SOFTWARE PASS: A DECENTRALIZED MARKETPLACE TO OWN SOFTWARE RIGHTS

BY,

NAME: G THULASI RAMAN

==

Abstract:

Software Pass is an innovative decentralized marketplace that aims to revolutionize the way software rights are managed and owned. In the current software industry, the distribution and licensing of software are typically centralized, which can lead to various challenges such as piracy, lack of transparency, and limited control for software developers and users. The Software Pass project seeks to address these issues by leveraging blockchain technology and smart contracts to create a secure, transparent, and user-centric marketplace for software ownership.

Benefits:

- Developers can protect their intellectual property and receive fair compensation for their software creations.
- Users gain access to a wider range of software options, easily purchase licenses, and ensure they are using genuine and legally acquired software.
- Enhanced security and transparency reduce the risk of malware and unauthorized usage.
- The elimination of intermediaries reduces costs for both developers and users.
- The platform fosters a vibrant community of software creators and consumers.

Key Features:

- Decentralized Ownership:

Software Pass allows developers to tokenize their software rights on the blockchain, effectively converting them into unique digital assets. These tokens represent ownership and usage rights of the software and can be traded securely within the marketplace.

- Smart Contracts:

Smart contracts play a vital role in ensuring secure and automated transactions. They govern the transfer of software ownership between parties, validate licenses, and automatically execute payments, thereby eliminating the need for intermediaries and reducing transaction costs.

- Transparent Licensing:

Through the use of blockchain, all software licenses and transactions are recorded on an immutable and transparent ledger. This ensures that software licenses are legitimate and accessible to all stakeholders, promoting trust and transparency within the marketplace.

- Enhanced Security:

Software piracy is a significant concern in the traditional software market. By utilizing blockchain's decentralized nature, Software Pass enhances security and reduces the risk of unauthorized distribution or replication of software. This feature provides peace of mind for both software developers and users.

- Incentive Mechanisms:

To encourage developers to participate in the platform and develop high-quality software, Software Pass implements incentive mechanisms such as rewards and reputation systems. Users can rate software and developers, and those with positive reviews can attract more buyers and establish themselves in the marketplace.

- Easy Integration:

Software Pass aims to be compatible with various software development environments and platforms, making it simple for developers to tokenize their software and list it on the marketplace. Users can seamlessly purchase and access software from a diverse range of developers and industries.

- Community Governance:

The Software Pass platform is community-driven, where decisions regarding platform upgrades, policies, and future developments are made through decentralized governance mechanisms. This ensures that the platform evolves according to the needs and desires of its user base.

Software Pass utilizes a combination of cutting-edge technologies to achieve its goals as a decentralized marketplace for owning software rights. The key technologies used in the project include:

1. Blockchain:

2. Smart Contracts : done using Solidity Programming

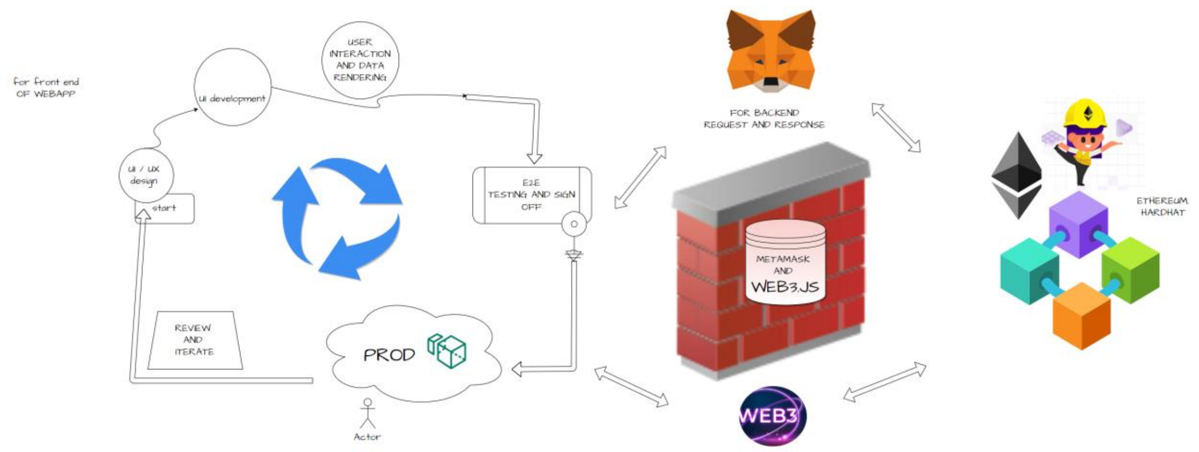
3. Ethereum : Blockchain Platform

4. Wallets: MetaMask Wallet

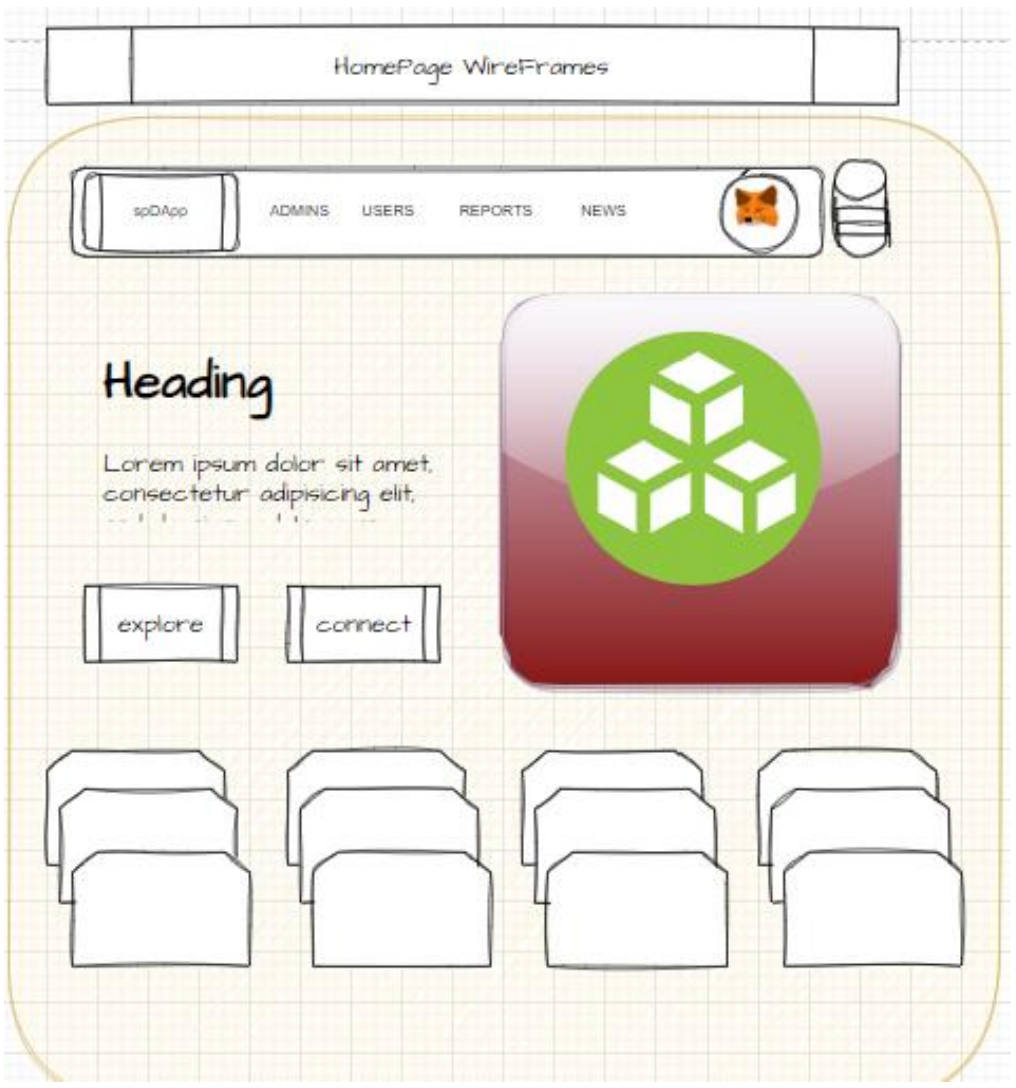
5. Web3.js

6. Front-end Library: React.js

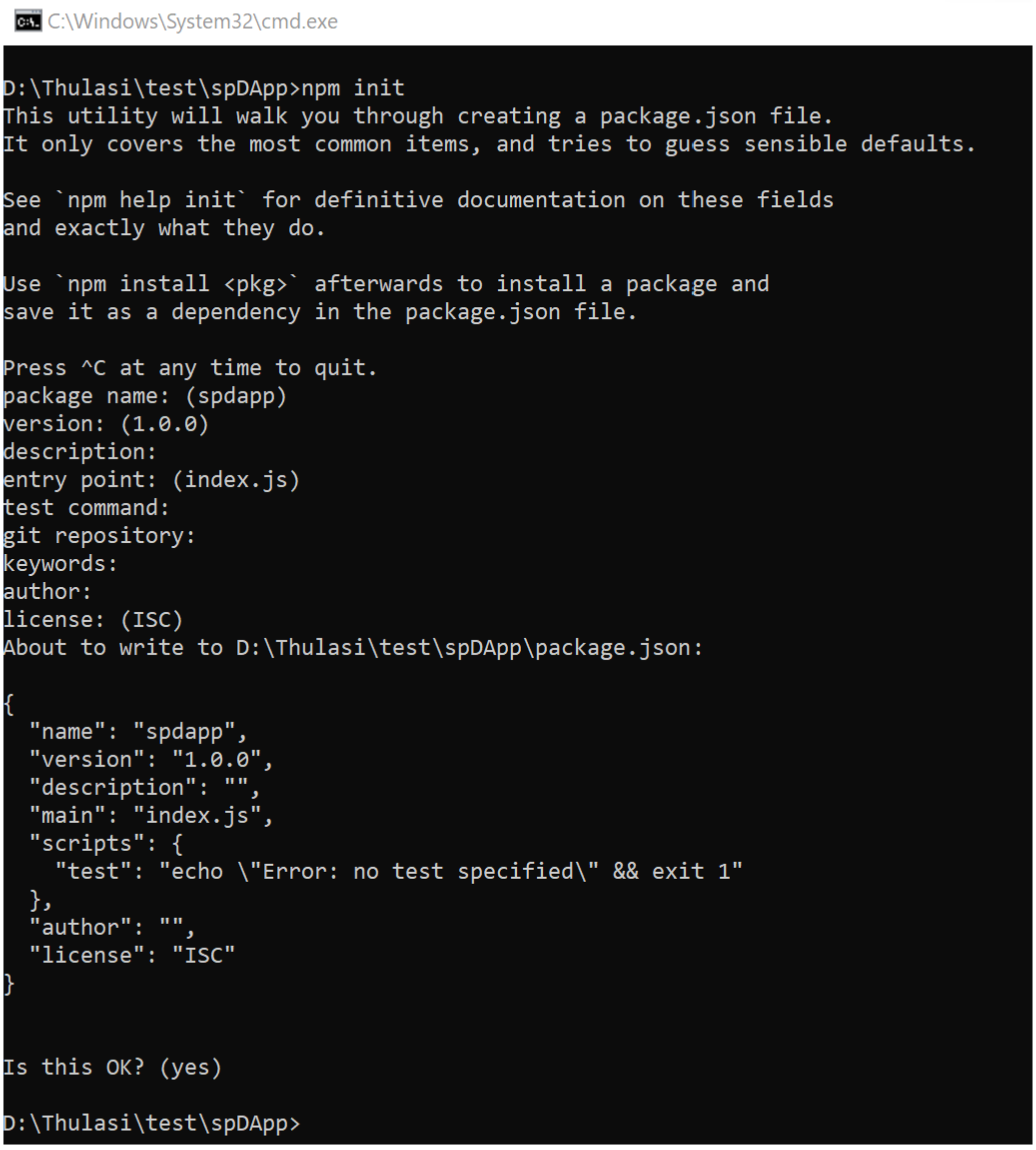
Project Setup sketch:



