Phork Studio

AI Production Studio — Create, Fork, Remix

Phase 1 MVP Engineering Report

February 24, 2026

INTERNAL ALPHA — MILESTONE M0–M7 COMPLETE

1. Executive Summary

This report documents the completion of the **Phase 1 MVP** for Phork, a closed-loop AI production studio. The MVP implements all core systems required by the engineering handoff specification, including the studio UI, backend API, job processing pipeline, asset provenance system, credit ledger, and project forking.

73	4	11	7
SOURCE FILES	PACKAGES	DB TABLES	MILESTONES DONE

Key Decisions Made

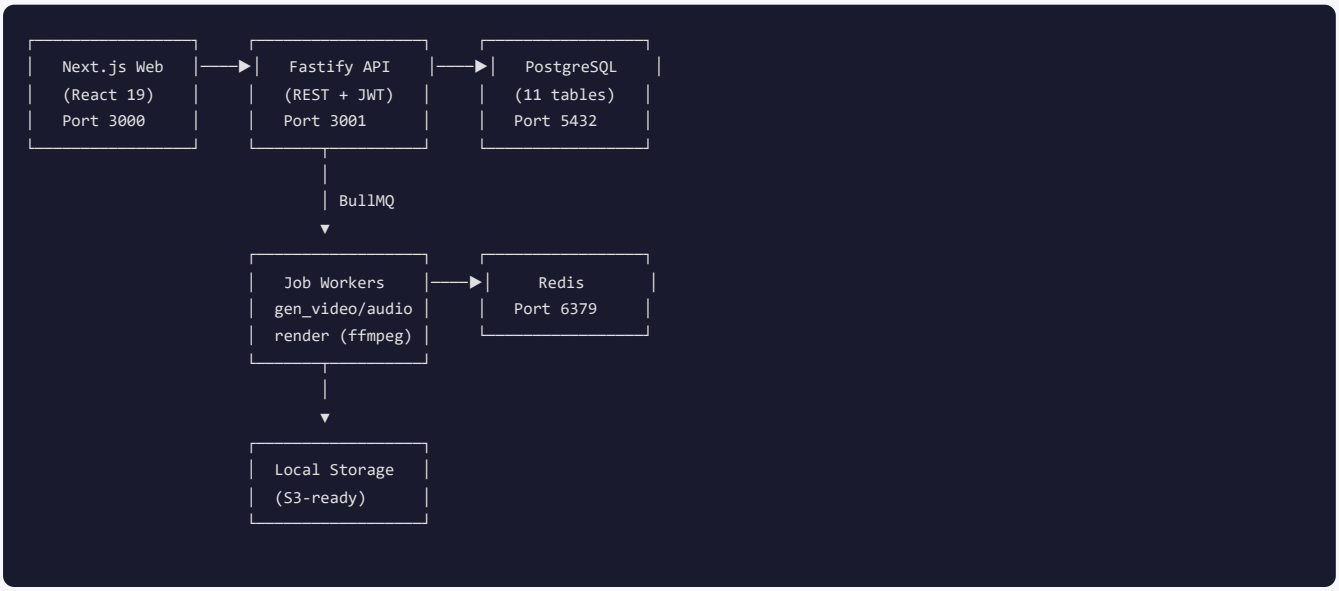
Decision	Choice	Rationale
Backend Language	TypeScript (Node.js)	Unified stack with frontend; strong typing across packages
API Framework	Fastify 5	High performance, plugin system, built-in validation
Frontend	Next.js 15 + React 19	App Router, server components, built-in routing
Database	PostgreSQL + Drizzle ORM	Relational fits DAG/ledger; Drizzle is type-safe and lightweight
Queue	BullMQ (Redis)	Mature, supports job retry/dedup, good Node.js integration
Monorepo	pnpm + Turborepo	Efficient workspace management, parallel builds, caching
Snapshot Strategy	JSON per commit	Simple, self-contained; recommended by spec for Phase 1
Fork Model	New project + parent ref	Clean separation; as recommended by spec
Generation Providers	Stub (FFmpeg-based)	Architecturally ready for real APIs; no external dependency for alpha

2. Architecture Overview

Monorepo Structure

```
phork/
├── apps/
│   ├── web/          # Next.js 15 frontend (port 3000)
│   └── api/          # Fastify 5 API + BullMQ workers (port 3001)
├── packages/
│   ├── db/           # Drizzle ORM schema + migrations (PostgreSQL)
│   └── shared/       # TypeScript types, enums, interfaces
├── docker-compose.yml # PostgreSQL 16 + Redis 7
├── turbo.json        # Turborepo task pipeline
└── pnpm-workspace.yaml
```

System Architecture



3. Database Schema

11 PostgreSQL tables organized across 6 domains. Full schema managed via Drizzle ORM with generated migrations.

Users & Workspaces

Table	Key Columns
`users`	id, email (unique), password_hash, display_name
`workspaces`	id, name, created_by
`workspace_members`	workspace_id, user_id, role

Credits

Table	Key Columns
`credit_accounts`	workspace_id (PK), balance
`credit_ledger`	id, workspace_id, job_id, delta, reason

Projects & Branching (DAG)

