, action buttons
- `UserList` — Paginated list of users (followers, following, blocked)
- `ReportModal` — Report form with reason selection and description
- `SuggestionCarousel` — Horizontal scrollable creator suggestions

Section 16: Subscriptions & Payments

UPDATED

> ⚡ **Payment Gateway Update (v2.1)**
>
> **Phase 3:** CCBill integration only. **Phase 10:** MirexPay integration deferred. One payment gateway at a time reduces risk and integration complexity.

API Endpoints

Method	Endpoint	Description
GET	`/api/creators/:id/tiers`	Get creator's subscription tiers
POST	`/api/tiers`	Create subscription tier (creator only)
PUT	`/api/tiers/:id`	Update subscription tier
DELETE	`/api/tiers/:id`	Delete subscription tier
POST	`/api/subscriptions`	Subscribe to a creator's tier
DELETE	`/api/subscriptions/:id`	Cancel subscription
GET	`/api/subscriptions`	Get user's active subscriptions
POST	`/api/tips`	Send tip to creator (post or message)
POST	`/api/posts/:id/unlock`	Purchase PPV post
GET	`/api/wallet`	Get wallet balance and summary
GET	`/api/wallet/transactions`	Get transaction history (paginated)
POST	`/api/wallet/deposit`	Initiate deposit (redirect to payment gateway)
POST	`/api/withdrawals`	Request withdrawal (creator, min $20)

Method	Endpoint	Description
GET	`/api/withdrawals`	Get withdrawal history
POST	`/api/webhooks/ccbill`	CCBill webhook handler (Phase 3)
POST	`/api/webhooks/mirexpay`	MirexPay webhook handler PHASE 10 - DEFERRED

Backend Logic

- **Tiers:** Up to 5 per creator. Price range $4.99–$99.99/month. Name, description, benefits list, badge color. Cannot delete tier with active subscribers (must migrate first).
- **Subscriptions:** Monthly recurring. Grace period (3 days) for failed payments. Auto-renewal via webhook. Access revoked on expiry.
- **Wallet:** Internal balance system. Deposits via payment gateway. All transactions logged. Platform takes commission (configurable, default 20%). Creator earnings = gross - platform commission.
- **Tips:** Minimum $1, maximum $500. Sent on posts or in messages. 100% goes to creator (minus platform commission). Animated notification to creator.
- **PPV:** One-time purchase. Price set by creator ($1–$500). Once purchased, permanent access. Blurred preview shown to non-purchasers.
- **Withdrawals:** Minimum $20. Processing time: 3–5 business days. Admin approval required. Methods: bank transfer, configurable.
- **Webhooks:** Idempotent handlers. Signature verification. Transaction logging for debugging. Retry handling.

Frontend Components

- `TierCard` — Subscription tier display with price, benefits, subscribe button
- `TierManager` — Creator CRUD interface for managing tiers
- `SubscribeModal` — Subscription checkout flow
- `TipModal` — Tip amount input with preset amounts ($1, $5, $10, $25, custom)
- `WalletPage` — Balance display, transaction history, deposit/withdraw
- `EarningsDashboard` — Revenue charts, breakdown, withdrawal management
- `PPVUnlockButton` — Purchase button with price and confirmation

- `TransactionList` — Filterable, paginated transaction table

Section 17: Real-time Chat

API Endpoints

Method	Endpoint	Description
GET	`/api/conversations`	Get conversation list (paginated, sorted by recent)
POST	`/api/conversations`	Create or get existing conversation with user
GET	`/api/conversations/:id/messages`	Get messages in conversation (paginated, cursor-based)
POST	`/api/conversations/:id/messages`	Send message (text, media, tip)
DELETE	`/api/messages/:id`	Delete message (for self or both)
POST	`/api/conversations/:id/read`	Mark conversation as read

Socket.IO Events

Event	Direction	Description
`message:send`	Client → Server	Send new message
`message:new`	Server → Client	New message received
`message:read`	Bidirectional	Message read receipt
`typing:start`	Client → Server	User started typing
`typing:stop`	Client → Server	User stopped typing
`typing:indicator`	Server → Client	Show typing indicator to other user
`user:online`	Server → Client	User came online
`user:offline`	Server → Client	User went offline

Backend Logic

- **Socket.IO:** JWT authentication on connection handshake. User joins personal room (user:{id}). Conversations as rooms.
- **Redis adapter:** Socket.IO Redis adapter for multi-instance support (added Phase 4).
- **Online status:** Tracked via Redis SET. Presence broadcasted to contacts on connect/disconnect.
- **Messages:** Stored in PostgreSQL. Real-time delivery via Socket.IO. Fallback to REST polling if WebSocket disconnected.
- **Media messages:** Presigned S3 upload, then send message with media URL.
- **Tip in chat:** Special message type with tip amount. Triggers payment processing.
- **Message permissions:** Blocked users cannot send messages. Creators can set message pricing (paid messages).

