Escryptow: Design and Implementation of an E-commerce Dapp

Abstract

The online marketplace and escrow system show great potential to be incorporated with and benefit tremendously from emerging blockchain technology. Our project focus on creating a decentralized, blockchain-based online marketplace that provides similar online transaction and shopping experiences for users while guaranteeing high degrees of user anonymity and autonomy. Building upon the smart contract for the peer-to-peer transactions we created in Quarter 1, we established a ready-to-launch website as the frontend client to deliver to users and added more comprehensive functionalities to the contract, together creating a marketing system that enables automated transactions between multiple users. The system will have advantages such as high user anonymity, reliable transaction without third-party interventions, accessible transaction histories, and a usable interface that accommodates different actions. We divided our task into parallel groups to design and collectively develop the website. The frontends' focus was the webpage design, which should transform our contract into an actual usable product. We created UI and visual elements to facilitate navigation experiences and build the connection between the website, the underlying smart contract, and all external resources necessary for blockchain transactions. The backends' primary goal was to create the advanced functionalities of the transaction system and multiple utility functionalities for the website. We developed the smart contracts through Remix IDE and deployed the contracts on the Goerli Ethereum testnet to eventually run as a holistic system.

Introduction

Blockchain technologies have undergone rapid expansion in years with the potential to be incorporated into many current applications and frameworks of the IT industry. Conventional software and website services are facing challenges as Blockchain-based systems provide an unparalleled degree of information reliability and integrity, and allow services to be made available without a trusted third-party provider. One of these industries in which blockchain technology has shown immense potential to reform is the online marketplace and escrow system. Currently, online marketplace services, such as Amazon and eBay, are predominantly offered by third-party corporations that gain profits from high service fees and are vulnerable to user data breaches.

A decentralized, Ethereum-based escrow system on the other hand requires negligible operational fees (i.e. gas fees) instead of a substantial portion of the seller's profits. Trusted third-party firms such as eBay were critical prior to blockchain technology since they address the issue of trust between unknown buyers and sellers as a mediator in a digital marketplace. However, with the open-sourced smart contracts of blockchains, new escrow systems can effectively displace third-party administrators and proceed with transactions based on transparent, mutually agreed protocols.

To explore alternative escrow system approaches, we aim to establish a decentralized, Ethereum-based escrow marketplace by creating a smart contract that adapts online transactions with Blockchain technology and building a full-stack escrow system. Eventually, the escrow system is expected to prompt the users to connect to Ethereum, interact with the smart contract, and accomplish automated, peer-to-peer transactions with other users (e.g. buyers and sellers). We design our smart contracts to allow buyers and sellers to interact through actions such as posting products to sell, buying products, canceling transactions, and sending confirmations at each step. Our smart contract will securely hold funds until the item is received by the buyer and all conditions of the smart contract are met. All steps will be automated to eliminate external mediators. Thereafter, we will design the front end to build connections between the smart contract, the cryptocurrency wallets (e.g. MetaMask), and the user-learnable, actionable interface.

Specifically, we divided our task into two parallel portions, the front-end escrow client building, and the back-end smart contract development. The front end will focus on creating UI and visual elements to facilitate navigation experiences. We host our escrow system through an online website, where users will be prompted to connect to their MetaMask wallets and interact with the smart contract to join, confirm, and cancel transactions. The content of the backend is based on the smart contract from the previous quarter, in which the team worked to create the additional functionalities of the website to replicate the user actions available on conventional online marketplace websites nowadays and make our prototype competitive. The team also devoted time to testing edge cases and compatibility between the contract and the front-end website.

Methods

General Overview

Our objective is to establish an e-commerce platform that is powered by blockchain technology, specifically Ethereum, to enable completely decentralized buying and selling of products between sellers and buyers. To ensure that the sale of a product is successful, we require both parties to have the incentive to complete the purchase. As a decentralized platform, we lack a mediator to establish trust between unknown buyers and sellers in a digital marketplace. Therefore, to guarantee the effectiveness of a product purchase, both the seller and the buyer will have to deposit the value of the product price as escrow in the smart contract. This deposit will not be accessible to either party until the transaction is completed or one of the parties cancels the operation.

