

Gemini Chat

Can we discuss a good release strategy. My current plans are to deploy the app to cloudflare when completed. I want to have a robust system that won't break the bank but can scale. I also want to make sure that the app will be able to support itself on release and be able to pay its own bills so I don't want to create any expensive features early on that are unsustainable



Service	Free Tier Limits	Cost Mitigation Strategy
Hosting (SvelteKit)	Cloudflare Pages: Unlimited sites, unlimited requests, generous build time.	SvelteKit Adapter: Ensure you are using the official Cloudflare Adapter. All front-end assets and static pages are served globally for free.
Backend/DB (Supabase)	Dedicated Postgres DB, 500MB Disk, 50,000 MAUs, 2GB File Storage, 100K Edge Function invocations/month.	Database Optimization: Use Row Level Security (RLS) exclusively to enforce business logic, minimizing the use of external Edge Functions for simple checks. Use TS to keep database interactions lightweight.

Service	Free Tier Limits	Cost Mitigation Strategy
Storage (Photos/Media)	Supabase Storage: 2GB included free. Cloudflare R2: 10GB storage, 1M Class A ops (write) / 10M Class B ops (read) free.	Offload Media: Move <i>all</i> photo/video storage to Cloudflare R2 as soon as possible, as its read operations are extremely generous. Free Tier storage (2GB) should be reserved only for essential database backups/critical files.
Serverless Logic (Edge Functions)	Supabase: 100K invocations/month. Cloudflare Workers: 100K requests/day (or 3M/month) free.	Reputation & AI: Route the high-volume/high-cost logic (like <code>calculateReputationTier</code> and AI API calls) through Cloudflare Workers instead of Supabase Edge Functions, using the Workers' much larger free quota for 90% of your dynamic calls.
AI API Costs	OpenAI/Claude: Pay-as-you-go, high variable cost.	Strictly Gated: AI features are Premium-only and usage is strictly governed by the AI Credit System . Monitor usage daily. If AI costs threaten revenue, immediately raise the price of credit packs or reduce the monthly credit allotment.


Phase (Release)	Features (Must-Haves)	Core Goal	Monetization Hook
Beta Launch (P1 Core)	Auth, Personal Team, Project Management (Max 3 Active Projects), Basic Photo Upload, Simple Messaging.	Test PMF & Core Usability. Get 100 dedicated users using the app for their <i>solo</i> planning.	None / Free Forever. Test retention and find core pain points.
Soft Launch (P2 Accountability)	Temp Teams/Event Creation, Attendance Tracking, Reputation System (Visible Tiers) , Post-Event Ratings.	Test Collaboration & Trust. Establish community governance and accountability. Drive event creation/participation.	FREE LIMITS: Enforce 5GB storage cap and Max 1 Collaborative Team join limit.

Phase (Release)	Features (Must-Haves)	Core Goal	Monetization Hook
V1.0 Launch (P3 Monetization)	Commission Marketplace (Beta) , Contract Builder (Draft/Template only), Stripe Connect Integration, Premium Subscription Launch , Unlimited Projects/Storage/Team Create/Join.	Achieve Self-Sustainability. Convert early power users (those needing more projects/storage) to Premium. Generate initial GMV via commissions.	PAID FEATURES LIVE: Paid \$/\$month and Team Premium subscriptions launch. AI Credit Add-Ons launch.
V1.5+ (Growth)	AI Features (Gated), Social Scheduling, Convention Planner, Advanced Integrations (Notion/Pinterest).	Drive Growth & Retention. Use unique features to differentiate and lock in power users.	Monetization scales automatically with usage: more projects → more storage/AI → higher revenue.



Stage	Activity	Goal	Target Audience
Pre-Beta	Landing Page: Create a single page with a strong value prop and an Email Waitlist . Offer the first 50 sign-ups a Lifetime Subscriber price lock.	Build initial hype and secure early testers.	Niche Subreddits (r/cosplay, r/cosplayers), Discord communities.
Beta (P1)	"The Problem Solver" Content: Share tutorials on social media (TikTok/Instagram Reels) that focus on the <i>pain points</i> (e.g., "Stop forgetting supplies!"). Offer the app as the solution.	Drive organic traffic and initial signups to test core features.	Micro-Influencers & Cosplay Streamers (offer early Lifetime access).

Stage	Activity	Goal	Target Audience
Soft Launch (P2)	The "Reliability" Push: Focus marketing on the Reputation System . Content: "Find reliable collaborators with our Reputation Tiers." This directly addresses a major community pain point (no-shows).	Encourage Collaboration. The feature is the marketing.	Event Organizers, Cosplay Photographers (who value punctuality).
V1.0 Launch (P3)	Commission GMV Focus: Promote the Marketplace as the Safe, Contract-Backed way to buy/sell commissions. Content: "Never get scammed again. Milestone payments protect you."	Drive Revenue. The safety/trust factor is the main sales pitch.	Commission Clients (the buyers) and Professional Commissioners (the sellers).



package dashboard usage (specifically Disk Size and Egress) against the free tier limits. Be prepared to upgrade to the Pro Tier (\$25/month) as building, updating, and releasing. Also would it be worth looking into getting investment for this project or would a grassroots approach be better?

Don't Marketing Until Profitable: Do not spend a single dollar on marketing until you have a revenue stream that is $\geq \$100$ per month.





Build Strategy: Spec-Driven & Cost-Optimized

Oct 21, 12:38 PM



Release Strategy: Build-Measure-Monetize (B-M-M)

Oct 21, 12:38 PM



Update Strategy: Iteration, Trust, and Lifecycle Management

Oct 21, 12:38 PM

So what do you think between getting investors vs grassroots?



Factor	Grassroots/Bootstrapped	Venture Capital/Investment
Core IP Validation	Safer. Your core feature is the Reputation System . This takes months of real user data and community consensus to validate.	Unrealistic Pressure. Investors demand rapid, exponential scale. You would be forced to focus on

Factor	<p>Grassroots/Bootstrapped</p> <p>Bootstrapping lets you prove the system works <i>before</i> being judged on it.</p>	<p>Venture Capital/Investment</p> <p>marketing and user acquisition before your core, high-dependency system is proven, leading to potential failure if the trust model breaks under stress.</p>
Financial Risk	<p>Extremely Low. Your SvelteKit/Supabase/Cloudflare stack is essentially free up to a significant scale. Your only risk is time and AI API overspend (which we have hard-gated).</p>	<p>High Burn Rate. Investment requires hiring (salaries), massive marketing spend (paid ads), and a large jump in operational costs, forcing a revenue goal that may be months away from being realistically achievable.</p>
Equity & Control	<p>100% Retention. You maintain full control over the product roadmap, ensuring every decision adheres to the community-first principles in your Constitution.</p>	<p>Significant Dilution. You would give away a large portion of your company (15%-30% for a seed round) based on a <i>plan</i>, not proven <i>revenue</i>. Investors could also push you to abandon the niche focus for broader appeal.</p>
Niche Focus	<p>Perfect Fit. The cosplay community thrives on organic trust and word-of-mouth. This aligns perfectly with a grassroots, community-first launch strategy.</p>	<p>Poor Fit (Early). Most VCs fear niche markets and push for pivots toward broader "project management" too early, risking the loss of your unique identity.</p>




Since this is a small scale project currently, I want to make sure that I have a clear idea for monetization models. I want this to be affordable and accessible to many people focusing on those who are new to cosplay or are wanting to grow in the cosplay community since more established creators might already have tools they use. I also want a diverse income source in case there is low premium user conversion



Tier	Price	Target User	Key Value Proposition	Gated Limits
Free Forever	\$0	Beginners / Casual Cosplayers	Access to the entire Reputation System , Basic Project Management, and Temp Team participation.	Max 3 Active Projects, Max 1 Collaborative Team (Joined), 5GB Storage, 20 AI Credits/Month .
Growth Premium	\$5 / month (or \$50 / year)	Growing Cosplayers / Dedicated Hobbyists (Your Target)	Removes core limitations to allow for serious progress and portfolio building.	Unlimited Projects, Create 3 Collaborative Teams, 50GB Storage, 500 AI Credits/Month , All Phase 4 Integrations (Notion, Pinterest, etc.).
Pro Premium	\$12 / month (or \$120 / year)	Professional Commissioners / Photographers / Collective Organizers	Provides the resources needed to run a small business.	Unlimited Everything, Video Storage (25GB), Advanced Analytics, Priority Support.
Team Premium	\$35 / month	Group Cosplays / Large Collectives (Up to 15 Members)	Shared resources that make group planning simple.	Shared 200GB Storage, 2000 AI Credits/Month (shared pool).



Fee Model	Details	Rationale
Marketplace Fee	A flat 5% fee on the total transaction value (GMV) of a completed commission, deducted from the Commissioner's payout before Stripe fees.	Standard marketplace model. It is 5% lower than platforms like Etsy, promoting adoption due to competitive pricing, while funding the secure payment infrastructure (Stripe, contracts).
Dispute Fee	A flat fee (e.g., \$10) charged to the party deemed at fault in a formally resolved commission dispute.	Encourages both parties to resolve issues amicably and only use the formal system when necessary.