Frontend Components

- `ChatPage` — Split layout (desktop) or fullscreen (mobile)
- `ConversationList` — List of conversations with search, online indicators
- `ChatWindow` — Active conversation with messages
- `MessageBubble` — Individual message (text, media, tip)
- `ChatInput` — Message input with emoji picker, media attach, tip button
- `TypingIndicator` — Animated dots showing other user is typing
- `OnlineStatus` — Green dot indicator on user avatars
- `useSocket` — Custom hook for Socket.IO connection management

Section 18: Notifications

API Endpoints

Method	Endpoint	Description
GET	`/api/notifications`	Get notifications (paginated, filterable)
GET	`/api/notifications/unread-count`	Get unread notification count
POST	`/api/notifications/:id/read`	Mark single notification as read
POST	`/api/notifications/read-all`	Mark all notifications as read
GET	`/api/notifications/preferences`	Get notification preferences
PUT	`/api/notifications/preferences`	Update notification preferences

Notification Types

Type	Trigger	Template
LIKE	Someone likes your post	"{user} liked your post"
COMMENT	Someone comments on your post	"{user} commented on your post"
FOLLOW	Someone follows you	"{user} started following you"
SUBSCRIBE	Someone subscribes to your tier	"{user} subscribed to {tier}"
TIP	Someone sends you a tip	"{user} tipped you ${amount}"
MESSAGE	New message received	"{user} sent you a message"
LIVE	Followed creator goes live	"{creator} is now live!"
STORY	Followed creator posts a story	"{creator} posted a new story"
MENTION	Someone mentions you	"{user} mentioned you in a post"

Type	Trigger	Template
SYSTEM	Platform announcement	Custom message from admin

Backend Logic

- **Creation:** NotificationService creates notification record in DB. Checks user preferences before creating.
- **Real-time delivery:** Socket.IO pushes notification to connected user immediately.
- **Email delivery:** BullMQ queue processes email notifications asynchronously (added Phase 5). Respects email preferences. Uses SES with templates.
- **Grouping:** Multiple likes on same post grouped ("User1, User2, and 3 others liked your post").
- **Preferences:** Per-type toggle for in-app and email separately. Default: all in-app ON, email only for follows/subscriptions/tips.

Frontend Components

- `NotificationBell` — Navbar icon with unread count badge
- `NotificationDropdown` — Quick preview dropdown (last 10 notifications)
- `NotificationPage` — Full page with filters and grouping
- `NotificationItem` — Single notification with avatar, action text, timestamp
- `NotificationPreferences` — Settings page for per-type toggles
- `Toast` — Pop-up notification toast for real-time alerts

Section 19: Stories

API Endpoints

Method	Endpoint	Description
GET	`/api/stories/feed`	Get stories from followed users (grouped by user)
POST	`/api/stories`	Create new story (image/video + overlays)
GET	`/api/stories/:id`	Get single story with view count
DELETE	`/api/stories/:id`	Delete story early
POST	`/api/stories/:id/view`	Record story view
GET	`/api/stories/:id/viewers`	Get story viewers list (creator only)

Backend Logic

- **Creation:** Upload media to S3 via presigned URL. Store story record with media URL, overlays (text, stickers as JSON), created timestamp.
- **Expiry:** Stories expire after 24 hours. Cron job runs every 15 minutes to soft-delete expired stories. BullMQ scheduled job (Phase 5+).
- **Views:** Tracked per user (unique views only). Creator can see viewer list. View count displayed.
- **Feed:** Stories grouped by user. Users with unseen stories shown first. Colored ring indicates unseen story.
- **Visibility:** Stories follow same visibility as creator's account (public or subscribers-only).

Frontend Components

- `StoryRing` — Circular avatar with gradient ring (unseen = colored, seen = gray)
- `StoryRow` — Horizontal scrollable row of StoryRings (top of feed)
- `StoryViewer` — Fullscreen story display with auto-advance, tap to next/prev, progress bars at top

- `StoryCreator` — Fullscreen story creation with camera capture, text overlay, stickers, filters
- `StoryProgress` — Segmented progress bar showing story segments

Section 20: Live Streaming & Video Calls

API Endpoints

Method	Endpoint	Description
POST	`/api/live/start`	Start live stream (returns stream key + RTMP URL)
POST	`/api/live/stop`	End live stream
GET	`/api/live/active`	Get list of active live streams
GET	`/api/live/:sessionId`	Get live session details (HLS URL, viewer count)
POST	`/api/video-calls/initiate`	Initiate video call with user (returns Vonage session)
POST	`/api/video-calls/:id/accept`	Accept incoming video call
POST	`/api/video-calls/:id/reject`	Reject incoming video call
POST	`/api/video-calls/:id/end`	End active video call

Backend Logic

- **Live Streaming:** Creator starts stream from /creator/go-live. Server generates unique stream key. Creator uses OBS to push RTMP stream. Server converts to HLS for viewers. Live chat via Socket.IO room. Viewer count tracked in real-time. Tips during live stream supported.
- **Stream key:** Unique per session, invalidated on stream end. RTMP URL provided for OBS configuration.
- **Live chat:** Socket.IO room per live session. Messages broadcast to all viewers. Rate limited (1 message per 2 seconds). Moderation controls for creator.
- **Video Calls:** Vonage/OpenTok SDK for WebRTC. Creator initiates call, fan accepts/rejects via Socket.IO signaling. Session duration tracked. Video call recording optional (creator setting).

- **Call signaling:** Socket.IO events for call initiation, acceptance, rejection, ending. Ringing timeout (30 seconds).