Table	Key Columns
`projects`	id, workspace_id, name, parent_project_id, forked_from_commit_id
`commits`	id, project_id, parent_commit_id, snapshot (JSONB)
`project_heads`	project_id (PK), head_commit_id

Assets & Safety

Table	Key Columns
`assets`	id, type, storage_url, mint_receipt_sig, provenance (JSONB)
`jobs`	id, type, status, idempotency_key (unique)
`safety_events`	id, job_id, category, action

4. API Surface

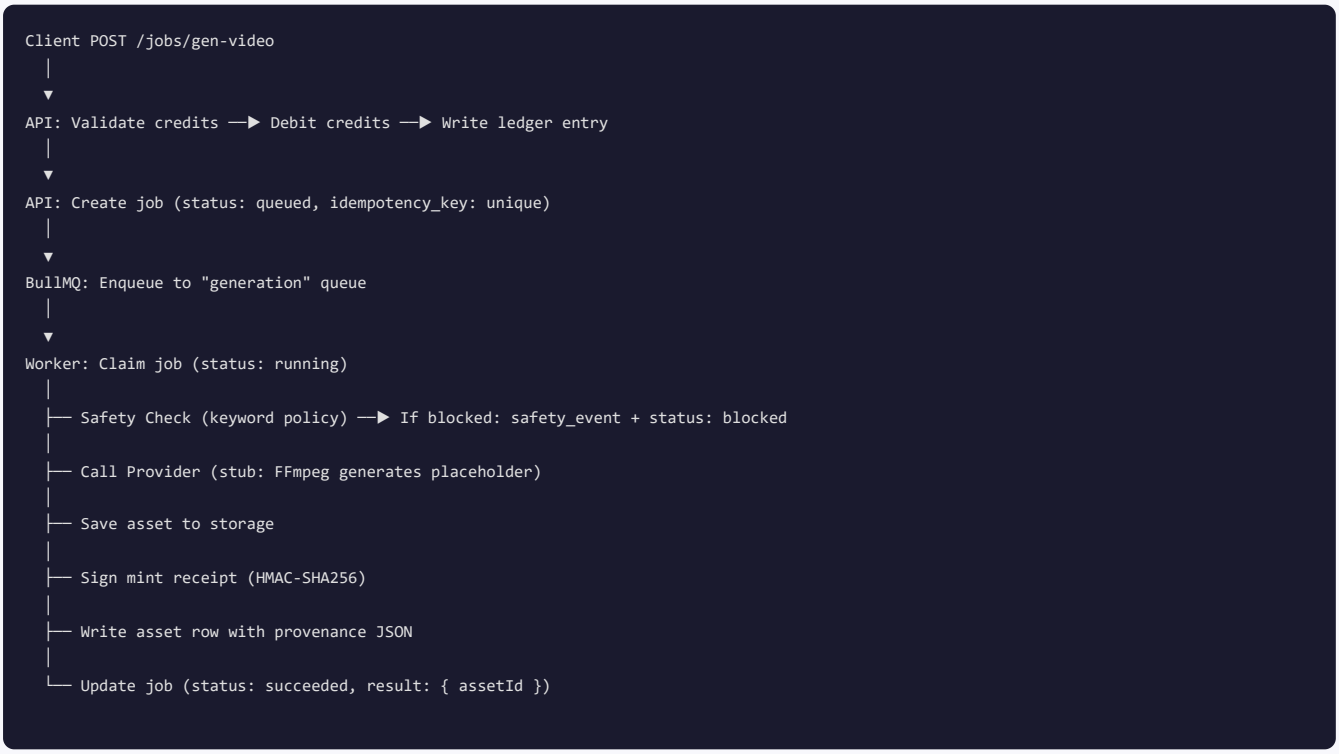
RESTful API with JWT authentication. All endpoints except auth require a valid Bearer token.

Method	Endpoint	Description	Auth
POST	`/auth/register`	Register + create workspace + 1000 credits	No
POST	`/auth/login`	Login, returns JWT	No
GET	`/auth/me`	Current user + workspaces	Yes
POST	`/projects`	Create project + initial commit	Yes
GET	`/projects/:id`	Project + head commit	Yes
GET	`/projects?workspaceId=`	List workspace projects	Yes
POST	`/projects/:id/commits`	Create commit (validates mint receipts)	Yes
GET	`/projects/:id/commits`	List commits	Yes
POST	`/projects/:id/fork`	Fork project from commit	Yes
POST	`/jobs/gen-video`	Queue video generation job	Yes
POST	`/jobs/gen-audio`	Queue TTS audio job	Yes
POST	`/jobs/render`	Queue render job (commit → mp4)	Yes
GET	`/jobs/:id`	Job status + result	Yes
GET	`/credits/balance`	Workspace credit balance	Yes
GET	`/credits/ledger`	Immutable ledger entries	Yes
GET	`/assets/:id`	Asset metadata + provenance	Yes
GET	`/assets/:id/file`	Stream asset binary	No*

* Asset file streaming is unauthenticated so the HTML video player can fetch directly.