Product	Price Point	Rationale	Cost Mitigation
AI Credit Packs	\$5 for 100 Credits	Directly offsets your highest variable cost (OpenAI/Claude APIs). Allows Free users to try out AI without subscribing.	Every pack sold directly funds 100+ high-cost API calls. Ensures AI is cost-neutral.
Export/Archival Fee	Free for teams < 20 GB, \$10 for teams > 20 GB (after 30 days of archiving).	Charges users for consuming massive cold storage resources, encouraging them to manage large libraries.	Ensures users consuming the most storage directly pay for the incremental cost of that storage.
Digital Templates	\$2 - \$5 per template	Sell high-quality, pre-made digital assets created by the community or by you (e.g., budget trackers, measurement charts, complex task breakdowns).	A zero-cost revenue source once the templates are created, leveraging the skills of the community and the platform's

Product	Price Point	Rationale	Cost Mitigation
			focus on organization.
			

expensive features (AI) are **always cost-neutral or profitable**.
I am also a photographer and I use other systems for booking and
eventing. API spend from bankrupting the project before it achieves
delivering images to customers but I think it would be useful to include
ale those options in this app so I can reduce my own personal costs but
also provide others a platform to help build their own business on top
strategy is highly defensible and sustainable for a bootstrapped project.
of. Not just for photographers but also for wig stylists, makeup artist,
etc.



Feature	Phase	Rationale	Cost Mitigation
Provider Profile & Rate Cards	P3 (Monetization)	<i>Must be ready for the V1.0 launch.</i> Uses existing profile tables.	Low cost. Uses existing DB structure (<code>user_profiles</code> extension).
Service Booking & Deposit	P3 (Monetization)	Integrates directly with the core Stripe Connect and Commission features, enhancing the GMV.	Low cost. Uses existing Stripe/Contract infrastructure.
Private Client Gallery	P6 (Advanced)	This is a Pro Premium/Team Premium feature only. Highly technical and	Gating: Revenue generated by Pro Premium subscriptions must offset R2 bandwidth

Feature	Phase	Rationale	Cost Mitigation
		high storage/bandwidth use.	and storage costs. This should be treated as a high-value, later-stage feature.



I would want to compete a bit with pixieset but they have a competitive free tier that might be hard to beat. But I do think having a basic free tier option would be good to help smaller creators get started by having a full suite of tools they can use with upgrade options that make sense



Feature	Cosplay App Free Tier	Competitive Edge over Pixieset Free
Storage	5GB (Shared with Project Storage)	Less than Pixieset (3GB), but integrated with all planning tools.
Galleries/Jobs	Unlimited Private Client Galleries.	Match Pixieset (Unlimited).
Client Proofing	Full Functionality: Client can select favorites, leave comments, and download <i>low-resolution, watermarked proofs</i> .	Match Pixieset functionality.
Final Delivery	Manual Delivery Required. Provider must deliver final, high-res files <i>outside</i> the system (e.g., Google Drive link in the chat).	Key Friction Point. Forces upgrade for convenience and speed.
E-commerce/Sales	0% Commission Fee on all sales/transactions processed via the Marketplace (Stripe Connect).	MASSIVE ADVANTAGE: Pixieset charges 15% commission on sales for their Free Tier. You eliminate this high cost for your users.
Branding	Cosplay App Branding and subtle watermark remain visible on public galleries.	Match Pixieset (removes branding on paid plans).
Business Tools	Full Access to Contract Builder, Invoicing, and Milestone Tracking (within Free Tier Project Limits).	Superior. Pixieset gates some studio management tools. You offer the full business foundation for free.



Upgrade Level	Trigger Point	Unlocked Provider Feature	Rationale
Growth Premium (\$5/mo)	Hitting 5GB Storage Limit OR wanting	Automated Final Delivery (High-res download link generation inside the	Targets the busy hobbyist/part-time professional who values time savings and needs

Upgrade Level	Trigger Point	Unlocked Provider Feature	Rationale
	Automated Delivery.	gallery). Video Proofing (Basic H.264 compression).	more storage. \$5 is a low hurdle to remove a massive convenience block.
Pro Premium (\$12/mo)	Needing Professional Branding OR hitting the 50GB Storage Limit.	Custom Domain & Branding (Remove Cosplay App logo). 500GB Storage dedicated to client galleries. RAW/Large File Support (for archiving).	Targets the full-time professional for whom branding and capacity are non-negotiable business expenses.

Advantage	Benefit to Growing Creator (Target User)
0% Commission on Sales (Free Tier)	Immediate Profitability: A 15% difference on a \$500 commission is \$75. This is a powerful financial incentive to choose your platform over Pixieset or others.
Reputation System Built-In	Business Security: The client gallery is tied to the Reputation System , ensuring the provider sees a clear trust score for the client <i>before</i> accepting the job. Pixieset doesn't offer community-wide accountability.
All-in-One Platform	Efficiency: They manage projects, contracts, teams, and client delivery all in one place, reducing the need for separate subscriptions (Trello, Square, Pixieset).

I also use capture one for photo editing as well as I have used Lightroom. How difficult would it be to create plugins for these applications to make integration with my website/app so its easier to see which images a user wants edited and lower delivery of final products




Application	SDK/API Availability	Integration Difficulty (Low-Cost Focus)
Lightroom Classic	High. Uses Lua scripting language and a comprehensive SDK. Primarily relies on syncing metadata (ratings, selections) via the cloud ecosystem.	Medium. Possible via the existing Adobe Sync ecosystem (Less expensive to implement, as you don't build a <i>full</i> plugin, but rather a bridge).
Capture One	Medium/Advanced. Offers a Developer Portal and SDK. macOS supports powerful AppleScript/JXA automation. Windows relies more on keyboard shortcuts.	High. Requires specialized development effort. The best initial approach is to leverage metadata read/write via XMP files, not a full two-way plugin.

Rating: 5 Stars

Label: Green

Feature	Original Phase	Revised Phase	Notes
Provider Profile & Rate Cards	P3	P3	Essential for service provider identity.
Service Booking & Deposit	P3	P3	Core monetization mechanism.
Client Proofing & Selection UI	P6 (Moved from Private Gallery)	P3 (CRITICAL)	Must be deployed with P3, as it generates the data for the next step.
XMP Metadata Export/Import	New Feature	P4 (Automation)	The low-cost sync solution. Deploy this immediately after the Client Proofing UI (P3) is stable.
Private Client Gallery	P6	P6	(Still P6) High storage/bandwidth feature, must be gated by Pro Premium .
Dedicated LR/C1 Plugins	New Feature	P7+ (Future Growth)	High development effort. Reserve for post-V1.0 when the app is generating consistent revenue to fund specialized development.




izing the lean financial model.

Looking at this project, what do you think the viability is for a single developer to create something like this?



Factor	Why It Works for a Solo Developer
Tech Stack	SvelteKit + Supabase + Cloudflare is arguably the most solo-friendly, high-performance stack available. It handles Auth, Database, RLS (security), and deployment with minimal configuration. This cuts 70% of the typical backend and DevOps burden.
Spec-Driven Dev	Using GitHub Spec Kit and AI agents forces you to define logic <i>before</i> coding. This prevents feature creep, minimizes rework, and keeps the code quality high, reducing long-term maintenance debt.
Niche Focus	You are solving a problem for a specific community you know (cosplay/photography). This means your initial user acquisition is focused (niche forums, Discord), making marketing manageable and feedback loops tight.
Gated Complexity	The roadmap strategically pushes the most difficult features (Dedicated LR/C1 Plugins, Advanced AI, Video Processing) to Phase 6/7 , when the app should be generating revenue to either fund that development or outsource it.
Revenue Diversity	The three pillars of monetization (Subscription, Commission Fee, A La Carte Credits) provide financial shock absorbers, making the business more resilient than a single-stream subscription model.



Phase (Build Time Estimate)	Estimated Solo Time Investment	Key Solo Burden
P1: Core Foundation (MVP)	3 - 4 Months	Getting Auth/RLS hardened, setting up core data models, design foundation.
P2: Accountability & Trust	2 - 3 Months	Building the complex Reputation Edge Function and rigorous testing of all RLS policies (must be bulletproof).
P3: Monetization (V1.0)	3 - 4 Months	Integrating Stripe Connect (webhooks, subscriptions, customer portal) is often a huge time sink due to edge cases and legal testing.
P4/P5: Automation & Growth	4 - 6 Months	Integrating complex third-party APIs (Instagram, TikTok, Notion), which break frequently and require constant maintenance.
Total to V1.0 (Revenue Ready)	8 - 11 Months	Total to feature completeness (V1.5)

Challenge / Role	Solo Burden & Strategy
Perfectionism & Burnout (Developer)	Strategy: Adhere to the MVP principle. Do not spend three weeks on a feature an AI agent could complete in two days. Set strict boundaries (no coding after 8 PM) to preserve energy.
Context Switching (All Roles)	Strategy: Time Blocking. Dedicate specific blocks to specific tasks: e.g., Tuesday mornings are Marketing & Support (responding to feedback), Thursday afternoons are Stripe Testing (financial safety).
Legal & Finance (Business Owner)	Strategy: Focus on the automated solutions: Stripe Connect handles most financial compliance (tax, invoicing). Use the Contract Builder feature for automated legal templates. Hire a lawyer for <i>one hour</i> to review your Terms of Service template, rather than building it yourself.
Quality Assurance (Tester)	Strategy: Rely on Specs. Use the <code>qa.md</code> files generated by the AI agent to force unit testing, especially on the Reputation and Payment logic. Use early beta users to find integration bugs.



