CSE 526 :: Blockchain: App Development

DECENTRALIZED MARKET PLACE USING BLOCKCHAIN:

Report

Name: - Chinmay Swami [50290385]

Pratik Kubal [50290804]

Part 1: - Web Application

In this task we created a marketplace using python’s Django framework. Our marketplace consists of following functionalities: -

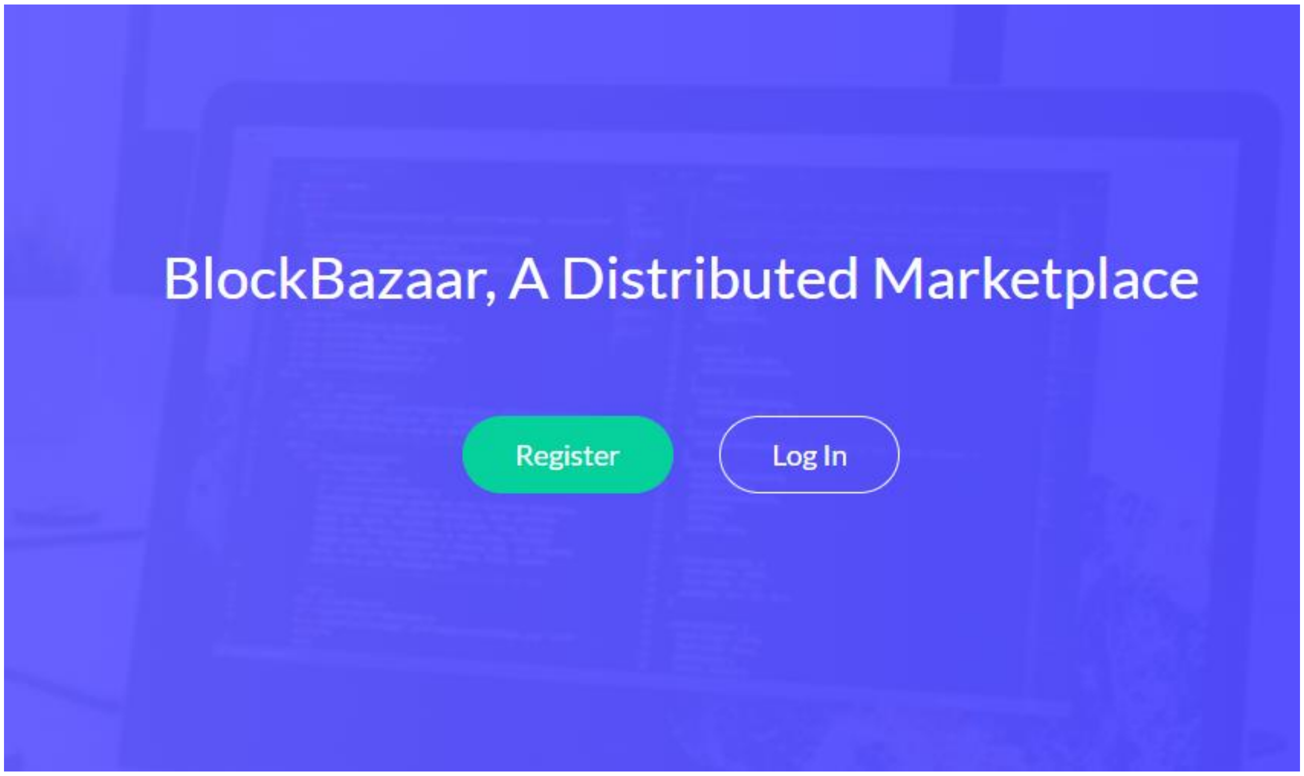
1. Login / Register
2. Sell Item
3. Buy Item

