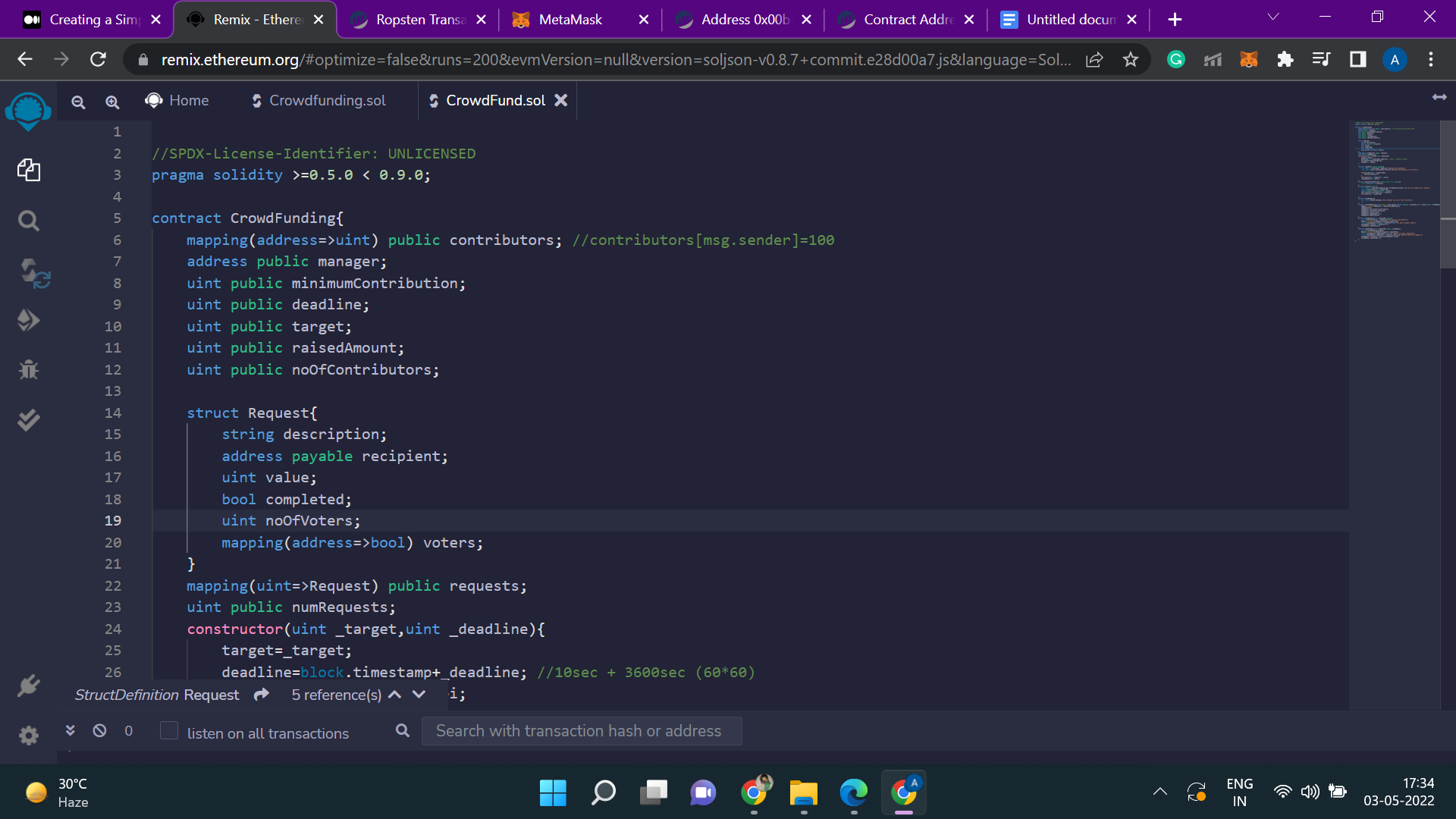
Code snippets:



Created a contract named CrowdFunding. To define the contributors and to link the address with them, mapping is used. We have created and set the address for the manager.

