Smart Contract Code Main Steps:

1. Creation of the virtual environment and installation of all necessary packages and software:

*This step consisted mainly of finding the best way to deploy a contract on a local chain so as to be able to test it on the language the project team was most used to:* ***Python****.*

*Initial tests were made on the* ***infura.io*** *server host with* ***MetaMask*** *chrome extension for ease of multiple computer contributions. The project contract deployment was then transferred for final tests and sample run notebook to* ***Ganache*** *to profit from multiple testing wallets.*

*The main package used to interact with the contract was the* ***Web3*** *package available on multiple programming languages (here python).*

1. Initial starting block was a premade fundraising contract found on **Github**

(<https://github.com/ankitbrahmbhatt1997/Ethereum_real_life_examples>)

*The author this Github provided 3 samples of main uses for smart contracts being* ***Auction, Lottery*** *and* ***Fundraising.*** *Considering our project focus is the implementation of Blockchain technology for NGOs, we used the latter contract as our* ***building block****.*

1. Creation of the main functions 🡪 *payable* and *PoW social consensus*

*Most of the initial work for this step was going through different forums (stackoverflow, Kaggle, etc.) to understand how* ***require() callbacks*** *are implemented in solidity smart contracts. After that, the task for the* ***payable*** *function was fairly straight-forward, as most of the simple smart contracts have similar payable functions with differing requirements. An* ***only admin modifier*** *was also implemented as found in most smart contracts.*

*The* ***Proof-of-Work social consensus*** *based on information miners voting to allow expenses using the funds gathered was a trickier endeavour. As solidity is not meant for calculations but mostly storing data and filtering/requiring inputs, the use of* ***structures, mappings and arrays*** *seemed like the best option to keep track of all inputs made by various addresses. To understand the different aspects of these types in solidity, several well-made websites and forums, such as the following, were used:*

<https://coursetro.com/posts/code/102/Solidity-Mappings-&-Structs-Tutorial>

<https://medium.com/upstate-interactive/mappings-arrays-87afc697e64f>

*For information on how each of this were used in the code please refer to the report and corresponding comments in the solidity contract and/or testing notebook.*

1. SCRAPTED – Transferring unused funds

*To avoid useless donations, our team initially decided on transferring extra funds present in a project to other similar projects done by the NGO. To implement this option, the use of* ***factory contract*** *was experimented on (*[*https://medium.com/@i6mi6/solidty-smart-contracts-design-patterns-ecfa3b1e9784*](https://medium.com/@i6mi6/solidty-smart-contracts-design-patterns-ecfa3b1e9784)*). These are contract holding other contracts so that interactions and transfers between them can be done.*

*However, after discussing with the professor and seeing how solidity is not meant for such interactions, the idea was put aside as it was considered too advanced for the purpose of this project.*

1. PLAN B – Providing a message to donors to avoid excess contribution.

*As we were informed the idea explained in 4) was too ambitious, we decided to replace it with a* ***require callback*** *restricting donations that were exceeding project goals (raised funds + donation > project goal). Nevertheless, to* ***inform donors*** *on the amounts required, a solidity version of string concatenation was created to concat the missing amount value to the error message of the* ***require callback.*** *Source:* <https://ethereum.stackexchange.com/questions/10811/solidity-concatenate-uint-into-a-string>

*Once again, as solidity is* ***not*** *a scripting language, this was no easy task as values can’t be negative, only bytes can be added to each other, returning hexes, etc.*

1. Provide payback if project does not go through

*So that donors would not be losing money of the project does not go through. We decided to implement an* ***additional mapping*** *to store and track* ***individual donations.*** *This allowed for the creation of an additional payback function of the exact donated amount (****minus the gas used to mine block****) which the donor could theoretically click on the UI* ***if, and only if,*** *the goal was not achieved but the deadline passed.*

1. Utility **python** functions:

*Another problem caused by solidity being the smart contract programming language was its* ***lack of datetime implementation*** *or* ***conversion possibilities*** *(bad calculation capabilities).*

*For* ***ease of use and understandability*** *of the contract for the NGO and/or users, 2 utility functions were created in the testing notebook:*

* ***Deadline conversion:*** *this function’s purpose was to transform a given deadline in number of days to a* ***deadline accessible by solidity: block numbers.*** *Using the data provided here* [*https://medium.facilelogin.com/the-mystery-behind-block-time-63351e35603a*](https://medium.facilelogin.com/the-mystery-behind-block-time-63351e35603a)*, and simple division, we created a loop* ***transforming number of days into number of mined blocks****, which the contract would then see as being the deadline. Ethereum block chain has a block mined every 10 to 19 seconds, we decided to go with 12 seconds as it was the most common number agreed upon online.*
* ***Dollar to Wei:*** *as Weis or even Ethers are most likely not something* ***potential users*** *of the smart contract are* ***familiar with****, we decided to implement a python function to retrieve the* ***real-time value*** *of ethers to convert donation into Weis* ***automatically*** *before transferring and avoiding this difficulty for potential clients. This was done using* ***coinapi*** *with the help of the following notebook:* [*https://notebooks.ai/santiagobasulto/coin-api-live-docs-4a628f39*](https://notebooks.ai/santiagobasulto/coin-api-live-docs-4a628f39)*.*