Front end wireframe:

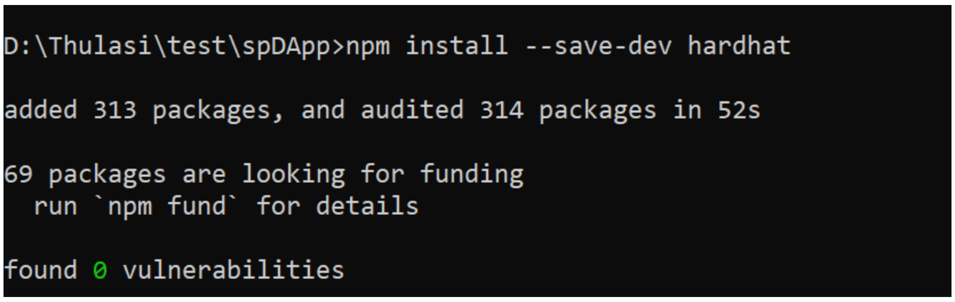


Installing Hardhat and environment setup

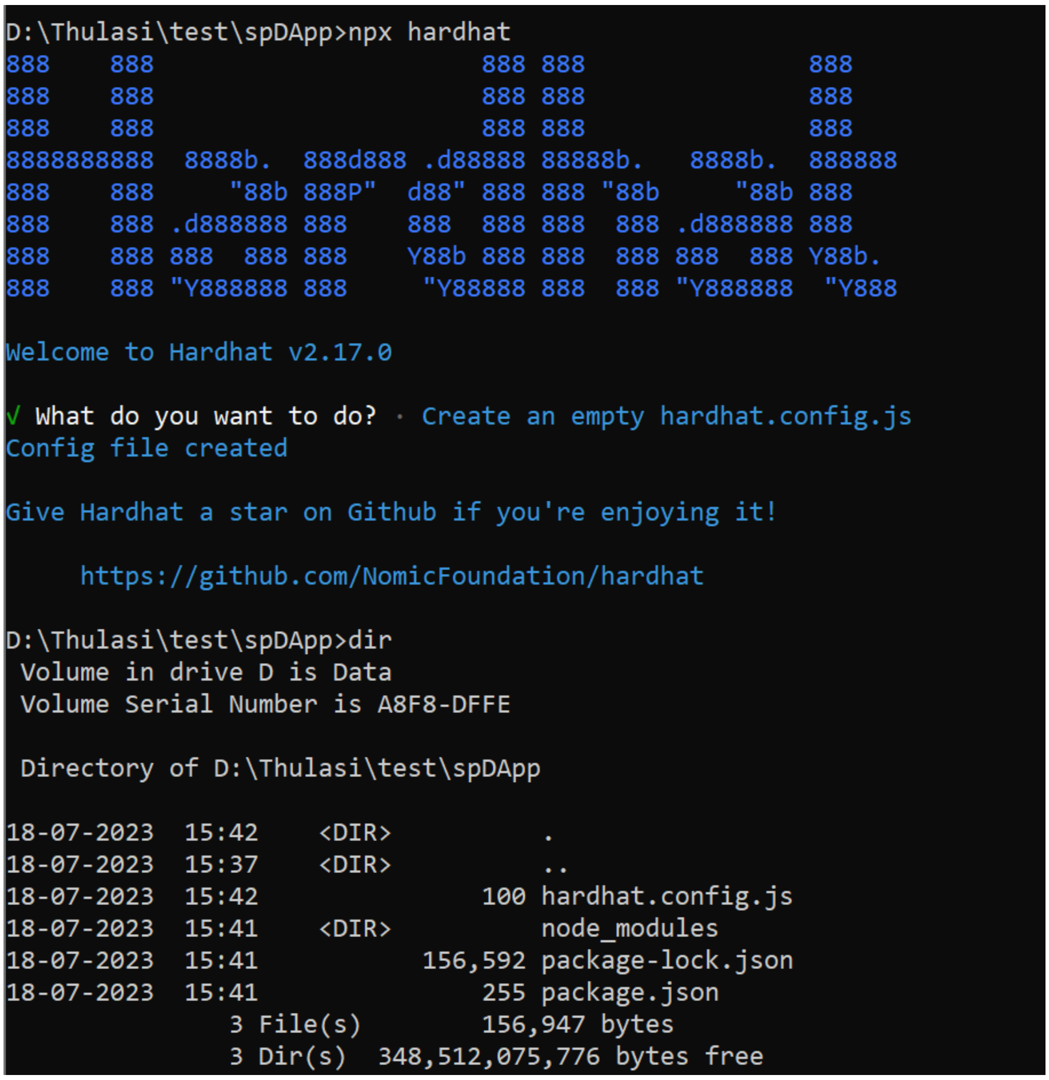

C:\Windows\System32\cmd.exe

```
D:\Thulasi\test\spDApp>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (spdapp)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to D:\Thulasi\test\spDApp\package.json:

{
  "name": "spdapp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes)

D:\Thulasi\test\spDApp>
```

Installing and initializing Hardhat



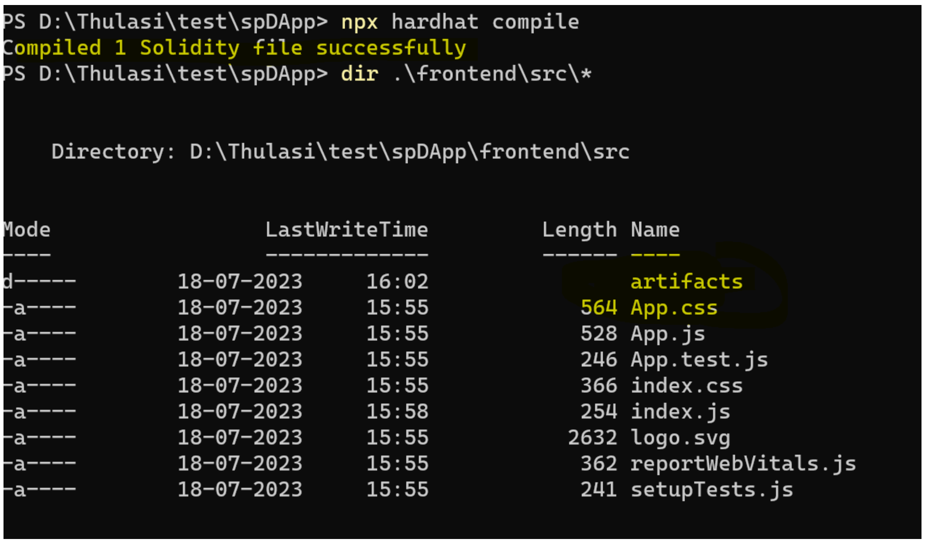
```
D:\Thulasi\test\spDApp>npm install --save-dev hardhat

added 313 packages, and audited 314 packages in 52s

69 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```



```
D:\Thulasi\test\spDApp>npx hardhat
888    888                      888 888               888
888    888                      888 888               888
888    888                      888 888               888
8888888888  8888b.  888d888 .d88888 88888b.   8888b.  888888
888    888     "88b 888P"  d88" 888 888 "88b     "88b 888
888    888 .d888888 888    888  888 888  888 .d888888 888
888    888 888  888 888    Y88b 888 888  888 888  888 Y88b.
888    888 "Y888888 888     "Y88888 888  888 "Y888888  "Y888

Welcome to Hardhat v2.17.0

√ What do you want to do? · Create an empty hardhat.config.js
Config file created

Give Hardhat a star on Github if you're enjoying it!

     https://github.com/NomicFoundation/hardhat

D:\Thulasi\test\spDApp>dir
 Volume in drive D is Data
 Volume Serial Number is A8F8-DFFE