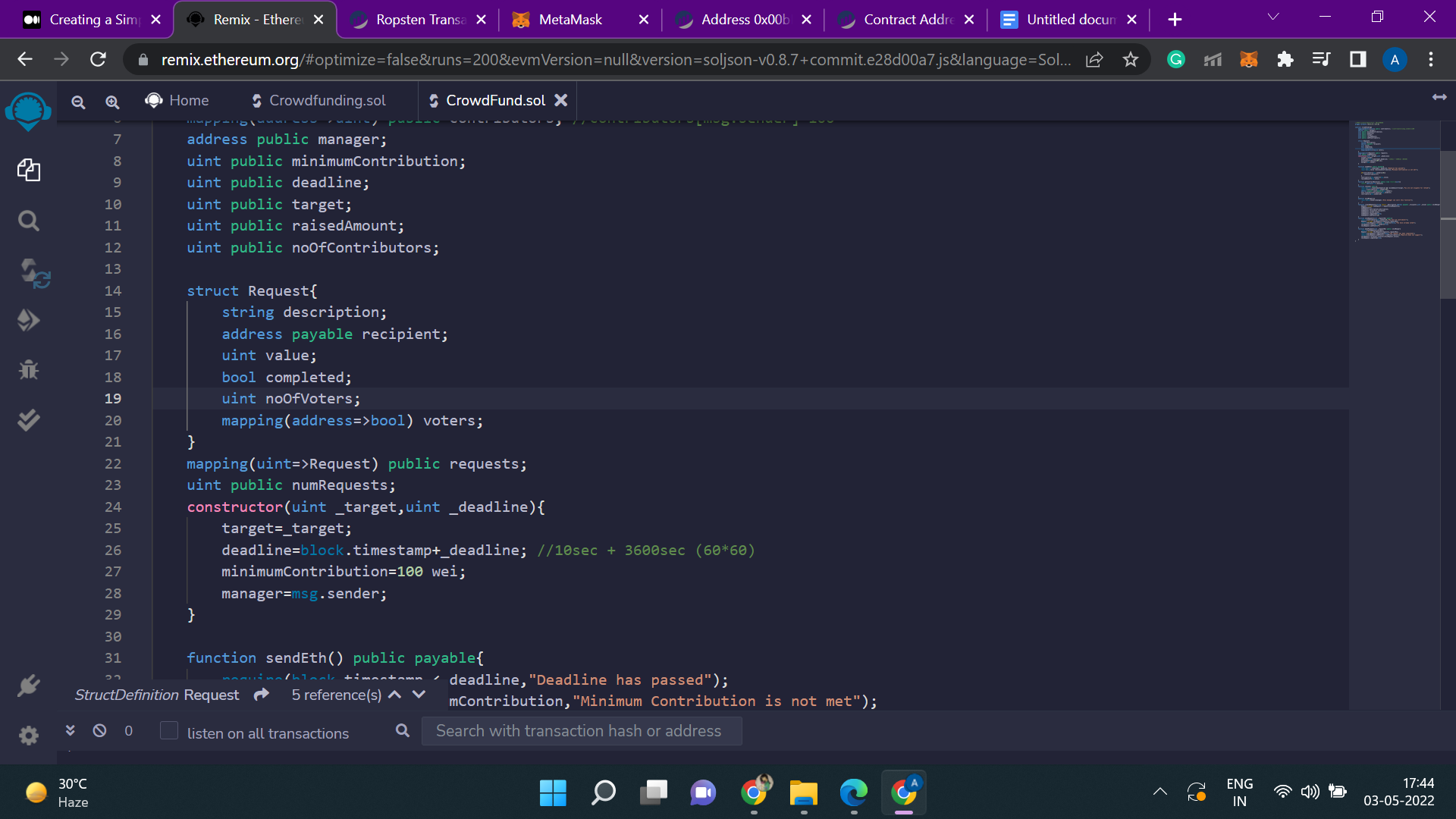
Following are the variables that we have defined:

i) “minimumContribution” - to set some minimum contribution amount for the contributors.

ii) “deadline” - to set the deadline of the request created.

iii) “raisedAmount” - to store the amount raised by the contributors.