5. Job Pipeline & Generation

Job Execution Flow



Job Types & Costs

Job Type	Credits	Phase 1 Provider	Output
gen_video	25	Stub (FFmpeg color + text overlay)	MP4 1280x720
gen_audio	5	Stub (FFmpeg silent MP3)	MP3
gen_image	10	Stub (FFmpeg color PNG)	PNG 1280x720
render	15	FFmpeg concat pipeline	MP4 (assembled)

Safety System

Phase 1 implements keyword-based content policy checking:

- **Blocked categories:** Deepfake/face-swap attempts, extreme violence, CSAM
- **Warning categories:** Weapon references, violence references (logged, not blocked)
- Safety events are stored in `safety_events` table with category, entity, and action
- Architecture supports future ML-based classification upgrade

Idempotency

Every job requires a unique `idempotency_key`. If a duplicate key is submitted, the API returns the existing job without creating a new one or debiting credits again. This prevents double-charging on retries.

6. Closed Ecosystem Enforcement

The platform enforces that **no external assets can be used in projects**. This is the foundational constraint of Phork.

Enforcement Mechanisms

Layer	Mechanism	Implementation
Asset Creation	Mint Receipt Signing	HMAC-SHA256 signature generated server-side when an asset is created by a job. Signature = HMAC(assetId:jobId, secret).
Commit Validation	Asset Reference Check	When creating a commit, the API validates every asset ID in the timeline snapshot exists in the DB and has a valid `mint_receipt_sig`.
No Upload API	No upload endpoint	There is no API endpoint for uploading external files. Assets can only be created by job workers.

Attempting to reference a non-existent or un-minted asset ID in a commit returns HTTP 400 with the message: *"Asset [id] not found or missing mint receipt. Only platform-generated assets are allowed."*

7. Project Forking (DAG)

Forking creates a new project that preserves the full commit history up to the fork point.

Fork Mechanics

- Fork creates a **new project** with `parent_project_id` and `forked_from_commit_id`
- Commit chain is walked backwards from fork point and re-created with new IDs
- Asset references are preserved (shared, not duplicated)
- New commits can diverge independently

8. Studio UI

8.1 Login / Registration

Email/password auth with toggle between sign-in and registration. Registration auto-creates a workspace and grants 1000 starter credits.