[How a solo developer managed to build a commercial SaaS product](#)



Launch Your SaaS in Under 7 Days with Next JS, Supabase & Payments - YouTube ⓘ
JavaScript Mastery · 743K views



Perfect, could we update those earlier documents with what we discussed?



Cosplay Planning App - Project Constitution
Oct 21, 12:38 PM

Here is my current constitution file. I just want you to help look it over and also look at our current plans and see if there is anything missing, anything that should be changed. Any conflicts. Or anything else you noticed.

Cosplans Constitution

Core Principles

I. Web-First with Mobile-Responsive Architecture

All features MUST be designed with mobile responsiveness as a core requirement. The system will be implemented in three phases: (1) responsive web application (SvelteKit), (2) Android native app (Flutter), (3) iOS native app (Flutter). The web application is the primary initial platform using SvelteKit for optimal performance and SEO; native mobile apps built with Flutter provide enhanced mobile experiences with shared codebase between Android/iOS.

Platform Deployment Strategy:

Phase 1 (Web): SvelteKit responsive web application serving all users across devices. Focus on mobile-responsive design, touch-friendly interfaces, progressive web app (PWA) capabilities, and server-side rendering for SEO. All core features implemented here first.

Phase 2 (Android): Flutter native app for Android providing enhanced mobile performance, offline capabilities, and native integrations (camera, notifications, location services). Features ported from validated web implementation.

Phase 3 (iOS): Flutter native app for iOS with same capabilities as Android. Shared Flutter codebase enables rapid deployment after Android validation.

All platforms MUST share a common backend API to ensure data consistency. UI components and design tokens SHOULD be shared where technically feasible to maintain visual consistency across platforms. Feature development follows the phase order: web → Android → iOS.

Rationale: Web-first deployment with SvelteKit enables rapid development, excellent performance (<3 sec loads on 3G), and immediate user feedback without app store delays. Flutter for mobile apps provides native performance with 100% code reuse between Android and iOS, accelerating Phases 2-3 development. Cosplayers need mobile access for on-location work, but web-responsive design serves this initially while building toward native app performance.

II. Real-Time Collaboration

Team members MUST be able to view and edit shared data in real-time. Changes to shoots, costumes, props, and schedules MUST be synchronized across all team members immediately. Offline capability with sync-on-reconnect is required for location scouting and remote shoots.

Rationale: Cosplay teams coordinate complex logistics across multiple people, locations, and timelines requiring instant communication and updates.

II.5. User Team Ownership Requirement

Every user MUST be part of at least one team they own at all times. During account creation and onboarding, the system MUST automatically create a default team with the new user as owner. Users MAY create additional teams and MAY join existing teams as members, but they MUST always maintain ownership of at least one team.

Team Deletion Rules: A team can only be deleted when there is only one member remaining AND that member owns at least one other team. The system MUST prevent deletion of a user's last owned team. When a user attempts to delete their only owned

team, the system MUST display an error message: "Cannot delete your only team. Create or join another team first."

Solo User Support: Users working independently effectively operate as a "team of one" with full owner permissions. This ensures a consistent permission model and data architecture across all use cases—whether users collaborate in large teams or work solo on personal projects. All core features (shoots, costumes, props, schedules) exist within a team context.

Rationale: The application is fundamentally team-centric in its architecture and data model. All features are designed around team workflows and team-scoped data. Requiring every user to own at least one team ensures consistent permission checking, data organization, and feature access. Solo cosplayers benefit from the same powerful features as collaborative teams without added complexity. The team ownership requirement prevents edge cases where users have no context to create or view content, and ensures proper cleanup when teams disband (remaining member must join/create another team before deletion).

III. External Integration Integrity

Google Maps, Google Calendar, Google Docs, Instagram, and email reminder integrations MUST maintain data consistency. Calendar events MUST sync bidirectionally with Google Calendar. Document summaries MUST preserve user edits while updating automated content. Email reminders MUST be reliable and respect user notification preferences. Instagram posts MUST sync read/write for scheduling and analytics. API failures MUST not corrupt local data.

Rationale: Teams rely on existing Google ecosystem tools, communication channels, and social media platforms. Email reminders ensure critical deadlines and shoot preparations are not missed. Instagram is the primary platform for cosplay community engagement and portfolio building; integrating social planning into shoot workflow prevents context switching and enables data-driven content strategy. Disrupting these workflows would reduce adoption and productivity.

IV. Customizable Workflow States

Teams MUST be able to customize shoot stages (idea → planning → scheduled → editing → posting → complete) and costume/prop tracking stages. Default workflows are provided but teams can modify stage names, add/remove stages, and define stage transition rules.

Costume/Prop Lifecycle Stages: The system MUST support comprehensive lifecycle tracking for costumes and props beyond simple acquisition/in-progress/ready states. Required lifecycle stages include:

Active ownership states: planned, acquiring, in-progress, ready, owned

Post-use states: sold (with sale date/price), damaged (with repair notes/cost), rented (with rental period/cost), lost (with incident details)

Storage states: stored (with location), loaned (with borrower/return date)

Teams MUST be able to add custom intermediate stages (e.g., "needs weathering", "waiting for materials", "in repair shop") and define allowed transitions between states. State changes MUST be timestamped and attributed to the user making the change. Historical state transitions MUST be preserved for inventory value tracking and insurance documentation.

Rationale: Different cosplay teams have varying processes, skill levels, and complexity requirements that cannot be served by rigid predefined workflows. Cosplayers invest significant money and time in costumes/props—often hundreds of dollars per piece. Tracking lifecycle states (sold, damaged, rented) enables accurate inventory management, helps prevent scheduling shoots with unavailable costumes, supports insurance claims for damaged items, and maintains historical value records for tax/resale purposes. Without granular lifecycle tracking, teams lose visibility into their costume portfolio value and availability.

V. Visual-First Content Management

Reference images, Instagram posts/reels, and progress photos MUST be first-class content types with inline viewing, annotation capabilities, and organized galleries. Visual content MUST be accessible from multiple views (calendar, kanban, detail pages) without navigation friction.

Rationale: Cosplay is fundamentally visual art requiring constant reference to source material, inspiration, and progress documentation.

V.5. Social Media Workflow Integration

The system MUST support seamless integration with Instagram and TikTok for content planning, scheduling, and portfolio management. Teams MUST be able to link Instagram accounts to shoots, plan content pillars (behind-the-scenes, finished costume, WIP, convention coverage), schedule posts with caption templates, and view analytics on published content. Social media planning features MUST support content calendars synchronized with shoot schedules, allowing teams to visualize their posting strategy alongside shoot logistics.

Social Media Features:

- Instagram account connection via OAuth (read/write access)
- TikTok account connection via OAuth for Phase 2+
- Content calendar: view planned posts aligned with shoot dates
- Caption template library: save reusable captions for different content types
- Hashtag management: save and track hashtag sets by content type
- Post scheduling: draft posts and schedule delivery to Instagram/TikTok
- Analytics dashboard: view engagement, reach, saves, and follower growth per post
- Content pillar tracking: categorize posts by type (BTS, finished, WIP, convention)
- Cross-team posting: owner/admin can approve posts before team members publish
- Draft collaboration: team members can draft posts, admins review before posting

Phase 1 Scope: Instagram integration (read/write, basic scheduling, analytics)

Phase 2+ Scope: TikTok integration, advanced analytics, influencer collaboration tools

Rationale: Cosplayers monetize and build community through social media; disconnecting social planning from shoot planning creates friction and prevents strategic content creation.

Instagram-native features (Reels, Stories, feed posts) drive discovery and follower growth. Supporting content calendars aligned with shoots enables teams to plan "shoot → edit → post" timelines holistically. Built-in hashtag and caption templates reduce friction for non-technical team members. Approval workflows prevent low-quality or off-brand posts while enabling team collaboration.

VI. Test-Driven Development

Tests MUST be written before implementation for all new features. Each user story MUST have corresponding test coverage that verifies acceptance criteria. Tests MUST be maintained as features evolve to ensure functionality remains tracked and verifiable. Feature completion is defined as passing tests, not just code existence.

Rationale: Test-driven development ensures features work as intended from inception, provides living documentation of expected behavior, catches regressions early, and enables confident refactoring. Without tests, feature quality degrades over time and bugs propagate undetected.

VI.5. Test Observability & Developer Experience

All automated tests (unit, integration, E2E) MUST be executable and visualizable through the development testing dashboard (spec 043). The dashboard MUST provide real-time test execution, visual component showcases, coverage visualization, and API mock management. UI components SHOULD have visual test stories that demonstrate all interactive states and variations without requiring Playwright tests for every state. Test results

MUST be accessible in-browser during development to provide immediate feedback without context switching to terminal.