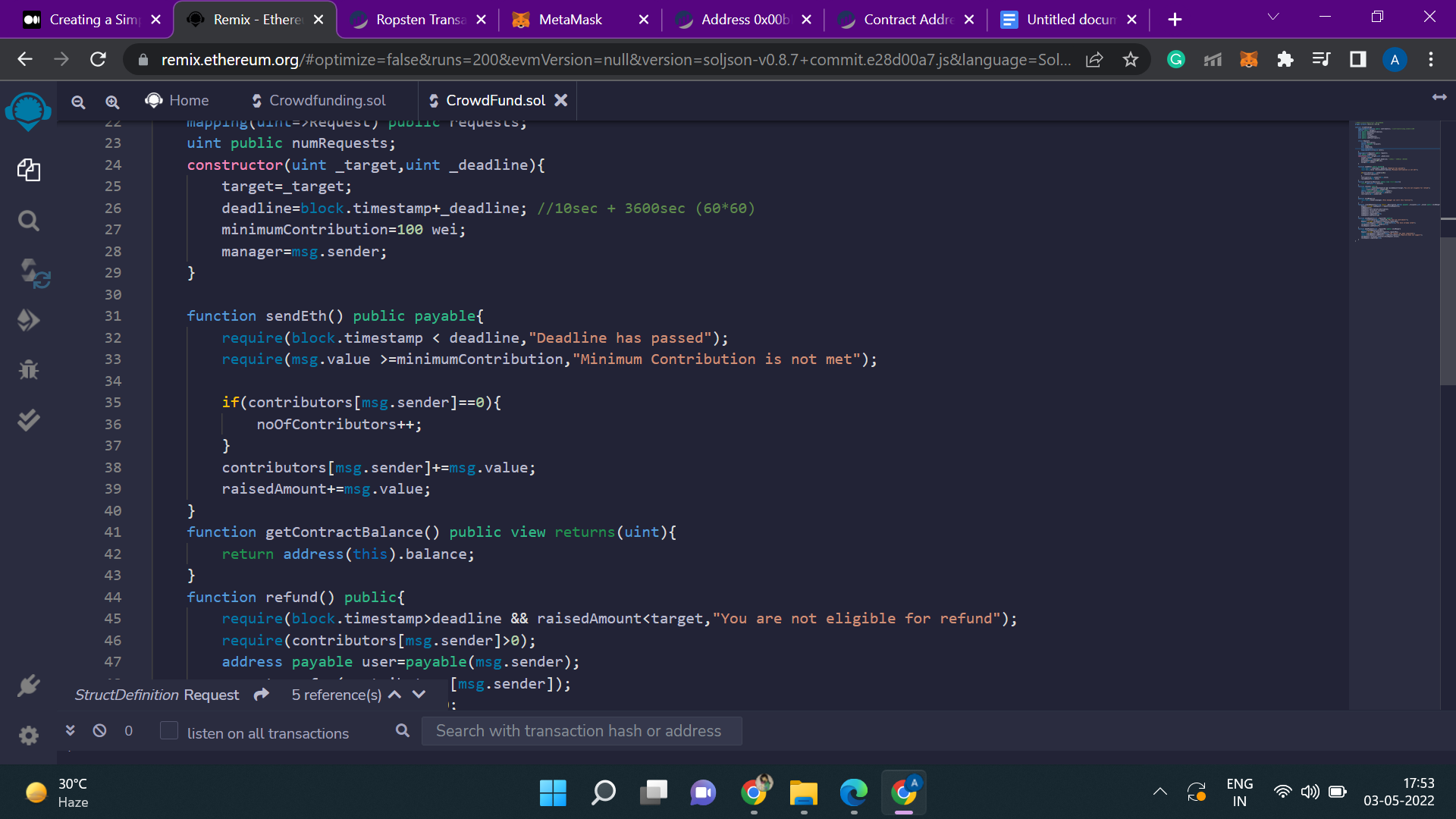
iv) “noOfContributors” - to count how many have contributed.



Here the manager who will deploy the contract will set the target and the deadline for the request.

To calculate the deadline, we have used “block.timestamp”, which is an unix timestamp. It indicates when the block is created. It is a global variable in solidity through which we can know the timestamp of the current block(measured in seconds).

We have initialised the “minimumContribution” to 100 wei and set the manager to msg.sender.



The function “sendEth()” is made public and payable so that the contributors can send ethers, keeping in mind the minimum contribution and deadline.