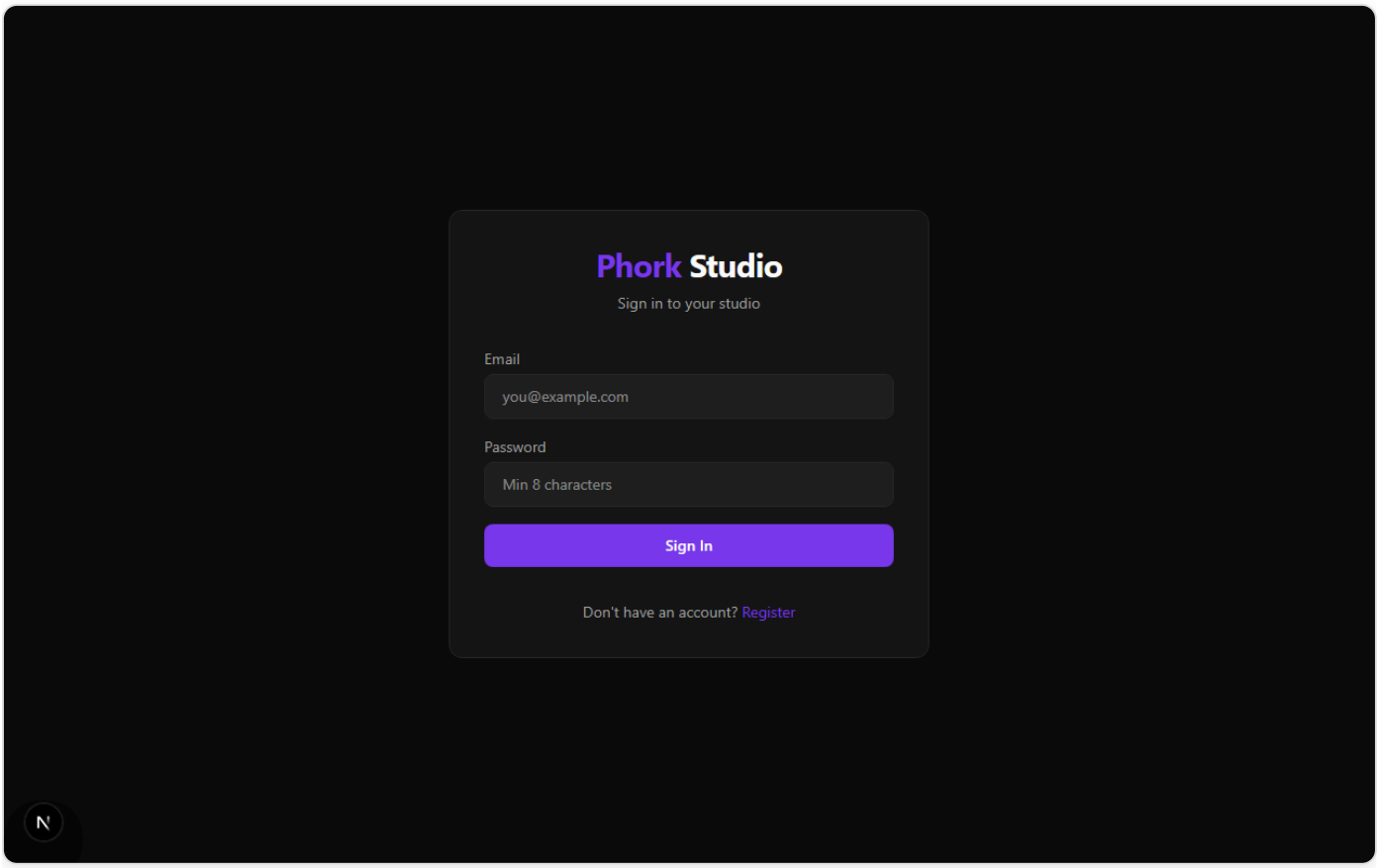


Figure 1: Login page with dark theme and purple accent branding

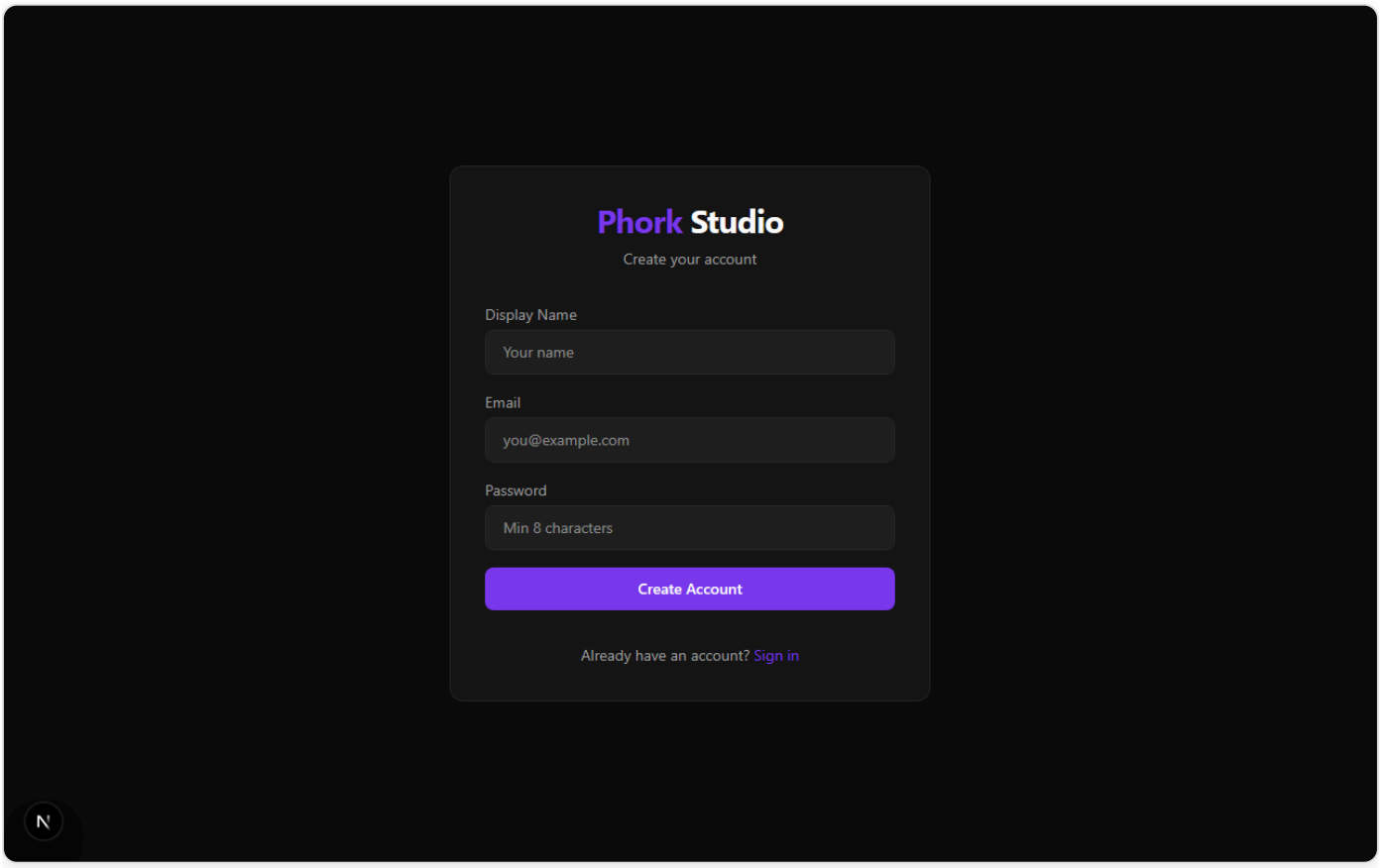


Figure 2: Registration form with Display Name, Email, and Password fields

8.2 Project Dashboard

Lists all projects in the workspace. Shows project name, description, fork badge, and creation date. "+ New Project" button opens inline creation form.

