internStudio

BLOCKCHAIN & WEB3 INTERNSHIP

TASK GUIDE

Build on decentralized infrastructure.
Smart contracts and real dApps.

SIX ADVANCED PROJECTS — CHOOSE ANY TWO

01 **Decentralized Voting Application**

02 **NFT Marketplace for Digital Assets**

03 **Decentralized Crowdfunding Platform**

04 **Supply Chain Provenance Tracking**

05 **Decentralized Staking & Rewards Protocol**

06 **On-Chain Identity & Credential Verification**

INTERNSHIP	REQUIRED TASKS	FOCUS
Blockchain/Web3	Complete any 2 projects	Smart Contracts · Ethereum/Solidity · Wallets · dApp Frontends

INTERNSHIP INSTRUCTIONS

Your Internship Tasks

Select and complete **any 2 projects** from the task list, deployed to a real testnet with a working frontend — not just a contract that compiles.

Design	Write	Test	Deploy	Integrate	Document

EVERY PROJECT SHOULD INCLUDE

- [ ] Smart contracts deployed to a public testnet
- [ ] Automated tests for contract logic
- [ ] Wallet-based authentication (MetaMask or similar)
- [ ] Protection against common vulnerabilities (reentrancy, overflow)
- [ ] Gas-usage awareness in contract design
- [ ] A working frontend that talks to the deployed contract
- [ ] Clear on-chain event logging
- [ ] Documentation of the contract's design and trust assumptions

BEFORE YOU SUBMIT

- [ ] Complete any 2 projects from the list
- [ ] Push contracts and frontend to GitHub
- [ ] Include the deployed testnet contract address
- [ ] Add a README with setup and deployment steps
- [ ] Add screenshots or a demo video
- [ ] Submit details via the submission form
- [ ] Share your work on LinkedIn as required

> **Suggested technologies:** Solidity · Hardhat or Foundry · Ethers.js/Web3.js · a public testnet (Sepolia or similar) · React for the frontend.

Submit your completed work here

Once both projects are deployed to testnet, documented and pushed to GitHub, submit your project details through the official submission form.

https://forms.gle/vxL5JE8cvRXXfDBv8

> **A note on quality.** Smart contracts are unusually unforgiving — untested logic can be exploited the moment it's live. Write tests for the failure paths, not just the happy path.

01 Decentralized Voting Application

Build a voting dApp where eligibility, ballots and results are all enforced on-chain.

REQUIREMENTS

- Voter eligibility enforced by the contract
- Time-bound voting windows
- Wallet-based voter authentication
- Emit events for auditability
- One-vote-per-address enforcement
- Transparent, tamper-proof result tallying
- Prevent double-voting via re-submission
- Frontend showing live results as votes come in

Focus: On-chain access control, tamper-proof tallying and transparent governance mechanics.

02 NFT Marketplace for Digital Assets

Build a marketplace where users mint, list and trade NFTs with real on-chain ownership transfer.

REQUIREMENTS

- NFT minting following a standard (ERC-721/1155)
- Listing and delisting of owned NFTs
- Royalty handling for creators on resale
- Wallet-connected frontend gallery
- Metadata stored appropriately (IPFS or similar)
- On-chain purchase with correct ownership transfer
- Protection against reentrancy in purchase flow
- Event-driven activity feed (mints, sales)

Focus: Token standards, marketplace mechanics and secure on-chain asset transfer.

03 Decentralized Crowdfunding Platform

Build a crowdfunding dApp where funds are held in escrow and released only when milestones are met.

REQUIREMENTS

- Campaign creation with a funding goal and deadline
- Milestone-based fund release
- Contributor voting on milestone approval
- Clear on-chain record of all contributions
- Contributions held in contract escrow, not sent directly
- Automatic refund if the goal isn't met
- Protection against reentrancy on withdrawal
- Frontend showing campaign progress in real time

Focus: Escrow mechanics, milestone-based release logic and contributor governance.

04 Supply Chain Provenance Tracking

Build a system that records a product's journey through a supply chain immutably on-chain.

REQUIREMENTS

- Each supply-chain step recorded as an on-chain event
- Product identity tied consistently across steps
- Query the full provenance history for any item
- Frontend for scanning/looking up a product's history
- Role-based permissions per participant (producer, shipper, retailer)
- Tamper-evident history of custody changes
- Handle a participant acting outside their role (rejected)
- Document the trust model between participants

Focus: Role-based on-chain permissions and an immutable, queryable provenance record.

05 Decentralized Staking & Rewards Protocol

Build a simplified DeFi protocol where users stake tokens and earn rewards over time.

REQUIREMENTS

- Token staking and unstaking flow
- Correct handling of stake changes mid-period
- Emergency pause mechanism
- Frontend showing current stake, accrued rewards and APY
- Reward accrual based on stake amount and duration
- Protection against common DeFi exploits (reentrancy, flash-loan-style manipulation)
- Accurate accounting under concurrent stake/unstake actions
- Document the reward-calculation formula

Focus: DeFi mechanics, reward accounting and defensive contract design.

06 On-Chain Identity & Credential Verification

Build a system for issuing and verifying credentials (certificates, memberships) on-chain, without a central authority controlling verification.

REQUIREMENTS

- Credential issuance restricted to authorized issuers
- Public, permissionless verification of a credential's validity
- Prevent forged or duplicate credential claims
- Frontend to issue, view and verify credentials
- Credentials tied to a wallet address
- Revocation support by the original issuer
- Clear separation between issuer and holder roles
- Document the trust model (who can issue, who can revoke)

Focus: Verifiable on-chain credentials, issuer authorization and decentralized verification.

FINAL STEP

Ready to Submit?

Verify everything below before you send your work in.

SUBMISSION

- [ ] GitHub repository
- [ ] Deployed contract address included
- [ ] README with deployment steps
- [ ] Screenshots or demo video
- [ ] Submission form completed
- [ ] LinkedIn post where required

DESIGN

- [ ] Gas usage considered in design
- [ ] On-chain events logged clearly
- [ ] Trust assumptions documented

CONTRACTS

- [ ] Deployed to a public testnet
- [ ] Automated tests cover core logic
- [ ] Common vulnerabilities checked (reentrancy, overflow)

FRONTEND

- [ ] Wallet connection works reliably
- [ ] Contract state reflected accurately in the UI
- [ ] Transactions and errors surfaced clearly to the user

Important. Complete any 2 projects. Contracts must be deployed to a real public testnet with a working frontend — a contract that only runs in a local sandbox is not a finished project.

Submission Form

Submit your repository, live link and screenshots here.

https://forms.gle/vxL5JE8cvRXXfDBv8

internStudio

LEARN · BUILD · GROW

www.internstudio.in

team.internstudio@gmail.com