PyBoLend

An ETH Lending and Borrowing Application

Greg Stevenson, Ben Eilers, Kyle Hagan, Jose Olasa

Executive Summary

PyBoLend is a proof of concept App, that allows for Lending and Borrowing of Crypto. The app is built on top of a Smart Contract and has a Streamlit front end for users to interact with the app. We additionally build in messaging feature using the Twilio API

PyBoLend has the following functionality:

- Lending
- Borrowing
- Repayment of Borrowed crypto
- Withdrawal of crypto
- Interest rate calculation and payment
- Sms notification system

With a successful proof of concept, the next steps for this app are to deploy it on a global test net and improve banking functions

Concept

- Create a decentralized lending and borrowing application.
- Inspiration from Aave protocol
- Transparent use of funds due to openness of blockchain

Approach

- Used Solidity to create underlying functionality of application
- Use only Ether for Lending and Borrowing for safer user experience.
- Front end of application uses Streamlit

Overview of App






Solidity
Smart Contract

Functions:
- Lend
- Borrow
- Withdraw
- Repay

Ganache

Test Blockchain:
- ETH accounts

Streamlit

Frontend:
- Primary interface with contract
- Interest Calculations
- Test functionality

Twilio

Messaging System:
- Provides users confirmation of actions

Demo

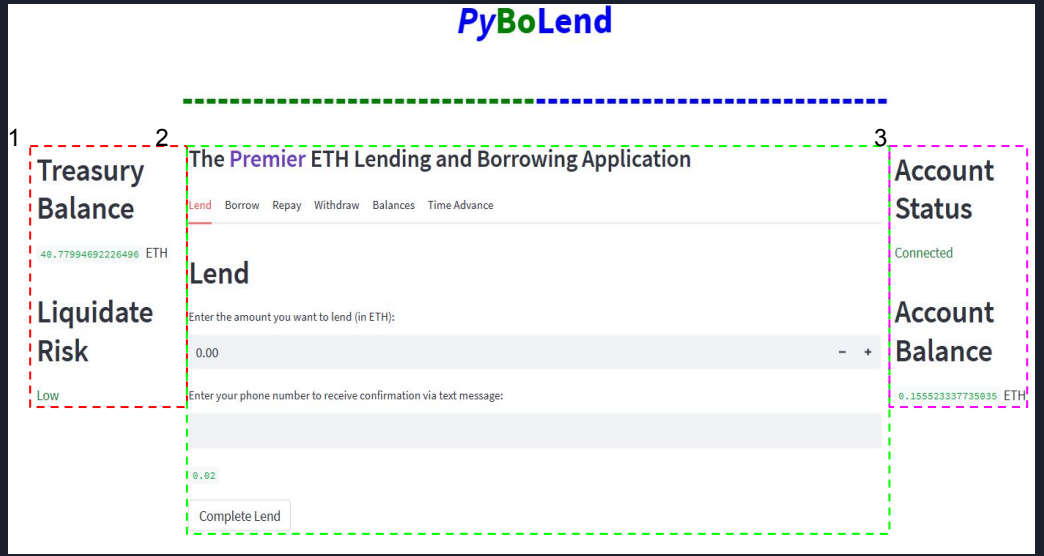
Smart Contract

- Lend function for sending Ether to the contract
- Borrow Function for user to get Ether with their loaned Ether as collateral
  - Must have Lend Balance to use Borrow function
  - Borrow balance can not exceed 80% of lend balance
- Repay for borrow balance to be repaid
- Withdraw
  - All borrows must be repaid before withdraw is allowed

Frontend-Interface

Directly connected to contract, the front-end provides a number of functional components:

1. Treasury details
2. Primary interface
   a. Lend
   b. Borrow
   c. Repay
   d. Withdraw
   e. Balance
   f. Time Advance (*illustrative only*)
3. User account details



Page specific screenshots available within the appendix.