Frontend Components

- `GoLivePage` — Stream setup form with key/URL display
- `LiveViewer` — HLS video player with live chat sidebar
- `LiveChat` — Real-time chat component for live streams
- `ViewerCount` — Animated viewer count display
- `VideoCallUI` — Vonage video call interface (local + remote video)
- `IncomingCallModal` — Incoming call notification with accept/reject
- `CallControls` — Mute, camera toggle, end call buttons

Section 21: Marketplace & Gamification

Marketplace API Endpoints

Method	Endpoint	Description
GET	`/api/marketplace`	Browse listings (paginated, filterable)
POST	`/api/marketplace`	Create new listing (creator only)
GET	`/api/marketplace/:id`	Get listing details with bid history
PUT	`/api/marketplace/:id`	Update listing (before any bids)
DELETE	`/api/marketplace/:id`	Delete listing (before any bids)
POST	`/api/marketplace/:id/bid`	Place bid on auction listing
POST	`/api/marketplace/:id/buy`	Buy Now on fixed-price listing

Marketplace Backend Logic

- **Listings:** Two types: fixed-price and auction. Up to 10 images. Categories: digital content, physical merchandise, experiences, custom content, shoutouts.
- **Auctions:** Starting bid, optional reserve price, duration (1–30 days). Minimum bid increment ($1 or 5%). Auto-close at end time. Winner notification. Anti-sniping: extend 5 minutes if bid in last 5 minutes.
- **Buy Now:** Immediate purchase. Deducts from wallet. Listing marked as sold. Seller notified.
- **Bidding:** Requires sufficient wallet balance (bid amount held). Outbid users have hold released. Winning bid charged on auction close.

Gamification API Endpoints

Method	Endpoint	Description
GET	`/api/badges`	Get all available badges

Method	Endpoint	Description
GET	`/api/users/:id/badges`	Get user's earned badges
GET	`/api/leaderboard`	Get leaderboard rankings (filterable)

Gamification Backend Logic

- **Badges:** Earned automatically based on criteria. Categories: content (post milestones), social (follower milestones), engagement (like/comment milestones), revenue (earning milestones), special (admin-awarded).
- **Rarity levels:** Common, Rare, Epic, Legendary. Displayed on profile with rarity glow effect.
- **Badge engine:** Event-driven. On relevant actions (post created, follower milestone), check badge criteria. Award if met. Notification sent.
- **Leaderboard:** Rankings by: subscribers count, tips received, posts created, engagement rate. Time periods: weekly, monthly, all-time. Creator-only leaderboards.

Frontend Components

- `MarketplaceBrowse` — Filterable grid of listing cards
- `ListingCard` — Image, title, price/bid, seller, time remaining
- `ListingDetail` — Full listing with gallery, bid history, action buttons
- `BidForm` — Bid amount input with validation and confirmation
- `CreateListingWizard` — Multi-step listing creation form
- `BadgeGrid` — Grid of earned/available badges with rarity styling
- `BadgeCard` — Single badge display with icon, name, description, progress
- `LeaderboardTable` — Rankings with rank, avatar, name, score, filter tabs

Section 22: Admin Panel Functions

API Endpoints

Method	Endpoint	Description
GET	`/api/admin/dashboard`	Get dashboard metrics and charts data
GET	`/api/admin/users`	Get users list (paginated, filterable, searchable)
GET	`/api/admin/users/:id`	Get detailed user info for admin
PATCH	`/api/admin/users/:id`	Update user (role, status, verification)
POST	`/api/admin/users/:id/suspend`	Suspend user (with duration and reason)
POST	`/api/admin/users/:id/ban`	Ban user permanently (with reason)
GET	`/api/admin/content`	Get flagged content for moderation
POST	`/api/admin/content/:id/action`	Take action on content (approve, remove, warn)
GET	`/api/admin/reports`	Get reports list (filterable by status, type)
PATCH	`/api/admin/reports/:id`	Update report status (resolve, dismiss)
GET	`/api/admin/payments`	Get all transactions with filters
GET	`/api/admin/withdrawals`	Get pending withdrawals
POST	`/api/admin/withdrawals/:id/approve`	Approve withdrawal request
POST	`/api/admin/withdrawals/:id/reject`	Reject withdrawal (with reason)

Method	Endpoint	Description
GET	`/api/admin/settings`	Get platform settings
PUT	`/api/admin/settings`	Update platform settings
POST	`/api/admin/badges`	Create new badge
PUT	`/api/admin/badges/:id`	Update badge criteria/display
DELETE	`/api/admin/badges/:id`	Delete badge
GET	`/api/admin/analytics`	Get platform analytics data
POST	`/api/admin/announcements`	Create platform announcement
GET	`/api/admin/audit-log`	Get audit log entries (paginated, filterable)

Backend Logic

- **Authorization:** All admin endpoints protected by role middleware (`requireRole('ADMIN')`). 403 for non-admin users.
- **Audit logging:** Every admin action logged: admin user, action type, target entity, timestamp, IP address, details. Immutable records (no delete/edit).
- **Dashboard metrics:** Cached (5 min) aggregated queries. Total users, new users (24h/7d/30d), active users (DAU/MAU), total revenue, pending withdrawals, open reports.
- **User management:** Search by name/email/username. Filter by role, status, date range. Bulk actions supported. Impersonation for debugging (with audit log).
- **Content moderation:** Queue-based review. Priority sorting (most reported first). Quick actions with keyboard shortcuts. Context preservation (see full conversation/thread).
- **Withdrawal processing:** Manual approval workflow. Admin can batch approve. Rejected withdrawals: funds returned to creator wallet with reason notification.