 Directory of D:\Thulasi\test\spDApp

18-07-2023  15:42    <DIR>          .
18-07-2023  15:37    <DIR>          ..
18-07-2023  15:42               100 hardhat.config.js
18-07-2023  15:41    <DIR>          node_modules
18-07-2023  15:41           156,592 package-lock.json
18-07-2023  15:41               255 package.json
               3 File(s)        156,947 bytes
               3 Dir(s)  348,512,075,776 bytes free
```

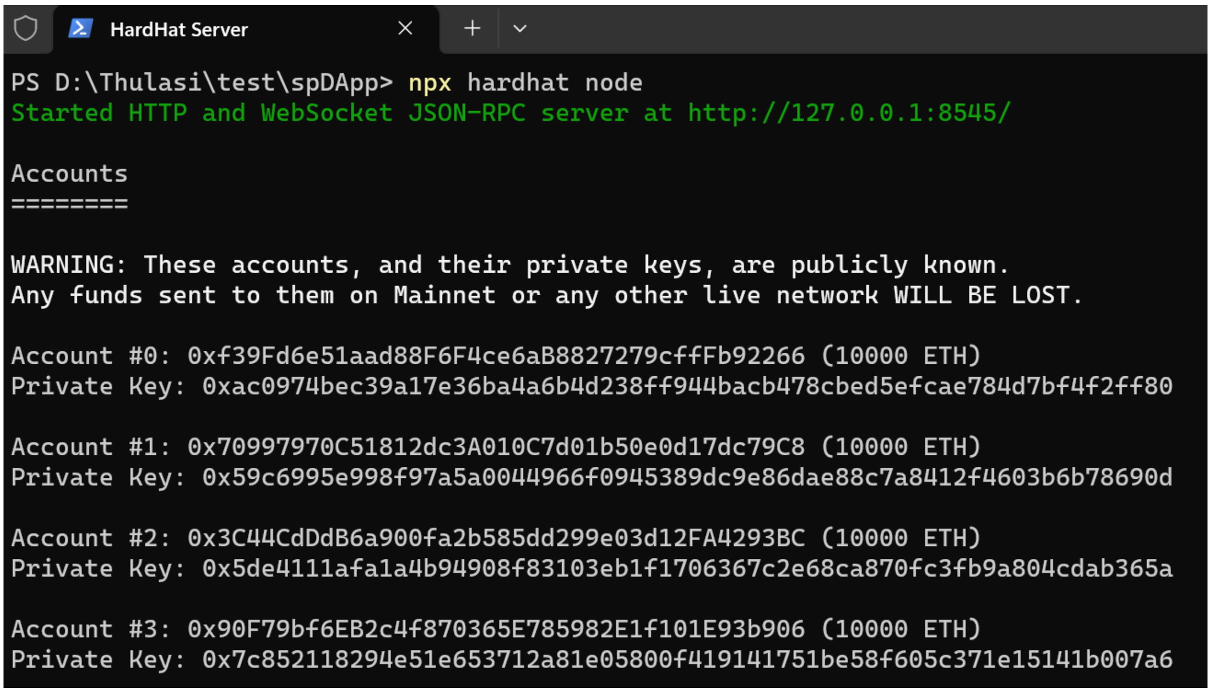
Once hardhat is initialized.

Upload the solidity Contract and compile



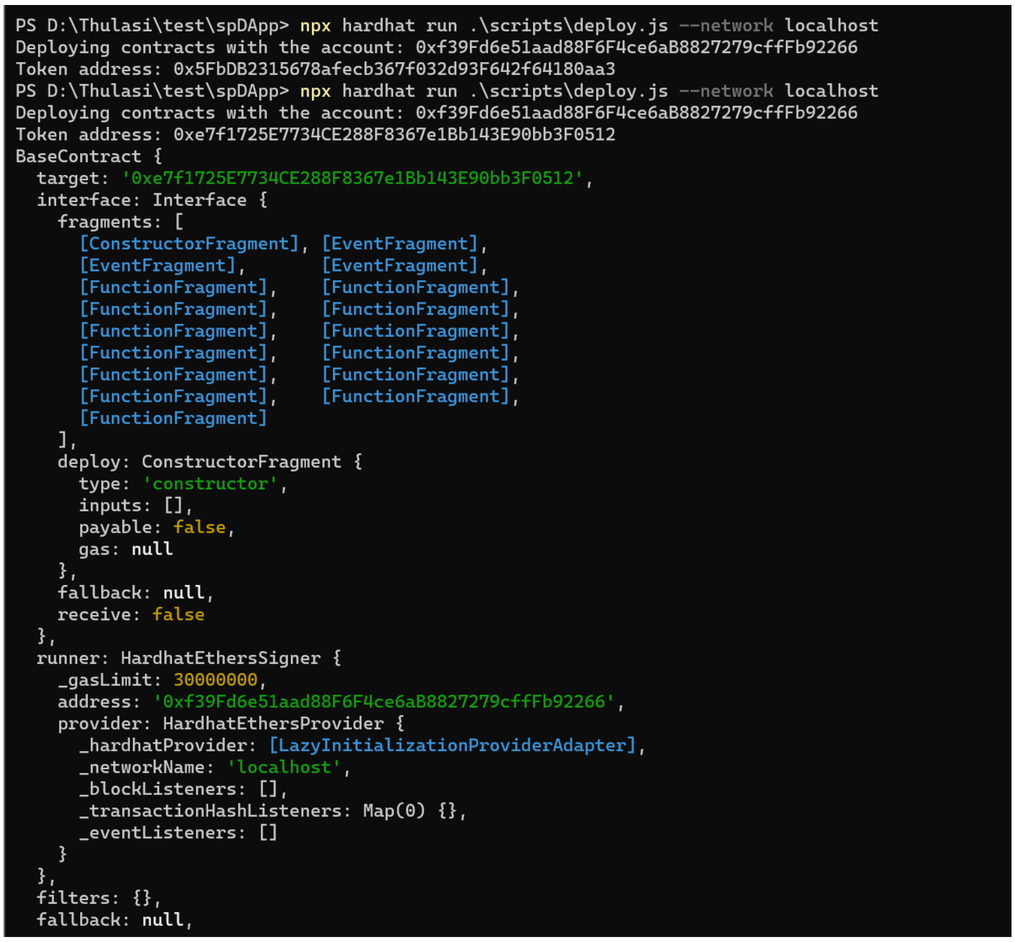
```
PS D:\Thulasi\test\spDApp> npx hardhat compile
Compiled 1 Solidity file successfully
PS D:\Thulasi\test\spDApp> dir .\frontend\src\*


    Directory: D:\Thulasi\test\spDApp\frontend\src


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        18-07-2023     16:02                artifacts
-a----        18-07-2023     15:55            564 App.css
-a----        18-07-2023     15:55            528 App.js
-a----        18-07-2023     15:55            246 App.test.js
-a----        18-07-2023     15:55            366 index.css
-a----        18-07-2023     15:58            254 index.js
-a----        18-07-2023     15:55           2632 logo.svg
-a----        18-07-2023     15:55            362 reportWebVitals.js
-a----        18-07-2023     15:55            241 setupTests.js
```

Start the local hardhat node (dev environment)



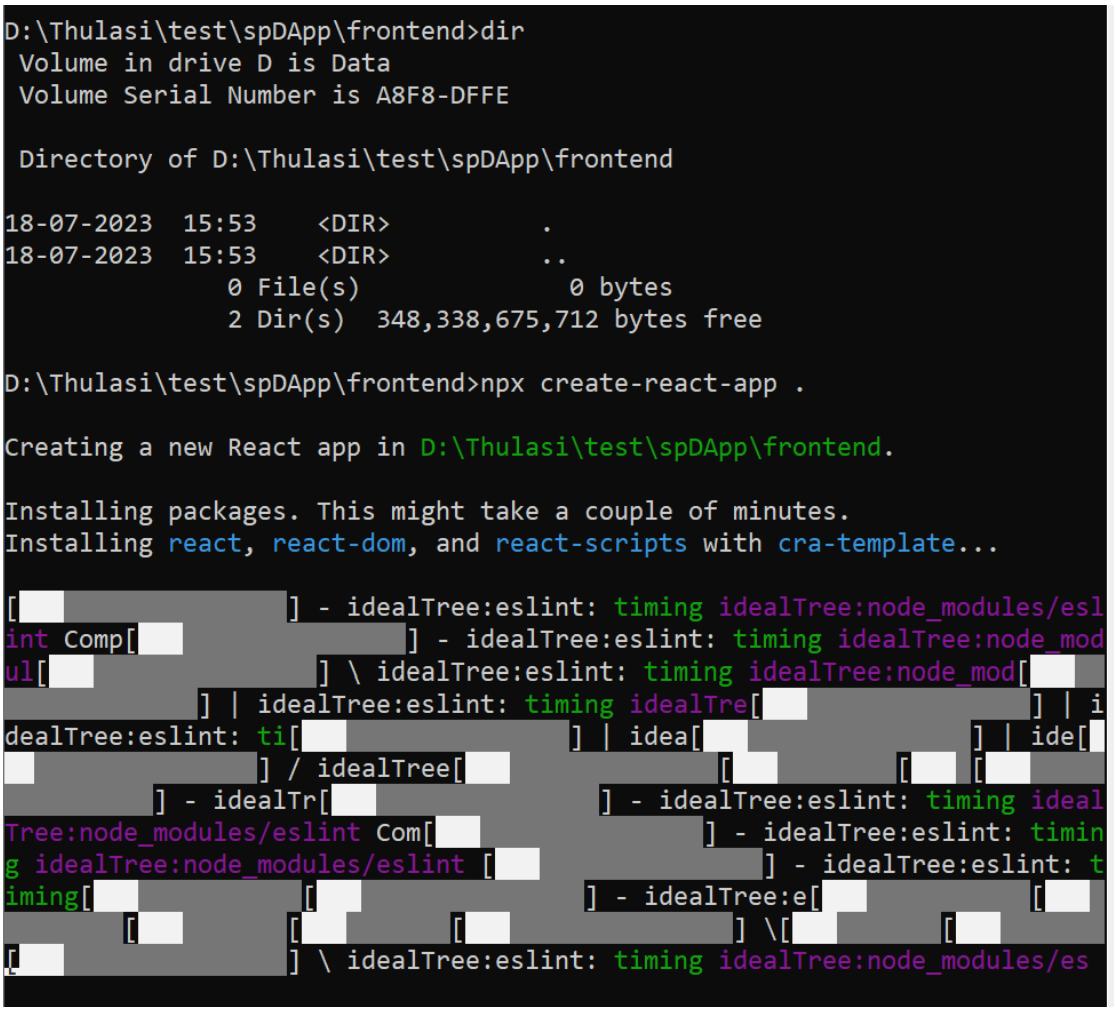
```
PS D:\Thulasi\test\spDApp> npx hardhat node
Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

Accounts
========

WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.

Account #0: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266 (10000 ETH)
Private Key: 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80

Account #1: 0x70997970C51812dc3A010C7d01b50e0d17dc79C8 (10000 ETH)
Private Key: 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d

Account #2: 0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC (10000 ETH)
Private Key: 0x5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a

Account #3: 0x90F79bf6EB2c4f870365E785982E1f101E93b906 (10000 ETH)
Private Key: 0x7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6
```

Deploy the script and get the contract instance

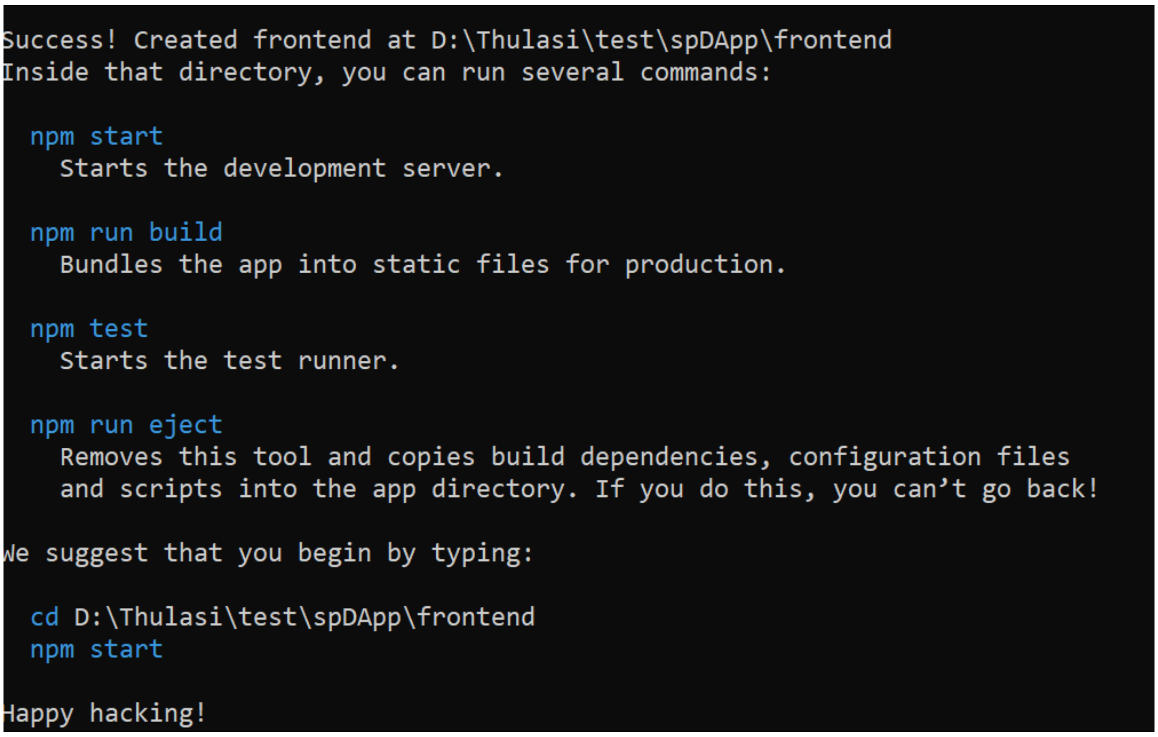