Test Dashboard Requirements:

Test execution MUST be triggerable from browser with real-time progress streaming

Test results MUST show pass/fail counts, timing, and detailed error messages

UI components MUST be viewable in isolation with interactive prop controls

Code coverage MUST be visualizable with line-by-line highlighting

API mocks MUST be manageable with inline response editing and delay simulation

Failed tests MUST show expected vs actual comparisons with diff viewer

Test execution MUST complete in <5 seconds for typical unit test suites

Dashboard MUST be excluded from production builds (dev-only routes)

Visual Component Verification:

Components SHOULD have *.stories.ts files for visual verification

Component stories MUST render in isolated sandbox with state controls

Accessibility audits (aXe) SHOULD run automatically for showcased components

Screenshot capture SHOULD be available for visual regression baseline

Coverage Standards:

Minimum 70% code coverage MUST be maintained across all features

Coverage reports MUST be accessible via dashboard with file-tree navigation

Coverage trends MUST be tracked over time to prevent degradation

Warnings MUST be shown if coverage drops below 70% threshold

Developer Productivity:

Debugging E2E failures SHOULD take <50% time compared to CLI-only workflow

Component states SHOULD be verifiable without writing test code

API response variations SHOULD be testable without modifying mock files

Test dashboard SHOULD be adopted by 80%+ of team within 3 months of rollout

Rationale: Immediate visual feedback on test results and component behavior accelerates development cycles, reduces context switching between editor and terminal, and catches issues earlier. Visual component stories enable designers and product managers to verify UI states without running full test suites. Accessible test tooling encourages higher test coverage and adoption of testing practices across the team. Without test observability, developers waste time switching contexts, debugging becomes inefficient, and test-driven development adoption suffers.

VII. Team Roles & Permissions vs. Crew Management

Teams MUST support two distinct role systems:

Team-Level Permissions (global): Owner, admin, member, viewer roles that determine what content a team member can access and modify across all shoots. Team owner has full control and can delegate permissions to admins. Team admins MUST be able to create, modify, and delete shoots, manage team membership, and configure team settings. Regular members MUST be able to view and edit shared content within their assigned permissions. Viewers have read-only access. Permission checks MUST be enforced consistently across all operations (API endpoints, UI actions, real-time sync).

External Crew Member Roles: Crew members with Cosplans accounts (whether discovered through the Creator Community marketplace or manually added) MAY be linked to a shoot as "viewer" or "member" without requiring them to join the team itself. This enables external photographers, makeup artists, and collaborators to participate in specific shoots while maintaining

clean team boundaries. External crew members can view the public detail view of the shoot (dates, location, logistics, visual references) but cannot modify team data or access other shoots unless explicitly invited to join the team. If external crew members later become full team members (joining the team), their account links persist and their role escalates from external crew to team member within their assigned permissions.

Crew Roles (per-shoot): Crew assignments are entirely informational and do not affect permissions. Each shoot MAY have a crew of personnel assigned with multiple roles per person (e.g., one person as both assistant and model). Supported crew roles include: photographer, cosplayer, makeup artist, prop master, hair stylist, assistant, and custom roles. Crew assignments track who was involved in each shoot for historical reference and future collaboration planning.

Visibility & Editing:

All team members can view crew assignments for a shoot (names visible to all)

All team members can view crew contact information (email, phone)

Only team owner and admins can add, edit, or remove crew members from shoots

Owner role has full control; admin role can manage day-to-day operations

Crew Management Interface: A dedicated crew page MUST display all personnel who have worked on shoots with the team, including work history (which shoots they participated in, roles assigned). Crew can be managed in two locations: (1) dedicated crew management page for team-wide personnel administration, and (2) inline within shoot detail views for convenience. Both views MUST support adding, editing, and removing crew assignments.

Team Member Onboarding: Team members joining a team MUST understand their team-level role (admin/member/viewer) and its implications. Team admins MUST be clearly identified for crew assignment and contact information access.

Rationale: Separating team permissions from crew roles simplifies the model: team-level roles control what data a user can access across the app, while crew roles purely document who participated in shoots for accountability and collaboration planning. The owner role provides structural flexibility for future expansion of privileges and governance rules without affecting existing admin/member/viewer permissions. Cosplay teams often work with external photographers, stylists, and models who may not be app users; crew management enables tracking collaborators without requiring them to create accounts. Allowing all team members to view crew contact information enables efficient coordination and scheduling across team members, while restricting management (add/edit/remove) to owner and admins prevents accidental roster changes. Dual crew management interfaces (dedicated page + shoot inline) balance efficiency (bulk crew management) with convenience (quick crew assignment during shoot planning).

VIII. Creator Community & Discovery Marketplace

Cosplayers, photographers, makeup artists, and other specialized roles **MUST** be able to create public creator profiles to build their reputation and be discovered by teams seeking their specific skills. Creator profiles are entirely optional; not all users need to participate in the marketplace. Teams planning shoots **MUST** be able to search for creators by role, geographic proximity, availability, and community rating, making it easy to find and book external collaborators.

IX. Bun Runtime Requirement

All Cosplans development and deployment **MUST** use Bun runtime instead of Node.js. Bun **MUST** be the official runtime for package management, development servers, build processes, and production deployment. This constitutional requirement ensures performance consistency, faster development iteration, and reduced resource usage across all environments.

Bun Usage Requirements:

Package Management: All dependencies MUST be managed via bun install, bun add, bun remove (never npm, yarn, or pnpm)

Development Server: Local development MUST use bun --bun run dev for maximum performance (3x faster installs, 2x faster dev server vs Node.js)

Build Process: Production builds MUST use bun run build and related build commands

Script Execution: All package.json scripts MUST be executed via bun run <script>

Production Runtime: Deployment environments MUST run on Bun runtime where supported

Performance Mandate: Bun provides measurable performance improvements that are constitutional requirements:

Package installations MUST be 3x faster than npm equivalent
Development server startup MUST be 2x faster than Node.js equivalent

Memory usage MUST be 20% lower than Node.js equivalent during development

Build times MUST be equivalent or faster than Node.js equivalent

Migration Policy: Any existing Node.js-based Cosplans installations MUST migrate to Bun runtime. Migration guides MUST be provided for development environment setup. Legacy Node.js support is deprecated and will be removed in future versions.

Rationale: Bun's superior performance directly improves developer productivity and user experience. Faster installs reduce onboarding friction; faster dev servers accelerate iteration cycles; lower memory usage enables development on resource-constrained environments. Constitutional requirement prevents regression to slower Node.js workflows and ensures all contributors benefit from performance improvements. Standardizing on Bun eliminates runtime inconsistencies and debugging issues across different developer environments.

Creator Profile Features:

Basic Info: Public username (separate from real name, can hide real identity), bio/about, profile photo, roles offered

(photographer, makeup artist, prop modeler, cosplayer, hair stylist, assistant, etc.)

Verification Badge: Earned by creators with 90+ days account history, consistent high rating (4.5+ stars), and minimum 5 completed bookings/month. Badge signifies active, trusted community members.

Availability Calendar: Visual calendar showing available dates/times, status (free or booked), and optional rate information per role

Rate Setting: Optional per-role pricing (or "contact for rates").

Rates may differ by role and availability distance

Location & Travel: Public location (zip code level, not exact address), configurable minimum travel distance, option to accept shoots outside normal range with clear distance indicator

Portfolio Links: External links to Instagram, portfolio website, or other platforms showcasing their work

Community Rating: Display of average rating and verified booking count (only counts completed, paid bookings)

Privacy Controls: Option to make entire profile private (hidden from search) or toggle visibility of specific information. Creators can pause profile without deleting it.

Discovery & Search:

Geographic Search: Search for creators within configurable distance radius (default 25 miles, configurable up to 500 miles)

Role Filtering: Filter by specific roles (photographer, makeup artist, prop modeler, etc.)

Availability Filtering: Show only creators available on specific dates/times

Rating/Reviews Filtering: Filter by minimum rating threshold (e.g., 4+ stars only)

Saved Searches: Teams MUST be able to save favorite creators or search profiles for quick re-access. Saved searches are a premium feature accessible to paid tier teams only.

Search Results: Display creator name, location/distance, primary role, rating, and verification status. Results are merit-based (not monetized rankings); featured listings are prohibited to prevent pay-to-play visibility gatekeeping.

Crew Assignment Integration: When assigning crew to a shoot, teams MUST be able to:

Search previously-created crew members (existing history)
Create new crew members manually (not marketplace participants)
Search the Creator Community to find and auto-invite creators matching the role and shoot availability
Auto-suggest creators based on shoot date, location, and required roles
Send one-click email invitations from search results
Teams can view suggested creators sorted by relevance (distance, availability match, rating). Clicking "Invite" triggers email to creator with shoot details (date, location, role, pay rate if applicable) and one-click link to view shoot details and accept/decline.