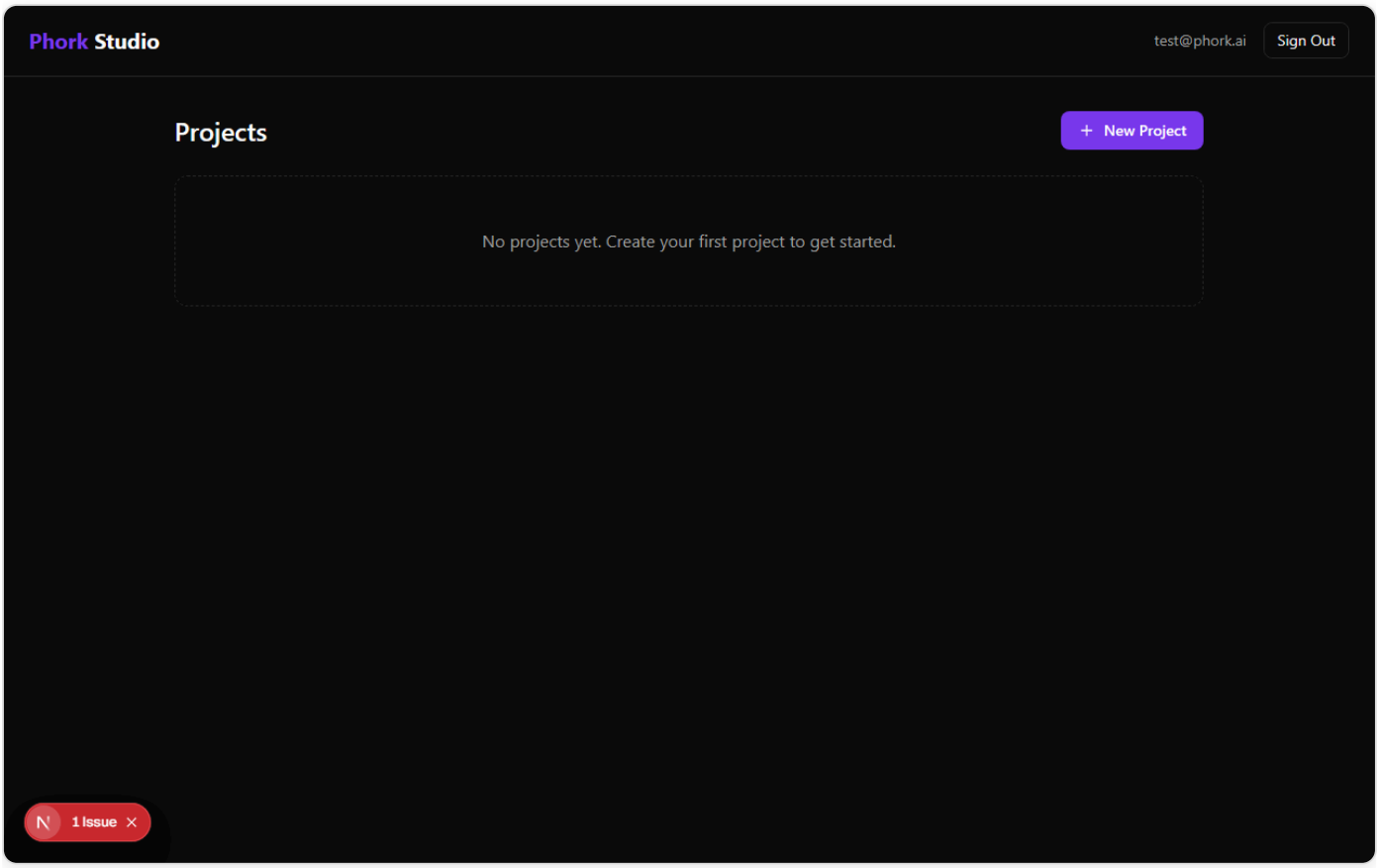


Figure 3: Studio dashboard with project grid (empty state shown)

