UnCodemy

Blockchain Mini Project Ideas PDF

A blockchain mini project should teach one idea in one to four weeks: how blocks and hashes link, how proof of work works, or how a simple smart contract stores and changes state. Building a mini blockchain in Python, a to-do list contract, an ERC-20 token and a simple voting dApp on a test network are good choices.

Key takeaways

- Build a small blockchain in Python first to understand hashing and blocks.
- Then write Solidity contracts in Remix, which runs in the browser.
- Use local networks or testnets only; no real money is needed.
- OpenZeppelin gives safe, standard token contracts.
- Each idea fits in one to four weeks.

Who this PDF is for

This PDF is for 2nd and 3rd year students who want to learn blockchain by building. Early ideas need only Python; later ones use Solidity in Remix IDE or Hardhat, MetaMask and a test network. Each plan lists the steps and what to explain in the viva.

How to choose your project

- **Concepts first**: hashing, blocks, consensus, then contracts.
- **Remix IDE** needs no setup and is ideal for first contracts.
- **Test networks only**, with free faucet tokens.
- **Keep the front end simple** or skip it for the first project.

Beginner mini projects (1 to 2 weeks)

1. Mini Blockchain in Python

- **Problem:** Build a simple chain of blocks linked by hashes.
- **Network and tools:** Your laptop only.
- **Tech stack:** Python, hashlib, Flask.
- **Main modules:** Block class, hashing, chain validation, tamper test, simple API.
- **Steps:** Create blocks with index, time, data and previous hash, validate the chain, show tampering breaks it.
- **Show in the viva:** How changing one block breaks every later hash.
- **Extensions:** Add proof of work.

2. Proof of Work Demo

- **Problem:** Show how mining difficulty changes time.
- **Network and tools:** Your laptop only.
- **Tech stack:** Python, matplotlib.
- **Main modules:** Nonce search, difficulty setting, timing chart.
- **Steps:** Search for a nonce giving a hash with leading zeros, time it for several difficulties, chart.
- **Show in the viva:** Why difficulty grows mining time quickly.
- **Extensions:** Compare with a proof of stake toy model.

3. Hello World Smart Contract

- **Problem:** Store and update a message on-chain.
- **Network and tools:** Remix VM (in browser).
- **Tech stack:** Solidity, Remix IDE.
- **Main modules:** State variable, setter, getter, event.
- **Steps:** Write the contract, deploy in Remix VM, call functions, read events.
- **Show in the viva:** Difference between a call and a transaction.
- **Extensions:** Restrict updates to the owner.

4. Hash-based File Integrity Checker

- **Problem:** Check whether a file has been changed.
- **Network and tools:** Your laptop only.
- **Tech stack:** Python, hashlib, Tkinter or Streamlit.
- **Main modules:** File hashing, stored hashes, compare, report.
- **Steps:** Hash files with SHA-256, save hashes, re-check and flag changes.
- **Show in the viva:** Properties of a cryptographic hash.
- **Extensions:** Store hashes on a testnet contract.

Intermediate mini projects (2 to 3 weeks)

5. To-do List dApp

- **Problem:** Store tasks in a contract and manage them from a web page.
- **Network and tools:** Hardhat local network, MetaMask.
- **Tech stack:** Solidity, Hardhat, ethers.js, HTML or React.
- **Main modules:** Task struct, add and toggle functions, events, front end.
- **Steps:** Write the contract, test it, deploy locally, connect MetaMask, list tasks.
- **Show in the viva:** How the front end talks to the contract through ethers.js.
- **Extensions:** Deploy on Sepolia.

6. ERC-20 Token

- **Problem:** Create your own test token and transfer it.
- **Network and tools:** Sepolia testnet or a local network.
- **Tech stack:** Solidity, OpenZeppelin ERC20, Hardhat or Remix.
- **Main modules:** Token contract, mint, transfer, balance check.

- **Steps:** Inherit OpenZeppelin ERC20, set supply, deploy, transfer between test accounts in MetaMask.
- **Show in the viva:** What the ERC-20 standard defines.
- **Extensions:** Add a faucet function with limits.

7. Simple Voting Contract

- **Problem:** Vote once per address for a candidate.
- **Network and tools:** Remix VM or a local network.
- **Tech stack:** Solidity, Remix or Hardhat.
- **Main modules:** Candidates, vote function, one-vote rule, results.
- **Steps:** Store candidates, map voters to a voted flag, count votes, test double voting fails.
- **Show in the viva:** How mappings work in Solidity.
- **Extensions:** Add a voting deadline.