Consequently, when a seller sells a product, they will need to deposit the price of the product for sale, regardless of the quantity sold. On the other hand, the buyer will have to deposit twice the price of the product value, where one-half will be used as escrow, and the other half will be transferred to the seller as a payment method at the end of the operation. After the transaction is complete, the seller will receive twice the price of the product (their escrow and the buyer's payment), while the buyer will receive their escrow back and the product.

Furthermore, our development includes additional functionalities to provide both sellers and buyers with more freedom when making purchases, similar to those found on platforms like eBay. These functionalities include the ability for sellers and buyers to cancel transactions, either by rejecting the buyer or canceling the purchase, as well as the ability for sellers to retrieve a product from the platform. Additionally, we have added the capability for buyers to add reviews once the transaction has been completed.

To develop a decentralized platform, the logic of the code is the governing entity that establishes the rules for the platform. For this reason, certain functions can only be applied under specific conditions. Furthermore, the timeline for a purchase follows a strict structure to prevent fraudulent transactions. This

structure is illustrated in the following flowchart (see Figure 1), which shows the logic behind our platform and the conditions under which certain operations can be carried out.

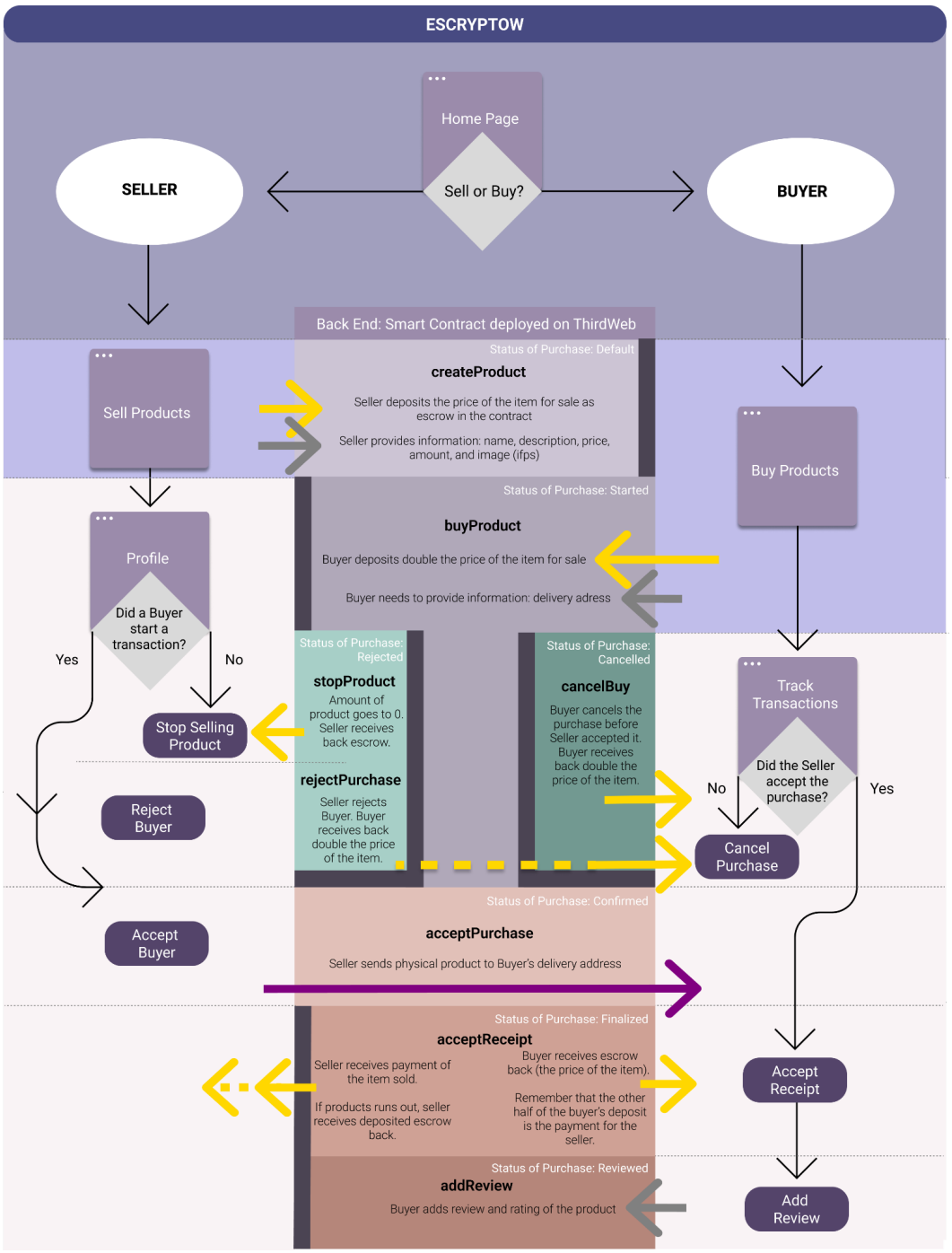

Figure 1: —

Backend

The objective of the backend of our project is to provide advanced functionalities for the transaction system and multiple utility features for the website. This was achieved by developing a smart contract using Solidity and the Remix IDE, which was later deployed on the Goerli Ethereum testnet using ThirdWeb. It is designed to enable buyers and sellers to interact through different actions such as posting products for

Initially, we started developing a contract that created a sub-contract for every transaction between seller and buyer, however, this was deemed ineffective due to high gas fees for each transaction. To solve this issue, we adapted the contract to allow multiple sellers and buyers to transact under the same single contract. We achieved this by utilizing mapping, arrays, and structs to keep track of all current and past transactions and states in our smart contract.

The smart contract includes several functions that can be divided into two categories: 'public' functions and 'public view' functions. Although both functions can be called both inside and outside the smart contract, the first type allows modifying the state of the contract, while the second type only allows reading the state of the contract. We use the first type of functions to ensure the proper functioning of our e-commerce platform, such as createProduct (seller), buyProduct (buyer), approvePurchase (seller), rejectPurchase (seller), approveReceipt (buyer), cancelBuy (buyer), stopProduct (seller), addRating (buyer). These functions are visible in the flowchart in Figure 1.