Frontend-Interest

Interest is calculated in using the Aave Interest Rate Function

- Ut is the Ratio of the Borrow Balance to the Treasury balance
- Uoptimal is 0.8 for our test case and represents where we shift our focus from encouraging borrowing to encouraging repayment

We implemented the interest rate functions under the Advance time function for testing. On an advance time the following occurs

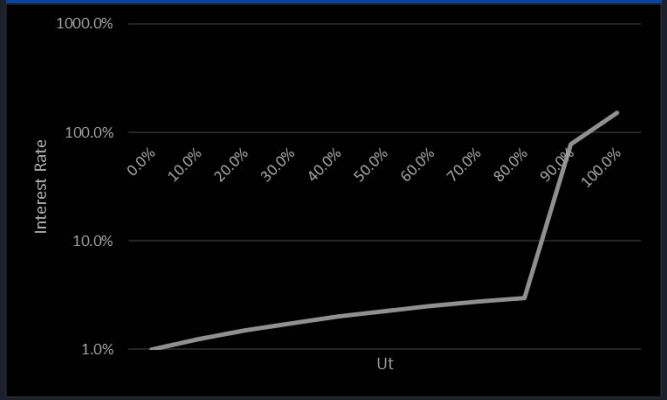
- Utilization rate is updated
- Interest on borrowed amounts are calculated and paid to contract and lender

Aave Interest Rate Function

$$if\ U \leq U_{optimal}: \quad R_t = R_0 + \frac{U_t}{U_{optimal}} R_{slope1}$$

$$if\ U > U_{optimal}: \quad R_t = R_0 + R_{slope1} + \frac{U_t - U_{optimal}}{1 - U_{optimal}} R_{slope2}$$

Interest Rate Output



Notification System

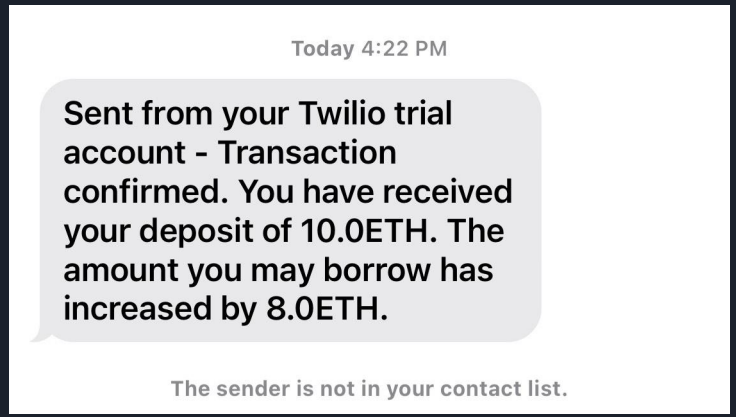
Notifications function as an extra layer of confirmation

~~SMTP~~ —> SMS

Allows users to receive confirmation via text message

Uses the twilio API and library

This code is stored in notification_manager.py



Conclusion

Goal

- Delivered functionality with lend, borrow, withdraw and repay transactions
- Successfully integrated our Streamlit platform with the pylend.sol lending contract
- Successfully at creating a decentralized lending and borrowing application

How we got there…

- Reviewed others in market
- We took an iterative approach
- Leveraged coding tools acquired throughout the program

Challenges

- Streamlit limitations
- Email notifications
- Floating point calculations in Solidity

