National College of Ireland

National College of Ireland

MSCCLOUD1JAN24I A
MSCCLOUD1JAN24I B
Weighting 50% Project
To Be Completed as Individual
Release: Week of 26th of Feb
Submission: Week of 15th of April

Blockchain Concepts and Technologies
5 Credits

Academic Honesty Declaration

I declare the following to be true for this submission:

- I have completed the task during the designated time window and declare it to be exclusively my own work.
- I have not received, or attempted to receive assistance in preparing this response from any other person during the assessment window.
- I have not provided, or offered to provide, assistance to any other student by any means during the assessment window.
- I have read and understand the National College of Ireland guidelines of Plagiarism.
- This Front-Page declaration is to be included within your submission

Additional Guidelines

- IEEE is not expected for this assessment, standard word or PDF format can be used for submission.
- Font Size 11 is recommended with Word Length specified within.
- Any Online material included within this assessment should be cited and referenced using the Harvard referencing style
- **The Use of any AI supportive tools is forbidden**

Web3 enabled Furniture Retail Dapp

Student id: 23202513

Beanstalk: http://furniture-env.eba-nc7pp6hi.ca-central-1.elasticbeanstalk.com/

Youtube: https://youtu.be/oQQbNDW_Y_Q

Introduction

In the development of a furniture retail Dapp app on Node.js, several key tools were utilized to enable various functionalities within the application. The name of the app of the app is Evok. It enables the users to buy products from the collection of the offerings, log in through their digital wallet(metamask), select their items and then make payments to the Ether on the sepolia test network. The tools were the industrial factor producing the quality of the project by making the user journey the smooth one and also at the same time ensuring that the behind platform operate securely and smoothly.

- Node.js: The runtime environment Node.js was being the primary factor in running the application which is the case especialy when it comes to JavaScript functionality out of web browser. This assisted with the back end development of the furniture selling application, which encompasses server-side logic and other Ethereum blockchain interactions by means of smart contracts.
- MetaMask: we integrated MetaMask into the app for user authentication and connection to the smart contract. It produced a secure and user-friendly way for users to log in via connecting to their accounts and interact with the Ethereum network, specifically the Sepolia Ether test network in this case which enables seamless payments and transactions which provides users with a trustful mode of payment.
- Coingecko API: Through Coin gecko API, there was a provision of real-time EUR to Ethereum conversional rates that were very instrumental. Through this integration, the website allowed users to view prices in their home currency and the platform provided a way to make decisions about what they wanted to purchase furniture or donations.
- Remix IDE: The Remix smart contract editor was most helpful for creating and deploying the smart contract, which was needed for the real estate selling app called "Evok". It fully equipped us with the very situation for the composing, designing and deploying Solidity smart contracts into the Ethereum's blockchain network. We thus developed the contract-based exchanges of addresses such as deposit, withdraw and balance from which we accessed and used to make payments.
- Elastic Beanstalk: An Elastic Beanstalk system delivers a high-performance environment. It is a cloud-deployed service which is successively proven to be the best solution for application hosting. Elastic Beanstalk helped us in the simplification process of the application deployment with options for scalability, monitoring, and management omitting managing the infrastructure completely.

Setup and configuration of tools

The following are the details about the installation and configuration of the tools we used in developing the Evok furniture retail DApp:

1. Node.js and npm

Installation[1]: we downloaded and installed the latest stable version of Node.js from the official website which is v20.12.2. Npm (Node Package Manager) was also included in this installation to manage the dependencies.

Configuration: No additional configuration was needed for Node.js for our development needs. But, we considered using a version control system like Git for managing our code.

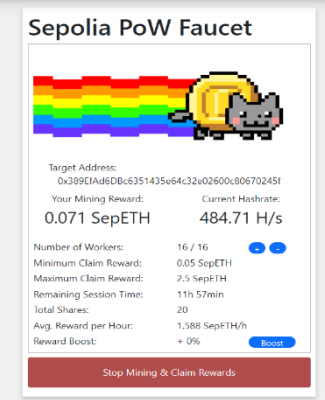
2. MetaMask

Installation: Downloaded and installed the MetaMask extension for our preferred web browser (Firefox in this case) from the official website.