```
PS D:\Thulasi\test\spDApp> npx hardhat run .\scripts\deploy.js --network localhost
Deploying contracts with the account: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Token address: 0x5FbDB2315678afecb367f032d93F642f64180aa3
PS D:\Thulasi\test\spDApp> npx hardhat run .\scripts\deploy.js --network localhost
Deploying contracts with the account: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266
Token address: 0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512
BaseContract {
  target: '0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512',
  interface: Interface {
    fragments: [
      [ConstructorFragment], [EventFragment],
      [EventFragment],       [EventFragment],
      [FunctionFragment],    [FunctionFragment],
      [FunctionFragment],    [FunctionFragment],
      [FunctionFragment],    [FunctionFragment],
      [FunctionFragment],    [FunctionFragment],
      [FunctionFragment],    [FunctionFragment],
      [FunctionFragment],    [FunctionFragment],
      [FunctionFragment]
    ],
    deploy: ConstructorFragment {
      type: 'constructor',
      inputs: [],
      payable: false,
      gas: null
    },
    fallback: null,
    receive: false
  },
  runner: HardhatEthersSigner {
    _gasLimit: 30000000,
    address: '0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266',
    provider: HardhatEthersProvider {
      _hardhatProvider: [LazyInitializationProviderAdapter],
      _networkName: 'localhost',
      _blockListeners: [],
      _transactionHashListeners: Map(0) {},
      _eventListeners: []
    }
  },
  filters: {},
  fallback: null,
```

Setting up the Frontend

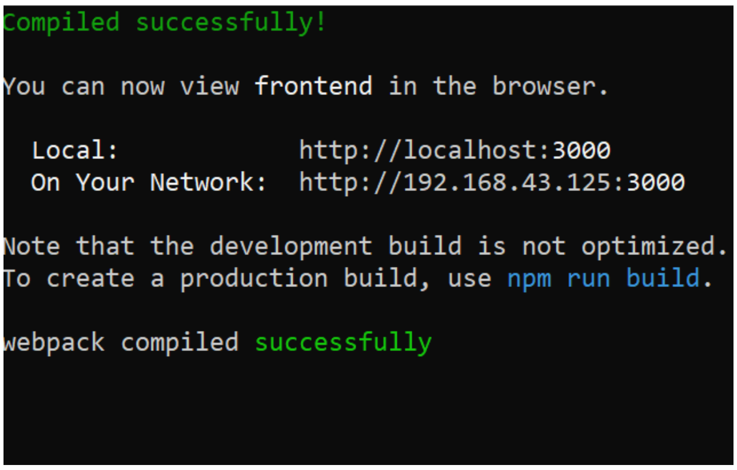


```
D:\Thulasi\test\spDApp\frontend>dir
 Volume in drive D is Data
 Volume Serial Number is A8F8-DFFE

 Directory of D:\Thulasi\test\spDApp\frontend

18-07-2023  15:53    <DIR>          .
18-07-2023  15:53    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  348,338,675,712 bytes free

D:\Thulasi\test\spDApp\frontend>npx create-react-app .

Creating a new React app in D:\Thulasi\test\spDApp\frontend.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

[               ] - idealTree:eslint: timing idealTree:node_modules/esl
int Comp[               ] - idealTree:eslint: timing idealTree:node_mod
ul[               ] \ idealTree:eslint: timing idealTree:node_mod[
          ] | idealTree:eslint: timing idealTre[               ] | i
dealTree:eslint: ti[               ] | idea[               ] | ide[
               ] / idealTree[               [        [    [
          ] - idealTr[               ] - idealTree:eslint: timing ideal
Tree:node_modules/eslint Com[               ] - idealTree:eslint: timin
g idealTree:node_modules/eslint [               ] - idealTree:eslint: t
iming[          [               ] - idealTree:e[               [
        [          [          [               ] \[          [
[               ] \ idealTree:eslint: timing idealTree:node_modules/es
```



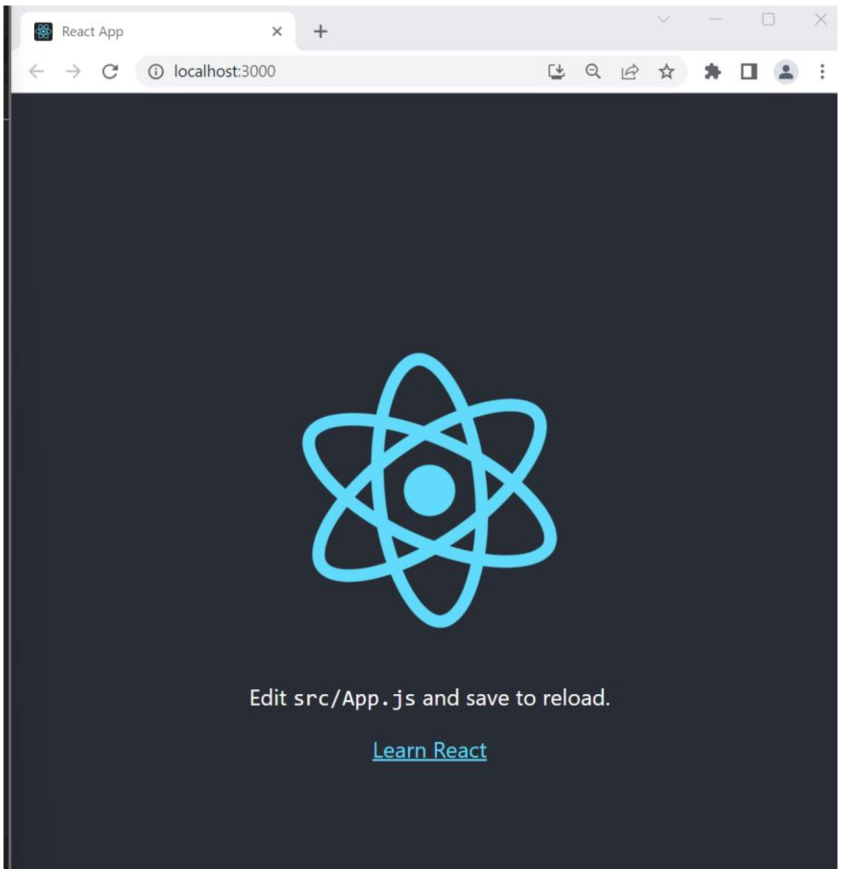
```
Success! Created frontend at D:\Thulasi\test\spDApp\frontend
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd D:\Thulasi\test\spDApp\frontend
  npm start

Happy hacking!
```

Verify the Webapp



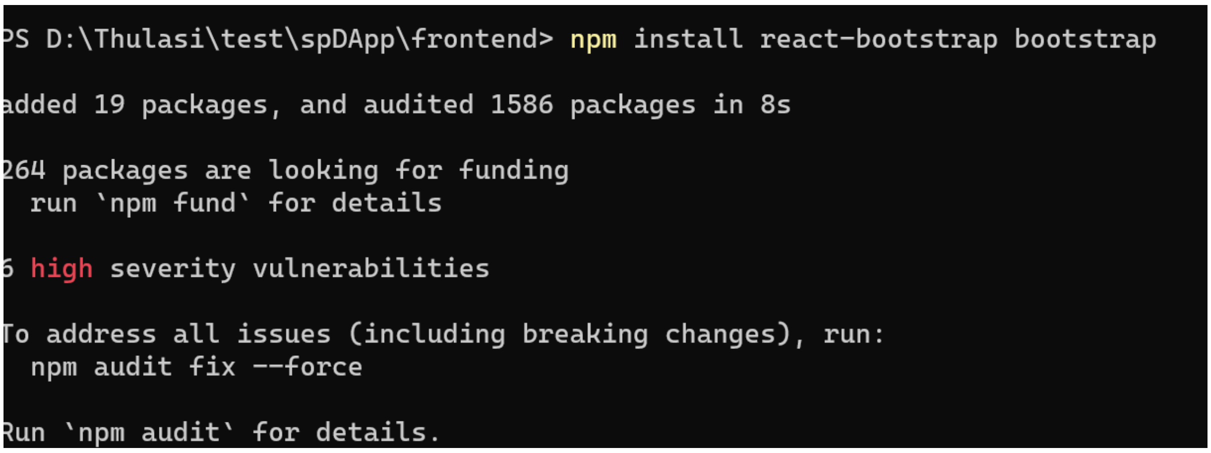
```
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.43.125:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