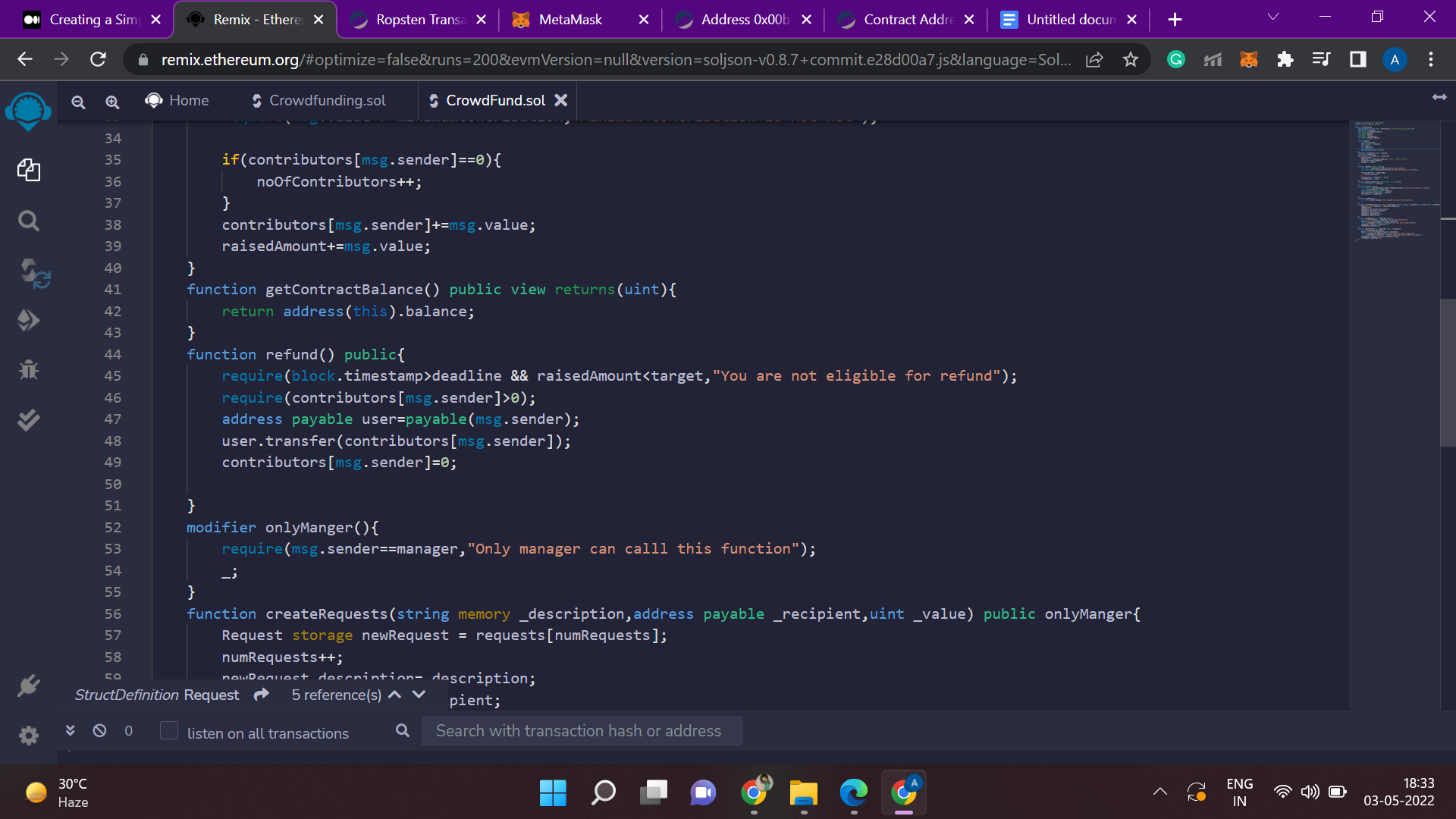
To ensure this, two require statements have been added to check:

i) whether the deadline has passed or not. If not, then display the message that “Deadline has passed”.

ii) whether the contributor have sent ethers greater than the minimum contribution set by the manager. If not, then display the message “Minimum Contribution is not met”.

After that, we will check whether the contributor is paying for the first time or have paid earlier. If paid for the first time, then increase the “noOfContributors” to 1 from 0; otherwise don’t increment it.