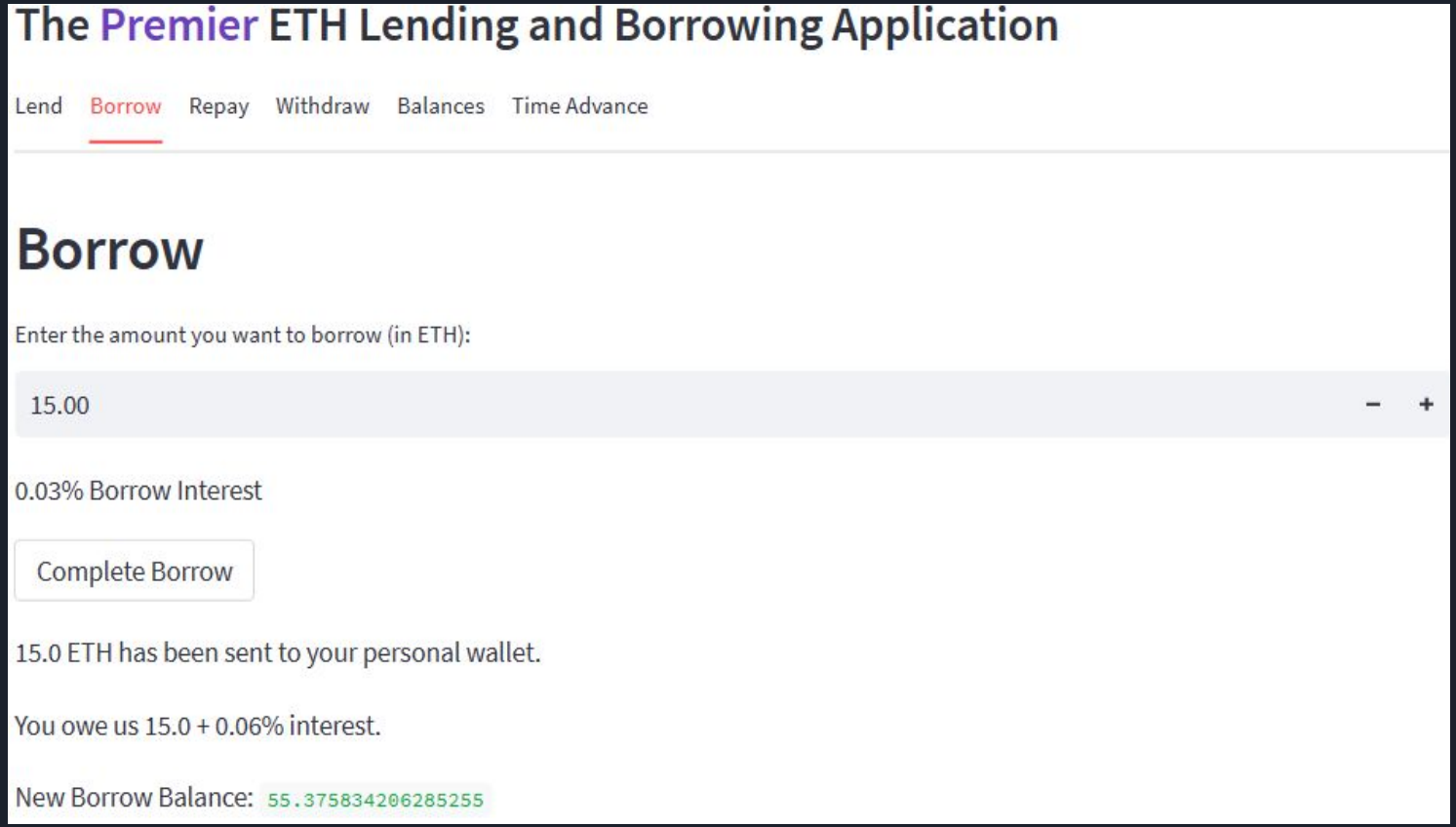
Next Steps

- Protocol ERC-20 that is given out to users
- Scale testing by moving from Ganache local to a Global testnet
- Program alternative form of collateralization
- Risk assessment to determine credit score proxy of borrowers
- Add in liquidation parameters into the pylend.sol smart contract
- Automatically advance time and calculate interest rate using the blockchain