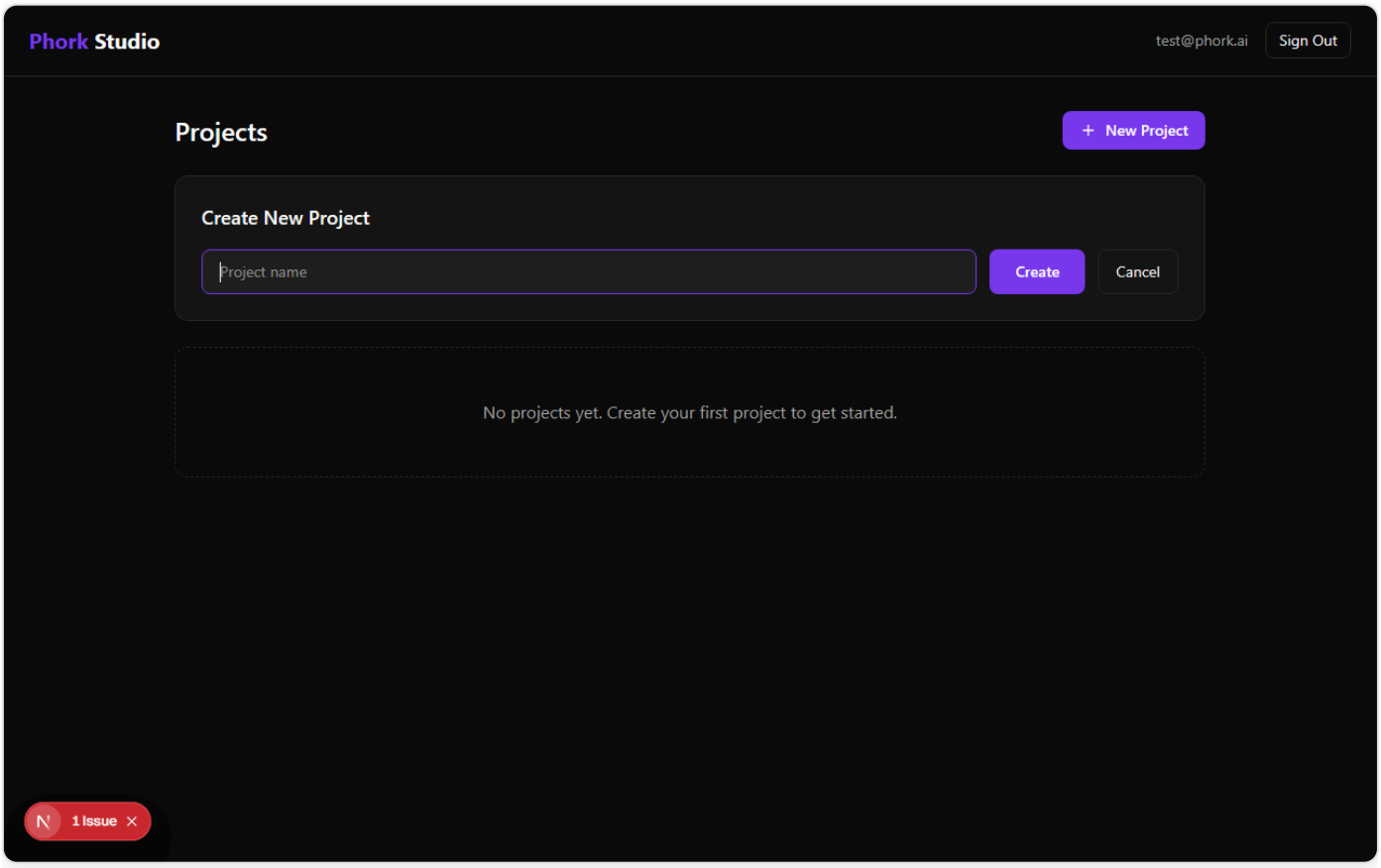


Figure 4: New Project inline creation form

8.3 Project Editor (3-Panel Layout)

The core studio interface with three panels:

- **Left:** Shot list with status indicators, add/remove/reorder
- **Center:** Preview player (video playback after render, or shot count placeholder)
- **Right:** Shot editor with visual prompt, audio TTS text, duration, subtitle

Top bar includes: back navigation, project name, credits counter, Provenance, Fork, Save, and Render buttons.

