OpenClaw

MVP Launch Strategy & Revised Development Plan

Managed AI Agent Rental Service for Shorts/Reels Creators

Document ID	OC-MVP-003
Version	3.0
Date	2026-02-03
Author	Jo Heejin
Role	Lead / Responsible Person
Status	Approved for Execution
Classification	Confidential - Internal Use Only

1. Executive Summary

The original v2.0 plan was designed for a 1-million-user scale, requiring 6 months of development with complex isolation technologies (Workers for Platforms Sandbox) and distributed state management (Durable Objects). This is over-engineering for initial market entry.

This v3.0 plan pivots to a "Fast but Robust" strategy: launch a revenue-generating MVP in 4 weeks (1 week preparation + 3 weeks development) while maintaining the security-critical payment verification architecture from v2.0.

Item	v2.0 (Previous)	v3.0 (Revised)
Timeline	24 weeks (6 months)	4 weeks (1 prep + 3 dev)
Architecture	Microservice (Workers for Platforms, Sandbox, Durable Objects)	Simple Monolith (Workers + D1)
Agent Execution	Per-tenant sandbox isolation	Trusted code execution (direct function call)
Concurrency Control	Durable Objects per-tenant state	D1 SQLite Transaction (BEGIN IMMEDIATE)
Payment	PortOne V2 + NICE Payments (full feature)	PortOne V2 + NICE Payments (card + simple pay only)
Infra Cost (monthly)	$15-50 estimated	Under $5 (Workers free tier + D1 free tier)
Break-even	333 users (at full infra cost)	Under 50 users (minimal infra)
Target DAU	10,000+	500 (initial validation)

Expected outcomes:

- 90% reduction in infrastructure cost
- 5x improvement in development velocity
- Immediate cash flow generation within 4 weeks
- Security posture maintained (Zero-Trust Billing preserved)

2. Core Middleware Architecture

This is the most critical section. It explains how we maintain payment integrity and security after removing Durable Objects and Sandbox -- the two most complex components from v2.0.

2.1 Architecture: Simple Monolith

Instead of a microservice architecture, we consolidate into a single serverless structure built on Cloudflare Workers and D1.

Layer	Technology	Role
Frontend	Next.js (Vercel or Cloudflare Pages)	SSR/SSG, PortOne SDK, Dashboard SPA
Auth	Clerk	Kakao/Google OAuth, session management, JWT
Middleware (API)	Cloudflare Workers + Hono Framework	Business logic, payment verification, credit management
Database	Cloudflare D1 (SQLite)	Transaction management, credit ledger, user data
Storage	Cloudflare R2	Agent output files (scripts, images)
AI	OpenAI API (direct call from Workers)	GPT-4o for script generation, trend analysis
Payments	PortOne V2 + NICE Payments	Card, KakaoPay, NaverPay, TossPay

[NOTE] Hono Framework is chosen for Workers because it is lightweight (~14KB), has built-in middleware support, and is the de facto standard for Cloudflare Workers API development. No Express.js or Fastify -- those are Node.js-specific and incompatible with Workers runtime.

2.2 Payment Verification Logic (Zero-Trust Billing)

This is the security architecture that we carry over from v2.0 without compromise. The fundamental principle: never trust frontend data for payment amounts. The server independently verifies every payment via PortOne API before granting credits.

Payment Flow (9 steps):

1. Order Creation (Server-side). When the user selects a credit package, the frontend calls POST /api/credits/prepare. The backend creates a PAYMENT_ORDER record in D1 with: merchant_uid (unique), expected_amount (KRW), package_id, user_id, status=PENDING. Returns merchant_uid to the frontend.

2. PortOne SDK Invocation (Frontend). Frontend calls IMP.request_pay() with the merchant_uid and amount from the server. PortOne SDK opens NICE Payments checkout UI.

3. User Authentication (PG). User completes card/KakaoPay/NaverPay/TossPay authentication on NICE Payments UI.

4. Callback (Frontend). PortOne returns imp_uid + merchant_uid to the frontend callback. Frontend relays imp_uid to backend via POST /api/credits/verify.

5. Cross-Verification (Server -- CRITICAL). Backend calls PortOne REST API: GET /payments/{imp_uid}. Retrieves actual paid amount, status, and payment method from PortOne.