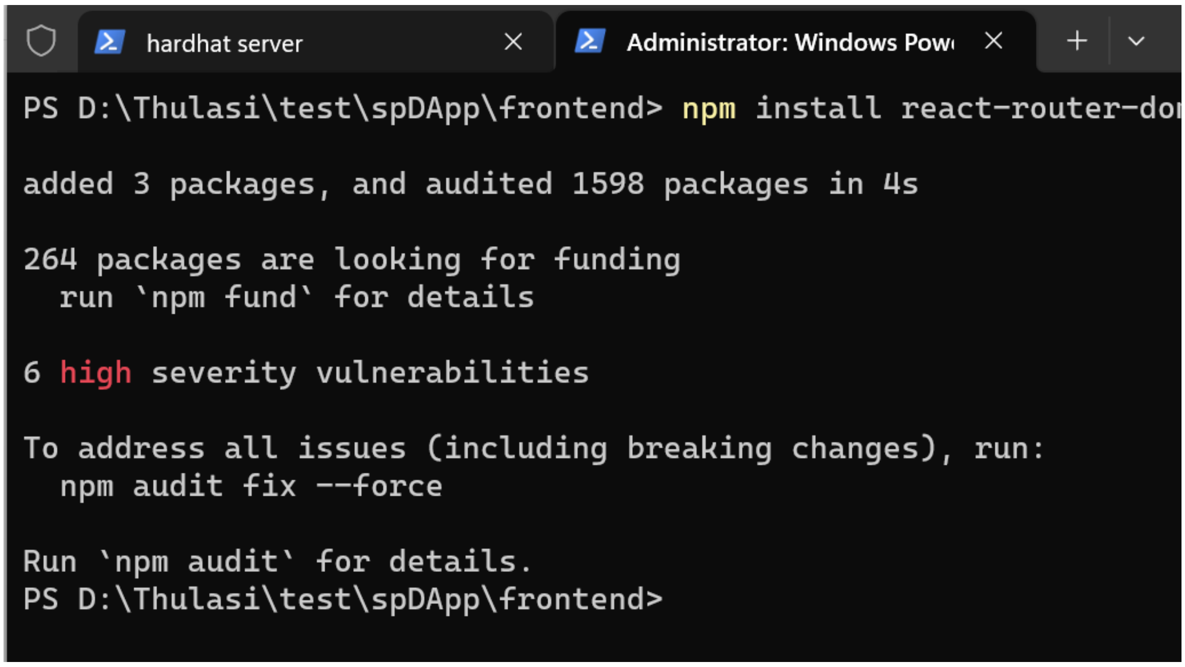
UI:



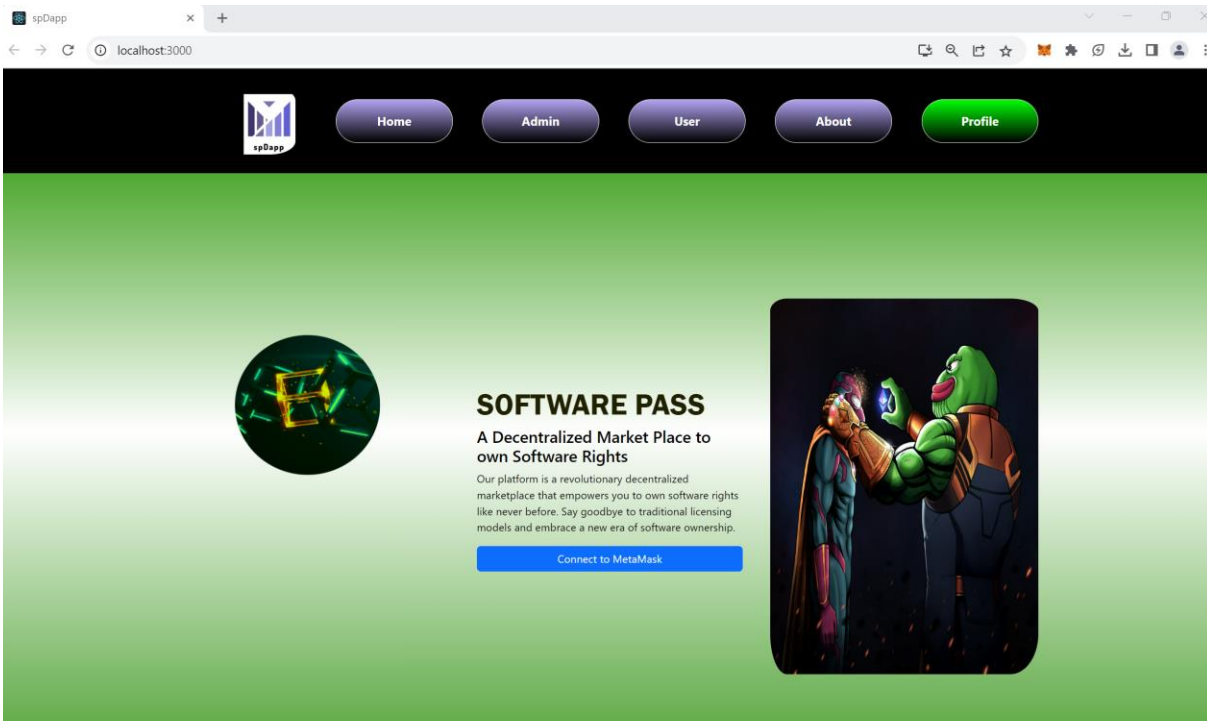
Edit src/App.js and save to reload.

Later with dependent libraries and front end mockups:


```
PS D:\Thulasi\test\spDApp\frontend> npm install react-bootstrap bootstrap

added 19 packages, and audited 1586 packages in 8s

264 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```