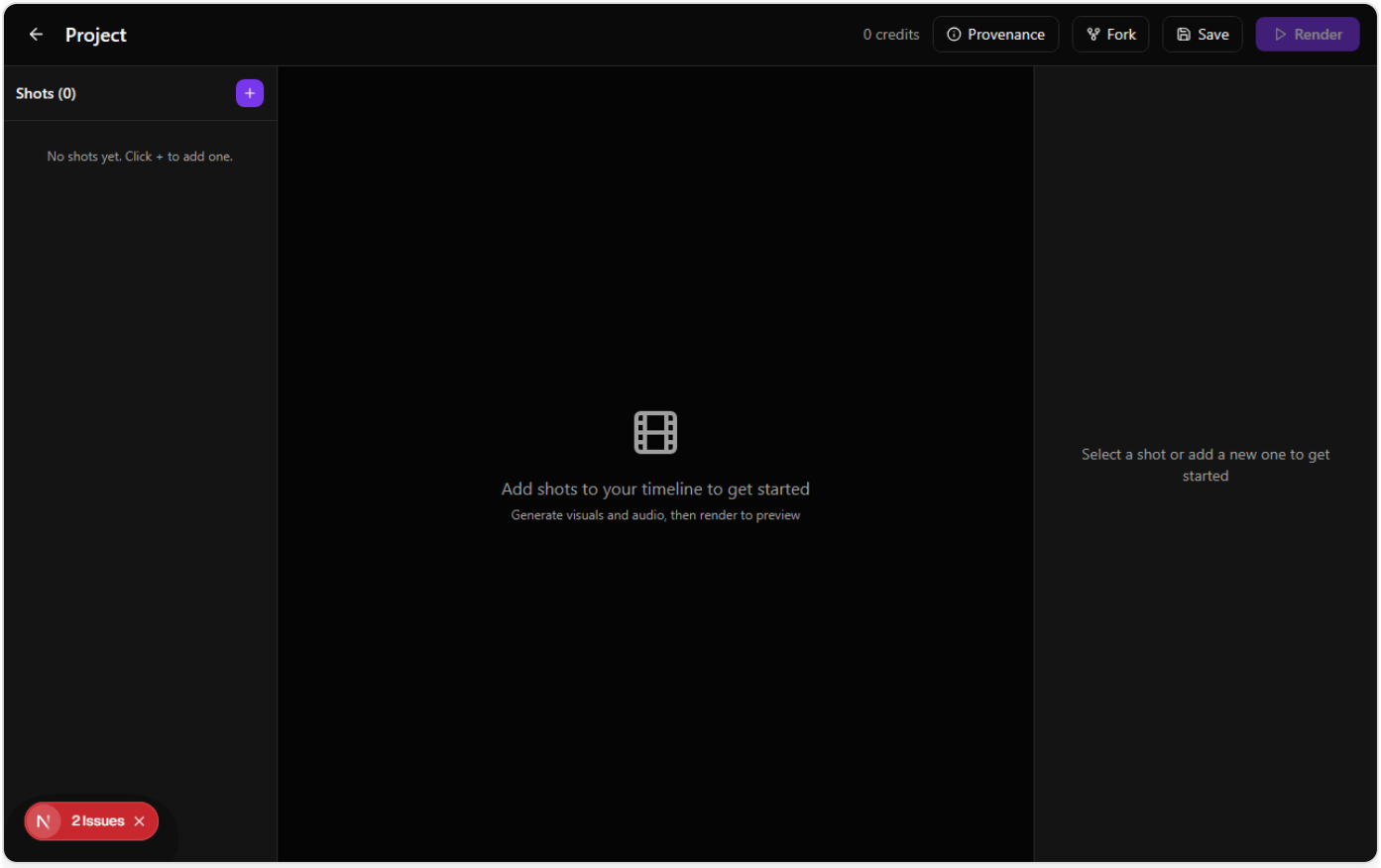


Figure 5: Project editor empty state — "Add shots to your timeline to get started"

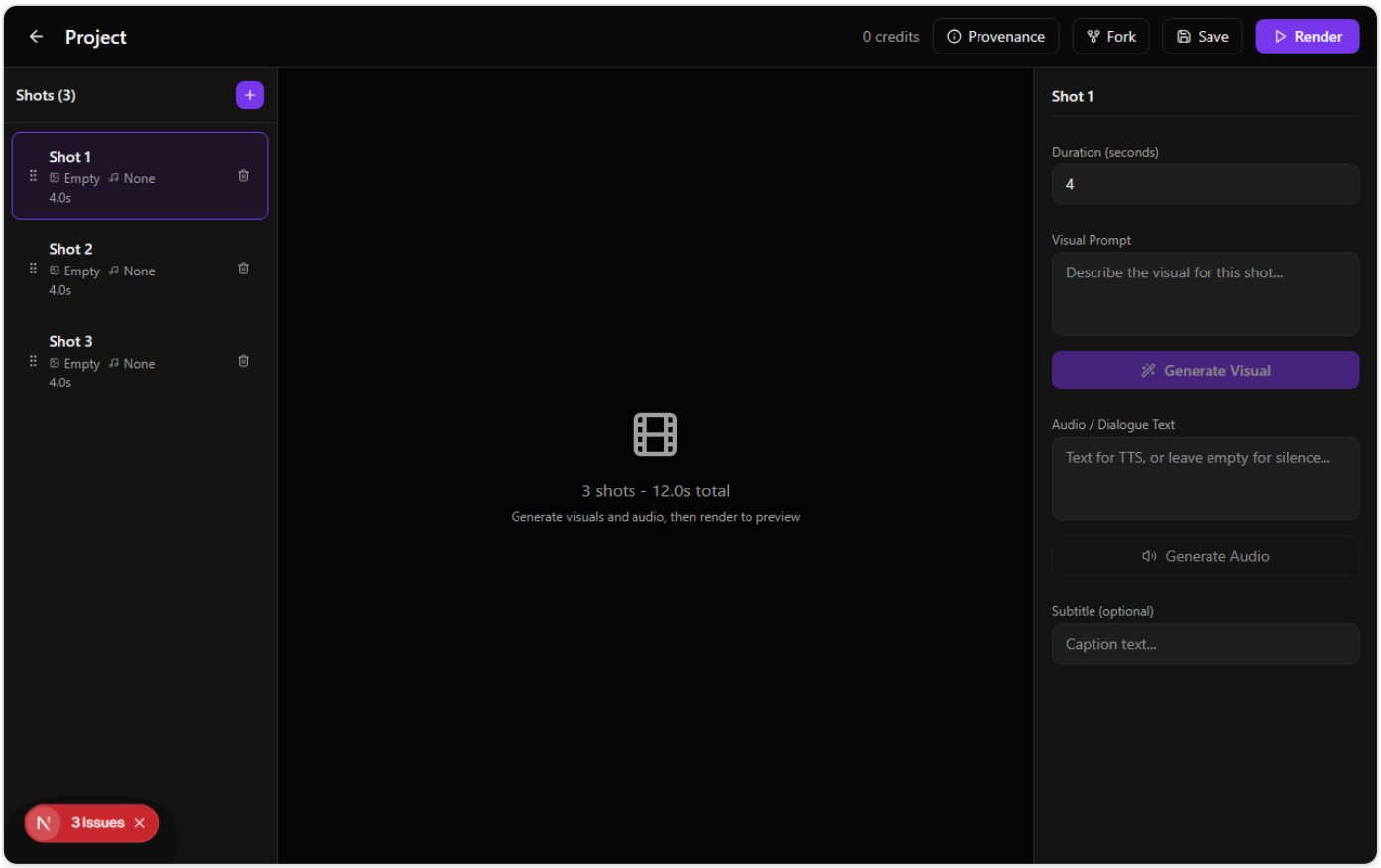


Figure 6: Project editor with 3 shots added, showing shot list, preview summary, and shot editor panel