<u>HOW TO EXECUTE THE CODE:-</u>

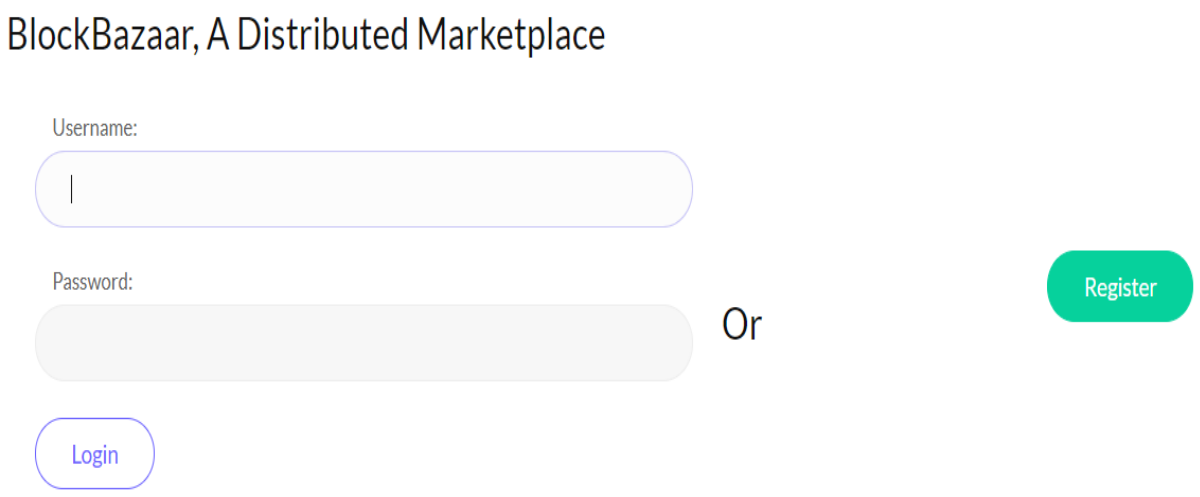
1. From CMD/Terminal browse to the MarketPlace_final folder.
2. Type python manage.py runserver
3. Type 127.0.0.1:8000 in the browser address bar