Section 23: Monorepo Structure

```
fansbook/
├── package.json                    # Root workspace config
├── turbo.json                      # Turborepo config (optional)
├── tsconfig.base.json              # Shared TypeScript config
├── .eslintrc.js                    # Root ESLint config (strict rules)
├── .prettierrc                     # Prettier config
├── .husky/                         # Git hooks
│   └── pre-commit                  # lint-staged trigger
├── .github/
│   └── workflows/
│       ├── ci.yml                  # CI pipeline (lint, test, build, Lighth
│       └── deploy.yml              # Deployment pipeline
│
├── packages/
│   └── shared/
│       ├── package.json
│       ├── tsconfig.json
│       └── src/
│           ├── index.ts            # Barrel export
│           ├── types/              # Shared TypeScript interfaces
│           │   ├── user.ts
│           │   ├── post.ts
│           │   ├── chat.ts
│           │   ├── payment.ts
│           │   └── ...
│           ├── schemas/            # Shared Zod validation schemas
│           │   ├── auth.schema.ts
│           │   ├── post.schema.ts
│           │   ├── user.schema.ts
│           │   ├── payment.schema.ts
│           │   └── ...
│           ├── constants/          # Shared constants and enums
│           │   ├── roles.ts
│           │   ├── limits.ts
│           │   └── ...
│           └── utils/              # Shared utility functions
│               ├── format.ts
│               ├── validation.ts
│               └── ...
│
├── apps/
```

```
│   ├── web/
│   │   ├── package.json
│   │   ├── tsconfig.json
│   │   ├── vite.config.ts
│   │   ├── tailwind.config.ts      # Tailwind v3.4 config
│   │   ├── index.html
│   │   ├── public/
│   │   │   ├── locales/            # i18n translation files
│   │   │   │   ├── en/
│   │   │   │   └── ...
│   │   │   └── ...
│   │   └── src/
│   │       ├── main.tsx            # App entry point
│   │       ├── App.tsx             # Root component + router
│   │       ├── components/         # Reusable UI components
│   │       │   ├── ui/             # shadcn/ui components
│   │       │   ├── layout/         # Navbar, Sidebar, MobileNav
│   │       │   ├── post/           # PostCard, PostComposer
│   │       │   ├── chat/           # ChatWindow, MessageBubble
│   │       │   ├── profile/        # ProfileHeader, FollowButton
│   │       │   └── ...
│   │       ├── pages/              # Route page components
│   │       │   ├── auth/           # Login, Register, Verify, etc.
│   │       │   ├── feed/           # Home, Explore
│   │       │   ├── profile/        # Profile, Settings
│   │       │   ├── creator/        # Dashboard, Analytics, Earnings
│   │       │   ├── chat/           # Messages
│   │       │   ├── admin/          # Admin panel pages
│   │       │   └── ...
│   │       ├── hooks/              # Custom React hooks
│   │       │   ├── useAuth.ts
│   │       │   ├── useSocket.ts
│   │       │   ├── useInfiniteScroll.ts
│   │       │   └── ...
│   │       ├── stores/             # Zustand stores
│   │       │   ├── authStore.ts
│   │       │   ├── themeStore.ts
│   │       │   └── ...
│   │       ├── services/           # API service functions
│   │       │   ├── api.ts          # Axios instance + interceptors
│   │       │   ├── authService.ts
│   │       │   ├── postService.ts
│   │       │   └── ...
│   │       ├── lib/                # Utilities and config
│   │       │   ├── i18n.ts
│   │       │   ├── socket.ts
│   │       │   └── ...
│   │       └── styles/
│   │           └── globals.css     # Tailwind directives + CSS variables
│   │
```

```
│   └── server/
│       ├── package.json
│       ├── tsconfig.json
│       ├── prisma/
│       │   ├── schema.prisma       # Database schema (28 models)
│       │   ├── migrations/         # Prisma migrations
│       │   └── seed.ts             # Database seed script
│       └── src/
│           ├── index.ts            # Server entry point
│           ├── app.ts              # Express app setup
│           ├── config/             # Environment and app config
│           │   ├── env.ts
│           │   ├── database.ts
│           │   └── ...
│           ├── middleware/          # Express middleware
│           │   ├── auth.ts         # JWT verification
│           │   ├── roleGuard.ts    # Role-based access
│           │   ├── rateLimiter.ts  # Rate limiting
│           │   ├── validate.ts     # Zod validation middleware
│           │   └── ...
│           ├── routes/             # Route definitions
│           │   ├── auth.routes.ts
│           │   ├── post.routes.ts
│           │   ├── user.routes.ts
│           │   └── ...
│           ├── controllers/        # Request handlers (THIN)
│           │   ├── auth.controller.ts
│           │   ├── post.controller.ts
│           │   └── ...
│           ├── services/           # Business logic
│           │   ├── auth.service.ts
│           │   ├── post.service.ts
│           │   ├── feed.service.ts
│           │   ├── payment.service.ts
│           │   └── ...
│           ├── socket/             # Socket.IO handlers
│           │   ├── index.ts
│           │   ├── chat.handler.ts
│           │   └── ...
│           ├── jobs/               # BullMQ job processors (Phase 5+)
│           │   ├── email.job.ts
│           │   └── ...
│           └── utils/              # Server utilities
│               ├── logger.ts       # Structured logger (pino/winston)
│               ├── s3.ts           # S3 helper functions
│               └── ...
```

Section 24: Database Schema

The Prisma schema defines **28 models** and **24 enums**. Below is the complete schema overview organized by domain.