```
PS D:\Thulasi\test\spDApp\frontend> npm install react-router-do

added 3 packages, and audited 1598 packages in 4s

264 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
PS D:\Thulasi\test\spDApp\frontend>
```

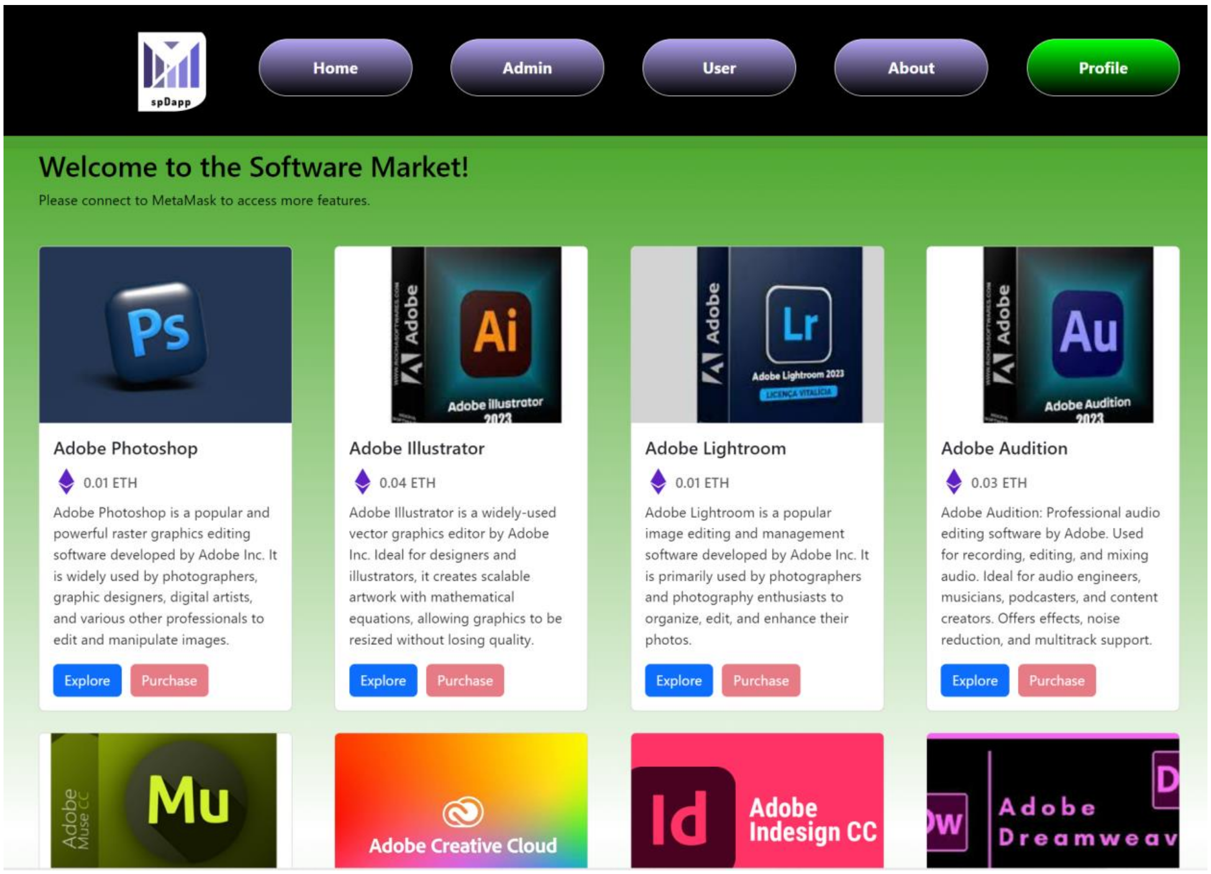


SOFTWARE PASS

A Decentralized Market Place to own Software Rights

Our platform is a revolutionary decentralized marketplace that empowers you to own software rights like never before. Say goodbye to traditional licensing models and embrace a new era of software ownership.

List of products available to view for generic users



Welcome to the Software Market!

Please connect to MetaMask to access more features.

Adobe Photoshop

0.01 ETH

Adobe Photoshop is a popular and powerful raster graphics editing software developed by Adobe Inc. It is widely used by photographers, graphic designers, digital artists, and various other professionals to edit and manipulate images.

Adobe Illustrator

0.04 ETH

Adobe Illustrator is a widely-used vector graphics editor by Adobe Inc. Ideal for designers and illustrators, it creates scalable artwork with mathematical equations, allowing graphics to be resized without losing quality.

Adobe Lightroom

0.01 ETH

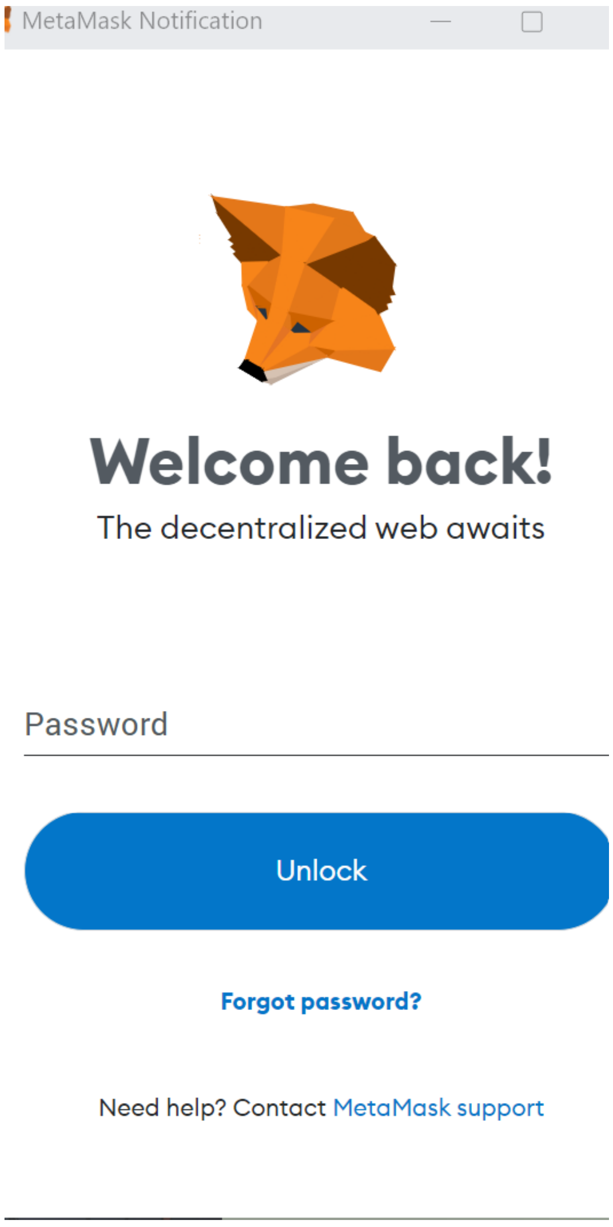
Adobe Lightroom is a popular image editing and management software developed by Adobe Inc. It is primarily used by photographers and photography enthusiasts to organize, edit, and enhance their photos.

Adobe Audition

0.03 ETH

Adobe Audition: Professional audio editing software by Adobe. Used for recording, editing, and mixing audio. Ideal for audio engineers, musicians, podcasters, and content creators. Offers effects, noise reduction, and multitrack support.

When connecting to wallet and account setup


MetaMask Notification

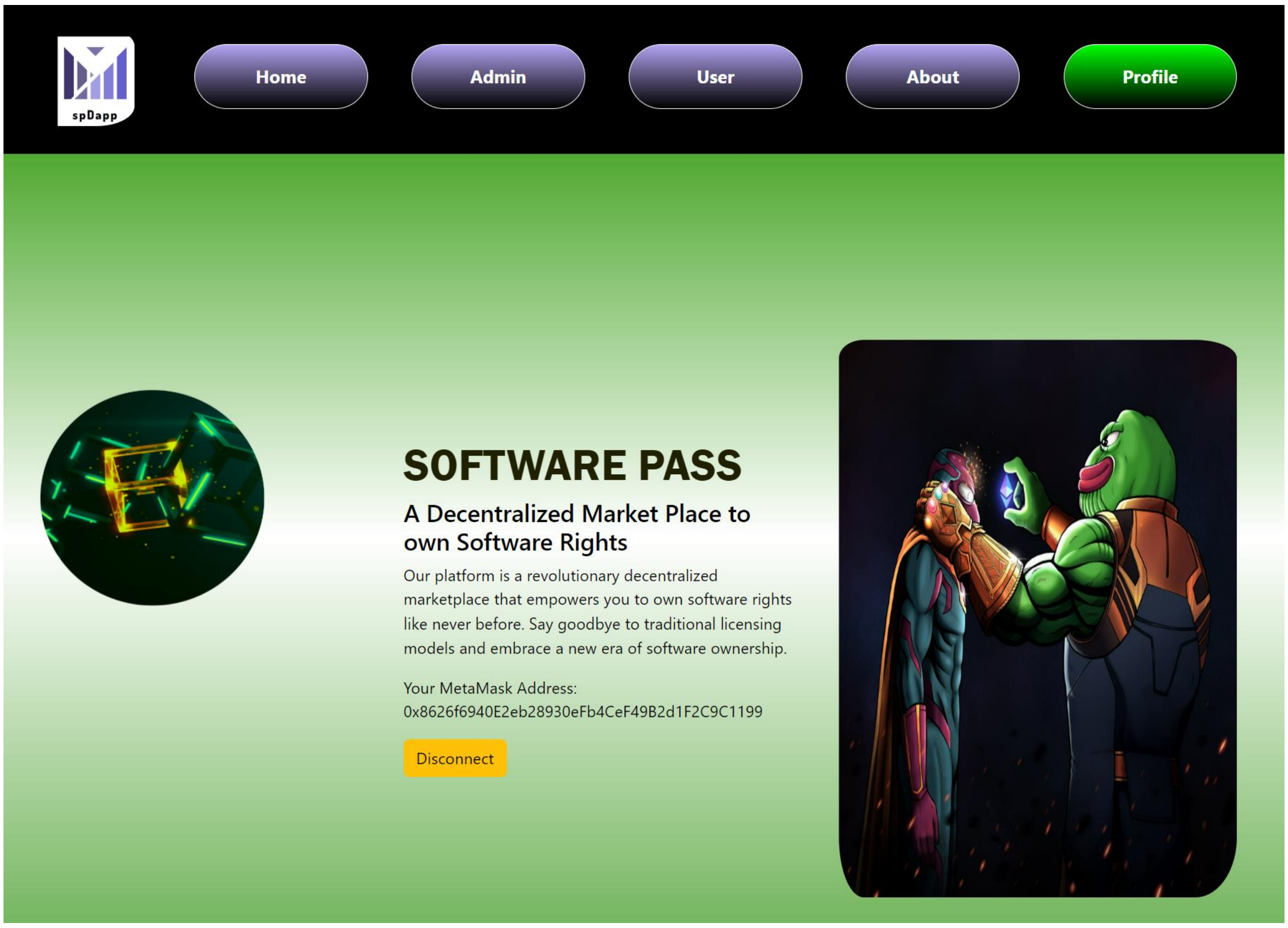
Welcome back!

The decentralized web awaits

Password

Need help? Contact MetaMask support

Once user is authenticated with wallet



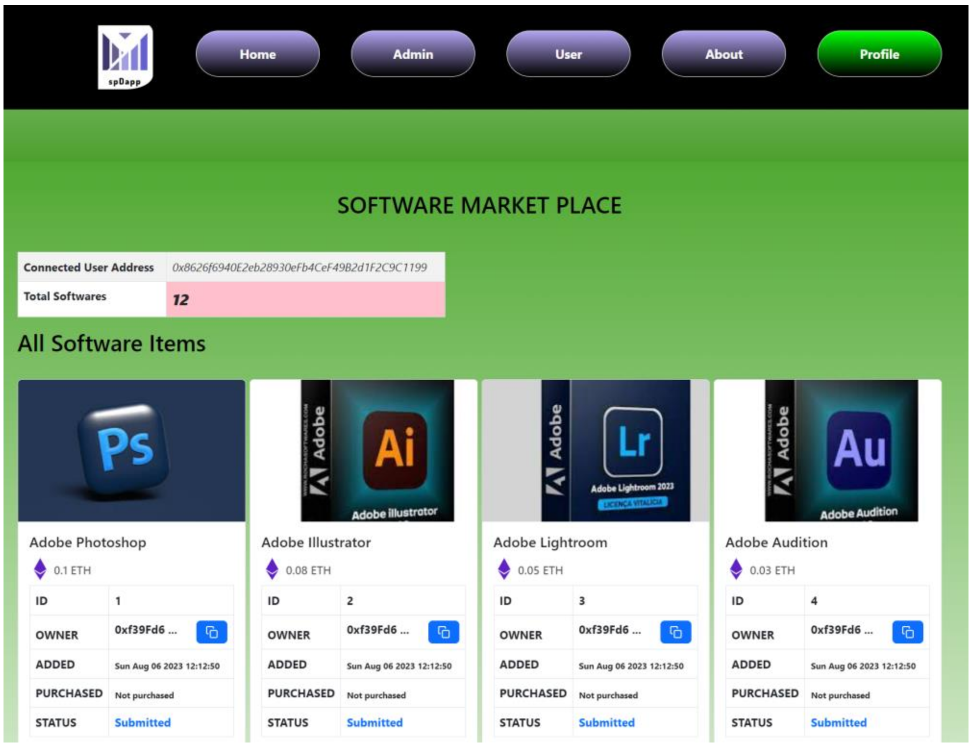
SOFTWARE PASS

A Decentralized Market Place to own Software Rights

Our platform is a revolutionary decentralized marketplace that empowers you to own software rights like never before. Say goodbye to traditional licensing models and embrace a new era of software ownership.

Your MetaMask Address: 0x8626f6940E2eb28930eFb4CeF49B2d1F2C9C1199

Products are available to purchase based on its status



SOFTWARE MARKET PLACE

Connected User Address	0x8626f6940E2eb28930eFb4CeF49B2d1F2C9C1199
Total Softwares	12

All Software Items

Adobe Photoshop
0.1 ETH

ID	1
OWNER	0xf39Fd6 ...
ADDED	Sun Aug 06 2023 12:12:50
PURCHASED	Not purchased
STATUS	Submitted

Adobe Illustrator
0.08 ETH

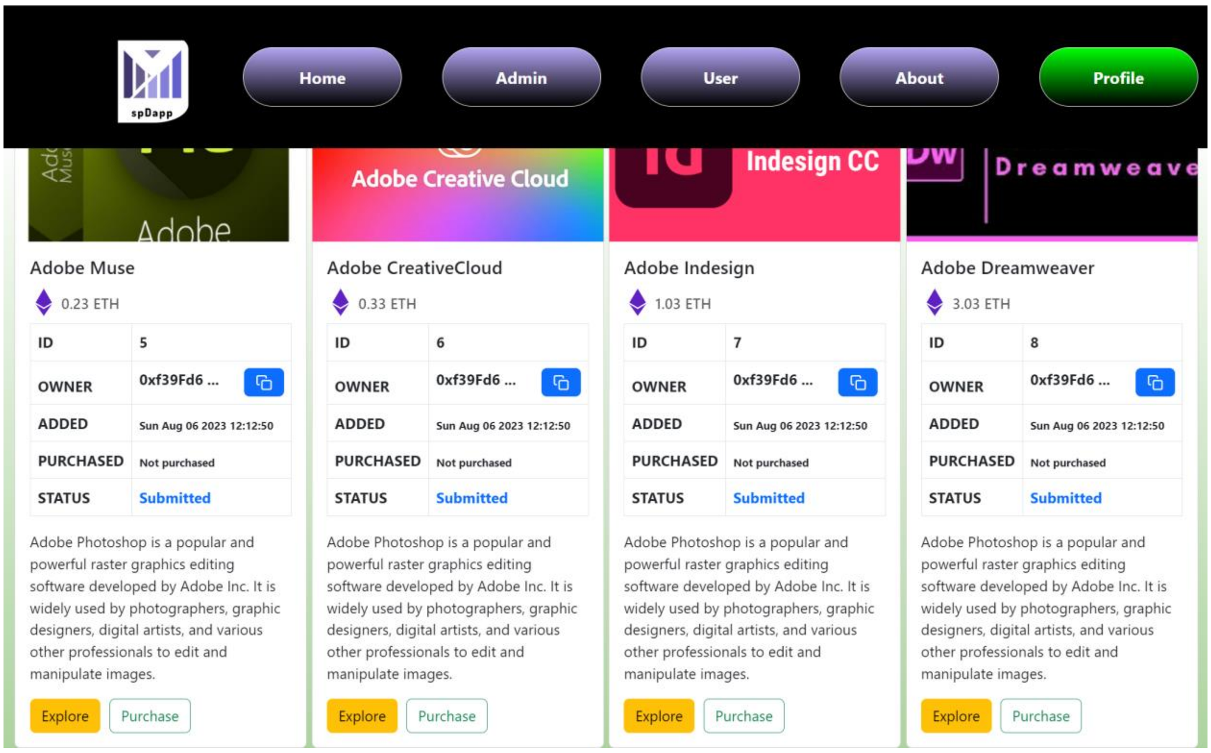
ID	2
OWNER	0xf39Fd6 ...
ADDED	Sun Aug 06 2023 12:12:50
PURCHASED	Not purchased
STATUS	Submitted

Adobe Lightroom
0.05 ETH

ID	3
OWNER	0xf39Fd6 ...
ADDED	Sun Aug 06 2023 12:12:50
PURCHASED	Not purchased
STATUS	Submitted

Adobe Audition
0.03 ETH

ID	4
OWNER	0xf39Fd6 ...
ADDED	Sun Aug 06 2023 12:12:50
PURCHASED	Not purchased
STATUS	Submitted



Adobe Muse

0.23 ETH

ID	5
OWNER	0xf39Fd6 ...
ADDED	Sun Aug 06 2023 12:12:50
PURCHASED	Not purchased
STATUS	Submitted

Adobe Photoshop is a popular and powerful raster graphics editing software developed by Adobe Inc. It is widely used by photographers, graphic designers, digital artists, and various other professionals to edit and manipulate images.

Explore Purchase

Adobe CreativeCloud

0.33 ETH

ID	6
OWNER	0xf39Fd6 ...
ADDED	Sun Aug 06 2023 12:12:50
PURCHASED	Not purchased
STATUS	Submitted

Adobe Photoshop is a popular and powerful raster graphics editing software developed by Adobe Inc. It is widely used by photographers, graphic designers, digital artists, and various other professionals to edit and manipulate images.

Explore Purchase

Adobe Indesign

1.03 ETH

ID	7
OWNER	0xf39Fd6 ...
ADDED	Sun Aug 06 2023 12:12:50
PURCHASED	Not purchased
STATUS	Submitted

Adobe Photoshop is a popular and powerful raster graphics editing software developed by Adobe Inc. It is widely used by photographers, graphic designers, digital artists, and various other professionals to edit and manipulate images.

Explore Purchase

Adobe Dreamweaver

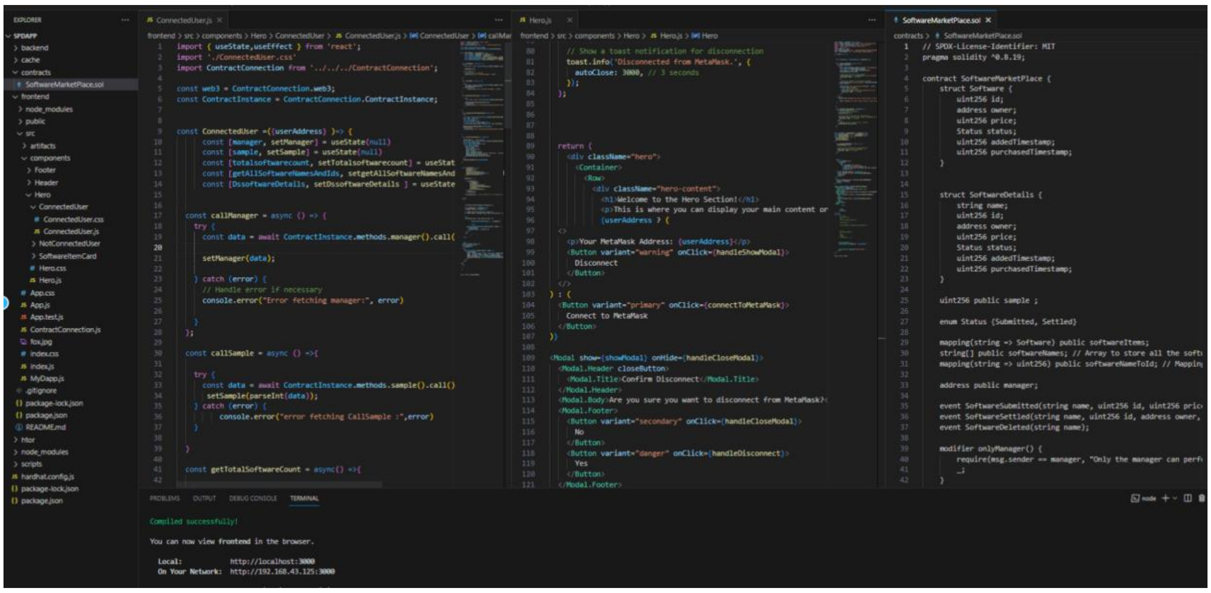
3.03 ETH

ID	8
OWNER	0xf39Fd6 ...
ADDED	Sun Aug 06 2023 12:12:50
PURCHASED	Not purchased
STATUS	Submitted

Adobe Photoshop is a popular and powerful raster graphics editing software developed by Adobe Inc. It is widely used by photographers, graphic designers, digital artists, and various other professionals to edit and manipulate images.

Explore Purchase

Webapp dev environment



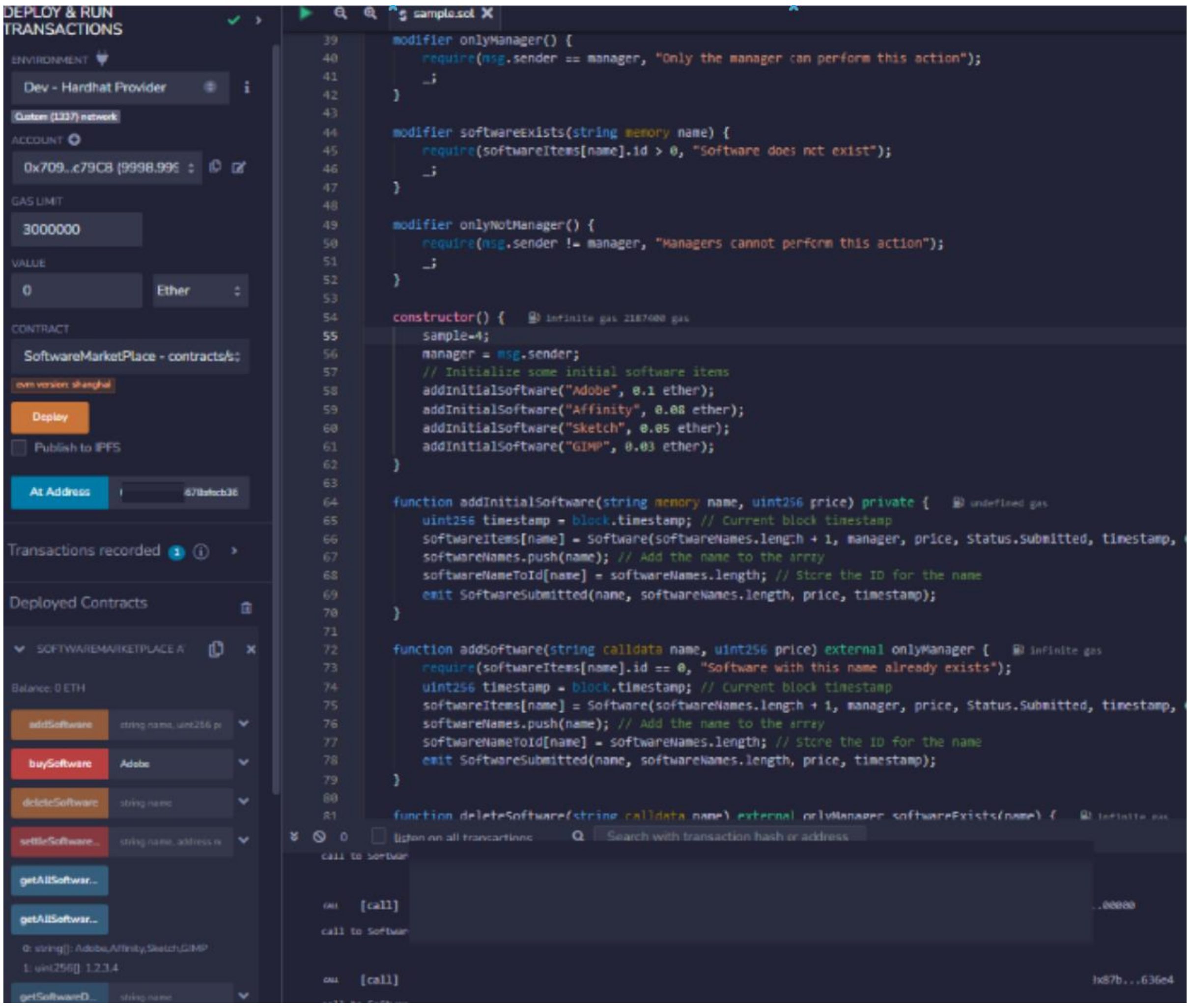
```
import { useState,useEffect } from 'react';
import './ConnectedUser.css'
import ContractConnection from '../../../ContractConnection';