6. Three-Way Validation. Backend checks: (a) PortOne status == 'paid', (b) PortOne amount == PAYMENT_ORDER.expected_amount in D1, (c) merchant_uid matches. ALL three must pass.

7. Credit Grant (Atomic D1 Transaction). If validation passes: BEGIN IMMEDIATE TRANSACTION -> INSERT CREDIT_TRANSACTION (type=PURCHASE) -> UPDATE CREDIT_BALANCE (increment) -> UPDATE PAYMENT_ORDER (status=COMPLETED) -> COMMIT. If any step fails, ROLLBACK. No partial credit grants.

8. Failure Handling. If validation fails: PAYMENT_ORDER status set to FAILED, reason logged (amount_mismatch / status_invalid / uid_mismatch), admin alert triggered. Zero credits granted.

9. Success Response. Frontend receives success, updates credit balance display, shows transaction confirmation.

[NOTE] Step 5-6 is the single most important security control in the entire system. An attacker could modify the amount parameter in the PortOne SDK call via browser devtools (e.g., pay 100 KRW but request 99,000 KRW worth of credits). The server-side cross-verification against the pre-created PAYMENT_ORDER makes this attack impossible.

2.3 Concurrency Control (Without Durable Objects)

Concern: Without Durable Objects, concurrent requests modifying the same user's credit balance could cause data corruption (race condition).

Solution: D1 SQLite Transaction with BEGIN IMMEDIATE. SQLite's BEGIN IMMEDIATE acquires a write lock at transaction start, preventing other write transactions from proceeding until the current one completes. This serializes credit operations per-user without requiring a distributed lock manager.

Practical capacity analysis:

- D1 write throughput: approximately 50-100 write transactions per second on a single database.
- Scenario: 500 DAU, average 5 agent runs per day = 2,500 credit deductions per day = approximately 0.03 transactions per second. This is 0.03% of D1's write capacity.
- Even at 5,000 DAU (10x growth), we would use approximately 0.3% of D1's write capacity.
- Durable Objects become necessary only when per-user write contention exceeds D1's single-writer throughput -- realistically at 50,000+ concurrent active users.

[NOTE] The '10,000 concurrent user' claim in the original draft was overly aggressive for D1. A conservative and honest estimate is: stable operation up to 500 DAU initially, scalable to 5,000 DAU without architectural changes, and Durable Objects migration path available within the Cloudflare ecosystem when needed.

2.4 Agent Execution Security

Change: Workers for Platforms (Sandbox) removed entirely.

Rationale: Sandbox isolation is required when executing untrusted user-uploaded code. OpenClaw agents are trusted code written and deployed by us. Each agent is a function that: (1) receives user configuration as

input, (2) calls OpenAI API with a curated prompt template, (3) returns structured output. No user-supplied code is executed at any point.

Removing the isolation layer actually improves performance: no sandbox initialization overhead (estimated 50-100ms saved per execution), direct access to D1 and R2 bindings, simpler error handling and logging.

Security controls maintained without Sandbox:

- Input validation: all user-provided configuration fields are sanitized server-side.
- Output size limits: agent outputs capped at 100KB text, 5MB images.
- Execution timeout: 30-second hard limit per Workers invocation.
- Rate limiting: per-user execution rate limit (10 runs per minute, 100 per hour).
- API key security: OpenAI API key stored in Workers secrets, never exposed to client.

3. MVP Feature Scope

Aggressive scope cutting to enable 3-week development. Every removed feature has a specific technical or business justification.

3.1 Must-Have (MVP Launch)

Feature	Implementation	Estimated Effort
Authentication	Clerk + Kakao OAuth (primary) + Google OAuth	< 1 day (Clerk handles everything)
Credit Purchase	PortOne V2 + NICE Payments: card, KakaoPay, NaverPay, TossPay	2-3 days (including verification logic)
Credit Balance & History	D1 ledger with transaction history page	1 day
Shorts Script Agent	OpenAI GPT-4o with curated prompt template	2-3 days (including prompt tuning)
Trend Research Agent	YouTube Data API + OpenAI analysis	1-2 days
Agent Result Display	Text output inline, download as .txt/.md	0.5 day
Dashboard	Credit balance, recent runs, quick action buttons	1-2 days
Landing Page	Hero section, pricing, CTA	1 day

