Shard Frontier — Wave-2 Submission Notes

Project Scope:

Shard Frontier is a BDAG-powered Web3 game ecosystem with the first end to end, fully functional technical loop built entirely on the BlockDAG Awakening Testnet.

• Players mine, forge, and refine NFT Shards
• NFT Shards contain on-chain traits, rarity classes, and family types
• Gameplay connects to X1 mining boosts through the backend
• Bonus tokens can be spent or burned via smart contracts
• The system is fully deployed on BlockDAG Awakening Testnet

Problem:

BlockDAG has millions of X1 miners, a growing developer base, and NFTs arriving — but no unified loop ties them together. Miners mine in isolation. NFTs exist in isolation. dApps don't yet leverage X1 mining sessions.

Solution:

We built the first ecosystem loop where:
• X1 mining sessions (simulated endpoints today) trigger n-game energy boosts. - A Railway backend processes gameplay → boosts → mining rewards.
• NFT Shards can be forged, refined, combined, and traded.
• All components run on BlockDAG Awakening Testnet with working smart contracts and RPC calls.

Highlights:

• Mobile-first 9:16 UI, fast, lightweight.
• WalletConnect integration on testnet (Chain ID 1043).
• Shard Inventory with real IPFS metadata and images.
• Validated burn engine direction (ERC-20 + BurnVault).
• Railway backend with clean latency and CI/CD.
• Minimal on-chain tx in demo path; off-chain gameplay for scale.

Scalability:

• Backend runs stateless on Railway; auto-scales horizontally.
• Postgres connection pooling; caching and rate-limit plan documented.
• Off-chain gameplay with on-chain settlement (forge/mint) → cost-efficient and scalable.
• API endpoints designed to be cache-friendly; contract calls minimized.

Ecosystem Blueprint (reusable by BlockDAG projects):

• Gaming integrations with X1.
• Mining-boost loops and burn-to-boost mechanics.
• NFT-driven reward mechanisms and crafting economies.
• Real-time analytics and expansion via Telegram/WhatsApp mini-apps. -

Composable shard system across multiple future games.

Wave 2 brings the first full end-to-end integration of:

• Game UI
• NFT system
• BDAG testnet
• Backend
• Burn mechanics
• X1 simulator

UI Visual Demo Link designed to engage and click on prompts (mobile only):

https://shard-frontier-production.up.railway.app
Password: biggames01

Shard Frontier expanded significantly since Wave-1, completing the first full technical loop on the BlockDAG Awakening Testnet: NFTs → backend → boosts → burn engine → sessions.

Smart Contracts (NEW in Wave-2):

• ShardNFT (ERC-721) live at 0x0F2F6F22Aa68b11295e2FbEb07416c8910481c11.
• Payable mint (5 BDAG), IPFS-based BaseTokenURI, Trait Matrix v1.1 support.
• Gameplay collect ore → combine soulbound shards → wallet connect → combine ECR-721 shards (3 grades) & mint → view in inventory
• BonusBDAGToken (ERC-20) + BurnVault deployed as a prototype burn engine.
• Mint → approve → transfer → burn workflows verified on Awakening Testnet.

Full working NFT Metadata & IPFS (NEW):

• Rebuilt correct IPFS structure: /images/{n}.png + /json/{id} (extensionless).
• Inventory screen now resolves tokenURI → IPFS → traits/images successfully.

Backend (NEW) — Railway + Postgres:

• Live backend with health endpoint, CI/CD from GitHub, Awakening RPC signer.
• Implemented X1 simulation endpoints:
• GET /x1/summary
• GET /x1/wallet/:address/boosts
• POST /x1/session/start
• POST /x1/session/complete
• Postgres attached; schemas drafted for sessions, boosts, burn_history.