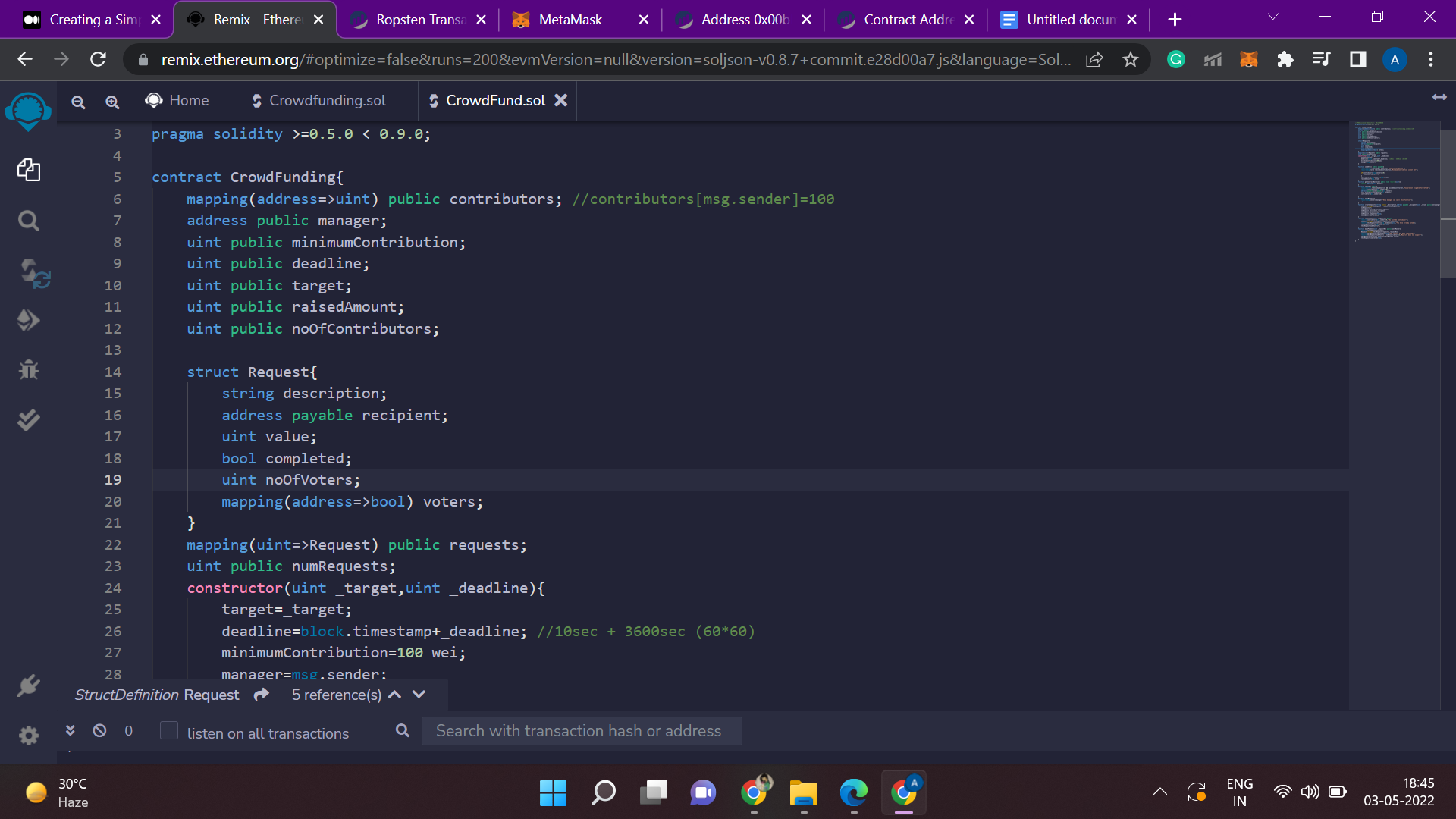
Also, if a contributor pays for more than 1 time, then the value it had paid will be considered and added to the “raisedAmount”.



This function “refund()” is responsible for the refund of ethers back to the contributors in case deadline is not met and when the amount raised is less than the target amount.