<u>HOW USE THE WEB APP:-</u>

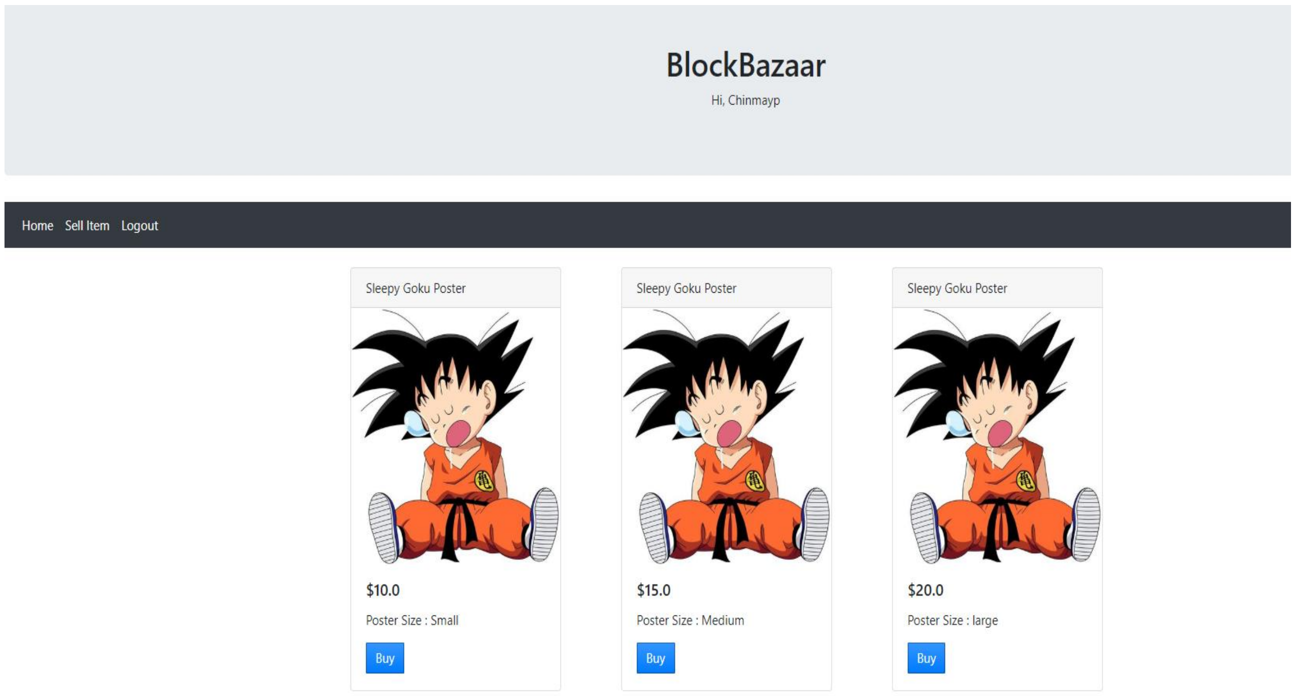
1. Login or Register



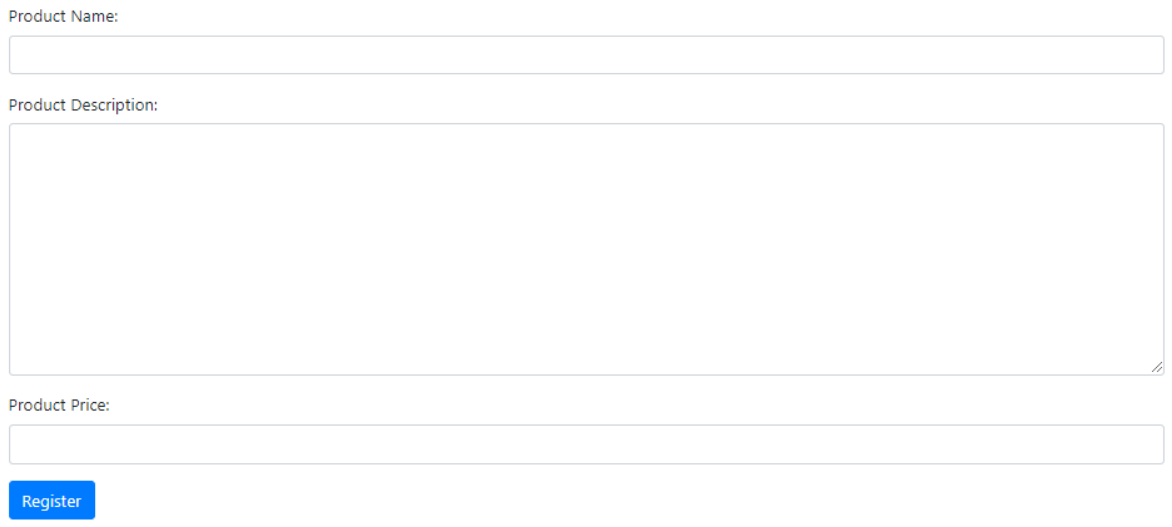
2. Login with credentials or sign up for a new account



3. This the page you will land where all the items available for buying will be displayed.



4. To sell product click on Sell Item. Enter the product details and click on register.



<u>DATABASE ORGANIZATION:-</u>

1. For User data we are using a predefined model provided by Django.
2. For storing details pertaining to products up for sale we have created table called Products.
3. For Storing transaction related data we have create another table called Transactions.

