DA51 Lab 8

Ethereum Decentralized Application (Dapp): Developing a Crowdfunding DApp

J. Gaber, gaber@utbm.fr, jaafar.gaber@gmail.com

Target set of this session:

- Develop a decentralized crowdfunding application (DApp) using Solidity for smart contract development and Truffle Suite for both **backend** and **frontend** integration.

The goal of this lab session is to develop a decentralized crowdfunding application on Ethereum. This application will enable users to create funding campaigns with specific deadlines, contribute Ether, and manage fund withdrawals. Additionally, you will implement a refund mechanism for campaigns that fail to reach their funding goal by the set deadline.

In other words, your task is to incorporate a feature that allows campaign creators to establish a deadline for their funding goal. Furthermore, you should implement a refund mechanism to return contributions if the campaign does not achieve its funding goal within the specified timeframe.

Tools Required:

- Node.js and NPM
- Truffle Suite
- Ganache (for local Ethereum blockchain simulation)
- MetaMask (Ethereum wallet)

Lab Tasks:

Setting up the Environment:

1) Install necessary tools and frameworks for blockchain development (see Lab 6).

Creating the Crowdfunding Contract

2) Initialize a Truffle Project: Create a new directory for your project and run truffle init.
   a) Write the Smart Contract:
   b) Create a new Solidity file in the contracts directory.
   c) Define a contract Crowdfunding with functions to start a campaign, contribute Ether, check funding status, and withdraw funds.
3) Compile the Contract: Run truffle compile.

Testing the Contract:

4) Write Test Cases:
   a) Create a new test file in the test directory.
   b) Write test cases using JavaScript to test the functionality of your contract.

5) Run Tests: Execute truffle test to run your tests.

Deploying the Contract

6) Configure Truffle for Deployment: Set up the truffle-config.js file to include network configuration for Ganache.
7) Deploy the Contract: Run truffle migrate to deploy your contract to the Ganache network.

Interacting with the Contract

8) Create a Frontend: Develop a simple frontend using HTML, CSS, and JavaScript.
9) Integrate with Web3: Use Web3.js to connect your frontend to your smart contract.
10) Test the DApp: Interact with your contract through the frontend and MetaMask.

Deliverables:

11) Source code of the Solidity contract.
12) Test cases and results.
13) Frontend code for the DApp.

Discussion and Conclusion:

14) A brief report documenting the development process, challenges faced, and how they were overcome.