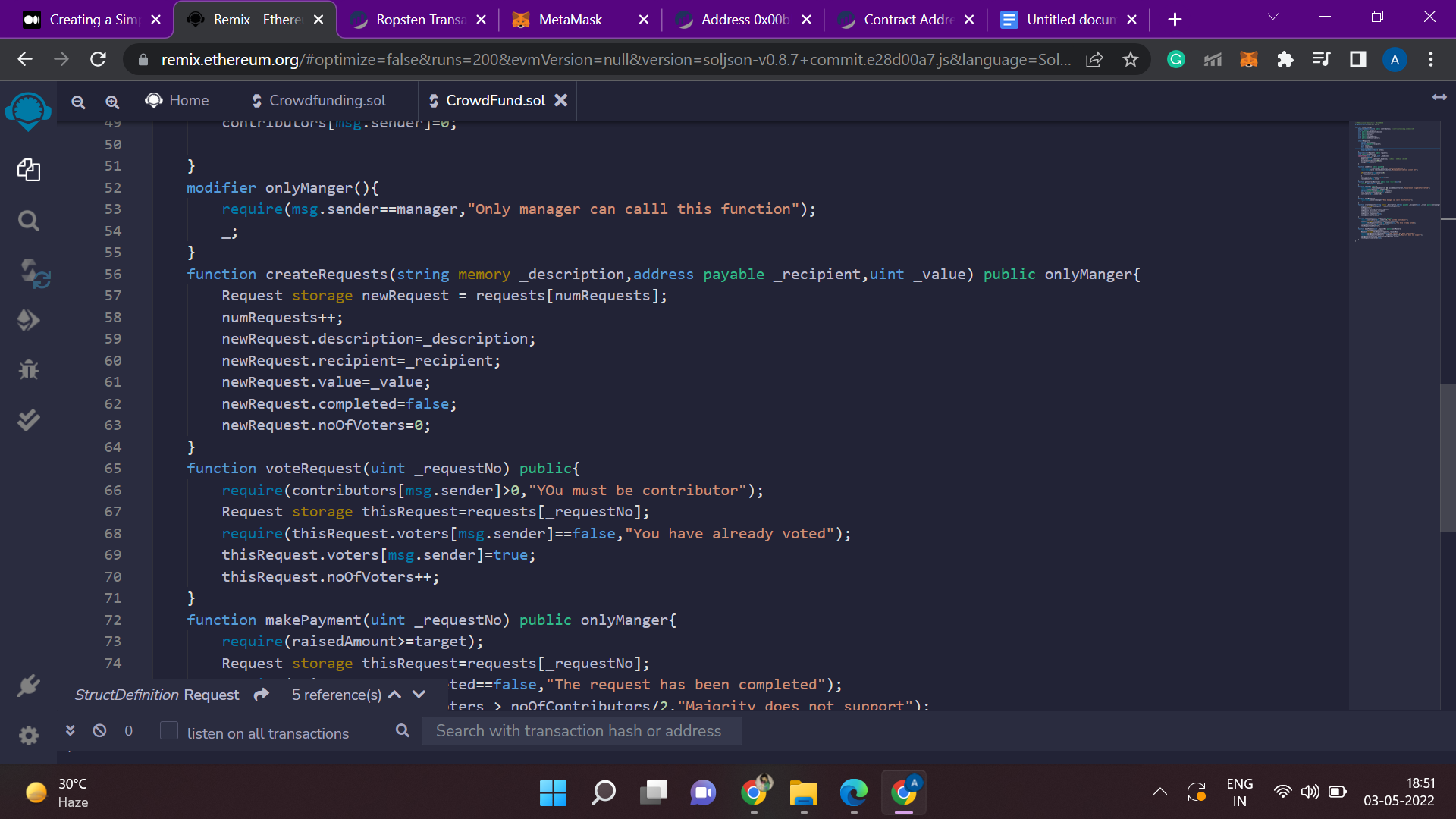
If a contributor asks for refund of the amount that it had never paid, then to ensure this we will add another require statement.

After the refund, the account of the “msg.sender” will be set to 0.



