

SMART CONTRACT  
TEST PROJECT

*Hello, Smart Contract Developer!*

Welcome to your test project! As you’ve probably heard, STRV’s company culture is rooted in freedom. We provide the opportunities, and it’s in your hands to shape them into something extraordinary. This approach is reflected in our hiring process as well.

Our doors are very, very open. We only do a basic screening of our candidates because we want everyone to get the opportunity to show us what they’ve got. So when you come in to meet us, know that we’re going in with genuine excitement. We’re ready to see how good you are and to be convinced that you’ll be an awesome addition to STRV. The same goes for this test project. Blow us away!

*Please note: By submitting the test project, you agree that the code will become the property of STRV.*

# PURPOSE

You’re probably curious to see what working at STRV looks and feels like. We’re curious to see your approach to a challenge. This small test project fulfills both purposes.

The test project is designed to reflect the sort of work we do at STRV, so it includes all major technologies and techniques that our team comes across on a daily basis. For experienced developers, it shouldn’t take more than 2 days (16 working hours) to complete. For slightly more junior developers, it might take a bit longer. But that’s okay! As we said, STRV gives a chance to nearly everyone. What matters is the effort, the determination, and the final outcome.

...Still, we do expect to have the test project back within a week. Taking longer is seen as the first sign that you might not be a perfect fit for our team.

Your test project will not be used for any other purpose than evaluation of you as a candidate for this position.

# INSTRUCTIONS

**Please design and implement an ERC-1155 multitoken smart contract in Solidity for an NFT Game.**

The Smart Contract should enable its users to use NFT features in a game about knights. It should allow them to collect gear and further improve it to increase its strength and value.

**PLEASE NOTE THAT YOU SHOULD NOT DEVELOP THE FRONTEND OF THE GAME. FOR THE PURPOSE OF THIS TEST PROJECT, FOCUS ONLY ON THE SMART CONTRACT (a.k.a THE BACKEND OF THE GAME)**.

Commit your source code to a repository on Bitbucket or Github. When you are done, invite [michal.klacko@strv.com](mailto:michal.klacko@strv.com) and [martin@strv.com](mailto:martin@strv.com) to your repository (MichalKlacko and MartinStava - on GitHub) and to the TestNet Blockchain. Please also send us the links to your bitbucket repository and the deployed address of the smart contract.

Finally, please send us the source code via the submission page — which you can find at the bottom of the email with the test project. If you have any questions, just let us know at [jobs@strv.com](mailto:jobs@strv.com).

# SPECIFICATIONS

**Language:** Solidity or Vyper

**Development environment:** Truffle or Hardhat

**Blockchain:** Ethereum - TestNet

## Project description

Design and implement an ERC-1155 multitoken smart contract for a medieval NFT Game.

Each player will be represented as a Knight (knight is not a token). Knights can collect various gear, specifically: Armor, Sword, Shield. The gear should be considered as a non-fungible token.

Players will be able to “burn” a shield, which will give them an 80% chance of getting a sword but also a 20% chance of not getting anything (and losing that shield).

And as a proper starting NFT game, we want to allow public minting of up to 1,000 items.

## Features Overview

* Minting Equipment
* Burn Equipment
* Need Basic Governance (allow ownership, pause, resume, change fees….)

## Minting Equipment

* The first 1,000 items (equipment) will be available for public minting
* Everyone can mint an unlimited amount of items for 0.1 ETH each until the total limit of 1,000 is reached
* Available equipment types are: Armor, Sword, Shield
* All equipment should have a name and ID (name is “Armor 1” where 1 is ID)
* There is no limit for the total supply of equipment (dynamic)

## Burn Equipment

* Each user can burn a shield
* After burning the shield, there is:
  + 80% chance of getting a new sword
  + 20% chance of not getting anything
  + The shield will be lost in both cases
* We should be able to change the rules of “burning”

# REVIEW PROCESS

There are a few technical restrictions that allow us to see how you fare with the technologies and processes we use on a daily basis. But in general, the actual implementation is quite open-ended.

The following recommendations should help you determine what to focus on, since these are the things we will be looking at while reviewing your project.

# RECOMMENDATION

* You can use OpenZeppeline as well if you’d like
* Demonstrate knowledge of testing smart contracts
* Remember: Random in Smart contracts is not the same as random in regular code —> use a verifiable source of tamperproof randomness
* Rules of the burning of items should be an upgradable feature…. One can say it’s a “Burning contract”

*Good luck from all of us!*