Creator Invitations & Booking Management:

Creators receive email invitations with shoot overview (date, location, role, pay rate)
One-click link allows creators to view detailed shoot information (visual references, shoot schedule, team contact info) without needing team membership
Creators can accept/decline invitation; status updates team members in real-time
Accepted invitations create shoot_crew associations; external creators appear with "external crew" badge in team views
Free Tier Crew: Free-tier creators can accept bookings and receive invitations but have limited booking management features (cannot view detailed shoot budget, limited communication tools)
Paid Tier Crew: Paid-tier creators can manage their bookings, view shoot budgets aligned with their role, receive booking confirmations, and access invoice/payment history
Team Booking & Payment Management: Teams MUST be able to:

Set pay rates for specific crew roles when creating a shoot
Track which crew members have accepted and will be compensated
Manage team "bank" account (shared budget that team members can contribute to)

Process payments to crew members upon shoot completion with optional automatic transfers
View payment history and invoices for accounting/tax purposes
Crew Booking Complete & Review Process: After a shoot is marked complete:

Teams can submit reviews and ratings (1-5 stars with optional comments) for each external creator who participated
Creators can submit reviews and ratings for the team
Ratings are only applied after shoot completion and payments are finalized
Review system MUST prevent fake/manipulation (only verified bookings count toward rating)
Report & Moderation System:

Users can report fake profiles, harassment, quality issues, or policy violations
Reports go to admin queue with context (screenshot, message, evidence)
Action workflow: assess report → contact involved parties → issue warnings or remove profiles for serious violations
Verified community badge can be revoked if creator receives multiple substantiated complaints or fails to maintain rating/activity standards
Transparent guidelines for what warrants profile removal or suspension
Community Showcase:

Featured gallery of recent verified creator work (Instagram posts, portfolio links, project highlights)
Opt-in participation: creators can manually submit recent projects for community showcase
No algorithmic curation; purely user-driven submissions displayed in chronological order
Showcase serves as marketing & inspiration for teams while building creator reputation
Public Profile Visibility:

Creator profiles are public and indexed for search (team members and non-members can discover them)

Creator profiles do NOT require team membership to view;
anyone can browse marketplace

Creators have full control over what personal information is
shown (can hide real name, show only username)

Crew members invited to specific shoots can view their public
profile as context

Phase Placement: Phase 1.5 (after core team shoot planning is
validated on web, before mobile app development). Allows
marketplace to launch on web with full functionality before
expanding to mobile.

Rationale: Cosplay teams need reliable access to specialized
skills (professional photographers, makeup artists, experienced
cosplayers) to execute complex shoots. Building an in-app
marketplace instead of sending teams to external contractor
platforms (Fiverr, TaskRabbit) keeps workflows unified and
enables data-driven collaboration. Merit-based discovery
(sorted by rating and relevance, not payment) ensures
accessibility for high-quality creators regardless of budget,
preventing pay-to-play gatekeeping. Creator verification badges
reward active community members and help teams quickly
identify trusted collaborators. Geographic search with travel
distance controls respects creator autonomy while enabling
teams to find nearby talent. Commission-based monetization
(5% free tier, 0% paid tier) aligns platform sustainability with
creator success: we only profit when creators get hired. This
revenue model encourages platform growth organically without
extracting value from creator earnings through transaction fees.
Booking management and team budget features enable
transparent, professional workflows between teams and
independent creators, building trust and repeat collaborations.

Platform Requirements

Web Application (Phase 1)

Performance: Initial page load MUST complete within 3 seconds
on 3G networks. Core features MUST be interactive within 5
seconds. Image loading MUST be progressive with low-
resolution previews.

Mobile Responsiveness: All interfaces MUST adapt to screen sizes from 320px to 4K. Touch targets MUST meet minimum 44px size. Form inputs MUST use appropriate mobile keyboards. Gestures (swipe, pinch-to-zoom) MUST be supported where relevant.

Progressive Web App: Service workers MUST enable offline read access to cached shoot details and reference images. Background sync MUST queue offline edits for later submission.

Browser Support: Latest 2 versions of Chrome, Firefox, Safari, and Edge. Mobile browsers: Chrome (Android), Safari (iOS).

Native Mobile Apps (Phases 2-3)

Offline Capabilities: Full read/write access to shoots, costumes, props, and reference images MUST work offline. Changes MUST sync automatically when connection resumes with conflict resolution.

Native Integrations: Camera integration for reference photo capture. Push notifications for shoot reminders and team updates. Location services for venue discovery. Native sharing to social media platforms.

Performance: App launch MUST complete within 2 seconds. Battery usage MUST not exceed social media app benchmarks during normal usage.

Platform Parity: Android and iOS apps MUST have identical features. UI/UX may adapt to platform conventions (Material Design vs. Human Interface Guidelines).

Development Workflow

Feature Priority Order: Core CRUD operations for shoots (SvelteKit web) → Mobile-responsive UI → Calendar integration → Google Maps integration → Social media planning (Instagram content calendar, scheduling, basic analytics) → Advanced views (kanban, map) → Document generation → TikTok

integration → Flutter Android app development → Flutter iOS app development.

Platform Development Flow:

Implement feature in SvelteKit web app with mobile-responsive design

Validate with users on web and mobile browsers

Port to Flutter Android app (if Phase 2 active)

Deploy to iOS using shared Flutter codebase (if Phase 3 active)

Testing Requirements: Tests MUST be written before implementation (test-first approach). Every user story MUST have corresponding tests for web responsive behavior.

Integration tests MUST verify Google API interactions.

Performance tests MUST validate mobile network conditions.

Native app features MUST have platform-specific tests. Test coverage MUST be maintained as features evolve.

Code Review Gates: Test coverage and passing status verification required before review. Mobile-responsive UI changes require testing on multiple device sizes and browsers. External API integrations require error handling and fallback behavior review. Database schema changes require migration plan approval. Permission-related changes require verification of role-based access control across all affected operations.

Security & Privacy Architecture

Data Privacy & GDPR Compliance

All user data MUST be stored in EU-based Supabase infrastructure to comply with GDPR requirements. The system MUST support the right to be forgotten: user account deletion MUST fully remove all associated data (shoots, costumes, props, team memberships) within 30 days. Data retention policies MUST be documented: user data is retained as long as the account is active; deleted accounts have data purged within 30 days; inactive accounts (no login for 2+ years) MAY be archived per user policy.

Third-Party Data Sharing: User data MUST NOT be shared with third parties for monetization. Google APIs (Maps, Calendar, Docs) receive only the minimum data required for their specific function. Email service providers receive only email addresses and notification content necessary for delivery. Analytics services MUST NOT receive personal identifiable information (PII). No data selling or licensing is permitted under any circumstance.

Data Processing Consent: User account creation MUST include explicit consent for data collection (shoots, costumes, team data), analytics collection (heuristics), and external API integrations (Google services, social media platforms). Users MUST be able to opt out of analytics collection without losing core functionality. Social account linking MUST include explicit disclosure of what data is accessed from each platform (e.g., Instagram linking accesses profile info and business account details). Creator marketplace participation requires additional opt-in consent: users must explicitly choose to make their profile public, appear in geographic search, share rating/booking history, and allow teams to review them. Creator privacy controls MUST allow granular toggling (e.g., private profile, hidden exact location, show only username).

Encryption & Transport Security: All data MUST be encrypted at rest in Supabase using industry-standard encryption. All data transmission MUST use TLS 1.3 or higher. API tokens and OAuth credentials MUST be rotated every 90 days. Sensitive data (budget values, personal notes, team member contact info) MUST have field-level encryption.

Incident Response: Security vulnerabilities MUST be disclosed privately via SECURITY.md contact information. Critical vulnerabilities (data breach, authentication bypass) MUST result in user notification within 24 hours. Backup and disaster recovery procedures MUST be tested quarterly with documented recovery time objective (RTO) of 4 hours and recovery point objective (RPO) of 1 hour maximum data loss.

Rationale: Cosplayers manage sensitive personal information (social media, costumes with monetary value, team member

contact details). GDPR compliance builds user trust and ensures legal operation in EU markets. Transparent data practices and ethical heuristics collection (not monetization) differentiate Cosplans from exploitative social platforms.

Authentication & Session Management

OAuth MUST be the primary authentication mechanism. Supported OAuth providers MUST include Google, Instagram/Facebook, and X/Twitter (formerly Twitter). Email authentication with passkey/WebAuthn MUST be available as primary credential option for users without social accounts. Two-factor authentication (2FA) MUST be optional for all users and strongly recommended for team admins; 2FA adoption MAY become mandatory for admin roles in Phase 2 after user feedback.

Supported Login Methods:

Google OAuth: Standard OAuth flow via Google Sign-In

Instagram/Facebook OAuth: Meta-managed credentials (Instagram Business accounts)

X/Twitter OAuth: Standard OAuth flow via X Sign-In

Email + Passkey: Primary credential for non-social-account users; WebAuthn-based

Email Authentication Requirements: Email authentication MUST use passkey/WebAuthn as the primary authentication factor. If user does not have passkey enrolled, email/password authentication MAY be used as temporary fallback (30-day grace period, after which passkey enrollment is required).

Password requirements (if used) MUST enforce minimum 12 characters, mixed case, numbers, and symbols. Password reset MUST use time-limited (15-minute) email confirmation tokens. Users are strongly encouraged to enroll passkeys to replace password-based auth.