To ensure that both buyers and sellers can see the products for sale on our platform and confirm each step of the purchase process, we need view functions that allow access to the information of the contract, without any cost to the user (zero gas fees). These functions are: getAllProducts, getStatus, observeBuyers, getDeliveryAddress, and their names indicate the function they perform.

Finally, we have integrated the correct functioning of IFPS into our smart contract to be able to visualize the images of each product uploaded by the seller. We have made certain modifications to our contract to be able to store the content-addressed hash and ensure that all images are stored in a decentralized manner.

Frontend

The frontend of our project focuses on creating a webpage to display to users and allow users to connect to and interact with the underlying smart contract and escrow system through simple, discoverable interfaces. The website serves as a central hub that facilitates communication between the users and provides an easy way for users to connect to the various tools and applications (e.g. MetaMask, Etherscan, etc) essential for blockchain operations. To provide a fluent, comfortable navigation experience for the users, the frontend website design is mainly concerned with two demands: the stable, efficient connection to the contract and external tools, and the discoverability and aesthetic of the visual elements.

We started the design process of the wireframe by creating a wireframe using Figma, in which we build the general layout of the various pages and preserved dummy buttons to add functionalities in the future. We used wireframes mainly to test out the placement of the website logo, buttons, and the visual representation of text and products. The wireframe provided a framework for the actual website development. Then, the team programmed the website with Javascript and implemented interactivities with tailwind CSS, the React framework, the npm package, and the Vite build tool. We also utilized several crypto-specific frontend packages, including Thirdweb and ethers. The contract was also designed to interact with the frontend using the integrated InterPlanetary File System (IPFS). This allows the contract to store metadata files, such as long descriptions and image files, as IPFS URLs, and the metadata is retrieved from IPFS on the frontend. The frontend code is hosted on Hostinger, and we have used a paid dedicated gateway to accelerate the necessary IPFS functionalities. Version control for the frontend code is managed on GitHub.