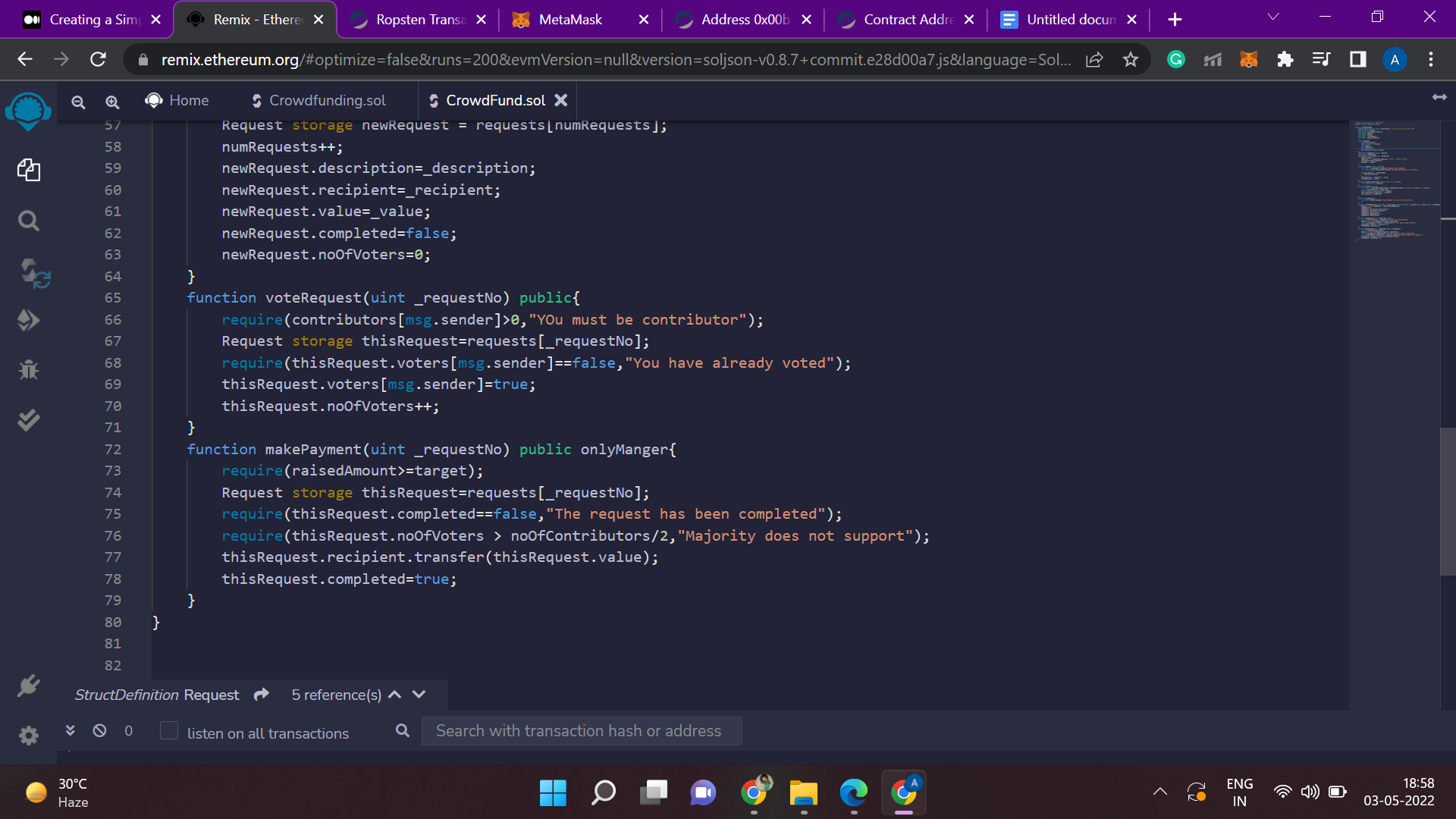
This will be from the manager’s point of view, when the ethers have to be taken out.

Here the address of the recipient is added, where all the ethers will be given.



The function “createRequests” is accessible only by the manager. Here the address will be specified for which we are creating the request. Total value(ether) required for the request is also passed to the function.

For every new request been created, it is pointing to the “struct Request”.