Public Usernames & Creator Profiles: Users participating in the Creator Community marketplace MUST have public username (separate from real name) for discoverability. Usernames are used in creator profile URLs and marketplace search results. Users MUST be able to keep real names private (private

account, show only username). Team member profiles (non-creators) are private by default and not indexed in marketplace search.

Session Management: Web sessions MUST expire after 30 days of inactivity OR when user explicitly logs out. Mobile app sessions MAY be longer (90 days) to balance security with user convenience on native platforms. All sessions MUST be terminated on password change, 2FA reconfiguration, or passkey re-enrollment. Session tokens MUST be signed JWTs with user ID, team IDs, and role information for efficient permission checking.

Account Recovery: Users MUST be able to recover account access via email verification or social account re-linking (if originally signed up with OAuth). Recovery tokens MUST expire after 24 hours. Account deletion MUST require confirmation email within 7 days to prevent accidental loss.

Social Account Linking: Users MAY link multiple social accounts (Google, Instagram/ Facebook, X/Twitter) to the same Cosplans account. When multiple social accounts are linked, user can sign in with any of them. Email/passkey account MAY also be linked to social accounts.

Rationale: OAuth via social media (Google, Instagram, X) reduces friction for cosplay community members who already use these platforms daily. Passkeys and 2FA provide defense-in-depth for users without social accounts or preferring passwordless auth. Instagram/Facebook integration aligns with social media planning features (Principle V.5); users can authenticate with the same social account they manage content from. Session management balances security (inactivity timeout) with usability (not requiring constant re-authentication). Multi-account linking flexibility accommodates users who switch between social platforms or prefer different auth methods for different contexts. Public usernames enable creator marketplace discoverability while allowing privacy-conscious users to keep real identity hidden from public profiles.

User Analytics & Ethical Data Collection

Heuristics Collection Principles

User analytics collection MUST focus exclusively on improving product experience and reducing workflow friction. Analytics data collection MUST explicitly exclude personal identifiable information (PII), content data (shoot titles, costume notes, team member names), and financial data (budget amounts, item costs).

Collected Heuristics (approved for tracking):

Feature engagement: which screens users visit, feature adoption rates

Workflow abandonment: where users drop off in multi-step processes (e.g., shoot creation, costume tracking)

Error rates: which forms/features produce errors or validation failures

Performance metrics: page load times, interaction latency, API response times

Accessibility: keyboard navigation usage, screen reader detection

Explicitly NOT Collected:

User or team names, email addresses, or contact information
Shoot titles, costume descriptions, budget amounts, or any content data

OAuth provider details or authentication method

Geolocation data (except aggregated venue popularity for Maps integration improvement)

Device identifiers or IP addresses (only anonymized metrics)

Analytics Data Retention & Opt-Out

Analytics data MUST be retained for 90 days maximum before deletion. Raw event logs MUST be aggregated into weekly and monthly statistics; granular events MUST be purged after 7 days. Users MUST be able to opt out of analytics collection via settings without affecting core application functionality. Teams with paid plans MAY have analytics opt-out honored as contractual commitment.

Analytics-to-Feature Translation

Heuristics data MUST inform product prioritization but MUST NOT be the sole decision driver. High abandonment rates flag workflow friction; feature engagement rates validate user demand; error rates identify quality issues. Manual product review by maintainer (during Phase 1-2) MUST validate that heuristics-identified changes align with core principles (mobile-responsive, accessibility, real-time collaboration).

Rationale: Ethical heuristics collection enables data-driven product improvement without exploiting user data. Transparent opt-out and strict PII exclusion maintain user trust. Analytics-to-prioritization process ensures features solve real problems rather than chasing engagement metrics that drive friction (common in social platforms).

Sustainability Model & Feature Paywalls

Cosplans MUST be economically sustainable through a freemium model that drives adoption while enabling platform maintenance and team growth. Revenue comes from two sources: subscription tier differentiation for team features and commission on creator marketplace transactions.

Team Free Tier

Storage: 2 GB per team (shared across all shoots, costumes, props, reference images)

Team members: Unlimited team members per organization

Shoots: Unlimited shoot planning

API requests: 1,000 API calls per day (sufficient for typical team workflows)

Crew management: Manually create crew members, invite external crew to specific shoots

Creator marketplace: Search and book creators; pay creators through Cosplans (5% commission deducted from payment)

Google integrations: Calendar sync, Maps, Docs

Basic email reminders: Shoot dates, upcoming events

Instagram integration: Content calendar, draft creation, manual post scheduling, basic analytics (Phase 1.5+)

Export: Limited to 1 export per week

Team Paid Tier (\$5/month)

All free tier features plus:

Storage: 20 GB per team

Creator marketplace: Book creators with 0% commission
(platform absorbs cost as customer acquisition)

Saved creator searches: Save and organize favorite
creators/searches for quick access

Advanced crew management: Bulk crew import, crew availability
tracking, role templates, in-app messaging with crew

Advanced email features: Custom reminders, scheduled
notifications, email digests

Instagram integration: Auto-scheduling (publish at optimal
times), advanced analytics (trending hashtags, follower insights,
competitor analysis)

Inventory valuation: Track costume/prop costs and lifecycle for
insurance documentation

Team budget & analytics: Shared team bank account, payment
tracking, spending insights, cost per shoot analysis

Advanced shoot analytics: Time tracking, ROI analysis, cost per
photo metrics

Priority support: 48-hour response time

Creator Free Tier

Public profile: Roles, availability calendar, ratings

Accept bookings: Marketplace commissions 5% on payments
received

Basic booking management: View invitations, accept/decline
bookings

Profile privacy: Full control over personal info visibility; can use
public username only

Creator Paid Tier (\$5/month)

All free tier features plus:

0% commission: Creators receive 100% of agreed pay (platform
absorbs commission)

Advanced booking management: Custom rate quotes, invoice
generation, payment history

Detailed shoot budgets: View what team allocated for creator's
specific role

Professional communication: In-app messaging with teams, scheduling assistance

Booking analytics: Track earnings, predict monthly income, export reports

Calendar integration: Sync bookings to Google Calendar/Outlook

Portfolio management: Upload portfolio samples to Cosplans profile

Referral rewards program: Earn commissions on referred creators/teams

Commission-Based Revenue Model

Marketplace transactions: 5% platform commission on all bookings for free tier users (both teams and creators)

Free tier cap: Max 50 transactions per month per user; users exceeding limit must upgrade to paid tier

Paid tier creators: 0% commission; platform absorbs fee as user acquisition strategy

Paid tier teams: 0% commission; encourages teams to book more creators (ecosystem growth)

Break-even analysis: Assuming avg \$50 commission/month per active free tier user (5% fee on ~\$1000 spend), platform reaches sustainability at ~200 active creators or 100 teams + 50 creators in paid tier. Achievable in Year 1 with modest marketing.

Self-Hosting & Code Licensing

Self-hosting is NOT supported in Phase 1. Codebase licensing MUST be decided before Phase 2: options under consideration are MIT (permissive, allows commercial use), AGPL (requires open-source derivatives), or proprietary. Decision MUST balance user trust (open-source visibility) against sustainability (preventing competitors from hosting free instances).

If open-source licensing (MIT/AGPL) is chosen, self-hosting documentation MUST be provided in separate repository. Self-hosted instances are NOT supported officially; security updates and data privacy compliance are the administrator's responsibility.

Payment Processing

Stripe integration (deferred until paid tier feature set is locked):

Manages team subscription billing and creator payment disbursements

Escrow model: Cosplans holds payment in transit until shoot marked complete and payments approved (protects both teams and creators)

Payment frequency: Weekly or on-demand creator payouts (via Stripe Connect or ACH)

Payment visibility: Both teams and creators see payment history, invoices, status in real-time

Infrastructure Cost Model

Supabase infrastructure costs (database, storage, API calls) scale with user growth. Free tier limits are calibrated to keep baseline monthly cost under \$50 per team (achieved through aggressive caching, image optimization, and API rate limiting). If infrastructure costs exceed sustainability threshold (defined as 50% of paid tier + marketplace revenue), pricing MAY be adjusted or storage limits MAY be reduced. Major pricing changes REQUIRE 90-day notice to affected users.

If Cosplans becomes economically unsustainable (costs exceed available budget indefinitely), service discontinuation MUST include 180-day data export window (backup downloads available free to all users) and code repository open-sourcing under MIT license to prevent user data trap.

Future Revenue Streams (Deferred)

Referral rewards: Earn commission for referring new users (Phase 2+ after marketplace stable)

SMS reminders: Optional paid add-on for SMS vs. email (Phase 2+)

Advanced analytics: Executive dashboards, trend analysis, forecasting (Phase 2+)

B2B onboarding: Consulting/training for large cosplay organizations (Phase 3+)

Event partnerships: Commission on Cosplans-sponsored convention integrations (Phase 3+)

Rationale: Commission-based pricing (not membership required) enables any creator to participate regardless of budget. Merit-

based discovery prevents pay-to-play gatekeeping. Paying teams 0% commission at paid tier saves money only after \$100/month spend, creating clear ROI. Revenue increases with platform adoption (more transactions = more commission); no predatory pricing or surprise costs. Aligns incentives: teams want active creators (better marketplace), creators want visibility (benefit from paid tier), platform wants ecosystem growth (revenue from scaling). Deferred Stripe integration allows marketplace to prove product-market fit on web before implementing complex payment flows.