<u>USE CASES:-</u>

<u>USE CASE #1</u>

Name	User Registration
Goal	User Registered
Pre-condition(s)	None
Outline	The user who wants to register has to give their account number, name and password to register on the website. He can register as a whole-seller, trader and normal customer on the website.
Post condition(s)	The user gets registered on the website

USE CASE #2

Name	User Posting
Goal	User posting an Item / Items which he wants to sell
Pre-condition(s)	User registered and verified if belonging to special entities on the platform
Outline	The process requires an image / s of the object/s which he wants to sell to the platform. This image /s will be used to display an item in the marketplace. He also lists the price of the item
Post condition(s)	The item gets listed on the marketplace for a price

USE CASE #3

Name	User Buying.
Goal	User buys Items.
Pre-condition(s)	User registered on the platform.
Outline	The registered user selects the items which he wants to buy, he then selects the payment method by which he wants to pay for the item. After successful payment the item belongs to the user.
Post condition(s)	The items get displayed as sold.

*Template of Use cases borrowed from CSE542 - Software Engineering by Matthew Hertz

Task 2: - Smart Contract Implementations.

The smart Contract is present in the file named Marketplace.sol.

In this task we had to implement functions in smart contract using solidity programming language. Smart contracts basically automate the task of verification and carries out the activities of verification and validation which makes up for the disintermediation property. Smart contract serves as an interface between the blockchain and the Web App developed in part 1.

FOLLOWING FUNCTIONS WERE IMPLEMENTED.

Name	Description
viewAddressDetails	Its accepts product ID and displays the owner of the product.
_addProductToSell	It's function that allows users to add a product to sell. It's a private function and is called in another function.
Sell	This is a public function that calls the _ addProductToSell.
registerChairperson	This function allows for registration of the Chairperson.
_createUser	This is a private function to register a new user.
registerUser	This is a public function that calls the _ createUser function.
_unregisterUser	This is a private function to unregister a user.
Unregister	This is apublic function that calls the _unregisterUser.
_settlePayment	This is a private function that deducts and credits the product price from buyers account and to sellers account.
_buyProduct	This is a private function used to get buyer details.
Buy	This is a public function that calls _buyProduct.
viewUserDetails	This function displays the details of the user who's address is passed to it.

NOTE:- Functions viewUserDetails, viewAddressDetails are just for debugging purpose and won't go into the final version of smart contract that would be deployed on the blockchain.

WE HAVE CREATED FOLLOWING MODIFIERS:-

To add a level of security we created multiple modifiers that would allow access to the function execution only if the constraints are satisfied else its reverted.

Name	Constraints checked
chairmanNotYetSelected	This modifier checks if the activity of selection of chairman is performed or not. If not it allows access to the function registerChairperson.
isHeChairperson	This modifier checks if the user calling the function is the chair person or not.

	We implemented this modifier since we had some functions that only chair person should be capable of carrying out.
isUserAllowedToBuySell	This modifier checks if the user is a valid user so that he can post a product to sell.
isProductAvailableForSelling	This modifier checks if the product is available of buying. It accepts product ID as an argument.
isUserAlreadyRegistered	This modifier checks if the user that's being registered is already an active member. This prevents unnecessary transaction of registering form happening. It accepts the address of the user that is being registered as an argument.
isUserAlreadyUnRegistered	This modifier checks if the user that's being unregistered is already unregistered or not an active member. This prevents unnecessary transaction of registering form happening. It accepts the address of the user that is being registered as an argument.
doesUserHaveBalance	This modifier checks if the user who is trying to buy a product has enough balance to carry out the transaction.

MAPPINGS :-

Mappings are one of the ways of storing data on the blockchain in an organized fashion. It stores data in the form of key-> value pair.

Name	Description
addressToUser	mapping(address => user) addressToUse; We use this mapping to map the Ethereum address of the user to the user details saved in the User structure
productToUser	mapping(uint => address) productToUser; We use this mapping to map the productID with the Ethereum address of the user. This way we keep track of products that are sold by the user and once the product is bought we use this info to settle the payment.

We have created a Boolean variable called isChairpersonSelected which is private and is used to keep track of whether chair person is nominated or not. Once the chair person is nominated we set this variable to true and then the modifier chairmanNotYetSelected uses this variable to control access to the function registerChairperson.