3.2 Removed / Deferred to v2.0

Removed Feature	Reason	When to Add
Virtual Account (vbank)	Async deposit confirmation adds webhook complexity. 90%+ of Korean online payments are card or simple pay.	When users explicitly request it
Automated Refund UI	Complex edge cases (partial consumption, FIFO tracking). Replace with manual CS refund via admin DB.	After 100+ paying users
Agent Marketplace UI	Search/filter/rating system is premature. Launch with 2 killer agents shown directly on dashboard.	After 5+ agents available
Sandbox Isolation	Unnecessary for trusted code execution (Section 2.4).	Only if user-uploaded agents are introduced
Durable Objects	D1 transactions sufficient for initial scale (Section 2.3).	When DAU exceeds 5,000
Cross-Platform Posting Agent	Requires YouTube/Instagram API approval process (weeks). Focus on content generation first.	After YouTube API approval obtained
Thumbnail Generation Agent	Requires image generation API (DALL-E/Midjourney) integration and cost management.	After revenue validates the model
Subscription/Recurring Billing	Adds billing lifecycle complexity. Simple prepaid credit model is sufficient for MVP.	When monthly retention data proves demand

4. 4-Week Development Roadmap

The roadmap is structured as 1 week of preparation (non-coding) followed by 3 weeks of development. Week 0 runs in parallel with early Week 1 tasks where possible.

Week 0: Preparation (Non-Coding)

Goal: Remove all blockers before coding begins.

Task	Owner	Dependency	Estimated Time
Register PortOne merchant account (admin.portone.io)	Jo Heejin	Business registration certificate	1 day (apply)
NICE Payments PG review application	Jo Heejin	PortOne account	3-5 business days (review)
PortOne sandbox/test mode setup	Jo Heejin	PortOne account	Same day
Clerk application setup + Kakao Developers app registration	Jo Heejin	Kakao developer account	0.5 day
Cloudflare account + Workers/D1/R2 provisioning	Jo Heejin	Credit card for Cloudflare	0.5 day
OpenAI API key provisioning + usage limit setup	Jo Heejin	OpenAI account	0.5 day
Domain registration (openclaw.co.kr) + DNS setup	Jo Heejin	Domain registrar	0.5 day
Git repository + CI/CD pipeline (GitHub Actions)	Jo Heejin	GitHub account	0.5 day

[NOTE] NICE Payments PG review is the critical-path item in Week 0. It requires a valid business registration certificate (saeopja deungnok jeungmyeongseo) and takes 3-5 business days. Apply on Day 1. Development can proceed using PortOne sandbox mode while the review is pending.

Week 1: Infrastructure & Payment ("Build the Cash Register")

Goal: A user can sign up, purchase credits, and see them in their balance.

Day	Task	Deliverable
Day 1-2	Next.js project setup + Cloudflare Workers API (Hono) + D1 schema migration	Working dev environment, DB tables created
Day 3	Clerk integration: Kakao OAuth + Google OAuth + session middleware	User can sign up and log in via Kakao
Day 4-5	PortOne V2 integration: prepare endpoint, SDK invocation, verify endpoint, D1 atomic credit grant	User can purchase credits with test card (sandbox mode)
Day 6	Credit balance page + transaction history + 100 free credits on signup	Full credit lifecycle visible in UI

Day	Task	Deliverable
Day 7	Security testing: amount tampering test, concurrent transaction test, error handling	All payment edge cases handled

Week 2: Agent Engine ("Build the Value")

Goal: A user can run the Shorts Script Agent and see useful output.

Day	Task	Deliverable
Day 1-2	OpenAI API integration in Workers + prompt engineering for Shorts script generation	Agent produces usable 30-60 second Shorts scripts
Day 3	D1 credit deduction logic with BEGIN IMMEDIATE TRANSACTION + agent run record creation	Credits deducted atomically on agent execution
Day 4	R2 storage integration: save agent output + result display page	User can view and download agent output
Day 5	YouTube Trend Research agent: YouTube Data API + OpenAI analysis	User gets trending topic recommendations for their niche
Day 6-7	Prompt quality iteration: test with real Shorts topics, refine prompt templates, A/B test outputs	Script quality reaches 'good enough to use' bar

[NOTE] Days 6-7 (prompt quality iteration) are the most important days in the entire roadmap. The technical integration (API call, credit deduction, storage) is straightforward. The product value lives or dies on whether the generated scripts are actually useful to a Shorts creator. Allocate full focus here. Test with at least 10 real topic categories and get feedback if possible.