Technical Architecture & Implementation Standards

Backend & API Design

All backend APIs MUST be versioned (v1, v2, etc.) with clear deprecation timeline (minimum 6-month notice before endpoint removal). Database schema MUST follow normalization principles (third normal form minimum) to prevent data anomalies. API documentation MUST be auto-generated from OpenAPI/Swagger specification and kept in sync with implementation.

Data Consistency: Real-time data synchronization MUST handle concurrent edits using operational transformation (OT) or conflict-free replicated data types (CRDT). When users make simultaneous edits to the same shoot/costume, the system MUST resolve conflicts deterministically and notify both users of the resolution. Offline edits MUST be queued locally and reconciled on reconnection; if conflicts exist, user MUST be prompted to review and resolve rather than silently overwriting.

Rate Limiting & Quotas: API rate limits MUST be enforced per tier: free tier (1,000 API calls/day), paid tier (unlimited). Requests exceeding quota MUST return HTTP 429 with retry-after header. Quota usage MUST be visible in user settings. Teams approaching quota limits MUST receive email warning at 80% and 95% utilization.

External API Resilience: All external API calls (Google Maps, Google Calendar, Instagram, email providers) MUST implement

circuit breaker pattern with exponential backoff. Timeout for external calls MUST be 60 seconds maximum. Failed integrations MUST NOT corrupt local state; errors MUST be captured, logged, and surfaced to user with retry option. Calendar sync failures MUST be queued for retry; failed email sends MUST not block shoot creation. Instagram post scheduling failures MUST queue drafts locally for manual retry.

Request/Response Format: APIs MUST use JSON exclusively. Response payloads MUST include pagination metadata (page, total, limit) for collections. Error responses MUST include error code, human-readable message, and reference to affected resource. API responses MUST include cache headers (ETag, Last-Modified) to enable client-side caching.

Recommended Technology Stack & Packages

The implementation MUST use the following vetted libraries and frameworks to accelerate development while maintaining code quality and architecture integrity:

Core Framework & UI:

SvelteKit (web framework) - built-in server-side rendering, API routes, file-based routing, and SSG

Tailwind CSS (styling) - utility-first CSS framework

Shadcn/svelte (component library) - accessible, copy-paste components built on Radix UI

Lucide Svelte (icons) - clean SVG icon library

Icon Import Requirement: When using Lucide icons in components, ALL icons MUST be explicitly imported in three places:

Import statement in `/src/lib/components/icons/LucideIcon.svelte` from 'lucide-svelte'

Added to the ICONS record in `LucideIcon.svelte`

Added to the `LucideIconName` type in `/src/lib/types/navigation.ts`

Missing any of these three steps causes icons to render as default box icons instead of the intended icon

Rationale: Lucide Svelte requires explicit icon registration; tree-shaking only includes imported icons in bundle

Theme System & Styling Requirements:

CSS Variable Requirement: ALL component styling MUST use CSS custom properties (theme variables) instead of hardcoded Tailwind color classes. This ensures components automatically adapt to theme changes (light/dark mode, custom themes).

Required Theme Variables:

- theme-background: Main background color for page content
- theme-foreground: Primary text color
- theme-sidebar-bg: Sidebar/navigation background
- theme-sidebar-border: Border colors for sidebar and UI elements
- theme-sidebar-text: Primary sidebar text color
- theme-sidebar-muted: Secondary/muted text in sidebar
- theme-sidebar-accent: Accent color for active/hover states
- theme-sidebar-hover: Hover background color
- theme-sidebar-active: Active item background color

Forbidden Patterns: Direct use of Tailwind color classes (e.g., text-slate-900, bg-white, border-blue-500) is PROHIBITED in components that should adapt to theme changes. Use text-[var(--theme-foreground)], bg-[var(--theme-background)], border-[var(--theme-sidebar-border)] instead.

Exception Cases: Marketing/landing pages and public documentation MAY use direct Tailwind colors if they have fixed branding requirements independent of user theme preferences.

Rationale: Theme variables enable seamless dark mode, custom themes, and accessibility (high contrast modes) without rewriting component styles. Hardcoded colors break when users switch themes and create visual inconsistency. CSS variables add minimal bundle size (~2KB) while providing dynamic theming capabilities essential for user customization and accessibility compliance.

Routing & Navigation:

SvelteKit file-based router (built-in, zero-overhead) - MUST be used exclusively for all navigation

Navigation APIs: goto() for programmatic navigation, beforeNavigate() for guards/confirmations, afterNavigate() for state updates, \$page store for current route

Third-party routing libraries MUST NOT be used (adds bundle size, bypasses SvelteKit optimizations)

Route structure MUST support: authenticated routes (dashboard, team context), public routes (landing, about, help, contact), auth routes (/login, /signup)

Rationale: SvelteKit's file-based router provides optimal performance (automatic code splitting, preloading, streaming SSR) with zero configuration. Native APIs integrate seamlessly with layouts, load functions, and form actions. No third-party router can match this performance while maintaining SvelteKit's developer experience.

Form Handling & Validation:

Sveltekit-Superforms (form state management) - server-first validation, CSRF protection, progressive enhancement

Zod (schema validation) - TypeScript-first schema validation with runtime type checking

Rationale: Superforms + Zod reduces form boilerplate by 10-15 days and prevents validation logic duplication

Real-Time Sync & Conflict Resolution:

Yjs (CRDT library) - replaces manual OT algorithm implementation, handles conflicts automatically

y-protocols (WebSocket provider for Yjs) - integrates Yjs with custom sync protocol

Rationale: Yjs eliminates 10-14 days of complex OT algorithm development; battle-tested in Figma, Notion

Database & Backend:

Supabase (PostgreSQL + auth + realtime) - includes PostgREST API, RLS, real-time subscriptions

Sharp (image processing) - optimized Node.js image resizing and WebP conversion

date-fns (date utilities) - modular date manipulation library

Authorization & Permissions:

@casl/ability (authorization library) - declarative permission rules (server + client)

Rationale: Centralized permission rules prevent scattered if-statements; 3-5 days saved on permission scaffolding

Testing:

Vitest (unit testing) - Vite-native test runner, 10-15% faster than Jest

@playwright/test (E2E testing) - browser automation for critical workflow testing

msw (Mock Service Worker) - API mocking for tests (mocks Supabase, SendGrid, Google APIs)

@testing-library/svelte (component testing) - behavior-focused component testing utilities

Rationale: MSW eliminates flaky external API tests; 5-7 days saved on test setup and mocking

Dependencies: @supabase/supabase-js (SDK), date-fns (date utilities)

Do NOT use (antipatterns for SvelteKit):

✗ Redux, Pinia, Zustand - state management overkill; use SvelteKit load() + writable stores

✗ Third-party analytics vendors - use custom PostgreSQL analytics per Principle IX

✗ Auth0, Okta - Supabase Auth is sufficient; avoids vendor lock-in

✗ Material Design, Bootstrap - Shadcn/svelte is lighter and more customizable

Implementation Time Savings: Using recommended packages reduces Phase 1 MVP timeline from 180-220 working days to 60-80 working days (100-140 days saved, ~55% time reduction).

Packages MUST be locked to specific versions in package.json; version upgrades MUST follow semantic versioning and be tested in staging before production deployment.

Data & Media Storage

Image uploads MUST be optimized: WebP format with quality 75 for web, fallback JPEG for older browsers. Responsive image sizes MUST be generated at upload (320px, 640px, 1280px, 2560px) for efficient mobile delivery. Thumbnails MUST be auto-generated within 30 seconds of upload for preview display.

Crew & Personnel Records: Crew members are individuals who have worked on shoots and MUST have a persistent record in the system (name, contact information). Multiple crew members

can be assigned to a single shoot with different roles. Crew role assignments are stored per-shoot, not globally. Contact information (email, phone) MUST be stored separately from shoot records and access-controlled (admins only). Crew records track collaboration history across all shoots for team reference.

Media CDN: Images MUST be served from CDN with 30-day cache expiration. Cache headers MUST include ETag for invalidation if images are edited. Progressive image loading MUST show low-resolution preview immediately, then load full resolution as image loads.

Database Backups: Database backups MUST be automated daily. Weekly restoration tests (full backup → restore to test database → verify data integrity) MUST be scheduled to ensure backups are actually restorable. Backup retention MUST be 30 days minimum. Disaster recovery procedure MUST have documented RTO (recovery time objective) of 4 hours and RPO (recovery point objective) of 1 hour maximum data loss.

Encryption at Rest: Sensitive data MUST be encrypted at field level in database: budget amounts, personal notes, user contact information. Encryption keys MUST be rotated annually. Master encryption key MUST be stored in secure key management service (not in code or environment variables).

Real-Time Synchronization

Offline edits MUST be queued locally with timestamp and user attribution. When connection resumes, queue MUST be processed in order with conflict detection. If conflict exists (server state differs from queue edit), both versions MUST be shown to user with 3-way merge view (original, local edit, server edit). User MUST manually resolve conflict; unresolved conflicts MUST block sync until resolved.