Backend Health:

https://radiant-fascination-production.up.railway.app/health

On the frontend

Frontend & UI (Jaime's Wave-2 Additions)--In House Minting:

• Complete mobile-first 9:16 UI with working prototype screens (Start, Home, Dashboard, Forge, Refine, NFT Forge, Garage, Hover Bay, Medals, Profile, Map, Inventory).
• Invisible hotspot navigation aligned to concept art.
• Inventory screen now fully functional: wallet integration, tokenURIs, IPFS images, metadata traits.
• WalletConnect + Chain 1043 testnet support.
• On-chain minting is intentionally script-only for stability during judging, later to become live minting button for any user to connect wallet, generate chance factor 60/30/10% Raw, Rare, Legendary shards from separate CID json and png files & sequential master token ID in contract, mint to user wallet.

Video demos of UI interface leading to inventory page displaying ERC 721 NFTs including metadata & images:

https://youtube.com/shorts/LK6ZYyigfY0?feature=shared

Highlights / Reviewer Path:

• Connect wallet → /inventory → see live NFTs and metadata.
• Backend and X1 endpoints accessible publicly.
• Burn engine validated (ERC-20 + BurnVault).
• Full Awakening Testnet compatibility achieved.

What This Wave Delivers:

• First end-to-end ecosystem loop on BDAG:
NFTs ↔ backend ↔ mining boosts ↔ token burning
• Blueprint reusable by BDAG community apps and future games.

On the backend

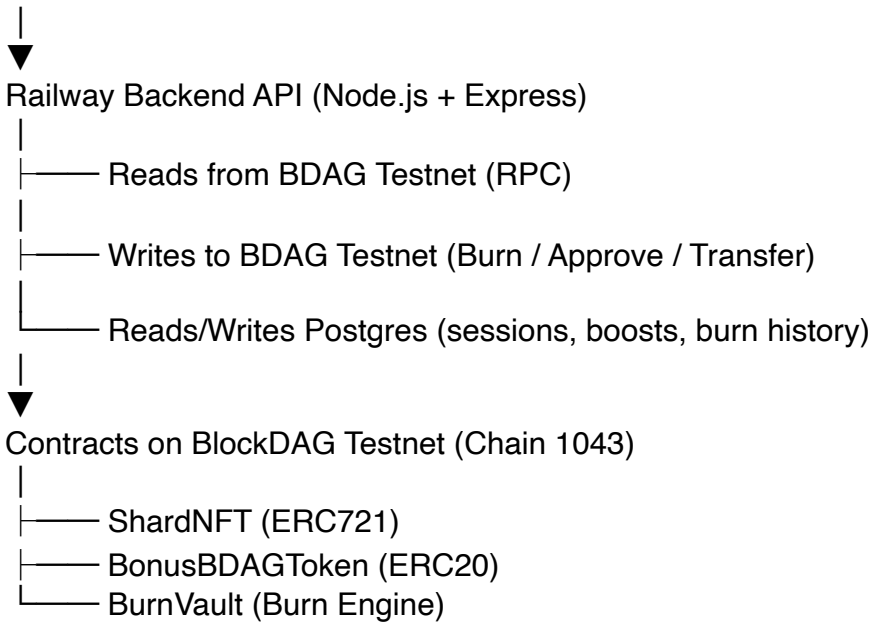
This section explains the full technical architecture connecting:

• Frontend (UI + NFT system)
• Backend (Railway Node + Postgres)
• BlockDAG Awakening Testnet
• X1 Mining Simulation API
• Burn Engine (ERC-20 + BurnVault)

It describes the flow of data from the moment a player interacts with the UI → through the backend → on-chain transactions → returned to the game.

Frontend (React + WalletConnect)

WalletConnect)



X1 Gameplay Loop – Technical Flow:

The X1 Loop simulates how Shard Frontier gameplay actions will generate mining boosts inside the BlockDAG ecosystem.

Step 1 — Player Opens Game UI
• UI served from Railway
• Player connects MetaMask (configured for Chain 1043)

Step 2 — UI Requests Player Boost Data

UI calls

GET /x1/wallet/:address/boosts

Backend computes:
• ore totals
• parts
• boosts axis scores
• caps
• timestamps

Returns JSON → displayed in the UI.

