SteeMart-A Marketplace powered by Cryptocurrency

Introduction

This is a proposal of the marketplace which is powered by the steem cryptocurrency.

What is Steem?

Steem is a cryptocurrency used to power the platform steemit. an incentivized blockchain social media platform. This just like other social media platforms where users create and publish the content, but here people get incentives for the work done. The block chain used is known as the steem blockchain, where steem is the fundamental unit. There are also few other units like steem power and steem dollars which are derived from the fundamental unit steem.

Where does this got started?

On July 4, 2016, Steemit, Inc., a company founded by Ned Scott and blockchain developer Dan Larimer, launched the social media platform Steemit as first application built upon Steem blockchain. Larimer had already developed a cryptocurrency exchange platform called BitShares in 2014.When Steem rolled out to users it translated to more than $1.2m with a $350m market capitalization. The currency saw an 1,800% increase in its price from $0.24 to $4.63 Steem has already paid more than $40m to its users, and there are almost one million accounts in the Steemit social network with one and a half million publications being made each month.

What is steem currency?

We have steem dollars and steem power as the other units derived from the steem

Steem dollars: These are the units given to the users as the reward for the content posted.1 steem dollar is equivalent to 1 USD. These also earn the 10% of interest annually.

Steem power: this symbolizes the power we have in the steemit platform, which you will get by the content posted and the votes for the content.

Based on the above details there are steem based Dapps which already existed, and few of them are

1.SteepShot
2.Utopian
3.Zappl
4.Dtube etc.,

SteeMart:

By considering the above advantages of steem we came up with the project idea to develop a steem based marketplace. Unlike other marketplaces the SteeMart will have some unique features which will be derived from the steem.

1.If we consider other market place the sellers will get money only when their product is purchased by the customer and there will be no incentives for the good reviews or the service provided, whereas in the SteeMart by considering the advantages of the steem platform there can be feature included like sellers getting rewards for the good products posted, best service provided and good reviews from the customers. Here we use the steem dollars as rewards.

2.This allows the customers to vote for the best product in any particular department out of all the listed products so that it will be helpful for the other customers as well as the voted customers will get rewards for their votes in the form of steem power.

3.This also allows the customers to get rewards if they post the useful reviews for the product they purchased.

Functionality of the SteeMart:

1.Users need to register with steem and get the approval before they can sell or buy things from the SteeMart.
2.Once after registering the login page leads to the home page
3.The application will have several features which allows users to manage the purchase history, payment methods and account settings as required.

Work flow of the application is as follows

Install the Server Dependencies
First, install express, the server framework we will be using, along with jsonwebtoken and body-parser by
running yarn add express jsonwebtoken body-parser --save. Next, create a server directory inside src, and
inside of server a index.js. This is where the server will run from.Also nodemon should be installed,
which automatically restarts the server after each change - otherwise we need to manually kill and restart
it. We can install nodemon globally by running yarn global add nodemon.
In src/server/index.js, add the following minimal Express application:

```
const express = require('express')
const bodyParser = require('body-parser')
const jwt = require('jsonwebtoken')
const app = express()
app.use(bodyParser.json())
app.post('/api/login', (req, res) => {
res.sendStatus(403)
})
app.listen(5000, () => console.log('Listening on port 5000.'))
```

We create a single /api/login endpoint. Run nodemon src/server/index.js in one terminal, and in another
use the follow curl command:

```
curl http://localhost:5000/api/login \
-X POST
```

Validating the User on the Server
Basically, the client will attempt to log in using some credentials, in our case a Steem username and
password. Once we authenticate the user, using Steem.js, the server will generate a "JSON Web Token"
using jsonwebtoken, which we installed earlier. We then send the token back to the client. On all
subsequent requests, the client will include the token. If the token is successfully validated by the
jsonwebtoken, the request is allowed. Our workflow will be as follows:
Client sends username/password to the server's api/login endpoint.
The server validates the login using Steem.js.
If the credentials are correct, create a token and send it back, with any extra information.
Now the client is "logged in". The client should resend the token each request to prove it is the
authenticated user, and not a hacker or other person impersonating the real user's identity.
Let's update the login endpoint to achieve the second step above. The explanation is as follows.

```
app.post('/api/login', async (req, res) => {
const { username, password } = req.body
// get posting public key
const account = await
steem.api.getAccountsAsync([ username ])

const pubKey =
account[0].posting.key_auths[0][0]
// for a given username/password combo,
// response contains { posting: 'private key',
postingPubkey: 'pub key' }
```

```
const { posting } =
steem.auth.getPrivateKeys(username,
password, ['posting'])
// See if the private key is a match to the
public key
const isValid =
steem.auth.wifIsValid(posting, pubKey)
if (isValid) {
res.json({ username })
} else {
res.sendStatus(403)
}
})
```

First, we extract the username and password from the request body. Next we use the getAccounts method
, which returns a bunch of public information about the account.
Next, we get the private keys based on the username and password combination. the getPrivateKeys
method still returns as public/private key combination. Now we validate the two keys using wifIsValid. If
the keys match, we return the original username. Otherwise, the original 403 forbidden. We still have not
touched on the JSON Web Token - that will come next, once we make sure this is working.
We test this using the curl as follows:

```
curl http://localhost:5000/api/login \
-X POST \
-d '{ "username": "your username",
"password": "your password" }' \
-H "Content-Type: application/json" \
```

Writing Tests for the Server Code
Now JWT authentication is working, let's finish up with some unit tests. First, a small tweak to
src/server/index.js is needed to run the server on a different port to the main application, and easily restart it between tests. Update the part of src/server/index.js where the server is created:

```
boot(port) {
return app.listen(port, () =>
console.log(`Listening on port ${port}.`))
}
if (!module.parent) {
boot(5000)
}
```

Literature survey

1.The concept and criticisms of steemit
https://poseidon01.ssrn.com/delivery.php?ID=384096025126005024003079118090092085050055046063064089024114107007093069067102085109050018006102038022052117113097079000005110009032000022033114017100114009008108106089035066020006006097110097096002024076106009 0

2.Sustainable growth and token economy design
https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&ved=2ahUKEwjyxNuY1tXhAhXlmuAKHeN4BesQFjACegQIBxAC&url=https%3A%2F%2Fwww.mdpi.com%2F2071-1050%2F11%2F1%2F167%2Fpdf&usg=AOvVaw0KZ9uFiLgB__D5EWukobmA

3.Personal analytics for societies and business with online platform
https://ieeexplore.ieee.org/document/8109887