Week 3: Polish & Launch

Goal: Beta launch with real payments (NICE Payments live mode).

Day	Task	Deliverable
Day 1	Landing page: hero section, pricing cards, Kakao login CTA, feature highlights	Public-facing landing page live
Day 2	Dashboard UI polish: loading states, error messages, responsive design, toast notifications	Professional-looking dashboard
Day 3	Switch from PortOne sandbox to live mode (requires NICE PG review completion)	Real payments work
Day 4	Security audit: amount tampering, XSS check, CSRF protection, rate limiting, API key exposure check	Security checklist passed
Day 5	Beta launch: deploy to production, announce in creator communities, collect feedback	Live product accepting payments
Day 6-7	Monitor: error logs, payment success rates, agent output quality, user feedback	Post-launch stability confirmed

5. Credit System Design

5.1 Credit Packages (KRW, VAT Included)

Package	Price (KRW)	Credits	Bonus	Per Credit	Target User
Free (signup)	0	100	-	-	Trial users
Starter	9,900	1,100	+100	9.0 KRW	Casual creators (1-2 Shorts/week)
Pro	29,900	3,500	+500	8.5 KRW	Regular creators (daily Shorts)
Business	99,000	12,000	+2,000	8.3 KRW	Power users / agencies

5.2 Credit Cost per Agent

Agent	Credits/Run	Approx. KRW Cost	OpenAI API Cost (est.)	Margin
Shorts Script Writer (GPT-4o)	20	200 KRW	~50-80 KRW (1K-2K tokens)	60-75%
YouTube Trend Analyzer	10	100 KRW	~30-50 KRW (YouTube API free + GPT analysis)	50-70%

With Starter package (9,900 KRW = 1,100 credits): a user can generate approximately 55 Shorts scripts or 110 trend analyses per month. For a daily Shorts creator, Pro package (3,500 credits) covers roughly 1 month of daily usage.

6. Risk Assessment

Risk	Severity	Probability	Mitigation
NICE Payments PG review delayed beyond 5 days	High	Low	Develop and test entirely on PortOne sandbox. Live switch is a config change (API key + PG code).
Script quality insufficient for paying users	Critical	Medium	Dedicate Week 2 Days 6-7 entirely to prompt iteration. Establish minimum quality bar before launch.
D1 write throughput bottleneck	Medium	Very Low	500 DAU = 0.03 TPS. D1 handles 50-100 TPS. 1000x headroom. Monitor via Cloudflare Analytics.
OpenAI API price increase	Medium	Low	Credit costs are configurable server-side. Adjust package pricing or switch to Claude API as fallback.
Amount tampering attack	Critical	Low	Server-side cross-verification (Section 2.2 Steps 5-6). Pre-created PAYMENT_ORDER prevents this entirely.
Concurrent credit deduction race condition	High	Low	D1 BEGIN IMMEDIATE TRANSACTION serializes writes. Tested under concurrent load in Week 1 Day 7.
Low initial adoption / no paying users	High	Medium	100 free credits lower the trial barrier. Target 3 specific Shorts creator communities for launch.
Kakao OAuth rejection / review delay	Medium	Low	Clerk handles OAuth complexity. Google OAuth as fallback authentication method.

7. Infrastructure Cost Analysis

7.1 Monthly Cost at 500 DAU

Service	Free Tier	Estimated Usage (500 DAU)	Monthly Cost
Cloudflare Workers	100K requests/day free	~15K requests/day	$0 (within free tier)
Cloudflare D1	5M reads, 100K writes/day free	~10K reads, ~2.5K writes/day	$0 (within free tier)
Cloudflare R2	10GB storage, 10M reads free	~5GB storage, ~50K reads/month	$0 (within free tier)
OpenAI API (GPT-4o)	N/A	~2,500 runs/day = ~5M tokens/day	$50-150 (variable)
Clerk Auth	10K MAU free	~500 MAU	$0 (within free tier)
Vercel (Next.js)	Hobby plan free	~50K page views/month	$0 (hobby plan)
Domain (openclaw.co.kr)	N/A	N/A	~15,000 KRW/year
PortOne	Free platform	N/A	$0 (no platform fee)
NICE Payments PG fees	N/A	2.5-3.5% per transaction	Variable (deducted from revenue)

Total fixed infrastructure cost: approximately $0-5/month at 500 DAU.