Models by Domain

Authentication & Users (Phase 0-1)

Model	Key Fields	Purpose
User	id, email, username, displayName, passwordHash, role, status, avatar, cover, bio, location, website, twoFactorSecret, twoFactorEnabled, onboardingStep, emailVerified, createdAt	Core user model
OtpCode	id, userId, code (hashed), type, expiresAt, used	Email verification & password reset OTPs
RefreshToken	id, userId, token (hashed), expiresAt, deviceInfo	JWT refresh token storage

Social Graph (Phase 2)

Model	Key Fields	Purpose
Follow	id, followerId, followingId, createdAt	Follow relationships
Block	id, blockerId, blockedId, createdAt	Block relationships

Content (Phase 2)

Model	Key Fields	Purpose
Post	id, authorId, text, visibility, isPinned, ppvPrice, likeCount, commentCount, createdAt	User posts
PostMedia	id, postId, url, type (IMAGE/VIDEO), order, thumbnail	Post media attachments
Comment	id, postId, authorId, parentId, text, createdAt	Post comments (threaded)
Like	id, postId, userId, createdAt	Post likes
Bookmark	id, postId, userId, createdAt	Post bookmarks

Payments (Phase 3)

Model	Key Fields	Purpose
SubscriptionTier	id, creatorId, name, price, description, benefits (JSON), order, isActive	Creator subscription tiers
Subscription	id, subscriberId, tierId, creatorId, status, startDate, endDate, renewalDate	Active subscriptions
Wallet	id, userId, balance, pendingBalance, totalEarned, totalSpent	User wallet
Transaction	id, walletId, type, amount, description, referenceId, status, createdAt	Wallet transactions
Payment	id, userId, amount, gateway, gatewayTransactionId, status, type, createdAt	External payment records
Tip	id, senderId, receiverId, amount, postId, messageId, createdAt	Tips on posts/messages
Withdrawal	id, creatorId, amount, paymentMethod, status, processedAt, rejectionReason	Creator withdrawal requests
PpvPurchase	id, userId, postId, amount, createdAt	PPV post purchases

Chat (Phase 4)

Model	Key Fields	Purpose
Conversation	id, participant1Id, participant2Id, lastMessageAt, lastMessage	Chat conversations
Message	id, conversationId, senderId, text, mediaUrl, mediaType, tipAmount, readAt, createdAt	Chat messages

Notifications (Phase 5)

Model	Key Fields	Purpose
Notification	id, userId, type, actorId, entityId, entityType, message, read, createdAt	User notifications
NotificationPreference	id, userId, type, inApp, email	Per-type notification settings

Stories (Phase 6)

Model	Key Fields	Purpose
Story	id, authorId, mediaUrl, mediaType, overlays (JSON), expiresAt, viewCount, createdAt	24-hour stories
StoryView	id, storyId, viewerId, viewedAt	Story view tracking

Live & Video (Phase 7)

Model	Key Fields	Purpose
LiveSession	id, creatorId, title, streamKey, rtmpUrl, hlsUrl, status, viewerCount, startedAt, endedAt	Live stream sessions
VideoCall	id, callerId, calleeId, vonageSessionId, status, duration, startedAt, endedAt	1-on-1 video calls

Marketplace & Gamification (Phase 8)

Model	Key Fields	Purpose
MarketplaceListing	id, sellerId, title, description, category, type (FIXED/AUCTION), price, startingBid, reservePrice, endsAt, status	Marketplace listings
Bid	id, listingId, bidderId, amount, createdAt	Auction bids
Badge	id, name, description, icon, rarity, criteria (JSON), category	Available badges
UserBadge	id, userId, badgeId, earnedAt	User-earned badges

Admin (Phase 9)

Model	Key Fields	Purpose
Report	id, reporterId, reportedUserId, reportedPostId, reason, description, status, resolvedBy, resolvedAt	Content/user reports
AuditLog	id, adminId, action, targetType, targetId, details (JSON), ipAddress, createdAt	Admin audit trail

Enums (24 total)

User & Auth

- **UserRole:** FAN, CREATOR, ADMIN
- **UserStatus:** ACTIVE, SUSPENDED, BANNED, DEACTIVATED
- **OtpType:** EMAIL_VERIFY, PASSWORD_RESET

Content

- **PostVisibility:** PUBLIC, SUBSCRIBERS, TIER_SPECIFIC
- **MediaType:** IMAGE, VIDEO

Payments

Chat & Notifications

- **SubscriptionStatus:** ACTIVE, EXPIRED, CANCELLED, PAST_DUE
- **TransactionType:** DEPOSIT, SUBSCRIPTION, TIP_SENT, TIP_RECEIVED, PPV_PURCHASE, PPV_EARNING, WITHDRAWAL, REFUND, BID_HOLD, BID_RELEASE, MARKETPLACE_PURCHASE, MARKETPLACE_EARNING
- **TransactionStatus:** PENDING, COMPLETED, FAILED, REFUNDED
- **PaymentGateway:** CCBILL, MIREXPAY
- **PaymentStatus:** PENDING, COMPLETED, FAILED, REFUNDED
- **WithdrawalStatus:** PENDING, PROCESSING, COMPLETED, REJECTED

- **MessageType:** TEXT, IMAGE, VIDEO, TIP
- **NotificationType:** LIKE, COMMENT, FOLLOW, SUBSCRIBE, TIP, MESSAGE, LIVE, STORY, MENTION, SYSTEM

Live & Marketplace

- **LiveStatus:** SCHEDULED, LIVE, ENDED
- **CallStatus:** RINGING, ACTIVE, ENDED, MISSED, REJECTED
- **ListingType:** FIXED_PRICE, AUCTION
- **ListingStatus:** ACTIVE, SOLD, EXPIRED, CANCELLED
- **ListingCategory:** DIGITAL_CONTENT, PHYSICAL_MERCH, EXPERIENCE,