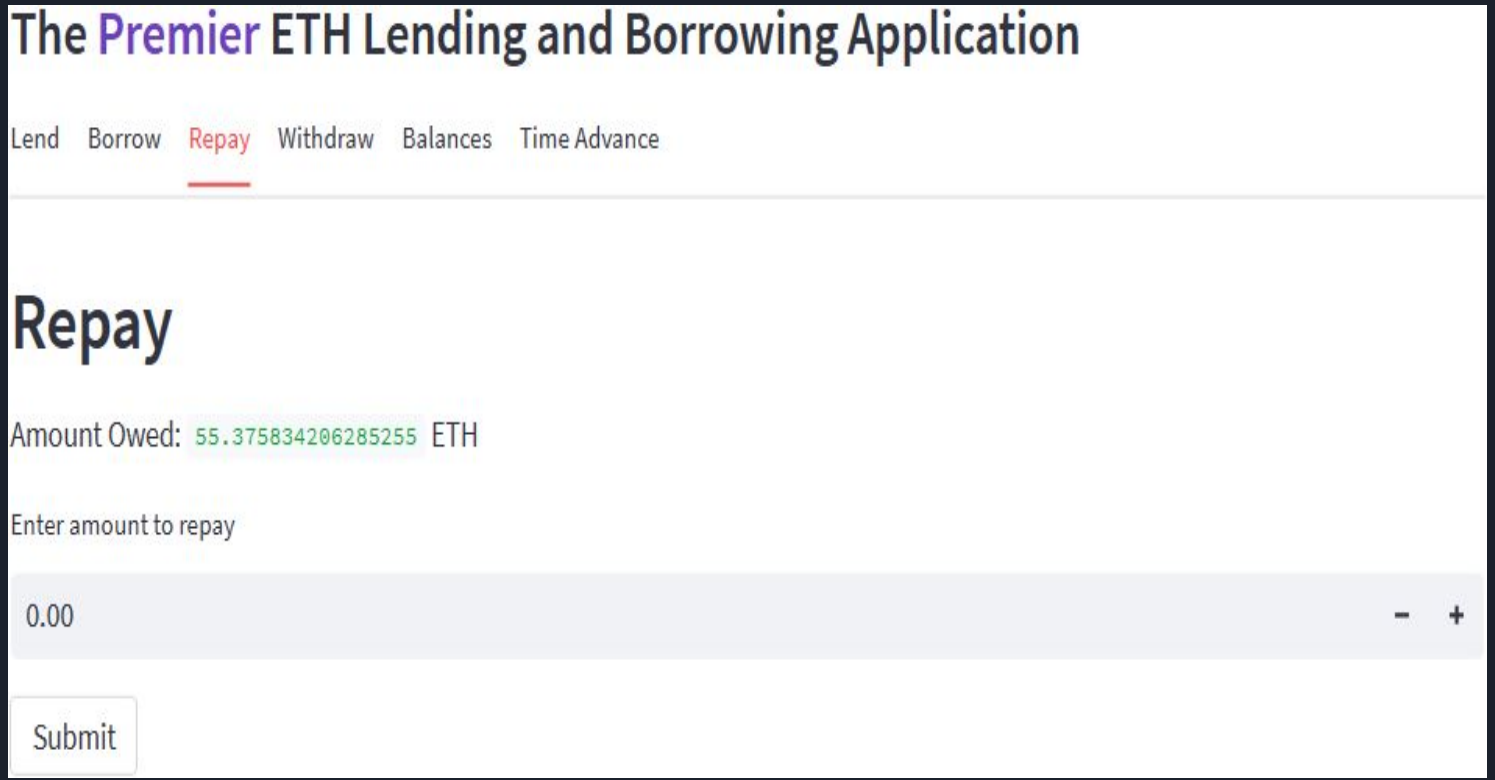
Questions?