Session Start (Mining Simulation)

When gameplay begins (e.g., mining run, mission, crafting):

Frontend calls:

POST /x1/session/start
Body:

{
"address": "0xPlayer",
"client_ts": <timestamp>,
"session_id": <uuid>
}

Backend:
• Creates new session in Postgres
• Issues nonce for verification
• Prepares the boost logic for session/complete

Session Completion (Reward Processing)

After user finishes a gameplay session:

Frontend calls:

POST /x1/session/complete
Body:
{
"address": "0xPlayer",
"session_id": <uuid>,
"ore_delta": { "dust": 60, "alloy": 20 }
}

POST /x1/session/complete
Body:
{
"address": "0xPlayer",
"session_id": <uuid>,
"ore_delta": { "dust": 60, "alloy": 20 }
}

POST /x1/session/complete
Body:
{
"address": "0xPlayer",
"session_id": <uuid>,
"ore_delta": { "dust": 60, "alloy": 20 }
}

Backend:
• Updates Postgres tables
• Recalculates boosts
• Returns reward summary
• UI updates boost bars or counters

———

Optional (Future) — Burn for Boost:

Players will be able to burn bonus tokens to gain gameplay boosts.

Flow:
1. UI → Backend: “burn 10 tokens”
2. Backend signs a burn transaction using its private dev wallet
3. BurnVault destroys tokens
4. Backend records burn in Postgres
5. Boosts increase for the player

This creates a spend → burn → boost gameplay mechanic.

Burn Engine Architecture (Wave-2)

Contracts Deployed:

Shards NFT (ERC-721) ---> 0x0F2F6F22Aa68b11295e2FbEb07416c8910481c11

BonusBDAGToken (ERC20) --->
0x58a97C15D759CCC54a66b2D2d5fC26A1F91103

Burn Vault (Burn Engine) ---> 0x058a97c15D759cCC54a6b02b02d5fc26A1F91103

Burn Flow:

approve(BurnVault, amount)
↓
burnVault.burn(amount)
↓
ERC-20 tokens sent to address(0)
↓
Backend logs burn event
↓
Frontend boosts/crafting actions update (Wave-3)

Scripts Built in Wave-2:

• deployBonusToken.js
• deployBurnVault.js
• approveBonusToken.js
• mintToMetaMask.js
• transferBonusToken.js
• burnBonusToken.js
• debugBurnSetup.js
• checkBonusState.js

How the Frontend Connects to X1 & Blockchain

Frontend Tech Stack:

• React (Vite)
• WalletConnect
• Ethers v6
• React Router
• Custom UI (Jaime's 9:16 screens with hotspots)

Frontend → Backend

• Fetch boosts
• Start/complete sessions
• Trigger burn operations
• View balances
• Mint NFTs (script-only for stability)

Frontend → On-Chain

Through WalletConnect:

• View NFT ownership
• Resolve metadata
• Show trait matrix
• Fetch bonus token balances

Postgres Database Layer:

Prepared tables:
• sessions
• ore_balances
• boosts_cache
• burn_history

Backend uses Postgres to:

• Track progress
• Track resource accumulation
• Store burn records
• Cache expensive calls for UI responsiveness

Shard Frontier now supports a complete technical loop:

UI → X1 session → backend → Postgres → bonus token burns → on-chain events → updated boosts → frontend display.
The BDAG Awakening Testnet is fully integrated end-to-end with Shard NFTs, Bonus tokens, BurnVault, and the X1 simulation API."

Known Constraints (Wave-2 safety):

• On-chain mint buttons are intentionally disabled in UI to avoid testnet gas/RPC

issues; mint via Hardhat scripts only.
• Occasional Awakening RPC latency is noted; retriable with scripts.
• Env vs fallback: UI reads contract from env with a safe fallback to the live address.

Credits:

• Frontend UI/flows, Inventory IPFS integration, concept-aligned screens: Jaime Ruff

• Backend (Railway + Postgres), ERC-20 + BurnVault + X1 simulation endpoints: Mohamed Adam