Gamification & Admin

- **BadgeRarity:** COMMON, RARE, EPIC, LEGENDARY
- **BadgeCategory:** CONTENT, SOCIAL, ENGAGEMENT, REVENUE, SPECIAL
- **ReportReason:** SPAM, HARASSMENT, NUDITY, COPYRIGHT, OTHER
- **ReportStatus:** OPEN, INVESTIGATING, RESOLVED, DISMISSED

CUSTOM_CONTENT,
SHOUTOUT

Section 25: Development Phases & Timeline UPDATED

Phase 0: Foundation UPDATED (4 weeks)

- Monorepo setup (npm workspaces, TypeScript, ESLint, Prettier)
- ⚡ ESLint strict config: max-lines 200, no-any, complexity 10, no-console
- ⚡ Husky + lint-staged pre-commit hooks (lint + typecheck + test)
- `packages/shared`: Zod schemas, types, constants, utils
- `apps/server`: Express skeleton, Prisma schema, PostgreSQL config
- ⚡ **NO Redis in Phase 0** (deferred to Phase 4)
- `apps/web`: Vite + React + Tailwind v3.4 + shadcn/ui, Router, i18n, theme
- Layout components: Navbar, Sidebar, MobileNav, ThemeToggle, responsive
- ⚡ Sentry integration (error monitoring from day 1)
- ⚡ Vitest setup with example tests
- ⚡ Lighthouse CI setup in GitHub Actions
- GitHub Actions CI (lint, type-check, test, Lighthouse)
- Hetzner staging deployment (PM2 + Nginx)

> **Phase 0 Deliverable**
>
> Running skeleton with CI/CD, design system, responsive layout, quality gates enforced from day 1.

Phase 1: Auth & Users (4 weeks)

- DB: User, OtpCode, RefreshToken migrations
- Backend: Register, Login, Logout, Refresh, OTP, Reset, 2FA (all endpoints)
- Backend: Profile CRUD, S3 uploads, role middleware
- Frontend: All auth pages (login, register, verify, reset, 2FA)
- Frontend: Onboarding wizard (4 steps), Settings pages (all sections)

- Frontend: useAuth, authStore, auto token refresh, protected routes
- Mobile: touch-friendly forms, keyboard handling, fullscreen modals
- ⚡ Unit tests for auth services, integration tests for auth API

> **Phase 1 Deliverable**
>
> Full auth system with 2FA, user profiles, onboarding wizard, all settings pages.

Phase 2: Social & Feed (4 weeks)

- DB: Follow, Block, Post, PostMedia, Comment, Like migrations
- Backend: Follow/unfollow, block, post CRUD, feed, comments, likes, bookmarks
- Backend: S3 presigned uploads, visibility enforcement, explore algorithm
- Frontend: PostComposer, PostCard, FeedList, InfiniteScroll, MediaGallery
- Frontend: Creator profile page, Explore page, Single post view
- Frontend: Follow button, BlurredOverlay, CommentSection
- Mobile: pull-to-refresh, full-bleed media, bottom sheet composer
- ⚡ Unit tests for feed/post services, component tests

> **Phase 2 Deliverable**
>
> Users can follow, post, like, comment, explore content.

⚡ MVP CHECKPOINT (After Phase 2)

Phase 0 + 1 + 2 = Working Product

- ✓ Signup, login, 2FA, profile, onboarding
- ✓ Create posts, follow creators, home feed, explore
- ✓ STOP and evaluate before Phase 3
- ✓ Deploy MVP to staging environment
- ✓ Get real user feedback
- ✓ Review code quality metrics (coverage, Lighthouse scores, bundle size)

✓ Acceptance criteria must be met before proceeding

✓ **Decision point: proceed to Phase 3 or iterate on MVP**

Phase 3: Subscriptions & Payments UPDATED (6 weeks)

- DB: SubscriptionTier, Subscription, Wallet, Transaction, Payment, Tip, Withdrawal migrations
- Backend: Tiers CRUD, subscribe/cancel
- ⚡ **CCBill integration ONLY** (sandbox → production)
- ⚡ MirexPay deferred to Phase 10
- Backend: Wallet, tips, PPV, withdrawals
- ⚡ NO BullMQ yet — use direct processing for payments
- Frontend: Tier management, Subscribe modal, Wallet, Earnings dashboard
- Frontend: Tip modal, PPV unlock, Withdrawal form, Revenue charts
- Mobile: bottom sheet modals, responsive charts
- ⚡ Payment webhook tests with mocks, wallet transaction tests

Phase 3 Deliverable

Full payment flow with CCBill, tipping, wallet, withdrawals.

Phase 4: Chat + Redis UPDATED (4 weeks)

- DB: Conversation, Message migrations
- ⚡ **Redis added here:** Socket.IO adapter + online status + session store
- Backend: Socket.IO with JWT, chat service, typing, read receipts, online status
- Frontend: Chat page (split desktop, fullscreen mobile), MessageBubble, emoji
- Frontend: Typing indicator, online dots, media in chat, tip in chat
- Mobile: fullscreen chat, keyboard handling, camera/gallery attachment
- ⚡ Socket.IO integration tests

Phase 4 Deliverable

Real-time messaging with all features + Redis infrastructure.