9. Acceptance Criteria Status

#	Criterion	Status	Evidence
1	**Closed ecosystem enforced** Cannot attach external files. Unknown asset IDs rejected.	COMPLETE	Mint receipt validation on commit creation. No upload endpoint exists. Integration test verifies rejection of fake asset IDs.
2	**Creation works** User can generate 3+ shots + audio and render 30–120s mp4.	COMPLETE	Full pipeline: gen_video + gen_audio workers → asset storage → FFmpeg render → mp4 output. UI supports generate/render flow.
3	**Forking works** Forked project preserves history. Fork can diverge and render.	COMPLETE	Fork API copies commit chain, sets parent_project_id. Integration test verifies fork + divergent commits.
4	**Provenance is complete** Every asset has provenance_json. Render references commit + shot assets.	COMPLETE	ProvenanceManifest written on every asset creation. Render provenance includes upstream asset IDs. UI provenance panel displays metadata.
5	**Costs/credits tracked** Immutable ledger. Project burn computable.	COMPLETE	credit_accounts + credit_ledger tables. Every job debits with ledger entry. Balance + ledger API endpoints.
6	**Jobs are reliable** Retries don't double-charge. Idempotency enforced.	COMPLETE	idempotency_key (unique constraint) on jobs table. Duplicate submission returns existing job. Integration test verifies single debit.

All 6 acceptance criteria are met.

10. Milestone Completion

Milestone	Scope	Status
M0	Repo scaffolding, DB migrations, storage utils, queue setup	COMPLETE
M1	Auth, workspaces, projects, commits, credit accounts + ledger	COMPLETE
M2	Asset table, storage writes, mint receipt signing, provenance	COMPLETE
M3	Job table, queue processing, idempotency, charging logic	COMPLETE
M4	gen_video, gen_audio, gen_image stubs + safety checks	COMPLETE
M5	FFmpeg render pipeline (concat shots → mp4)	COMPLETE
M6	Studio UI: login, project list, timeline editor, fork dialog, provenance panel	COMPLETE
M7	Seed script + 8-test integration suite	COMPLETE

11. QA Harness

Seed Script

`apps/api/src/scripts/seed.ts` — Creates test data for manual QA:

- Test user: `test@phork.ai` / `testpass123`
- Workspace with 5000 credits
- "Demo: Space Adventure" project with 3 video assets + 2 commits
- Proper mint receipts, provenance records, and ledger entries

Integration Test Suite

`apps/api/src/scripts/test-flows.ts` — 8 automated tests covering:

#	Test	What It Verifies
1	Health Check	/health returns 200 with status: ok
2	Registration & Auth	Register, login, /auth/me all work correctly
3	Closed Ecosystem	Commit with fake asset ID is rejected (400)
4	Project & Commits	Create project, create commits, list commits
5	Credits System	Balance starts at 1000, ledger endpoint works
6	Job Idempotency	Duplicate job returns same ID, credits debited once
7	Forking	Fork preserves history, divergent commits work
8	Job List & Status	List jobs by project, get individual job

12. How to Run

Prerequisites

- Node.js 20+ (tested with v24.12.0)
- pnpm (installed globally)
- Docker (for PostgreSQL + Redis)
- FFmpeg (on PATH, for generation stubs and rendering)

Startup Commands

```
# 1. Install dependencies
pnpm install

# 2. Start infrastructure
docker-compose up -d          # PostgreSQL 16 + Redis 7

# 3. Push database schema
pnpm db:push

# 4. (Optional) Seed test data
cd apps/api && npx tsx src/scripts/seed.ts

# 5. Start API server (Terminal 1)
pnpm --filter @phork/api dev

# 6. Start job workers (Terminal 2)
pnpm --filter @phork/api dev:worker

# 7. Start frontend (Terminal 3)
pnpm --filter @phork/web dev

# 8. Run integration tests
cd apps/api && npx tsx src/scripts/test-flows.ts
```

13. Next Steps (Post-Phase 1)

The following are explicitly **out of scope for Phase 1** but the data model and architecture have been designed to support them:

Feature	Design Readiness
Real AI providers (Replicate, ElevenLabs, OpenAI)	Swap stub functions in `workers/generation.ts`; provenance schema already supports provider/model/version
Public publishing on phorked.ai	`visibility` field exists on projects (currently only 'private')
Fork licensing enforcement	`fork_license` field stored (no_forks, forks_nc, forks_revshare, sharealike)
HLS streaming	Render pipeline produces mp4; HLS packaging is an additional post-process step
C2PA/Content Credentials	Provenance JSON captures all required source data; C2PA signing is additive
Payment processing & payouts	Credit system is in place; payment integration would extend credit_accounts
Frame-accurate fork from timestamp	Current: fork at shot boundary. Frame-level requires timeline format extension

---