Configuration:

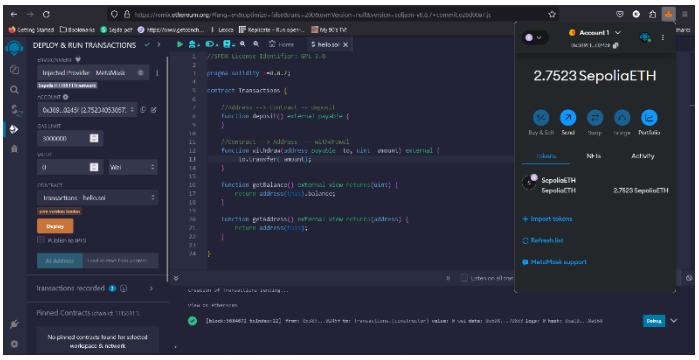
Created a new MetaMask wallet.[2]

Ensured we have some Ether (ETH) in your wallet for testing purposes on the Sepolia test network. We claimed some free Sepolia ETH from Alchemy's Sepolia Faucet and Seplia POW faucet[3]



Within MetaMask settings, connected to the Sepolia test network. This allows interaction with testnet contracts without using real Ether.

3. Remix IDE



Access: Remix is a web-based IDE, so no installation was required. Just opened REMIX[4] in the Firefox browser.

Configuration: No specific configuration was needed. Remix provides a ready-to-use development environment. Just write the code for smart contract, compile and deploy.

Here, we created 4 functions – deposit, withdraw, getBalance, getAddress. Users can then connect to smart contracts.[5]

4. CoinGeckoAPI

Access: we needed a CoinGecko API[6] key. So we created a free account on their official website and obtained an API key from the account dashboard.

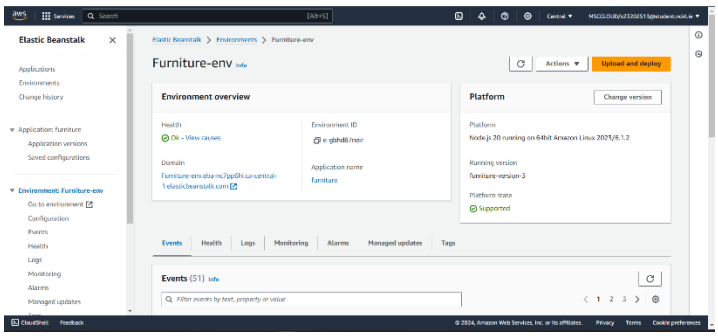
```
async function loginWithMetaMask() {
  try {
      // Fetch exchange rate from API
      const response = await fetch('https://api.coingecko.com/api/v3/simple/price?ids=ethereum&vs_currencies=eur');
      const data = await response.json();
      const exchangeRate = data.ethereum.eur;

      // Calculate total in Ether
      let total = 0;
      const cart = JSON.parse(localStorage.getItem('cart')) || [];
      cart.forEach(item => {
          total += item.price * item.quantity; // Update total based on item price and quantity
      });
      const totalInEther = total / exchangeRate;

      // Redirect to payments page with updated total amount in Ether
      window.location.href = `payment.html?amount=${totalInEther.toFixed(6)}`;
  } catch (error) {
      console.error("Error:", error);
      alert("Failed to fetch exchange rate. Please try again.");
  }
}
```

Integration: We created an api key for the live conversion of EUR to ETH and we integrated it in our chosen Node.js framework. Everytime a user checkouts, The function loginwithMetaMask() from login page fetches the realtime euro to eth conversion rate and redirects to checkout page with amount mentioned in ether. Similarly for donations page, when user donates some amount in euro, the api converts it into ether and opens metamask to deposit the amount.

5. AWS Elastic Beanstalk

Account Setup: Used the student account provided to us by college.