const web3 = ContractConnection.web3;
const ContractInstance = ContractConnection.ContractInstance;


const ConnectedUser =({userAddress} )=> {
    const [manager, setManager] = useState(null)
    const [sample, setSample] = useState(null)
    const [totalsoftwarecount, setTotalsoftwarecount] = useStat
    const [getAllSoftwareNamesAndIds, setgetAllSoftwareNamesAnd
    const [DssoftwareDetails, setDssoftwareDetails ] = useState


  const callManager = async () => {
    try {
      const data = await ContractInstance.methods.manager().call(

      setManager(data);

    } catch (error) {
      // Handle error if necessary
      console.error("Error fetching manager:", error)

    }
  };

  const callSample = async () =>{

    try {
      const data = await ContractInstance.methods.sample().call()
       setSample(parseInt(data));
    } catch (error) {
        console.error("error fetching CallSample :",error)
    }

  }

  const getTotalSoftwareCount = async() =>{

```

```
      // Show a toast notification for disconnection
      toast.info('Disconnected from MetaMask.', {
        autoClose: 3000, // 3 seconds
      });
    };



    return (
      <div className="hero">
        <Container>
          <Row>
            <div className="hero-content">
              <h1>Welcome to the Hero Section!</h1>
              <p>This is where you can display your main content or
              {userAddress ? (
    <>
      <p>Your MetaMask Address: {userAddress}</p>
      <Button variant="warning" onClick={handleShowModal}>
        Disconnect
      </Button>
    </>
  ) : (
    <Button variant="primary" onClick={connectToMetaMask}>
      Connect to MetaMask
    </Button>
  )}

