This document presents the main architectural decisions for the solution

BCMLOAN (Blockchain Microloans)

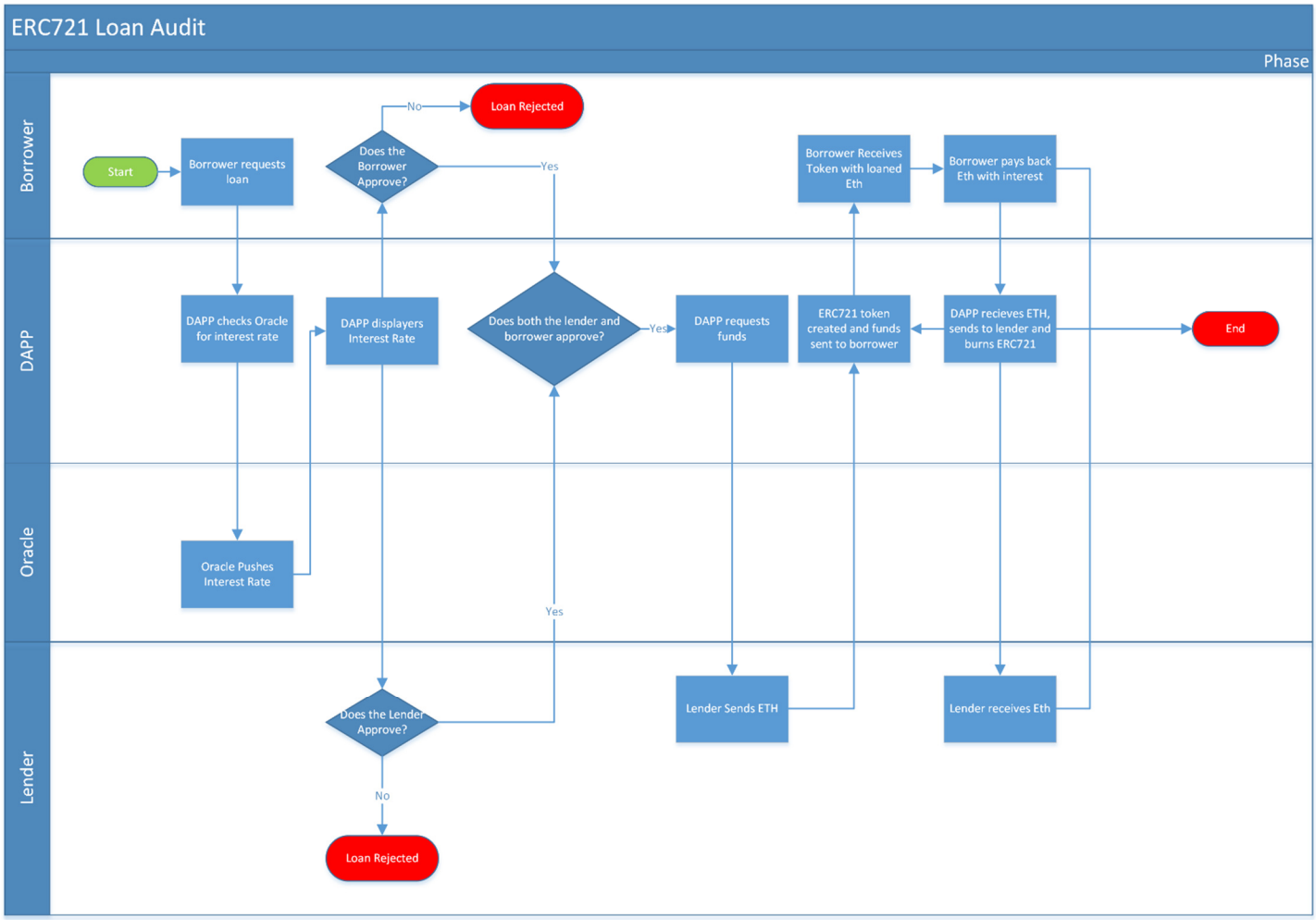
Team: Matthew Glezos, Shaun Lerner, Felipe Muriel Andrade, and Yeimi Pena

Goal

This document aims to describe the main architectural decisions for the solution BCMLOAN. These architectural decisions respond to the identified requirements and support the design and development of a minimum valuable product (MVP) that represents the required functionalities.

Flow Diagram

The flow diagram represents what the lifecycle of the Ethereum loan would be starting with the borrower of request, the borrower and the lender agreeing on the interest, the creation of the loan with the ERC721 token, and the burning of the token once the Ethereum plus interest has been returned to the lender. The stakeholders in this flow chart are the Borrower, Lender, DAPP, and Oracle.



Assumptions:

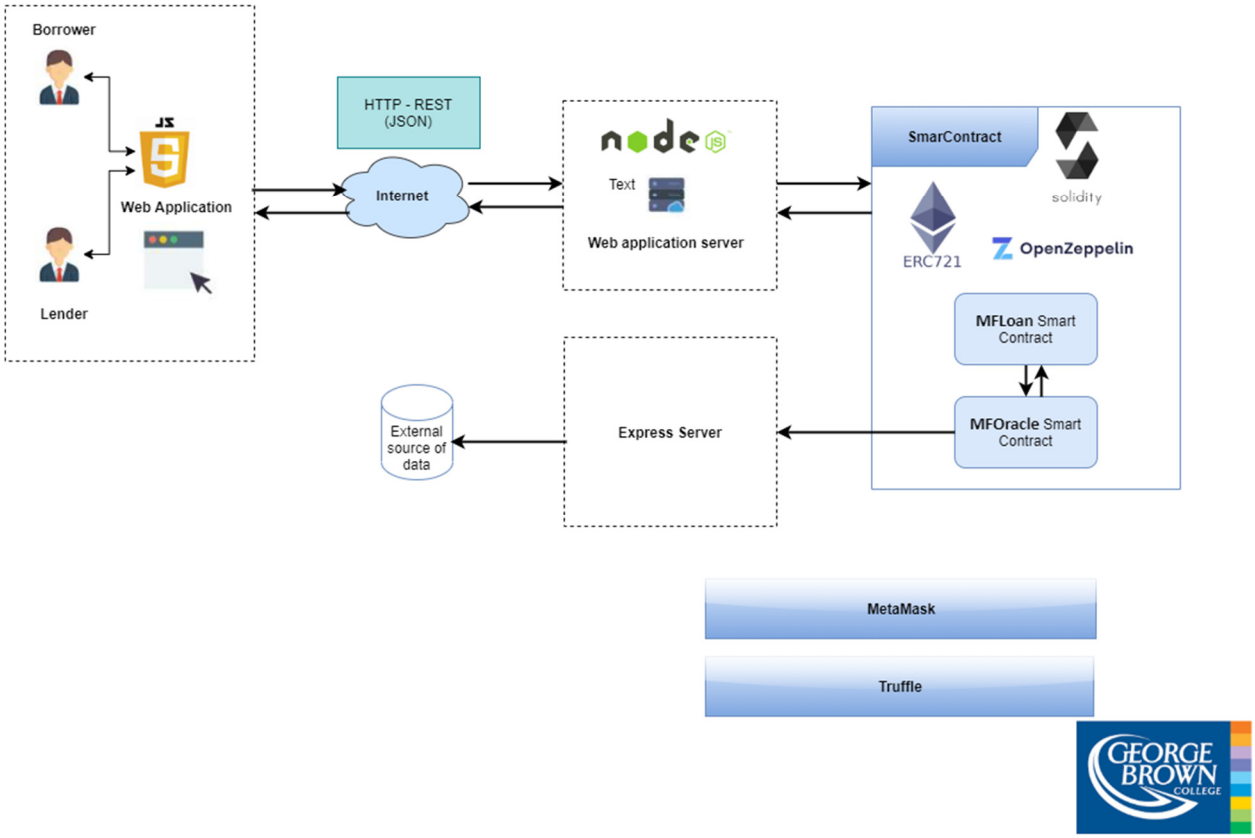
a. Lender and Borrower are both assumed to be in agreement for our purposes.
b. We're trusting the Zillow API that the interest rate is accurate.

c. Loans and payments will be in ETH (a conversion to CAD could happen at a future date).

Architecture of the Web application

The architecture represents the required components for BCMLOAN and their integration. The solution is a web application that contains the following components.

High Level Architecture Microfinance Diagram



1- Web user interface. It contains two web pages.
Request Loan. This web page supports a loan process.
   - A borrower requests a loan to a particular lender; the borrower includes the value of the loan and his address.
   - A lender approves and sends a token to the borrower for the amount of the loan.

Pay a Loan. This web supports the payment of a loan.
   - A borrower pays the loan to a lender including the rate interest.

2- Web Server. It executes the functions required by the web page and it is connected through an API with the smart contract platform.

3- Smart contract platform. It contains two smart contracts (MFLoan and MFOracle). The smart contracts were designed and developed in Remix and deployed in Ganache. For the implementation of ERC721 standard used in MFLoan, it was imported the interface implemented by Openzeppelin.

4- Oracle. The oracle contains a REST interface to consult data of the interest rate (external source of information) and the smart contract (MFOracle) to consult data from an external source.
5- MetaMask is employed to deploy the smart contracts in the local environment and Truffle is used to develop and execute the tests of the smart contracts' functionalities.

Architecture of Smart Contracts

As was described in the architecture of the application, BCMLOAN has two smart contracts that were designed considering the requirements and the design patterns learned during the classes. The objective and general architecture of each smart contract is described bellow.

- **MFLoan.** This contract contains the required structure to create a token that represents a loan. It also has the functions to approving and transfer a loan as well as approve and transfer the loan's payments.

  Structure

  Loan structure. This structure contains the information required for a loan.

  Id
  Borrower
  Lender
  dateBorrowed
  amountBorrowed
  interestRate
  amountOwedToPayback
  dueDate

  Functions

  - confirmNewLoan (address borrower, address lender, uint amount) returns (uint tokenId)
    - Validations of addresses, balance, and amount of loan
    - Mint a token
    - Transfer the token
  - paybackLoan(uint amount)
    - Validations of addresses, payback amount, due date, and balance.
    - Burn the token after the payment is performed.
  - getLoan (address _borrower). Return the information of the loan
  - getLoanOwedAmt (address _borrower). Return the amount that a borrower has to pay.

- **MFOracle.** This contract consults interest rates for loans from an external source. To consult the interest rates, MFLoan will be connected with MFOracle.

  Functions

  - getRate(). Return the interest rate.
  - setRate(uint256 rate). Update the interest rate after consult in the external source.

Data definition

Data definition refers to the required data for each contract.

- Data required in the **MFLoan** smart contract

Data	Type
Id	Uint
Borrower	Address
Lender	Address
dateBorrowed	uint
amountBorrowed	uint
interestRate	uint
amountOwedToPayback	uint
dueDate	uint

- Data required in the **MFOracle** smart contract.

Data	Type
iRate100	uint256

Technology required

For the development and deployment of BCMLOAN, we used the following technologies.

- Full stack (HTML, CSS, JavaScript, React)
- Truffle
- Remix – Solidity
- Ganache
- Github
- Metamask