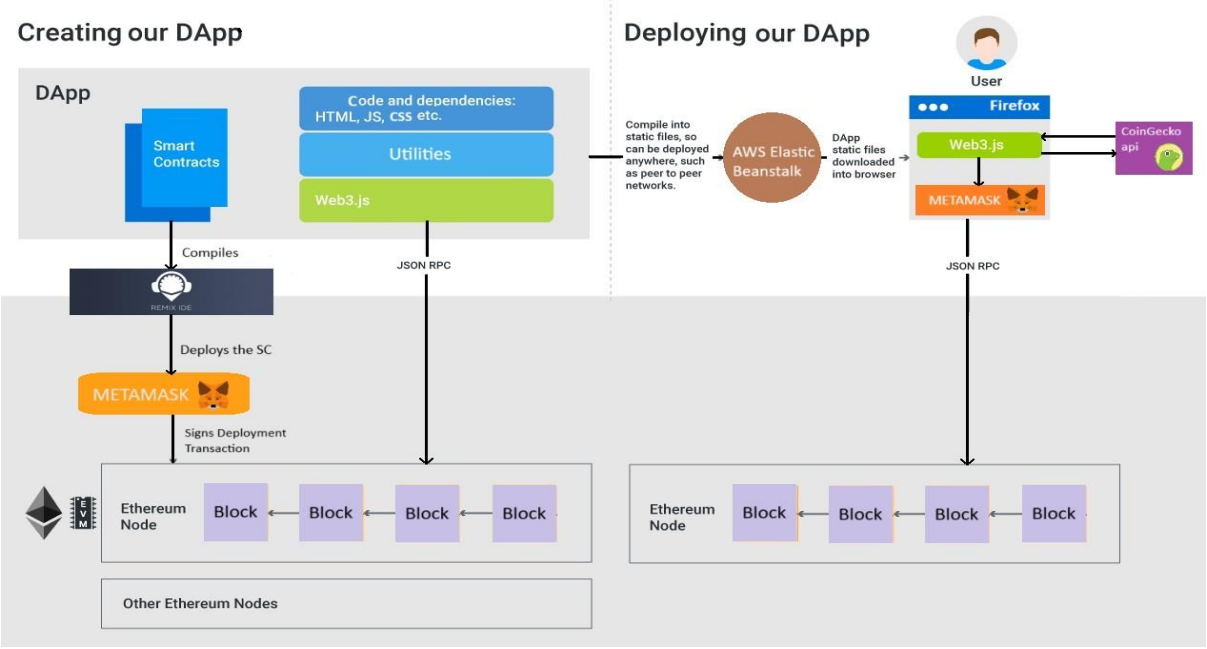
Service Configuration[7]: In the AWS Management Console, navigate to the Elastic Beanstalk service.



Followed the guided setup process to create a new application and environment. Specified our Node.js application code and the configuration settings. After completing all the steps, the application got deployed very easily within minutes.

To summarize this, The Evok app is built using Node.js, Meta Mask, Sepolia Ether Test Network, CoinGecko API, and AWS Elastic Beanstalk – a highly robust solution. When the required environment at the server side was previously created using Node.js as a runtime version for the execution of Java Script and now Metamask provides a secure and user-friendly interface for the same purpose. CoinGecko API allows the exchange of Euros to Ethereum for the transactions. On the other hand, AWS Elastic Beanstalk delivers the deployment of web apps and services management to increased availability and scaling capability. The app has an admin panel for facilitating transactions and withdrawing the ether to the owner's account or any other account. With this, users can transact with minimal hassle and obtain a safe experience.

Architectural Diagram



The architecture diagram describes the process of creating and deploying our decentralized application (DApp) on the Ethereum blockchain. Here's a breakdown of the process:

Developing the DApp:

- The first step involved creating the DApp[8]. This likely involves writing the code using HTML, JavaScript, and CSS.
- Smart Contract Creation: Once the DApp front end is coded, smart contracts are written. Smart contracts are self-executing code on the Ethereum blockchain which facilitates, verifies, and allows the negotiation of a transaction between two people.

These smart contracts are then compiled using Remix IDE and then deployed to ETH blockchain after the user signs the the deployment transaction with their MetaMask account.

Deployment:

Once the compilation of the Dapp is done, the smart contract and static files (HTML, CSS, JS) are uploaded to an Amazon Elastic Beanstalk service. Elastic Beanstalk is a deployment service for AWS that helps us to quickly deploy and scale web applications and APIs.

User Interaction:

Users can access the DApp in their desired browser, such as Firefox. In order to interact with the blockchain, one will either need a platform like Metamask, or possibly a specific wallet. MetaMask is an user interface crypto wallet designed for interaction with the Ethereum blockchain. Finally, the Web3.js (our web3 app) in the browser can connect with the smart contract using json rpc calls. A JSON (JavaScript Object Notation) RPC (Remote Procedure Call) is a protocol used for communication between different systems. The architecture diagram depicts the steps involved in creating a on the Ethereum blockchain. It starts from an overview of creating the DApp, smart contracts and deploying those contracts, and how users can interact with it.