Phase 5: Notifications + BullMQ UPDATED (2 weeks)

- DB: Notification, NotificationPreference migrations
- ⚡ **BullMQ added here:** email queue + cron jobs
- Backend: NotificationService, Socket.IO push, BullMQ + SES email, preferences
- Frontend: Bell icon + badge, dropdown, full page, filters, toasts, preferences
- Mobile: swipe to dismiss, pull-to-refresh
- ⚡ E2E tests for critical flows (Playwright setup)

> **Phase 5 Deliverable**
>
> In-app + email notifications + background job infrastructure.

Phase 6: Stories (2 weeks)

- DB: Story, StoryView migrations
- Backend: Story CRUD, S3 upload, 24h expiry cron, view tracking
- Frontend: StoryRing, StoryViewer (fullscreen), StoryCreator
- Mobile: horizontal scroll rings, fullscreen gestures, camera capture

> **Phase 6 Deliverable**
>
> Instagram-style 24-hour disappearing stories.

Phase 7: Live Streaming & Video Calls (4 weeks)

- DB: LiveSession, VideoCall migrations
- Backend: RTMP/HLS, stream keys, live chat, viewer count, Vonage integration
- Backend: Call signaling via Socket.IO, duration tracking
- Frontend: Go Live page, Live viewer + chat, Video call UI
- Mobile: portrait/landscape live view, fullscreen call

> **Phase 7 Deliverable**
>
> Live streaming via OBS + 1-on-1 video calls.

Phase 8: Marketplace & Gamification (4 weeks)

- DB: MarketplaceListing, Bid, Badge, UserBadge migrations
- Backend: Listing CRUD, bidding, auction close, Buy Now, badge engine, leaderboard
- Frontend: Marketplace browse/detail, bid form, create listing wizard
- Frontend: Badge display, Leaderboard page
- Mobile: 2-column grid, filter drawer, sticky bid form

> **Phase 8 Deliverable**
>
> Full marketplace with auctions + gamification system.

Phase 9: Admin Panel (4 weeks)

- Backend: All admin API endpoints with audit logging
- Frontend: Admin layout, Dashboard, User management, Content moderation
- Frontend: Reports, Payments, Withdrawals, Settings, Badges, Analytics
- Frontend: Announcements, Audit log
- Mobile: admin responsive (cards instead of tables)

> **Phase 9 Deliverable**
>
> Full admin panel for platform management.

Phase 10: Polish & Launch UPDATED (4 weeks)

- ⚡ **MirexPay integration** as second payment gateway
- Complete i18n, Creator analytics dashboard
- Performance: lazy loading, code splitting, CDN, image optimization
- Security audit: XSS, CSRF, CSP, rate limiting, input sanitization
- AWS production deployment (ECS + RDS + ElastiCache + S3 + CloudFront)
- ⚡ Sentry already running since Phase 0 — review and tune alerts
- Load testing, cross-browser QA
- Mobile: final responsive QA across devices

> **Phase 10 Deliverable**
>
> Production-ready platform with dual payment gateways.

Timeline Summary

Phase	Name	Weeks	Cumulative
0	Foundation	4	4 wk
1	Auth & Users	4	8 wk
2	Social & Feed	4	12 wk
-	⚡ **MVP CHECKPOINT**	-	**STOP & EVALUATE**
3	Subscriptions & Payments	6	18 wk
4	Chat + Redis	4	22 wk
5	Notifications + BullMQ	2	24 wk
6	Stories	2	26 wk
7	Live & Video	4	30 wk
8	Marketplace & Gamification	4	34 wk
9	Admin Panel	4	38 wk
10	Polish & Launch + MirexPay	4	42 wk

> **Total Timeline**
>
> **~42 weeks (~10 months) sequential.** With 4–5 developers and phases 4–8 parallelized: **~7–8 months.**

Section 26: Risk Resolution NEW

⚡ NEW SECTION — ADDED IN V2.1

6 identified risks with concrete mitigations built into this plan:

1. Scope Creep

Impact: HIGH Probability: HIGH

Mitigation: Phase gates with acceptance criteria. MVP checkpoint after Phase 2. Feature flags for experimental features. No new features added mid-phase. Each phase has clear deliverables that must be met before proceeding to the next.

2. Slow Site / Poor Performance

Impact: HIGH Probability: MEDIUM

Mitigation: Lighthouse CI with hard limits (90/95/95). Bundle size budget (200KB gzipped). Every route lazy loaded. Virtual scrolling for lists. Image optimization pipeline. API response time monitoring via Sentry. Performance budget violations block deployment.

3. Code Becoming a Mess (Again)

Impact: CRITICAL Probability: HIGH

Mitigation: ESLint strict config from day 1. Max 200 lines per file. No any type. Thin Controller → Service → Repository mandatory. Pre-commit hooks block bad code. Cyclomatic complexity limit of 10. No console.log. Code review required for all PRs. No exceptions.

4. Security Vulnerabilities

Impact: CRITICAL Probability: MEDIUM

Mitigation: Helmet.js + rate limiting from Phase 0. OWASP Top 10 checklist reviewed per phase. Input validation via shared Zod schemas. SQL injection prevented by Prisma ORM. XSS prevented by React's default escaping. CSRF tokens on state-changing requests. Content Security Policy headers.

5. Payment Integration Failures

Impact: HIGH Probability: MEDIUM

Mitigation: CCBill sandbox testing first (Phase 3), production later. Webhook simulator for local testing. Mock payment service for unit tests. MirexPay deferred to Phase 10 (one gateway at a time). Idempotent webhook handlers. Transaction logging for debugging.

