ICT3009: Blockchain and Smart Contracts

Due on Fri, February 26th, 2021

Assignment Part 1

Dr Joshua Ellul

Instructions

- This is individual assignment work.
- This part of the assignment carries 85% of the final ICT3009 grade.
- The firm submission deadline is Monday 26th February 2021. Hard copies are not required to be handed in.
- A soft-copy of the report and all related files must be uploaded to the VLE upload area by the same deadline. All files must be archived into a single .zip file. It is the student's responsibility to ensure that the uploaded zip file and all contents are valid.
- You are to allocate 90 to 100 hours for this assignment.
- Reports (and code) that are difficult to follow due to low quality in the writing-style/organisation/presentation will be penalised.
- You are to document any steps taken in designing and implementing your smart contracts, associated tests and user interfaces.
- You must include all source and any compilation or configuration files and any scripts required to compile and/or run the smart contracts, tests and user interfaces.

Problem 1

You are to create Solidity smart contracts to cater for the following scenario.

It is becoming increasingly harder for younger and low income individuals to get loans from banks. At the same time current interest rates on savings are low, and many would like to be able to invest in areas that provide higher returns (which would also entail higher risk). Decentralised lending to anonymous individuals is very risky. A company wants to implement a Peer-to-Peer (P2P) lending system which allows for trusted third parties to provide guarantees for specific borrowers in exchange for a cut of the interest paid back by the borrower.
The following functionality should be encoded within:

- Individuals looking for loans can make a request for a loan by inputting the following details: the amount of Ether they would like to borrow, the date by which they promise to pay back the full amount, and the interest in Ether they promise to pay back upon paying back the full amount.
- Third-party guarantors can choose to provide a guarantee that the amount being requested by the borrower will be paid back to the lender by sending the amount of Ether being requested by the borrower. This amount is to be sent into the smart contract after individuals have made a request for loans, and before borrowers have granted a loan. The guarantor must also specify the amount of interest in Ether they will keep from the amount to be paid by the borrower. Once a guarantee is placed, the borrower must accept or reject the guarantee. Rejecting the guarantee will result in the guarantor's money being returned to the guarantor.
- Lenders should be able to view: (i) the current requests for loans; (ii) whether a guarantee has been placed on a specific request; (iii) the guarantor's address (this address could then be translated into a third party's identity off-chain); and (iv) the amount of interest in Ether that the lender will make once the full amount is paid (i.e. the full interest amount less the amount of interest that the guarantor will keep).
- A lender can then chose to provide the loan by sending the appropriate Ether along with identification of the specific loan request they are sending funds for. The funds should be sent to the borrower at this point.
- If a lender does not receive the full loan amount and the expected interest by the date agreed upon, then the lender can withdrawn the guarantee placed by the guarantor.
- If the borrower pays back the full loan amount and the full interest amount then: (i) the guarantor's funds should immediately be sent back to the guarantor along with the interest amount to be sent to the guarantor; and (ii) the lender should receive the full loan amount along with the amount of interest due to the lender.

You should ensure that users cannot abuse of the functionality in any way.
[Marks: 30]

Problem 2

Create an ERC20 token named 'LoanToken' which can be used as a replacement for using Ether, and modify the code from problem 1 to use the ERC20 token.
[Marks: 10]

Problem 3

You are to write Mocha/Chai tests that demonstrate that both the functionality implemented in problem 1 above holds, as well as that functionality which you would like to ensure is not possible is also tested.

[Marks: 20]

Problem 4

You are to provide a user interface that allows for the different parties to interact with the smart contract. In respect to this assignment, it is functionality of the user interface that is important, and not the styling of the interface.

[Marks: 10]

Problem 5

Write a report that describes: (i) the smart contracts, how the desired functionality is achieved, and how undesired functionality is protected against; (ii) the tests, how the tests ensures that the desired functionality is tested, and similarly how the tests ensure undesired functionality is tested against; (iii) and the user interface which interacts with the smart contracts (using web3).
The report should also highlight any security, bug, or other considerations that you took whilst writing your smart contracts, as well as any design patterns you have followed whilst implementing your system.

[Marks: 30]