8. Lottery Contract (Learning Exercise)

- **Problem:** Pick a random winner from test entries.
- **Network and tools:** Local network or Sepolia.
- **Tech stack:** Solidity, Hardhat, Chainlink VRF documentation for the advanced version.
- **Main modules:** Entry, pick winner, payout, owner controls.
- **Steps:** Build with test ETH only, explain why block data is poor randomness, read about verifiable randomness.
- **Show in the viva:** Why on-chain randomness is hard.
- **Extensions:** Integrate Chainlink VRF on a testnet.

Stretch mini projects (3 to 4 weeks)

9. NFT Minting Page

- **Problem:** Mint a simple NFT with an image.
- **Network and tools:** Sepolia testnet, IPFS.
- **Tech stack:** Solidity (ERC-721 via OpenZeppelin), IPFS, React.
- **Main modules:** Contract, metadata JSON, IPFS upload, mint button.
- **Steps:** Upload image and metadata to IPFS, mint with token URI, view in a wallet.
- **Show in the viva:** What token URI metadata contains.
- **Extensions:** Add a mint limit per wallet.

10. Peer-to-peer Blockchain Network

- **Problem:** Run your Python blockchain on several nodes.
- **Network and tools:** Several terminals or laptops on one network.
- **Tech stack:** Python, Flask, requests.
- **Main modules:** Node registration, block broadcast, longest-chain rule.
- **Steps:** Run three nodes, register peers, mine on one, sync the others.
- **Show in the viva:** How the longest-chain rule resolves conflicts.
- **Extensions:** Add transaction signing.

11. Digital Signature Demo

- **Problem:** Sign and verify messages with key pairs.
- **Network and tools:** Your laptop only.
- **Tech stack:** Python cryptography library or ethers.js.
- **Main modules:** Key generation, signing, verification, tamper test.
- **Steps:** Generate keys, sign a message, verify, change one character and verify again.
- **Show in the viva:** Public and private keys in plain words.
- **Extensions:** Verify a MetaMask signature in a contract.

12. Charity Donation Tracker

- **Problem:** Show donors where each donation goes.
- **Network and tools:** Sepolia testnet.
- **Tech stack:** Solidity, Hardhat, React.
- **Main modules:** Donate function, spending records, public history page.
- **Steps:** Record donations and spends with events, show history on a page.
- **Show in the viva:** Why events are cheaper than storage for history.
- **Extensions:** Add donor receipts as NFTs.

Mini project report structure

A mini project report is usually 10 to 20 pages. Keep it short and clear:

1. **Title page and certificate** in your department's format.
2. **Introduction:** the problem in one paragraph and your objectives as a list.
3. **Tools used:** Python or Solidity version, Remix or Hardhat, MetaMask, network used.
4. **Design:** one block diagram or flowchart, plus a diagram of contract functions or block structure.
5. **Implementation:** what each module does, with 3 to 6 screenshots.
6. **Results:** screenshots of deployed contracts, transaction hashes and test runs.
7. **Conclusion and future scope,** then references.

Ask your guide for the exact format before you start writing.

Common mistakes to avoid

- Using a real wallet with real funds for testing.
- Sharing your seed phrase or private key in code.
- Skipping tests and finding bugs after deployment.
- Not explaining why the project needs a blockchain.

Frequently asked questions

What is the easiest blockchain mini project?

Building a mini blockchain in Python. It teaches blocks, hashes and validation without any network setup.

Do I need to buy cryptocurrency?

No. Remix VM and local Hardhat networks are free, and testnet tokens from faucets have no real value.

Which language is used for smart contracts?

Solidity is the most common for Ethereum and compatible chains. Remix IDE lets you write and run it in the browser.

Can blockchain be a 2nd year mini project?

Yes. Start with the Python blockchain or a simple Solidity contract, which need only basic programming.

Does this PDF include full source code?

No. It gives the problem, data, stack, modules and steps for each project. You write the code yourself, which is what examiners test in the viva.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- Remix IDE documentation
- Solidity documentation
- OpenZeppelin Contracts
- Ethereum developer documentation

About this PDF

Prepared by the Uncodemy training team as practice project ideas. Datasets and tools named are public; check each dataset's licence before use. We update it every January. Corrections: info@uncodemy.com.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550