Sync Performance: Real-time updates MUST propagate to all team members within 2 seconds of edit submission. Sync MUST not block user from making additional edits (optimistic updates).

If server rejects edit, user MUST be notified with error reason and offered conflict resolution.

Testing & Quality Assurance

All new features MUST have accompanying unit tests (minimum 70% code coverage) and integration tests. External API integrations MUST be mocked in tests to enable deterministic testing without depending on Google services availability. Test database MUST be isolated per test run (fresh database state, no test data leakage).

Performance Testing: Web application MUST be load-tested before production deployment. Performance benchmark: initial page load <3 seconds on simulated 3G network (1.6 Mbps download, 750 kbps upload). Interaction latency <200ms for all UI interactions. Mobile apps MUST be tested on actual iOS and Android devices, not emulators only.

End-to-End Testing: Critical workflows (create shoot → add costumes → share with team → sync calendar) MUST have automated end-to-end tests using Playwright or equivalent. Permission enforcement MUST be tested: verify that viewers cannot edit, that only admins can delete shoots, that team members cannot access other teams' data.

Accessibility Testing: All interfaces MUST meet WCAG 2.1 AA compliance. Keyboard navigation MUST work for all interactive elements (no mouse-only workflows). Color contrast MUST be 4.5:1 minimum for normal text. Screen reader testing MUST verify that images have alt text and form inputs have labels. Accessibility tests MUST be automated where possible (axe-core, Lighthouse).

Deployment & Operations

Deployments MUST be automated via CI/CD pipeline. Code changes MUST pass all tests before deployment proceeds. Staging environment MUST mirror production configuration; all deployments MUST be validated in staging before production

release. Database migrations MUST include rollback procedures and be tested in staging first.

Environment Management: All configuration (API keys, database URLs, feature flags) MUST be managed via environment variables or secure vault service (not hardcoded). Separate secrets MUST be used for dev, staging, and production environments.

Monitoring & Alerting: Real-time error monitoring MUST track errors from all layers (backend, frontend, integrations). Error rate >1% during 5-minute window MUST trigger alert. API latency >1 second (p95) MUST trigger alert. Database query latency >500ms MUST trigger alert. Monitoring dashboard MUST be accessible to maintainer with historical trending (last 7 days, last 30 days).

Logging Standards: All errors, API calls, and permission-sensitive operations MUST be logged with timestamp, user ID, action, and result. Logs MUST be retained for 30 days minimum. Sensitive data (passwords, tokens, personal info) MUST NEVER be logged. Log levels (DEBUG, INFO, WARN, ERROR) MUST be configurable per environment.

Incident Response: Critical incidents (data loss, authentication bypass, major outage >30 minutes) MUST be documented with: incident description, timeline, root cause, resolution, and preventive measures. Post-mortems MUST result in actionable fixes or monitoring improvements.

State Management (Frontend)

Client-side state MUST be managed consistently across web and mobile applications. Form state MUST persist across navigation to prevent data loss. Component state (expanded/collapsed, sorted) MAY be ephemeral (lost on page reload) unless it's critical to workflow.

Client Caching: API responses MUST be cached locally for 5 minutes before re-fetching. Cache invalidation MUST happen on user action (create/edit/delete) or explicit refresh. Stale cache

MUST be displayed immediately while fresh data loads in background (stale-while-revalidate pattern).

Undo/Redo: Data mutations (create, edit, delete shoots/costumes) MUST support undo for 5 minutes after action. Undo history MUST NOT survive page reload. Undo MUST NOT revert permission changes or role assignments (prevent accidental privilege escalation undo).

Internationalization & Accessibility

Phase 1 MUST support English only. Phase 2 MAY add internationalization (i18n) framework for future language support. All user-facing text MUST be extracted to translation files to enable easy localization. Date/time formatting MUST be locale-aware (not hardcoded MM/DD/YYYY).

Accessibility Scope: Keyboard navigation MUST be fully functional (Tab, Shift+Tab, Enter, Escape, Arrow keys). Screen reader support MUST be tested with NVDA (Windows) and VoiceOver (macOS/iOS). Focus indicators MUST be visible (outline, background color). Skip-to-main-content link MUST be present on every page.

Rationale: Technical requirements provide guardrails for implementation decisions, ensure consistency across platforms, and prevent common pitfalls (data loss on sync, performance regressions, accessibility compliance). Clear standards enable faster development and reduce rework when migrating features between web and mobile apps.

Governance

Constitution supersedes all feature decisions and architectural choices. All pull requests MUST verify mobile-responsive compliance, integration integrity, test-driven development compliance, and role-based permission enforcement. Feature complexity beyond core cosplay workflows MUST be justified against user adoption impact.


Amendment Process: Constitution changes require documentation of impact on existing integrations, mobile performance, security/privacy posture, sustainability model, and technical architecture. Manual conflict review occurs when implementation reveals tension between principles; maintainer reviews conflict, documents decision rationale, and updates constitution or implementation accordingly. Major principle changes (new core tenets, removal of existing principles) require documented rationale and alignment with user feedback.

Version: 2.4.0 | Ratified: 2025-10-16 | Last Amended: 2025-10-18
(Test observability principle added, spec 043 created)




Section / Principle	Conflict / Redundancy Noticed	Proposed Resolution & Rationale
P I. Web-First	Sets SvelteKit performance goal at <3 sec loads on 3G.	STATUS: KEEP. This is a perfect, measurable goal that reinforces the Bun/SvelteKit choice. The Bun Runtime Requirement (P IX) strongly supports this goal.
P II.5. Team Ownership	Deletion Rule: "A team can only be deleted when there	STATUS: CONFLICT WITH LATEST PLAN. This should be simplified to align with the "Permanent Portfolio" principle.

Section / Principle	Conflict / Redundancy Noticed	Proposed Resolution & Rationale
	is only one member remaining AND that member owns at least one other team.	Resolution: REMOVE the restrictive deletion rule. Replace it with: "The Personal Team (auto-created) MUST NOT be subject to deletion; it is only removed upon full user account deletion. Collaborative/Temp Teams follow the P3 Deletion Policy."
P III. External Integration	Status is currently broad, but the specific, high-cost integration details (Stripe, XMP Sync) are now in the roadmap.	STATUS: REFINED. Keep this principle as-is but ensure all <i>new</i> integrations (like XMP/Lightroom Sync) adhere to the P6 Deferral timeline due to complexity/cost.
P VII. Crew Management	Mentions Creator Community Marketplace and 5% commission multiple times within the permission section.	STATUS: REDUNDANCY. This financial logic is better centralized in the Sustainability Model section. Keep the core definitions of Team-Level Permissions (Owner/Admin/Member) and External Crew Roles separate and simple here.



Missing Principle / Linkage	Necessary Addition	Placement
Trust, Forgiveness, & Tiering	NEW: P VIII. Trust & Accountability (Tiered): The system's integrity relies on the Reputation System (Tiered) . All public collaboration features must use Tiers for vetting. The system MUST include a Reputation Grace mechanism to mitigate penalties for emergencies.	Elevate the Reputation System from a feature (mentioned in P VIII) to a core architectural principle.
Financial/Competitive Edge	NEW: P X. Sustainable & Commission-Free Model: Monetization relies on a three-pillar model (Subscription, Fees, A La Carte). The platform MUST	Centralizes the core competitive and financial advantage,

Missing Principle / Linkage	Necessary Addition	Placement
	provide 0% commission on all transactions for Paid Tier users (both creators and teams) to be competitively dominant. The free tier commission is 5%.	ensuring AI agents prioritize this logic.
AI Usage Restriction	NEW: P XI. Assistive AI & Opt-In: All AI features MUST be Opt-In by the user. AI is strictly limited to planning and organizational assistance and MUST NOT generate final creative or public-facing content without explicit disclosure ([AI-Assisted] tag).	Crucial safety and trust principle to prevent misuse and ensure compliance with ethical guidelines.
Solo Developer Focus	NEW: P XII. Solo-Viability and Cost Control: Design and development MUST prioritize RLS (Row Level Security) for logic enforcement over high-cost Edge Functions, minimizing operational costs to support the solo-bootstrapped strategy.	A constitutional mandate to enforce the low-cost infrastructure priority.



Section	Old Wording	New Wording to Use	Rationale
Sustainability Model	Free Tier Creator Commission: 5% commission deducted from payment.	Free Tier Creator Commission: 5% platform fee deducted from payment.	Use "platform fee" instead of "commission" to be precise, as this is a transactional fee, not a commission on a print sale.
Sustainability Model	Paid Tier Creator: 0% commission; platform absorbs cost as customer acquisition strategy.	Paid Tier Creator: 0% Platform Fee; they receive 100% of the agreed pay.	Emphasizes the competitive advantage (100% payout) more clearly.

Section	Old Wording	New Wording to Use	Rationale
Platform Requirements	Native Mobile Apps (P2-3): Offline Capabilities: Full read/write access...	Native Mobile Apps (P2-3): Offline Capabilities: Full read/write access... utilizing Yjs for CRDT-based conflict resolution on sync.	Links the feature requirement directly to the specific solution chosen in the Technical Architecture section (Yjs), making the requirement clearer.