Appendix

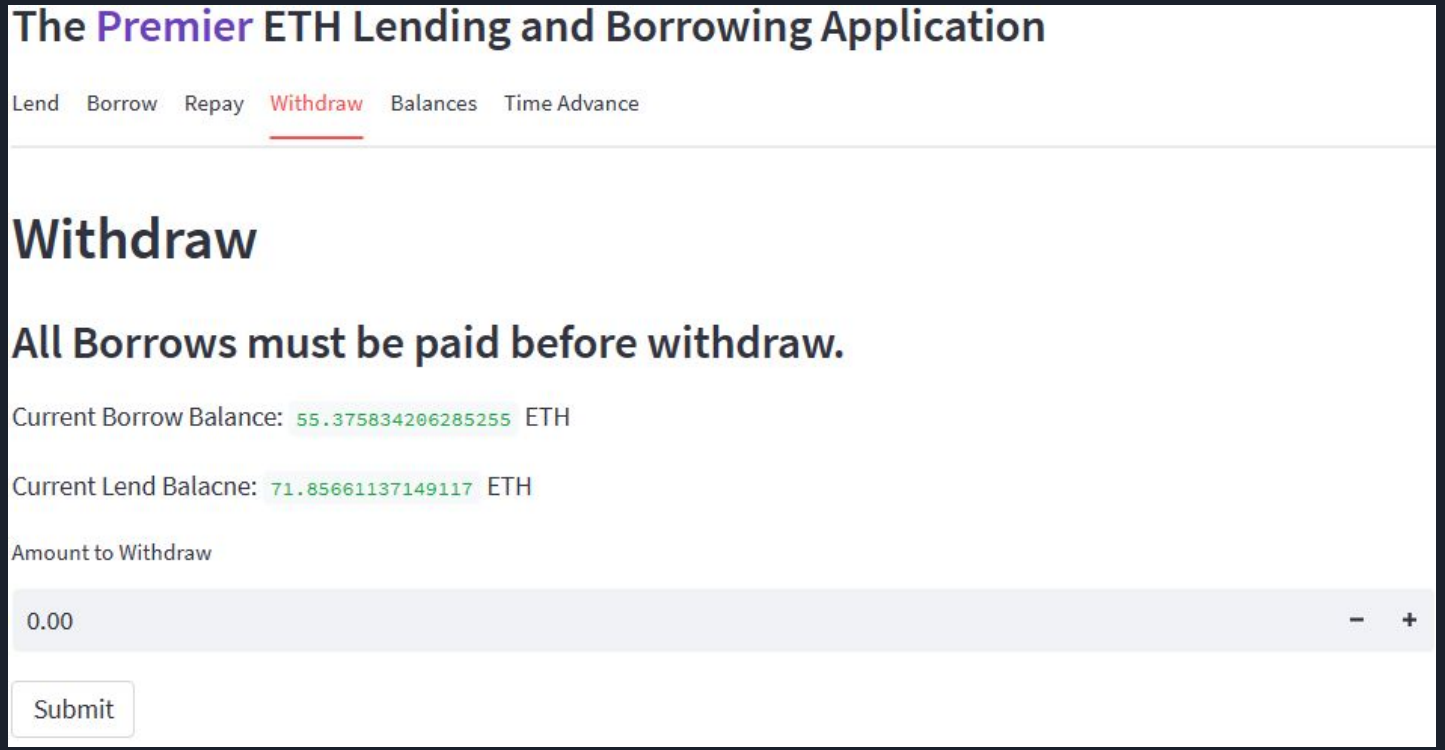
Borrow Experience



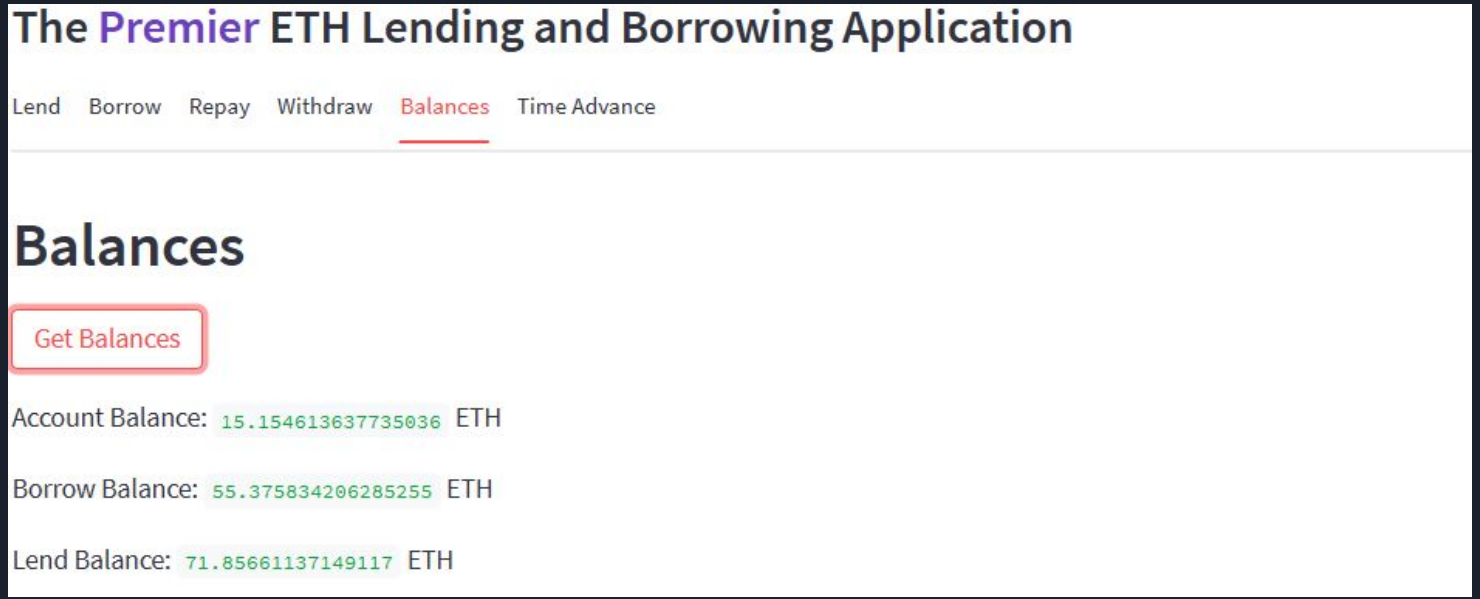
Repay Experience



Withdrawal Experience



Balance Inquiry Experience



Time Advance - Illustrative Experience Only