  <Modal show={showModal} onHide={handleCloseModal}>
    <Modal.Header closeButton>
      <Modal.Title>Confirm Disconnect</Modal.Title>
    </Modal.Header>
    <Modal.Body>Are you sure you want to disconnect from MetaMask?<
    <Modal.Footer>
      <Button variant="secondary" onClick={handleCloseModal}>
        No
      </Button>
      <Button variant="danger" onClick={handleDisconnect}>
        Yes
      </Button>
    </Modal.Footer>
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract SoftwareMarketPlace {
    struct Software {
        uint256 id;
        address owner;
        uint256 price;
        Status status;
        uint256 addedTimestamp;
        uint256 purchasedTimestamp;
    }


    struct SoftwareDetails {
        string name;
        uint256 id;
        address owner;
        uint256 price;
        Status status;
        uint256 addedTimestamp;
        uint256 purchasedTimestamp;
    }

    uint256 public sample ;

    enum Status {Submitted, Settled}

    mapping(string => Software) public softwareItems;
    string[] public softwareNames; // Array to store all the softu
    mapping(string => uint256) public softwareNameToId; // Mappin

    address public manager;

    event SoftwareSubmitted(string name, uint256 id, uint256 pric
    event SoftwareSettled(string name, uint256 id, address owner,
    event SoftwareDeleted(string name);

    modifier onlyManager() {
        require(msg.sender == manager, "Only the manager can perf
        _;
    }
```

```
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.43.125:3000
```

Contract development and testing: Remix



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Dev - Hardhat Provider
Custom (1337) network
ACCOUNT: 0x709...c79C8 (9998.999
GAS LIMIT: 3000000
VALUE: 0 Ether
CONTRACT: SoftwareMarketPlace - contracts/s
evm version: shanghai

At Address [redacted] 67bfecb36

Transactions recorded 1

Deployed Contracts
SOFTWAREMARKETPLACE A
Balance: 0 ETH
addSoftware: string name, uint256 pr
buySoftware: Adobe
deleteSoftware: string name
settleSoftware...: string name, address n
getAllSoftwar...
getAllSoftwar...
0: string[]: Adobe,Affinity,Sketch,GIMP
1: uint256[]: 1,2,3,4
getSoftwareD...: string name

```
    modifier onlyManager() {
        require(msg.sender == manager, "Only the manager can perform this action");
        _;
    }

    modifier softwareExists(string memory name) {
        require(softwareItems[name].id > 0, "Software does not exist");
        _;
    }

    modifier onlyNotManager() {
        require(msg.sender != manager, "Managers cannot perform this action");
        _;
    }

    constructor() {    infinite gas 2187600 gas
        sample=4;
        manager = msg.sender;
        // Initialize some initial software items
        addInitialSoftware("Adobe", 0.1 ether);
        addInitialSoftware("Affinity", 0.08 ether);
        addInitialSoftware("Sketch", 0.05 ether);
        addInitialSoftware("GIMP", 0.03 ether);
    }

    function addInitialSoftware(string memory name, uint256 price) private {    undefined gas
        uint256 timestamp = block.timestamp; // Current block timestamp
        softwareItems[name] = Software(softwareNames.length + 1, manager, price, Status.Submitted, timestamp,
        softwareNames.push(name); // Add the name to the array
        softwareNameToId[name] = softwareNames.length; // Store the ID for the name
        emit SoftwareSubmitted(name, softwareNames.length, price, timestamp);
    }

    function addSoftware(string calldata name, uint256 price) external onlyManager {    infinite gas
        require(softwareItems[name].id == 0, "Software with this name already exists");
        uint256 timestamp = block.timestamp; // Current block timestamp
        softwareItems[name] = Software(softwareNames.length + 1, manager, price, Status.Submitted, timestamp,
        softwareNames.push(name); // Add the name to the array
        softwareNameToId[name] = softwareNames.length; // Store the ID for the name
        emit SoftwareSubmitted(name, softwareNames.length, price, timestamp);
    }

    function deleteSoftware(string calldata name) external onlyManager softwareExists(name) {    infinite gas
```

```
listen on all transactions    Search with transaction hash or address
call to Softwar [redacted]
[call] [redacted] ..00000
call to Softwar [redacted]
[call] 0x87b...636e4
```

Solidity code:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract SoftwareMarketPlace {
   struct Software {
      uint256 id;
      address owner;
      uint256 price;
      Status status;
      uint256 addedTimestamp;
      uint256 purchasedTimestamp;
   }



   struct SoftwareDetails {
      string name;
      uint256 id;
      address owner;
      uint256 price;
      Status status;
      uint256 addedTimestamp;
      uint256 purchasedTimestamp;
   }

   uint256 public sample ;

   enum Status {Submitted, Settled}

   mapping(string => Software) public softwareItems;
   string[] public softwareNames; // Array to store all the software names
```

```
mapping(string => uint256) public softwareNameToId; // Mapping to get software ID by name

address public manager;

event SoftwareSubmitted(string name, uint256 id, uint256 price, uint256 addedTimestamp);
event SoftwareSettled(string name, uint256 id, address owner, uint256 purchasedTimestamp);
event SoftwareDeleted(string name);

modifier onlyManager() {
   require(msg.sender == manager, "Only the manager can perform this action");
   _;
}

modifier softwareExists(string memory name) {
   require(softwareItems[name].id > 0, "Software does not exist");
   _;
}

modifier onlyNotManager() {
   require(msg.sender != manager, "Managers cannot perform this action");
   _;
}

constructor() {
   sample=4;
   manager = msg.sender;
   // Initialize some initial software items
   addInitialSoftware("Adobe Photoshop", 0.1 ether);
   addInitialSoftware("Adobe Illustrator", 0.08 ether);
   addInitialSoftware("Adobe Lightroom", 0.05 ether);
   addInitialSoftware("Adobe Audition", 0.03 ether);
```

```
        addInitialSoftware("Adobe Muse", 0.23 ether);
        addInitialSoftware("Adobe CreativeCloud", 0.33 ether);
        addInitialSoftware("Adobe Indesign", 1.03 ether);
        addInitialSoftware("Adobe Dreamweaver", 3.03 ether);
        addInitialSoftware("Adobe XD", 4.03 ether);
        addInitialSoftware("Adobe Premiere", 0.02 ether);
        addInitialSoftware("Adobe Animate", 0.003 ether);
        addInitialSoftware("Adobe Express", 0.23 ether);
    }

    function addInitialSoftware(string memory name, uint256 price) private {
        uint256 timestamp = block.timestamp; // Current block timestamp
        softwareItems[name] = Software(softwareNames.length + 1, manager, price, Status.Submitted,
timestamp, 0);
        softwareNames.push(name); // Add the name to the array
        softwareNameToId[name] = softwareNames.length; // Store the ID for the name
        emit SoftwareSubmitted(name, softwareNames.length, price, timestamp);
    }

    function addSoftware(string calldata name, uint256 price) external onlyManager {
        require(softwareItems[name].id == 0, "Software with this name already exists");
        uint256 timestamp = block.timestamp; // Current block timestamp
        softwareItems[name] = Software(softwareNames.length + 1, manager, price, Status.Submitted,
timestamp, 0);
        softwareNames.push(name); // Add the name to the array
        softwareNameToId[name] = softwareNames.length; // Store the ID for the name
        emit SoftwareSubmitted(name, softwareNames.length, price, timestamp);
    }

   function deleteSoftware(string calldata name) external onlyManager softwareExists(name) {
    uint256 softwareIndexToDelete = softwareNameToId[name] - 1; // Get the index of the software
item to delete
```

```
   require(softwareIndexToDelete < softwareNames.length, "Software item not found");

   // Swap and pop to remove the software item from the array
   softwareNames[softwareIndexToDelete] = softwareNames[softwareNames.length - 1];
   softwareNames.pop();

   // Update the softwareNameToId mapping to reflect the new index of the software item
   softwareNameToId[softwareNames[softwareIndexToDelete]] = softwareIndexToDelete + 1;

   // Delete the software item from the softwareItems mapping
   delete softwareItems[name];

   emit SoftwareDeleted(name);
}



   function settleSoftwareOwnership(string calldata name, address newOwner) external payable
softwareExists(name) onlyNotManager {
      Software storage software = softwareItems[name];
      require(software.status == Status.Submitted, "Software is already settled");
      require(msg.value >= software.price, "Insufficient payment");

      address previousOwner = software.owner;
      software.owner = newOwner;
      software.status = Status.Settled;
      software.purchasedTimestamp = block.timestamp; // Current block timestamp

      emit SoftwareSettled(name, software.id, newOwner, software.purchasedTimestamp);

      // Transfer the payment to the previous owner
```

```
    (bool transferSuccess, ) = previousOwner.call{value: msg.value}("");
    require(transferSuccess, "Transfer to previous owner failed");
}

function getSoftwareStatus(string calldata name) external view returns (Status) {
    return softwareItems[name].status;
}

function getSoftwareDetails(string calldata name)
    external
    view
    returns (
        uint256,
        address,
        uint256,
        Status,
        uint256,
        uint256
    )
{
    Software memory software = softwareItems[name];
    return (
        software.id,
        software.owner,
        software.price,
        software.status,
        software.addedTimestamp,
        software.purchasedTimestamp
    );
}
```

```
    function getAllSoftwareNamesAndIds() external view returns (string[] memory, uint256[] memory)
{
        uint256[] memory softwareIds = new uint256[](softwareNames.length);
        for (uint256 i = 0; i < softwareNames.length; i++) {
            softwareIds[i] = i + 1;
        }
        return (softwareNames, softwareIds);
    }
    function buySoftware(string calldata name) external payable softwareExists(name)
onlyNotManager {
        Software storage software = softwareItems[name];
        require(software.status == Status.Submitted, "Software is already settled");
        require(msg.value >= software.price, "Insufficient payment");


        address previousOwner = software.owner;
        software.owner = msg.sender;
        software.status = Status.Settled;
        software.purchasedTimestamp = block.timestamp; // Current block timestamp
        emit SoftwareSettled(name, software.id, msg.sender, software.purchasedTimestamp);
        // Transfer the payment to the previous owner
        (bool transferSuccess, ) = previousOwner.call{value: msg.value}("");
        require(transferSuccess, "Transfer to previous owner failed");
    }
    function getTotalSoftwareItems() external view returns (uint256) {
        return softwareNames.length;
    }
    function getAllSoftwareItems() external view returns (SoftwareDetails[] memory) {
        SoftwareDetails[] memory allSoftware = new SoftwareDetails[](softwareNames.length);
        for(uint256 i = 0; i < softwareNames.length; i++) {
            Software memory software = softwareItems[softwareNames[i]];
            allSoftware[i] = SoftwareDetails(
                softwareNames[i],
                software.id,
                software.owner,
                software.price,
                software.status,
                software.addedTimestamp,
                software.purchasedTimestamp
            );
        }
        return allSoftware;
    }
}
```