The only significant variable cost is the OpenAI API, which scales linearly with usage and is covered by the 60-75% margin built into credit pricing.

7.2 Break-Even Analysis

- Minimum viable revenue: OpenAI API cost must be covered by credit revenue.
- At 60% margin: every 9,900 KRW Starter purchase generates ~6,000 KRW gross margin.
- Break-even: approximately 20-30 Starter-equivalent purchases per month.
- This means 20-30 paying users (not total users) is the break-even point.

8. Post-MVP Expansion Path

The MVP is designed with clear expansion points. Each addition is triggered by specific metrics, not arbitrary timelines.

Expansion	Trigger Metric	Estimated Effort	Architecture Impact
Virtual Account payment	User requests > 10/month	1-2 days (webhook handler)	Add webhook endpoint + WEBHOOK_EVENTS table
Thumbnail Generation agent	Script agent NPS > 7	2-3 days (DALL-E integration)	Add image processing, R2 storage increase
Agent Marketplace UI	5+ agents available	3-5 days	New frontend pages, search index
Subscription billing	Monthly retention > 60%	3-5 days	PortOne billing key, subscription lifecycle
Durable Objects migration	DAU > 5,000	1-2 weeks	Gradual migration, D1 remains as source of truth
Automated refund system	CS refund requests > 20/month	2-3 days	Admin UI + PortOne cancel API integration
SEO/Tag agents	Revenue > 2M KRW/month	1-2 days each	Additional prompt templates
Cross-platform posting	YouTube API approval obtained	3-5 days	OAuth scope expansion, external API integration

9. Security Checklist (Pre-Launch)

All items must pass before beta launch in Week 3 Day 4.

Category	Check Item	Test Method	Status
Payment	Amount tampering (modify SDK amount in devtools)	Manual: change amount, verify server rejects	Pending
Payment	Replay attack (reuse imp_uid)	Manual: submit same imp_uid twice, verify idempotency	Pending
Payment	Invalid merchant_uid	Manual: send forged merchant_uid, verify rejection	Pending
Credit	Concurrent credit deduction (race condition)	Script: 10 simultaneous deduction requests	Pending
Credit	Negative balance prevention	Manual: attempt agent run with 0 credits	Pending
Auth	Unauthorized API access (no session)	Manual: call API endpoints without auth header	Pending
Auth	Cross-user data access	Manual: attempt to read another user's runs/credits	Pending
Input	XSS in agent configuration fields	Manual: inject script tags in config inputs	Pending
Input	SQL injection in search/filter	Manual: inject SQL in query parameters	Pending
API	Rate limiting (brute force)	Script: 100 rapid requests, verify rate limit kicks in	Pending
Secrets	API keys not exposed in frontend bundle	Inspect: check browser network tab, page source	Pending
Secrets	Environment variables not in git repository	Inspect: check .env in .gitignore, scan git history	Pending

10. Conclusion

This plan does not sacrifice technical quality. It selects appropriate technology for the current business stage.

Security:

PortOne cross-verification with pre-created PAYMENT_ORDER eliminates payment fraud risk. This is the same security model used by production Korean e-commerce platforms. The financial risk is effectively zero.

Performance:

Removing unnecessary abstraction layers (Sandbox, Durable Objects) actually improves response latency. Fewer moving parts means fewer failure points.

Scalability:

Everything runs within the Cloudflare ecosystem. When traffic growth demands it, upgrading to Durable Objects or Workers for Platforms is a migration within the same platform -- not a re-architecture. The expansion path is pre-mapped (Section 8).

The most reliable way to validate a product is not to build the perfect system, but to launch quickly and observe whether real users open their wallets. This plan enables that observation within 4 weeks.

Prepared by	Jo Heejin (Lead / Responsible Person)
Date	2026-02-03
Signature	