Smart contracts

Smart contracts are autonomous contracts with the terms of the agreement written within code, thus, the contract only gets activated once the predetermined conditions have been met. They function by the blockchain technology, providing encrypted, tracible and automated transactions among participants which do not require an intermediary. This paper targets the utilizations, usage and the implications of smart contracts in different disciplines.

Use and Purpose of Smart Contracts:

Financial Sector: financial industry is able to perform its financial operations, namely loans, transactions and insurance claims through the use of smart contracts. This provision enables smooth automated contract execution which dramatically reduces chances of errors, fraud or in other way from being manually intervened.

Supply Chain Management: Smart contacts that register commodity transfer as well as the terms of agreement between contracting parties and thus taking supply chains' traceability and transparency to another level is made possible by them. They have the advantage of keeping a permanent register of transactions making certain the fulfillment of agreements and they also help in settling the disputes.

State of the Art Examples:

DeFi (Decentralised Finance): The absence of any traditional middlemen makes DeFi platforms use smart contracts to offer a range of financial services incorporate int stable currencies such lending, borrowing, and trading. Uniswap is a kind of the decentralised exchanges and, likewise, there exists one more protocol which is called Compound. Those two protocols are decentralised lending protocols, too.

Non-Fungible Tokens (NFTs): The typical use is applying smart contracts. As a result, they are singular digital assets reflected on a blockchain. This makes it possible for anyone to put assets into Smart Contracts and see where they have been time-stamped, along with who owns them as well. Examples: An NBA Top Shot, CryptoKitties, and OpenSea are a few articulations.

Impact and Insights:

Smart contracts and blockchain technology have a huge role in revolutionizing the creation and management of tokens, NFTs, and applications. According to me, they provide various benefits which includes:

Democratization of finance: The decentralized finance protocols enables the financial services to be at the reach of every human being on the planet without having any restrictions on geography or social status.

Digital ownership: NFTs have made it possible that artisans get the real money from the digital product ownership. Moreover, due to the limiting of the value ownership in the digital economy, collectors can react and make much more sums in the digital markets.

Automated governance: Smart contracts staff the old-school system based on the centralized governance, leading to independent communities where people lead and govern themselves with freedom to make a decision..

But, the rise of smart contracts and blockchain technology have also raised some ethical and social concerns, including:

Regulatory challenges: The decentralization structure of the blockchain and smart contracts create difficulties for the regulators in order to enforce the statutory laws that are related to consumer's protection, investors' rights and compliance that are in place already.

Environmental impact: The conventionally accepted consensus mechanisms in blockchain networks which making use of energy-intensive such as proof-of-work, notwithstanding the fact the still gives causes to worry the Environment sustainability of Blockchain technology.

Security risks: Smart contracts which are coded to act like people cannot bypass flaws and exploits even though they have an inmutable nature. Security best practices and auditing processes should not be neglected anytime and are the ones that help to diminish the risks and to protect users' funds and data.

Simply put, smart contracts are a new phenomenon which could change the way things are conducted even in industries vast and diverse, thanks to their mechanization and transparent nature. Nevertheless, the maximum efficiency of digital technologies is hampered by several obstacles, namely, the cybersecurity and regulatory issues, as well as the constraints related to validation of data and other solutions.

Reference List

[1]. https://www.twilio.com/docs/usage/tutorials/how-to-set-up-your-node-js-and-express-development-environment
[2]. https://www.techopedia.com/how-to/how-to-set-up-a-metamask-wallet
https://www.toptal.com/ethereum/one-click-login-flows-a-metamask-tutorial
[3]. https://sepolia-faucet.pk910.de/
[4]. https://www.c-sharpcorner.com/article/how-to-use-remix-ide-to-create-and-deploy-smart-contracts/
https://medium.com/coinmonks/solidity-tutorial-how-to-use-remix-ide-for-solidity-smart-contract-development-d0d2ce6da051
[5]. https://www.quicknode.com/guides/ethereum-development/smart-contracts/how-to-interact-with-smart-contracts
[6]. https://www.coingecko.com/api/documentation
[7]. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.deploy-existing-version.html
[8]. https://www.freecodecamp.org/news/build-a-beginner-friendly-javascript-application/