HACKBLOX 2026

Web3 Track

Problem Statements · 24-Hour Hackathon · Intermediate Level

1 Decentralized Freelance Escrow Platform

Domain: DeFi / Payments

Background

Freelancers and clients on Web2 platforms (Upwork, Fiverr) rely on a centralized company to hold funds and resolve disputes, and lose a cut of every payment to platform fees. A trustless escrow removes the middleman while still protecting both parties.

Problem Statement

Build a decentralized escrow dApp where a client locks funds in a smart contract when hiring a freelancer. Funds are released in milestones as work is approved, and a basic dispute mechanism lets a third party (arbitrator) resolve disagreements.

Core Requirements

- Smart contract that locks client funds (ETH or a test ERC-20) tied to a specific job
- Support for at least 2–3 milestones per job, each released independently on client approval
- Freelancer can mark a milestone as "delivered"; client can approve or raise a dispute
- Simple arbitrator role that can resolve a disputed milestone (release to freelancer or refund client)
- Frontend with wallet connect (MetaMask) showing job status and milestone progress

Bonus / Stretch Features

- Store job details/deliverables on IPFS and reference the hash on-chain
- Reputation score for freelancers based on completed jobs
- Support ERC-20 stablecoin payments instead of raw ETH

Suggested Tech Stack

Solidity + Hardhat/Foundry, Ethers.js/Wagmi, React/Next.js, MetaMask, IPFS (web3.storage/Pinata), testnet (Sepolia)

2 On-Chain Verifiable Credentials (Soulbound Certificates)

Domain: Identity / Education

Background

Fake degrees and certificates are a real problem, and verifying a credential today usually means emailing the issuing institution and waiting days. Non-transferable (“soulbound”) NFTs can act as tamper-proof, instantly verifiable credentials tied permanently to a wallet.

Problem Statement

Build a dApp where an authorized issuer (e.g., a university or course platform) can mint non-transferable NFT certificates to a student's wallet, and anyone can instantly verify a certificate's authenticity and issuer on-chain.

Core Requirements

- Smart contract implementing a non-transferable (soulbound) NFT standard — block transfers after minting
- Only whitelisted issuer addresses can mint certificates
- Certificate metadata (name, course, date, issuer) stored on IPFS, hash referenced on-chain
- Public verification page: enter a wallet address or token ID and see all valid certificates + issuer info
- Basic issuer dashboard to mint new certificates

Bonus / Stretch Features

- Revocation mechanism for issuers (mark a certificate invalid without transferring it)
- QR code linking to the on-chain verification page for physical/PDF certificates
- Multiple issuer tiers (e.g., university admin can approve new sub-issuers/departments)

Suggested Tech Stack

Solidity (ERC-721 modified for non-transferability), OpenZeppelin, Ethers.js, React, IPFS, Sepolia/Polygon Mumbai testnet

3 Milestone-Based Decentralized Crowdfunding

Domain: DeFi / DAO tooling

Background

Traditional crowdfunding (Kickstarter, GoFundMe) releases all funds upfront once a goal is hit, leaving backers no recourse if a project stalls or the creator disappears. Releasing funds against verified milestones, with backer voting power, better aligns incentives.

Problem Statement

Build a crowdfunding dApp where backers pledge funds to a campaign, but funds are released to the creator in stages — only after backers vote to approve each milestone. If a milestone vote fails, backers can claim a refund of their remaining pledged funds.

Core Requirements

- Smart contract for creating a campaign with a funding goal, deadline, and defined milestones
- Backers can pledge funds (ETH/test ERC-20) before the deadline
- If goal is met, funds are locked and released milestone-by-milestone only after majority backer approval (simple on-chain voting, weighted by contribution)
- If a milestone vote fails or the goal isn't met by the deadline, backers can withdraw their remaining funds
- Frontend showing campaign progress, milestone status, and a voting interface for backers

Bonus / Stretch Features

- Campaign discovery page with filters (category, funding progress)
- Quadratic voting instead of linear (contribution-weighted) voting
- Creator reputation based on past successfully completed campaigns

Suggested Tech Stack

Solidity + Hardhat, OpenZeppelin (ERC-20/reentrancy guards), Ethers.js, React/Next.js, Sepolia testnet, The Graph (optional, for indexing campaign events)

Notes for Participants

All three problem statements are scoped to be buildable solo or in a team of 2–4 within 24 hours: one core smart contract, one frontend, and wallet integration.

Evaluation Criteria

Criteria	Weight
Working demo on testnet	40%
Smart contract correctness / security basics	25%
UI / UX	20%
Creativity / bonus features	15%