DLOAN

Distributed Systems Project
Winter Session 2024/25

Group Members:

Alessandro Cassani
Matteo Incremona
Lorenzo Molinari
Caterina Sapuppo
Daniele Zucchi

December 10, 2024

1 Theme Description

DLOAN is a decentralized application (DApp) designed to revolutionize the traditional loan system by making it faster, more secure, and more accessible through blockchain technology and smart contracts. The conventional loan process is often lengthy, complex, and document-intensive. DLOAN simplifies this by creating a marketplace for borrowers and lenders.

In the DApp, users can act as either **borrowers**, who can decide the amount they want to borrow, the repayment period, and the interest rate, or **lenders**, who can review loan requests and choose to participate.

Once an agreement is reached between a borrower and a lender, a smart contract is created to manage the process. Smart contracts manage agreements by transferring funds, locking borrower collateral, and handling repayments transparently.

2 Implementation

2.1 Backend

The backend of the DLOAN DApp is implemented using **Solidity Smart Contracts**, which handles all aspects of loan management on the blockchain. In *LoanTypes.sol*, there is the data structure of the loan, which contains variables like the borrower's address, the lender's address, the loan amount, the duration, and the interest rate at which the loan has been issued. *LendingPlatform.sol* is the main contract that handles the core functionality of the DApp. Borrowers can create loan requests by specifying the loan amount, the duration, and the interest rate along with collateral (stake) equal to twice the loan amount. Lenders can browse active loan requests and provide funding. The smart contract ensures that all terms, such as interest rates and deadlines, are securely recorded. Borrowers can repay their loans at any time, including the loan amount and accrued interest, to fulfill the agreement. If a borrower fails to repay, the collateral is used to reimburse the lender, ensuring the lender is always protected. The deployment script (*deploy.js*) is written in **JavaScript** using the **Hardhat** framework. It deploys the smart contracts to the Ethereum blockchain. Logs the deployed contract addresses and ensures they are properly initialized for use in the DApp.

2.2 Frontend

The frontend of the DLOAN DApp is implemented using **React**. The frontend components allow users to interact with the platform, such as creating loan requests, viewing available loans, and managing repayments.

Ethers.js has been used for interacting with the Ethereum blockchain. It enables the frontend to connect to the blockchain network, read data from smart contracts (e.g.,

loan requests and active loans), and send transactions, such as funding loans or repaying them. The DApp requires users to have **Metamask** (a browser extension wallet for Ethereum) installed. Users must connect to the local blockchain network by adding it through Metamask's settings. This ensures a seamless connection between the frontend and the smart contracts deployed on the blockchain.

3 Challenges Faced

While developing the backend, several challenges emerged, mainly related to transaction management. A key challenge was to implement transaction handling within the smart contract. This included managing the flow of ETH between borrowers, lenders, and the contract itself. Another challenge included tracking the status of loan applications and active loans, updating them, and ensuring that all state changes were atomic and consistent. In the frontend, establishing reliable communication with the Application Binary Interface (ABI) of the deployed smart contracts has been challenging. Using Ethers.js, the frontend needed to interact with the smart contract functions effectively, ensuring data was properly formatted and transactions were executed without errors. Debugging these interactions, especially when mismatches arose between the frontend and the smart contracts, demanded extensive testing and optimization. Also, the fluctuating value of ETH/USD added an additional layer of complexity. The frontend integrated APIs calculate and display the Ethereum equivalent value both at loan origination and repayment. This functionality was critical for maintaining accurate and transparent repayment calculations, ensuring users had a clear understanding of their financial obligations.

4 Possible Future Improvements

A potential future enhancement could involve implementing a margin call mechanism. This would address the unlikely scenario where the borrower fails to repay and the value of ETH drops significantly, rendering the collateral insufficient to cover the lender's repayment. To enable this, the value of the collateral would need to be continuously monitored to ensure it doesn't fall below a specified threshold. We did not implement this because such a scenario is highly unlikely, requiring ETH to drop by 50% or more within just a few days for it to occur. Another potential implementation could involve allowing multiple lenders to participate in the same loan and connecting with a single borrower through a shared smart contract. Lenders would have the flexibility to choose the portion of the loan they wish to contribute, enabling the system to accommodate smaller lenders with limited portfolios.