6. Scaling Issues

Impact: MEDIUM Probability: LOW (at launch)

Mitigation: Stateless server design from day 1. Redis adapter for Socket.IO when added (Phase 4). Database connection pooling via Prisma. Horizontal scaling ready (ECS auto-scaling). CDN for static assets. Database indexes defined with schema. Query performance monitoring via Sentry.

Section 27: Deployment Strategy UPDATED

Staging Environment (Hetzner)

```
Nginx (reverse proxy + SSL termination)
├── /              -> React SPA (Vite build static files)
├── /api           -> Express server (PM2, 2 instances)
└── /socket.io     -> Socket.IO (sticky sessions via Nginx)

PostgreSQL 15 (on same server or managed)
⚡ Redis 7 (added in Phase 4, not initial setup)
⚡ BullMQ worker process (added in Phase 5)
```

Production Environment (AWS)

```
CloudFront CDN
├── S3 bucket (React SPA static files)
└── S3 bucket (user media uploads)

ALB (Application Load Balancer, sticky sessions for Socket.IO)
├── ECS Fargate / EC2 instances
│   ├── Express API container (auto-scaling group)
│   └── Socket.IO container (auto-scaling group)
│
├── BullMQ Worker container (Phase 5+)

RDS PostgreSQL (Multi-AZ, automated backups, read replicas)
ElastiCache Redis (cluster mode, 2 nodes) - added Phase 4+

SES (transactional email, verified domain)
Route 53 (DNS management)
ACM (SSL/TLS certificates)
CloudWatch (monitoring, alerts, dashboards)
⚡ Sentry (error tracking from Phase 0 - external service)

S3 (media uploads)
├── Lifecycle policies (move to Glacier after 1 year)
└── CloudFront distribution for fast media delivery
```

CI/CD Pipeline (GitHub Actions)

```
on push to main:
  1. Install dependencies (npm ci)
  2. Lint (ESLint — strict rules, zero tolerance)
  3. Type-check (tsc --noEmit — all packages)
  4. Run tests (Vitest — coverage thresholds)
  5. ⚡ Lighthouse CI check (90/95/95 minimums)
  6. ⚡ Bundle size check (200KB gzipped max)
  7. Build (vite build + tsc)
  8. Deploy to Staging (automatic)
  9. Smoke tests on staging
  10. Manual approval gate
  11. Deploy to Production
  12. Post-deploy health check
```

Deployment Notes

- **Zero-downtime deploys:** Rolling update strategy in ECS. Blue-green deployment option for major releases.
- **Database migrations:** Run before deploy via CI/CD. Backward-compatible migrations only (expand-contract pattern).
- **Rollback:** Automated rollback if health check fails. Previous container image retained. Database rollback via Prisma migrate.
- **Secrets management:** AWS Secrets Manager for production. Environment variables for staging. Never in code.

Section 28: Phase 0 Verification Checklist

UPDATED

After Phase 0 completion, verify **ALL** of the following before moving to Phase 1:

#	Check	Expected Result
1	**Dev Server**	`npm run dev` starts Vite (port 5173) AND Express API (port 4000) concurrently
2	**Frontend Loads**	React app loads in browser with Tailwind CSS v3.4 styling applied correctly
3	**Theme Toggle**	Dark/Light/System theme toggle works, persists to localStorage across refresh
4	**i18n**	Language switching works, English translations load, fallback works
5	**Navbar**	Top navbar renders correctly on all breakpoints (mobile/tablet/desktop)
6	**Sidebar**	Left sidebar renders on desktop (1024px+) with nav links
7	**Mobile Nav**	Bottom navigation bar shows on mobile (<768px) with 5 icons
8	**Responsive**	Layout adapts correctly at all breakpoints (sm, md, lg, xl, 2xl)
9	**API Health**	`GET http://localhost:4000/api/health` returns `{ status: "ok" }`
10	**Database**	Prisma connects to PostgreSQL, migrations run successfully
11	**DB Seed**	Seed creates admin user + sample data
12	⚡ **No Redis Check**	Redis NOT required in Phase 0 (deferred to Phase 4)
13	**Shared Package**	`@fansbook/shared` imports work in both apps/web and apps/server

#	Check	Expected Result
14	**CI Pipeline**	GitHub Actions runs on push: lint + typecheck + test + Lighthouse
15	**Staging**	Deployed to Hetzner, accessible via URL
16	**Components**	shadcn/ui components render correctly with Tailwind v3.4
17	**Router**	Routes work: /, /login, /register, /explore, /settings, /admin, 404
18	⚡ **ESLint Strict**	All anti-mess rules active (max-lines, no-any, complexity, no-console)
19	⚡ **Husky Hooks**	Pre-commit runs lint + typecheck + test, blocks on failure
20	⚡ **Sentry**	Error tracking active, test error appears in Sentry dashboard
21	⚡ **Vitest**	Test suite runs, example tests pass
22	⚡ **Lighthouse CI**	Scores meet minimum thresholds (90/95/95)
23	⚡ **Bundle Size**	Initial JS bundle under 200KB gzipped

Phase 0 Gate

ALL 23 checks must pass before starting Phase 1. This is not optional. Phase 0 establishes the foundation that every subsequent phase depends on. A weak foundation = a messy codebase by Phase 3.

End of Document

Fansbook Platform — Complete Rebuild Plan

Version 2.1 | February 2026

28 Sections | 28 Database Models | 24 Enums | 11 Development Phases | ~42 Weeks

4 New Sections: Anti-Mess Rules, Performance Budget, Testing Strategy, Risk Resolution