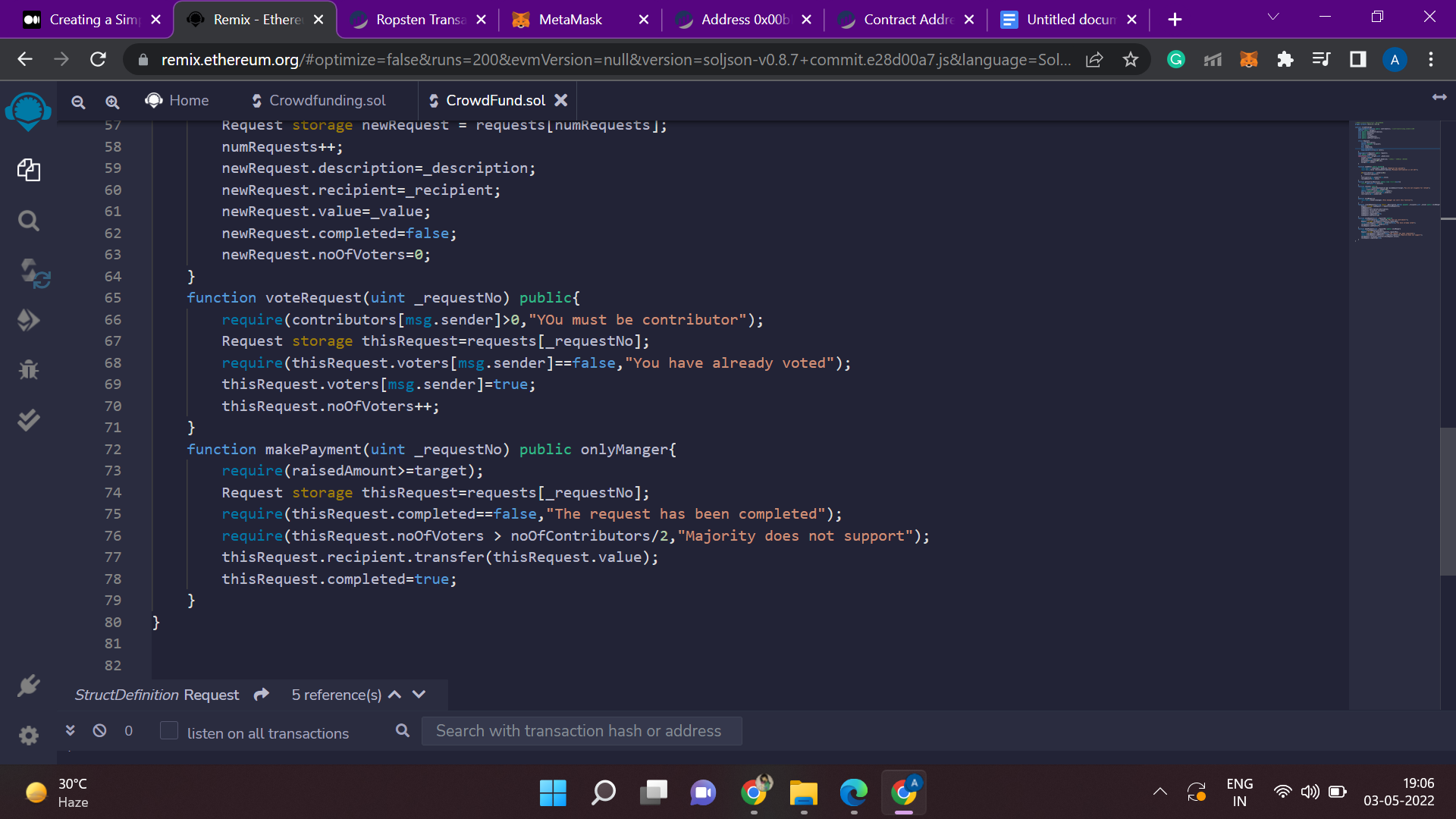


The function “voteRequest” is used to count the number of votes of the contributors for a particular request.

First it will check if the person has even contributed any ether or not. If not, then it is not eligible to vote.

Then it will point to that request for which majority of the contributors want to contribute ethers and vote.

If a contributor has already voted, then it’s vote will not be counted and error message will be displayed. And if it is voting for the first time, then increase the number of voters (“noOfVoters”).



The function “makePayment()” will tell the request number for which payment is to be made. This function is accessible only by the manager. It will check:

i) if the raised amount is greater than the target or not.

ii) if the request is already completed.

iii) if number of votes is greater than 50% of the total contributors.

After all the requirements have been satisfied, the manager will transfer the raised amount to the recipient and the request will be marked as completed.