The frontend website requires users to sign in to their MetaMask as login, for which buttons are provided as the utility navigation to guide the users to their browser extensions and sign in to the cryptocurrency wallet. The main navigation of the website after signing in comprises connection status to the wallet, a display of currently available products (like typical online shopping websites), and actionable buttons for current transactions.

Results

[add screenshots for some of the steps]

Our platform, named Escryptow, is available through the following link: ***link***. In this section, we provide a step-by-step explanation of how users can utilize our platform to buy or sell products. The guide is divided into two sections, Seller and Buyer, which explain the respective processes for selling and buying.

Requirements

1. Create an account using MetaMask. For more information on this process, the following website can be helpful: https://www.geeksforgeeks.org/how-to-install-and-use-metamask-on-google-chrome/
2. Obtain GoerliEth. As Goerli ETH is not a real cryptocurrency, we can obtain 0.2 Goerli ETH per day using the following website: https://goerlifaucet.com

Seller

1. Connect the MetaMask wallet to the platform.
2. Navigate to the "Sell Products" page.
3. Fill in all required fields for the product, including "Your Product," "Price," "Amount," "Description," and "Image." The "Deposit" field will be automatically filled based on the price.
4. Confirm the transaction, including the deposit as escrow and gas fees, using MetaMask. When the seller clicks on "Create a new product," MetaMask will automatically pop up. After the transaction is processed, the product will be visible to all users on the platform.
5. Navigate to the seller's profile.
6. Retrieve the products or wait for a buyer to purchase them.

7. When a buyer is interested in purchasing the product, it will appear under "Current Waiting Buyers." The seller can then confirm or reject the buyer. If the seller confirms the buyer, they should proceed to send the physical product to the buyer's specified delivery address.
8. To confirm or reject a buyer, the seller must include the specific buyer's address by copying and pasting it.
9. Regardless of whether the seller confirms or rejects the buyer, they must confirm the transaction on MetaMask.
10. If the seller confirms the buyer, they will receive payment once the buyer accepts the receipt. The seller will be able to retrieve their deposit once the product's amount reaches zero or if they decide to retrieve the product.

Buyer

1. Connect the MetaMask wallet to the platform.
2. Navigate to the “Buy Products” page.
3. Decide which product to buy and click on it.
4. Fill the blank with your current address and click on “Buy Product”. The buyer should take note of the price of the product since they will need to deposit double the amount. After the purchase, the buyer will receive half of the deposit back, and the other half will be sent to the seller.
5. Confirm the transaction using MetaMask.
6. Navigate to the “Track Transactions” page.
7. Refuse to purchase or wait for seller to accept the buyer and send the product.
8. When the buyer receives the product, they can click on "Approve Your Receipt". Once the seller accepts the buyer, the purchase cannot be rejected.
9. Confirm the transaction using MetaMask.
10. After the purchase is finalized, the buyer will have the possibility to review the product.

Discussion

[add]

Conclusion

[add]

Appendix

ThirdWeb framework to host current contract Store | Goerli | thirdweb

Contributions

To ensure the success of our decentralized E-commerce platform, we assigned specific tasks among our team members as follows:

William Li: Worked with Guangyu to develop the frontend. Created and refined wireframe of website. Worked with the backend team to accommodate functionalities of contract and designed visual elements of the website.

Antoni Liria Sala: Worked with Huy Trinh on the development of the backend (smart contract). Adapted the smart contract to accept multiple buyers and multiple sellers under the same single contract. Worked with the frontend team to ensure the functionality of certain view functions.

Huy Trinh: Worked with Antoni Liria on the development of the backend (smart contract). Worked on the development of the frontend, specifically on the connection between the smart contract and the interface to allow sellers and buyers to easily use our platform for purchase transactions. Already connected createProduct, buyProduct, and getAllProducts in frontend and backend which are ready to use.

Guangyu Yang: Worked with Will to develop the frontend, created templates for the website. Updated contract to accommodate IPFS functionalities and integrated IPFS with frontend.