Sep 10, 2023 5 min read

Ten beginner project ideas after you learn Solidity

Updated: Oct 14, 2023

Now that you've completed our solidity tutorial, what's next?

You now have enough knowledge to build any of the following projects. Knowledge comes by study, but skill comes with practice. You need both knowledge and skill if you want to be a successful solidity developer! Here are some suggestions about what you can build to apply everything you've learned so far.

You have everything you need to know, so get your hands dirty if you are serious about becoming a solidity engineer!

These projects are ordered approximately from easiest to hardest, but you have all the background knowledge you need to build any of these.

1. Purchase NFT with ERC20 tokens

Bulld a classic NFT that can only be minted by paying with a particular ERC20 token.

2. Time unlocked ERC20 / vesting contract

A payer deposits a certain amount of tokens into a contract, but the receiver can only withdraw 1/n tokens over the course of n days.

3. NFT Swap Contract

Two people want to trade their NFTs in a trustless way. A user creates a swap on the contract, which is a pair of address, ids where the address is the smart contract address of the NFT and the id is the tokenId of the NFT. One person can deposit an NFT only if the id matches the address and id. The counterparty can deposit only if their NFT matches the address and id of the swap.

Once both are deposited, either party can call swap.

Some corner cases to think about:

- **how long, if at all, should the traders be forced to keep their NFT in the contract?**

4. Crowdfunding ERC20 contract

Your contract should have a createFundraiser() function with a goal and a deadline as arguments. Donators can donate() to a given fundraiserId. If the goal is reached before the deadline, the wallet that called createFundraiser() Can withdraw() all the funds associated with that campaign. Otherwise,

if the deadline passes without reaching the goal, the donators can withdraw their donation. Build a contract that supports Ether and another that supports ERC20 tokens.
Some corner cases to think about:

- what if the same address donates multiple times?
- what if the same address donates to multiple different campaigns?

5. English auction contract

A seller calls deposit() to deposit an NFT into a contract along with a deadline and a reserve price. Buyers can bid on that NFT up until the deadline, and the highest bid wins. If the reserve price is not met, the NFT is not sold. Multiple auctions can happen at the same time. Buyers who did not win can withdraw their bid. The winner is not able to withdraw their bid and must complete the trade to buy the NFT. The seller can also end the auction by calling sellerEndAuction() which only works after expiration, and if the reserve is met. The winner will be transferred the NFT and the seller will receive the Ethereum.

6. Simple NFT Marketplace

Sellers can sell() their NFT while specifying a price and expiration. Instead of depositing the NFT into the contract, they give the contract approval to withdraw it from them. If a buyer comes along and pays the specified price before the expiration, then the NFT is transferred from the seller to the buyer and the buyer's ether is transferred to the seller.
The seller can cancel() the sale at any time.
Corner cases:

- What if the seller lists the same NFT twice? This can theoretically happen since they don't transfer the NFT to the marketplace.

7. Stake Together

A contract owns 1,000,000 cloud coins. Anyone who stakes cloud coin into the contract starting on the beginDate and holds it for 7 days will receive a reward proportional their portion of the total stake at the expiration. For example, suppose Alice stakes 5,000 cloud coin, but the total amount staked at expiration is 25,000 cloud coin. Alice will then be entitled to 200,000 of the rewards, because she accounted for 20% of all the users.

Warning: it's very easy to accidentally compute the rewards in such a way that a malicious actor can abuse the system. Think carefully about the corner cases!

8. Simple lottery

Any user can call createLottery and a lottery will be created with a ticket purchase window for the next 24 hours. Once the 24 hours is up, there is a 1 hour delay, then the lottery is over. Generating random numbers safely on Ethereum is tricky, but for the purpose of this, relying on a future blockhash (which the players cannot predict), is good enough for this project. After createLottery is called people can purchaseTicket for a particular lotteryId. The lottery must consist of a deadline for when purchasing tickets stops, and time afterwards when the future blockhash determines the winner. The winner must then claim the winnings within 256 blocks (the maximum lookback of the blockhash function), otherwise, everyone can get their tickets back.

9. ERC1155 Bingo

Use ERC1155 tokens to simulate a deck of cards that can take on any value from 1-25 inclusive. Each player starts with a 5x5 2d array of the numbers 1-25 randomly arranged. Every n blocks, players can mint a new card which has a random number. Whoever gets the first 5 in a row (bingo) wins.

10. On-chain blackjack

Unlike regular blackjack, it is extremely difficult to hide the dealer's card, so have everyone have a fully open hand. Real blackjack is usually played with multiple decks to make card counting less effective, so you can have random number generator produce a random number from [2-10] but keep in mind ten, jack, queen, and king are all 10, so you need to make the probabilities proportional. Similarly, and Ace can be a 1 or an 11. The dealer must hit until they are at least 21.
Because smart contracts cannot move state forward on their own, anyone can call dealerNextMove() to keep the game moving forward if it is the dealer's turn. In a real application, you'd need an offchain computer to keep the dealer going, but let's not worry about this for now.
You should force players to make their move within 10 blocks to avoid anyone holding the game up.

Practice makes perfect

The reason our learn solidity resource is accompanied with the Solidity exercises, is because it's a trap to learn coding purely by studying. It's more important to code! Now you can do something a little more challenging than solving bite-sized exercises.
Really, stop reading tutorials, start creating projects! If you get stuck, reach out in our discord community, search on Google, or ask a modern AI chatbot for help.

Quality assurance: write unit tests and measure the gas costs

We've already taught you to use the foundry framework for writing solidity unit tests. It's simply not acceptable to deploy untested code that handles other people's money. Although these are just educational contracts, you should get into the habit of writing tests early in your journey.

Try thinking like an attacker too. What sort of funny inputs would an attacker try if they were trying to mess with the system?

Finally, pay attention to the gas cost that foundry reports. You can convert the gas units to dollar cost by using our Ethereum gas price calculator and learn how that calculator works by reading our article on how to convert eth gas to dollars.

Then consider learning Solidity gas optimization tips and applying them to your project.

If you really want to go the extra mile, build a simple frontend app for connecting to these contracts. We suggest using the wagmi js library for this.

Then what?

Done? We suggest learning how to gas optimize your contracts by reading our gas optimization book and taking the courses linked therein. Then try to gas optimize the contracts you've created!

Learn More with RareSkills

If you've completed all of the above and are looking to go even further with a community, maybe consider taking our advanced Solidity bootcamp.